SKP Engineering College

Tiruvannamalai – 606611

A Course Material

on

Computer Architecture

By

G.Nanda kumar

Assistant Professor

Computer Science and Engineering Department

Quality Certificate

This is to Certify that the Electronic Study Material

Subject Code : CS6303

Subject Name : Computer Architecture

Year/Sem: III/VI

Being prepared by me and it meets the knowledge requirement of the University curriculum.

Signature of the Author

Name: G.Nanda kumar

Designation: Assistant Professor

This is to certify that the course material being prepared by Mr. G.Nanda kumar is of the adequate quality. He has referred more than five books and one among them is from abroad author.

Signature of HD

Name:

Seal:

Signature of the Principal

Name: Dr.V.Subramania Bharathi

Seal:

CS6303 COMPUTER ARCHITECTURE L T P C 3 0 0 3

OBJECTIVES:

- To make students understand the basic structure and operation of digital computer.
- To understand the hardware-software interface.
- To familiarize the students with arithmetic and logic unit and implementation of fixed point and floating-point arithmetic operations.
- To expose the students to the concept of pipelining.
- To familiarize the students with hierarchical memory system including cache memories and virtual memory.
- To expose the students with different ways of communicating with I/O devices and standard I/O interfaces.

UNIT I OVERVIEW & INSTRUCTIONS 9

Eight ideas – Components of a computer system – Technology – Performance – Power wall – Uniprocessors to multiprocessors; Instructions – operations and operands – representing instructions – Logical operations – control operations – Addressing and addressing modes.

UNIT II ARITHMETIC OPERATIONS 7

ALU - Addition and subtraction – Multiplication – Division – Floating Point operations – Subword parallelism.

UNIT III PROCESSOR AND CONTROL UNIT 11

Basic MIPS implementation – Building datapath – Control Implementation scheme – Pipelining – Pipelined datapath and control – Handling Data hazards & Control hazards – Exceptions.

UNIT IV PARALLELISM 9

Instruction-level-parallelism – Parallel processing challenges – Flynn's classification – Hardware multithreading – Multicore processors

UNIT V MEMORY AND I/O SYSTEMS 9

Memory hierarchy - Memory technologies – Cache basics – Measuring and improving cache performance - Virtual memory, TLBs - Input/output system, programmed I/O, DMA and interrupts, I/O processors.

OUTCOMES: At the end of the course, the student should be able to:

- Design arithmetic and logic unit.

- Design and anlayse pipelined control units
- Evaluate performance of memory systems.
- Understand parallel processing architectures.

TEXT BOOK:

1. David A. Patterson and John L. Hennessey, “Computer organization and design‟, Morgan Kauffman / Elsevier, Fifth edition, 2014.

REFERENCES:

1. V.Carl Hamacher, Zvonko G. Varanesic and Safat G. Zaky, “Computer Organisation“, VI th edition, Mc Graw-Hill Inc, 2012.

2. William Stallings “Computer Organization and Architecture” , Seventh Edition , Pearson Education, 2006.

3. Vincent P. Heuring, Harry F. Jordan, “Computer System Architecture”, Second Edition, Pearson Education, 2005.

4. Govindarajalu, “Computer Architecture and Organization, Design Principles and Applications", first edition, Tata McGraw Hill, New Delhi, 2005.

5. John P. Hayes, “Computer Architecture and Organization”, Third Edition, Tata Mc Graw Hill, 1998.

6. http://nptel.ac.in/.

CONTENTS

S.No	Particulars	Page
1	Unit – I	06
2	Unit – II	39
3	Unit – III	83
4	Unit – IV	120
5	Unit – V	148

Unit - I

OVERVIEW & INSTRUCTIONS

PART-A

1. Define Computer Architecture.

Computer Architecture refers to the attributes of a system visible to a programmer or the attributes that have a direct impact on the logical execution of a program. Examples of architectural attribute Instruction Set The number of bits used to represent various data types (numbers, characters) IO mechanisms & techniques for addressing memory.

2. Define Computer Organization.

Computer organization refers to the operational units and their inter connections that realize the architectural specifications**Example** for Organizational attributes: Hardware details transparent to the programmer such as Control signals,Interface between the computer & Peripherals and the memory technology used.

3. What is difference between Computer Architecture and Computer Organization?

S. no	Computer Architecture	Computer Organization
1.	It refers to the attributes that have a direct impact on the logical execution of the program.	It refers to the operational units and their interconnections that realize the architectural specifications
2.	Architectural attributes includes the Instruction set, data types, no of bits used to represent the data, I/O mechanisms.	Organizational attributes include those h/w details such as control signals, interfaces b/w the computer memory & I/O peripherals

4. What is cache memory? (Nov/Dec-2013)

The small and fast memory is called cache memory. The frequently accessed information are transferred from main memory to cache memory. It gives fast access of data**.**

5. What is the function of control unit?

The Control unit is the main part of the computer that coordinates the entire computer operations. Data transfers between the processor and memory controlled by the control unit through timing signal.

6. What are the registers generally contained in the processor?

MAR – Memory Address Register.
MDR – Memory Data Register.
IR – Instruction Register.

R_0 – Rn – General purpose Register.PC – Program Counter.

7. What do you mean by Memory address register (MAR) and Memory data register(MDR)?

The MAR holds the address of the location to be accessed. The MDR contains the data to be written into or read out of the addressed location.

8. What is Multiprogramming or multi tasking?

The OS manages the concurrent execution of several application programs to make the best possible uses of computer resources. This pattern of concurrent execution is called multiprogramming or multitasking.

9. What is elapsed time of computer system?

The total time required to execute the total program is called elapsed time. It is affected by the speed of the processor, the disk and the printer.

10. What is processor time of a program?

The period during which the processor is active is called processor time of a program. It depends on the hardware involved in the execution of individual machine instructions.

11. Define clock rate?

The clock rate is given by, **R=1/P,**where P is the length of one clock. It can be measure as cycles per second (Hertz).

12. What is meant by clock cycle?

Processor circuit is controlled by a timing signal called a clock. The clock defines regular time intervals, called clock cycle. To execute the machine instruction the processor divides the action to be performed into sequence of basic steps; each step can be completed in one clock cycle.

13. Write down the basic performance equation?(Apr/May-2014)

T=N*S/R

Where T-Processor time
N-Number of machine instructions
S-Number of basic steps needed to execute one machine instruction
R-Clock rate

14. What is meant by addressing mode? Mention most important of them. (**May/June 2013)**

The addressing mode is defined as the different ways in which the location or of an operand is specified in an instruction.

The different types of addressing modes are:

Immediate addressing mode
Register addressing mode
Direct or absolute addressing mode
Indirect addressing mode
indexed addressing mode
Relative addressing mode

Auto increment,
Auto decrement

15. Define Register addressing mode with an example.

In register addressing mode, the operand is the contents of a processor register. The name (address) of the register is given in the instruction.**Effective address (EA) = Ri, Where Ri is a processor register.**

16. Define absolute addressing mode with an example.

In absolute addressing mode, the operand is in a memory location. The addresses of this location are given explicitly in the instruction. This is also called as direct addressing mode. **EA = Loc Where loc is the memory address.**

17. What is relative addressing mode? When is it used? (May/ June 2012) (Nov/Dec-14)

The effective address is determined by the index mode using program counter in place of the general purpose registersThis address is commonly used to specify the target address in branch instruction**Example: JNZ BACK.**This instruction causes program executive to go to the branch target location identified by the name BACK, if the branch condition is satisfied.

18. What is indirect addressing mode?

The Effective address of the operand is the contents of a register or memory location whose address appears in the instruction **EA = [Ri] or EA = [Loc]**

19. What is indexed addressing mode?

The Effective address of the operand is generated by adding a constant value to the register**. EA = X + [Ri].**

20. Define auto increment mode of addressing?

The Effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next item in the list. **EA = (Ri) +**

21. Define auto decrement mode of addressing?

The contents of a register specified in the instruction are first automatically decremented and are then used as the effective address of the operand. **EA = - (Ri)**

22. List the basic instruction types?

The various instruction types are,
Three address instructions
Two-address instructions
Single-address instructions
Zero-address instructions

23. Name five parts of the computer components.

There are five parts in computer components
Input Memory,Arithmetic and logic,Output andControl units.

24. What is register?

In a computer, a register is one of a small set of data holding places that are part of a computer processor . A register may hold a computer instruction , a storage address, or any kind of data (such as a bit sequence or individual characters). Some instructions specify registers as part of the instruction.For example, an instruction may specify that the contents of two defined registers be added together and then placed in a specified register.

25. List the phases, which are included in the each instruction cycle?(or) . What are major steps to process an instruction.

Fetch: fetches instruction from main memory (M)**Decode:** decodes the instruction's opcode**Load:** loads (read) from Memory any operands needed unless they are already in CPU Registers **Execute**: Executes the instruction via a register-to-register operation using an appropriate functional unit of the CPU such as a fixed–point adder.**Store:** Stores (write) the results in Memory unless they are to be retained in CPU register.

26. What are the speedup techniques available to increase the performance of a computer?

Cache: It is a fast accessible memory often placed on the same chip as the CPU. It is used to reduce the average time required to access an instruction or data to a single clock cycle.
Pipelining: Allows the processing of several instructions to be partially overlapped.
Super scalar: Allows processing of several instructions in parallel (full overlapping)

27. What are Timing signals?

Timing signals are signals that determine when a given action is to take place. Data transfers between the processor and the memory are also controlled by the control unit through timing signals.

28. Distinguish between auto increment and auto decrement addressing mode. (April/May 2010)

Auto increment	**Auto decrement**
1) The effective address of the operand is the contents of the register specified in the instruction. After accessing the operand, the contents of the register are incremented to address the next location.	**1)** The contents of a register specified in the instruction are decremented and then are used as effective address to access a memory location.
2) Auto increment is symbolically represented as $(R_i)+$.	**2)** Auto decrement mode is symbolically represented as $-(R_i)$.
3) Example: move(R_2), R_0+	**3) Example:** Move R_1, $-(R_0)$

29. What is an opcode? How many bits are needed to specify 32 distinct operations? (April/May 2011)

An **opcode** is the first byte of an instruction in machine language which tells the hardware what operation needs to be performed with this instruction. Every processor/controller has its own set of opcodes defined in its architecture. Opcode is the operation to be performed on data. An opcode is followed by data like address, values etc if needed. 5 bits are needed to specify 32 distinct operations.

30. Define word length.(Nov/Dec 2011)

In computer architecture, a **word** is a unit of data of a defined bit length that can be addressed and moved between storage and the computer processor.The number of bits in the word is called **word length.**

31. Suppose you wish to run a program P with 8.5 * 109 instructions on a 5 Ghz machine with CPI of 0.8. What is the expected CPU time? (Nov/Dec 2010)

Percentage of elapsed time=(User CPU Time + System CPU Time)/Elapsed Time
Expected CPU time =0.8-0.2*8.5*109=1.36.

32. Mention the registers used for communications between processor and main memory. (May/June 2010)

MAR(Memory Address Register): The **Memory Address Register (MAR)** is a CPU register that either stores the memory address from which data will be fetched to the CPU or the address to which data will be sent and stored.

MDR (Memory Data Register): It is the register of a computer's control unit that contains the data to be stored in the computer storage (e.g. RAM), or the data after a fetch from the computer storage. It acts like a buffer and holds anything that is copied from the memory ready for the processor to use it.

33. What is SPEC? Specify the formula for SPEC rating. (May/ June 2012)(Apr/May-2014)

SPEC is a nonprofit consortium of 22 major computer vendors whose common goals are “to provide the industry with a realistic yardstick to measure the performance of advanced computer systems” and to educate consumers about the performance of vendors’ products.
The formula for SPEC rating is as follows:

SPEC rating (ratio) = TR / TC;

where,

TR = Running time of the Reference Computer;
TC = Running time of the Computer under test;

.

34. Give an example each of zero-address, one-address, two-address and three-address instructions. (May/June 2006)

Zero address- push (Push the value as top of stock)
One address- INC CL (If carry set, increment CL by one)
Two address- Add A,B (A← A+B)
Three address- Add A,B,C (A← B+C)

35. Which data structures can be best supported using (a) indirect addressing mode (b) indexed addressing mode? (May/June 2006)

(a) Indirect addressing mode – Pointer data structure
(b) Indexed addressing mode- Array data structure

36. What are the four basic types of operations that need to be supported by an instructor set? (May/June 2006)
Data transfer between memory and the processor register.
Arithmetic and logic operations on Data
Program sequencing and control
I/O transfer.

37. A memory byte location contains the pattern 00101100. What does this pattern represent when interpreted as a number? What does it represent as an ASCII Code? (Nov/Dec 2007)
Interpreted number is 44.
ASCII code is NULL/idle.

38. What is Big-Endian and Little-Endian representations.(Nov/Dec-2014)
The big-endian is used when lower byte addresses are used for the more significant bytes(The leftmost bytes) of the word.**The little-endian** is used for the opposite ordering, where the lower byte addresses are used for the less significant bytes (the rightmost bytes) of the word.

39. What is the use of Instruction register?
It holds the instructions that are currently being executed.

40. State the basic performance equation of a computer?(Apr/May-2014)
T= (NxS)/R
Where,
N- Number of instructions
S- Average numbers of steps needed to execute one instruction.
R- Clock rate.

41. How to measure the performance of the system?
Response time
Throughput.

42. **What is register indirect addressing mode? When is it used?(Nov/Dec-2013)**
The effective address of the operand is the contents of a register or memory location whose address appears in the instruction. Ex: **Add (R2),R0**

43.State Amdahl's Law.(NOV/DEC-2014)
Amdahl's Law is used to measure the performance of the multiprocessor:
Parallel- All processors fully used, which is enchanced mode. Serial- Only one Speedup in enchanced mode is the time spent in parallel mode.Subtitute the speedup value in equation(1)

44.List the eight great ideas invented by computer architects.(NOV/DEC-2015)
Design for *Moore's Law*
Use *abstraction* to simplify design

Make the *common case fast*
Performance *via parallelism*
Performance *via pipelining*
Performance *via prediction*
Hierarchy of memories
Dependability via redundancy

45.**Distinguish pipelining from parallelism. (Nov/Dec-2015)**
Parallelism means parallel computing more than one processors are running in parallel. There may be some dedicated hardware running in parallel for doing the specific task. parallelism increases the performance . **The pipelining** is an implementation technique in which multiple instructions are overlapped and also it increase the performance of the system.In pipelining there are different hazards like data hazards, control hazards etc.

PART-B

1.Explain about Eight ideas.(or) What are the great eight ideas to improve the performance of the system?

Eight ideas

Eight great ideas that computer architectures have been invented in the last 60 years of computer design.**Over View**

There are eight ideas great ideas

Design for *Moore's Law*
Use *abstraction* to simplify design
Make the *common case fast*
Performance *via parallelism*
Performance *via pipelining*
Performance *via prediction*
Hierarchy of memories
Dependability via redundancy

Design for Moore's Law:

The one constant for computer designers is rapid change, which is driven largely by ***Moore's Law.*** It states that integrated circuit resources double every 18-24 months.We use an "**Up and to the right** " Moores Law graph to represent designing for change. **Use abstraction to simplify design:**It use **abstractions** to represent the design at different levels of representation**; lower-level details are hidden to offer a simpler model at higher levels.** We'll use the abstract **painting icon** to represent this second great idea. **Make the common case fast:**Making the **common case fast** will tend to enhance performance better than optimizing **the rare case.** The common case is often simpler than the rare case and hence is often easier to enhance.Ex:sports car as the icon for making the common case fast**Performance via parallelism**The instruction are executed at parallel . It increase the speed. so the performance will be improved We use multiple jet engines of a plan as our icon **for parallel performance. Performance via pipelining:**Instructions are overlapped during the execution of the program. It increase the speed so the performance will be improved **Performance via prediction:**The correct rediction will give accurate result, but the misprediction gives wrong result.We use the fortune – tellers crystal ball as our prediction icon **Hierarchy of memories:**Programmers want memory to be **fast , large,**

and cheap , as memory speed often shapes performance , capacity limits the size of problems that can be solved , and the cost of memory today is often the majority of computer cost .We use a layered triangle icon to represent the memory hierarchy.The shape indicates speed , cost and sizeThe closer to the top, the faster and more expensive per bit the memoryThe wider the base of the layer, the bigger the memory.**Dependability via redundancy**Computers not only need to be fast; they need to be dependable. Since any physical device can fail , we make systems **dependable** by including redundant components that can take over when a failure occurs and to help detect failures.

2.Give detail description about Components of a computer system (Nov/Dec-2014)
Components of a computer system
Digital computer systems consist of three distinct units. These units are as follows:
OVER VIEW:
Input unit
Central Processing unit
Arithmetic and logic unit:
Memory unit
Output unit These units are interconnected by electrical cables to permit communication between them. This allows the computer to function as a system.
Input Unit (IU):
A computer must receive both data andprogram statementstofunction properly and able to solve problems. The method of feeding data and programs to a computer is accomplished by an input device.
Computer input devices read data from a source, such as magnetic disks, and translate that data into electronic impulses for transfer into the CPU. Some typical **input devices are a keyboard, a mouse, or a scanner.**
Central Processing Unit (CPU):
The brain of a computer system is the CPU.A central control section and work areas are required to perform calculations or manipulate data.
The CPU is the computing center of the system. It consists of a control section, an arithmetic-logic section, and an internal storage section (main memory). Each section within the CPU serves a specific function. The system unit also includes circuit boards, memory chips, ports and other components. A microcomputer system cabinet will also house disk drives, hard disks, etc., but these are considered separate from the CPU.
Control Unit (CU):
It is the part of a CPU or other device that directs its operation. The control unit tells the rest of the computer system how to carry out a program's instructions.
It directs the movement of electronic signals between memories—which temporarily holds data, instructions and processed information—and the ALU.
It also directs these control signals between the CPU and input/output devices. The control unit is the circuitry that controls the flow of information through the processor, and coordinates the activities of the other units within it.
In a way, it is the "brain", as it controls what happens inside the processor, which in turn controls the rest of the PC.
Arithmetic And Logic Unit(ALU):

- Unit usually called the ALU is a digital circuit that performs two types of operations— arithmetic and logical.

- Arithmetic operations are the fundamental mathematical operations consisting of addition, subtraction, multiplication and division. Logical operations consist of comparisons. That is, two pieces of data are compared to see whether one is equal to, less than, or greater than the other.

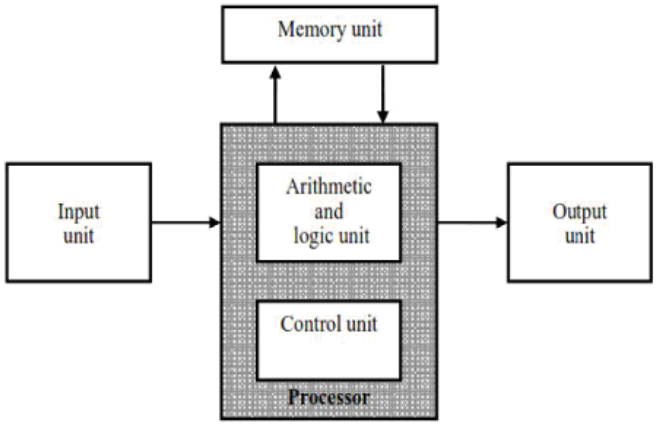


Figure 1.1. Functional Unit

Memory Unit

The function of the memory unit is to store programs and data. There are two classes of storage, called **primary and secondary**. *Primary storage* is a fast memory that operates at electronic speeds.Programs must be stored in the memory while they are being executed. The memory contains a large number of semiconductor storage cells, each capable of storing one bit of information.These cells are rarely read or written as individual cells but instead are processed in groups of **fixed size called *words*.** The number of bits in each word is often referred to as the word length of the computer. Typical **word lengths range from 16 to 64 bits.** The capacity of the memory is one factor that characterizes the size of a computer.Programs must reside in the memory during execution. Instructions and data can be written into the memory or read out under the controller of the processor. It is essential to be able to access any word location in the memory as quickly as possible. Memory in which any location can be reached in a short and fixed amount of time after specifying its address is **called random-access Memory (RAM).** The time required to access one word is called the Memory access time. This time is fixed, independent of the location of the word being accessed. It typically ranges from a few nanoseconds (ns) to about 100 ns for modem RAM units.Thus additional, cheaper, ***secondary storage*** is used when large amounts of data and many programs have to be stored, particularly for information that is access infrequently. A wide selection of secondary storage devices is available, including **magnetic disks and tapes and optical disks**

Output Unit:

Its function is to send processed results to the outside world.

The most familiar example of such a device is a printer. Printers employ mechanical impact heads, inkjet streams, or photocopying techniques, as in laser printers, to perform the printing. It produces printers capable of printing as **many as 10,000** lines per minute.

Some units, such as graphic displays, provide both an output function and an input function. The dual role of input and output of such units are referred with single name as I/O unit in many cases**. Speakers, Headphones and projectors are some of the output devices.**

Storage devices such as hard disk, floppy disk, flash drives are also used for input as well as output.

3.Briefly explain about Technology of computer generations.

Technology

The Computer generations means step by step growth in the technology. There are five generations of computer. They are given below.

Five types of Generation:

Over View

First Generation
Second Generation
Third Generation
Fourth Generation
Fifth Generation

First Generation Computers:

Year *: 1942 – 1955*
Component used *: Vacuum Tube*
Language used *: Machine Level Language*

The first generation computers used vacuum tube circuitry as such these were quite large. It was one of the high speed electronic switching devices available at that time.The generation computers used machine language (0, 1).It is difficult language and it required lot of time to write programs. No operating system is required. The first generation computers used the method of "*Stored Program*" concept.

Advantages

These computers were the fastest calculating device of their time.
These computers could perform computations in milli seconds.

Disadvantages

It occupies large space.
High heat production
High power consumption

Eg: EDSAC, EDVAC

Second Generation Computers:

Year : *1955 – 1964*
Component Used *: Transistors*
Language used *: Assembly Level Language*

The Second generation computers used Transistors.
The storage capacity is higher than the first generation computers.
Batch operating system rules the second generation computers.

Advantages

Small in size
Better speed
• Less hardware failure

Disadvantages

High cost, Frequent maintenance is required.**Eg: UNIVAC-1180, IBM-1620**

Third Generation Computers:

Year *: 1964 – 1975*
Component Used *: Integrated Circuits (IC)*
Language used *: High Level Language*

The third generation computers used Integrated Circuits(IC).The ICs were made of Silicon. The ICs provided vast internal storage and increased the operational efficiency.during the period, the mini computers were developed. Time sharing concept also introduced in this generation.

Advantages:

Easily portable
Less H/W failure
General purpose computer
Less heat generated.

Disadvantages:

Highly sophisticated Technology required for the manufacture of IC chips. Need Air conditioning.**Eg: IBM-360 series**

Fourth Generation Computers:

Year *: 1975 – 1989*
Component used *: Microprocessor (MP)*
Language used *: 4GL*

A micro processor produces a Micro computer.
Microcomputer is a stand alone and easy to use device.
Microcomputers are relatively inexpensive.
The fourth generation language is introduced which were easy to learn and understand.

Advantages

These computers had large and faster primary and secondary storage.
Object oriented languages are supported & these are general purpose computers.

Disadvantages

Highly sophisticated technology required for the manufacture of LSI chips.

Eg: Intel-400L

Fifth Generation Computers

Year *: 1989 – at present*
Concept used *: Artificial Intelligent (AI)*• Computers are machines that perform only when instructed. But they are unable to act or think on their own.• Recent researches are aimed at designing the logic for the thinking computer (AI) for building expert systems and knowledge based systems.• This computer is using magnetic bubble memories and other recent developments are on the way. These computers will be based on advances in silicon technology.

Advantages:

The super computers fall under this generation. Supports Parallel programming

Three Concepts

Mega Chips.
Parallel Processing.
Artificial Intelligence (AI).

4.Explain about Performance of a computer system. Or State the CPU performance equation and discuss the factors that affect performance. (Nov/Dec-2014)

Performance

For best performance, it is necessary to design the compiler, machine instruction set and hardware in a co-ordinate way. **Over viewPerformance CPU Performance and Its FactorsInstruction PerformanceSPEC CPU benchmark: Implication of Amdahl's Law Performance EvalutionResponse time** Also called **execution time.** The total time required for the computer to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, CPU execution time, and so on.
Throughput Also called **bandwidth**. Another measure of performance, it is the number of tasks completed per unit time.**Elapsed Time** The total time required to execute the program is called the elapsed time. It Depends on all the units in computer system.
Processor Time
The period in which the processor is active is called the processor time. It Depends on hardware involved in the execution of the instruction. To maximize performance, we want to minimize response time or execution time for some task. Thus, we can relate performance and execution time for a computer X:This means that for two computers X and Y, if the performance of X is greater than the performance of Y, we have

$$\text{Performance}_X > \text{Performance}_Y$$

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{Execution time}_Y}$$

$$\text{Execution time}_Y > \text{Execution time}_X$$

That is, the execution time on Y is longer than that on X, if X is faster than Y.To relate the performance of two different computers quantitatively. We will use the phrase "X is *n* times faster than Y"—or equivalently "X is *n* times as fast as Y"—to mean

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

If X is *n* times faster than Y, then the execution time on Y is *n* times longer than it is on X:

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

Measuring Performance:
Time is the measure of computer performance:
i.e) if the computer that performs the same amount of work in the least time is the fastest. Program ***execution time*** is measured in seconds per program. However, time can be defined in different ways, depending on what we count. The most straightforward definition of time is called ***wall clock time*****,** ***response time*****, or** ***elapsed time*****.** These terms mean the total time to complete a task, including disk accesses, memory accesses, input/output (I/O) activities, operating system overhead etc.,**Wall –clock time -** how long it takes (typically, time in seconds) for our program to execute, from the time its is invoked to the time it completes.**CPU Time- CPU execution time** Also called **CPU time**. The actual time the CPU spends computing for a specific task. **System CPU time** - The CPU time spent in the operating system performing tasks on behalf of the program.**I/O Time –** time spend reading and writing data from/ to memory.**User CPU time -** The CPU time spent in a program itself.**clock cycle** Also called **tick**, **clock tick**, **clock** time for one clock period, usually of the processor clock, which runs at a constant rate. **clock period** The length of each clock cycle.**CPU Performance and Its Factors:**Users and designers often examine performance using different metrics. A simple formula relates the most basic metrics (**clock cycles and clock cycle time**) to CPU time:

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

Alternatively, because clock rate and clock cycle time are inverses,

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

This formula makes it clear that the hardware designer can improve performance by reducing the number of clock cycles required for a program or the length of the clock cycle.

Instruction Performance: The performance equations above did not include any reference to the number of instructions needed for the program. One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction. Therefore, the number of clock cycles required for a program can be written as

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \text{Average clock cycles per instruction}$$

The term **clock cycles per instruction**, which is the average number of clock cycles each instruction takes to execute, is often abbreviated as **CPI.**

The Classic CPU Performance Equation

The basic performance equation in terms of **instruction count** (the number of instructions executed by the program), CPI, and clock cycle time:

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

or, since the clock rate is the inverse of clock cycle time:

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

The CPI values can be computed by

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

SPEC CPU benchmark:

The Performance Measure is the time it takes a computer to execute a given benchmark.

A non-profit organization called SPEC(System Performance Evaluation Corporation) selects and publishes representative application program.

$$\text{SPEC rating} = \frac{\text{Running time on reference computer}}{\text{Running time on computer under test}}$$

Implication of Amdahl's Law Amdahl's Law is used to measure the performance of the multiprocessor:

$$\text{speedup} = \frac{\text{Fraction enchanced}}{}$$

Assume that the program operates in two modes: Parallel- All processors fully used, which is enchanced mode. Serial- Only one processor in use. Speedup in enchanced mode is the time spent in parallel mode.

5.Explain about Power Wall.

The Power Wall

Figure 1.3 shows the increase in clock rate and power of eight generations of Intel microprocessors over 25 years. Both clock rate and power increased rapidly for decades, and then flattened off recently.The reason they grew together is that they are correlated, and the reason for their recent slowing is that we have run into the practical power limit for cooling commodity microprocessors.

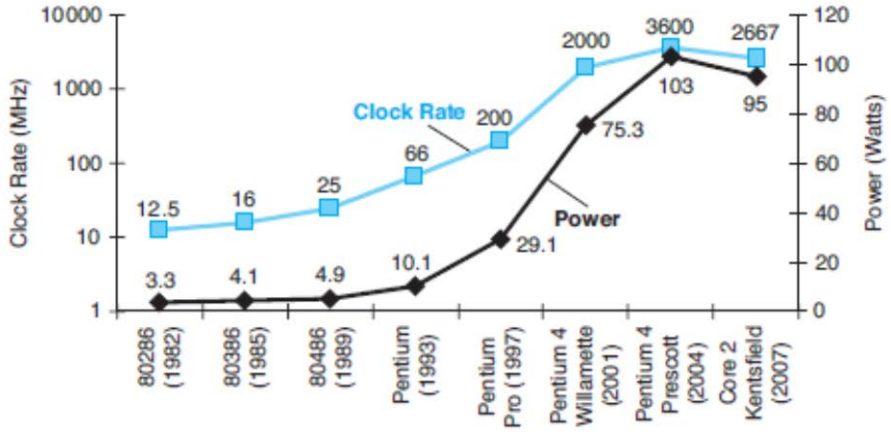


FIGURE 1.3 .Clock rate and Power for Intel x86 microprocessors over eight generations and 25 years.

The Pentium 4 made a dramatic jump in clock rate and power but less so in performance.The Prescott thermal problems led to the abandonment of the Pentium 4 line. The Core 2 line reverts to a simpler pipeline with lower clock rates and multiple processors per chip.The dominant technology for integrated circuits is called CMOS (complementary metal oxide semiconductor). For CMOS, the primary source of power dissipation is so-**called dynamic power**—that is, power that is consumed during switching. The dynamic power dissipation depends on the capacitive loading of each transistor, the voltage applied, and the frequency that the transistor is switched: **Power = Capacitive load × Voltage2 × Frequency switched**Frequency switched is a function of the clock rate. The capacitive load per Transistor is a function of both the number of transistors connected to an output **(called the *fanout*)** and the technology, which determines the capacitance of both wires and transistors.

6. Explain about Uniprocessors to Multiprocessors

Uniprocessors to Multiprocessors:

This chart plots performance relative to the VAX 11/780 as measured by the SPECint benchmarks (see Prior to the mid-1980s, processor performance growth was largely technology driven and averaged about 25% per year.The increase in growth to about

52% since then is attributable to more advanced architectural and organizational ideas. By 2002, this growth led to a difference in performance of about a factor of seven.

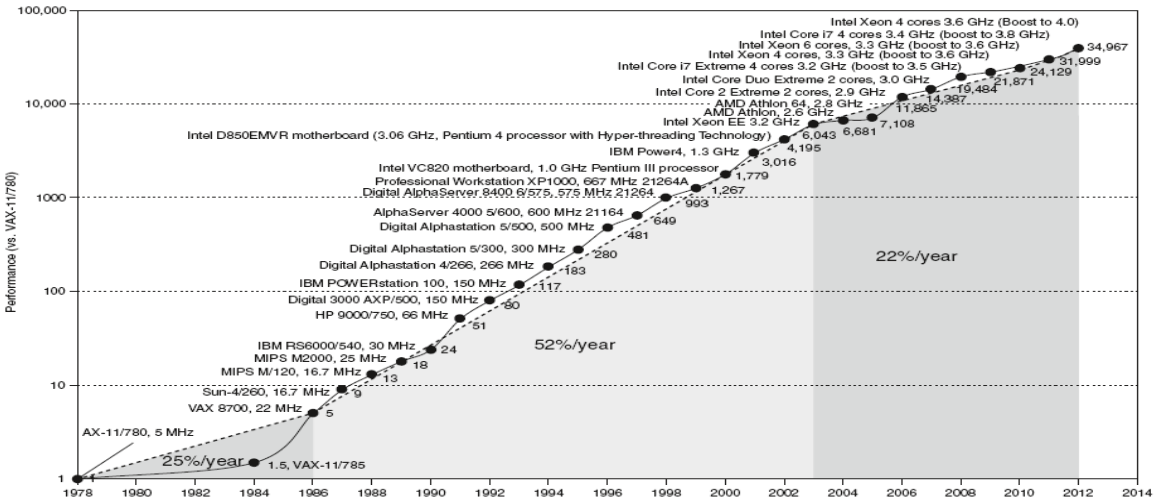


FIGURE 1.4 Growth in processor performance since the mid-1980s.

Performance for floating-pointoriented calculations has increased even faster. Since 2002, the limits of power, available instruction-level parallelism, and long memory latency have slowed uniprocessor performance recently, to about 20% per year.

Product	AMD Opteron X4 (Barcelona)	Intel Nehalem	IBM Power 6	Sun Ultra SPARC T2 (Niagara 2)
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	~ 2.5 GHz ?	4.7 GHz	1.4 GHz
Microprocessor power	120 W	~ 100 W ?	~ 100 W ?	94 W

FIGURE 1.5. Number of cores per chip, clock rate, and power for 2008 multicore microprocessors

Why has it been so hard for programmers to write explicitly parallel programs?

The first reason is that parallel programming is by definition performance programming, which increases the difficulty of programming. Not only does the program need to be correct, solve an important problem, and provide a useful interface to the people or other programs that invoke it, the program must also be fast. Otherwise, if you don't need performance, just write a sequential program.The **second reason** is that fast for parallel hardware means that the programmer must divide an application so that each processor has roughly the same amount to do at the same time, and that the overhead of scheduling and coordination doesn't fritter away the potential performance benefits of parallelism.As an analogy, suppose the task was to write a newspaper story. Eight reporters working on the same story could potentially write a story eight times faster.To achieve this increased speed, one would need to break up the task so that each reporter had something to do at the same time. Thus, we must ***schedule*** the **subtasks**.If anything went wrong and just one reporter took longer than the seven others did, then the benefits of having eight writers would be diminished. Thus, we must *balance the load* evenly to get the desired speedup

Thus, care must be taken to ***reduce communication and synchronization overhead***. For both this analogy and parallel programming, the challenges include scheduling, load balancing, time for synchronization, and overhead for communication between the parties.

7. Explain about Instructions & Operation of the computer hardware.

Operation of the computer hardware:

Every computer must be able to perform arithmeticThe MIPS assembly language notation**add a, b, c**Instructs a computer to add the two variables b and c and to put their sum in a This notation is rigid in that each MIPS arithmetic instruction performs only one operation and must always have exactly three variables. The following code shows an equivalent MIPS code **:**

ADD $s1,$s2, $s3

The sum of b and c is placed in a . here , the variables a,b and c are assumed to be stored in the register $s1,$s2 and $s3 all arithmetic immediate value are signed extended

MIPS operands

Name	Example	Comments
32 registers	$s0-$s7, $t0-$t9, $zero, $a0-$a3, $v0-$v1, $gp, $fp, $sp, $ra, $at	Fast locations for data. In MIPS, data must be in registers to perform arithmetic, register $zero always equals 0, and register $at is reserved by the assembler to handle large constants.
2^{30} memory words	Memory[0], Memory[4], ..., Memory[4294967292]	Accessed only by data transfer instructions. MIPS uses byte addresses, so sequential word addresses differ by 4. Memory holds data structures, arrays, and spilled registers.

MIPS assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add $s1,$s2,$s3	$s1 = $s2 + $s3	Three register operands
	subtract	sub $s1,$s2,$s3	$s1 = $s2 – $s3	Three register operands
	add immediate	addi $s1,$s2,20	$s1 = $s2 + 20	Used to add constants
Data transfer	load word	lw $s1,20($s2)	$s1 = Memory[$s2 + 20]	Word from memory to register
	store word	sw $s1,20($s2)	Memory[$s2 + 20] = $s1	Word from register to memory
	load half	lh $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	load half unsigned	lhu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	store half	sh $s1,20($s2)	Memory[$s2 + 20] = $s1	Halfword register to memory
	load byte	lb $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	load byte unsigned	lbu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	store byte	sb $s1,20($s2)	Memory[$s2 + 20] = $s1	Byte from register to memory
	load linked word	ll $s1,20($s2)	$s1 = Memory[$s2 + 20]	Load word as 1st half of atomic swap
	store condition. word	sc $s1,20($s2)	Memory[$s2+20]=$s1;$s1=0 or 1	Store word as 2nd half of atomic swap
	load upper immed.	lui $s1,20	$s1 = 20 * 2^{16}	Loads constant in upper 16 bits
Logical	and	and $s1,$s2,$s3	$s1 = $s2 & $s3	Three reg. operands; bit-by-bit AND
	or	or $s1,$s2,$s3	$s1 = $s2 | $s3	Three reg. operands; bit-by-bit OR
	nor	nor $s1,$s2,$s3	$s1 = ~ ($s2 | $s3)	Three reg. operands; bit-by-bit NOR
	and immediate	andi $s1,$s2,20	$s1 = $s2 & 20	Bit-by-bit AND reg with constant
	or immediate	ori $s1,$s2,20	$s1 = $s2 | 20	Bit-by-bit OR reg with constant
	shift left logical	sll $s1,$s2,10	$s1 = $s2 << 10	Shift left by constant
	shift right logical	srl $s1,$s2,10	$s1 = $s2 >> 10	Shift right by constant
Conditional branch	branch on equal	beq $s1,$s2,25	If ($s1 == $s2) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne $s1,$s2,25	If ($s1!= $s2) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt $s1,$s2,$s3	If ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than; for beq, bne
	set on less than unsigned	sltu $s1,$s2,$s3	If ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than unsigned
	set less than immediate	slti $s1,$s2,20	If ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant
	set less than immediate unsigned	sltiu $s1,$s2,20	If ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant unsigned
Unconditional jump	jump	j 2500	go to 10000	Jump to target address
	jump register	jr $ra	go to $ra	For switch, procedure return
	jump and link	jal 2500	$ra = PC + 4; go to 10000	For procedure call

Fig .1.6 : MIPS Assembly language

All arithmetic operations have exactly three operands, no more and no less, conforms have this conforms to the philosophy of keeping the hardware simple. This situation illustrates the first of four underlying principles of hardware design.***Design Principle 1:*** **Simplicity favors regularity.Compiling Two C Assignment Statements into MIPS**__**Example 1:**__This egment of a C program contains the five variables a, b, c, d, and e. Since Java volved from C, this example and the next few work for either high-level programming language:

```
a = b + c;
d = a - e;
```

Answer

The translation from C to MIPS assembly language instructions is performed by the *compiler*. Show the MIPS code produced by a compiler. A MIPS instruction operates on two source operands and places the result in one destination operand. Hence, the two simple statements above compile directly into these two MIPS assembly language instructions:

```
add a, b, c
sub d, a, e
```

__**Example 2:**__

A somewhat complex statement contains the five variables f, g, h, i, and j:

```
f = (g + h) - (i + j);
```

What might a C compiler produce?

Answer

The compiler must break this statement into several assembly instructions, since only one operation is performed per MIPS instruction. The first MIPS instruction calculates the sum of g and h. We must place the result somewhere, so the compiler creates a temporary variable, called t0:

```
add t0,g,h # temporary variable t0 contains g + h
```

Although the next operation is subtract, we need to calculate the sum of i and j before we can subtract. Thus, the second instruction places the sum of i and j in another temporary variable created by the compiler, called t1:

```
add t1,i,j  # temporary variable t1 contains i + j
```

Finally, the subtract instruction subtracts the second sum from the first and places the difference in the variable f, completing the compiled code:

```
sub f,t0,t1 # f gets t0 - t1, which is (g + h) - (i + j)
```

8.Expalin about Operands of the Computer Hardware:

Operands of the Computer Hardware Over View memory operand constant or immediate operands Index register*The operands of the computer hardware using Design principal two:Design Principle 2:* Smaller is faster.A very large number of registers may increase the clock cycle time simply because it takes electronic signals longer when they must travel farther.Use fewer register to conserve energy**Example: Compiling a C Assignment Using Registers:**It is the compiler's job to associate program variables with registers. Take, for instance, the assignment statement from our earlier example:

```
f = (g + h) - (i + j);
```

The variables f, g, h, i, and j are assigned to the registers $s0, $s1, $s2, $s3, and $s4, respectively. What is the compiled MIPS code?**Answer**The compiled program is very similar to the prior example, except we replace the variables with the register names mentioned above plus two temporary registers, $t0 and $t1, which correspond to the temporary variables above:

```
add $t0,$s1,$s2 # register $t0 contains g + h
add $t1,$s3,$s4 # register $t1 contains i + j
sub $s0,$t0,$t1 # f gets $t0 - $t1, which is (g + h)-(i + j)
```

Memory Operands

Programming languages have simple variables that contain single data elements. But they also have more complex data structures(many more data elements)—arrays and structures.Thus, MIPS must include instructions that transfer data between memory and registers. Such instructions are called **data transfer instructions**. To access a word in memory, the instruction must supply the memory **address**.Memory is just a large, **single-dimensional array**, with the address acting as the index to that array, starting at 0. For example, in Figure 1.7, the address of the third data element is 2, and the value of Memory[2] is 10

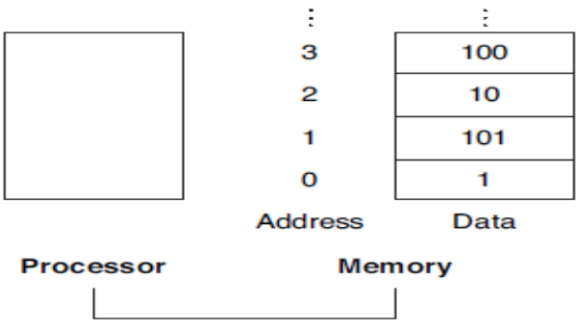


FIGURE 1.6 Memory addresses and contents of memory at those locations

The data transfer instruction that copies data from memory to a register is traditionally called ***load*.**The format of the load instruction is the name of the operation followed by the register to be loaded, then a constant and register used to access memory.The sum of the constant portion of the instruction and the contents of the second register forms the memory address. The actual MIPS name for this instruction is lw, standing for ***load word.***

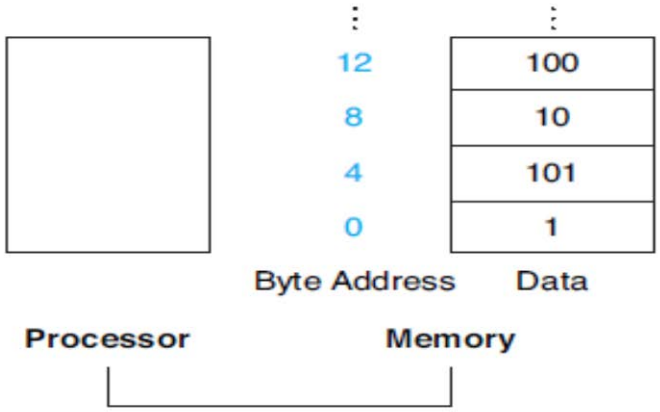


FIGURE 1.7 Actual MIPS memory addresses and contents of memory for those words.

Alignment restriction:

In MIPS, words must start at addresses that are multiples of 4. This requirement is called an **alignment restriction.**Alignment restriction A requirement that data be aligned in memory on natural boundaries .many architecture have alignment restriction**Big endian and little Endian:**8 bit bytes are divided into two parts:Address of the left most byte is called “big endian” and right most byte is called “little endian “**Compiling an Assignment When an Operand Is in Memory:Example 1:**Let’s assume that A is an array of 100 words and that the compiler has associated the variables g and h with the registers $s1 and $s2 as before. Let’s also assume that the starting address, or *base address,* of the array is in $s3. Compile this C assignment statement:**MIPS Code**In the given statement, there is a single operation. Whereas, one of the operands is in memory, so we must carry this operation in two steps:Step 1: load the temporary register($s3) + 8Step 2: perform addition with h(($s2)), and store result in g($s1)

```
lw   $t0,8($s3) # Temporary reg $t0 gets A[8]

add  $s1,$s2,$t0 # g = h + A[8]
```

The constant in a data transfer instruction (8) is called the *offset,* and the register added to form the address ($s3) is called the ***base register*.Example 2:Compiling Using Load and Store** What is the MIPS assembly code for the C assignment statement

```
A[12] = h + A[8];
```

below?
Assume variable h is associated with register $s2 and the base address of the array A is in $s3.

MIPS code

```
lw   $t0,32($s3)  # Temporary reg $t0 gets A[8]
add  $t0,$s2,$t0  # Temporary reg $t0 gets h + A[8]
```

The final instruction stores the sum into A[12], using 48 (4 × 12) as the offset and register $s3 as the base register

```
sw   $t0,48($s3)  # Stores h + A[8] back into A[12]
```

Load word and store word are the instructions that copy words between memory and registers in the MIPS architecture. Other brands of computers use other instructions along with load and store to transfer data.**Constant or Immediate Operands**Constant variables are used as one of the operand for many arithmetic operation in MIPS architecture.The constants would have been placed in memory when the program was loaded.To avoid load instruction used in arithmetic instruction we can use one operand is a constantThis quick add instruction with one constant operand is called *add immediate* or addi. To add 4 to register $s3, we just write

***Design Principle 3:* Make the common case fast.**

```
lw $t0, AddrConstant4($s1)  # $t0 = constant 4
add $s3,$s3,$t0             # $s3 = $s3 + $t0 ($t0 == 4)
```

assuming that $s1 + AddrConstant4 is the memory address of the constant 4.

```
addi    $s3,$s3,4           # $s3 = $s3 + 4
```

Advantage of constant operands

It uses less energy
It performs operation in more fast

Index register

The register in the data transfer instructions was originally invented to hold an index of an array with the offset used for the starting address of an array. Thus, the base register is also called the *index register.*

9. Discuss about the various techniques to represent instruction in computer system. (Apr/May-2015)

representing Instructions in the Computer

Instructions are kept in the computer as a series of high and low electronic signals and may be represented as numbers.Each instruction is used to perform some task and representing those instructions will be varied from one language to another language. To represent the instruction we need to use the instruction format specified by particular language.**Instruction format**Instruction format is a form of representation of an instruction composed of fields of binary numbers.**Registers**In computer hardware registers are referred to by almost all instructions, there must be a convention to map register names into numbers. In MIPS assembly language, registers **$s0 to $s7** map onto registers **16 to 23**, and registers **$t0 to $t7** map onto registers **8 to 15.** Hence, $s0 means register 16, $s1 means register 17, $s2 means register 18, . . . , $t0 means register 8, $t1 means register 9, and so on.**Machine language**Binary representation used for communication within a computer ystem.MIPS instruction takes exactly 32 bits—the same size as a data wordInstructions used in **machine language** and a sequence of such instructions *machine code*(0,1).**Example 1.Translating a MIPS Assembly Instruction into a Machine Instruction**Let's do the next step in the refinement of the MIPS language as an example. We'll show the real MIPS language version of the instruction represented symbolically as

```
add $t0,$s1,$s2
```

first as a combination of decimal numbers and then of binary numbers.

Answer:

The decimal representation is

0	17	18	8	0	32

Each of these segments of an instruction is called a *field*. The first and last fields containing 0 and 32 in this case) in combination tell the MIPS computer that This instruction performs addition. The second field gives the number of the register that is the first source operand of the addition operation (17 = $s1),Third field gives the other source operand for the addition (18 = $s2). The fourth field contains the number of the register that is to receive the sum (8 = $t0). The fifth field is unused in this instruction, so it is set to 0. This instruction adds register $s1 to register $s2 and places the sum in register $t0.This instruction can also be represented as fields of binary numbers as opposed to decimal:

000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Here 10001 is the binary value for 17 as like this remaining field value will be represented in binary

Hexadecimal number

Computer can use binary numbers to reading and writing data in binary numbers. We avoid that by using a higher base than binary that converts easily into binary. Since almost all computer data sizes are multiples of 4, **hexadecimal** (base 16) numbers are popular.ince base 16 is a power of 2, we can trivially convert by replacing each group of four binary digits by a single hexadecimal digit, and vice versa

Hexadecimal	Binary	Hexadecimal	Binary	Hexadecimal	Binary	Hexadecimal	Binary
0_{hex}	0000_{two}	4_{hex}	0100_{two}	8_{hex}	1000_{two}	c_{hex}	1100_{two}
1_{hex}	0001_{two}	5_{hex}	0101_{two}	9_{hex}	1001_{two}	d_{hex}	1101_{two}
2_{hex}	0010_{two}	6_{hex}	0110_{two}	a_{hex}	1010_{two}	e_{hex}	1110_{two}
3_{hex}	0011_{two}	7_{hex}	0111_{two}	b_{hex}	1011_{two}	f_{hex}	1111_{two}

Fig 1.8.Hexadecimal to binary conversion

Example

decimal numbers with *ten*,binary numbers with *two*, andHexadecimal numbers with *hex.* (If there is no subscript, the default is base 10.)**MIPS Fields:**Hence, we have a conflict between the desire to keep all instructions the same length and the desire to have a single instruction format. This leads us to the final hardware design principle:

***Design Principle 4:* Good design demands good compromises**.

The compromise chosen by the MIPS designers is to keep all instructions the same length, thereby requiring different kinds of instruction formats for different kinds of instructions.

MIPS Fields has two kinds of format such asR-type or R- format(for register)I-type or I-format(for immediate)

R-type or R- format :

MIPS fields are given names to make them easier to discuss:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Here is the meaning of each name of the fields in MIPS instructions:

■ *op:* Basic operation of the instruction, traditionally called the **opcode**.
■ *rs:* The first register source operand.
■ *rt:* The second register source operand.
■ *rd:* The register destination operand. It gets the result of the operation.
■ *shamt:* Shift amount. ■ *funct:* Function. This field, often called the *function code,* selects the specific variant of the operation in the op field.

I-format:

A second type of instruction format is called *I-type* (for immediate) or *I-format* and is used by the immediate and data transfer instructions.

The fields of I-format are

Drawback of different format
Multiple formats complicate the hardware.
It increase the complexity.
Solution for above problem is we have to use similar format for that designers must keep all instruction in the same length.

10.Explain about Logical Operations:
Logical Operations:Over view:
Logical shift operation
Shift left logical (sll)
Shift right logical (srl).
Logical AND operation
Logical OR operation
Logical NOT operation
Logical NOR operation
–Useful to operate on fields of bit or individual bits**AND** A logical bit-by bit operation with two operands that calculates a 1 only if there is a 1 in *both* operands.**OR** A logical bit-by bit operation with two operands that calculates a 1 if there is a 1 in *either* operand.**NOT** A logical bit-by bit operation with one operand that inverts the bits; that is, it replaces every 1 with a 0, and every 0 with a 1.**NOR** A logical bit-by bit operation with two operands that calculates the NOT of the OR of the two operands. That is, it calculates a 1 only if there is a

Logical operations	C operators	Java operators	MIPS instructions
Shift left	<<	<<	sll
Shift right	>>	>>>	srl
Bit-by-bit AND	&	&	and, andi
Bit-by-bit OR	|	|	or, ori
Bit-by-bit NOT	~	~	nor

Fig .1.9. .C and Java logical operators and their corresponding MIPS instructions

Logical shift operation :
Shift operation moves all the bits in a word to the left or right side and filling the emptied bits with 0 Based on the direction of shifting it can be classified in two types
Shift left,Shift rightThe actual name of the two MIPS shift instructions are called ***shift left logical*** **(sll) and** ***shift right logical*** **(srl).**
Shift left logical (sll):
It moves all the bits in a word to the left side and empty position is filled with 0
Example
MSB 7 6 5 4 3 2 1 0 LSB

0	**0**	**0**	**1**	**0**	**1**	**1**	**1**

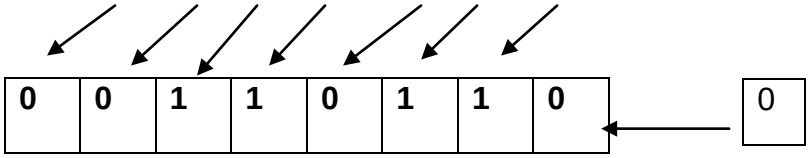


Fig.1.10. shift left logical of a binary number by 1

It moves all the bits in a word to the right side and empty position is filled with 0

Example

MSB 7 6 5 4 3 2 1 0 LSB

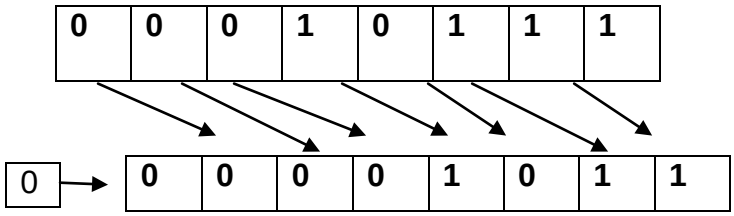


Fig.1.11. shift right logical of a binary number by 1

if register $t2 contains

$$0000\ 0000\ 0000\ 0000\ 0000\ 1101\ 1100\ 0000_{two}$$

and register $t1 contains

$$0000\ 0000\ 0000\ 0000\ 0011\ 1100\ 0000\ 0000_{two}$$

then, after executing the MIPS instruction

```
and $t0,$t1,$t2     # reg $t0 = reg $t1 & reg $t2
```

the value of register $t0 would be

$$0000\ 0000\ 0000\ 0000\ 0000\ 1100\ 0000\ 0000_{two}$$

Logical OR operation:A logical bit-by bit operation with two operands that calculates a 1 if there is a 1 in *either* operand.**Example**if register $t2 contains

$$0000\ 0000\ 0000\ 0000\ 0000\ 1101\ 1100\ 0000_{two}$$

and register $t1 contains

$$0000\ 0000\ 0000\ 0000\ 0011\ 1100\ 0000\ 0000_{two}$$

then, after executing the MIPS

```
or $t0,$t1,$t2 # reg $t0 = reg $t1 | reg $t2
```

instruction

is this value in register $t0:

$0000\ 0000\ 0000\ 0000\ 0011\ 1101\ 1100\ 0000_{two}$

Logical NOT operation

A logical bit-bybit operation with one operand that inverts the bits; that is, it replaces every 1 with a 0, and every 0 with a 1.

Example

$t1 =	*0010*	*1010*	*1100*	*1000*
NOT $t1 =	1101	0101	0011	0111

Logical NOR operation

A logical bit-bybit operation with two operands that calculates the NOT of the OR of the two operands. it calculates a 1 only if there is a 0 in *both* operands.

Exampleif register $t2 contains

$0000\ 0000\ 0000\ 0000\ 0000\ 1101\ 1100\ 0000_{two}$

and register $t1 contains

$0000\ 0000\ 0000\ 0000\ 0011\ 1100\ 0000\ 0000_{two}$

then, after executing the MIPS instruction

```
nor $t0,$t1,$t3 # reg $t0 = ~ (reg $t1 | reg $t3)
```

is this value in register $t0:

$1111\ 1111\ 1111\ 1111\ 1100\ 0011\ 1111\ 1111_{two}$

11. Explain about Control operation with examples.

Control operation

Decision making is commonly represented in programming languages using the *if* statement, sometimes combined with *go to* statements and labels.MIPS assembly language includes two decision- making instructions, similar to an *if* statement with a *go to*. The first instruction is

```
beq register1, register2, L1
```

This instruction means go to the statement labeled L1 if the value in register1 equals the value in register2. The mnemonic beq stands for *branch if equal*.The second instruction is

```
bne register1, register2, L1
```

It means go to the statement labeled L1 if the value in register1 does *not* equal the value in register2.The mnemonic bne stands for *branch if not equal*. These two instructions are traditionally called **conditional branches**.

conditional branches:In MIPS language beq and bne stands for conditional branches.An instruction that requires the comparison of two values and that allows for a subsequent

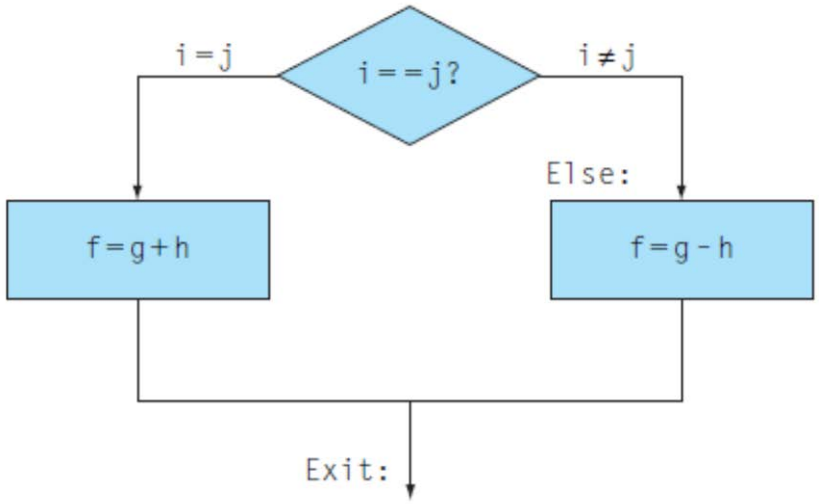
transfer of control to a new address in the program based on the outcome of the comparison.

Example:Compiling *if-then-else* into Conditional Branches:

In the following code segment, f, g, h, i, and j are variables. If the five variables f through j correspond to the five registers $s0 through $s4, what is the compiled MIPS code for this C *if* statement?

```
if (i == j) f = g + h; else f = g - h;
```

Answer



The flow chart shows what the MIPS code should do.For first instructions we have to use beq because it expresses the comparison between i and j.Second instruction we have to use bne. Generally the code will be more efficient to test the opposite conditionHere the opposite condition performs the subsequent then part of the if and we use the branch if register are not equal instruction (bne)

```
bne $s3,$s4,Else   # go to Else if i ≠ j
```

The next assignment statement performs a single operation, and if all the operands are allocated to registers, it is just one instruction:

```
add $s0,$s1,$s2    # f = g + h (skipped if i ≠ j)
```

To distinguish between conditional and unconditional branches, the MIPS name for this type of instruction is *jump*, abbreviated as j (the label Exit is defined below).

```
j Exit    # go to Exit
```

The label Exit that is after this instruction, showing the end of the *if-then-else* compiled code:

```
Else:sub $s0,$s1,$s2   # f = g - h (skipped if i = j)
Exit:
```

Loops

Looping statements are used to execute the same task more than one time until certain condition gets failed.**Example:Compiling a *while* Loop in C**

Here is a traditional loop in C:

```
while (save[i] == k)
        i += 1;
```

Assume that i and k correspond to registers $s3 and $s5 and the base of the array save is in $s6. What is the MIPS assembly code corresponding to this C segment?

Answer

The first step is to load save[i] into a temporary register.Before we can load save[i] into a temporary register, we need to have its address. Before we can add i to the base of array save to form the address, we must multiply the index i by 4 due to the byte addressing problem. We need to add the label Loop to it so that we can branch back to that instruction at the end of the

```
Loop: sll  $t1,$s3,2    # Temp reg $t1 = i * 4
```

loop.

To get the address of save[i], we need to add $t1 and the base of save in $s6:

```
add $t1,$t1,$s6     # $t1 = address of save[i]
```

Now we can use that address to load save[i] into a temporary register:

```
lw  $t0,0($t1)      # Temp reg $t0 = save[i]
```

The next instruction performs the loop test, exiting if save[i] ≠

```
bne  $t0,$s5, Exit  # go to Exit if save[i] ≠ k
```

k:

The next instruction adds 1 to i:

```
addi  $s3,$s3,1     # i = i + 1
```

The end of the loop branches back to the *while* test at the top of the loop. We just add the Exit label after it, and we're done:

```
      j    Loop          # go to Loop
Exit:
```

Basic Block:

A sequence of instructions without branches (except possibly at the end) and without branch targets or branch labels (except possibly at the beginning).

Comparison instructions:MIPS compilers use the slt, slti, beq, bne, and the fixed value of 0 (always available by reading register $zero) to create all relative conditions: equal, not equal, less than, less than or equal, greater than, greater than or equal. This is called comparison instructions.

Case/Switch Statement

Most programming languages have a *case* or *switch* statement that allows the programmer to select one of many alternatives depending on a single value.

Switch statement can be implemented in two ways:

Using chain of *if-then-else* statements.Using jump address table or jump table.

12. What is the need for addressing in a computer system? explain the different addressing modes with suitable examples. (Apr/May-2015,Nov/Dec -2014)

Addressing Modes

The different ways in which the location of an operand is specified in an instruction are referred to as addressing modes. Different types of addresses involve tradeoffs between instruction length, addressing flexibility and complexity of address calculation.

OverView

The different types of addressing modes are:
Immediate addressing mode
Direct or absolute addressing mode
Indirect addressing mode
Register addressing mode
Indexed addressing mode(Displacement)
Relative addressing mode
Auto increment
Auto decrement
Implied (Stack, and a few others

Immediate Addressing and Small Operands

The operand is given explicitly in the instruction.
Example: **MOVE #200 , R0**
The above statement places the value 200 in the register R0. A common convention is to use the sharp sign (#) in front of the value to indicate that this value is to be used as an immediate operand.A great many immediate mode instructions use small operands (8 bits). In 32 or 64 bit machines with variable length instructions space is wasted if immediate operands are required to be the same as the register size.Some instruction formats include a bit that allows small operands to be used in immediate instructions.ALU will zero-extend or sign-extend the operand to the register size

Instruction

op-code	operand

Figure 1.12 Immediate

Direct Addressing (Absolute addressing mode):

The operand is in a memory location; the address of this location is given explicitly in the instruction. (In some assembly languages, this mode is called Direct.Example: **MOVE LOC, R2**This instruction copies the contents of memory location of LOC to register R2.

- Address field contains address of operand
- Effective address (EA) = address field (A)
- e.g. add ax, count or add ax,[10FC]

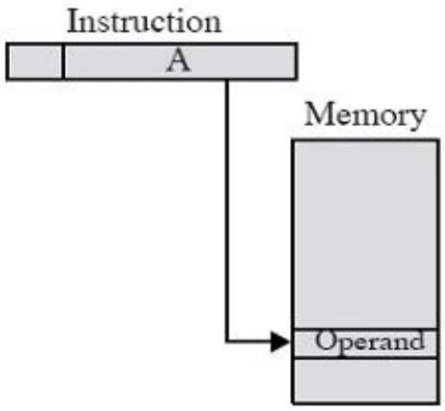


Figure 1.13. Direct Addressing

Memory-Indirect Addressing:

The effective address of the operand is the contents of a register or memory location whose address appears in the instruction. Example **Add (R2),R0** Register R2 is used as a pointer to the numbers in the list, and the operands are accessed indirectly through R2

The initialization section of the program loads the counter value n from memory location N into Rl and uses the immediate addressing mode to place the address value NUM 1, which is the address of the first number in the list, into R2.

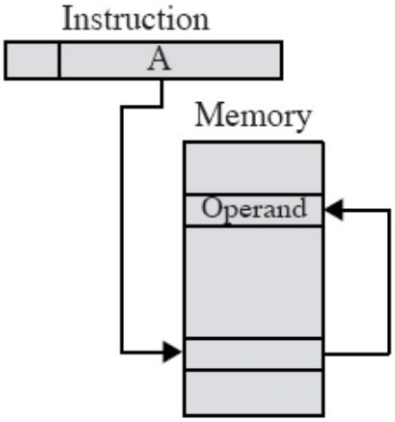


Figure.1.14. Memory-Indirect Addressing

Register Direct Addressing :

The operand is the contents of a processor register; the name (address) of the register is given in the instruction.Example: **MOVE R1,R2**This instruction copies the contents of register R2 to R1.Operand(s) is(are) registers EA = R• There are a limited number of registers Therefore a very small address field is neededShorter instructions Faster instruction fetch.

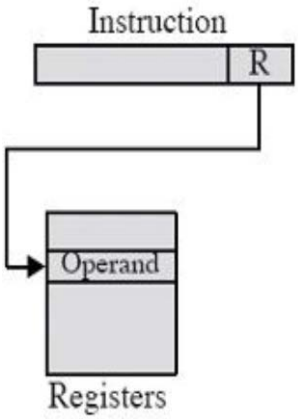


Figure 1.15. Register Direct Addressing:

Register Indirect Addressing:

• Similar to memory-indirect addressing;

EA = (R)

Operand is in memory cell pointed to by contents of register R

Large address space (2n)

One fewer memory access than indirect Addressing

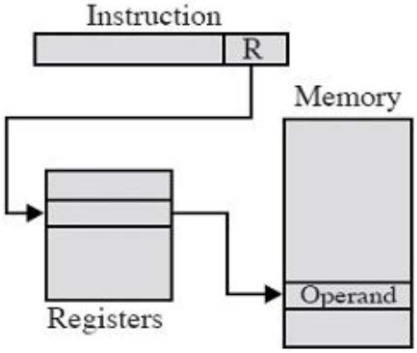


Figure 1.16. Register Indirect Addressi

Displacement Addressing(Index addressing mode)

The effective address of the operand is generated by adding a constant value to the contents ofa register.The register used may be either a special register provided for this purpose, or, more commonly; it may be anyone of a set of general-purpose registers in the processor.In either case, it is referred to as an index register. We indicate the Index mode symbolically as **X(Ri).**Where X denotes the constant value contained in the instruction and Ri is the name of the register involved. The effective address of the operand is given by **EA = X + [Ri].**The contents of the index register are not changed in the process of generating the effectiveaddress EA = A + (R) Combines register indirect addressing with direct addressin Address field hold two values A = base value R = register that holds displacement

or vice versa

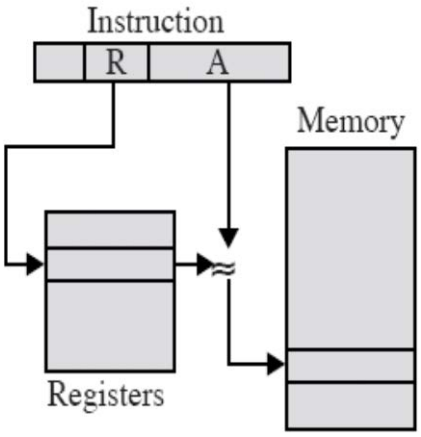


Figure 1.17. Displacement Addressing

Types of Displacement Addressing:

- Relative Addressing
- Base-register addressing
- Indexing

Relative Addressing:

We have defined the Index mode using general-purpose processor registers. A useful version of this mode is obtained if the program counter, PC, is used instead of a general purpose register. Then, X(PC) can be used to address a memory location that is X bytes away from the location presently pointed to by the program counter.Since the addressed location is identified **"relative"** to the program counter, which always identifies the current execution point in a program, the name Relative mode is associated with this type of addressing.EA = X + (PC)

Base-Register Addressing:

A holds displacement R holds pointer to base address• R may be explicit or implicit segment registers in 8x86 are base registers and are involved in all EA computationsx86 processors have a wide variety of base addressing formatsmov eax,[edi + 4 * ecx]sub [bx+si-12]

Auto- increment mode:

The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next item in a list.We denote the Auto increment mode by putting the specified register in parentheses, to show that the contents of the register are used as the effective address, followed by a plus sign to indicate that these contents are to be incremented after the operand is accessed. Thus, the Auto increment mode is written as **(Ri) +**As a companion for the Auto increment mode, another useful mode accesses the items of a list in the reverse order:

Auto- decrement mode:

The contents of a register specified in the instruction is first automatically decremented and is then used as the effective address of the operand. We denote the Auto decrement mode by putting the specified register in parentheses, preceded by a minus sign to indicate that the contents of the register are to be decremented before being used as the effective address.Thus, we write **- (Ri)**

Stack Addressing

Operand is (implicitly) on top of stack

Eg— PUSH, POP

X87 is a stack machine so it has instructions such as
FADDP ; st(1) <- st(1) + st(0); pop stack
; result left in st(0)
FIMUL qword ptr [bx]
; st(0) <- st(0) * 64 integer pointed to
; by bx

Name	Assembler syntax	Addressing function
Immediate	#Value	Operand = Value
Register	Ri	EA = Ri
Absolute (Direct)	LOC	EA = LOC
Indirect	(Ri)	EA = [Ri]
	(LOC)	EA = [LOC]
Index	X(Ri)	EA = [Ri] + X
Base with index	(Ri,Rj)	EA = [Ril + [Rj]
Base with index and offset	X(Ri,Rj)	EA = [Ri] + [Rj] + X
Relative	X(PC)	EA = [PC] +X
Autoincrement	(Ri)+	EA = [Ri]; Increment Ri
Autodecrement	-(Ri)	Decrement Ri; EA = [Ri]

EA = effective address Value = a signed number

Table 1.18. Generic addressing modes

13. Consider the computer with three instruction classes and CPI measurements as given below and instruction counts for each instruction class for the same program from two different compilers are given. Assume that the computer's clock rate is 4 GHZ. Which code sequence will execute faster according to execution time? (Nov/Dec-2014)

Code from		CPI for the instruction class		
A	B	C		
CPI		1	2	3

Code from		Instruction count for each class		
A	B	C		
Compiler 1		2	1	2
Compiler 2		4	1	1

ANSWER

Sequence 1 executes 2 + 1 + 2 = 5 instructions. Sequence 2 executes 4 + 1 + 1 = 6 instructions. Therefore, sequence 1 executes fewer instructions. We can use the equation for CPU clock cycles based on instruction count and CPI to find the total number of clock cycles for each sequence:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

This yields
CPU clock cycles1 = (2 × 1) + (1 × 2) + (2 × 3) = 2 + 2 + 6 = 10 cycles
CPU clock cycles2 = (4 × 1) + (1 × 2) + (1 × 3) = 4 + 2 + 3 = 9 cycles

So code sequence 2 is faster, even though it executes one extra instruction. Since code sequence 2 takes fewer overall clock cycles but has more instructions, it must have a lower CPI. The CPI values can be computed by

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

Unit - II

ARITHMETIC OPERATIONS

PART-A

1. What is the function of ALU?

Most of the computer operations (arithmetic & logic) are performed in ALU. The data required for the operation is brought by the processor and the operation is performed by the ALU.

2. What are the various ways of representing signed integers in the system?

Sign and magnitude system
1's complement system
2's complement system

3. State the rule for floating point addition. (May/June 2013)

Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents. Set the exponent of the result equal to the larger exponent. Perform addition /subtraction on the mantissa and determine the sign of the Result. Normalize the resulting value, if necessary.

4. Write the logic equations of a binary half adder. (April/May 2011)

Binary Adder•Binary Addition–single bit addition–sum of 2 binary numbers can be larger than either number–need a "carry-out"to store the overflow•Half-Adder–2 inputs (x and y) and 2 outputs (sum and carry)

	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic equations of a binary half adder

S=X XOR Y (XOR)
C=X.Y (AND)

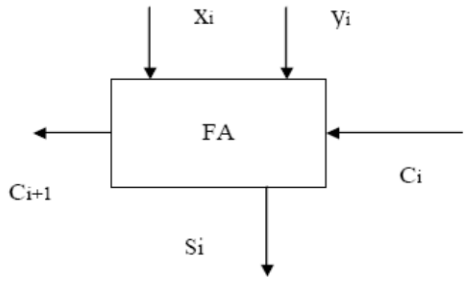
5. How is the number 25 represented in BCD and ASCII code ? (Nov/Dec 2010)

25→(0010 0101) $_{BCD}$
2→(011 0010) $_{ASCII}$
5→(011 0101)$_{ASCII}$

6. What is the purpose of guard bits used in floating point operations?

The guard bits are the extra bit which is used to retain the intermediate steps to increase the accuracy in the final results.
Example: In 32 bit single precision floating point representation the Mantissa bits are limited to 24 bit including leading 1. Some oprations which results in extra bits are called Guard bits.

7. Draw the symbolic representation of the full adder and give the expression for the sum. (Nov/Dec 2006)



Sum=$S_1=X_1+Y_1+C_1$

8. In Confirming to the IEEE standard mention any four situations under which a processor sets exception flag. (Nov/Dec 2006)

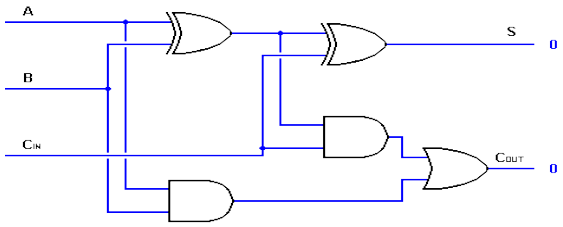
(i) Underflow
(ii) Overflow
(iii) Divide by zero
(iv) Invalid.

9. Why floating point number is more difficult to represent and process than integer? (May/June 2007)

The position of the binary point in the floating point number is variable, it must be given explicitly in the floating point representation.

The floating point number is represented by its sign, a mantissa and an exponent to an implied base for the scale factor. Hence it is more difficult than integer representation.

10. Draw a full adder circuit and give the truth table?(May/June2007)

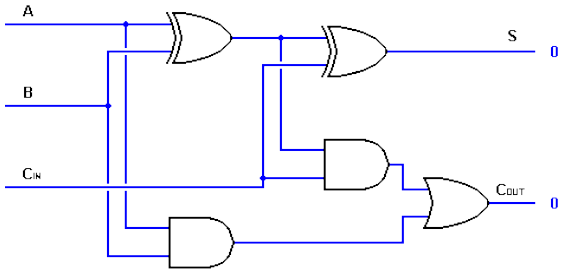


Input bit for number A	Input bit for number B	Carry bit input C_{IN}	Sum bit output S	Carry bit output C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

11. A memory byte location contains the pattern 00101100. What does this pattern represent when interpreted as a number? What does it represent as an ASCII Code? (Nov/Dec 2007)

Interpreted number is 44.
ASCII code is NULL/idle.

12. Draw the full adder circuit using two half adder? (Nov/Dec 2007)



13.What are the various ways of representing signed integers in the system? (Nov/Dec 2007)

Sign and Magnitude system
1's complement system
2's complement system

14. What are tristate-gate? (April/May 2008)

The logic has three output states
Low: Like normal TTL 0 State
High: Like normal TTL 1 State
High impedance: Neither high or low.

15. Define underflow and overflow. (April/May 2008)

In an unsigned fixed point addition, if there is a carry from the MSB position, it indicates an overflow. The resulting number is too small, it indicates underflow.

16. Write rule for addition in floating point operation. (Nov/Dec 2008)

Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.Set the exponent of the result equal to the large exponent.Perform addition on the mantissa and determine the sign of the result. Normalize the result if necessary.

17. What is meant by the stored program concept? Discuss. (May/June 2007)

A set of instruction that performs a task is called a program. Usually the program is stored in the memory. The processor fetches the instructions that take up the program from the memory, one at a time and perform the desired operation.

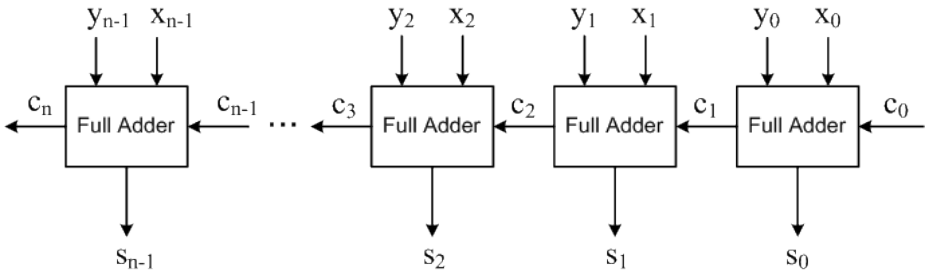
18. Discuss the principle behind the booth's multiplier. (May/June 2007)

The multiplier is represented as a difference between two numbers. This is given by the sequence of required operation by recoding principle.The recoding table is given as

Multiplier		Version of multiplicand selected by bit i
Bit i	Bit i-1	
0	0	0 x M
0	1	+1 x M
1	0	-1 x M
1	1	0 x M

19. What is ripple carry adder? (May/June 2007)

The full adder shares the logic gates(NAND,NOR etc.) in generating S1 and Ci+1 (Sum and Carry). The cascaded connection of n adder blocks can to used to add two n-bit number. Since the carries must propagate, or ripple, through this cascade, the configuration is called as n-ripple carry adder.

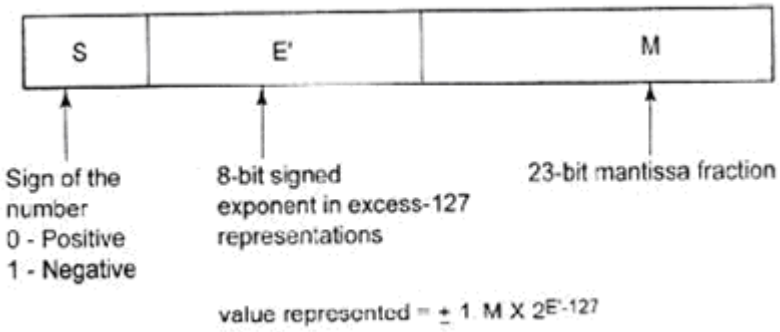


20. Define IEEE floating point single and double precision standard. (Apr/May-2006, Nov/Dec -2005)

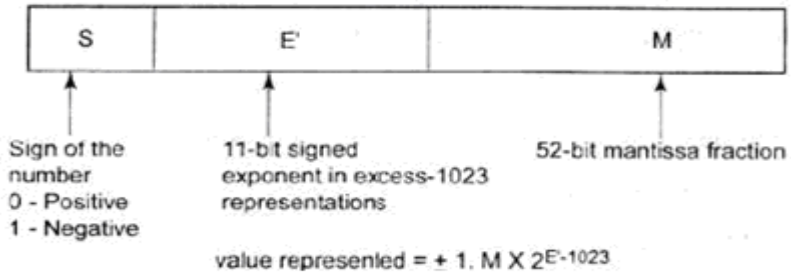
This standard for representing floating point numbers in 32 bits has been developed and specified in detailed by the instance by the Institute of Electrical and Electronics Engineers(IEEE). The IEEE standard describes the floating point representation and the way in which the four basic arithmetic operations are to be performed on these floating point operands.Here are two types of representation of floating point numbers

1.Single Precision
2. Double Precision

1.Single Precision:



2**.Double precision:**



21. How can we speed up the multiplication process?

There are two techniques to speed up the multiplication process:

(i) The first technique guarantees that the maximum number of summands that must be added is n/2 for n-bit operands.(ii) The second technique reduces the time needed to add the summands.

22. What is bit pair recoding? Give an example.

Bit pair receding halves the maximum number of summands. Group the Booth-recoded multiplier bits in pairs and observe the following:The pair (+1 -1) is equivalent to the pair (0 +1). That is instead of adding -1 times the multiplicand m at shift position i to +1 x M at position i+1, the same result is obtained by adding +1x M at position i. Example: 11010 - Bit pair receding value is 0 - 1 - 2.

23. What are the two methods of achieving the 2's complement?
Take the 1's complement of the number and add 1. Leave all least significant O's and the first unchanged and then complement the remaining bits.

24. What are the advantages / features of using Booth algorithm?
It handles both positive and negative multiplier uniformly.
It achieves efficiency in the number of additions required when the multiplier has a few large blocks of 1 's.The speed gained by skipping 1 's depends on the data.

25. Write the algorithm for restoring division.
Do the following for n times:
(i) Shift A and Q left one binary position.
(ii) Subtract M and A and place the answer back in A.
(iii) If the sign of A is 1, set q0 to 0 and add M back to A.
Where, A - Accumulator, M - Divisor, Q - Dividend.

26. Write the algorithm for non restoring division.
Do the following for n times:
Step 1: Do the following for n times:
If the sign of A is 0, shift A and Q left one bit position and subtract M from A, otherwise, shift A and Q left and add M to A. Now, if the sign of A is 0, set q0 to 1, otherwise, set q0 to 0.Step 2: If the sign of A is 1, add M to A.

27. What is the chopping? (Nov/dec-2010)
Simple way to truncate or remove the guard bits and make no changes in the retained bits.

28 Explain Binary Addition and Subtraction.
For addition use normal binary addition
0+0=sum 0 carry 0
0+1=sum 1 carry 0
1+1=sum 0 carry 1
Monitor MSB for overflow Overflow cannot occur when adding 2 operands with the different signs If 2 operand have same sign and result has a different sign, overflow has occurred
Subtraction:

Take 2's complement of subtrahend and
add to minuend
i.e. a - b = a + (-b)

29. **Define Binary Multiplication.**

A complex operation compared with addition and subtraction Many algorithms are used, esp. for large numbers Simple algorithm is the same long multiplication taught in grade school Compute partial product for each digit Add partial products.

30. Define Binary Multiplication Algorithm.

Repeat n times: If Q0 = 1 Add M into A, store carry in C, Shift CF, A, Q right one bit so that: An-1 <- CF Qn-1 <- A0 Q0 is lost Note that during execution Q contains bits from both product and multiplier.

31. What is a co- processor.

A co-processor a separate instruction set processor that is closely that is coupled to CPU and whose instructions and registers are direct extensions of CPU

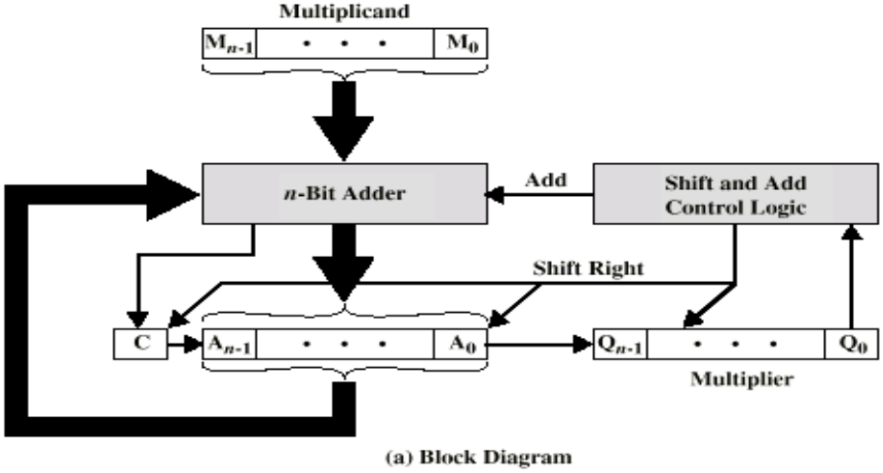
32. Give the advanced features of ALU.

Floating point arithmetic circuit
Pipelined circuit
Co- processor

33. Write Multiplication of positive numbers (Example).

1011 Multiplicand (11 dec)
x 1101 Multiplier (13 dec)
1011 Partial products
0000 Note: if multiplier bit is 1 copy multiplicand (place value)
1011 otherwise zero
1011 Product (143 dec)
10001111 Note: need double length result

34. Diagram for Unsigned Binary Multiplication



(a) Block Diagram

35. Write Floating Point Examples.

(a) Format

$1.1010001 \times 2^{10100} = 0\ 10010011\ 10100010000000000000000 = 1.638125 \times 2^{20}$
$-1.1010001 \times 2^{10100} = 1\ 10010011\ 10100010000000000000000 = -1.638125 \times 2^{20}$
$1.1010001 \times 2^{-10100} = 0\ 01101011\ 10100010000000000000000 = 1.638125 \times 2^{-20}$
$-1.1010001 \times 2^{-10100} = 1\ 01101011\ 10100010000000000000000 = -1.638125 \times 2^{-20}$

(b) Examples

38.Define Guard Bits

In 32 bit single precision floating point representation the mantissa bits are limited to 24 bits including implicit leading 1. But some operations may ***result in extra bits called guard bits*** *and these bits should be retained during the intermediate steps to increase the accuracy in final results.*

36. Examples for Floating Point Arithmetic Operations

Floating Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$X + Y = \left(X_s \times B^{X_E - Y_E} + Y_s\right) \times B^{Y_E}$ $X - Y = \left(X_s \times B^{X_E - Y_E} - Y_s\right) \times B^{Y_E}$ } $X_E \le Y_E$ $X \times Y = (X_s \times Y_s) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_s}{Y_s}\right) \times B^{X_E - Y_E}$

Examples:

$X = 0.3 \times 10^2 = 30$
$Y = 0.2 \times 10^3 = 200$

$X + Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3 = 0.23 \times 10^3 = 230$
$X - Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3 = (-0.17) \times 10^3 = -170$
$X \times Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^5 = 6000$
$X \div Y = (0.3 \div 0.2) \times 10^{2-3} = 1.5 \times 10^{-1} = 0.15$

37. What is a Truncation?

Similarly, allowing guard bits during intermediate steps results in extended mantissa. Thus this extended mantissa should be truncated to 24 bits while generating final results.**There are several ways to truncate**

Chopping
Von-Neumann rounding
Rounding

38. Define Chopping:

Chopping is the simplest way to do truncation (i.e) **Remove the guard bits** and make no changes in the retained bits.

Example0. b_{-1} b_{-2} b_{-3} 000
0. b_{-1} b_{-2} b_{-3} 111 and truncated to 0. b_{-1} b_{-2} b_{-3}Error in chopping ranges from 0 to almost , and result in biased approximation of b_{-3} position.

39. What is a Von-Neumann rounding?

In this method, if the bits to be removed are all 0' s, they are simply dropped , with no changes to the retained bits.If any of the bits to be removed are 1, the least significant bit of the retained but isset to 1.**Example**

$0.\ b_{-1}\ b_{-2}\ b_{-3}$ 000 ⟶ $0.\ b_{-1}\ b_{-2}\ b_{-3}$

$0.\ b_{-1}\ b_{-2}\ b_{-3}$ 100 ⟶ $0.\ b_{-1}\ b_{-2}$ 1 LSB to 1

Error in this technique is larger than chopping and the approximation is unbiased and hence results in high probability of accuracy.

40. What is a Rounding?

Rounding achieves the cloest approximation to the number being truncated and is an unbiased technique.A 1 is added to the LSB position of the bits to be retained if there is a 1 in the MSB position of the bits being removed.

ExampleThus $0.\ b_{-1}\ b_{-2}\ b_{-3}\ 1\ \ldots$ is rounded to $0.\ b_{-1}\ b_{-2}\ b_{-3} + 0.0001$

$0.\ b_{-1}\ b_{-2}\ b_{-3}\ 0\ 1\ \ldots$ is rounded to $0.\ b_{-1}\ b_{-2}\ b_{-3}$ o … … is rounded to $0.\ b_{-1}\ b_{-2}\ b_{-3}$ … this provides the desired approximation.

41. What are the types of ALU?

Two types of ALU:

Combinational ALU

Sequential ALU

42. What is a spatial expansion in ALU?

form a single km-bit ALU by connecting k copies of the m-bit ALU IC . Each component ALU concurrently process a separate "slice" of m bits.

43. What is a Temporal expansion?

Use a m-bit ALU chip to perform an operation on km-bit consecutive steps.

In each step, ALU process a separate m-bit slice of each operand -→ multicycle processing

44. Give any 2' complement multiplier algorithm.

Two types of

Robertson's algorithm ,Booth's algorithm

45. When a ALU is said to be bit sliced?

An ALU is said to be bit sliced if each component ALU concurrently process a separate "slice" of m bits from each Km – bit operand

46. What are the type of micro operations. ?

There are four types of micro operations

Arithmetic micro operation-→ performs arithmetic operations on the data stored in the registers.Logical operations --→ performs bit manipulation operation on the data stored in the registersRegister transfer micro operations -→ transfers binary information from one register to another registerShift micro operations -→ performs shift operations on the data stored in the registers

47. What is a Exceptions? (Nov/Dec-2014)

If a number has the exponent value less than -126 we say underflow has occurred.If a number has the exponent value greater than +127 we say overflow has occurred. Such conditions are called exceptions.**Exception flag is set if** Exponent overflowExponent underflowSignificant underflow,Significant overflow.

48. Discuss the principle of operation of a carry save adder.(Nov/Dec -2005)

Multiplication requires addition of several summands. Carry – Save adder (CSA) speeds up the addition of summands . here instead of carry rippling along the rows , they are saved and introduced in to next row in correct weight position.

Example

49.What are the two attractive features of Booth algorithm?

It handles both positive and negative multipliers uniformly It achieves some efficiency in the number of additions required when the multiplier has a few large blocks of ones.

50. Give an example for the worst case of Booth algorithm .The worst case is shown as below 0 1 0 1 0 1 0 1 0 +1 -1 +1 -1 +1 -1 +1 -1 +1

In the worst case each bit of the multiplier selects the summands. This results in more number of summands.

51. What are the two techniques for speeding up the multiplication operation?

- Bit Pair recoding
- CSA (Carry Save Adder)

52. How bit pair recoding of multiplier speeds up the multiplication process?

It guarantees that the maximum number of summands that must be added is n/2 for n bit operands

53. How CSA speeds up multiplication?

It reduces the time needed to add the summands. Instead of letting the carries ripple along the rows, they can be saved and introduced into the next row, at the correct waited position.

54. What is the advantage of non restoring over restoring division?

Non restoring division avoids the need for restoring the contents of register after an successful subtraction.

55. What is the need for adding binary 8 value to the true exponential in floating point numbers?

This solves the problem of negative exponent. Due to this the magnitude of the numbers can be compared. The excess-x representation for exponents enables efficient comparison of the relative sizes of the two floating point numbers.

56.How over flow occur in Subtraction? (Apr/May-2015)

if 2 Two's Complement numbers are subtracted, and their signs are different, then overflow occurs if and only if the result has the same sign as the subtrahend.

Overflow occurs if

- (+A) − (−B) = −C
- (−A) − (+B) = +C

Example: Using 4-bit Two's Complement numbers (−8 ≤ x ≤ +7)

Subtract −6 from +7

```
  (+7) 0111          0111
 −(−6) 1010 -> Negate -> +0110
 ----------          -----
  13                 1101 = −8 + 5 = −3 : Overflow
```

57.How over flow occur in Addition?

If 2 Two's Complement numbers are added, and they both have the same sign (both positive or both negative), then overflow occurs if and only if the result has the opposite sign. Overflow never occurs when adding operands with different signs.

i.e. Adding two positive numbers must give a positive result

Adding two negative numbers must give a negative result

Overflow occurs if

(+A) + (+B) = −C

(−A) + (−B) = +C

Example: Using 4-bit Two's Complement numbers (−8 ≤ x ≤ +7)

```
 (−7)   1001
+(−6)   1010
 (−13) 1 0011 = 3 : Overflow (largest −ve number is −8)
```

58. What do you mean by sub word parallelism? (Apr/May-2015)

Parallelism will improve the performance of computer. Many application needs the performance high compared to small applications.For example consider the graphical and multimedia applications.Every desktop microprocessor has its own graphical displays, support the graphicsoperations many transistors are used.Many graphics systems used 8 bits to represent each of the three primary colors plus 8 bits for a location of a pixel.

The primary colors are RGB-Red, Green, Blue and it has to be represent using 8 bits.

PART-B

1. Arithmetic Operations

Introduction

1's Complement Addition

Add two numbers and if carry occurs then the carry is added to the result.

1's Complement Subtraction

Take the 1's complement of the subtractedAdd to the minuendIf carry occurs then add carry to the result.**Example 1**: consider 35 - 22 both represented as 7-bit numbers with a sign b +35 in binary is: 00100011 +22 in binary is 0001011 -22 in binary is: 10010110 -22 in 1's complement is: 11101001The sum to be calculated is therefore the sum of the binary for +35 and the 1's complement for -22:

```
  00100011
+ 11101001
  100001100
```

This addition produces a 9th bit. In 1's complement addition, if this occurs, the extra bit is carried to the LSB column and added. Hence:

```
  00001100
+        1
  00001101
```

The final answer has a 0 for its sign bit. This tells us two things: the answer is positive the answer is represented in binary notationThe answer is the positive binary number $0001101=13_{10}$. **Example 2**: 22 - 35, again both represented as 7-bit numbers with a sign bit. +22 in binary is: 00010110

+35 in binary is: 00100011

-35 in 10100011

-35 in 1's complement is: 11011100

The sum to be calculated is below

```
  00010110
+ 11011100
  11110010
```

This time the addition does not produce a 9th bit, but the sign bit is 1. In this case it again tells us two things: the answer is negative the answer is represented in 1's complement notationSo to get the final answer we need to turn our answer into binary. If the 1's complement notation is 11110010 then the binary representation is 10001101 (note - the sign bit doesn't change, it's still a negative number!). This is the binary for -13.

2's Complement Addition:

Two's complement addition follows the same rules as binary addition.Add the numbers and if carry occurs discard the carry.

Example

```
5 + (-3) = 2 0000 0101 = +5
         + 1111 1101 = -3
---------------------------------------------------------------------------
         0000 0010 = +2
```

2's Complement Subtraction:

- Take the 2's complement of the subtrahend
- Add to the minuend
- If carry occurs then discard the carry.

Example:

4 - 12 = (-5) 0000 0111 = +7
+ 1111 0100 = -12

1111 1011 = -5

1. Briefly explain about Arithmetic and Logic Unit(ALU).(or) What are the operations of ALU Explain with neat diagram.

ALU

The various circuits used to execute data processing instructions are usually combined into a single circuit called an **arithmetic logic unit (ALU).**

The complexity of ALU is determined by the way in which its arithmetic instructions are realized.

Over View:

Two types of ALU:

Combinational ALU

Sequential ALU

Combinational ALU

It combines the functions of a two's-complement adder- subtracter with those of a circuit that generates word-based logic functions of the form *f(X,Y)*, **for example**, **AND, XOR, NOT.**It implements most of a CPU's fixed point data-processing instructions. Shown in figure an ALU that has separate subunits for logical and arithmetic operations.The particular class of operation (logical and arithmetic) to be performed is determined by a " mode" control line M attached to a two- way multiplexer that channels the required result to the output bus Z.The specific operation performed by the desired submits is determined by a **"select"** control lines S as shown.The ALU's logical operations are performed bitwise: that is the same operation f is applied to everypair of data lines $\mathbf{x_i y_i}$. the maximum number of distinct logical operations of the form f($\mathbf{x_i}$, $\mathbf{y_{i.}}$) is 16 which is the distinct truth tables of two Boolean variables. Hence the select bus S needs to be of size 4 at most , as in figure. S can also be used to select up to 16 different arithmetic operations such as X + Y , X – Y, Y – X, X +1 (increment) , X – 1 (decrement), and so on , as needed.The logical operations in figure . can be obtained by generating all four minterms of *f(X,Y), namely,*

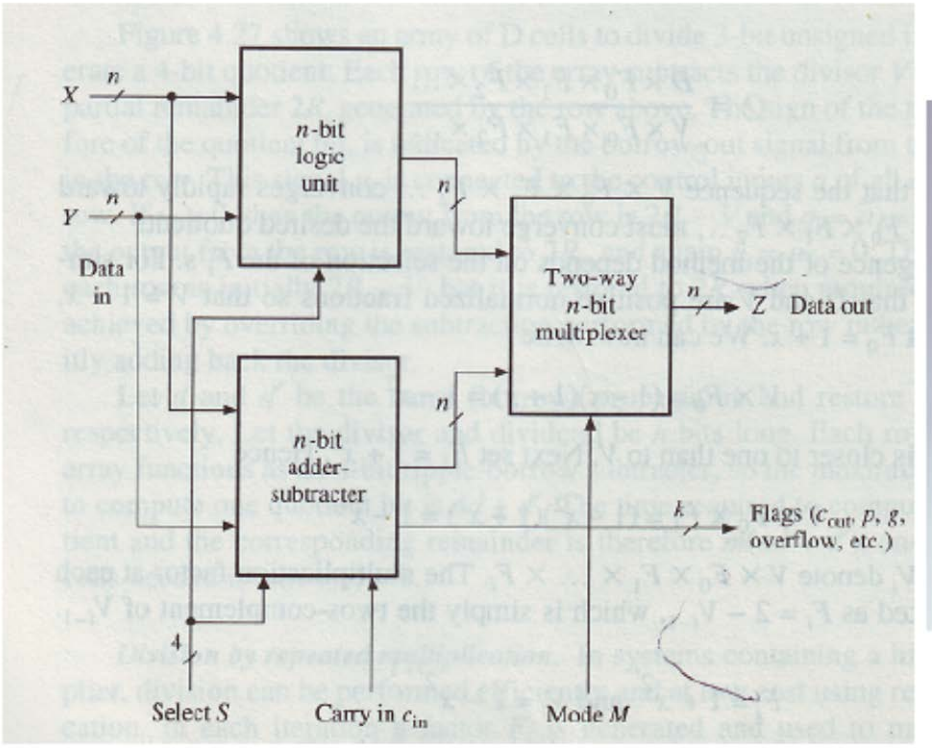


Figure 2.1 A basic n – bit arithmetic – logic unit(ALU)

$m_3 = x_i y_i$. $m_2 = x_i y_i'$ $m_1 = x_i' y_i$ $m_0 = x_i' y_i'$

for every pair $x_i y_i$ of data bits and by using the control lines $S = S_3 S_2 S_1 S_0$ to select desired subsets of the minterms to be ORed together . in particular , if we construct the sum – of – product expression

$f(x_i, y_i) = m_3S_3 + m_2S_2 + m_1S_1 + m_0S_0$ ---------- **1**

$= x_i y_i S_3 + x_i y_i' S_2 + x_i' y_i S_1 + x_i' y_i' S_0$ ---------- **2**

then we see that every combination of S3 S2 S1 S0 produces a different function . for **example, S = 0110 makes** $f(x_i y_i) = x_i y_i' + x_i' y_i$ which is **EXCLUSIVE – OR.**

Because of the bitwise nature of the logic operation, we can replace x_i and y_i with the n- bit words X and Y

$$f(X, Y) = X Y S_3 + X Y' S_2 + X'Y S_1 + X'Y' S_0 \quad \text{----------} \quad 3$$

we can now implement the logic unit directly from equation (3), using several n- bit word gates as in **figure.** The adder – subtracter can be designed by any of the techniques presented earlier, with appropriate additional connections to X , Y, and S.

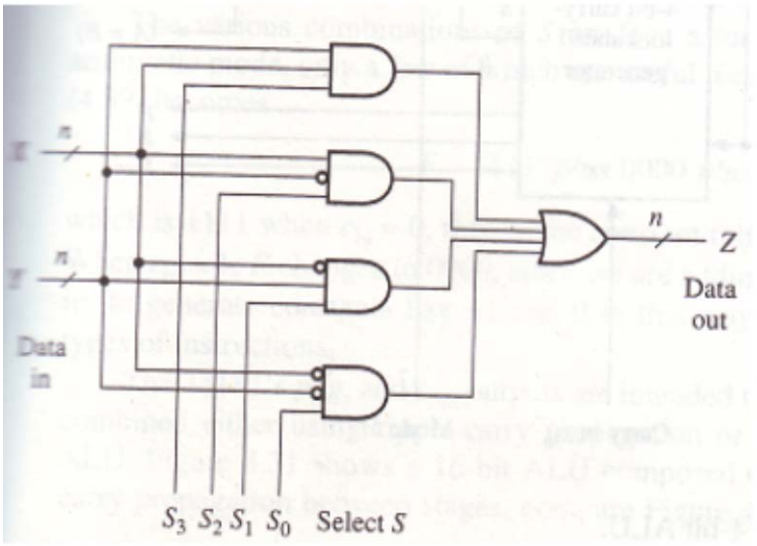


Figure 2.2 An n – bit logic unit that realizes all 16 two – variable function

Sequential ALU

Combinational multipliers and dividers are more complex and slower than addition/subtraction. The number of gates in the multiply – divide logic is greater than that of adder – subtractor by a factor of n.low-cost sequential circuits where add/subtract takes one clock cycle and multiplication/division multicycles.Figure shows a widely used sequential ALU design that ais at minimizing hardware costs . this ALU organization is found in the IAS computer and in many computers built after IAS. It is intended to implement multiplication and division using one of the sequential digit- by-digit shift – and – add / subtractor algorithms .Three one – word registers are used for operand storage :

The accumulator AC

The multipler – quotient registers MQThe data register DRAC and MQ are organized as a single register AC. MQ capable of left – and right – shifting.Additional data processing is provided by a combinational ALU capable of addition, subtraction, and logical operations; this unit derives its inputs from AC and DR and places its results in AC.The MQ register is so called it stores the multiplier during multiflication and the quotient during division. Dr stores the multiplicand or divisior,while the result (product or quotient and remainder) is stored in the register – pair Ac. MQ.

The role of these registers is defined concisely as follows

Addition	AC := AC + DR
Subtraction	AC := AC – DR
Multiplication	AC.MQ := DR × MQ
Division	AC.MQ := MQ/DR
AND	AC := AC and DR
OR	AC := AC or DR
EXCLUSIVE-OR	AC := AC xor DR
NOT	AC := not(AC)

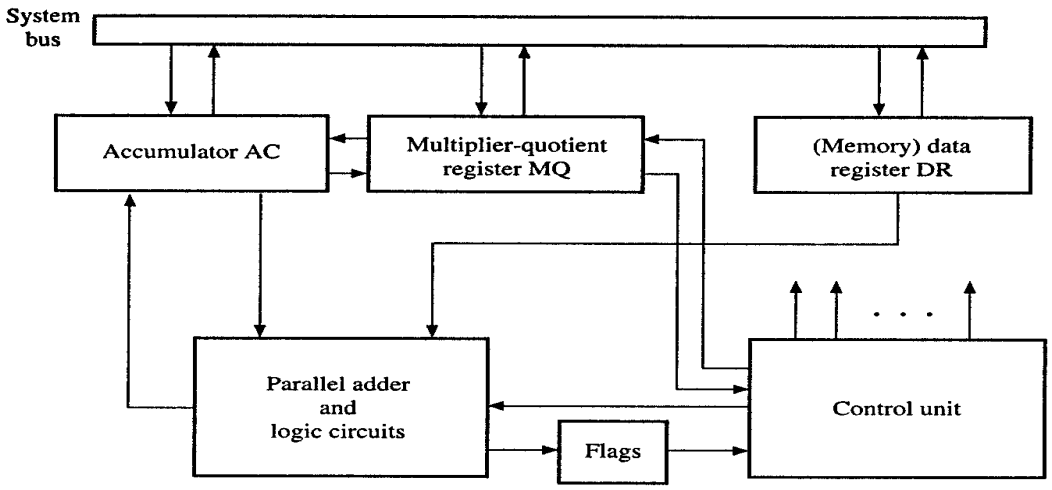


Figure 4.32
Structure of a basic sequential ALU.

DR can serve as a memory data register to store data addressed by an instruction address field ADR. Then DR can be replaced by M(ADR) in the above list of ALU operations, resulting in a one – address memory – referencing format

ALU expansion

1. **Spatial expansion(bit-sliced ALU) :**
   form a single km-bit ALU by connecting k copies of the m-bit ALU IC.
   Each component ALU concurrently process a separate "slice" of m bits.
2. **Temporal expansion** : use a m-bit ALU chip to perform an operation on km-bit consecutive steps. In each step, ALU process a separate m-bit slice of each operand -→ multicycle processing.

2.Briefly explain about Addition and subtraction in Fixed Point Operations with neat diagram and tables.

Addition and subtraction or Fixed Pint

Over view

Basic Adder
Parallel Adder
Ripple carry adder
High speed adder(Carry-look ahead adder), Subtracters

Basic Adder

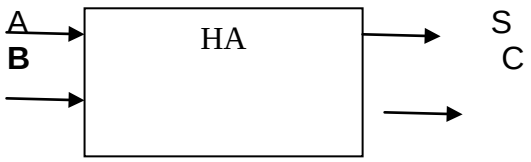
Adders are the basic building blocks of all the arithmetic circuits, adders add two binary numbers and give out sum and carry as output.

Basically we have two types of adders.

Half adder ,Full adder.

Half Adder

Adding two single-bit binary values, A,B produces a Sum bit and a carry out Carry bit. This operation is called half addition and the circuit to realize it is called a half adder.



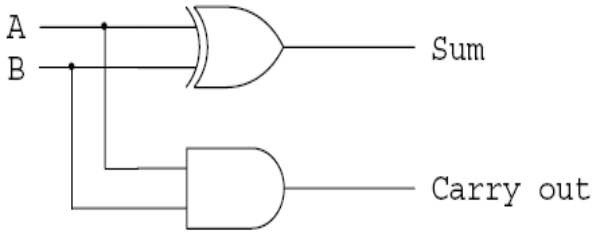
Sum = A'B + AB' =**A ⊕ B,**
Carry = AB

Figure 2.4 Half – adder block diagram

Truth Table:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 2.5 Logic Circuit



Full Adder

Full adder takes three bit input. Adding two single-bit binary values, X, Y with a carry input bit C-in produces a sum bit S and a carry out C-out bit.

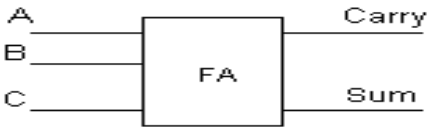


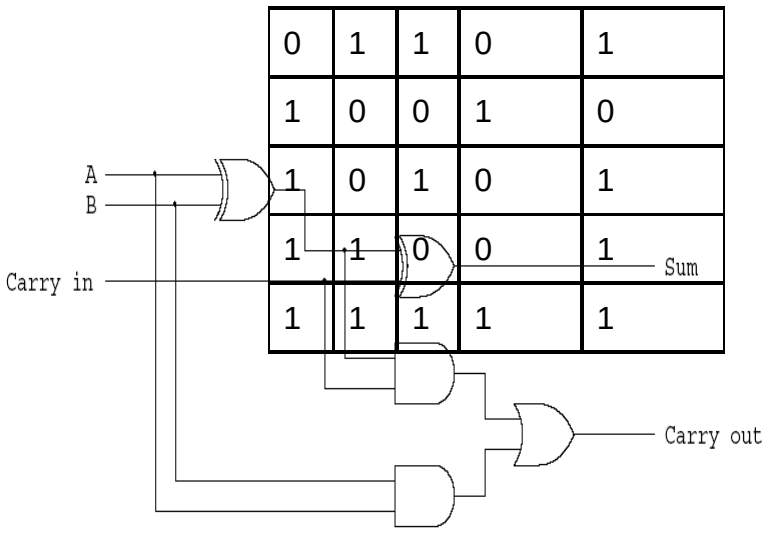
Figure 2.6. Full adder block diagram

Symbol

Sum =A ⊕B⊕C
Carry = (A ⊕B)C+A.B

Truth Table

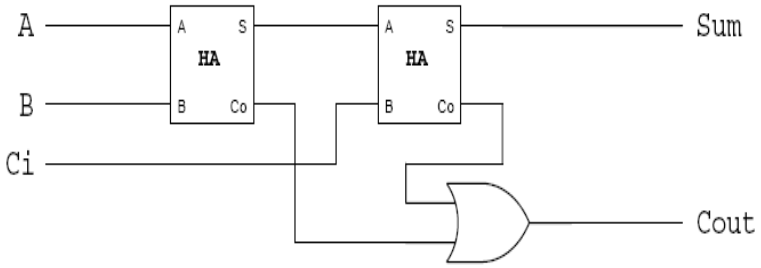
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0



0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 2.7.Logic Circuit

Figure 2.8. Full adder using 2 half adders



Parallel Adder:

A parallel adder is one which has separate adder circuit for each bit. From the control units viewpoints, the parallel adder performs addition of multiple stage (bit position), simultaneously.

However , the internal addition mechanism differs in different types of parallel adder.

There are basically two types of parallel adders:

Carry Ripple Adder

(Carry-lookahead adder)

. Carry Ripple Adder

Adds two 4-bit numbers: A = A_0, A_1, A_2, A_3 , B = B_0, B_1, B_2, B_3. Producing the sum S = S3 S2 S1 S0 and C-out = C4 from the most significant.

A_3 A_2 A_1 A_0

B_3 B_2 B_1 B_0--

S_3 S_2 S_1 S_0

The first column requires only a half adder. For any column above the first there may be a carry from preceding column. Therefore we must use a full adder for each column above the first.

Total Propagation delay = 2 nD = 8D or 8 gate delays

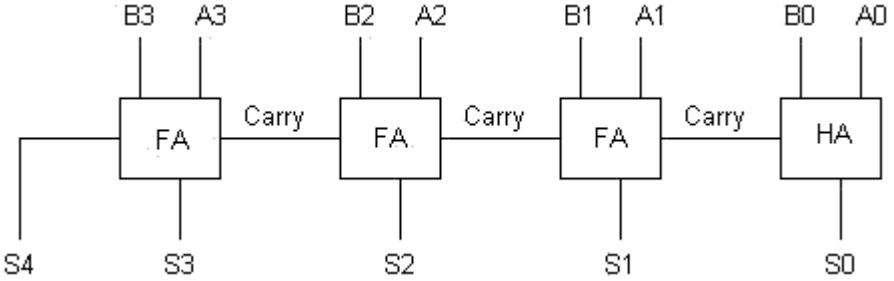


Figure. 2.9 An n- bit ripple – carry adder composed of n 1-bit(full) adders

<u>High speed adder(Carry-look ahead adder)</u> (April/May 2011,Nov/Dec-2014)

High speed adders are designed to improve speed and reduce the time required to determine to form carry signals.One approach is to compute the input carry needed by stage I directly from carry like signals obtaining from all the preceding stages i-1, i-2, …, 0, rather than waiting for normal carries to ripple slowly from stage to stage

This principle is followed by **Carry-lookahead adder**

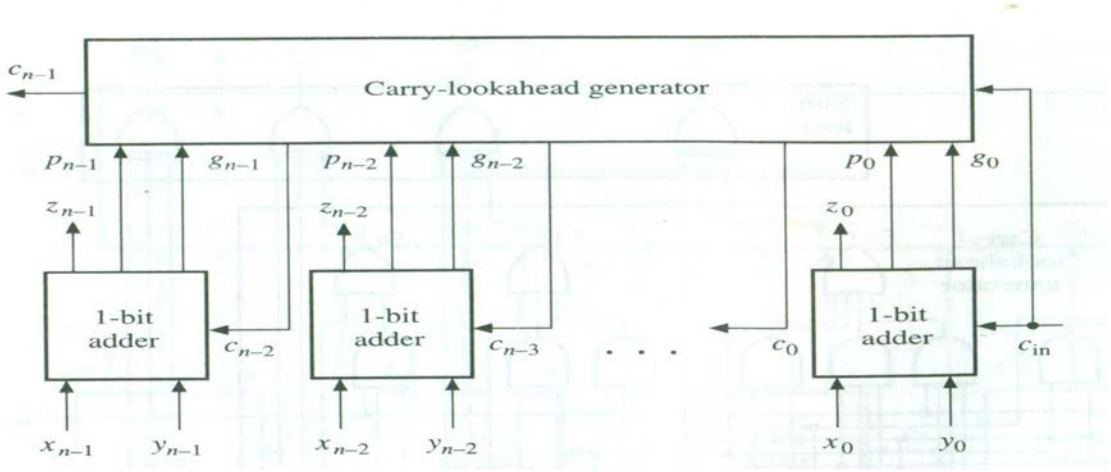


Figure 2.10. Overall structure of carry-lookahead adder

An n-bit carry-lookahead adder is formed n-stages, each of which is basically full adder modifying by replacing its carry output line C_i by two auxiliary signals called p_i and g_i or propagate and generate respectively, which is defined as,$\mathbf{g_i = x_iy_i, p_i = x_i + y_i}$The name generate comes from the fact that stage I generates a carry of 1(ci = 1) independent of the value of ci-1 if both xi and yi are 1; that is, of xiyi = 1Stage i propagates c_{i-1};that is, it makes $c_i = 1$ in response to $c_{i-1} = 1$ if x_i or y_i is 1, if $x_i + y_i = 1$Now the usual equation $c_i = x_iy_i + x_ic_{i-1} + y_ic_{i-1}$, denoting the carry signal c_i to be sent to stage i+1, can be rewritten in terms of g_i and p_i,

$$\mathbf{c_i = g_i + p_ic_{i-1}} \qquad \text{----------} \rightarrow \mathbf{1}$$

Similarly, c_{i-1} can be expressed in terms of g_{i-1}, p_{i-1} and c_{i-1},

$$\mathbf{c_{i-1} = g_{i-1} + p_{i-1}c_{i-2}} \qquad \text{----------} \rightarrow \mathbf{2}$$

By substituting 2 in 1,

$$\mathbf{c_i = g_i + p_i\,(g_{i-1} + p_{i-1}c_{i-2})}$$

$$\mathbf{c_i = g_i + p_ig_{i-1} + p_ip_{i-1}c_{i-2}}$$

- Continuing in this way, ci can be expressed as a sum of products function of the p and g outputs of all the preceding stagesFor example, the carries in a four-stage carry-

lookahead adder are defined as$c_0 = g_0 + p_0c_{in}c_1 = g_1 + p_1g_0 + p_1p_0c_{in}c_2 = g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0c_{in}c_3 = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0 + p_3p_2p_1p_0c_{in}$

We can further simplify the design by nothing that the sum equation for stage I,

$$Z_i = x_i \text{ XOR } y_i \text{ XOR } c_{i-1}$$

is equivalent to

$$Z_i = p_i \text{ XOR } g_i \text{ XOR } c_{i-1}$$

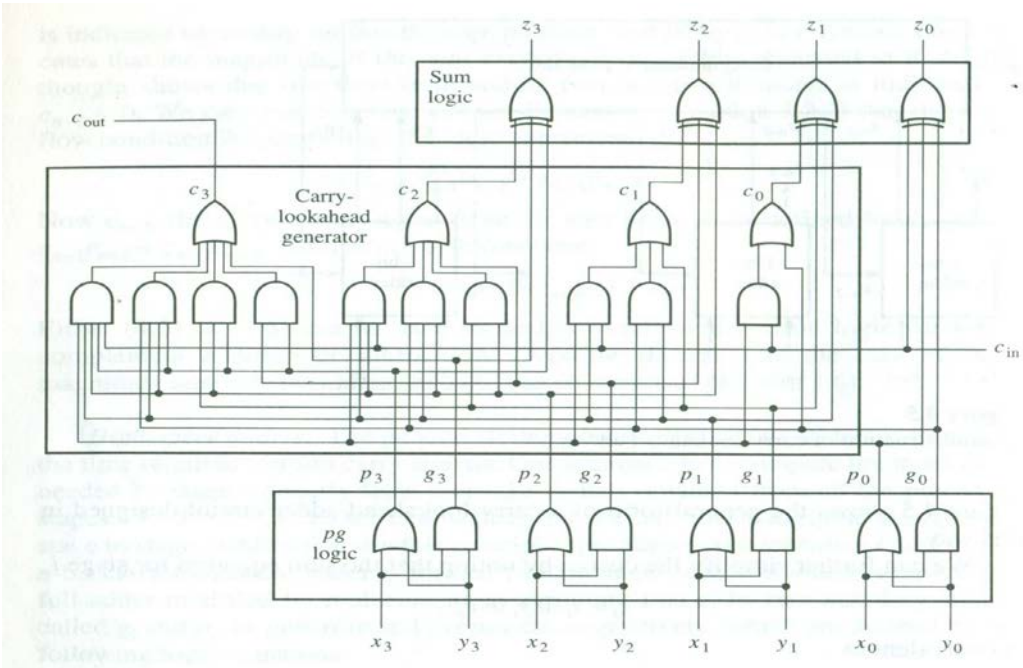
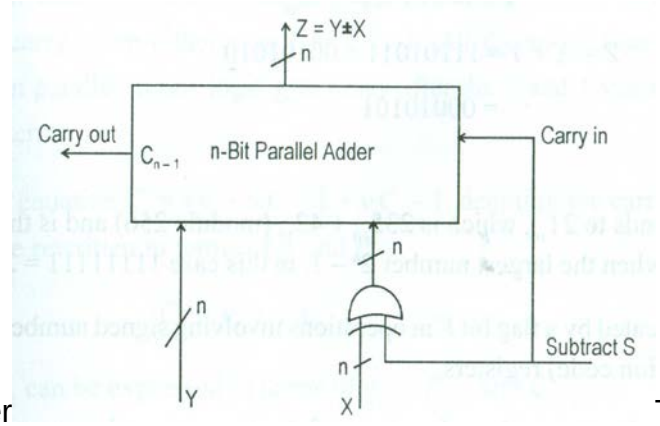


Figure.2.11. 4-bit carry look ahead adder

The 4-bit carry-lookahead adder has four levels of logic gates, so the adder's maximum delay is 4d, where d is the average gate delayThis delay is independent of any input n as long as carry generation is defined by two level logicHowever number of gates grows in proportion to n2 as n increasesIn contrast, the number of gates in a two level adder of sum products type grows exponentially with n**Subtracters**The best way to add or subtract numbers which have 1 as the sign bit depends on the number code in useAdding –X to Y is equivalent to subtracting X from Y, so the ability to add negative numbers implies the ability to do subtractionSubtraction is relatively simple with 2's complement code because negation is very easy to implementIf $X = x_{n-1}x_{n-2}\ldots x_0$ is a 2's complement integer, then negation is realized by $-X = x'_{n-1}x'_{n-2}\ldots x'_0 + 1$, ---------- **(1)**where + denotes addition modular one way to obtain the 1's complement portion. $X' = x'_{n-1}x'_{n-2}\ldots x'_0$ of –X in (1) uses the word-based EXCLUSIVE OR function X **XOR** s with a control variable SWhen S = 0, X **XOR** S = X, but when S X **XOR** S = X'Suppose that Y and X **XOR** S are now applied to the inputs of an n-bit adder. This addition of 1 required by (1) to the carry input



line of the adder The control lines selects the addition operation Y + X when S = 0 and thsubtraction operationY – X = Y + X' + 1, when S = 1Thus extending a parallel adder to perform 2's complement subtraction as well as addition merely requires connecting n 2-input EXCLUSIVE OR gates to the adder's input, these gates are represented by a single n-bit word gate**Example**X = 11101011 and Y = 00101000Denoting -21_{10} and 40_{10} respectively, in 2's complement codeZ = X + Y = 1110101011 + 00101000= 00010011Which corresponds to $-21_{10} + 40_{10} = +19_{10}$

3.Briefly explain about the process of Binary Multiplication with example. (or) Explain the sequential version of multiplication algorithm and its hardware. (April/May-15)

Multiplication

Integer multiplication is more complex than addition. It takes more time than addition as multiplication depends on bit pattern of the operands (A* B). There are intelligent algorithms which further shorten the time by shortcut techniques. The length of the product is 2n bits when the multiplier and multiplicand are each n bits The following are four basic rules for binary digits multiplication

1	0×0=0
2	0×1=0
3	1×0=0
4	1×1=1

Example :

+7 0 0 111 0 0 1 1 1

(-) (+)

-3 1 1 1 0 1 → 2 s C 0 0 0 1 1

---

0 0 1 0 0 (+4) ---

--- 0 1 0 1 0 (+10)

Over view:

Binary Multiplication

ooth's multiplication algorithm

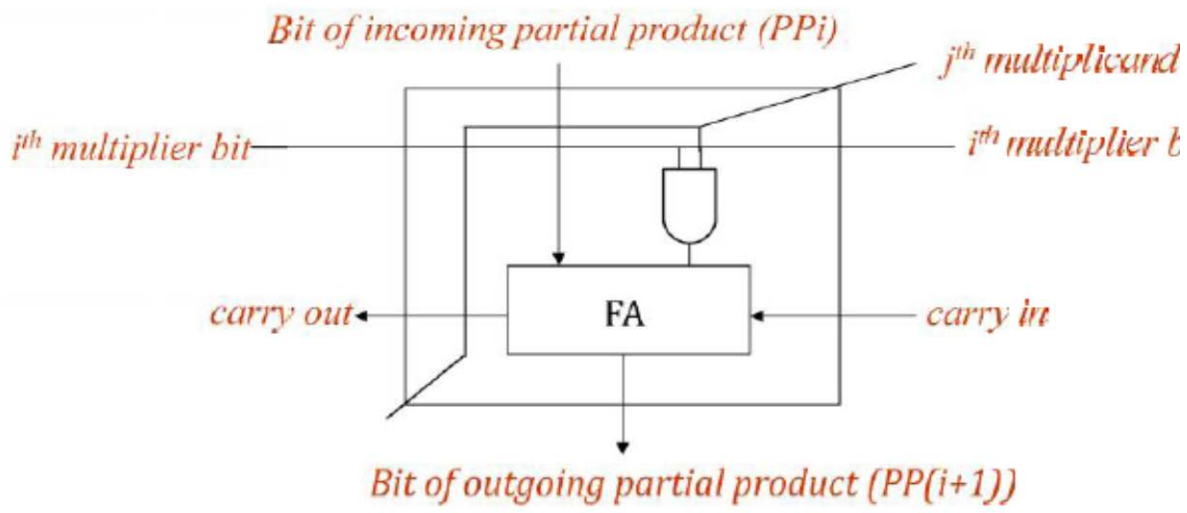
Fast multiplication

Binary Multiplication

Unsigned multiplication can be viewed as addition of shifted versions of the multiplicand.Alternative would be to add the partial products at each stage. If the *ith* bit of the multiplier is 1, shift the multiplicand and add the shifted multiplicand to the current value of the partial product. Hand over the partial product to the next stage. Value of the partial product at the start stage is 0.
Example:

```
    1011      Multiplicand (11)
  ×1101       Multiplier (13)
  ------
    1011    }
   0000     }  Partial products
  1011      }
 1011       }
---------
10001111    Product (143)
```

Figure 2.12. Multiplication of Unsigned Binary Integers



Typical multiplication cell

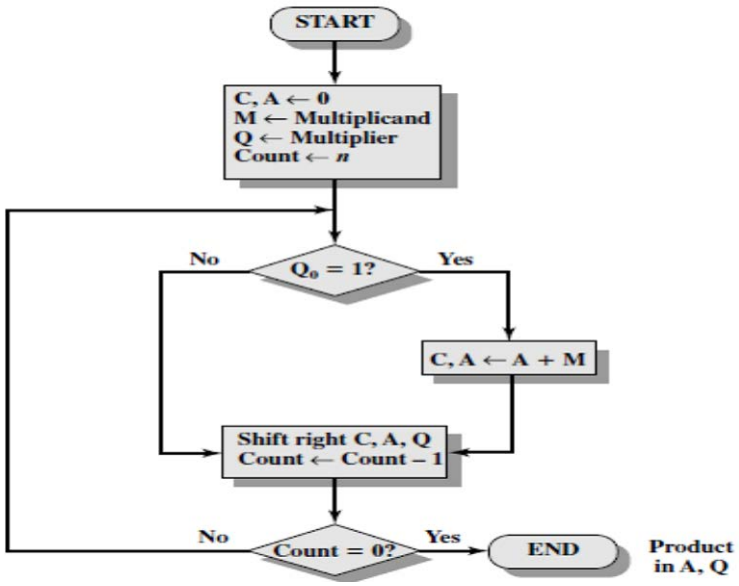
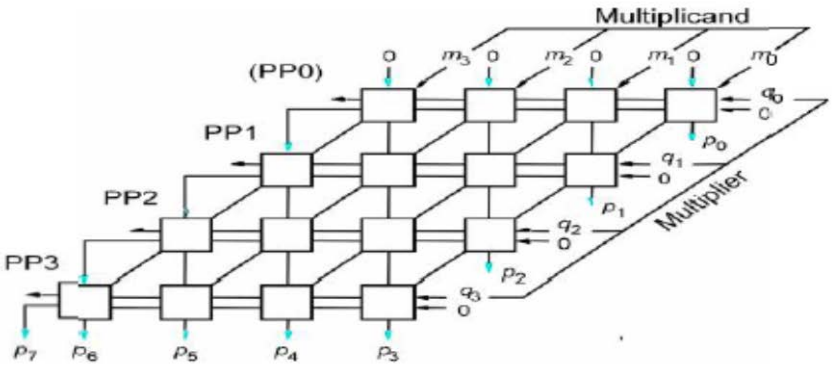


Figure 2.13. Flowchart for Unsigned Binary Multiplication

There are two types of Unsigned multiplication
Combinatorial array multipliers
Sequential multiplication
Combinatorial array multipliers are:
Extremely inefficient.Have a high gate count for multiplying numbers of practical size such as 32-bit or 64-bitnumbers.Perform only one function, namely, unsigned integer product.Improve gate efficiency by using a mixture of combinatorial array techniques and sequential techniques requiring less combinational logic.



Array Implementation

2. Sequential multiplication: (Apr/May-2015)

The simplest way to perform multiplication is to use the adder circuitry in the ALU for a number of sequential steps.The block diagram shown **in figure 3.4.** shows the arrangement for sequential multiplicationRegister A and Q combined to hold PPi (Partial Product), while

multiplier bit qi generates the signal .Add/No add. This signal controls the addition of M to PPi. The product is generated in n cycles.The carry-out of the adder is saved in flip-flop C. At the start the multiplier is loaded into the register Q and multiplicand into register M.Register A and Q are cleared to 0. At the end of each cycle the C, A and Q are shifted by one bit position to allow the growth of partial product as the multiplier is shifted out of register Q.

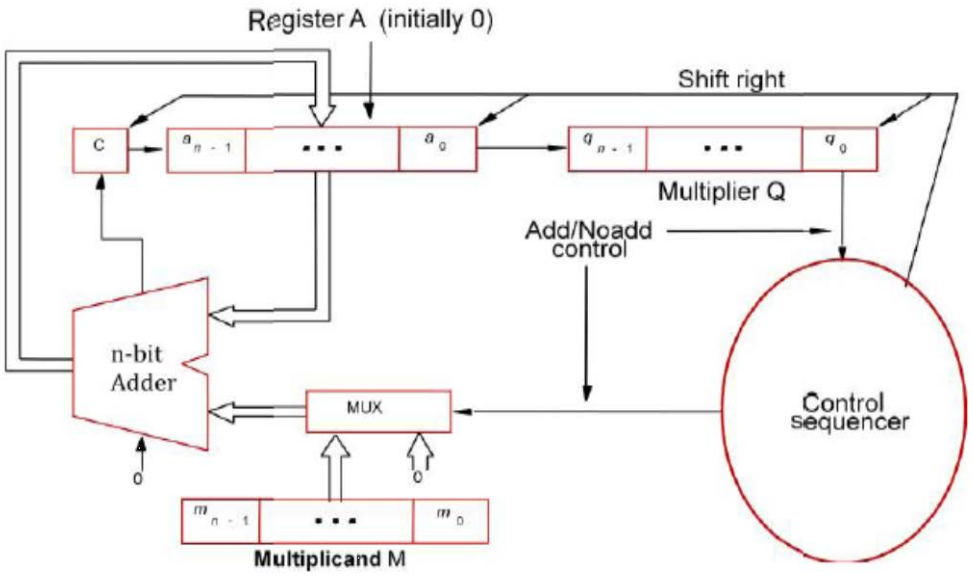


Figure 2.13. a Sequential Multiplication

Because of shifting the multiplier bit qi appears at the LSB position to generate Add/Noadd signal. After n cycles the higher order half of the product are held in register A and low order half in register Q.

The multiplication example is shown **in figure 3.5.**

- **Example :Multiplicand (11) – 1 0 1 1 Multiplier (13) - 1 1 0 1**

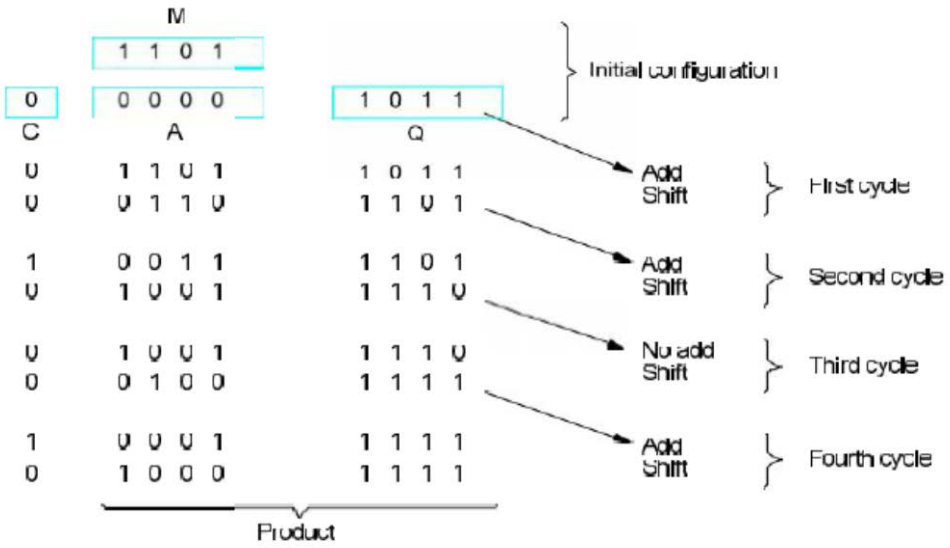


Figure 2.14 Multiplication example

4. Explain about Booth's multiplication algorithm rules with example.

Booth's multiplication algorithm

- It is a multiplication algorithm that multiplies **two signed** binary **numbers in** two's complement notation.
- The algorithm was invented by Andrew Donald Booth **in 1950** while doing research on crystallography at Birkbeck College in Bloomsbury, London.
- Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed.
- Booth's algorithm is of interest in the study of computer architecture.
- If two operands are unsigned numbers multiplication is easy. But any of the number is negative normal multiplication will not work. Example of signed number multiplication is given below

Example

```
    1011  Multiplicand (decimal-5
x   1101  Multiplier   (decimal-3)
    1011
   0000
  1011
 1011
10001111                (decimal-113)
```

The two possible solutions areSolution 1 Convert to positive if requiredMultiply as above If signs of both operands were different, take the 2's complement of the result.**Solution 2** Booth's algorithm

Booth algorithm uses the following principle

$2^n+2^{n-1}+\ldots\ldots\ldots\ldots 2^{n-k}= 2^{n+1}-2^{n-k}$

So the product can be generated one addition and one subtraction of the multiplicand. This scheme extends to any number of blocks of **1s** in the multiplier.Eg:, M*(01111010)=M*(+1 0 0 0 -1 +1 -1 0)In general the Booth scheme, when moving from 0 to 1 then -1 is selected and moving from 1 to 0 then +1 is selected as the multiplier scanning from left to right. This algorithm clearly extends to any number of blocks of 1s in a multiplier, including the situation in which a single 1 is considered as a block. If the first bit is 1 then consider the previous bit is 0.

Advantages of Booths algorithm

Treats positive and negative numbers uniformly.

String of 1's and 0's can be skipped with shift operation for faster execution.

Booth recording table

Multiplier	Multiplicand

Bit i	Bit i-1	Multiplicand Selected
0	0	0*M
0	1	+1*M
1	0	-1*M
1	1	0*M

Fig .2.19.Booth recording table

Example:
Let the multiplicand A=110011 and multiplier B=101100. Multiplier recoded using Booth's algorithm is –1 +1 0 –1 0 0
1 1 0 0 1 1
-1 +1 0 -1 0 0

ALGORITHM:
The Multiplicand is placed in M register and multiplier is loaded into register Q. Registers A and Q_{-1} is cleared initially.A bit of multiplier is examied together with the bit in Q_{-1}. If these bits are same (0-0,1-1) then all bits of A,Q,Q_{-1} are shifted right 1 bit.If two bits are differ then the multiplicand is added to or subtracted from A depending on 0-1 or 1-0 then right shift occurs. In either case the right shift A_{n-1} to A_{n-2} occurs and A_{n-1} maintains the same bit for maintaining the sign. This is called **Arithmetic shift**.

FLOWCHART

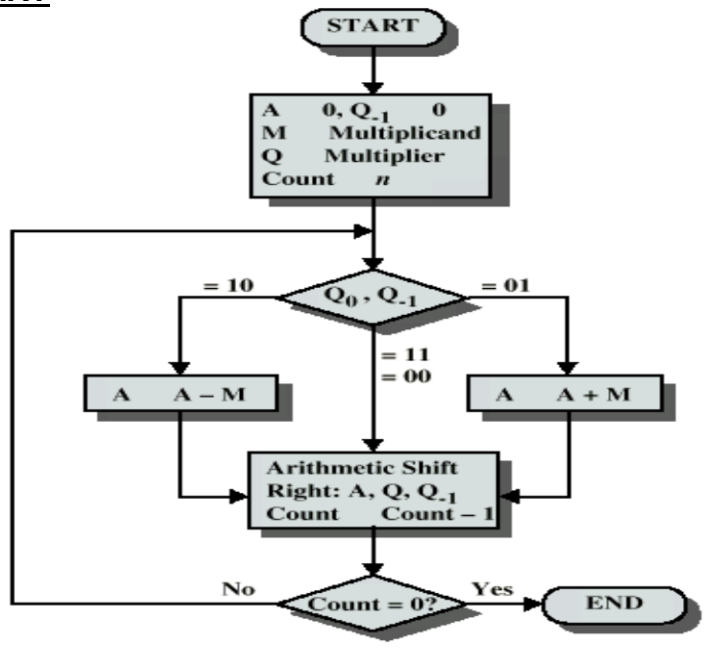


Figure 2.20.. Booth's Algorithm for Twos Complement Multiplication

Figure. Example of Booth's Algorithm (7*3=21)

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	First Cycle
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	Third Cycle
0001	0101	0	0111	Shift	Fourth Cycle

0111		
× 0011	(0)	
11111001	1–0	
0000000	1–1	
000111	0–1	
00010101	(21)	

(a) $(7) \times (3) = (21)$

0111		
× 1101	(0)	
11111001	1–0	
0000111	0–1	
111001	1–0	
11101011	(−21)	

(b) $(7) \times (-3) = (-21)$

1001		
× 0011	(0)	
00000111	1–0	
0000000	1–1	
111001	0–1	
11101011	(−21)	

(c) $(-7) \times (3) = (-21)$

1001		
× 1101	(0)	
00000111	1–0	
1111001	0–1	
000111	1–0	
00010101	(21)	

(d) $(-7) \times (-3) = (21)$

Figure 2.21. Examples Using Booth's Algorithm

5.Give detail description about Fast Multiplication or Bit pair recoding of multipliers with example. (Nov/Dec-2014)

Fast Multiplication

There are mainly two techniques used for speeding up the multiplication process.
The first technique reduces the maximum number of summands are added.
The second technique reduces the time needed to add all summands.

BIT –Pair recoding of multipliers:

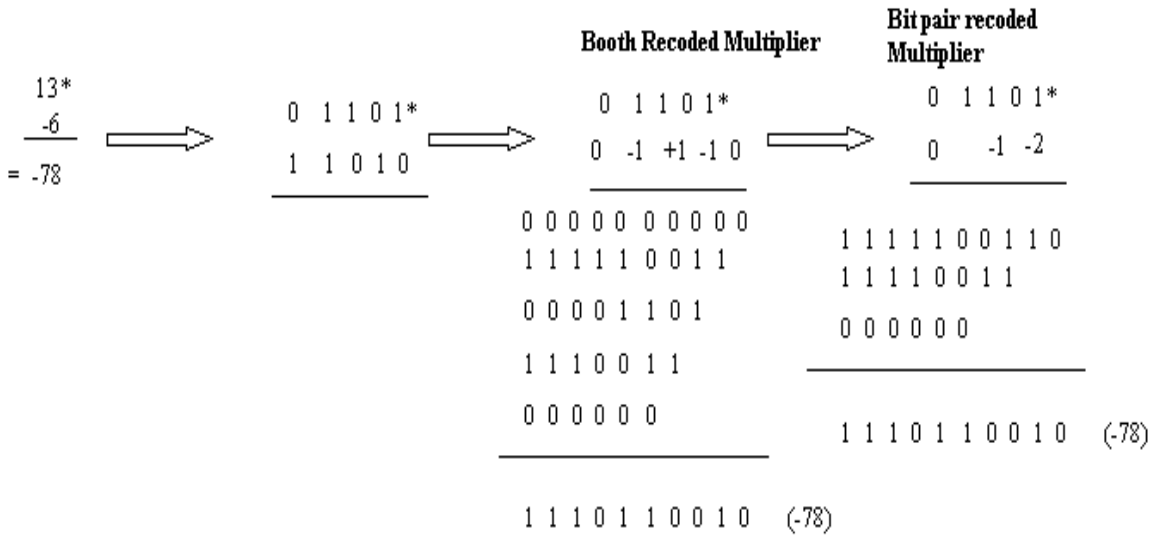
In this technique halves the maximum number of summands.It is directly derived from the Booth's algorithm.Group the Booth recoded multiplier bit in pairs starting from the right. Table of multiplicand selection decisions is given below.

Multiplier Bit pair		Multiplier bit on the Right	Multiplicand selected at i
i+1	i	i-1	
0	0	0	0*M
0	0	1	+1*M
0	1	0	+1*M

0	**1**	**1**	**+2*M**
1	**0**	**0**	**-2*M**
1	**0**	**1**	**-1*M**
1	**1**	**0**	**-1*M**
1	**1**	**1**	**0*M**

Fig.2.22. BIT –Pair recoding of multipliers

Example☹Nov/Dec-2014)



6.Explain about Division method with Longhand Method (Unsigned Division).

Division:

The division is more complex arithmetic than mortification. Given a dividend D and a divisor V, the division of D by V should satisfy the following equation.

D = (Q * V) + R. Where

Q->is the quotient

R->is the remainder which may be zero

Over View:

Longhand Method (Unsigned Division) Restoring Non - Restoring division:signed Division Longhand Method: (Unsigned Division)

Position the divisor appropriately with respect to the dividend and performs a subtraction.

If the remainder is zero or positive, a quotient bit of 1 is determined, the remainder is extended by another bit of the dividend, the divisor is repositioned, and another subtraction

is performed. If the remainder is negative, a quotient bit of 0 is determined, the dividend is restored by adding back the divisor, and the divisor is repositioned for another subtraction

Example

- Consider the following example of unsigned number division.

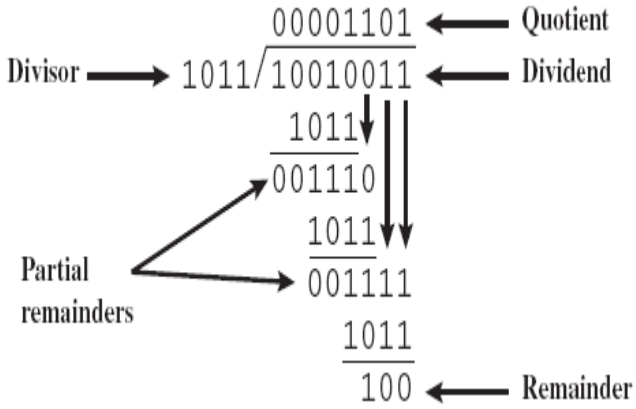


Figure 2.23Manual division method

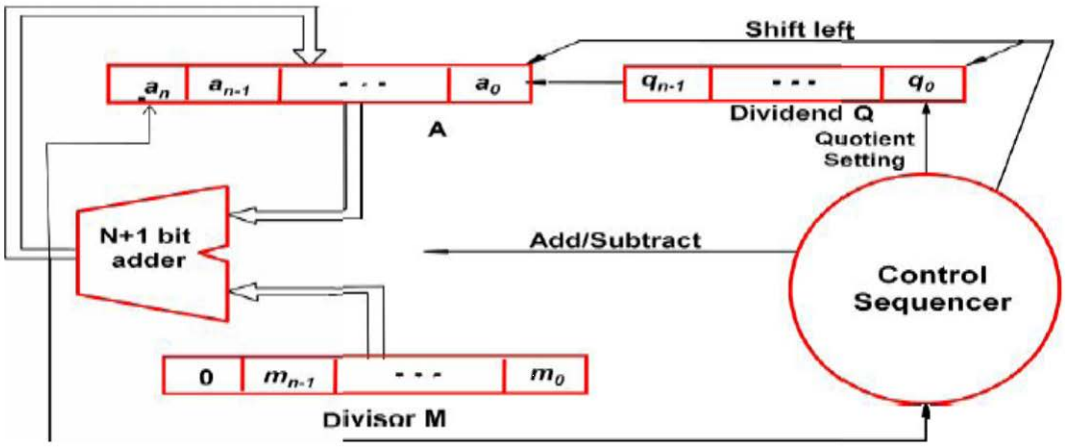


Figure 2.24. for Binary division

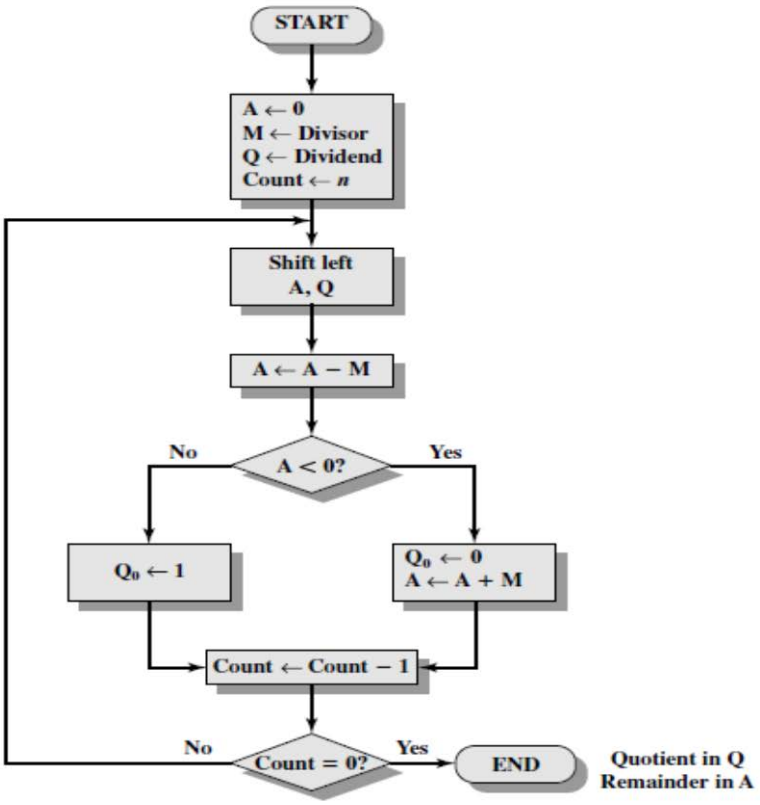


Figure 2.25. **Flowchart for Unsigned Binary Division**

7.Explain How to do Restoring Division with Algorithm and example.

Restoring

Register A is initially loaded with 0 and it consists of n+1 bits, where n is the number of bits in the dividend.
Dividend is loaded in register Q and Register M is loaded with the divisor.
After division is complete n bit quotient is in register Q and remainder is in register A
Extra bit on A and M accommodates the sign bit during subtractions.

Algorithm

Do the following n times
Shift A and Q left one binary position Subtract M from A, and place the answer back in A
If the sign of A is 1, set q0 to 0 and add M back to A (restore A); otherwise, set q0 to 1 and
Repeat these steps *n* times

Example: 1000/11
A= 000000, M=00011, Q=1000

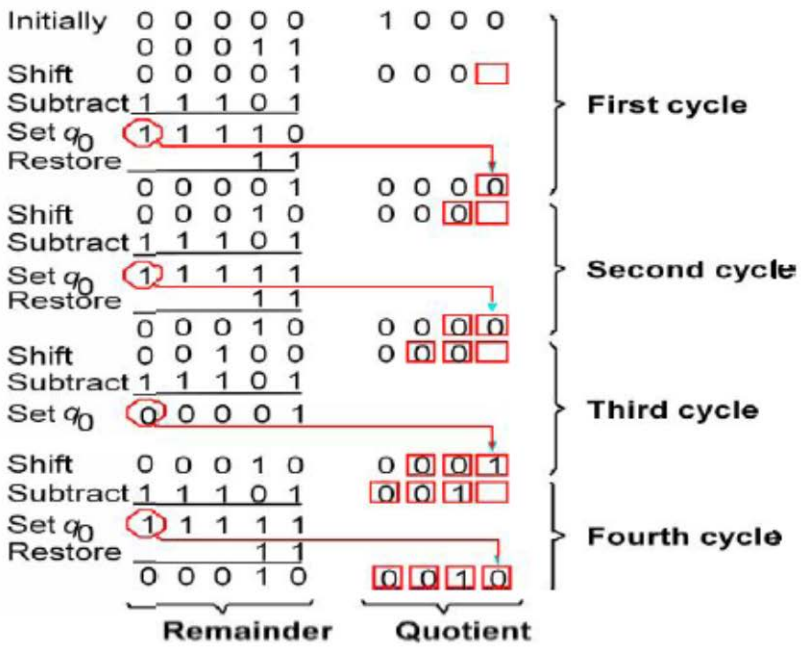


Figure 2.26. Example for Restoring division

8. Explain How to solve Non - Restoring division Algorithm with example.

Non - Restoring division:
In this restoring can be avoided.
Algorithm:
Step 1: Do the following n times
a) If the sign of A is 0 , then shift A &Q left one bit and subtract M from A, otherwise shift A &Q left one bit and add M to(b) If the sign of A is 0 then set q_0 to 1, otherwise set q_0 to 0.
Step2: If the sign of A is 1 then add M to A.Step 2 is needed to leave the proper positive remainder in A at the end of n cycles of step 1
Example: 1000/11

A= 000000, M=00011, Q=1000

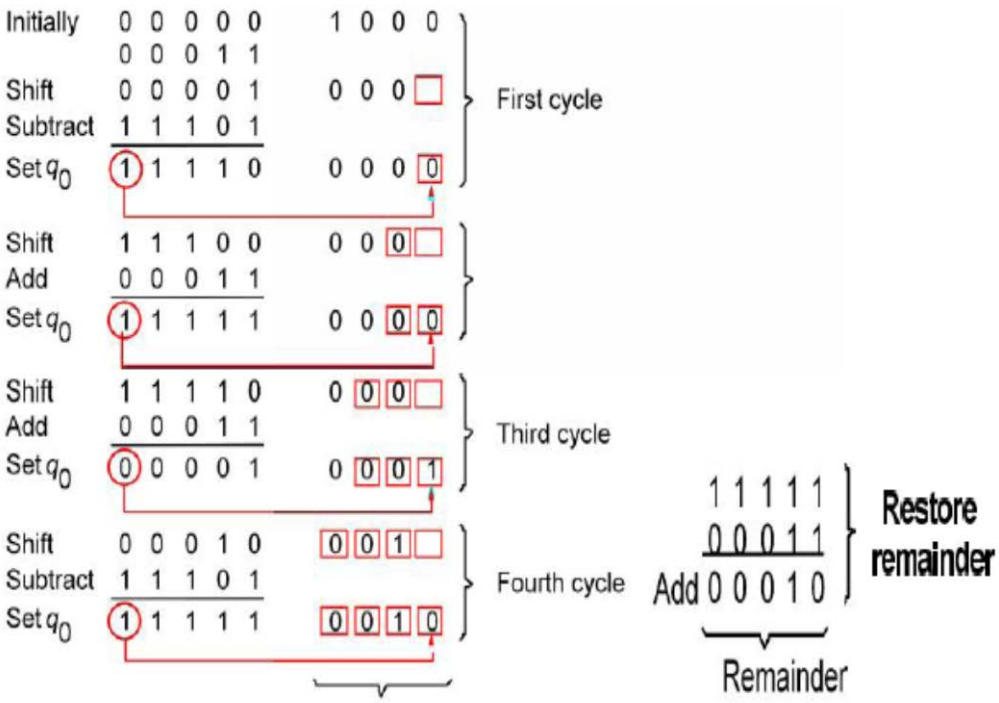


Figure 2.27. Example for Non- Restoring division

9.Explain about Signed Division Algorithm With example.

Algorithm:

Step 1: Load divisor into M register and dividend into Q register.

Step 2: Shift A, Q left 1 bit position.

Step 3: If M and A have
same signs, perform A ← A - M
otherwise, A ← A + M

Step 4: Preceding operation is successful if sign of A is same before and after operation. If operation is successful or A=0, then Q0←1.If operation is unsuccessful and 0,then Q0←0 and restore previous value of A.Step 5:. Repeat steps 2 through 4 as many times as there are bit positions in Q.Step 6: Remainder is in A.If sign of divisor and dividend are same, quotient is in Q.Otherwise, correct quotient is twos complement of Q.

Example: 7/ (-3)

A	Q	M=1101
0000	0111	Initial value
0000	111-	Shift
1101	1110	Add/set q_0
0000		Restore A
0001	110-	Shift

1110	1100	Add/set q_0
0001		Restore A
0011	100-	Shift
0000	1001	Add/set q_0
0001	001-	Shift
1110	0010	Add/set q_0
0001		Restore A

Quotient=0010 in Register Q
Remainder is in register A=0001
Quotient=0010 in Register Q but the sign of divisor and dividend are different so the actual quotient is 2's complement of (Q) = 1110

10.Explain about Floating Point Operations with example.(or) Explain how floating point addition is carried out in a computer system. Give an example for a binary floating point addition.(April/May-15)

Floating Point operations:They are four type of operations
Addition
Subtraction
Multiplication
Division
Over view
Floating Point Numbers representation:Arithmetic operations on floating Point Numbers
Guard Bits and Truncation **Floating Point Numbers representation:**Floating point numbers based on the scientific notation and is capable of representing very large and very small numbers without increase the number of bits and also used to represent the numbers has both fractional and integer part (real numbers).
For example, 123.456 could be represented as 1.23456×10^2. In hexadecimal, the number 123.abc might be represented as $1.23abc \times 16^2$. General form is **$M*2^E$IEEE standard for floating point representation:**IEEE Standard 754 floating point is the most common representation today for real numbers on computers.IEEE floating point numbers have three basic components: the **sign, the exponent, and the mantissa**. **Mantissa** is also known as the **significan*t*** is the magnitude of the number. **Exponent** is the number of places the decimal point (binary point) is to be moved.**Sign bit** denotes the sign of the number. 0 denotes a positive number and 1 denotes a negative number.**In IEEE standard there are three forms to represent the floating point numbers.** Single precision.,Double precision
Single Precision
The standard of 32-bit floating point representation has been developed and specified by the Institute of Electrical and Electronics Engineering(IEEE)
This standard describes both the representation and the way in which the four basic arithmetic operations are to be performed. This can be represented as
←----------32 bits----------→

Maximum value represented is ± 1.M X $2^{E'-127}$

The sign of the number is given in the first bit, followed by a representation for the exponent of scale factor.Instead of the signed exponent E, the value actually stored in the exponent field is an unsigned integer E' = E+127. This is called the excess-127 format.This E' is in the range 0 ≤ E' ≤ 255The end values of this range, 0 and 255, are used to represent special values. Therefore the range of E' for normal values is 1 ≤ E' ≤ 254. This means that the actual exponent E, is in the range -126 ≤ E ≤ 127The last 23 bits represent the mantissa. Since binary normalization is used, the MSB of the mantissa is always equal to 1. This bit is not explicitly represented; it is assumed to be to the immediate left of the binary point.Hence, the 23 bits actually represent the fractional part, that is, the bits to the right of the binary point

Example: 1.001010…0 X 2^{-87}

0	00101000	001010…….

This is called single-precision representation because it occupies a single 32 bit word. The scale factor has a range of 2^{-126} to 2^{+127}. The 24 bit mantissa provides approximately the same precision as a 7-digit decimal value

Example(Single precision)

To sum the follows given(116.52)10

Solution:

(116.52)10 **Integer part**

2 116
2 58 -0
2 29 -0
2 14- 1
2 7-0
2 3-1
(1110100)2

Fractional part

0.52
0.52*2=1.04
0.04*2=0.08
0.08*2=1.06
0.06*2=1.02
(.1011)2
(116.52)10=(1110100.1011)2

$(1.1101001011)*2^6$

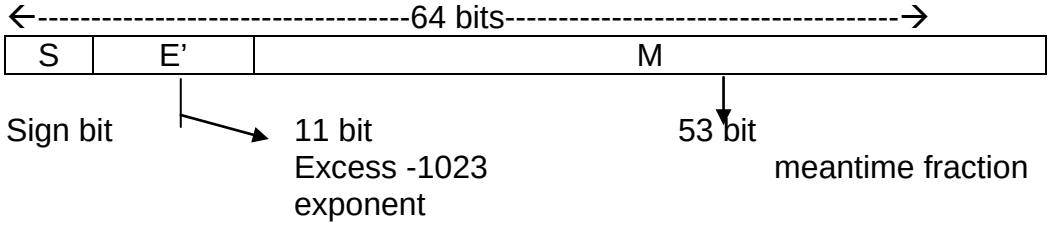
E=6
E^1=E+127
=6+127
E^1=(133)10

2 133 -
2 66-1
2 33-0
2 16-1
2 8-0
2 4-0
2 2-0
2 1-0

E1=10000101
0 10000101 1101001011

Double Precision

To provide more precision and range for floating-point numbers, the IEEE standard also specifies a double-precision format as,



The double precision format has increased exponent and mantissa ranges
The 11-bit excess 1023 exponent E' has the range 1 ≤ E' ≤ 2046 for normal values, with 0 and 2047 used to indicate special values. Thus the actual exponent E is in the range –1022 ≤ E ≤ 1023, providing scale factors of 2^{-1022} to 2^{1023}. The 53-bit mantissa provides a precision equivalent to about 16 decimal digits.**Maximum value represented is** ± 1.M X $2^{E'-1023}$Note: A computer must provide at least single-precision representation to conform to the IEEE standard.**Example(Double Precision)**

To sum of the follows given(116.52)10

Integer Part

2 116-0
2 58-0
2 29-0
2 4-0
2 7-0
2 3-1
2 1-1 **1110100**

Fractional Part

0.52
0.52*2=1.04
0.04*2=0.08
0.08*2=1.06
0.06*2=1.02

Exceptions

If a number has the exponent value less than -126 we say underflow has occurred
If a number has the exponent value greater than +127 we say overflow occurred.

ARITHMETIC OPERATIONS OF FLOATING POINT NUMBERS(NOV-DEC-15)

In floating point operations it is assumed that each floating point number has a mantissa in signed magnitude representation and heased exponent.If the exponents differ, the mantissa of floating point numbers must be shifted with respect to each other before they are Added,SubtractedThere are four basic phases of the algorithm for addition and subtraction:

1. Check for zeros.
2. Align the significands.
3. Add or subtract the significands.
4. Normalize the result.

Add/Subtract Rule

Choose the number with the smallest exponent and shift its mantissa right n times where n is the difference between the exponents.Set the exponent of the result as that of the larger exponent.Perform addition or subtraction on the mantissa and determine the sign of the result.Normalize the resulting value if necessary.

Addition and subtraction

Consider two floating point numbers
$X=M1 . r^{e1}$
$Y=M2 .r^{e2}$ assume e1≥e2
Let us see the rules for addition and subtraction **Step 1:** select the number with a smaller exponent and shift its mantissa right , a number of steps equal to the difference in xponents (e2 – e1).Eg ; if the numbers are$4.75 * 10^2$ $6.8 * 10^4$
Then, the number$1.75 * 10^2$ is selected and converted to $0.0175*10^4$
Step 2: set the exponent of the result equal to the larger exponent **Step 3:** perform addition subtraction on the mantissas and determine the sign of the result**Step 4:** normalize the result, if necessary let us perform the addition and subtraction of single precision umbers using above rules,Example : add single precision floating point numbers A and B, where
A=44900000H
B=42A00000H

Solution:

Step 1 : represent numbers in single precision format
A=44900000H
A=01000100100100000
B=42A00000H
B=01000010101000000
A=0 1000 1001 00100000
B=0 1000 0101 01000000

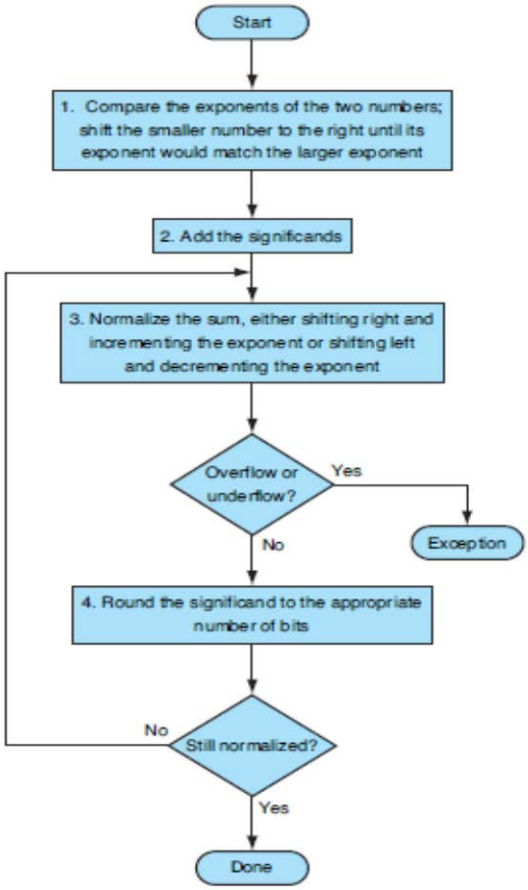
Exponent for A= 1000 1001

E^1 =10001001
E^1 =137
E^1 =E+127
137-127=E

E=10

B=10000101
B=133

E1 =E+127
133=E+12
E=133-127
E=6
A=10
M=00100000 =A
B=6
M=01000000 =B
Number B has smaller exponent with difference 4 1-1 ence its mantissa is shifted right by 4 bits.
B= M= 00000100
Step II: shift mantissa B=00000100000
Step III: add mantissa
Mantissa of A =00100000
Mantissa of B = 00000100
Mantissa of = 00100100
Result As both numbers are positive sign of the result is positive.Result = 0 10001001 00100100**Step 4:**Result =449200004We have seen addition process for floating point numbers similar process is used for subtraction of floating point numbers.**SUBTRACTION:** In subtraction two mantissa are subtracted instead of addition and the sign of greater mantissa is assigned to the result.Subtract single point precision floating point numbers A and B (A-B) where
A=4490000001-1
B=42A000001-1
SOLUTION:
Step I: represent numbers in single precision format
A=0 10001001 00100000
B=0 1000010 101000000
EXPONENT FOR A=10001001
1 *27 + 1*23 +1
128+8+1 = 137ACTUAL exponent
$A=E^1$=1000 1001 =137E^1=E+127
E=137-127
E=10
B=10000101 1*27+122+1*20=128+4+1=133**E=6**
Number B is has smaller exponent with difference 4 1-1 ence mantissa is shifted right by 4 bits as shown belwo**Step II:** shift mantissaShifted mantissa of B =00000100000
Step III: subtract mantissa
Mantissa of A=00100000Mantissa of B=00000100Result of mantissa = 00011100
Mantissa for A is greater than mantissa for B.Sign of result is sign of AResult = 01000 1001 00011100
Result=448E 0000H
FLOWCHART:



Exceptions

If a number has the exponent value less than -126 we say underflow has occurred.

If a number has the exponent value greater than +127 we say overflow has occurred. Such conditions are called exceptions.**Exception flag is set if**

Exponent overflow
Exponent underflow
Significant underflow
Significant overflow

Exponent overflow: A positive exponent exceeds the maximum possible exponent value. In some systems, this may be designated as or**Exponent underflow:** A negative exponent is less than the minimum possible exponent value (e.g., is less than). This means that the number is too small to be represented, and it may be reported as 0.**Significant underflow:** In the process of aligning significant, digits may flow off the right end of the significant. As we shall discuss, some form of rounding is required.**Significant overflow:** The addition of two significant of the same sign may result in a carry out of the most significant bit. This can be fixed by realignment, as we shall explain.

Advantages

Contains only positive numbers

Simple to compare their relative magnitudes without being concerned with their sign.

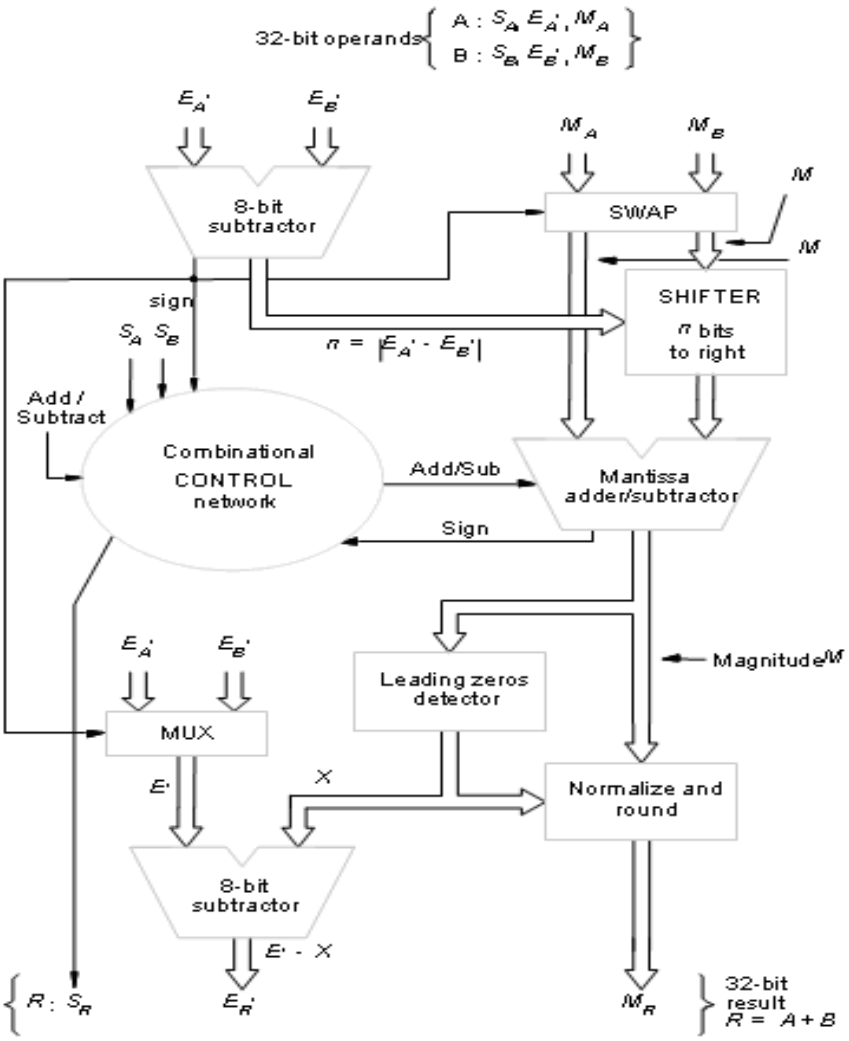


Figure.2.32. The hardware circuit for Floating point addition- subtraction unit.

Multiply Rule

If any exponent is zero then result=0.

Else add the exponents and subtract 127.

Multiply the mantissas and determine the sign of the result.

Normalize & round the resulting value if necessary.

Example:

$0.3 * 10^{2} *$

$0.2* 10^{3}$

$(0.3*0.2) * 10^{2+3} = 0.06*10^{5}$

Flow Chart

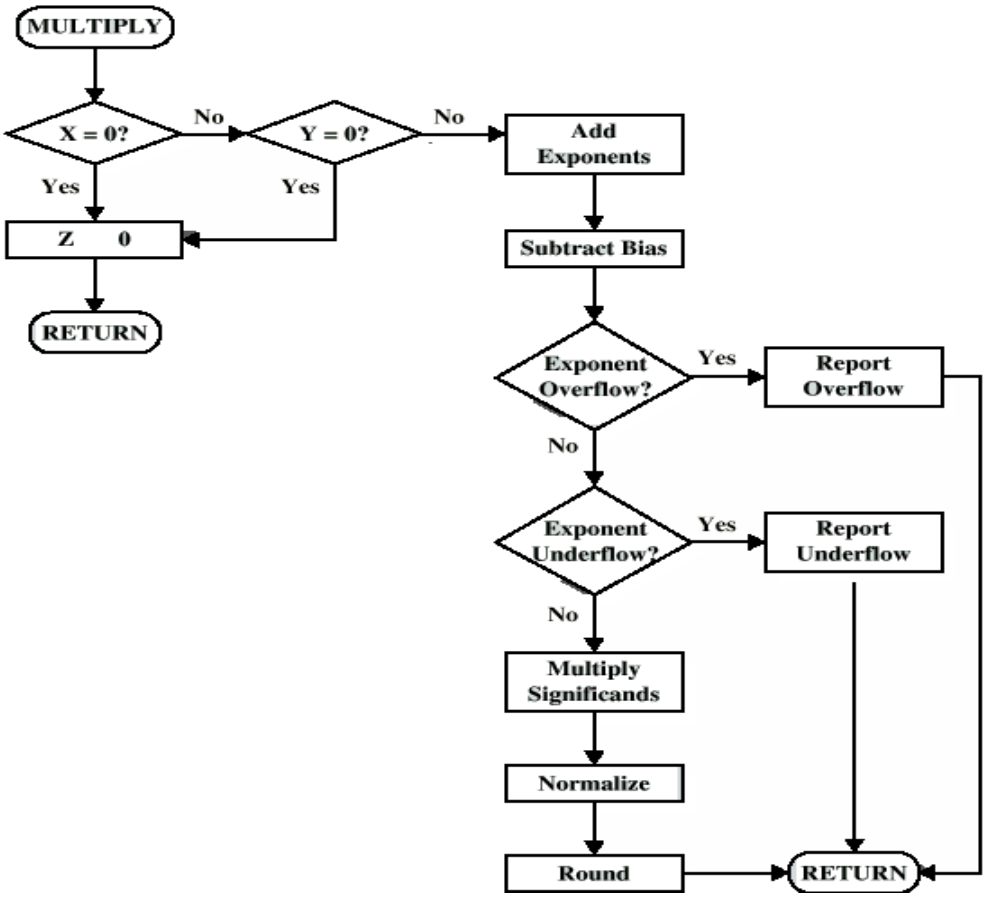


Figure 2.33. Floating-Point Multiplication (Z; X * Y)

Divide Rule

Check for zero if divisor is 0 then divide error Else dividend is 0 then result is 0.Else subtract the exponents and add 127.Divide the mantissas and determine the sign of the result.Normalize and round the resulting value if necessary.

Example: $0.3 * 10^{2} *$

$0.2 * 10^{3}$

$(0.3*0.2) * 10^{2-3} = 1.5*10^{-1}$

Flow Chart

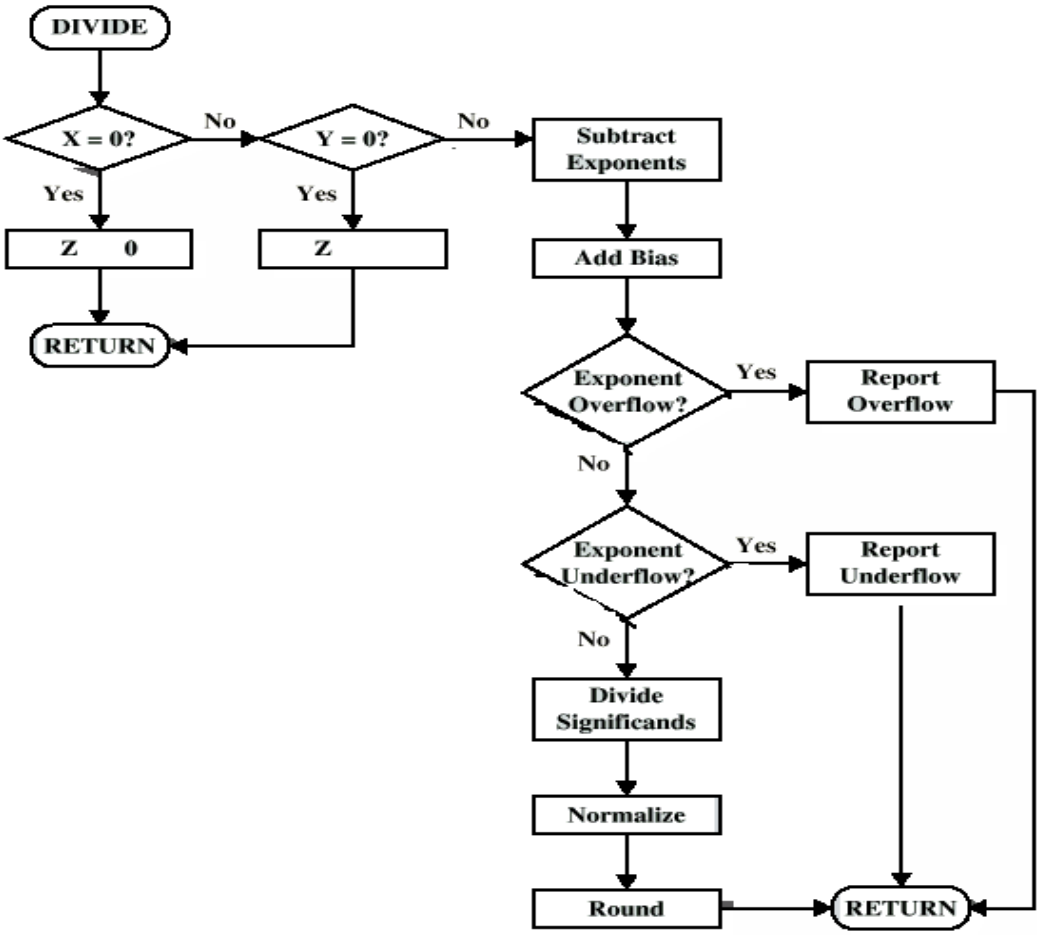


Figure 2.34 Floating-Point Division (Z; X_Y)

Precision Considerations

Guard Bits and Truncation

In 32 bit single precision floating point representation the mantissa bits are limited to 24 bits including implicit leading 1. But some operations may ***result in extra bits called guard bits*** *and these bits should be retained during the intermediate steps to increase the accuracy in final results*

Truncation

Similarly, allowing guard bits during intermediate steps results in extended mantissa. Thus this extended mantissa should be truncated to 24 bits while generating final results. There are several ways to truncate

Chopping

Von-Neumann rounding

Rounding

Chopping: Chopping is the simplest way to do truncation (i.e) **Remove the guard bits** and make no changes in the retained bits**Example**

0. b_{-1} b_{-2} b_{-3} 000

0. b_{-1} b_{-2} b_{-3} 111 and truncated to 0. b_{-1} b_{-2} b_{-}

Error in chopping ranges from 0 to almost , and result in biased approximation of b_{-3} position.**Von-Neumann rounding**In this method, if the bits to be removed are all 0' s, they are simply dropped , with no changes to the retained bits.If any of the bits to be removed are 1, the least significant bit of the retained but isset to 1.

Example

$0.\ b_{-1}\ b_{-2}\ b_{-3}$ 000 $0.\ b_{-1}\ b_{-2}\ b_{-3}$
$0.\ b_{-1}\ b_{-2}\ b_{-3}$ 100 $0.\ b_{-1}\ b_{-2}$ 1 LSB to 1
Error in this technique is larger than chopping and the approximation is unbiased and hence results in high probability of accuracy.**Rounding:**Rounding achieves the cloest approximation to the number being truncated and is an unbiased technique.
A 1 is added to the LSB position of the bits to be retained if there is a 1 in the MSB position of the bits being removed.**Example**Thus $0.\ b_{-1}\ b_{-2}\ b_{-3}$ 1 ... is rounded to $0.\ b_{-1}\ b_{-2}\ b_{-3}$ + 0.0001 $0.\ b_{-1}\ b_{-2}\ b_{-3}$ 0 1 ... is rounded to $0.\ b_{-1}\ b_{-2}\ b_{-3}$ o … ... is rounded to $0.\ b_{-1}\ b_{-2}\ b_{-3}$ … this provides the desired approximation

11.Give detail description about Subword parallelism.

Subword parallelism

Parallelism will improve the performance of computer. Many application needs the performance high compared to small applications.For example consider the graphical and multimedia applications.Every desktop microprocessor has its own graphical displays, support the graphics operations many transistors are used. Many graphics systems used 8 bits to represent each of the three primary colors plus 8 bits for a location of a pixel. The primary colors are RGB-Red,Green,Blue and it has to be represent using 8 bits.For teleconference and video games application it has tospeakers and microphones in addition to that we need to provide some other extra features.In such application we use audio samples that, have to be represented using 8 bits.For audio samples 8 bits are not sufficient to represent it and it requires more than 8 bits but 16 bits are sufficient.Every microprocessor has special support so to store the bytes and halfwords it requires only less space from the memory.Many graphics and audio applications are perform the same operation again and again.So far such kinds of application we need to speed up the process using a technique called parallelism.Parallelism used to perform simultaneous operations on vectors of data used in specific applicationsParallelism performs simultaneous operations short vectors of following values: Sixteen 8 bit operands
Eight 16 bit operands
Four 32 bit operands
Two 64 bit operands
The parallelism must occur within a wide word ,in case it extends means that is called as subword parallelism.Subword parallelism also known as data level parallelism or vector or SIMD parallelism.Multimedia application requires arithmetic instructions to support narrower operationNarrow operations must be operated easily in parallel manner.For example,

DATA TARNSFER	ARITHMETIC	LOGICAL/COMPARE
VLDR,F32	VADD,F32,VADD{L,W} {S8,U8,S16,U16,S32,U32}	VAND.64,VAND.128
VSTR,F32	VSUB,F32,VSUB(L,W} {S8,U8,S16,U16,S32,U32}	VORR.64,VORR.128
VLD {1,2,3,4}.{18, 116,132}	VMUL,F32,VMULL {S8,U8, S16,U16,S32,U32}	VERR.64,VERR.128
VST {1,2,3,4}. {18,116,132}	VMLA,F32,VMLAL {S8,U8, S16,U16,S32,U32}	VBIC.64,VBIC.128
VMOV.{18,116,132,	VMLS,F32,VMLSL {S8,U8,	VORN.64,VORN.128

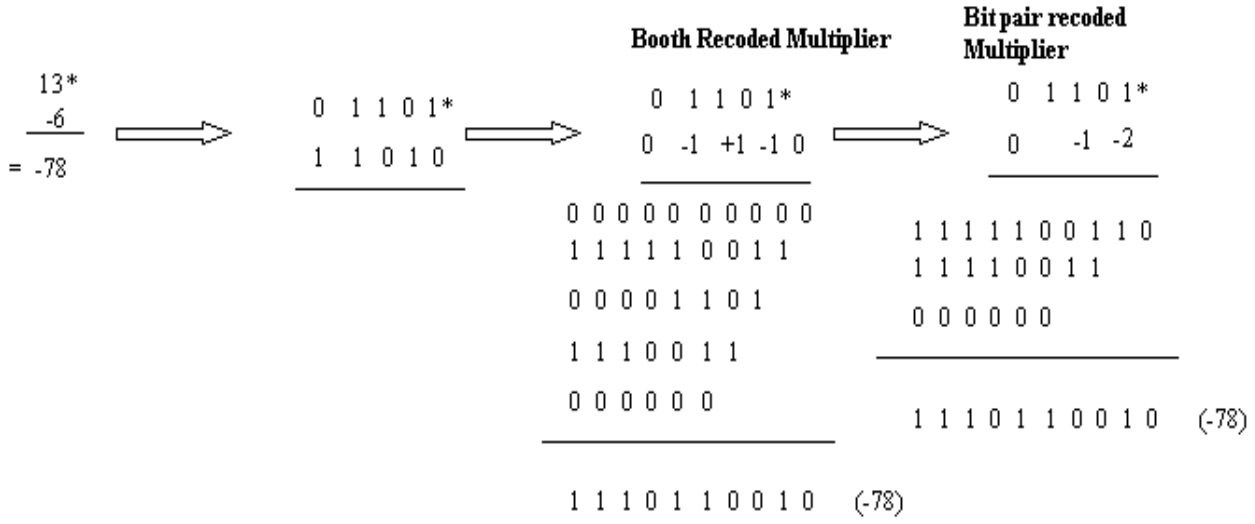
F32},#imm	S16, U16,S32,U32}	
VMVN.{18,116,132, F32},#imm	VMAX {S8,U8,S16,U16, S32,U32}	VCEQ{18,116,132,F32}
VMOV.{164,1128}	VMIB {S8,U8,S16,U16,S32,U32}	VCGE{S8,U8,S16,U16,S32,U 32,F32}
VMVN.{164,1128}	VABS{S8,S16,S32,F32}	VCGT{S8,U8,S16,U16,S32,U 32,F32}
	VNEG{S8,S16,S32,F32}	VCLE{S8,U8,S16,U16,S32 ,U32,F32}
	VSHL{S8,U8,S16,U16,S32, U32,S64,U64}	VCLT{S8,U8,S16,U16,S32,U3 2,F32}
	VSHR{S8,U8,S16,U16,S3 2,U32,S64,U64}	VTST{8,16,32}

32 bit floating point numbers

Table 2.35. basic NEON instructions

{} used to show optional variations of the basic operations.{S8,U8,8} stands for signed and unsigned 16 bit intergers. {S16,U16,16} stands for signed and unsigned 16 bit intergers or 16 bit type lessdata{S32,U32,32} stands for signed and unsigned 16 bit intergers or 16 bit type less data.{F32} stands for signed and unsigned 32 bit floating point numbers of which 4 fit in a 128 bit register.Vector load reads one n element structure from memory into 1,2,3 or 4 NEON registers.It loads a single n-element structure to one lane.If element of the register are not loaded means,that is known as unchanged.Vector store writes one n-element structure into memory from 1,2,3 or4 NEON registers.

12.Multiply the following pair of signed nos. using booth's bit – pair recoding of the multiplie. A = +13 (Multiplicand) and B = -6 (Multiplier). (Nov/Dec 2014)



Q13. Divide (12) $_{10}$ by (3) $_{10}$ using the restoring and Non restoring division algorithm withstepbystepintermediateresultandexplain. (Nov/Dec 2014)

Restoring

Register A is initially loaded with 0 and it consists of n+1 bits, where n is the number of bits in the dividendDividend is loaded in register Q and Register M is loaded with the divisor. After division is complete n bit quotient is in register Q and remainder is in register A Extra bit on A and M accommodates the sign bit during subtractions. **Algorithm**

Do the following n times

Shift A and Q left one binary position

Subtract M from A, and place the answer back in A

If the sign of A is 1, set q0 to 0 and add M back to A (restore A); otherwise, set q0 to

Repeat these steps *n* times

Dividend – 12 binary value 1100 (Q)

Divisior-3 binary value 0011 (M)

Subtract M 2's Complement 1100

1 ---

Subtract M 1 1101

	A	Q	
initially	00000	1100	
Shift left	00001	100	
Sub M	11101		First Cycle
Set q_0	11 1 10	100	
Restore M	0 0 0 11		
	000001		
Shift left	000011	00	
Sub M	11101		Second Cycle
Set q_0	00 0 00	00	
Shift left	00000	0	
Sub M	11101		Third Cycle
Set q_0	11101	0	
Restore M	0 0 0 11		
	000000		
Shift left	00000		
Sub M	11101		Forth Cycle
Set q_0	11101		
Restore	0 0 0 11		
	000000		Quotient

Remainder

Non - Restoring division:

In this restoring can be avoided.

Algorithm:

Step 1: Do the following n timesa) If the sign of A is 0 , then shift A &Q left one bit and subtract M from A, otherwise shift A &Q left one bit and add M to Ab) If the sign of A is 0 then set q_0 to 1, otherwise set q_0 to 0.Step2: If the sign of A is 1 then add M to A.

Step 2 is needed to leave the proper positive remainder in A at the end of n cycles of step

1.**Example**

Dividend – 12 binary value 1100 (Q)

Divisior-3 binary value 0011 (M)

Subtract M 2's Complement 1100

1

Subtract M 1 1101

	A	Q	
Initially	00000	1100	
Shift left	00001	100	
Sub M	11101		First Cycle
Set q_0	11 1 10	100	
Shift left	11101	00	
Add M	00011		Second Cycle
Set q_0	00 0 00	0	
Shift left	00000	0	
Sub M	11101		Third Cycle
Set q_0	11101	0	
Shift left	11010		
Add M	00011		Fourth Cycle
Set q_0	11101		
Add M	00011		
	00000		
	Remainder		

Unit-III

Processor and Control Unit

Part-A

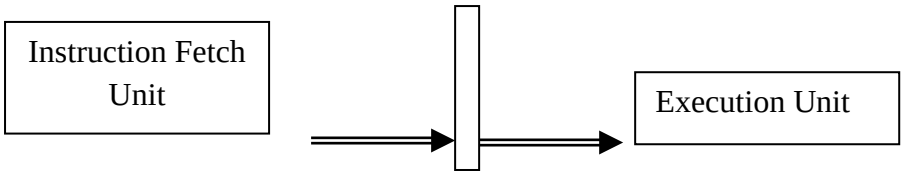
1.What is pipelining and what are the advantages of pipelining? (Apr/May-2010)

Pipelining is process of handlining the instruction concurrently .the pipelining processor executes a program by one after another **Fetch**: read the instruction from the memory. **Decode**: decode the instruction and fetch the source operand. **Execute**: perform the operation specified by the instruction **Write**: store the result in the destination location.

Advantages: Improve the overall throughput of an instruction set processor

Applied to design of complex data path units such as multiplexers and floating points adders.

2. Draw the hardware organization of two-stage pipeline?



Inter stage buffer

3. Name the four stages of pipelining.? (Nov/Dec2007) (or) What are the steps in pipelining processor.

Fetch	:	Read the instruction from the memory.
Decode	:	Decode the instruction and fetch the source operands.
Execute	:	Perform the operation specified by the instruction
Write	:	Store the result in the destination location.

4. Write short notes on instruction pipeline.

The various cycles involved in the instruction cycle. These fetch ,decode and execute cycles for several instructions are performed simultaneously to reduce overall processing time. This process is referred as **instruction pipelining.**

5. What is the role of cache in pipelining?(Nov/Dec-2011)What is the need to use the cache memory in pipelining concept?

Each stage in a pipeline is expected to complete its operation in one clock cycle. But the accessing time of the main memory is high. So it will take more than one clock cycle to complete its operation. So we are using cache memory for pipelining concept. The accessing speed of the cache memory is very high.

6. State different types of hazards that occur in pipeline?(April/May'2003) (Nov/Dec'2004)

The various pipeline hazards are:

Structural hazards

Data or Data dependent hazards

Instruction or control hazards

7. What is data hazard in pipelining?(Nov/Dec 2007/2008)

A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in pipeline. As a result some operation has be delayed and the pipeline stalls. Arise when an instruction depends on the results of a previous instruction in a way that is exposed by overlapping of instruction in pipeline **Types** RAW,WAWandWAR

8. What are instruction hazards? Or control hazard?

They arise while pipelining branch and other instructions that change the contents of program counter. The simplest way to handle these hazards is to stall the pipeline stalling of the pipeline allows few instructions to proceed to completion while stopping the execution of those which results in hazards

9. What is meant by bubbles in pipeline?

Any condition that causes the pipeline to be is known as pipeline stall. This is also known as bubble in the pipeline. Once the bubble is created as a result of a delay, a bubble moves down stream until it reaches the last unit.

10. What is structural hazard? ? (Nov/Dec-2008) ,(Apr/May-2014)

When two instructions require the use of a given hardware resource at the same time this hazard will occur. The most common case of this hazard is memory access

.

11. How can we eliminate the delay in data hazard?

In pipelining the data can be executed after the completion of the fetch operation. The data are available at the output of the ALU once the execute stage completes.

Hence the delay can be reduced if we arrange for the result of fetch instruction to be forwarded directly for use in next step. This is known as operand forwarding.

12. How can we eliminate data hazard using software?

The data dependencies can be handled with the software. The compiler can be used for this purpose. The compiler can introduce the two cycle delay needed between instruction I1 and I2 by inserting NOP (no operation)

```
I1:   MUL   R2, R3, R4
      NOP
      NOP
I2:   ADD   R5, R4, R6
```

13. When will the instruction have sideeffect?

Some time an instruction changes the contents of a register other than the destination. An instruction that uses an auto increment or auto decrement addressing mode is san example. R2, R4This instruction will take the carry value present in the condition code register. So it refers the register which is not represented in the instruction

14. Define branch penalty.

The time lost as a result of a branch instruction is often referred to as the branch penalty. This will cause the pipeline to stall. So we can reduce branch penalty by calculating the branch address in early stage.

15. What is the use of instruction queue in pipeline?
Many processors can fetch the instruction before they are needed and put them in queue called instruction queue. This instruction queue can store several instructions.

16. Define dispatch unit.
It is manly used in pipeline concept. It takes the instruction from the front of the instruction queue and sends them to the execute unit for execution.

17. What is meant by branch folding and what is condition to implement it?
The instruction fetch unit has executed the branch instruction concurrently with in the execution of other instruction s. This occurs only if at the time of branch is encountered at least one instruction is available in the queue than the branch instruction.

18. What is meant by delay branch slot?
A location following branch instruction is called as branch delay slot. There may be more than one branch delay slot, depending on the execution time.
The instruction in the delay slot is always fetched and at least partially execution before the branch decision is made.

19. Define delayed branching.
It is a technique to handle the delay branch slot instructions. We can place some useful instruction in the branch delay slot and execute these instructions when the processor is executing the branch instruction. If there is no useful instruction in the program we can simply place NOP instruction in delay slot. This technique will minimize the branch penalty.

20. Define static and dynamic branch prediction.
The branch prediction decision is always the same every time a given instruction is executed. This is known as static branch prediction.
The prediction may change depending on execution history is called dynamic branch predction.

21. List the two states in the dynamic branch prediction
LT: Branch is likely to be taken.
LNT: Branch is likely not to be taken.

22.List out the four stages in the branch prediction algorithm.
ST :Strongly likely to be taken
LT :Likely to be taken
LNT:Likely not to be taken
SNT:Strongly not to be taken

23. Define Register renaming?(Nov/Dec -2009)

When temporary register holds the contents of the permanent register, the name of permanent register is given to that temporary register is called as **register renaming**.
For example, if I2 uses R4 as a destination register, then the temporary register used in step TW2 is also referred as R4 during cycles 6 and 7, that temporary register used only for instructions that follow I2 in program order.

24. What are the major characteristics of pipeline?

Pipelining cannot be implemented on a single task, as it works of splitting multiple tasks into a number of subtasks and operating on them simultaneously. The speedup or efficiency is achieved by using a pipeline depends on the number of pipe stages and the number of available tasks that can be subdivided.

25. What is precise exception? (Apr/May-2009)

A **precise exception** is one in which all instruction prior to the faulting instruction are complete and instruction following the instruction, including the faulting instruction do not change the state of the machine.

26. List the techniques used for overcoming hazard.

Data forwarding
Adding sufficient hardware
Stalling instructions
Document to find instruction in wrong order.

27. What are the techniques used to present control hazard?

Scheduling instruction in delay slots
Loop unrolling Conditional execution
Speculation(by both compiler and CPU).

28. Write down the expression for speedup factor in a pipelined architecture.

The speedup for pipeline computer is
$S=(K+n-1)t_p$
Where,
k-number of segments in a pipeline.
n-number of instruction to be executed.
t_p- cycle time.

29. List the types of data hazards.

i. RAW(Read After Write)
ii. WAW(Write After Write)
iii. WAR(Write After Read)
iv. RAR(Read After Read)

30. Define stall.

Idle periods are called stalls. They are also often referred to as bubbles in the pipeline.

31. Give 2 examples for instruction hazard.

● Cache miss

● Hazard in pipeline.

33. A =5 A<-3+A A<-4+A

32.What hazard does the above two instructions create when executed concurrently? (Apr/May-2011)
If these operations are performed in the order given , the result is 32 . but , if they were performed concurrently, the value is 5. So output is wrong.

33. What are the disadvantages of increasing the number of stages in pipelined processing?(APR/MAY-2011) (or)
What would be the effect, if we increase the number of pipelining stages?(Nov/Dec-2011)Speedup: Speedup is defined by**S(m)=T(1)/T (m)Where T (m)** is the execution time for some target workload on an m-stage pipeline and **T(1)** is the execution time for same target workload on a non pipelined Processor.

34. How can memory access be made faster in a pipelined operation? Which hazards can be reduced by faster memory access? (Apr/May 2010)
The goal in controlling a pipelined CPU is maximize its performance with respect to the target work loads**Performance measures:**The various performance measures of pipelining are**ThroughputCPISpeedupDependencies Hazards Hazards can be reduced by faster memory access?**
Structural hazardsData or Data dependent hazards, Instruction or control hazards.

35. List the key aspects in gaining the performance in pipelined systems (May/June 2009)
The goal in controlling a pipelined CPU is maximize its performance with respect to the target work loads
Performance measures:
The various performance measures of pipelining are
Throughput
CPI
Speedup
Dependencies
Hazards

36. What are the two types of branch prediction technique available? (MAY/JUNE2009)
The two types of branch prediction techniques are
Static branch prediction
Dynamic branch prediction

37.Distinguish between static and dynamic branch prediction(MAY/JUNE2009)

STATIC BRANCH PREDICTION	DYNAMIC BRANCH PREDICTION
Branch can be predicted based on branch codes type statistically	It used recent branch history during program execution, information is stored in buffer called branch target buffer(BTB)

It may not produce accurate result every time	Processing of conditional branches with zero delay

38. What is meant by hazard in pipelining? Define data and control hazards.(May/Jun-2013)(Apr/May-2012)

Hazards:

The idle periods in any of the pipeline stage due to the various dependency relationships among instructions are said to be stalls.**A data hazard** is any condition in which either the source or the destination operands of an instruction are not available at the time expected in pipeline. As a result some operation has be delayed and the pipeline stalls.Arise when an instruction depends on the results of a previous instruction in a way that is exposed by overlapping of instruction in pipeline **ypes**

RAW

WAW. WAR

Control hazards.

They arise while pipelining branch and other instructions that change the contents of program counter. The simplest way to handle these hazards is to stall the pipeline stalling of the pipeline allows few instructions to proceed to completion while stopping the execution of those which results in hazards

39. Why is branch prediction algorithm needed? Differentiate between the static and dynamic techniques.(May/Jun-2013)

ST :Strongly likely to be taken
LT :Likely to be taken
LNT :Likely not to be taken
SNT :Strongly not to be taken

STATIC BRANCH PREDICTION	DYNAMIC BRANCH PREDICTION
Branch can be predicted based on branch codes type statistically	It used recent branch history during program execution,information is stored in buffer called branch target buffer(BTB)
It may not produce accurate result every time	Processing of conditional branches with zero delay

40. What is meant by speculative execution? (Apr/May-2012) (or) What is the need for speculation? (Nov/Dec-2014)

A technique allows a superscalar processor to keep its functional units as busy as possible by executing instructions before it is known that they will be needed.The Intel P6 uses speculative execution. Compare branch prediction, speculative evaluation.

41. What is precise and imprecise exception? (Apr/May-2009)

If the execution occurs during an instruction,all subsequent instructions that may have been executed are discarded. This is called **precise exception.**If one instruction causes an exception and succeeding instructions are permitted to complete execution, then the processor is said to have **imprecise exception**

42. What is a Data path element?

A unit used to operate on or hold data within a processor. In the MIPS implementation, the data path elements include the instruction and data memories, the register file, the ALU, and adders.

43. What is a Sign-extend?
To increase the size of a data item by replicating the high-order sign bit of the original data item in the high order bits of the larger, destination data item.

44. Define Branch target address.
The address specified in a branch, which becomes the new program counter (PC) if the branch is taken. In the MIPS architecture the branch target is given by the sum of the offset field of the instruction and the address of the instruction following the branch.

45. Define Register file.
A state element that consists of a set of registers that can be read and written by supplying a register number to be accessed.

46. Define Forwarding.
This technique is called bypassing**.** A method of resolving a data hazard by retrieving the missing data element from internal buffers rather than waiting for it to arrive from programmer visible registers or memory.

47. What is a branch prediction buffer? (Apr/May-2015)
Also called branch **history table**. A small memory that is indexed by the lower portion of the address of the branch instruction and that contains one or more bits indicating whether the branch was recently taken or not.

48. Give the features of the addressing mode suitable for pipelining. (Apr/May-2014)

- They access operand from memory in only one cycle.
- Only load and store instruction are provided to access memory.
- The addressing modes used do not have side effects.
- Three basic addressing modes used do not have these features are register, register indirect and index. The first two require bus address computation. In the index mode, the address can be computed in one cycle, whether the index value is given in the instruction or in registration.

49. What are R-type instruction? (Apr/May-2015)

MIPS fields are given names to make them easier to discuss:

Here is the meaning of each name of the fields in MIPS instructions:

■ *op:* Basic operation of the instruction, traditionally called the **opcode**.
■ *rs:* The first register source operand.

■ *rt:* The second register source operand.
■ *rd:* The register destination operand. It gets the result of the operation.
■ *shamt:* Shift amount. ■ *funct:* Function. This field, often called the *function code,* selects the specific variant of the operation in the op field.

PART-B

Q1. Briefly explain about Basic MIPS Implementation:
A Basic MIPS Implementation:
The MIPS instruction set:(Micro Instruction per Second)
The memory-reference instructions load word (lw) and store word (sw)
The arithmetic-logical instructions add, sub, AND, OR, and slt
The instructions branch equal (beq) and jump (j), which we add lastThis subset does not include all the integer instructions (for example, shift, multiply, and divide are issuing), nor does it include any floating-point instructionHowever, the key principles used in creating a data path and designing the control are illustrated**An Overview of the Implementation**MIPS instructions, including the integer arithmetic-logical instructions, the memory-reference instructions, and the branch instructions. Much of what needs to be done to implement these instructions is the same, independent of the exact class of instruction.
For every instruction, the first two steps are identical: Send the program counter (PC) to the memory that contains the code and fetch the instruction from that memory.2. Read one or two registers, using fields of the instruction to select the registers to read. For the load word instruction, we need to read only one register, but most other instructions require that we read two registers'.After these two steps, the actions required to complete the instruction depend on the instruction class. .
For example,
All instruction classes, except jump, use the arithmetic-logical unit (ALU) after reading the registers.The memory-reference instructions use the ALU for an address calculation, the arithmetic-logical instructions for the operation execution, and branches for comparison.After using the ALU, the actions required to complete various instruction classes differ.A memory-reference instruction will need to access the memory either to read data for a load or write data for a store.An arithmetic-logical or load instruction must write the data from the ALU or memory back into a register. Lastly, for a branch instruction, we may need to change the next instruction address based on the comparison; otherwise, the PC should be incremented by 4 to get the address of the next instruction.

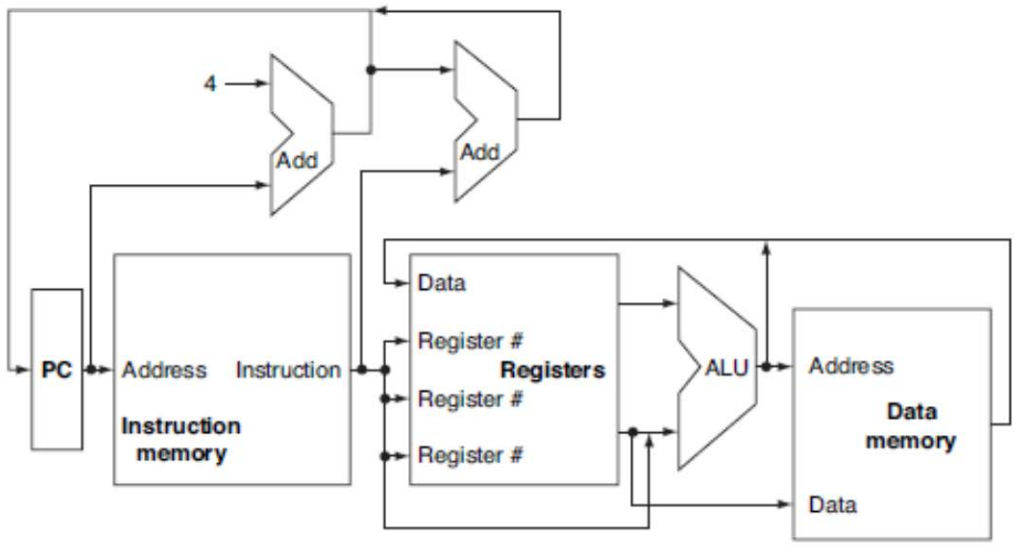


FIGURE 3.1 An abstract view of the implementation of the MIPS subset showing the Major functional units and the major connections between them

All instructions start by using the program counter to supply the instruction address to the instruction memory. After the instruction is fetched, the register operands used by an instruction are specified by fields of that instruction.Once the register operands have been fetched, they can be operated on to compute a memory address (for a load or store), to compute an arithmetic result (for an integer arithmetic-logical instruction), or a compare (for a branch).The result from the ALU or memory is written back into the register file. Branches require the use of the ALU output to determine the next instruction address, which comes either from the ALU (where the PC and branch offset are summed) or from an added that increments the current PC by 4. The thick lines interconnecting the functional units represent buses, which consist of multiple signals. The arrows are used to guide the reader in knowing how information flows. Since signal lines may cross, we explicitly show when crossing lines are connected by the presence of a dot where the lines cross.

Q2.Give detail description about Building a Data path.

3.2. Building a Data path:

Data path element: A unit used to operate on or hold data within a processor. In the MIPS implementation, the data path elements include the instruction and data memories, the register file, the ALU, and adders.

Figure 3.2.a shows the first element we need:

A memory unit to store the instructions of a program and supply instructions given an address. Figure 3.2.b also shows the **program counter (PC)**, is a register that holds the address of the current instruction.Figure 3.2.c Lastly, we will need an adder to increment the PC to the address of the next instruction.

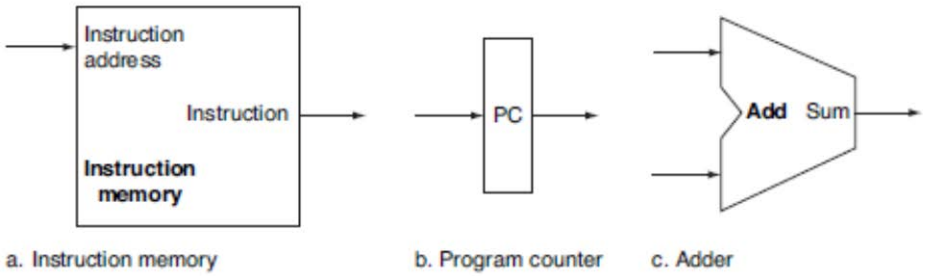


FIGURE 3.2. Two state elements are needed to store and access instructions, and an adder is needed to compute the next instruction address.

The state elements are the instruction memory and the program counter. The instruction memory need only provide read access because the datapath does not write instructions.

Since the instruction memory only reads, we treat it as combinational logic: the output at any time reflects the contents of the location specified by the address input, and no read control signal is needed. The program counter is a 32-bit register that is written at the end of every clock cycle and thus does not need a write control signal. The adder is an ALU wired to always add its two 32-bit inputs and place the sum on its output."Simply by wiring the control lines so that the control always specifies an add operation. We will draw such an ALU with the label *Add*, as in Figure 3.2.,To prepare for executing the next instruction, we must also increment the program counter. so that it points at the next instruction, 4 bytes later Figure 3.3.shows how to combine the three elements from Figure 3.2 to form a datapath that fetches instructions and increments the PC to obtain the address of the next sequential instruction.

R-format instructions:

They all read two registers, perform an ALU operation on the contents of the registers, and write the result to a register. ***These are called R-type instructions* or *arithmetic-logical instructions*** (since they perform arithmetic or logical operations). This instruction class includes **add, sub, AND, OR, and slt,** Recall that a typical instance of such an instruction is add $t1,$t2,$t3, which reads $t2 and $t3 and writes $t1.**Register file**:he processor's 32 general-purpose registers are stored in a structure called a **register file.** A register file is a **collection of registers** in which any register can be read or written by specifying the number of the register in the file. The register file contains the register state of the computer.

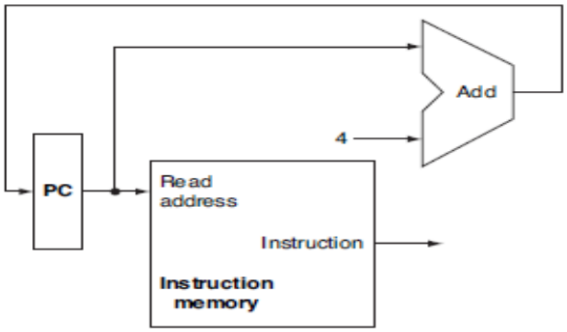


FIGURE 3.3 A portion of the datapath used for fetching instructions and incrementing the program counter. The fetched instruction is used by other parts of the datapath.

input to the register file that specifies the register number to be read and an output from the register file that will carry the value that has been read from the registers.

To write a data word, we will need two inputs:

One to specify the ***register number*** to be written and one to supply the ***data*** to be written into the register. The register file always outputs the contents of whatever register numbers are on the Read register inputs. Writes, however, are controlled by the write control signal, which must be asserted for a write to occur at the clock edge. Figure 3.4a shows the result; we need a total of four inputs (**three for register** numbers and one for **data)** and two outputs (both for data). The register number inputs are 5 bits wide to specify one of 32 registers (32 = 25), whereas the data input and two data output buses are each 32 bits wide.

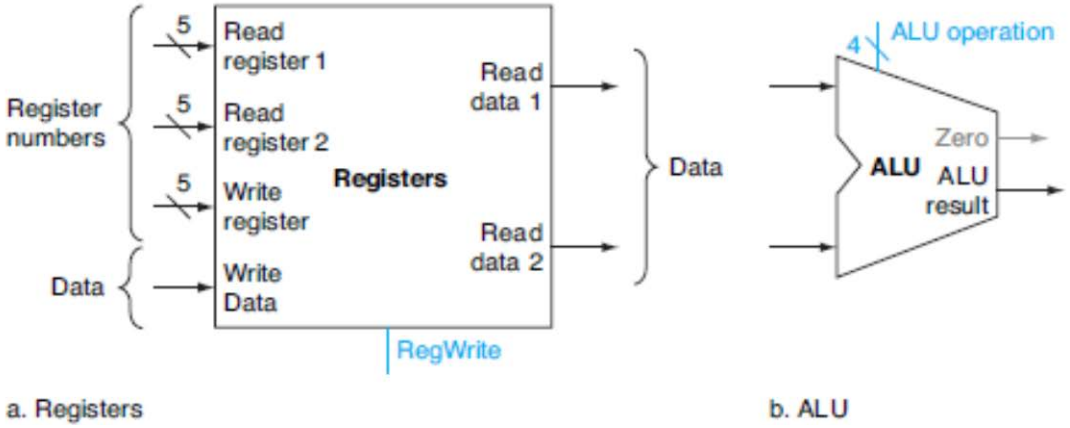


Figure 3.4. Register and ALU

- Figure 3.4b shows the ALU, which takes two 32-bit inputs and produces a 32-bit result, as well as a 1-bit signal if the result is 0. The 4-bit control signal of the ALU.
- **sign-extend** To increase the size of a data item by replicating the high-order sign bit of the original data item in the high order bits of the larger, destination data item.
- **Sign-extend** the 16-bit offset field in the instruction to a 32-bit signed value, and a data memory unit to read from or write to. The data memory must be written on store instructions; hence, data memory has read and write control signals, an address input, and an input for the data to be written into memory. Figure 3.5 shows these two elements.

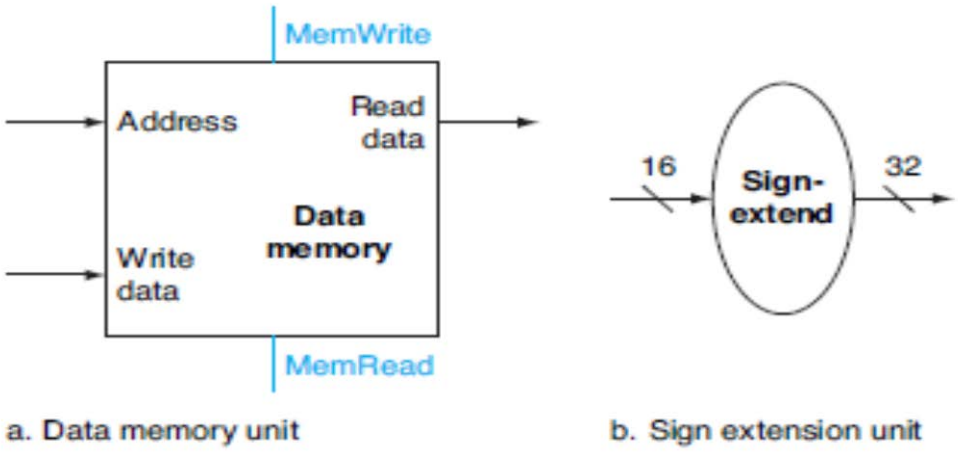


FIGURE 3.5. The two units needed to implement loads and stores, in addition to the register file and ALU

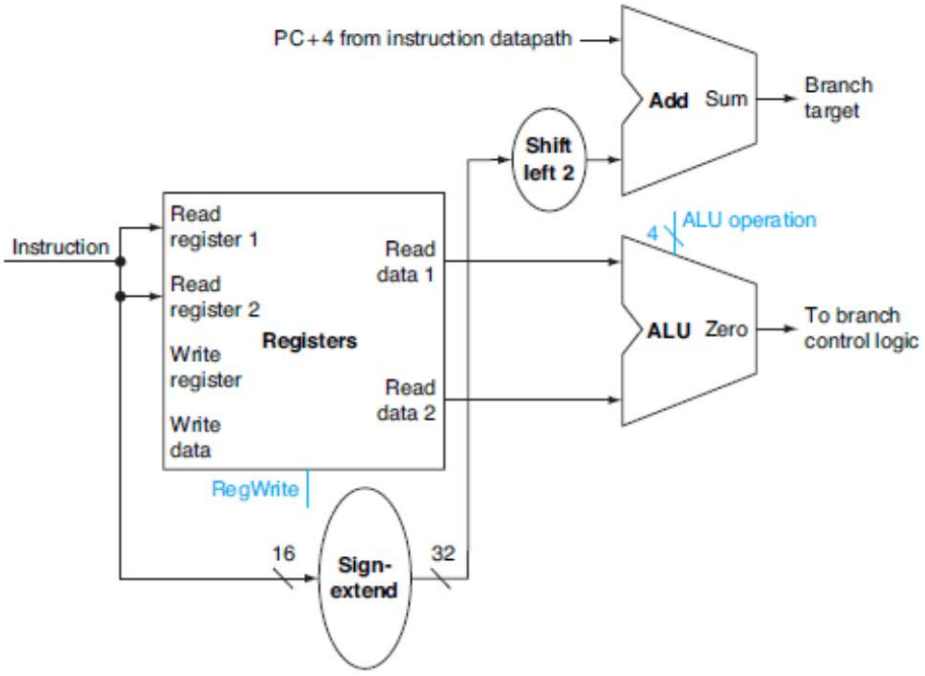
The beq instruction has three operands, two registers that are compared for equality, and a 16-bit offset used to compute the **branch target address** relative to the branch instruction address. Its form is beq $t1,$t2,offset. To implement this instruction, we must compute the branch target address by adding the sign-extended offset field of the instruction to the PC.

There are two details in the definition of branch instructions

The instruction set architecture specifies that the base for the branch address calculation is the address of the instruction following the branch. Since we compute PC + 4 (the address of the next instruction) in the instruction fetch datapath, it is easy to use this value as the base for computing the branch target addressThe architecture also states that the offset field is shifted left 2 bits so that it is a word offset; this shift increases the effective range of the offset field by a factor of 4.

To deal with the latter complication, we will need to shift the offset field by 2.

Branch taken. A branch where the branch condition **is satisfied** and the program counter (PC) becomes the branch target.All unconditional branches are taken branches.**ranch not taken or (untaken branch**) .A branch where the branch condition **is false** and the program counter (PC) becomes the address of the instruction that sequentially follows the



branch

FIGURE 3.6 The datapath for a branch uses the ALU to evaluate the branch condition and a separate adder to compute the branch target as the sum of the incremented PC and the sign-extended, lower 16 bits of the instruction (the branch displacement), shifted left 2 bits.

The unit labeled ***Shift left 2*** is simply a routing of the signals between input and output that adds 00_{two} to the low-order end of the sign-extended offset field; No actual shift hardware is needed, since the amount of the "shift" is constant Since we know that the offset was sign-extended from 16 bits, the shift will throw away only "**sign bits**." Control logic is used to decide whether the incremented PC or branch target should replace the PC, based on the Zero output of the ALU.

3.Briefly explain about Control Implementation scheme.

Control Implementation scheme:

It contains the ALu control, Designing the main control unit and operation of the Data path.

Over view:

- The ALU Control:
- Designing the Main Control Unit
- Operation of the Datapath:

The ALU Control:

The MIPS ALU defines the **6** following combinations **of four control inputs:**

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

Depending on the instruction class, the ALU will need to perform **one of these first five functions.** (NOR is needed for other parts of the MIPS instruction set not found in the subset we are implementing.)For load word and store word instructions, we use the ALU to compute the memory address by addition.For the R-type instructions, the ALU needs to perform one of the five actions (AND, OR, subtract, add, or set on less than), depending on the value of the 6-bit funct (or function) field in the low-order bits of the instruction.For branch equal, the ALU must perform a subtraction. We can generate the 4-bit ALU control input using a small control unit that has as inputs the function field of the instruction and a 2-bit control field, which we call ALUOp.LUOp indicates whether the operation to be performed should be add (00) for loads and stores, subtract (01) for beq, or determined by the operation encoded in the funct field (10). The output of the ALU control unit is a 4-bit signal that directly controls the ALU by generating one of the 4-bit combinations shown previously.In Figure 3.7 we show how to set the ALU control inputs based on the 2-bit ALUOp control and the 6-bit function code. Later in this chapter we will see how the ALUOp bits are generated from the main control unit.

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

FIGURE 3.7 How the ALU control bits are set depends on the ALUOp control bits and the different function codes for the R-type instruction.

The opcode, listed in the first column, determines the setting of the ALUOp bits. All the encodings are shown in binary.

Notice that when the ALUOp code is 00 or 01, the desired ALU action does not depend on the function code field; in this case, we say that we “don’t care” about the value of the function code, and the funct field is shown as XXXXXX. When the ALUOp value is 10, then the function code is used to set the ALU control input.here are several different ways to implement the mapping from the 2-bit ALUOp field and the 6-bit funct field to the four ALU operation control bits.Because only a small number of the 64 possible values of the function field are of interest and the function field is used only when the ALUOp bits equal 10, we can use a small piece of logic that recognizes the subset of possible values and causes the correct setting of the ALU control bits.As a step in designing this logic, it is useful to create a truth table for the interesting combinations of the function code field and the ALUOp bits, as we’ve done in Figure 3.8; This **truth table** shows how the 4-bit ALU control is set depending on these **two input fields.**

ALUOp		Funct field						
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
0	0	X	X	X	X	X	X	0010
0	1	X	X	X	X	X	X	0110
1	0	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	0	X	X	0	1	0	0	0000
1	0	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

FIGURE 3.8. The truth table for the 4 ALU control bits (called Operation).

The inputs are the **ALUOp** and **function code field**. Only the entries for which the ALU control is asserted are shown.Some don’t-care entries have been added. For example, the ALUOp does not use the encoding 11, so the truth table can contain entries 1X and X1, rather than 10 and 01. te that when the function field is used, the first 2 bits (F5 and F4) of these instructions are always 10, so they are **don’t-care terms** and are replaced with XX in the truth table.**Don’t-care term:** An element of a logical function in which the output does not depend on the values of all the inputs.

4. Briefly explain about How to Design the Main Control Unit

.

Designing the Main Control Unit

To understand how to connect the fields of an instruction to the datapath, it is useful to review the formats of the **three instruction classes:**

The R-type instruction classes,

Branch instruction classes, and

Load-store instruction classes

. Figure 3.9. Shows these formats.

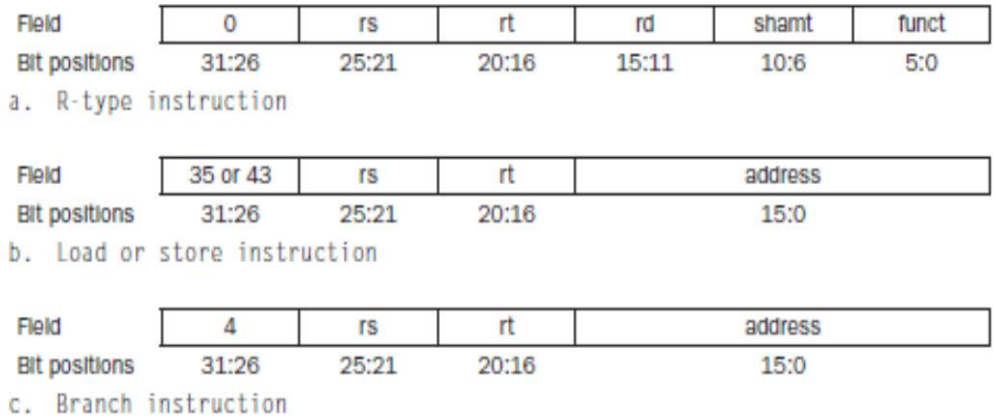


FIGURE 3.9. The three instruction classes (R-type, load and store, and branch) use two different instruction formats

Instruction format for R-format instructions, which all have an opcode of 0. These instructions have three register operands: rs, rt, and rd. Fields rs and rt are sources, and rd is the destination The ALU function is in the funct field and is decoded by the ALU control design in the previous section. The R-type instructions that we implement are add, sub, AND, OR, and slt.**(b).** Instruction format for load (opcode = 35ten) and store (opcode = 43ten) instructions. The register rs is the base register that is added to the 16-bit address field to form the memory address. For loads, rt is the destination register for the loaded value. For stores, rt is the source register whose value should be stored into memory. **(c).** Instruction format for branch equal (opcode = 4). The registers rs and rt are the source registers that are comparFed for equality. The 16-bit address field is sign-extended, shifted, and added to the PC+4 to compute the branch target address.There are several major observations about this instruction format that we will rely on:■ The op field, also called the **opcode**, is always contained in bits 31:26. We will refer to this field as Op[5:0].■ The two registers to be read are always specified by the rs and rt fields, at positions 25:21 and 20:16. This is true for the R-type instructions, branch equal, and store. The base register for load and store instructions is always in bit positions 25:21 (rs).■ The 16-bit offset for branch equal, load, and store is always in positions 15:0.■ The destination register is in one of two places. For a load it is in bit positions 20:16 (rt), while for an R-type instruction it is in bit positions 15:11 (rd). Thus, we will need to add a multiplexor to select which field of the instruction is used to indicate the register number to be written.Using this information, we can add the instruction labels and extra multiplexor(for the Write register number input of the register file) to the simple datapath.Figure 3.10 shows these additions plus the ALU control block, the write signals forstate elements, the read signal for the data memory, and the control signals for the multiplexors. Since all the multiplexors have two inputs, they each require a single control line.Figure 3.10. shows seven single bit control lines plus the 2-bit ALUOp controlsignal. We have already defined how the ALUOp control signal works, and it is useful to define what the seven other control signals do informally before we determine how to set these control signals during instruction execution. Figure Figure 3.11.describes the function of these seven control lines.

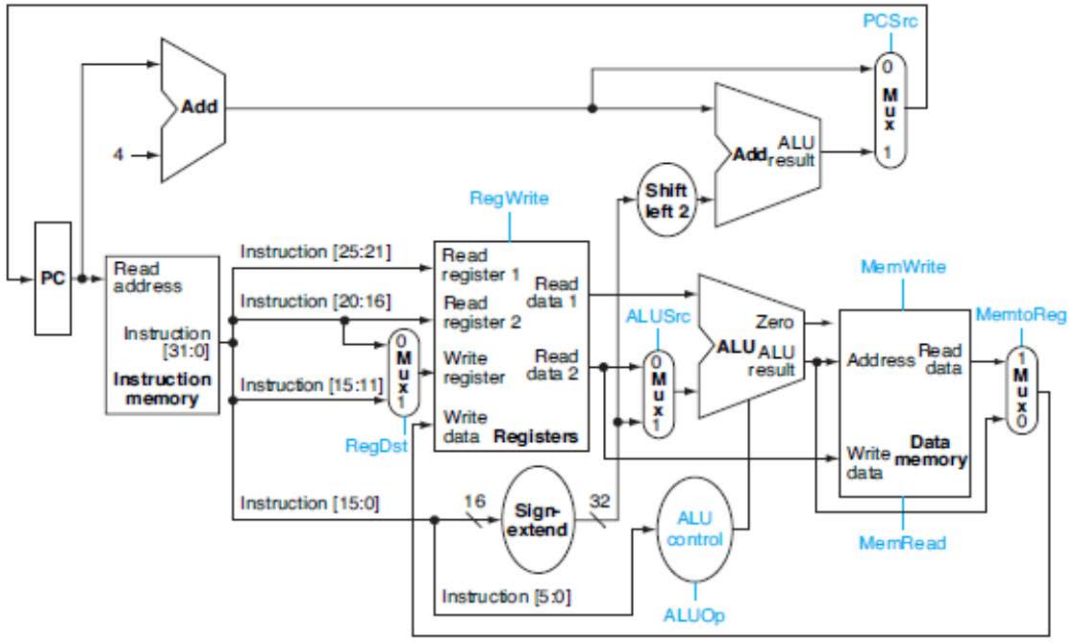


FIGURE 3.10- the datapath of all necessary multiplexors and all control lines identified.

The control lines are shown in color. The ALU control block has also been added. The PC does not require a write control, since it is written once at the end of every clock cycle; the branch control logic determines whether it is written with the incremented PC or the branch target address.

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the rt field (bits 20:16).	The register destination number for the Write register comes from the rd field (bits 15:11).
RegWrite	None.	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None.	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None.	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

Figure 3.11.**The effect of each of the seven control signals.**

- When the 1-bit control to a two-way multiplexor is asserted, the multiplexor selects the input corresponding to 1. Otherwise, if the control is deasserted, the multiplexor selects the 0 input.Remember that the state elements all have the clock as an implicit input and that the clock is used in controlling writes. Gating the clock externally to a state element can create timing problems.

5.Briefly explain about Operation of the Data path with neat diagram.
Operation of the Datapath:

The input to the control unit is the 6-bit opcode field from the instruction.The outputs of the control unit consist of three 1-bit signals that are used to control multiplexors (RegDst, ALUSrc, and MemtoReg), three signals for controlling reads and writes in the register file and data memory (RegWrite, MemRead, and MemWrite), a 1-bit signal used in determining whether to possibly branch (Branch), and a 2-bit control ignal for the ALU (ALUOp). An AND gate is used to combine the branch control signal and the Zero output from the ALU;The AND gate output controls the selection of the next PC. Notice that PCSrc is now a derived signal, rather than one coming directly from the control unit.

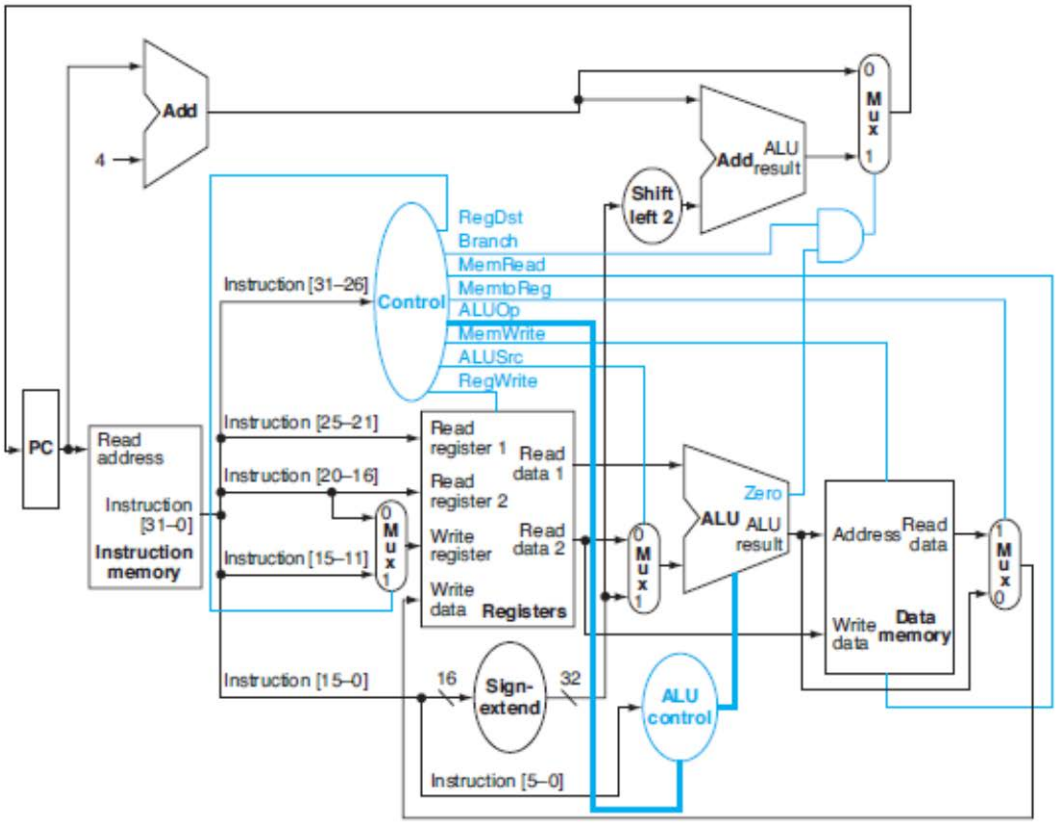


FIGURE 3.12. The simple datapath with the control unit.

Figure 3.12. shows the operation of the datapath for an R-type instruction, such as add $t1,$t2,$t3. Although everything occurs in one clock cycle, we can think of four steps to execute the instruction;

These steps are ordered by the flow of information:

1. The instruction is fetched, and the PC is incremented.
2. Two registers, $t2 and $t3, are read from the register file; also, the main control unit computes the setting of the control lines during this step.
3. The ALU operates on the data read from the register file, using the function code (bits 5:0, which is the funct field, of the instruction) to generate the ALU function.
4. The result from the ALU is written into the register file using bits 15:11 of the instruction to select the destination register ($t1). Similarly, we can illustrate the execution of a load word, such as

```
lw $t1, offset($t2)
```

in a style similar to Figure 3.12.. Figure 3.12 shows the active functional units and asserted control lines for a load. We can think of a load instruction as operating in

Five steps (similar to the R-type executed in four):
1. An instruction is fetched from the instruction memory, and the PC is incremented.2. A register ($t2) value is read from the register file.3. The ALU computes the sum of the value read from the register file and the Sign-extended, lower 16 bits of the instruction (offset).
4. The sum from the ALU is used as the address for the data memory.5. The data from the memory unit is written into the register file; the register destination is given by bits 20:16 of the instruction ($t1) .**The datapath in operation for a load instruction:**The control lines, datapath units, and connections that are active are highlighted. store instruction would operate very similarly. The main difference would be that the memory control would indicate a write rather than a read, the second register value read would be used for the data to store, and the operation of writing the data memory value to the register file would not occur.**The datapath in operation for a branch-on-equal instruction**:Finally, the branch-on-equal instruction, such as **beq $t1,$t2,offset,** in the same fashion. It operates much like an R-format instruction, but the ALU output is used to determine whether the PC is written with PC + 4 or the branch target address. Figure 3.12.shows the four steps in execution:1. An instruction is fetched from the instruction memory, and the PC is incremented.2. Two registers, $t1 and $t2, are read from the register file.3. The ALU performs a subtract on the data values read from the register file. The value of PC + 4 is added to the sign-extended, lower 16 bits of the instruction (offset) shifted left by two; the result is the branch target address.The Zero result from the ALU is used to decide which adder result to store into the PC.

6.Explain a 4-stage instruction pipeline. Also explain the issues affecting pipeline performance. (Or) Discuss the basic concepts of pipelining._ (Apr/May-2012, May/Jun-2013)
Pipelining
Role of cache memory
Pipelining Performance

PIPELINING:
Pipelining is an implementation technique in which multiple instructions are overlapped during the execution of instruction to improve the performance.
pipeline can be visualized as a collection of processing segments through which binary information follows.In computer architecture Pipelining means executing machine instructions concurrently. The pipelining is used in modern computers to achieve high performance. The speed of execution of programs is influenced by **many factors.** The processor executes a program by fetching and executing instructions, one after the other.Let Fi and Ei refer to the fetch and execute steps for instruction Ii.
Executions of a program consists of a sequence of fetch and execute steps, as shown in Figure 3.l.execute steps, as shown in Figure 3.13

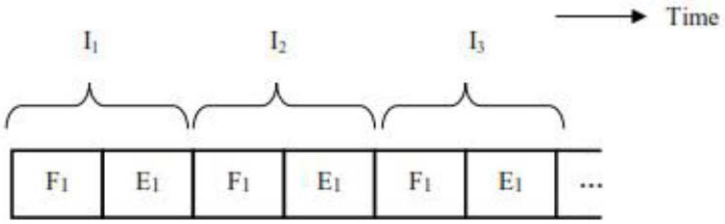


Figure 3.13.Sequential executions of instructions.

Now consider a computer that has two separate hardware units, one for fetching instructions andanother for executing them, as shown in Figure 3.14 The instruction fetched by the fetch unit is deposited in an intermediate storage buffer, B1.This buffer is needed to enable the execution unit to execute the instruction while the fetch unit is fetching the next instruction.The results of execution are deposited in the destination location specified by the instruction. The data can be operated by the instructions are inside the block labeled **"Execution unit".**

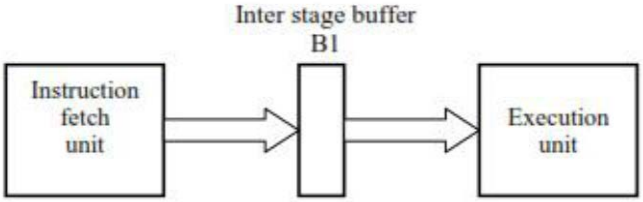


Figure 3.14.Hardware organization of pipelining.

The computer is controlled by a clock whose period is such that the fetch and execute steps of any instruction can each be completed in one clock cycle. Operation of the computer proceeds as in Figure 3.15In the first clock cycle, the fetch unit fetches an instruction I1 (step F1) and stores it in buffer Bl at the end of the clock cycle.In the second clock cycle, the instruction fetch unit proceeds with the fetch operation for instruction I2 (step F2). Meanwhile, the execution unit performs the operation specified by instruction I1, which is available to it in buffer Bl (step E1). By the end of the second clock cycle, the execution of instruction I1 is completed and instruction I2 is available. Instruction I2 is stored in B1, replacing I1, which is no longer needed. Step E2 is performed by the execution unit during the third clock cycle, while instruction I3 is being fetched by the fetch unit. In this manner, both the fetch and execute units are kept busy all the time.

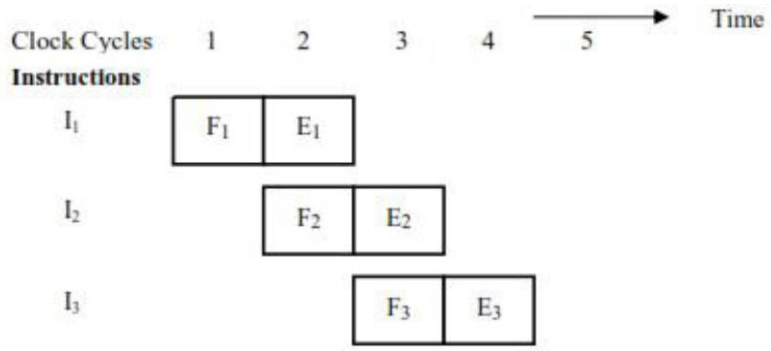


Figure 3.15. Pipelined executions of instructions (Instructions Pipelining)

A pipelined processor may process each instruction in **four steps, as follows:**

F Fetch: read the instruction from the memory.

D Decode: decode the instruction and fetch the source operand(s).

E Execute: perform the operation specified by the instruction.

W Write: store the result in the destination location.

The sequence of events for this case is shown in Figure **3.16**. Four instructions are in progress at any given time. This means that four distinct hardware units are needed, as shown in Figure 3.17.

These units must be capable of performing their tasks simultaneously and without interfering with one another. Information is passed from one unit to the next through a storage buffer.

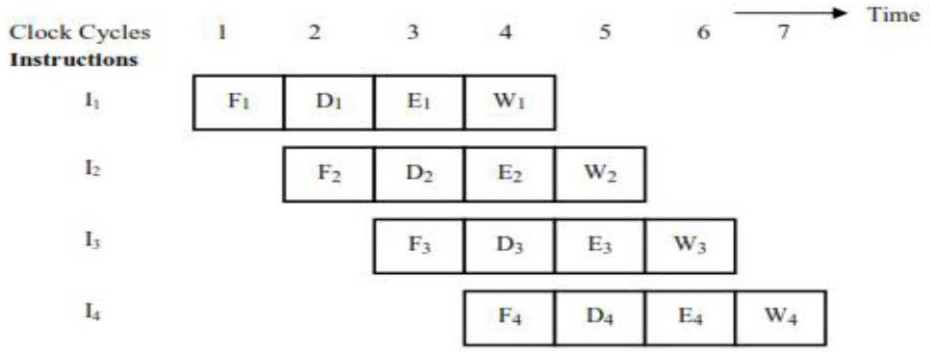


Figure 3.16 Instruction execution divided into four steps.

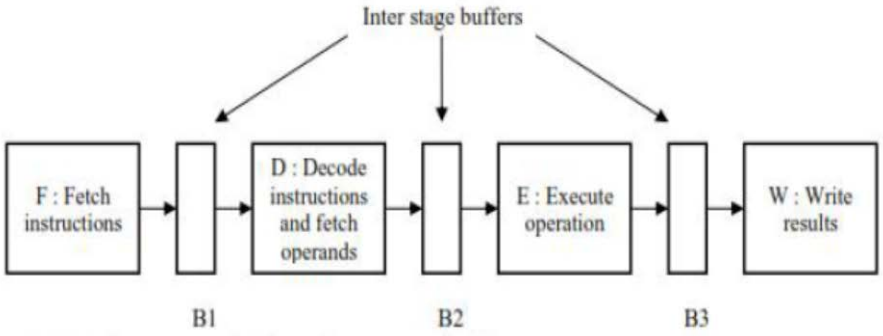


Figure 3.17. Hardware organization of a 4- stage pipeline

For example, during clock cycle 4, the information in the buffers is as follows:

- Buffer B1 holds instruction I3, which was fetched in cycle 3 and is being decoded by the Instruction-decoding unit.Buffer B2 holds both the source operands for instruction I2 and the specification of theoperation to be performed.Buffer B3 holds the results produced by the execution unit and the destination information forinstruction 11.

ROLE OF CACHE MEMORY:

Pipelining is most effective in **improving performance** if the tasks being performed in different stages require about the same amount of time.The use of cache memories solves the memory access problem. In particular, when a cache is included on the same chip as the processor, access time to the cache is usually the same as the time needed to perform other basic operations inside the processor.This makes it possible to divide instruction fetching and processing into steps that are more or less equal in duration. Each of these steps is performed by a different pipeline stage, and the clock period is chosen to correspond to the longest one.**systems**

7.Explain about Pipeline Performance (or) How to measure the performance of a pipeline.or List the key aspects in gaining the performance in pipelined.(Apr/May-2010)

Pipeline performance:

The pipelined processor in Figure 3.4 completes the processing of one instruction in each clock cycle, which means that the rate of instruction processing is four times that of sequential operationThe potential increase in performance resulting from pipelining is proportional to the number of pipeline stages.However, this increase would be achieved

only if pipelined operation as depicted in **Figure 3.17** could be sustained without interruption throughout program execution. Unfortunately, this is not the case.

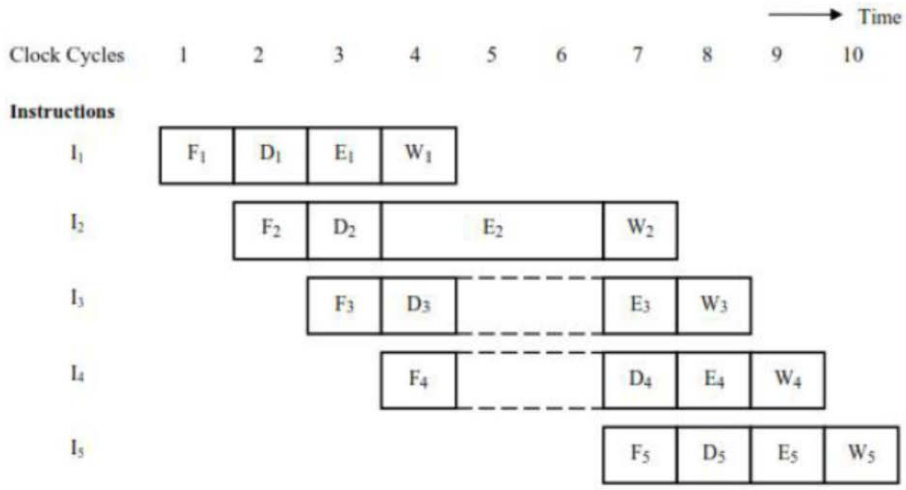


Figure 3.17.Effect of an execution operation taking more than one clock cycle

Performance measures: The various performance measures of pipelining are

Throughput

CPI

Speedup

Dependencies

Hazards

Throughput

The number of instruction executed per second

CPI(clock cycle per instruction) The CPI and MIPS can be related by the equation

CPI=f/MIPS Where F is the clock frequency in MHz

Speedup: Speedup is defined by **S(m)=T(1)/T (m)**

Where T (m) is the execution time for some target workload on an m-stage pipeline and T(1) is the execution time for same target workload on a non pipelined **Processor.**

DependenciesIf the output of any stage interferes the execution of other stage then dependencies exists.There are two types of dependencies. They are**control dependency**

data dependency Hazard Any condition that causes the pipeline to stall is called a ***hazard***.

There are three type of hazard**I. Data HazardsII. Control/instruction hazardsIII. Structural Hazard**Figure 3.17 shows an example in which the operation specified in instruction I2 requires three cycles to complete, from cycle 4 through cycle 6. Thus, in cycles 5 and 6, the Write stage must be told to do nothing, because it has no data to work with. Meanwhile, the information in buffer B2 must remain intact until the Execute stage has completed its operation. This means that stage 2 and, in turn, stage1 are blocked from accepting new instructions because the information in B1 cannot be overwritten.

Thus, steps D4 and F5 must be postponed as shown in figure 3.6. Pipelined operation in Figure 3.6 is said to have been stalled for two clock cycles. Normal pipelined operation resumes in cycle 7.

I. Data Hazards

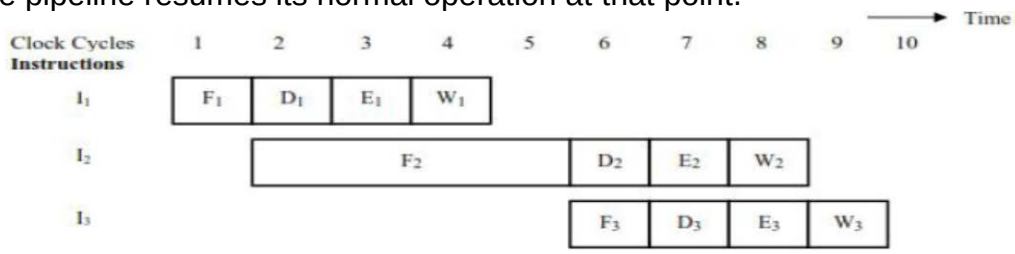
A ***data hazard*** is any condition in which either the source or the destination operands of an instructionare not available at the time expected in the pipeline. As a result some operation has to be delayed, and the pipeline stalls.

II. Control/instruction hazards.

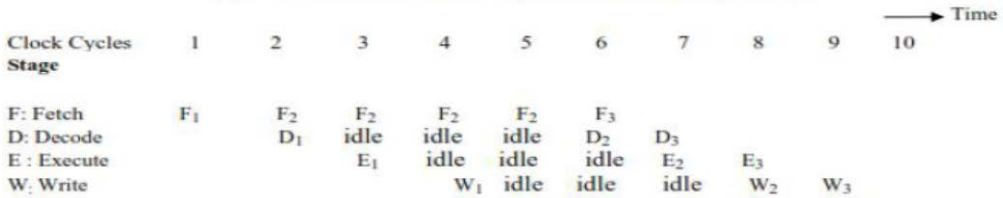
The pipeline may also be stalled because of a delay in the availability of an instruction.
For example,
This may be a result of a miss in the cache, requiring the instruction to be fetched from the main memory. Such hazards are often called ***control hazards or instruction hazards***

- The effect of a cache miss on pipelined operation is illustrated in Figure 3.18 Instruction I1 is fetched from the cache in cycle 1, and its execution proceeds normally.

- However, the fetch operation for instruction I2, which is started in cycle 2, results in a cache miss. The instruction fetch unit must now suspend any further fetch requests and wait for I2 to arrive. We assume that instruction I2 is received and loaded into buffer Bl at the end of cycle 5.
- The pipeline resumes its normal operation at that point.



(a) Instruction execution steps in successive clock cycles.



(b) Functions performed by each processor stage in successive clock cycles.

.

Figure 3.18. Pipeline stall caused by a cache miss in F2

Structural Hazard

A third type of hazard that may be encountered in pipelined operation is known as a **structural hazard**.

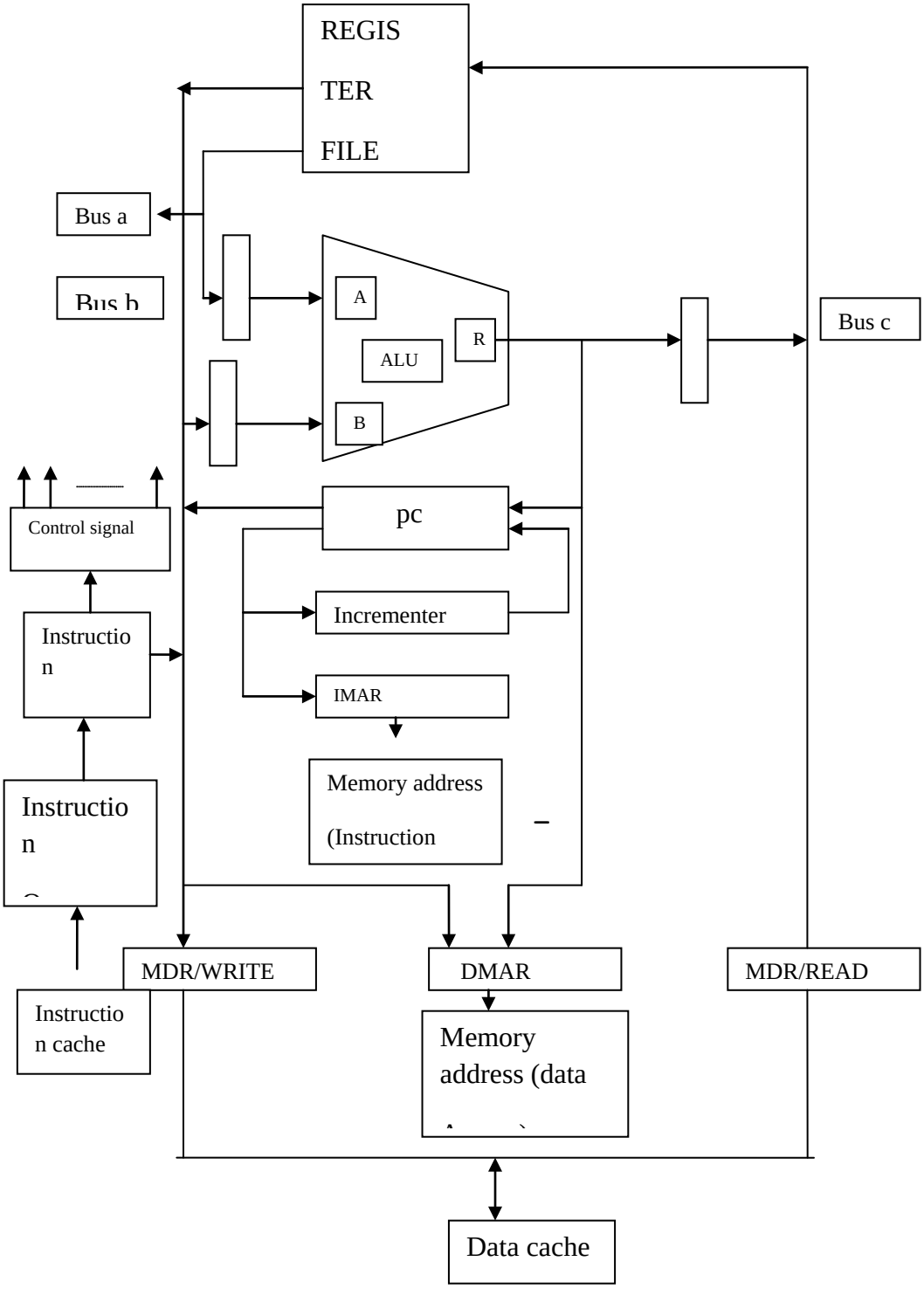
This is the situation when **two instructions require the use of a given hardware resource at the same time.** The most common case in which this hazard may arise is in access to memory.One instruction may need to access memory as part of the Execute or Write stage while another instruction is being fetched.If instructions and data reside in the same cache unit, only one instruction can proceed and the other instruction is delayed.Many processors use separate instruction and data caches to avoid this delay. An example of a structural hazard is shown in Figure This figure shows how the load instruction.**oad X(Rl),R2**can be accommodated in our example 4-stage pipeline.The memory address, X+[Rl], is computed in step E2 in cycle 4, then memory access takes place in cycle 5. The operand read from memory is written into register R2 in cycle 6. This means that the execution step of this instruction takes two clock cycles (cycles 4 and 5).

8.Give detail description about Pipelined data path and control. Explain Data path and its control in detail (Nov/Dec-2014, Apr/May-2012)

Pipelined data path and control

Consider the three-bus structure suitable for pipelined execution with a slight modification to support a 4-stage pipeline as shown in figure 3.18. Several important changes are
1. There are ***separate instruction and data caches*** that use separate address and data connections to the processor. This requires two versions of the MAR register, IMAR for accessing tile instruction cache and DMAR for accessing the data cache.
2. The PC is connected ***directly*** to the IMAR, so that the contents of the PC can be transferred to IMAR at the same time that an independent ALU operation is taking place.
3. The data address in DMAR can be obtained ***directly from the register file or from the ALU*** to support the register indirect and indexed addressing modes.
4. Separate MDR registers are provided for ***read and write operations*.** Data can be transferred directly between these registers and the register file during load and store operations without the need to pass through the ALU.
5. ***Buffer registers*** have been introduced at the inputs and output of the ALU. These are registers SRCl, SRC2, and RSLT. Forwarding connections may be added if desired.
6. The instruction register has been replaced with an ***instruction queue***, which is loaded from the instruction cache.
7. The output of the instruction decoder is connected to the ***control signal pipelin*e**. This pipeline holds the control signals in buffers B2 and B3 in Figure 3.17

The following operations can be performed ***independently*** in the processor of Figure 3.19:
Figure 3.19.Data path modified for pipelined execution with interstage buffers at the input and output of the ALU
The following operations can be performed independently in the process of fig:
Reading an instruction from the instruction cacheIncrementing the pcDecoding the instructionReading from or writing into the data cache.Reading the contents of up to two registers from the register file.Writing in to one register in the register filePerforming an ALU operationthe structure provides the flexibility required to implement the four-stage pipeline.
For example: I1, I2, I3, I4
Be the sequence of four instructions.
Write the result of instruction I1 into the register file.
Read the operands of instruction I2 from the register file.
Decode instruction I3
Fetch instruction I4 and increment the PC.


REGIS
TER
FILE
Bus a
Bus b
A
ALU
R
B
Bus c
Control signal
pc
Incrementer
Instructio
n
IMAR
Memory address
(Instruction
Instructio
n
MDR/WRITE
DMAR
MDR/READ
Instructio
n cache
Memory
address (data
Data cache

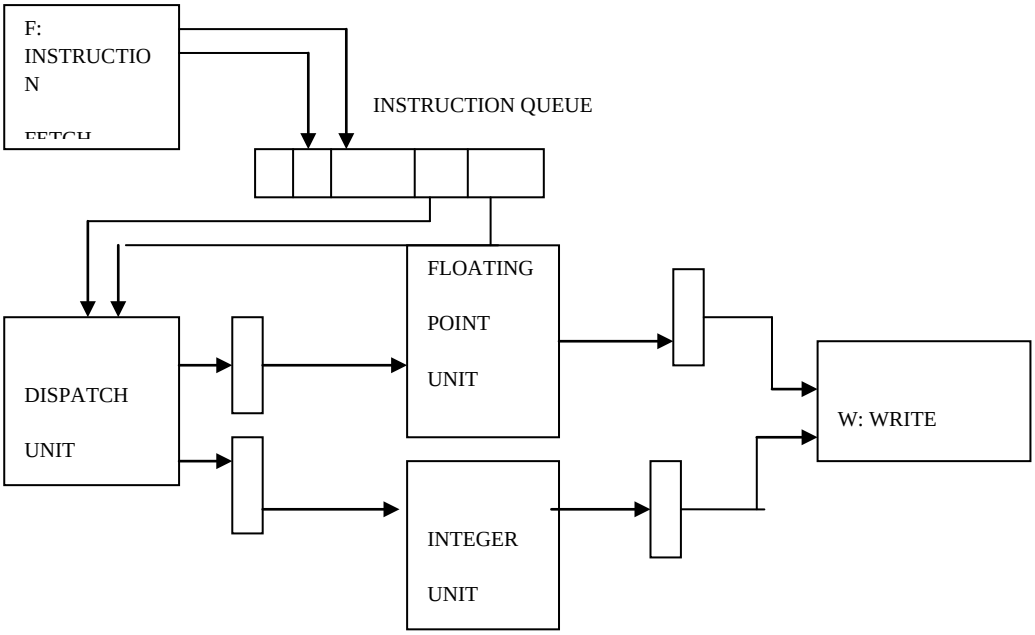


Figure 3.20. Use of instruction queue in hardware organization

What is Hazard? Explain its types with suitable example Or Explain the different types of pipeline hazard with suitable examples (Nov/Dec-2014,Apr/May-2015)

- 3.6. Data hazard
- 3.7. Instruction hazard
- Structural hazard

Briefly explain about how to handle the Data hazard(or) Pipeline Hazard (Nov/Dec-2014,Apr/May-2015)(or) What is a data hazard? How do you overcome it? and discuss its side effects. (Apr/May 2014)

Handling Data hazard

Data hazard

When either the source or the destination operands of an instruction are not available at the time expected in the pipeline and as a result pipeline is stalled, we say such a situation is a **data hazard.**Consider a program with two instructions I1 followed by I2; when this program is executed in a pipeline; the execution of these two instructions can be performed concurrently. An such case, the result of I1 may not be available for the execution of I2. If the result of I2 is dependent on the result of I1, we may get incorrect result if both are executed concurrently. For example; assume A=8 in the following two operations.

I_1:A<- A + 6

I_2:B<- A X 2

When these two operations are performed in the order given, the result is B=28. But if they are performed concurrently, the value of A used in computing B would be the original value 8; leads to an incorrect result.In the case data used in the I_2 depend on the result of I1. The hazard due to such situation is called data **hazard** or **data dependent hazard.**To avoid incorrect result we have to execute dependent instruction one after the other.The data dependency arises when the destination of one instruction is used as a source in the nest instruction. For example the two instruction

Mul R_2, R_3, R_4

Add R_5, R_4, R_6

The contents of register R2 is multiplied by the contents of R_3 and the result is stored in R_4, which in turn is one of the two source operands of the Add instruction. Assume that the multiply operation takes one clock cycle to complete, execution would proceed as shown in figure below.

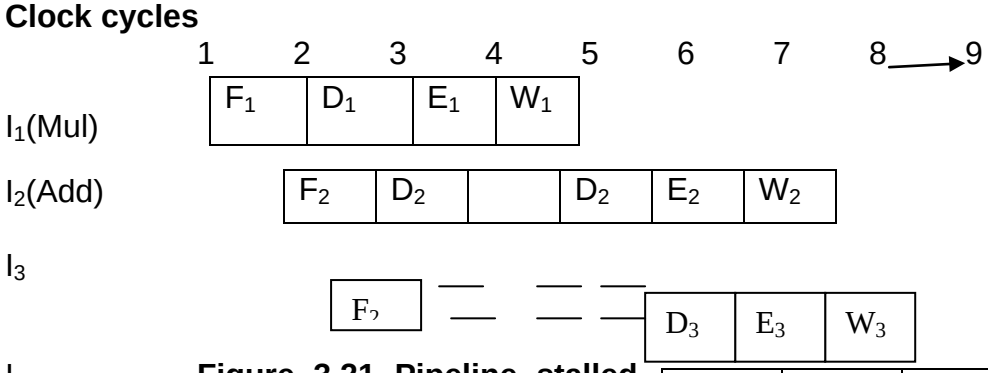
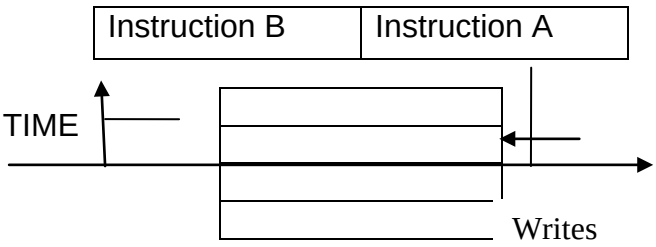


Figure 3.21 Pipeline stalled by data dependency between D2 and w1

In the above figure the decode unit decodes the Add instruction in cycle 3, it realizes that R_4 is used as a source operand. Hence, the D step of that instruction cannot be completed until the step W of multiply instruction has been completed. Step D_2 must be delayed to clock cycle 5, and is shown as step D_{2A} in the figure. Instruction I_3 is fetched in cycle 3, but its decoding must be delayed because step D_3 cannot precede D_2. Hence, pipelined execution is stalled for two cycles.

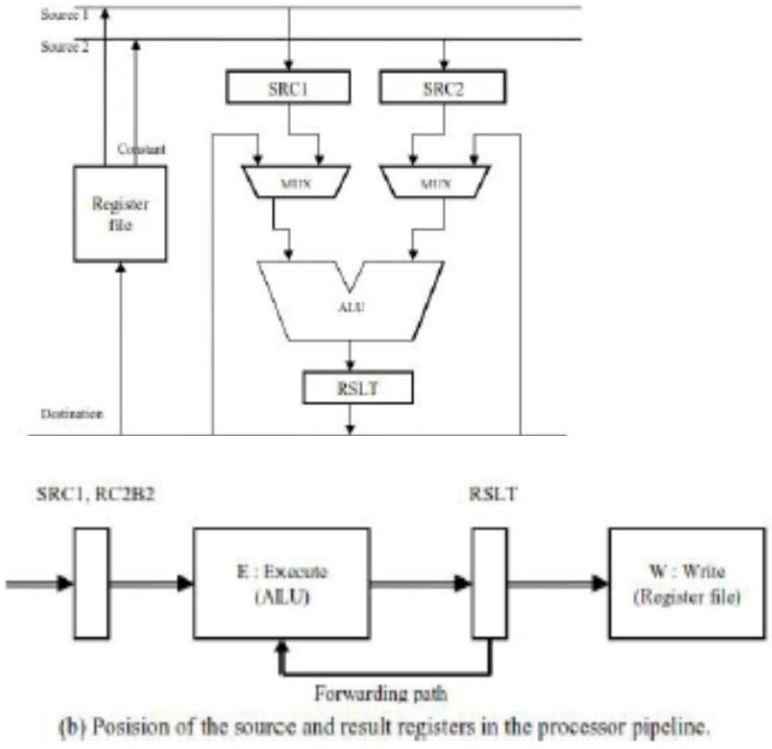
Operand Forwarding:

Consider instruction A and B as illustrated in the figure, B tries to read a register before A has written if and gets the old value.



This is quite common and called **read after write data hazard.** This situation is solved with a simple hardware technique called **operand forwarding** Add $ s0, $ t0, $ t1

Sub $ t2, $ s0, $ t3 (A) Forwarding Datapath



(b) Posision of the source and result registers in the processor pipeline.

Figure 3.22.Operand forwarding in a pipeline processor

Figure shows a part of the processor data path involving the ALU and the register file. This arrangement is similar to the three bus structure except that registers SRC1, SRC2 and RSLT have been added.

- These registers constitute inter stage buffers needed for pipelined operation as shown in figure b.

- The two multiplexers connected at the inputs to the ALU allow the data on the destination bus to be selected instead of the contents of either the SRC1 or SRC2 register.

- The instruction's in figure 1 is executed in the data path of figure a and b . After decoding instruction I_2 and detecting the data depending, a decision is made to use data forwarding. Register R_2 is read and loaded in register. SRC1 in clock cycle 3.

  In the next cycle, the product produced by instruction I, is available in register RSLT and because of the forwarding connection it can be used in steep E_2. Hence execution of I_2 proceeds without interruption.

Handling Data Hazard in Software:

Another way to avoid data dependencies is to use software.

In the software approach compiler can introduce two cycle delay needed between instruction I_4 and I_2 in figure by NOP(no operation) instruction as follows:

I_1: Mul R_2,R_3,R_4
NOP
NOP
I_2: Add R_5,R_4,R_6

In the responsibility for detecting such dependencies is left entirely to the software the compiler must insert NOP instruction to obtain a correct result. This possibility illustrates the close link between the compiler and the hardware.

Side Effects:

- When a location other one explicitly named in an instruction as a destination operand is affected, the instruction is said to have a side effect.
- An instruction that uses an auto increment or auto decrement addressing mode is an example.
- In addition to storing a new data in its destination location, the instruction changes the contents of a source register used to access one of its operands.

For example,

A stack instructions, such as push and pop, produce similar side effects because they implicitly use the auto increment and auto decrement addressing modes

- Another possible side effect involves the condition code flags, which are used by instructions such as conditional branches and add with carry.

Add R_1,R_3

Add with carry R_2,R_4.

- An implicit dependency exists between these two instructions through the carry flag. This flag is set by the first instruction and used in the second instruction, which performs the operation. R_4<-[R_2] + [R_4] + carrystructions that have side effects give rise to multiple data dependencies, which lead to a substantial increase in the complexity of the hardware or software needed to resolve them. For this reason, instructions designed for execution on pipelined hardware should have few side effects. Ideally, only the content of the destination location, either a register or a memory location, should be affected by any given instruction. Side effects, such as setting the condition code flags or updating the contents of an address pointer, should be kept to a minimum.

10.Explain How to handle Instruction Hazards Or control hazards.(or) Describe the techniques for hand ling control hazards in pipelining.(Nov/Dec-2014,Apr/May-2015, May/Jun-2013) Handling Instruction Hazards Or control hazardsOver View: This type of hazard arise from pipeline of branch and other instructions that change the contents of PC ie. Trying to make a decision based on the results of instruction while others are executing..**Unconditional Branch:**

Figure shows a sequence of instructions being executed in a two-stage pipeline instruction I_1 to I_3 are stored at consecutive memory address and instruction I_2 is a branch instruction.

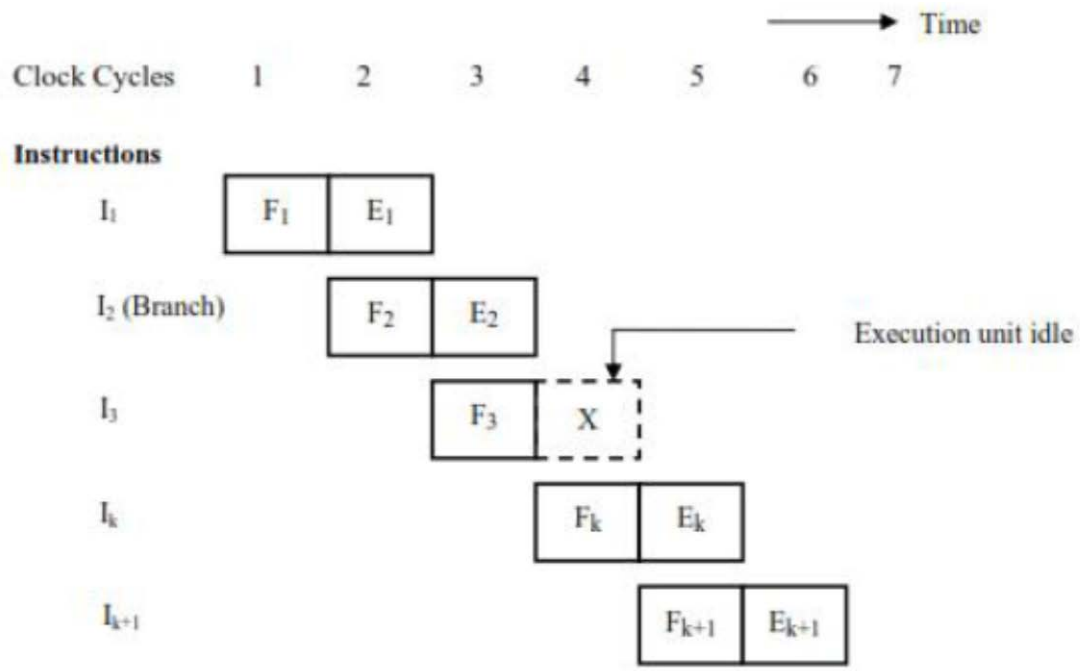
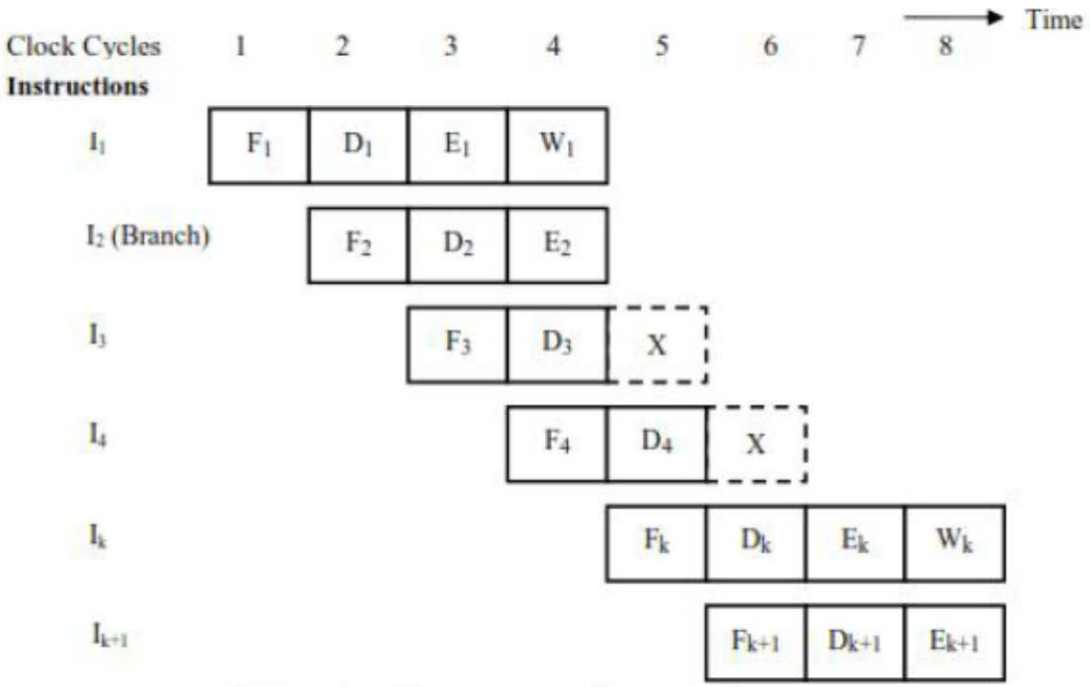


Figure 3.23. An idle cycle caused by a branch instruction

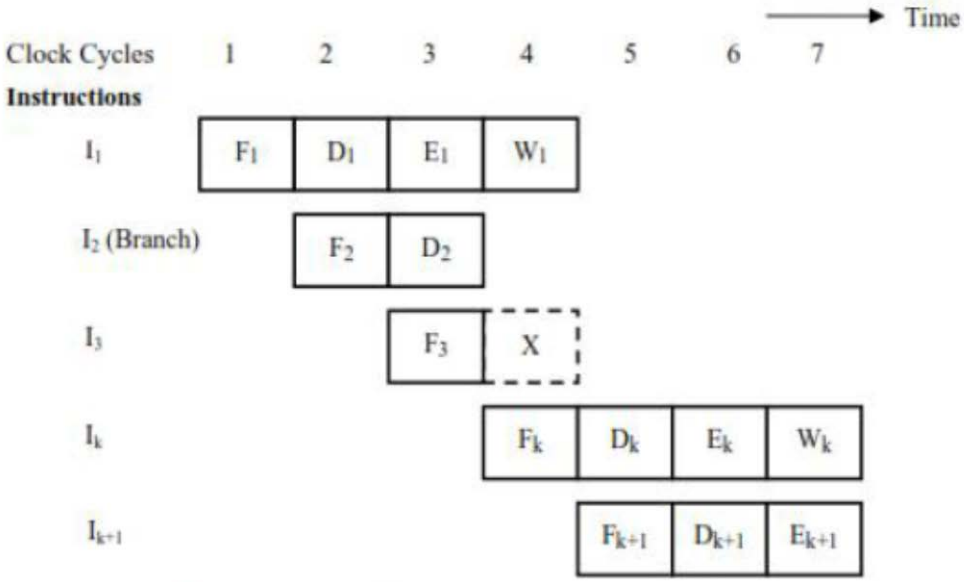
If branch is taken as shown in figure, then the PC value is not known till the end of I_2. Next three instructions are fetched even though they are not required. Hence they have to be flushed after branch is taken and new set of instructions have to be fetched from the branch address.

In figure, clock cycle 3, the fetch operation for instruction I_3 is in progress at the same time the branch instruction is being decoded. In clock cycle 4, the processor must discard I_3,which has been incorrectly fetched , and fetch instruction I_k. Thus the pipeline is stalled for one clock cycle.The time lost as a result of branch instruction is often referred to as the **branch penalty.**The branch penalties can be reduced by proper scheduling through compiler technique.e basic idea behind these techniques is to fill the 'delay slot' with some useful instruction which in most cases will be executed.For longer pipeline, the branch penalty may be higher. Reducing the branch penalty requires the branch address to be computed earlier in the pipeline. The instruction fetch unit has dedicated hardware to identify a branch instruction and compute branch target address as quickly as possible after an instruction is fetched.

- With these additional hardware both these tasks can be performed in step D_2, leading to the sequence of events shown in figure. In this case the branch penalty is only one clock cycle.



Fig(a). Branch address computed in execute stage



Fig(b). Branch address computed in decode stageFigure 3.24. Branch Timing

Instruction Queue and Pre fetching:

- The **Fetch unit** may contain instruction queue to store the instruction before they are needed to avoid interruption.

- Another unit called **dispatch unit** takes instruction from the front of the queue and sends them to the section unit.
- The dispatch unit also performs the decoding function. This organization is shown in figure.

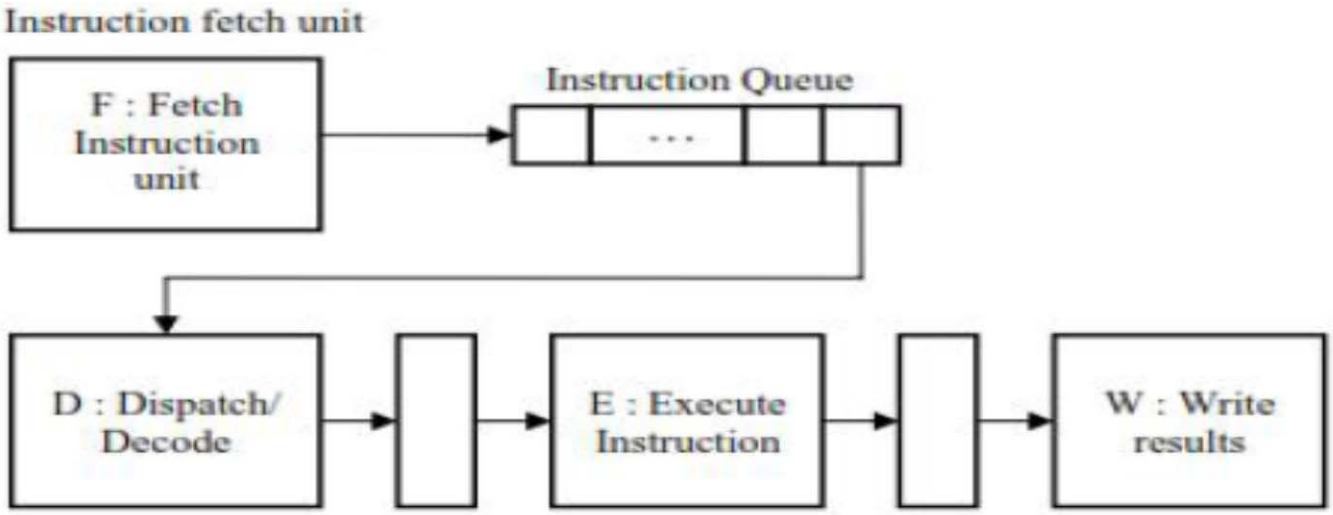


Figure 3.25. Hardware organization of instruction queue

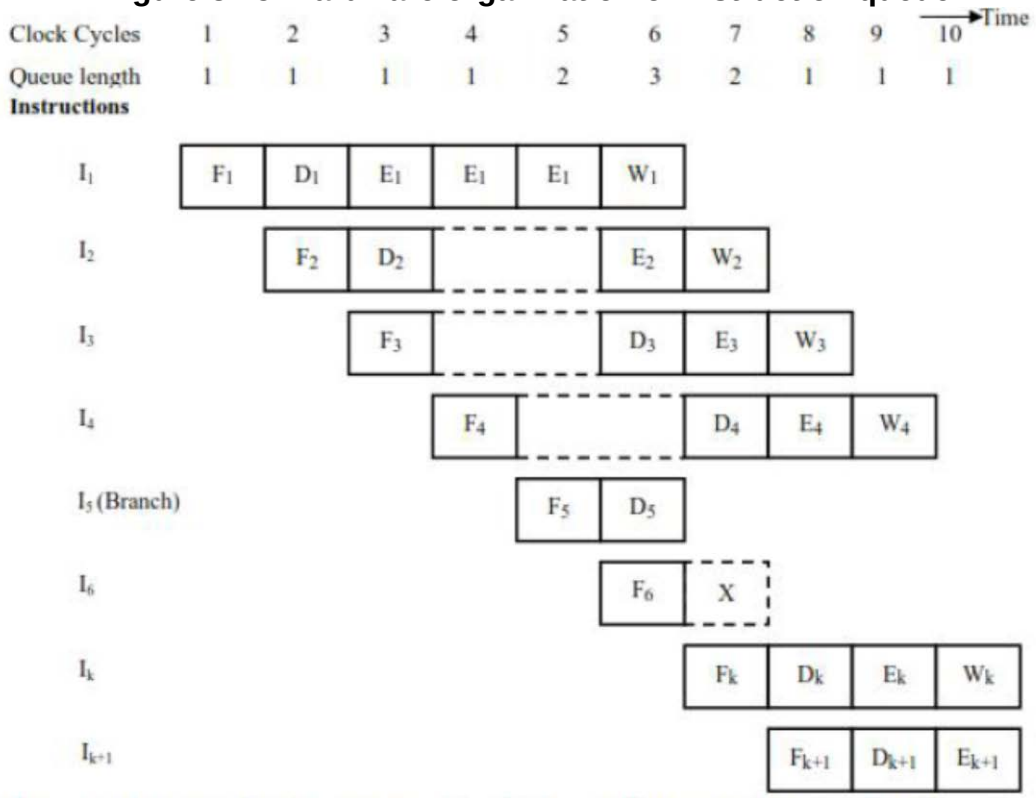


Figure 3.26 Branch timing in the presence of an instruction queue. Branch target address in computed D stage

The fetch unit must have sufficient decoding and processing capability to recognize and execute branch instructionThe fetch unit always keep the instruction queue filled at all times.Fetch unit continues to fetch instructions and add them to the queue.Similarly if there is a delay in fetching instructions, the dispatch unit continues to issue instruction from the instruction queue. Figure shows how the queue length changes and how it affects the relationship between different pipeline stages.Every fetch operation adds one instruction to the queue and each dispatch operation reduces queue length by one. Hence queue length remains same for first four clock cycle.Instruction I_5 is a branch instruction Its target instruction, I_k , is fetched in cycle 7, and instruction I_6 is discarded. The branch instruction

would normally cause a stall in cycle 7 as a result of discarding instruction I_6. Instead, instruction I_4 is dispatched from queue to the decoding stage. After discarding I6, the queue length drops to 1 in cycle 8. The queue length will be at this value until another stall is encountered.From figure we observe the sequence of instruction completions instruction I_1,I_2,I_3,I_4 and I_k complete execution in successive clock cycle. Hence the branch instruction does not increase the overall execution time. This is because the instruction fetch unit has executed branch instruction concurrently with the execution of other instruction. This technique is referred to as **branch folding.** Branch folding occurs only if at the time a branch instruction encountered, at least one instruction is available in the queue other than the branch instruction.

11.Explain about Conditional Branches: (Apr/May -2014)

Conditional Branches:

The conditional branching is a major factor that affects the performance of instruction pipelining. When a conditional branch is executed if may or may not change the PC.

If a branch changes the PC to its target address, it is a taken branch, if it falls through, it is not taken. The decision to branch cannot be taken until the execution of that instruction has been completed.**Delayed Branch:**he location following the branch instruction is called branch delay slot. There may be more than one branch delay slot depending on the time it takes to execute the branch instruction.The instruction in the delay slot are always fetched at least partially executed before the branch decision is made and the branch target address is completedA technique called **delayed branching** can minimize the penalty caused by conditional branch instruction.The instruction in the delay slot are always fetched. Therefore, arrange the instructions which are fully executed, whether or not the branch is taken.Place the useful instructions in the delay slot.If no useful instructions available, fill the slot with NOP instructions.

EXAMPLE:

```
---------------------------------------------------------------------------
LOOP    shift-Left      R1
        Decrement       R2
        Branch = 0      LOOP
NEXT    Add             R1 , R2
```

(a). Original program loop

```
---------------------------------------------------------------------------
LOOP    Decrement       R2
        Branch = 0      LOOP
        shift-Left      R1
NEXT    Add             R1 , R2
```

(b). Reordering instructions

Figure 3.27. Reordering of instructions for a delayed branch

Register R_2 is used as counter to determine the number of times contents of R_1 are shifted left. For a processor with one delay slot, the instructions can be recorded a above. For a processor with one delay slot, the instructions can be reordered as shown in figure(b). The shift instruction is fetched while branch instruction is being executed.After evaluating the branch condition, the processor fetches the instruction at LOOP or at NEXT, depending on whether the branch condition is true or false respectively. In either case, it completes the

execution of the shift instructions.The sequence of events during the last two passes in the loop is illustrated in figure.

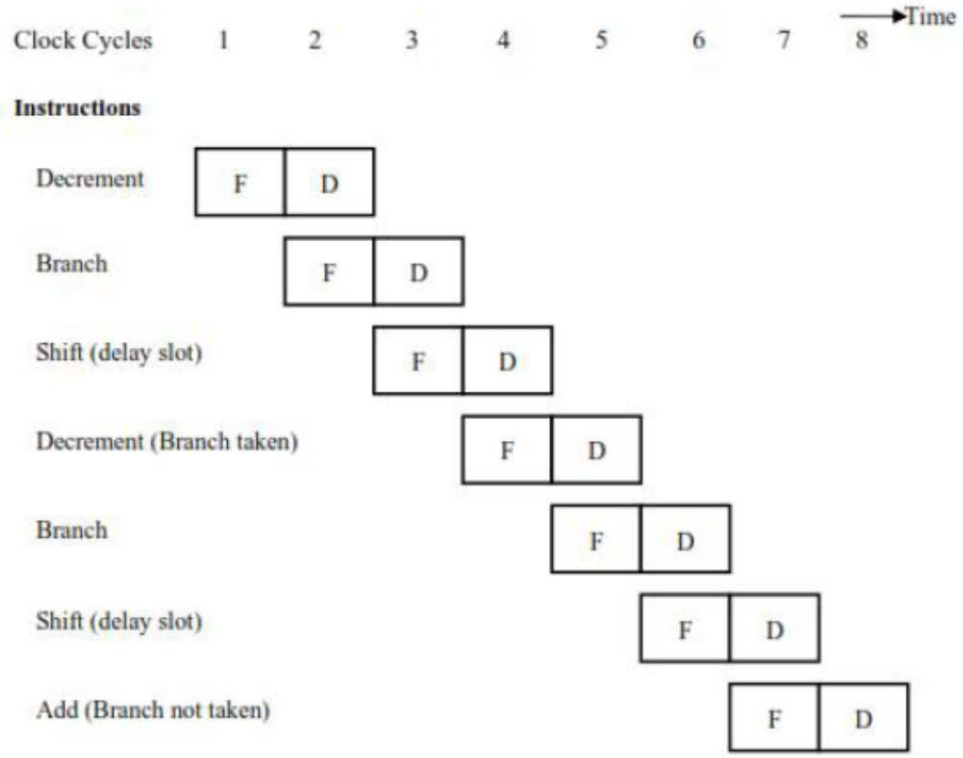


Figure 3.28.Execution timing showing the delay slot being filled during two passes through the loop

Pipelined operation is not interrupted at any time, and there are no idle cycles. Branching takes place one instruction later than where branch instruction appears in the sequence, hence named "delayed branch".

12.Give detail description about Branch prediction Algorithm.

Branch prediction:

Speculative execution means

Static prediction

Dynamic Branch Prediction:

Branch prediction

Prediction techniques can be used to check whether a branch will be valid or not valid.

The simplest form of branch prediction is to assume that the branch will not take place and to continue to fetch instructions in sequential address order. Until the branch condition is evaluated, instruction execution along the predicted path must be done on a speculative basis.**Speculative execution means** that instructions are executed before the processor is certain that they are in the correct execution sequence. Figure illustrate the incorrectly predicted branch.

- Figure shows a compare instruction followed by Branch > 0 instruction. In cycle 3 the branch prediction takes ;

- The fetch unit predicts that branch will not be taken and it continues to fetch instruction I_4 as I_3 enters the Decode Stage.

- The result of compare operation are available at the ends of cycle 3. The branch condition is evaluated in cycle 4.

- At this point, the instruction fetch unit, realizes that the prediction was incorrect and the two instruction in the execution pipe are purged.

- A new instruction I_k is fetched from the branch target address in clock cycle 5. We will examine prediction schemes static and dynamic prediction.

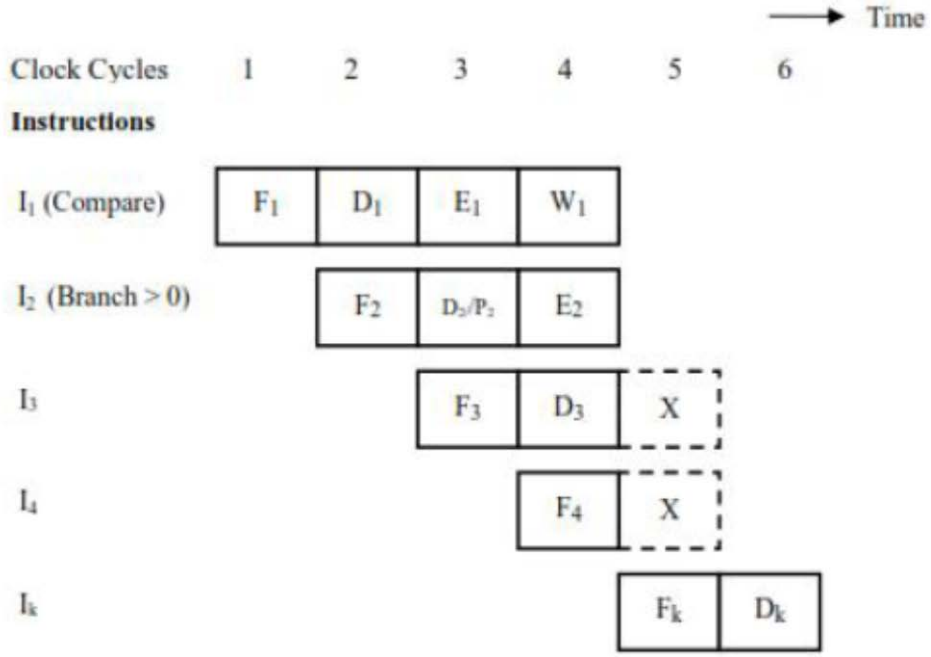


Figure 3.29.Timing when branch decision has been incorrectly predicted as not taken

Static prediction

Static prediction is usually carried out by the compiler and it is static because the prediction is already known even before the program is executed.

Dynamic Branch Prediction: (May/Jun-2013)

Dynamic prediction in which the prediction **may change depending on execution** history.

Algorithm:

If the branch taken recently, the next time if the same branch is executed, it is likely that the branch is takeState 1:LT: Branc State 2:LNT:Branch is likely not to be take.

The algorithm is stated in state LNT when the branch is executed.

If the branch is taken, the machine moves to LT. Otherwise it remains in state LNT.

The branch is predicted as taken if the corresponding state machine is in state LT, otherwise it is predicted as not take.

The branch is predicted as taken if the corresponding state machine is in state LT, otherwise it is predicted as not take.

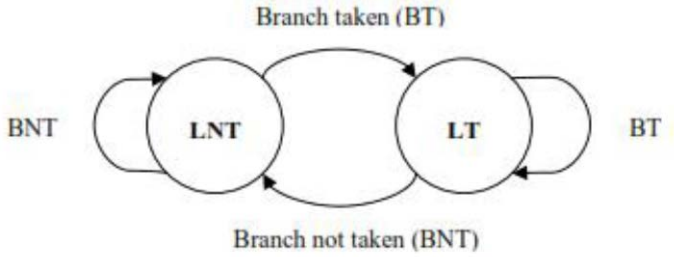


Figure 3.30. A 2-State machine representation of branch-prediction

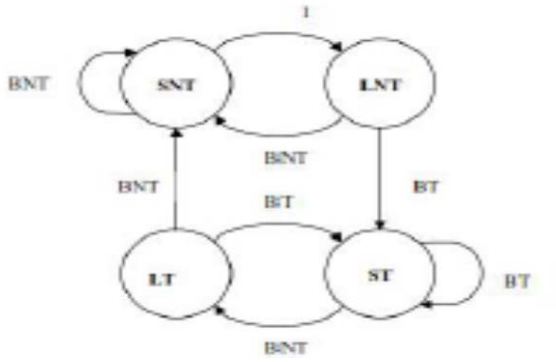


Figure 3.31. A 4-State machine representation of branch-prediction

Algorithm:

An algorithm that uses 4 states, thus requiring two bits of history information for each branch instruction is shown in figure. The four states are:

ST : Strongly likely to be taken
LT : Likely to be taken.
LNT : Likely not to be taken
SNT : Strongly likely not to be taken.

STEP 1: Assume that the state of algorithm is initially set to LNT.]
STEP 2: If the branch is actually taken change to ST, otherwise it is changed to SNT
STEP 3: When a branch instruction is encountered, the branch will be taken if the state is either LT or ST and it begins to fetch instructions at the branch target address. Otherwise , it continues to fetch instructions in sequential address order.When in state SNR, the instruction fetch unit predicts that the branch will not be taken.'If the branch is actually taken, that is if the prediction is incorrect, the state changes to LNT. The state information used in dynamic branch prediction algorithm require 2 bits for 4 states and may be kept by the processes in a variety of ways:Use look-up table, which is accessed using low-order part of the branch of instruction address.Store as tag bits associated with branch instruction in the instruction cache.

13.Explain in detail how exceptions are handled in MIPS architecture? (Apr/May-2015)

In **MIPS Architecture** two types of exceptions that our current implementation can generate are

- Execution of an undefined instruction and
- An arithmetic overflow.

- The basic action that the processor must perform when an exception occurs is to save the address of the offending instruction in the ***exception program counter* (*EPC*)** and then transfer control to the operating system at some specified address.

- After transferring control to operating system it must take appropriate action to provide some service to the user program
- Operating system will provide the following services to user program

    - Taking some predefined action in response to an overflow,
    - Stopping the execution of the program and reporting an error.

- After performing whatever action is required because of the exception, the operating system can terminate the program or may continue its execution, using the EPC to determine where to restart the execution of the program.
- For the operating system to handle the exception, it must know the reason for the exception,
- There are two main methods used to communicate the reason for an exception.
- **The method used in the MIPS architecture is to include**
  A first method ,is use **status register** (called the *Cause register*), which holds a field that indicates the reason for the exception.

- A second method, is to use **vectored interrupts**. In a vectored interrupt, the address to which control is transferred is determined by the cause of the exception. **For example,** to accommodate the two exception types listed above, we might define the following two exception vector addresses:

Exception type	Exception vector address (in hex)
Undefined instruction	$8000\ 0000_{hex}$
Arithmetic overflow	$8000\ 0180_{hex}$

  - The operating system knows the reason for the exception by the address at which it is initiated.
  - The addresses are separated by 32 bytes or eight instructions, and the operating system must record the reason for the exception and may perform some limited processing in this sequence.
  - When the exception is not vectored, a single entry point for all exceptions can be used, and the operating system decodes the status register to find the cause.
  - To handle exception we can add a few extra registers and control signals to their basic implementation

Implementation of exception in MIPS architecture

To Implementing the exception system used in the MIPS architecture, with the single entry point being the address 8000 0180hex. (Implementing vectored exceptions is no more difficult.)***EPC:*** A 32-bit register used to hold the address of the affected instruction. (Such a register is needed even when exceptions are vectored.) ***Cause Register****:*A register used to record the cause of the exception.In the MIPS architecture, this register is 32 bits, although some bits are currently unused. Assume there is a five-bit field that encodes the two

possible exception sources mentioned above, with 10 representing an undefined instruction and 12 representing arithmetic overflow.

Exceptions in a Pipelined Implementation

In a pipelined implementation treats exceptions as another form of control hazard.In MIPS exception address we add an additional input with the value 8000 0180hex in the multiplexor that supplies the new PC value;A Cause register to record the ause of the exception; and an Exception PC register to save the address of the instruction that caused the exception. The 8000 0180hex input to the multiplexor is the initial address to begin fetching instructions in the event of an exception.Although not shown, the ALU overflow signal is an input to the control unit.

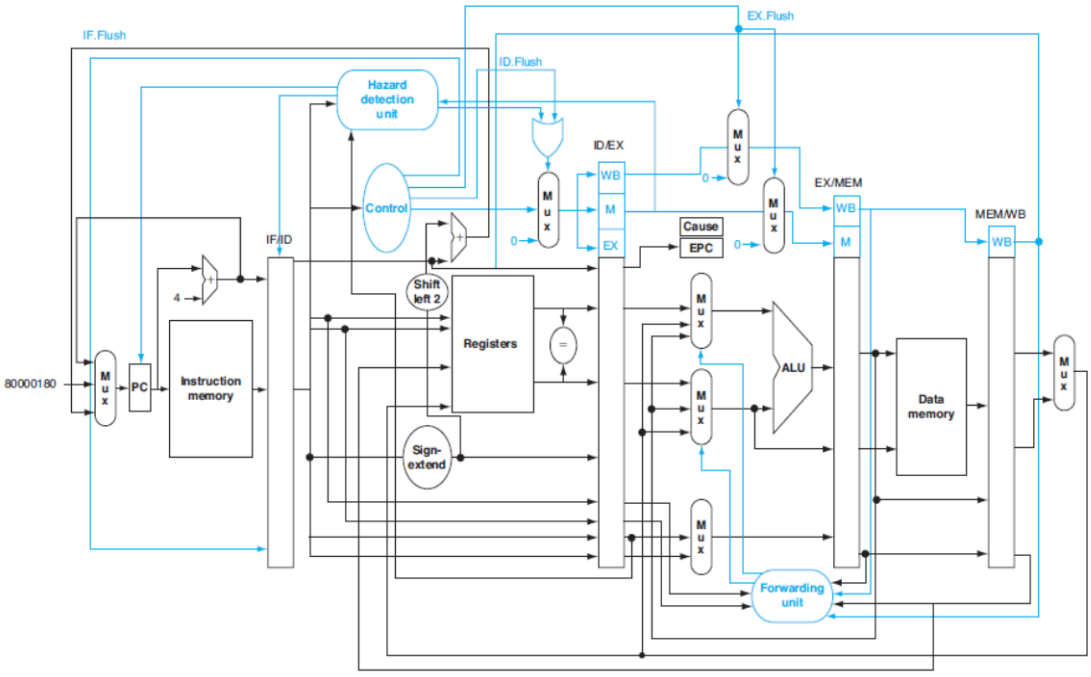


FIGURE 3.32.The data path with controls to handle exceptions.

- This example points out a problem with exceptions: if we do not stop execution in the middle of the instruction, the programmer will not be able to see the original value of register $1 that helped cause the overflow because it will be clobbered as the Destination register of the add instruction.
- Because of careful planning, the overflow exception is detected during the EX stage; hence, we can use the EX.
- Flush signal to prevent the instruction in the EX stage from writing its result in the WB stage.
- Many exceptions require that we eventually complete the instruction that caused the exception as if it executed normally.
- The easiest way to do this is to flush the instruction and restart it from the beginning after the exception is handled.
- The final step is to save the address of the offending instruction in the exceptionprogram counter (EPC). In reality, we save the address + 4, so the exception handling routine must first subtract 4 from the saved value.

Unit IV

Parallelism

Part-A

1. What is Instruction Level Parallelism?

All processors use pipelining to overlap the execution of instructions and improve performance. This potential overlap among instructions is **called instruction-level parallelism (ILP),** since the instructions can be evaluated in parallel.

2.What is Loop –level parallelism?

The simplest and most common way to increase the ILP is to exploit parallel- ism among iterations of a loop. This type of parallelism is often called **loop-level parallelism.**

```
for (i=1; i<=1000; i=i+1)
x[i] = x[i] + y[i];
```

Every iteration of the loop can overlap with any other iteration, although within each loop iteration there is little or no opportunity for overlap.

3.What is Vector instruction ?

An important **alternative method for exploiting loop-level parallelism** is the use of vector instructions .A **vector instruction** exploits data- level parallelism by operating on data items in parallel.

4.List the various types of dependencies in ILP.

Data dependencesName DependencesControl Dependences

5.Define antidependance?

An ***antidependence*** between instruction *i* and instruction *j* occurs when instruction *j* writes a register or memory location that instruction *i* reads.The original ordering must be preserved to ensure that *i* reads the correct value

6.Define output dependence?

An ***output dependence*** occurs when instruction *i* and instruction *j* write the same register or memory location. The ordering between the instructions must be preserved to ensure that the value finally written corresponds to instruction *j*

7.What is Register Renaming?

A name dependence is not a true dependence, instructions involved in a name dependence can execute simultaneously or be reordered, if the name (register number or memory location) used in the instructions is changed so the instructions do not conflict.

This renaming can be more easily done for register operands, where it is called ***register renaming***. Register renaming can be done either statically by a compiler or dynamically by the hardware.

8.List the data hazards in ILP?

The possible data hazards are
RAW
WAW
WAR

9.What is Loop unrolling ? and What are the advantages of loop unrolling?
Loop unrolling is a simple but useful method for increasing the size of straight-line code fragments that can be scheduled effectively. This transformation is useful in a variety of processors.**ADV:**unrolling improves the performance of the loop by eliminating overhead instructions.Loop unrolling can also be used to improve scheduling. Because it eliminates the branch, it allows instructions from different iterations to be scheduled together.

10.What is Branch prediction buffer / Branch history table? (Nov/Dec-2014)
The simplest dynamic branch-prediction scheme is a ***branch-prediction buffer*** **or** ***branch history table***. A branch-prediction buffer is a small memory indexed by the lower portion of the address of the branch instruction. The memory contains a bit that says whether the branch was recently taken or not.

11.What is dynamic scheduling?
Dynamic scheduling, in which the hardware rear- ranges the instruction execution to reduce the stalls while maintaining data flow and exception behavior.

12.List the advantages of Dynamic Scheduling.
It enables handling some cases when dependences are unknown at compile time (for example, because they may involve a memory reference), and it simplifies the compiler.
Perhaps most importantly, it allows the processor to tolerate unpredictable delays such as cache misses, by executing other code while waiting for the miss to resolve.
Almost as importantly, dynamic scheduling allows code that was compiled with one pipeline in mind to run efficiently on a different pipeline.

13.What is an imprecise exception?
An exception is *imprecise* if the processor state when an exception is raised does not look exactly as if the instructions were executed sequentially in strict program order. Imprecise exceptions can occur because of two possibilities:The pipeline may have *already completed* instructions that are *later* in pro- gram order than the instruction causing the exception.The pipeline may have *not yet completed* some instructions that are *earlier* in program order than the instruction causing the exception.

14.List the limitations of loop unrolling.
There are **three different types of limits** to the gains that can be achieved by loop unrolling: A decrease in the amount of overhead amortized with each unrollCode size limitationsCompiler limitations.

15. How many bits are in the (0,2) branch predictor with 4K entries? How many entries are in a (2,2) predictor with the same number of bits?

Answer The predictor with 4K entries has
$20 \times 2 \times 4K = 8K$ bits

16.How many branch-selected entries are in a (2,2) predictor that has a total of 8K bits in the prediction buffer?

We know that $2^2 \times 2 \times$ Number of prediction entries selected by the branch = 8
Hence, the number of prediction entries selected by the branch = 1K.

17. **What is Amdahl's Law? (Apr/May-2015)**

Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used. Law defines the speedup that can be gained by using a particular feature.

Amdahl's Law is used to measure the performance of the multiprocessor:

$$S = \frac{1}{\frac{\text{Fraction enchanced}}{\text{Speedup enchanced}} + (1 - \text{Fraction enchanced})} \quad \ldots(1)$$

18.Define exception precise and imprecise exceptions, repeat interval and initiation interval.

Exception Precise: Precise exception means that all instructions before the faulting instruction are committed and those after it can be restarted from scratch.**Imprecise Exception:** An exception is imprecise if the processor state when an exception is raised does not look exactly as if the instructions were executed sequentially in strict program order. **Repeat Interval or Initiation Interval:** The initiation or repeat interval is the number of that must elapse between issuing two operations of a given type. For example, we will use the latencies and initiation intervals.

19. Define Structural, Data & Control Hazards.

Structural hazards arise from resource conflicts when the hardware cannot support all possible combinations of instructions simultaneously in overlapped execution.**Data hazards** arise when an instruction depends on the results of a previous instruction in a that is exposed by the overlapping of instructions in the pipeline. **Control hazards** arise from the pipelining of branches and other instructions that change the PC.

20. What are two approaches to exploiting ILP?

Static Technique - Software Dependent
Dynamic Technique - Hardware Dependent .

21. Write the 5 levels of branch prediction and define them. (MAY/JUNE 2013)

The five levels of branch prediction are,

Perfect-> all branches and jumps are perfectly predicted at the start of execution.

Tournament-based branch predictor-> the prediction scheme uses a correlating 2-bit predictor and a noncorrelating 2-bit predictor together with a selector, which chooses the best predictor each branch**Standard 2-bit predictor with 512 2-bit entries->** In addition, we assume a 16- entry buffer to predict returns.**Profile-based->** a static

predictor uses the profile history of the program and predicts that the branch is always taken or always not taken based on the profile. **None ->** No branch prediction is used, though jumps are still predicted. Parallelism is largely Limited to within a basic block.

22. Explain about Multithreading. (**Nov/Dec-2014)**

Multithreading computers have hardware support to efficiently execute multiple threads.These are distinguished from multiprocessing systems (such as multi-core systems) in that the threads have to share the resources of a single core: the computing units, the CPU caches and the **translation lookaside buffer (TLB).** Where multiprocessing systems include multiple complete processing units, multithreading aims to increase utilization of a single core by leveraging thread-level as well as instruction-level parallelism.

23. Write the advantages of Multithreading.

If a thread gets a lot of cache misses, the other thread(s) can continue, takingadvantage of the unused computing resources, which thus can lead to faster overallexecution, as these resources would have been idle if only a single thread wasexecuted.If a thread cannot use all the computing resources of the CPU (becauseinstructions depend on each other's result), running another thread permits to notleave these idle. If several threads work on the same set of data, they can actuallyshare their cache, leading to better cache usage or synchronization on its values.

24. Write the disadvantages of Multithreading.

Multiple threads can interfere with each other when sharing hardware resources such as caches or translation lookaside buffers (TLBs).Execution times of a single-thread are not improved but can be degraded, even when only one thread is executing. This is due to slower frequencies and/or additional pipeline stages that arc necessary to accommodate thread-switching hardware. Hardware support for Multithreading is more visible to software, thus requiring more changes to both application programs and operating systems than Multi processing.

25.What is SMT (OR) Simultaneous multithreading? (NOV/DEC’12)

Simultaneous multithreading, often abbreviated as SMT, is a technique for improving the overall efficiency of superscalar CPUs with hardware multithreading. SMT permits multiple independent threads of execution to better utilize the resources provided by modern processor architectures.

26. What are the Disadvantages of SMT?

Simultaneous multithreading cannot improve performance if any of the shared resources are limiting bottlenecks for the performance. In fact, some applications run slower when simultaneous multithreading is enabled. The software developers that they have to test whether simultaneous multithreading is good or bad for their application in various situations and insert extra logic to turn it off if it decreases performance.

27. What are the types of Multithreading?

Block multi-threading

Interleaved multi-threading

28. Write the advantages of heterogeneous multi core architectures?

Efficient adaptation to application diversity
Efficient use of die area for a given thread parallelism

29. What is Multi-Core? (MAY/JUN'12)(APR/MAY'11)(NOV/DEC'11)

At its simplest, multi-core is a design in which a single physical processor contains the core logic of more than one processor. It's as if an Intel Xeon processor were opened up and inside were packaged all the circuitry and logic for two (or more) Intel Xcon processors.The multi-core design takes several such processor "cores" and packages them as a single physical processor. The goal of this design is to enable a system to run more tasks simultaneously and thereby achieve greater overall system performance.

30. Write the software implications of a Multi-Core processor? (APR/MAY'11)

- Multi-core systems will deliver benefits to all software, but especially multi-threaded programs. All code that supports
- HT Technology or multiple processors, for example, will benefit automatically from multi-core processors, without need for modification.
- Most server-side enterprise packages and many desktop productivity tools fall into this category .

31. What is fine grained multithreading? (MAY/JUNE 2013)

It switches between threads on each instruction, causing the execution of multiple Threads to be Interleaved.

32.What is Flynn's classification? (Nov/Dec 14)

Flynn''s classification based on Instruction and Data stream.
It contains four types.1.Single instruction stream, single data stream(SISD)2. Single instruction stream, multiple data stream(SIMD)3.Multiple instruction stream, single data stream(MISD)4.Multiple instruction stream, multiple data stream(MIMD)

33. Compare UMA and NUMA multiprocessors. (April/May-15)

UMA Model:

- For shared address, centralized memory Multiprocessor
- Tightly coupled systems
- Suitable for general purpose and time sharing applications by multiple users.

NUMA Model:

For shared address, distributed memory Multiprocessor.
The access time varies with the location of the memory word
Shared memory is distributed to local memories
All local memories form a global address space accessible by all processors.

34. Differentiate between strong scaling and weak scaling. (April/May-15)

Strong scaling means that at a constant problem size the parallel speedup increases linearly with the number of used processors.Whereas by weak scaling we mean that

the time to solve a problem with increasing size can be held constant by enlarging the number of used processors.Strong scaling is usually limited by Amdahl's law to a certain number of hosts that depends on the problem size. Weak scaling is easier to achieve for hundreds of nodes and is often merely limited by the resources (e.g. memory) per node.

PART-B

1.Explain Instruction level Parallel processing. State the challenges of parallel processing. (Nov/Dec-2014)

Instruction Level Parallelism (Apr/May-2012),(Nov/Dec-13,14)

4.1.1. Concepts and Challenges

Instruction-level parallelism (ILP) is used to overlap the execution of instructions using pipeline concept to improve performance of the system. The various techniques that are used to increase amount of parallelism are reduces the impact of data and control hazards and increases processor ability to exploit parallelism. There are two approaches to exploiting ILP. **1. Static Technique – Software Dependent 2. Dynamic Technique – Hardware Dependent The static technique** is compiler-intensive approach, which have broader adoption in the embedded market than the desktop or server markets, the new IA-64 architecture and Intel's Itanium, use this more static approach. **The dynamic technique** is hardware intensive approach, which dominate the desktop and server markets and are used in a wide range of processors, including: the Pentium III and 4, the Althon, the MIPS R10000/12000, the Sun ultraSPARC III, the Power-PC 603, G3, and G4, and the Alpha 21264. **Pipelining:** Pipelining is an implementation technique where by multiple instructions are overlopped in execution when they are independent of one another.**Stalls**: **pipeline stall** is a delay in execution of an instruction in an instruction pipeline in order to resolve a hazard**Hazards:** In CPU design, **Hazards** are problems with the instruction pipeline in central processing unit (CPU) microarchitectures when the next instruction cannot execute in the following clock cycle, and can potentially lead to incorrect computation results. There are ically three types of hazards. **Types of hazards:Structural Hazard –** An instruction in the pipelineneeds a resource being used by another instruction in the pipeline**Data Hazard –** An instruction depends on a data value produced by an earlier instruction. **Control Hazard –** Whether or not an instruction should be executed depends on a control decision made by an earlier instruction.The simplest and most common way to increase the amount of parallelism is loop-level parallelism. Here is a simple example of a loop, which adds two 1000-element arrays, that is completely parallel:

ILP Vs Parallel Processing

ILP	PARALLEL PROCESSING
• Overlap individual machine operations (add, mul, load…) so that they execute in parallel • Transparent to the user • Goal: Speed up execution	• Having separate processors getting separate chunks of the program • (processors programmed to do so) • Non – transparent to the user • Goal: speed up and quality up

Basics of instruction – level parallelism.

Methods to enhance performance of ILP

Loop Level Parallelism
Techniques to convert LLP to ILP
Vector Instructions

Methods to enhance performance of ILP:

The simplest and most common way to increase the ILP is to exploit parallelism among iteration of a loop.This type of parallelism is often called **Loop- level parallelism**

Loop Level Parallelism The parallelism that exist within a loop is called Loop level parallelism.A loop, which adds two 1000- element arrays, that is completely parallel.

Example:

```
for (i=1;i<=1000; i=i+1)
        x[i] = x[i] + y[i];
```

Every iteration of the loop can overlap with any other iteration, although within each loop iteration there is little or no opportunity for overlap. There are a number of techniques for converting such LLP into ILP are basically work by unrolling the loop either statically by the compiler or dynamically by the hardware.

Techniques to convert LLP to ILP:

Converting loop level parallelism into instruction level parallelism

Loop unrolling: Either the compiler or the hardware is able to exploit the parallelism inherent in the loop**Data dependences (also called true data dependences),Name dependencesAnti dependenceOutput dependenceControl dependences.**

An important alternative method for exploiting loop-level parallelism is the use of vector instructions.

Loop Level Parallelism:

The parallelism that exist within a loop is called Loop level parallelism.A loop, which adds two 1000- element arrays, that is completely parallel.

Ex:

```
for (i=1;i<=1000; i=i+1)
x[i] = x[i] + y[i];
```

Vector Instructions:

A vector instructions operates on a sequence of data items. Vector based processor are used in graphics, digital signal processing and multimedia applications**.**

EX:

```
x[i] = x[i] + y[i];
```

1.Load the X value
2.Load the Y value
3. Add X and Y value
4. Store to result value to X.

Data Dependence and Hazards:

Parallel Instructions:

To exploit instruction-level parallelism, determine which instructions can be executed in parallel. If two instructions are parallel, they can execute simultaneously in a pipeline without causing any stalls.**Dependent instructions:**If two instructions are dependent they are not parallel and must be executed in order. There are three different types of

dependences: **Data dependences (also called true data dependences),Name dependencesControl dependences.Data Dependences: Consider 2 instructions: Instruction i Instruction J**An instruction j is data dependent on instruction i if either of the following holds: • Instruction i produces a result that may be used by instruction j, or • Instruction j is data dependent on instruction k, and instruction k is data dependent on instruction i.For example, consider the following code sequence
That increments a vector of values in memory (starting at 0(R1) and with the last element at 8(R2)) by a scalar in register F2:

EXAMPLE

```
Loop: L.D      F0,0(R1)        ;  F0=array element
ADD.D          F4,F0,F2        ;  add scalar in F2
S.D            F4,0(R1)        ;store result
DADDUI         R1,R1,#-8       ;decrement pointer 8 bytes
BNE            R1,R2,LOOP      ;  branch !=zero
```

The data dependences in this code sequence involve both floating point data:

```
Loop:   L.D    F0,0(R1)        ;F0=arrayelement
ADD.D          F4,F0,F2        ;add scalar in F2
S.D            F4,0(R1)        ;store result and integer data:
DADDIU         R1,R1,-8        ;decrement pointer
                               ;8 bytes (per DW)
BNE            R1,R2,Loop      ; branch R1!=zero
```

Both of the above dependent sequences, the arrow showsach instruction depending on the previous one. The arrows here and in following examples show the order that must be preserved for correct execution. The arrow points from an instruction that must precede the instruction that the arrowhead points to.
The importance of the data dependences:
(1) Indicates the possibility of a hazard,
(2) Determines the order in which results must be calculated, and
(3) Sets an upper bound on how much parallelism can possibly be exploited.

4.1.4..Name Dependences:

The name dependence occurs when two instructions use the **same register or memory location**, called a name, but there is no flow of data between the instructions associated with that name. There are two types of name dependences between an instruction i that precedes instruction j in program order: **An anti dependence** between instruction i and instruction j occurs when instruction j writes a register or memory location that instruction i reads. The original ordering must be preserved to ensure that i reads the correct value. **An output dependence** occurs when instruction i and instruction j write the same register or memory location. The ordering between the instructions must be preserved to ensure that the value finally written corresponds to instruction j.Both anti-dependences and output dependences are name dependences, as opposed to true data dependences, since there is no value being transmitted between the instructions. **Register renaming:**
This renaming can be more easily done for register operands, where it is called register renaming. Register renaming can be done either statically by a compiler or dynamically by the hardware.

Data Hazard with Types:

Data Hazards:
A hazard is created whenever there is dependence between instructions, and they are close enough that the overlap caused by pipelining, or other reordering of instructions, would change the order of access to the operand involved in the dependence.

Classification of Hazards:
Data hazards may be classified as one of three types, depending on the order of read and write accesses in the instructions.

RAW (read after write)
WAW (write after write)
WAR (write after read)Consider two instructions i and j, with i occurring before j in program order. RAW (read after write):
j tries to read a source before i writes it, so j incorrectly gets the old value. This hazard is the most common type and corresponds to a true data dependence. Program order must be preserved to ensure that j receives the value from i. **WAW (write after write):**j tries to write an operand before it is written by i. The writes end up being performed in the wrong order, leaving the value written by i rather than the value written by j in the destination. This hazard corresponds to an output dependence.**WAR (write after read):**j tries to write a destination before it is read by i, so i incorrectly gets the new value. This hazard arises from an anti dependence. WAR hazards cannot occur in most static issue pipelines even deeper pipelines or floating point pipelines because all reads are early (in ID) and all writes are late (in WB). **Control Dependences:** A control dependence determines the ordering of an instruction, i, with respect to a branch instruction so that the instruction i is executed in correct program order. Every instruction, except for those in the first basic block of the program, is control dependent on some set of branches, and, in general, these control dependences must be preserved to preserve program order.

EX:

if p1 {

S1;

};

if p2 {

S2;

}

S1 is control dependent on p1, and S2is control dependent on p2 but not on p1.

Control Dependences Constraints:

An instruction that is control dependent on a branch cannot be moved **before** the branch so that its execution is **no longer controlled** by the branch. **Ex:** we cannot take an instruction from the then-portion of an if-statement and move it before the if-statement..An instruction that is not control dependent on a branch cannot be moved after the branch so that its execution is controlled by the branch. **Ex:** we cannot take a statement before the if-statement and move it into the then-portion.

Preserving Control dependence:

Control dependence is preserved by two properties in a simple pipeline, First, instructions execute in program order.Second, the detection of control or branch hazards ensures that an instruction that is control dependent on a branch is not executed until the branch direction is known.

ILP ArchitecturesComputer architecture : is a contract (instruction format and the interpretation of the bits that constitute an instruction) between the class of programs that are written for the architecture and the set of processor implementation of that architecture.**ILP Architecture**: information embedded in the program pertaining to available parallelism between instructions and operation in the program**ILP Architecture Classifications:Sequential architectures:** The program is not expected to convey any explicit information regarding parallelism **(superscalar Processors)Dependence Architecture:** The program explicitly indicates the dependences that exist between operations **(Data flow processors)Independence Architecture:** The program provides information as to which operations are independent of one another **(VLIW Processors).**

1. Explain parallel processing challenges with example:

parallel processing challenges

Parallel processing challenges are explainable with **Amdahl' s law.**

TTwo important hurdles in parallel processing which are difficult or easy Determined by the application and by the architectureThe **first major challenge** in parallel processing arises with the limited parallelis available in program.Limitation in available parallelism make it difficult to achieve **good speedups** in any parallel processor.

Solution:

Amdahl's Law is used to measure the performance of the multiprocessor:

$$\text{Speedup} = \frac{1}{\quad\quad} \quad \ldots..(1)$$

Fraction enchanced
+ (1- Fraction enchanced)
Speedup enchanced

Assume that the program operates in two modes:

1. Parallel- All processors fully used, which is enchanced mode.
2. Serial- Only one processor in use.Speedup in enchanced mode is the time spent in parallel mode.

Subtitute the speedup value in equation(1)

$$80= \frac{1}{\frac{\text{Fraction parallel}}{100} + (1- \text{Fraction parallel})} \quad \ldots.(2)$$

Simplifying the equation(2) gives,

0.8=Fraction parallel + 80 × (1- Fraction parallel)=1

80- 79.2 × (Fraction parallel)=1

$$\text{Fraction parallel}= \frac{80-1}{79.2}$$

Fraction parallel= 0.9975.

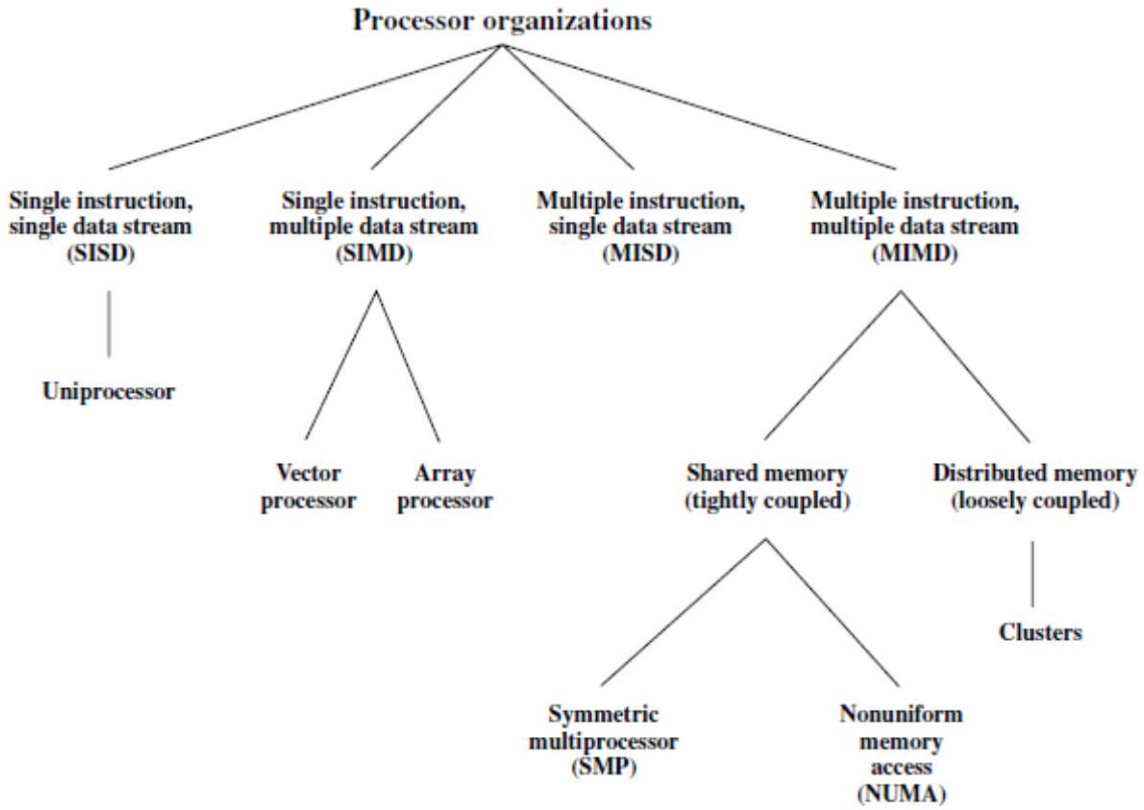
Hence to achieve a speedup of 80 with 100 processors, only 0.25% of original computation can be sequential.To achieve linear speedup(speedup of n with n processors) the entire programs must usually be parallel with no serial portions. 1.Programs do not operate in fully parallel or sequential mode, but often use less than the full complement of the processor when running in parallel mode.2.The second major challenge in parallel processing involves the large latency of remote access in a parallel processor.The communication of data between processors in the exiting shared memory multiprocessor may cost anywhere from 50 clock to over 1000 clock cycles. This cost is mainly depend on the communication mechanism, the type of interconnection network and the scale of the multiprocessor. The effect of long communication delays is more substantial.**CPI=Base CPI +Remote request rate × remote request cost**

LONG LATENCY TO REMOTE MOMORY:Exampl Suppose we have an application running on a 32-processor multiprocessor, which has a 200 ns time to handle reference to remote memory .For those application ,assume that all the references except those involving communication hit in the local memory hierarchy, which is slightly Optimistic .processors are stalled on a remote request , and the processor clock rate is 2GHz. If the base CPI (assuming that all preference hit in the cache)is 0.5, how much ster is the multiprocessor if there is no communication versus if 0.2% of the instructions involve a remote communication reference? Thus the multiprocessor with all local reference is 1.3/0.5=2.6 times faster. In practice, the performance analysis is much more complex, since some fraction of the non-

communication references will miss in the local hierarchy and the remote access time does not have a single constant value. For example, the cost of a remote reference could be quite a bit worse , since contention caused by many references trying to use the global interconnect can lead to increased delays.**Two Biggest Challenges** (1) Insufficient parallelism (2) Long- latency remote communicationThe problem of inadequate application parallelism must be attacked primarily in software with new algorithm that can have better parallel performance.Reducing the impact of long remote latency can be attacked both by the architecture and by the programme.For example, we can reduce the frequency of remote accesses with either hardware mechanisms, such as caching shared data, or software mechanisms, such as restricting the data to make more accesses local. We can try to tolerate the latency by using multithreading or by using pre- fetching. mechanisms, such as caching shared

3.Explain in detail about the flynn's taxonomy?(or) Discuss about SISD,MIMD,SIMD,SPMD and Vector systems. (April/May-15)

Flynn's taxonomyOver view:
Processor organization
Flynn's taxonomy Classification
Processor organization
Multiprocessor is a tightly coupled computer system having two or more processing units (**Multiple Processors**) each sharing main memory and peripherals, in order to simultaneously process programs. Multiprocessing is a type of processing in which two or more processors work together to execute more than one program simultaneously, the term Multiprocessor refers to the hardware architecture that allows multiprocessing.**Use:**Multiprocessors are used to increase performance and improving availability.**ParallelArchitechture:**A parallel computer is a collection of processing elements that cooperate and communicate to solve large problems fast in a parallel manner.



Figurer 4.1 .A Taxonomy of Parallel Processor Architectures

Classification:

1. Flynn's taxonomy
2. Classification based on the memory arrangement
3. Classification based on communication

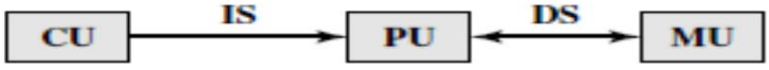
   Classification based on the kind of parallelism

Flynn's Taxonomy:

1.Single instruction stream, single data stream(SISD)
2. Single instruction stream, multiple data stream(SIMD)
3.Multiple instruction stream, single data stream(MISD)
4.Multiple instruction stream, multiple data stream(MIMD)

Single instruction stream, single data stream(SISD):

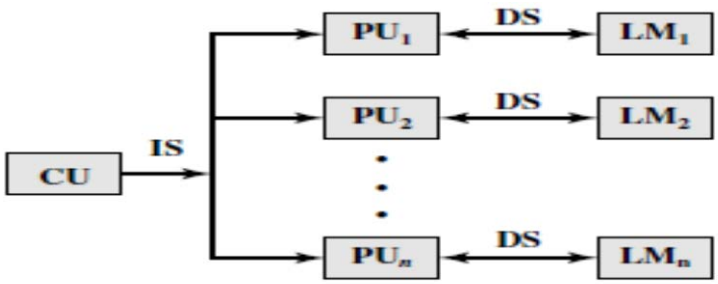
A single processor executes a single instruction stream to operate on data stored in a single memory.This category is the uniprocessor**.**



(a) SISD

Single instruction stream, multiple data stream(SIMD):

The same instruction is executed by multiple processors using different data stream
Data level parallelisEach Processor has its own data memoryThere is a single instruction memory and control processor. **Vector and array processors fall into this category,**

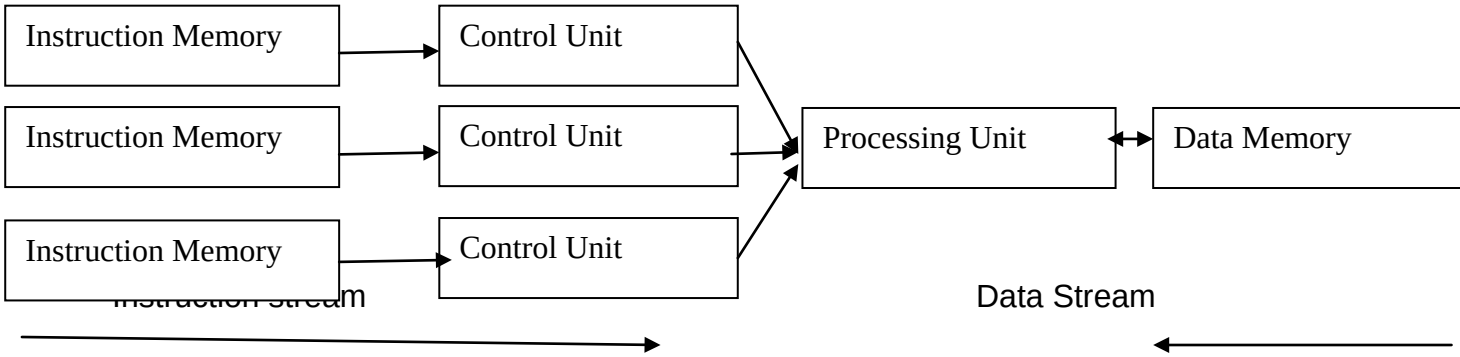


(b) SIMD (with distributed memory)

CU = Control unit
IS = Instruction stream
PU = Processing unit
DS = Data stream
MU = Memory unit
LM = Local memory

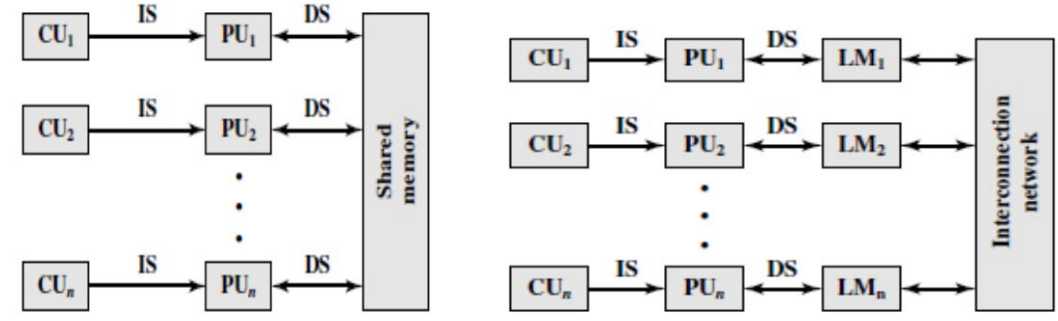
Multiple instruction streams, single data stream (MISD):

A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence.This structure is not commercially implemented.



Multiple instruction stream, multiple data stream(MIMD):

- A set of processors simultaneously execute different instruction sequences on different data sets. SMPs, clusters, and NUMA systems fit into this category.Each processor fetches its own instructions and operates on its own dataThread level Parallelism.It can be divided into two types
  - Shared memory
  - Distributed memory



(c) MIMD (with shared memory)

(d) MIMD (with distributed memory)

CU = Control unit
IS = Instruction stream
PU = Processing unit
DS = Data stream
MU = Memory unit
LM = Local memory

Figure 4.5. Alternative Computer Organizations

SPMD: Single Program Multiple Data (SPMD) StructureSingle source program is written and each processor will execute its personal copy of this program, although independently and not in synchronism.The source program can be constructed so that parts of the program are executed by certain computers and not others depending upon the identity of the computer.Software equivalent of SIMD; can perform SIMD calculations on MIMD hardware.

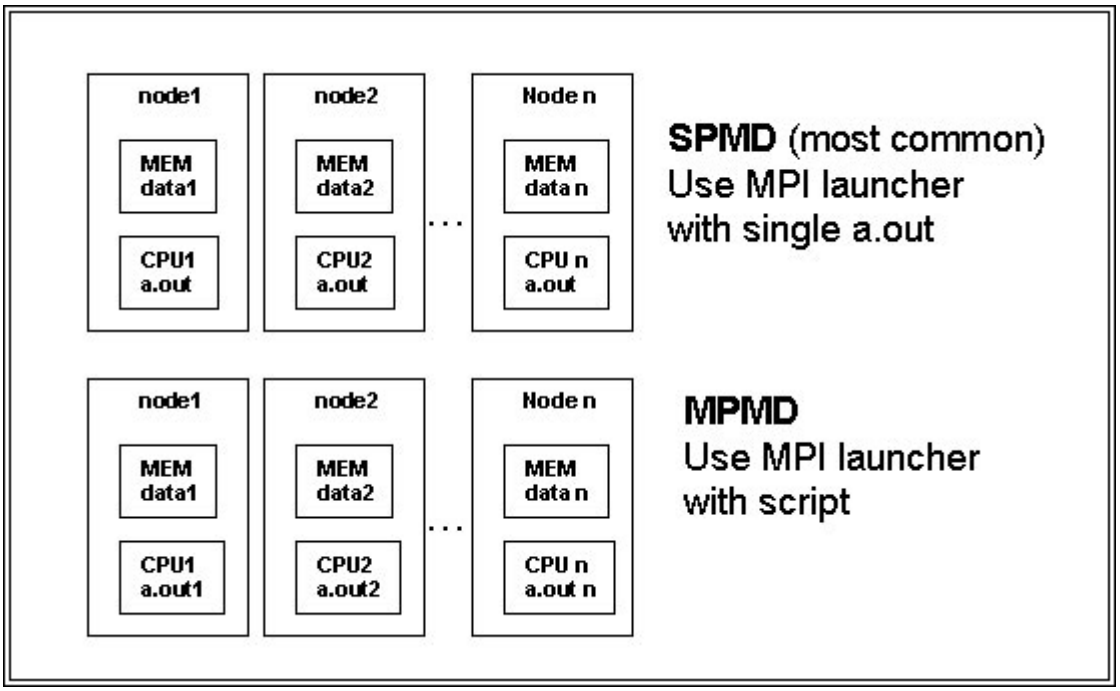


Figure .4.6.SPMD and MPMD

MPMD:

Multiple Program, Multiple Data: multiple autonomous processors simultaneously operating at least 2 independent programs. Typically such systems pick one node to be the "host" ("the explicit host/node programming model") or "manager" (the "Manager/Worker" strategy), which runs one program that farms out data to all the other nodes which all run a second program. Those other nodes then return their results directly to the manager. **VECTOR systems.**Vector computer: instruction set includes operations on vectors as well as scalars Two ways to implement vector computersPipelined vector processor (e.g. Cray): streams data through pipelined arithmetic unitsProcessor array: many identical, synchronized arithmetic processing elements

Classification based on the memory arrangement:

Centralized shared memory(UMA-nuiform memry access)

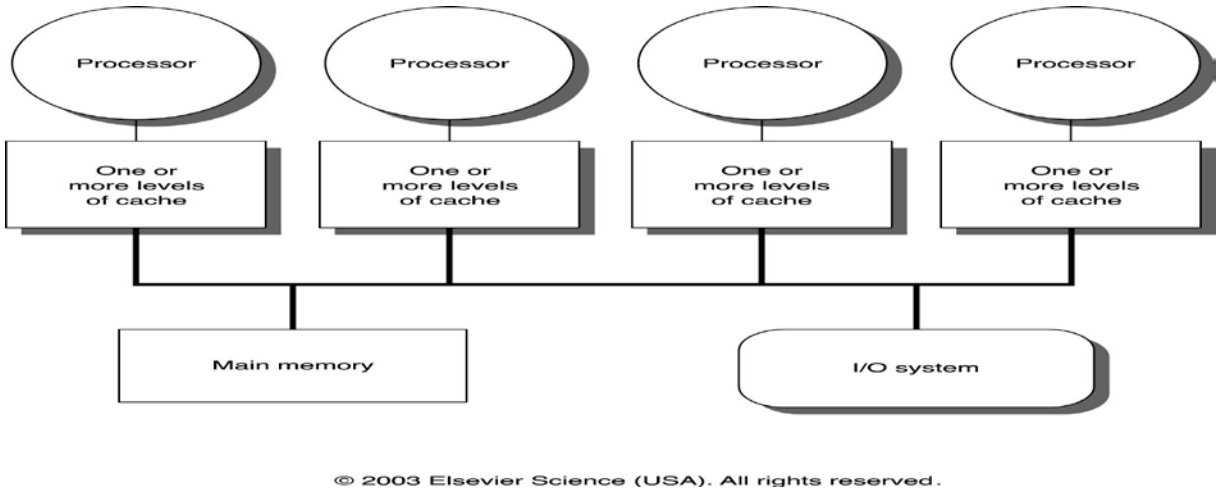


Figure 4.7.Basic structure of a centralized shared – memory multi processor.

Multiple processor- cache subsystems share the same physical memory, typically connected by a bus.

In large design, multiple buses, or even a switch may be used, but the key architecture property: Uniform access time I/O all memory from all processor remains.

Advantage:

Large caches can satisfy the memory demands of a small number of processors.

Disadvantages:

Scalability problem: less attractive for large processors.

2.Physically Distributed memory multiprocessor

Each directory is responsible for tracking the caches that share the memory address of the portion of memory in the node

Figure 4.8.A directory is added to each node to implement cache coherence in a distributed – memory multiprocessor.

Advantages:
Cost effective way to scale memory bandwidth
Lower memory latency for local memory access
Disadvantage:
Longer communication latency for communicating data between processors
Software model more complex.
3. Communication models and memory architecture:1.**Distributed shared memory**
DSM is a multiprocessor with a shared address space in which communication occurs through a shared space(via loads and store operation)

Shared Memory Multiprocessors

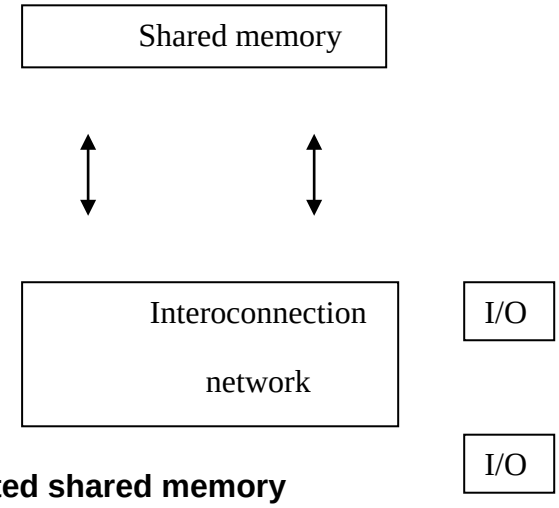


Figure 4.9.Distributed shared memory

UMA Model:

For shared address, centralized memory Multiprocessor
Tightly coupled systems
Suitable for general purpose and time sharing applications by multiple users.

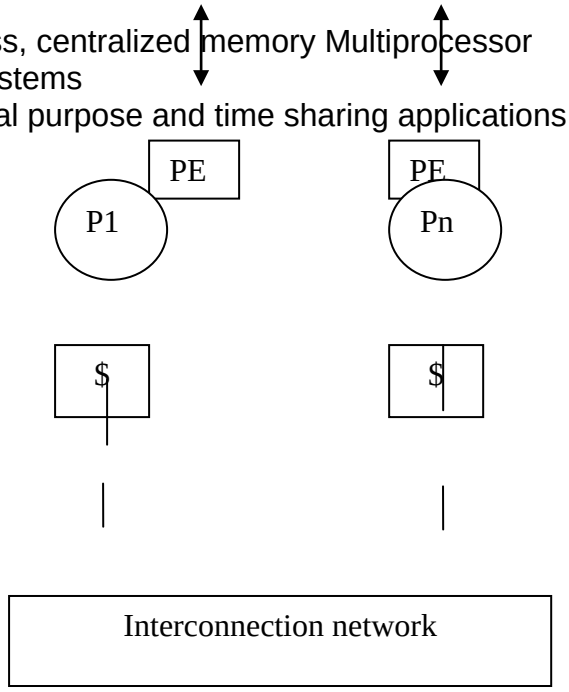
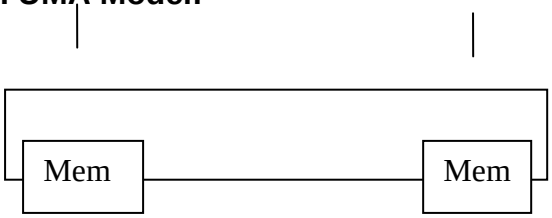


Figure 4.10. UMA Model:

NUMA Model:



For shared address, distributed memory Multiprocessor.
The access time varies with the location of the memory word
Shared memory is distributed to local memories

All local memories form a global address space accessiblle by all processors

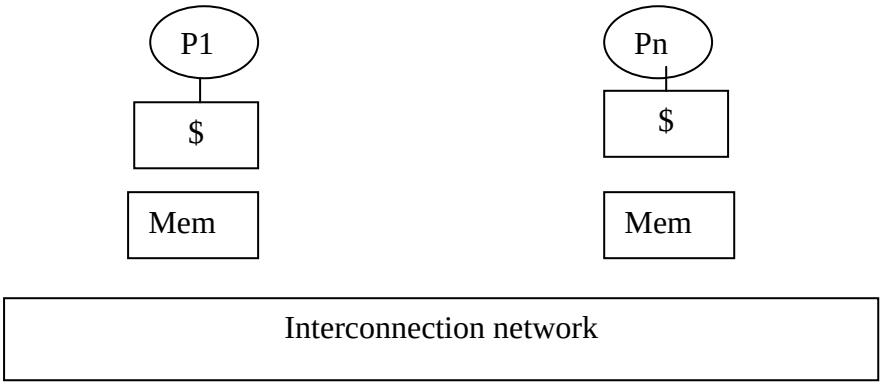


Figure 4.11.NUMA Model:

Message Passing Multiprocessors:

Message passing multiprocessor is a multiprocessor with multiple address space which communicaton of data occurs by explicitly passing messages among the processors.**Multiple address space:**The same physical address on two different processors refers to two different locations in two different memories.1.**Synchronous Message passing**- It is a way in which the initiating processors sends a request and waits the reply is returned before continuing.2. **Asynchronous Message passing**: It is a way in which the initiating processor sends the request message continuously without waiting for the reply message.

Performance metrics for Communication Mechanisms:

1.**Communication bandwidth**-It is limited by processor, memory, and interconnection bandwidths.2.**Communication latency**=Sender overhead+Time of flight+Transmission time+Receiver overhead.3.**Communication latency hiding**= The running time on multiprocessors with the same communication latency but different support for latency hiding. Also the amount of latency that can be hidden is application dependent.

Advantages of Different Communication Mechanisms:

Advantages of shared Memory communication Model:

Ease of programming when communication patterns are complex or vary dynamically during execution.Ability to develop applications using familar SMP model, attention only on performance critical accesses.Lower communication overheadHardware controlled caching

Advantages of Message Passing Communication Model:

The H/W can be simpler
Communication explicit
Synchronization
Easier to use sender initiated communication

4. Explain in detail about the software and hardware multithreading. & Compare and contrast fine grained multi threading and coarse grained multi threading.(Nov/Dec 14),(April/May-15)

The Software And Hardware Multithreading.Over View

Software multithreading

Hardware multithreading:

Software Multithreading:

Software multithreading is a piece of software that is aware more than one core/processor and can use these to be able to simultaneously complete multiple tasks.On single core, multi-threading can only be used to hide latency.Multi-threaded applications running on multi-core platforms have different design considerations than do multi-threaded applications running on single-core platforms.On single core can make assumptions to simplify writing and debugging.these may not be valid on multi-core platforms. areas like memory caching and thread priority.

Advantage:

Increased responsiveness and worker productivity

Increased application responsiveness when different tasks run in parallel.

Improved performance in parallel environments

When running computations on multiple processors

More computation per cubic foot of data center

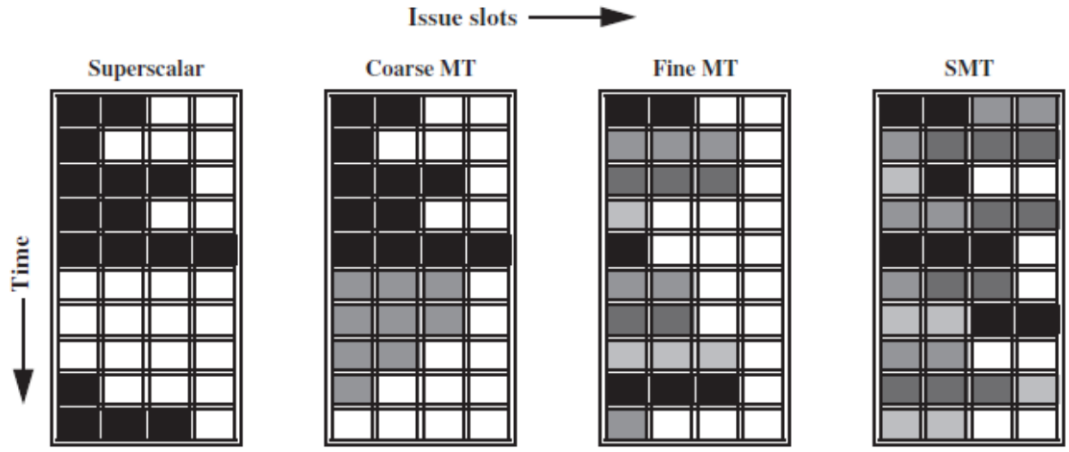
Web based apps are often multi-threaded by nature.

Hardware Multithreading:

Hardware multithreading is a multithreading that allow multiple threads to share a the functional units of a single processor in an overlapping fashion.The processor must duplicate the independent state of each threadRegister filePCMemory can be shared through the virtual memory mechanism.Hardware supports fast thread switching.

Approaches:

The two main approaches in hardware multithreading are**Fine-grain multithreading:**Switch threads after each cycleUsing a round-robin fashion Interleave instruction executionIf one thread stalls, others are executed. **Coarse grain multithreading:** Only switch on long stall(L2-cache miss) Simplifies hardware, but doesn`t hide short stalls(data hazards).**Simultaneous Multithreading:** SMT is a variation on multithreading that uses resources of a multiple-issue. Dynamically scheduled processors to exploit TLP at the sometime it exploits ILP i.e., convert thread level parallelism into more ILP.**It exploits the following features of modern processors: Multiple functional units:**Modern processors typically have more functional units available than a single thread can utilize. **Register renaming and dynamic scheduling**Multiple instruction from independent threads can co exit and co execute. **Illustration:**The following diagram illustrates the differences in a processor`s ability to exploit the resources of a superscalar for the following processor configuration.A superscalar processor with no multithreading supportA superscalar processor with coarse grain multithreading A superscalar processor with fine grain multithreading A superscalar processor with simultaneous multithreading(SMT)



Horizontal dimension represents the instruction issue capability in each clock cycle.Vertical dimension represents a sequence of clock cycles.Empty slots indicate that the corresponding issue slots are unused in that clock cycle.

Superscalar processor with no multithreading support: The use of issue slots is limited by a lack of ILP. Stalls such as an instruction cache miss leaver the entire processor idle.

Superscalar processor with coarse grain multithreading: The long stalls are partially hidden by switching to another thread that uses the resources of the processor.Reduced the number of completely idle clock cycles.But, the ILP limitations still lead to idle cycles.

Superscalar processor with fine grain multithreading: The interleaving of threads eliminates fully empty slots. The ILP limitations still lead to significant number of time slots within individual clock cycles because only one thread issues instructions in a given clock cycles. **Superscalar processor with simultaneous multithreading(SMT):** The thread level parallelism (TLP) and instruction level parallelism (ILP) are exploited simultaneously with multiple threads using the issue slots in a single clock cycle.**The issue slot usage is limited by the following factors**Imbalances in the resource needssource availability over multiple thrNumber of active threads consideredFinite limitation of buffer Ability to fetch enough instruction from multiple threads.Practical limitations of what instructions combinations can issue from one thread and multiple threads.**Design Challenges In SMT:**

Impact of fine grained scheduling on single thread performance:A preferred thread approach sacrifices neither throughput nor single thread performance?Unfortunately with a preferred thread, the processor is likely to sacrifice some throughput, when preferred thread stalls.Pipeline is less likely to have a mix of instruction from several threads resulting in greater probability that either empty slots or a stall will occur.**Design Challenges: The design challenges of SMT processor include the following**Larger register file needed to hold multiple contextsNot affecting clock cycle time, especially in Instruction issue more candidate instructions need to be consideredInstruction completion choosing which instructions to commit may be challenging Ensuring that cache and TLB conflicts generated by SMT do not degrade performance.**Observations:** The following two observation that are made with respect to these problems arePotential performance overhead due to multithreading is small.Efficiency of current superscalar is low with the room for significant improvement.**Works well if**Number of compute intensive threads does not exceed the number of threads supported in SMT. Threads have highly different characteristics(one thread doing mostly integer operations, another mainly doing floating point operations)**Does not work well if**Threads try to utilize the same function units Assignment problems:A dual

processor system each processor supporting 2 threads simultaneously (OS thinks there are 4 processors)2 compute intensive application processes might end up on the same processor instead of different processors(OS does not see the difference between SMT and processors!)

Potential performance advantages from SMT:

The following table shows the features of an aggressive SMT design incorporated into an aggressive superscalar for the evaluation of an SMT expensing starts with an aggressive superscalar that has roughly double the capacity of existing superscalar processor.

Processor characteristic	capability
Integer functional units	6(include 4 loads/stores per cycle)
Pipeline depth	9 stages
Floating-point functional units	4
Instruction	32
Renaming registers	100 each for integer and floating point
Commit capability	Upto 12 instruction and floating point
TLB	128 entries each for instructions and data
Primary instruction cache	128 KB,2-way set associative, single ported, 2-cycle Penalty, 32 outstanding misses.
Primary data cache	128 KB, 2-way set associative, dual ported, 2-cylefill penalty.
L2 cache	16 MB, direct-mapped,20-cycle latency, fully pipelined.
L1-L2 bus/refill	256 bits wide, 2-cycle latency
Store buffer	32 entries
Memory system	90-cycle latency, full pipelined
Branch-target buffer	1K entries, 4-way set associative
Hardware context for SMT	8
Fetch policy	8 instruction pe clock.

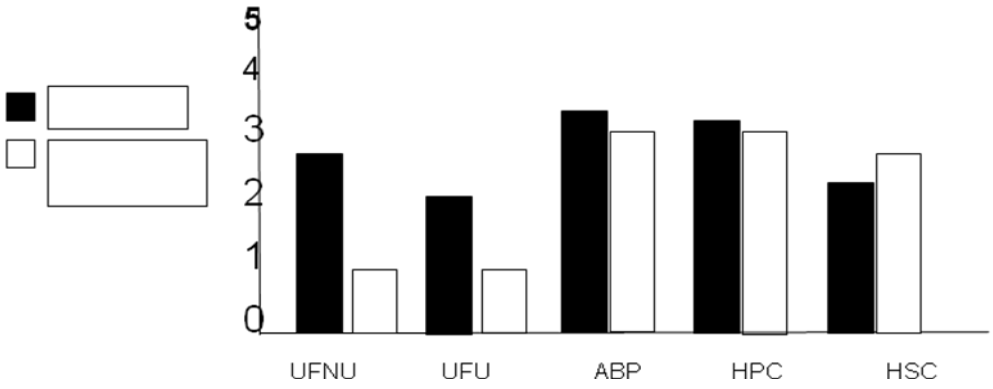
Performance advantage:

The following diagram shows the advantage in throughput, which is measured as instructions per clock, of SMT with eight contexts to the superscalar processor.

Comparison of the SMT processor to the base superscalar processor: The SMT processor are compared to the base superscalar processor in several key measures.

Utilization of functional units
Utilization of fetch units
Accuracy of branch predictor
Hit rates of primary caches
Hit rates of secondary caches

Comparison of the SMT processor to the base superscalar processor:



o

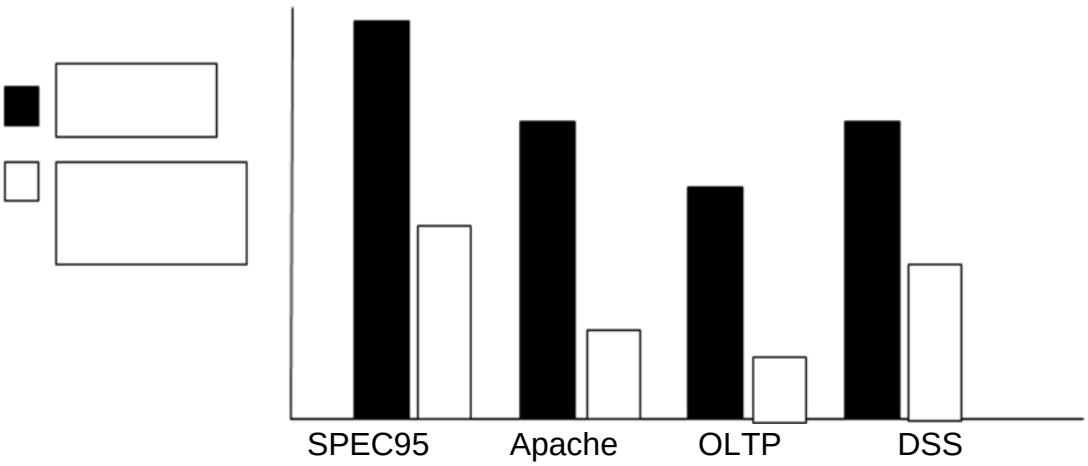
Performance improvement:

The key to maximize SMT performance is to share the following

Issue slots

Function units

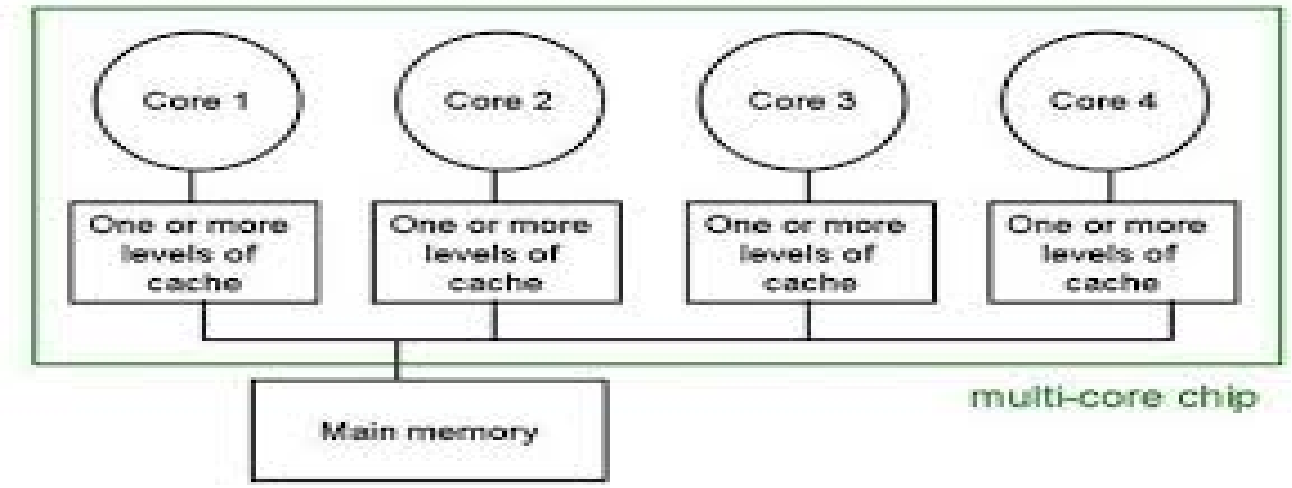
Renaming registers



5.Briefly explain about multi-core Processors with neat diagram.(Nov/Dec-14)

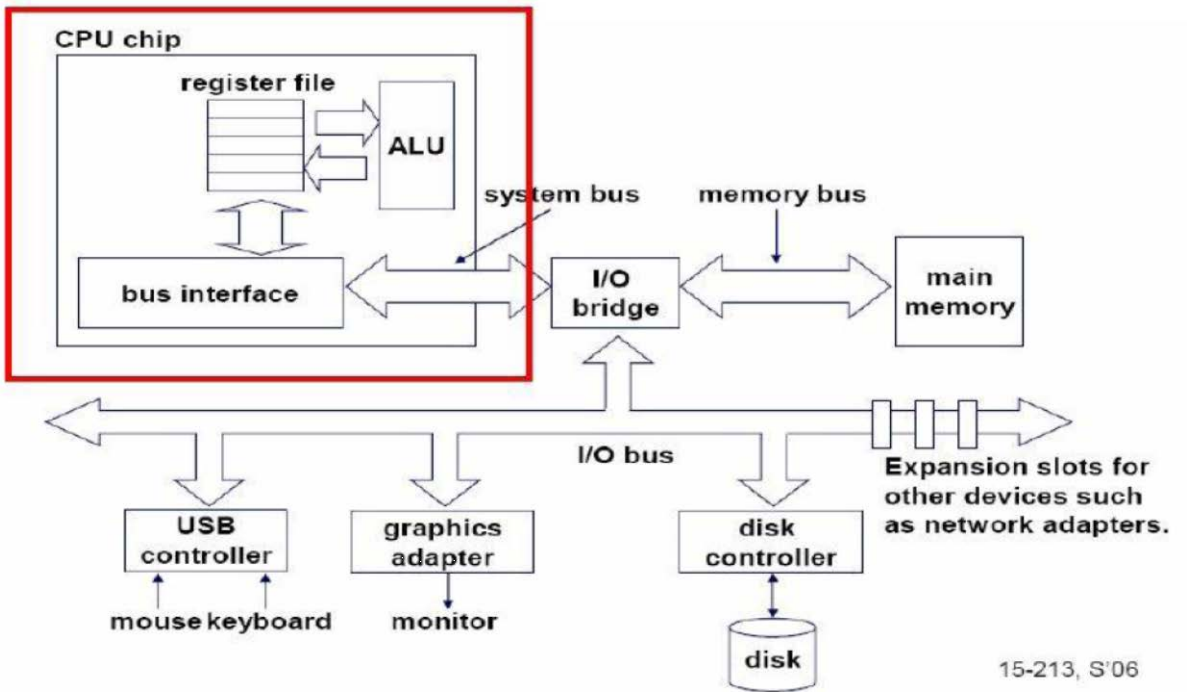
It integrates two or more independent cores (normally a CPU) into a single package composed of a single integrated circuit(IC), called a die, or more dies packaged together, each executing threads independently.Every functional unit of a processor is duplicated.

A dual-core processor contains two cores, and a quad-core processor contains four cores.

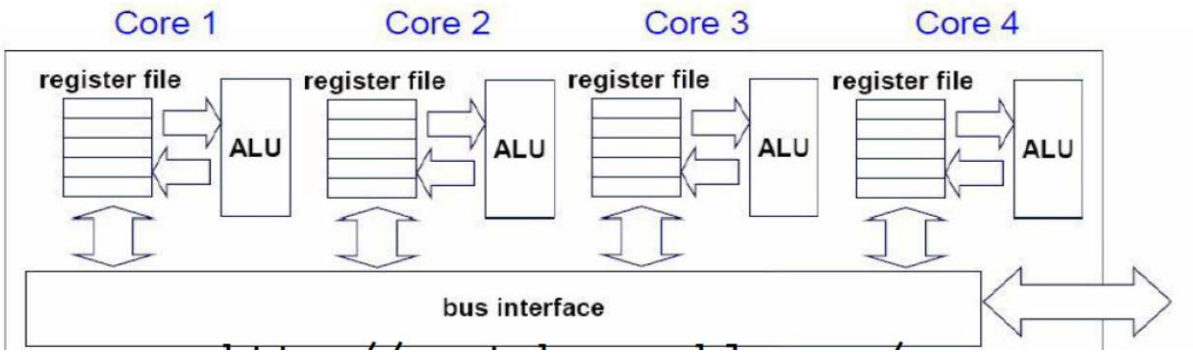


All core shares the same memoryBut in each core contains Local Memory.

Single Core CPU chip:



Multicore CPU chip:



3.5. Multi-Core Processors

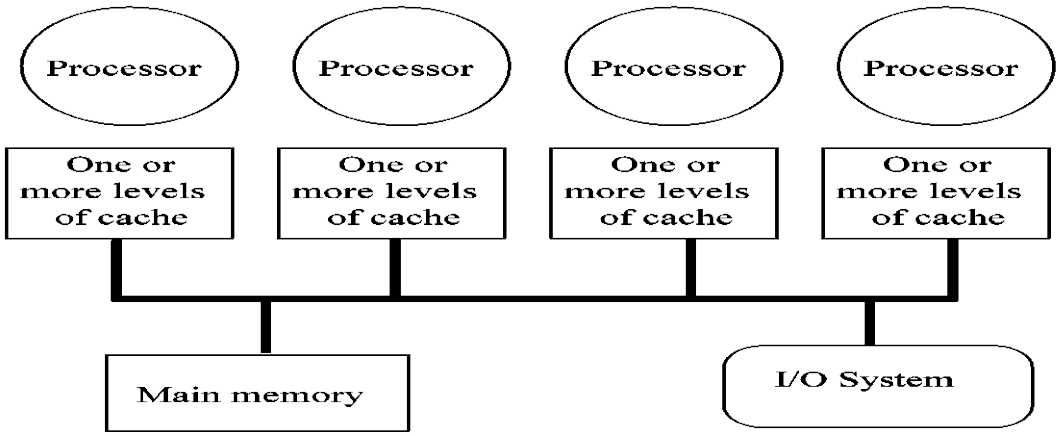
Major MIMD Styles:

1. Centralized shared memory ("Uniform Memory Access" time or "Shared Memory Processor")
2. Decentralized memory (memory module with CPU)

Advantages: Scalability, get more memory bandwidth, lower local memory latency.
Drawback: Longer remote communication latency, Software model more complex

Two types: Shared Memory and Message passing

1.Centralized shared memory Architecture:



4.12.Figure Basic structure of a centralized shared – memory multi processor.

- Memory: centralized with uniform access time (“UMA”) and bus interconnect.
- Examples: Sun Enterprise 5000 , SGI Challenge, Intel SystemPro

Multiple processor- cache subsystems share the same physical memory, typically connected by a bus.In large design, multiple buses, or even a switch may be used, but the key architecture property: Uniform access time I/O all memory from all processor

remains.**Advantage:**Large caches can satisfy the memory demands of a small number of processors**Disadvantages:**
Scalability problem: less attractive for large processors.

Distributed – Memory multicore Processor:

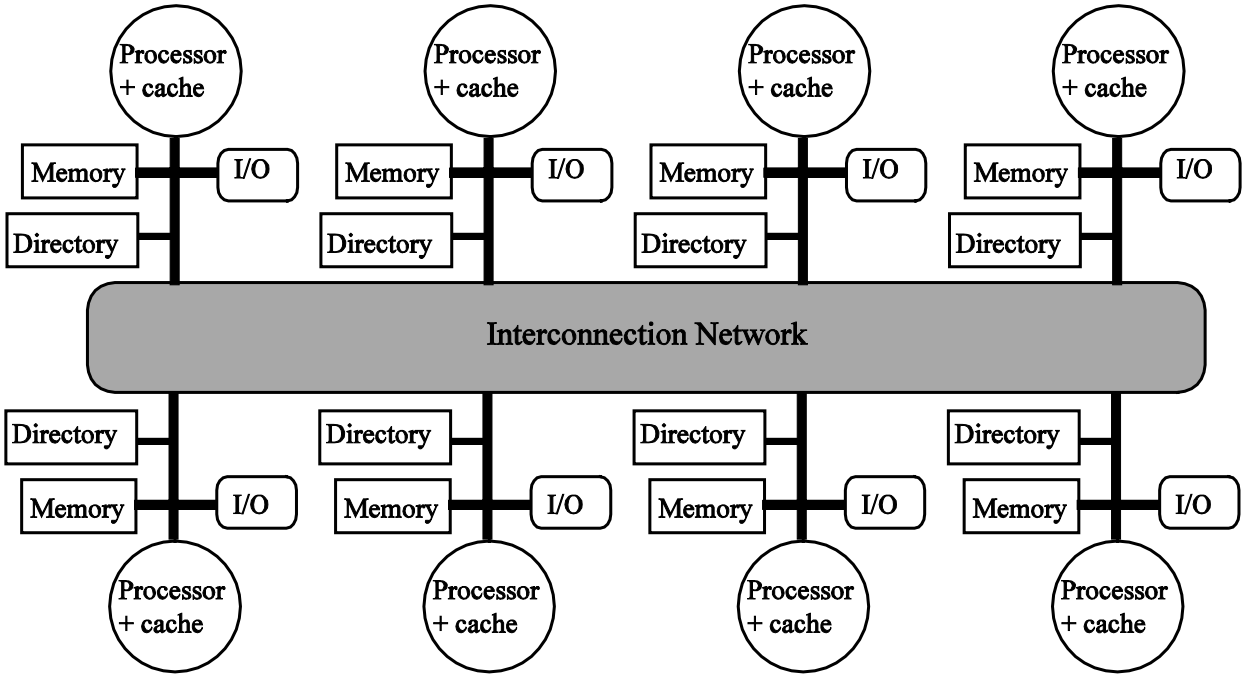


Figure.4.13. A directory is added to each node to implement cache coherence in a distributed – memory multiprocessor.

Each directory is responsible for tracking the caches that share the memory address of the portion of memory in the node

Advantages:

- Cost effective way to scale memory bandwidth
- Lower memory latency for local memory access

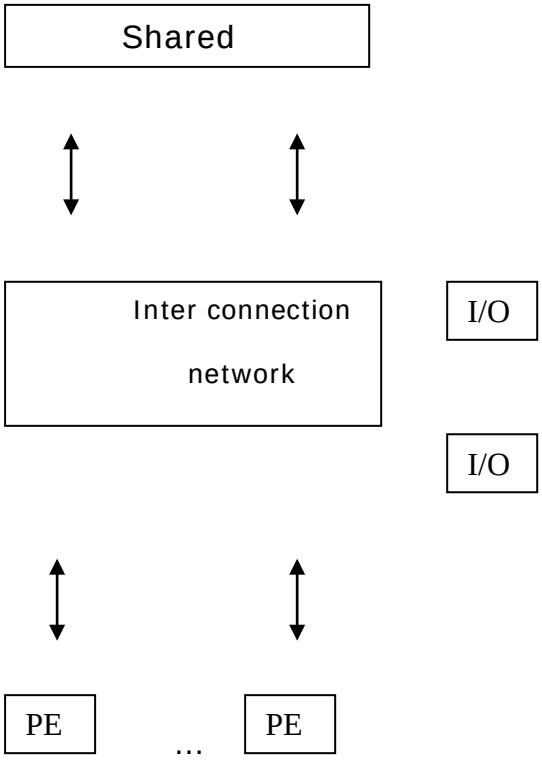
Disadvantage:

- Longer communication latency for communicating data between processors
- Software model more complex.

Communication Models

1.**Distributed shared memory:**

DSM is a multiprocessor with a shared address space in which communication occurs through a shared space(via loads and store operation)



Figurer 4.14.Distributed Shared Memory Multiprocessors

Shared address space: The address space that is shared i.e., the same physical address on two processor refers to the same location in memory.**Uniform memory access (UMA): Non uniform memory access (NUMA):**

Unit V

Memory And Input And Output

Part-A

1.What is memory system?

Every Computer contains several types of devices to store the instructions and data for its operation. These storage devices plus the algorithm implements by hardware and software needed to manage the stored information from the memory system of computer.

2.Write the classification of memory.

a. CPU Register
b. Main memory
c. Secondary Memory
d. Cache.

3.Define Static Memories and Dynamic Memories.

Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memories. In Dynamic Memories such cells do not retain their state indefinitely.

4. What is read access time?

A basic performance measure is the average time to read a fixed amount of information for instance, one word from the memory. This parameter is called the read access time.

5. Define RAM.

In storage location can be accessed in any order and access time is independent of the location being accessed, the memory is termed as random access memory

6. Difference between static RAM and Dynamic RAM. (MAY/JUNE 2009)

S.no	STATIC RAM	DYNAMIC RAM
1.	They are fast	They are slow
2.	They are very expensive	They are less expensive
3.	They require several transistors	They require less no several transistors
4.	They retain their state indefinitely	They do not retain their state indefinitely

7. Differentiate asynchronous DRAM with synchronous DRAM?

S.no.	Asynchronous DRAM	Synchronous DRAM
1.	The timing of the memory device is controlled asynchronously	The timing of the memory device controlled synchronously.
2	Here specialized memory controlled circuit that provides the necessary control information	The memory operations are synchronized with a clock signals.
3	Separate refresh circuit is not used	It uses the separate refresh circuit.

8. List the differences between SRAM AND DRAM? (APRIL/MAY 2008)
SRAM: Static random access memory. It tends to be faster and they require no refreshing.
DRAM: Dynamic random access memory. Data is stored in the form of charges. So continuous refreshing is needed.

9. What is volatile memory?
A memory is volatile if the loss of power destroys the stored information. Information can be stored indefinitely in a volatile memory by providing battery backup or other means to maintain a continuous supply of power.

10. What are the types of memory (or) What are the categories of memories? (MAY/JUNE 2013)
SRAM (Static Random Access Memory)
DRAM (Dynamic Random Access Memory)

11. What is flash memory.
Flash memory is a memory storage device for computers and electronics. It is most often used in devices like digital cameras, USB flash drives, and video games. It is quite similar to EEPROM.. Flash can keep its data intact with no power at all. It is much used in small electronics because it is small and has no moving parts.

12. What is RAMBUS memory?
The key feature of Rambus technology is a fast signaling method used to transfer information between chip using narrow bus.

13. What is write-through protocol?
For write operation, the cache location and the main memory location are updated simultaneously.

14. Give the difference between EEPROM and Flash memory?
The primary difference between EEPROM and flash memory is that flash restricts writes to multiple kilobytes blocks, increasing the memory capacity per chip by reducing area of control.

15. Differences between cache memory and virtual memory.
In caches, replacement is primarily controlled by the hardware. In VM, replacement is primarily controlled by the OS.
The Number of bits in the address determines the size of VM, where as cache size is independent of the address size.

16. What is meant by Interleaved Memory Or What is memory interleaving? (MAY/JUNE 2012 & 13, NOV/DEC 2010)

Banks of memory are often one word wide , so bus width need not be changed to access memory. However several independent areas of memory can be accessed simultaneously by using interleaved memory.

17. What is write back protocol?
In this scheme, only the block in cache is modified. The main memory when the block must be replaces in the cache. This requires the use of a dirty bit to keep track of blocks, that have been modified.

18. What is virtual memory and what are the benefits of virtual memory?(APRIL/MAY 2010) (MAY/JUNE 2009)
irtual memory is a technique is used to extend the apparent size of the physical memory.
It uses secondary storage such as disks, to extend the size of physical memory.Techniques that automatically move program and data blocks into the physical memory when they are required for execution are called **virtual memory techniques.**The parts of it not currently being executed are stored in secondary memory and transferred to main memory when it is required**Benefits**Store the data more than the size of the physical memoryEntire program need not be in main memory

19. Give the features of a ROM cell (APRIL/MAY 2008)
It is usually non-volatile memory meaning that when you turn power off to the electronic device the ROM memory retains its contents.This is typically found on computer mother boards were start-up instructions are installed. ROM is permanent memory, so it never loses what is stored in it.

20. Define Locality of Reference. (NOV/DEC 2009),(APRIL/ MAY 2008 & 14).
Many applications continually reference large amounts of data. Often, the link to this data is slow, as in the cases of primary or secondary memory references, database queries, or network requests. This leads to poor performance in the application.
To improve application efficiency by exploiting the principle of Locality of Reference also Known as Caching.

21. What is Translation Look aside Buffer? Or What is TLB? (MAY/JUNE 2006),(NOV/DEC 2011)
A translation look aside buffer (TLB) is a cache that memory management hardware uses to improve virtual address translation speed. Server processors use a TLB to map virtual and physical address spaces, and it is nearly always present in any hardware which utilizes virtual memory.

22. Define memory access time?
The time required to access one word is called the memory access time. Or It is the time that elapses between the initiation of an operation and the completion of that operation.

23. Define memory cycle time?
It is the minimum time delay required between the initiations of two successive memory operations.
Eg. The time between two successive read operations.

24. What is MMU?

MMU is the Memory Management Unit. It is a special memory control circuit used for implementing the mapping of the virtual address space onto the physical memory.

Programs can be larger than physical memory

Entire program need not be in memory

25. What are the Characteristics of semiconductor RAM memories?

- They are available in a wide range of speeds.
- Their cycle time range from 100ns to less than 10ns.
- They replaced the expensive magnetic core memories.
- They are used for implementing memories.

26. What are the Characteristics of SRAMs?

- SRAMs are fast.
- They are volatile.
- They are of high cost.
- Less density.

27. What are the Characteristics of DRAMs?

- Low cost.
- High density.
- Refresh circuitry is needed.

28. Define Memory Latency?

It is used to refer to the amount of time it takes to transfer a word of data to or from the memory.

29. What are asynchronous DRAMs?

In asynchronous DRAMs, the timing of the memory device is controlled asynchronously. A specialized memory controller circuit provides the necessary control signals RAS and CAS that govern the timing. The processor must take into account the delay in the response of the memory. Such memories are asynchronous DRAMs.

30. What are synchronous DRAMs?

Synchronous DRAMs are those whose operation is directly synchronized with a clock signal.

31. Define Bandwidth?

When transferring blocks of data, it is of interest to know how much time is needed to transfer an entire block. Since blocks can be variable in size it is useful to define a performance measure in terms of number of bits or bytes that can be transferred in one second. This measure is often referred to as the memory bandwidth.

32. What is double data rate SDRAMs? Or What is DDR SDRAM? (NOV/DEC 2011)

Double data rates SDRAMs are those which can transfer data on both edges of the clock and their bandwidth is essentially doubled for long burst transfers.

33. What is RamBus technology?

- The key feature of Ram bus technology is a fast signaling method used to transfer information between chips.
- Instead of using signals that have voltage levels of either 0 or V supply to represent the logic values, the signals consist of much smaller voltage swings around a reference voltage , vref. Small voltage swings make it possible to have short transition times, which allows for a high speed of transmission.

34. What are the special features of Direct RDRAMs?

It is a two channel Rambus.
It has 18 data lines intended to transfer two bytes of data at a time.
There are no separate address lines.

35. What are the disadvantages of EPROM?

The chip must be physically removed from the circuit for reprogramming and its entire contents are erased by the ultraviolet light.

36. Differentiate flash devices and EEPROM devices.

Flash devices

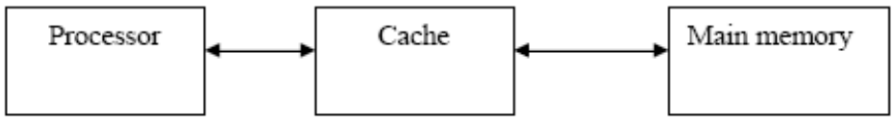
It is possible to read the contents of a single cell, but it is only possible to write an entire block of cells.
Greater density which leads to higher capacity.
Lower cost per bit.
Consumes less power in their operation and makes it more attractive for use in portable equipments that is battery driven.

EEPROM devices

- It is possible to read and write the contents of a single cell.
- Relatively more cost
- Consumes more power.

37. What is cache memory?(NOV/DEC-2013)

It is a small, fast memory that is inserted between the larger, slower main memory and the processor. It reduces the memory access time.



38. What are the two aspects of locality of reference?. Define them.

Two aspects of locality of reference are temporal aspect and spatial aspect. Temporal aspect is that a recently executed instruction is likely to be executed again very soon.
The spatial aspect is that instructions in close proximity to a recently executed instruction are also to be executed soon.

39. What are the two ways in which the system using cache can proceed for a write operation?

- Write through protocol technique.
- Write-back or copy back protocol technique.

40. What is write-through protocol?

For a write operation using write through protocol during write hit: the cache location and the main memory location are updated simultaneously.For a write miss, the information is written directly to the main memory.

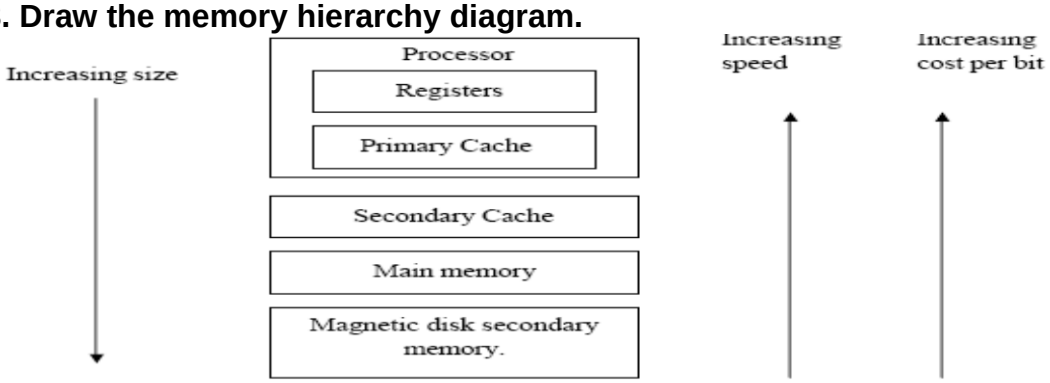
41. What is write-back or copy back protocol?

For a write operation using this protocol during **write hit:** the technique is to update only the cache location and to mark it as updated with an associated flag bit, often called the dirty or modified bit. The main memory location of the word is updated later, when the block containing this marked word is to be removed from the cache to make room for a new block.

42. What are the mapping technique? Discuss the different mapping techniques used in cache memory. (NOV/DEC-2006)

- Direct mapping
- Associative mapping
- Set Associative mapping

43. Draw the memory hierarchy diagram.



44. What is the formula for calculating the average access time experienced by the processor?

t_{ave}=hc +(1-h)M

Where,

h =Hit rate

M=miss penalty

C=Time to access information in the cache.

45. What is the formula for calculating the average access time experienced by the processor in a system with two levels of caches?

tave =h1c1(1-h1)h2c2+(1-h1)(1-h2)M

where,

h1=hit rate in L1 cache

h2=hit rate in L2 cache

C1=Time to access information in the L1 cache.

C2=Time to access information in the L2 cache.

46. What are pages?

All programs and data are composed of fixed length units called pages. each consists of blocks of words that occupies contiguous locations in main memory.

47. How many memory chips are needed to construct 2M*16 memory system using 512K * 8 static memory chips? (APRIL/MAY 2010)

4096/512 =16 chips 16 memory chips are needed to construct 2M*16 memory system using 512K * 8 static memory chips.

48. How is disk access time calculated? (NOV/DEC 2010)

Disk access time can be calculated by adding up the 'seek time' and the average 'latency time'**Seek time**: the time needed for the read/write arm to look for the desired track.**Latency time** (also called rotational delay): the time it takes for the desired sector on the track to spin around to the read/write arm (since the piece of data required might be on the other side of the disk relative to the read/write at the moment).**Transfer time**: the time a hard disk drive needs to read and transmit one block of data Disk access time in hard disk drives is measured in millisecondsEven though this might seem fast, CPUs are still able to calculate much faster than this, causing hard disk drives to be slow in comparison

Disk access time = seek time + latency time + transfer time

49. What is the use of EEPROM?(APRIL/MAY 2011)

EEPROM stands for Electrically Erasable Programmable Read-Only Memory.

It is a type of non-volatile memory used in computers and other electronic devices to store small amounts of data that must be saved when power is removed.

e.g., calibration tables or device configuration.

50. What is DDR SDRAM?(NOV/DEC 2011)

- Double data rate synchronous dynamic random-access memory (DDR SDRAM) is a class of memory integrated circuits used in computers.
- The SDRAM transfer data on the both edges of the clock, their bandwidth is essentially doubled for long burst transfers. Such devices are known as double-data-rate SDRAMs.

51. An address space is specified by 24 bits and the corresponding memory space by 16 bits: How many words are there in the virtual memory and in the main memory?(OR)An address space is specified by 24 bits & the corresponding memory space is 16 bits. (MAY/JUNE 201,12 &13)

a) How many words are there in address space?

b) How many words are there in memory space?

c) If a page has 2k words, how many pages & blocks are in the system?

Solution:-

a) Address space = 24 bits

224 = 24.220 = 16M words

b) Memory space: 16 bits

216 = 64k words

c) page consists of 2k words

Number of pages in add space = 16M/2K = 8000

Number of blocks = 64k/2k = 32 blocks

52. Define hit ratio(NOV/DEC -2006)
Hit rate=No of hits/no of bus cycles *100%

53. What is memory mapped I/O? (APRIL/MAY-2014)
When the I/O devices and the memory share the same address space, the arrangement is called memory mapped I/O.

54. What is program controlled I/O? (NOV/DEC-11)
In program controlled I/O , the processor repeatedly checks a status flag to achieve the required synchronization between the processor and an input and output device

55. What are the various mechanisms for implementing I/O operations?
- Program controlled I/O
- Interrupts
- DMA

56. What are vectored interrupts?
To reduce the time involved in the polling process, a device requesting an interrupt may identify itself directly to the processor. Then the processor can immediately start executing the corresponding ISR. The scheme is based on this approach and is called vectored interrupts.

57. What are the 2 independent mechanisms for controlling interrupt request?
At the device end, an interrupt enable bit in a control register determines whether the device is allowed to generate an interrupt request.
At the processor end, either an interrupt enable bit in the PS or a priority structure determines whether a given interrupt request will be accepted.

58.What is DMA? (NOV/DEC-2014)
Transfer of a block of data directly between an external device and main memory, with out continuous intervention by the processor is called DMA.

59.What is DMA controller?
DMA transfers are performed by a control circuit that is part of the I/O device interface. This circuit is known as DMA controller.

60. What is bus arbitration?
It is the process by which the next device becomes the bus master is selected and bus master ship is transferred to it.

61. What are the possible data transfer modes available with peripherals.(NOV/DEC-2010)
Data transfer to and from peripherals
Programmed I/O
Interrupt-initiated I/O
Direct Memory Access (DMA)

I/O Processor (IOP)

62. Define an interrupt.(NOV/DEC-2010)

An **interrupt** is a signal to the processor or an instruction in software usually indicating an event that needs immediate attention.The processor responds by suspending its current activities, saving its state, and executing a small program called an *interrupt handler* (interrupt service routine, ISR) to deal with the event.

63.Comparison between programmed i/o and interrupt driven i/o. (NOV/DEC-2014)

S.No	Programmed I/o	Interrupt driven I/O
1	It is implemented without interrupt hardware support.	It is implemented using interrupt hardware support.
2	It does not depend on interrupts status.	Interrupt must be enabled to process interrupt driven I/O.
3	It does not need initialization of stack.	It needs initialization of stack
4	System throughput decreases as number of I/O devices connected in the system increases.	System throughput does not depend on number of I/O devices connected in the system.

64. What is the advantage of DMA?

It is simple and cheap.

It requires the least number of lines and this number is independent of the number of master in the system.

65. **What is the purpose of Dirty/Modified bit in cache memory?(NOV/DEC-14)**

- A **dirty bit** or **modified bit** is a bit that is associated with a block of computer memory and indicates whether or not the corresponding block of memory has been modified. The dirty bit is set when the processor writes to (modifies) this memory.

- The bit indicates that its associated block of memory has been modified and has not yet been saved to storage. When a block of memory is to b replaced, its corresponding dirty bit is checked to see if the block needs to be written back to secondary memory before being replaced or if it can simply be removed. Dirty bits are used by the CPU cache and in the page replacement algorithms of an operating system.

66. What is the need to implement memory as a hierarchy? (APRIL/MAY-15)

The hierarchical arrangement of storage in current computer architectures is called the memory hierarchy. It is designed to take advantage of memory locality in computer programs. Each level of the hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels.

67. Point out how DMA can improve I/O speed. (APRIL/MAY-15)

The DMA can access memory directly and also an I/O request can be formulated as multi character or multi block transfer, which will prevent interrupts after each character or block.

A single independent DMA controller can handle several devices connected through an internal I/O bus.

PART-B

1.Analyze the memory hierarchy in terms of speed, size and cost.(NOV/DEC 2010)

Memory

A memory unit is a collection of storage cell together with associated circuits needed to transfer information in and out of storage.The memory stores binary information in groups of bits called words.A word in memory is an entity of bits that more in and out of storage as a unit.A memory word is a group of 1's and 0's and may represent a number an instruction code, one or more alphanumeric characters or any other binary coded information.A group of eight bit is called a byte.Most computer memories use words whose number of bits is a multiple of 8The capacity of memories in commercial computers is usually stated as the total number of bytes that can be stored.

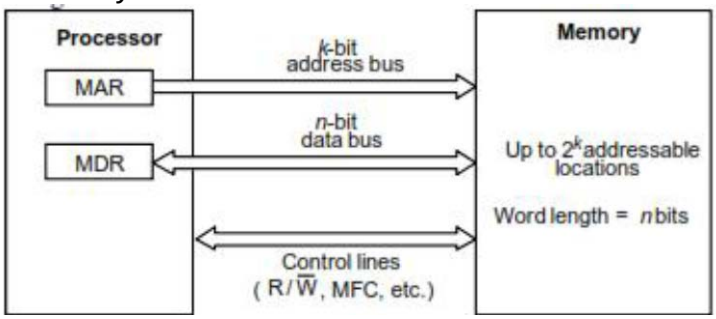


Figure 5.1. Connection of the memory to the processor

- Each word in memory is assigned an identification number called an address, starting from 0……to 2(K-1)

  Where K is the number of address lines

Example:

A 16 bit computer that generates 16 bit address is capable of addressing unto 2(16) = 64 K memory location.**Memory read operation**Place the memory in MAR(Memory Address Register)Set R/W signal to 1Memory responds by placing data on data lines and asserts the MFC signal.The processor loads the data on the data lines into the MDR (Memory Data Register).The time between the read and MFC signal is the memory access time and theminimum time delay between the initiation of two successive memory operationis called memory cycle timeThe processor speed is generally faster than memory speed and hence the memory cycle time is the bottleneck is the system.To reduce the memory access time, cache memory is used.Cache memory is small, faster memory placed between main memory and the processor and holds a active segments of a program and the data.Another concept of memory is the virtual memory which is used to increase the apparent size of the physical memory (main memory).In virtual memory concept, the addresses by the processor are referred to as virtual address or logical address **Memory Hierarchy (SPEED, SIZE AND COST)**An ideal memory would be fast, large and inexpensive. It is that a very fast memory can be implemented if SRAM chips are used. But these chips are expensive because their basic cells have six transistors and large number of basic cells is required to built a single chip. So it is impractical to build a large memory using SRAM chips. The alternatives to use DRAM chips, which have simpler basic cells and thus, they are less expensive. But such memories are significantly slower.Although dynamic memory units can be implemented at a reasonable cost with the capacity of hundreds of

megabytes, still the affordable size is small compared to the large voluminous data. This leads to the solution of secondary storage, mainly magnetic disks, to implement large memory spaces. Very large disks are available at a reasonable price, and they are used extensively in computer systems, but they are much slower than the semiconductor memory units.**So we conclude the following:** .A large amount of cost-effective storage can be provided by magnetic disks. .A large, yet affordable, main memory can be built with dynamic RAM technology.

A smaller unit is the SRAM, where speed is more such as in cache memories.All of these different types of memory units are employed effectively in a computer. The entire computer memory can be viewed as a hierarchy. The access is faster in processor registers and they are at the top in memory hierarchy.The next level of the hierarchy is a relatively small amount of memory that can be implemented directly on the processor chip. This memory is called processor cache, holds copies of instructions and data from the larger memory Cache is classified into **two levels**.A **primary cache** is located on the processor chip and it is small because it completes for space, which must implement many other functions. The primary cache is referred to as level1(L1) cache. A larger, **secondary cache** is placed between the primary cache and the rest of the memory. It is referred to as level2(L2) cache. It is usually implemented using SRAM chips.The most common way of designing computers is to include a primary cache on the processor chip and a larger secondary cache.The next level in the hierarchy is called the main memory. This rather large memory in implemented using dynamic memory components, typically in the form of SIMMs, DIMMs or RIMMs. The main memory is much larger but significantly slower than the cache memory.Disk devices provide a huge amount of inexpensive storage. they are slow compared to the semiconductor devices used to implement the main memory

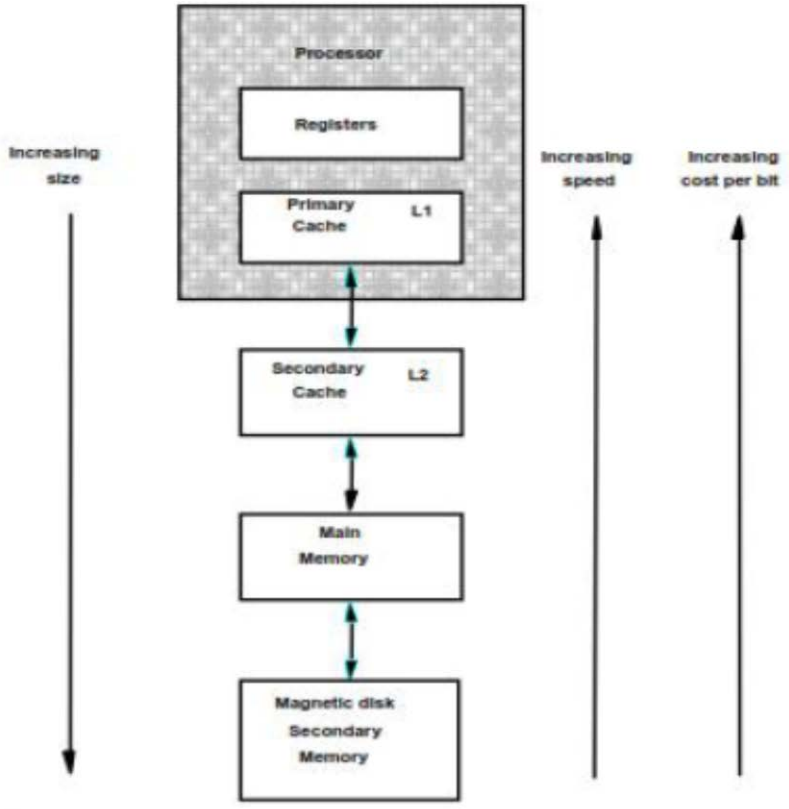


Figure 5.2: Memory Hierarchy

.

- A hard disk drive (**HDD; also hard drive, hard disk, magnetic disk or disk drive**) is a device for storing and retrieving digital information, primarily computer dataIt consists of one or more rigid (hence "**hard") r**apidly rotating discs (often referred to as platters), coated with magnetic material and with magnetic heads arranged to write data to the surfaces and read it from them.During program execution, the speed of memory access is of utmost importance. The key to managing the operation of the hierarchical memory system in Figure 4.13 is to bring the instructions and data that will be used in the near future as close to the processor as possible. This can be done by using the hardware mechanisms.

Q2. Elaborate on the various Memory technologies and its relevance.(April/May-15)

5.2. Memory technology

Ever computer contain several types of devices to store he instructions and data required for its operation. Theses storage devices plus the algorithms – implemented by hardware and/or software – needed to manage the stored information from the **memory system** of the computer.Memory Device Characteristics Random - Access Memories Serial – Access Memories **Memory Device Characteristics**A CPU should have rapid, uninterrupted access to the external memories where itsPrograms and the data they process are stored so that the CPU can operate at or near its maximum speed.**Memory types**The information – storage components of a computer can be placed in **four groups ,**as illustrated in figure CPU registersMain (primary) memorySecondary memoryCache**CPU registers**Theses high - speed registers in the CPU serve as the working memory for temporary storage of **instructions and data** .They usually form a general – purpose registers file for storing data as it is processed .A capacity of 32 data words is typical of a **register file**, and each register can be accessed, that is ,read from or written in to, with **single clock cycle** (a few nanoseconds) .**Main (primary) memory**This large, fairly fast external memory stores programs and data that are in active use. Storage locations in main memory are addressed directly by the CPU's l**oad and store** instructions.While an IC technology similar to that of a CPU register file is used , access is slower because of main memory's large capacity and the fact that it is physically separated from the CPU.Main memory capacity is typically between **1 and 2^{10} megabytes** , where megabytes ,also denoted **1 MB, is 2^{20} bytes , and 2^{10} megabytes = 2^{30} bytes is referred to as a gigabytes(1 GB)**. Access times of **five or more clock cycles** are usual. **Secondary memory (or auxiliary memory)**

Large memory capacity & much slower than main memoryStore system programs, large data files, and the like which are not continually required by the CPUInformation in secondary storage is accessed indirectly via input-output programsRepresentative technologies used for secondary memory are magnetic and optic disks**Cache:**

A cache 'storage capacity is less than that of main memory ,but with an access time of one to three cycle, the same much faster than main memory because some or all of it can reside on the same Ic as the CPU.

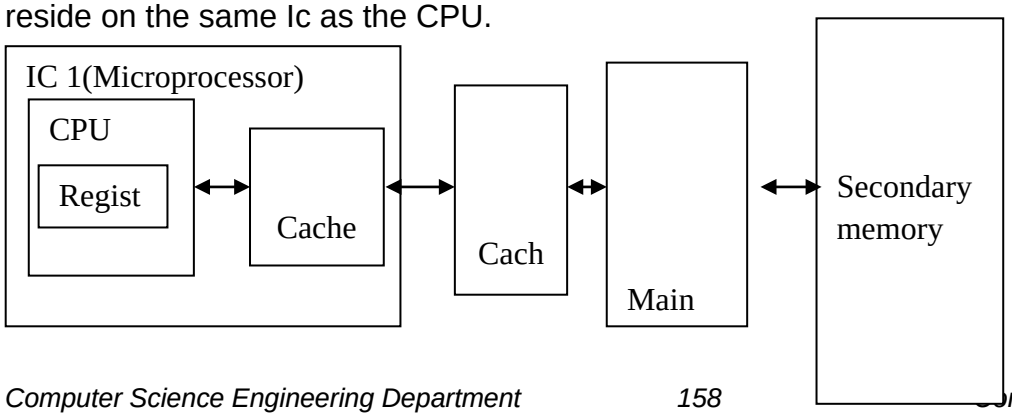


Figure 5.3.conceptual organization of a multilevel memory system in a computer

Storage capacity (S). Expressed in multiple of bits, or bytes.

– 210 bits = 1024 bits = 1 Kb; 1024 Kb = 1 Mb 8 bits = 1 Byte = 1 B

– 1024 Mb = 1 Gb; 1024 Gb = 1 Tb

Cost.

The price include not only the cost of the information storage cells themselves but also the cost of the peripheral equipment or access circuitry essential to the operation of the memory.Let C be the price in dollars of a complete memory system with S bits of storage capacity, and specific cost c of the memory as follow

Cost, c = C/S dollars/bit

3. Access time:

The average time required to read a fixed amount of information, e.g., one word, from the memory - read access time or, more commonly, the access time of the memory (tA)The write access time is defined similarly - it is typically, but not always, equal to the read access timeAccess time depends on the physical characteristics of the storage medium, and also on the type of access mechanism usedtA is usually calculated from the time a read request is received by the memory unit to the time at which all the requested information has been made available at the memory output terminalsThe access rate bA of the memory defined as 1/tA and measured in words per second.**4. Access modes:** The order of sequence in which information can be accessed **Random-access memory** (RAM) - if locations may be is independent of the location being accessedSemiconductor memoriesEach storage location has a separate access (addressing) Mechanism**Permanence of storage :-** The stored information may be lost over a period of time unless appropriate action is taken**Destructive readout** (reading the memory destroys the stored information)Each read operation must be followed by a write operation that restores the original state of the memory.The restoration is usually carried out automatically, by **write back** into the location originally addressed from a buffer register.**Dynamic storage:**Over a period of time, a stored charge tends to leak away, causing a loss of information unless the charge is restoredThe process of restoring is called **refreshing** (dynamic memories)**Volatility**A memory is said to be volatile if the stored information can be destroyed by a power failure.

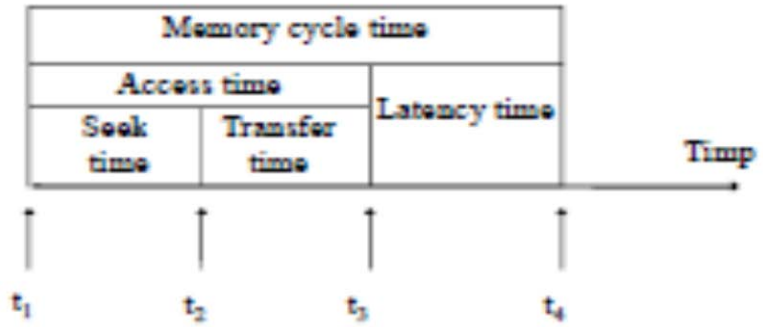
Cycle time and data transfer rate:

- Cycle time (tM) is the time needed to complete any read or write operation in the memory.
- This means that the minimum time that must elapse between the initiation of two different accesses by the memory can be greater than tM. Data transfer rate or bandwidth (bM) - the maximum amount of information that can be transferred to or from the memory every second (= 1/tM)

$$b_M = \frac{1}{t_M} \ [words/sec] \qquad b_M = \frac{w}{t_M} \ [bits/sec]$$

w = memory bus width

Cycle time vs access time



- In cases where $t_A \neq t_M$ both are used to measure memory speed
- **8. Power**: specific power

$$p = \frac{P_{dot}}{S} \ [W/bit]$$

- **9. Geometry**: only for semiconductor memories
  - N lines × M columns

3.Write a note on random access memories (or) Draw and discuss the structure of the internal memory organization? (or) Explain about Static & Dynamic memory systems. (NOV/DEC 2007)Draw and explain basic cell of RAM. Also Explain different type of RAM(MAY/JUN-2010) (or) Draw different memory address layouts and brief about the techniques used to increase the average rate of fetching words from the main memory. (NOV/DEC-14)

Random - Access Memories

RAM memories

Semiconductor memories in which the basic storage cell are transistor circuit have been used high speed CPU register since the 1950, later in 1970 it become economical to produce large RAM chips suitable for main memory application.

Semiconductor memories are available in a wide range of speeds. Their cycle time ranges from 100ns to less than 10ns.Introduction they were very expansive, because of rapid advances in VLSI (Very Large Scale Integration) technology, the cost of semiconductor memories has dropped dramatically.The semiconductor memories are available in a wide range of speedsThe cycle times ranges from 100ns to 10ns.

Semiconductor memories fall under two categories.

- SRAM (Static RAM).
- DRAM (Dynamic RAM).

Internal organization of memory chips

Memory cells are usually organized in the form of an array, in which each cell in capable of storing one bit of informationEach row of cells constitute a memory wordAll cells of a row are connected to a common line known as the word line, which is driven by address decoder of the chipEvery column of cells are connected to a **Sense/Write** circuit by 2 bit linesThe Sense/Write circuits are connected to the data input/output lines of the chip

During a **Read** operation, the information stored in the cells selected by the word line and transmit this information to the output data linesDuring a **Write** operation, the Sense/Write circuits receive input information and store it in the cells of the selected word lineThe fig shows about small memory chip consisting of 16 words of 8 bit each which is 16x8 organizations.The data input and the data output of each sense/write circuit are connected to the data bus of a computer.

Two control lines,

-R/W

- -CS

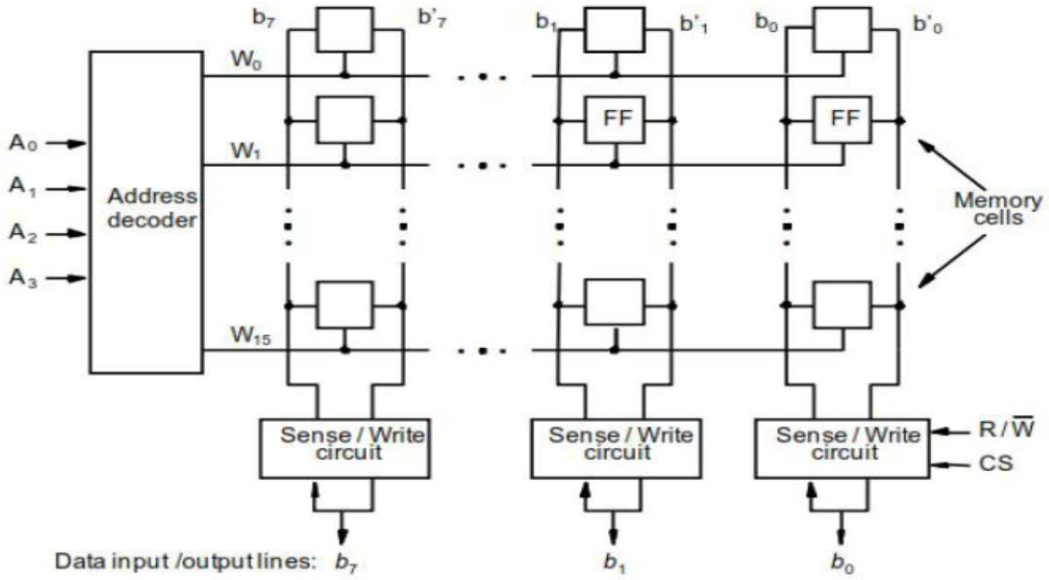


Figure 5.4 Organization of bit cell in a memory chip

- Are provided in addition to the address and data lines.This memory stores 128 bits and requires 14 external connections for address, data and control line and two lines

for power supply and ground connection.The 1k (1024) memory can be organized as 128x8 memories, requiring a total of 19 external connections as shown in fig. The row address in fig above, select a row of 32 cells all of which are accessed in parallel. Based on the column address, only one of the 32x32 cells is connected to the external data line by the output multiplexer and input multiplexer.A 4m-bit chip may have a 512kx8 organization, in which case 19 address and 8 data input output pins are needed.

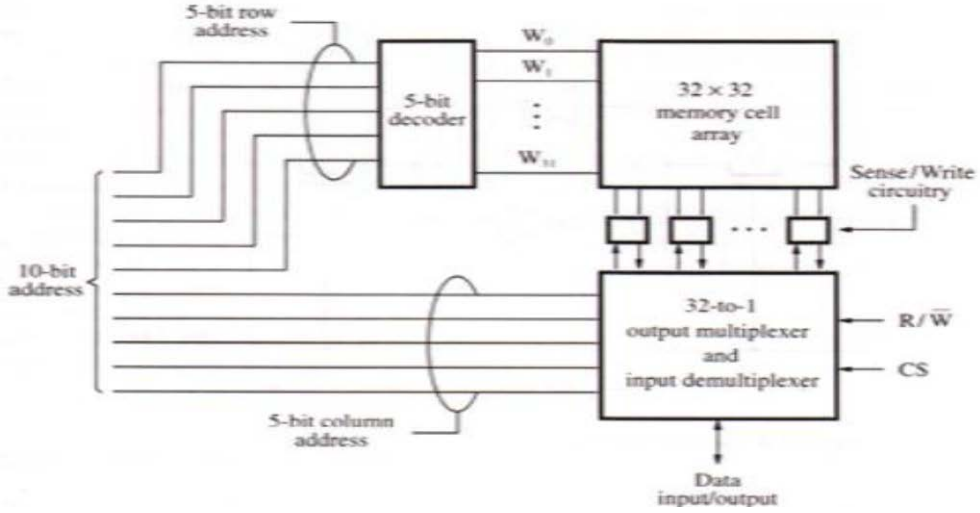


Figure 5.5. Organization of 1K x 1 memory chip

Static Memories

Memory circuits which is capable of retaining their state as long as power is applied are known as **static memories**Two inverters are cross-connected to form a latchThe latch is connected to 2 bit lines by T1 and T2These transistors act as switches that can be opened or closed under control of the word lineWhen the word line is at ground level, the transistors are turned off and the latch retains its state

Read operation

To read the state of the SRAM cell, the word lines closes the switches T1 and T2 if the cell state is 1The signal on bit line b is high end on bit line b' is lowThe case is reverse, if cell state is 0Thus b and b' are complement to each otherSense/Write circuits which is present at the end of the bit lines monitor the state of b and b' and sets the output according to that

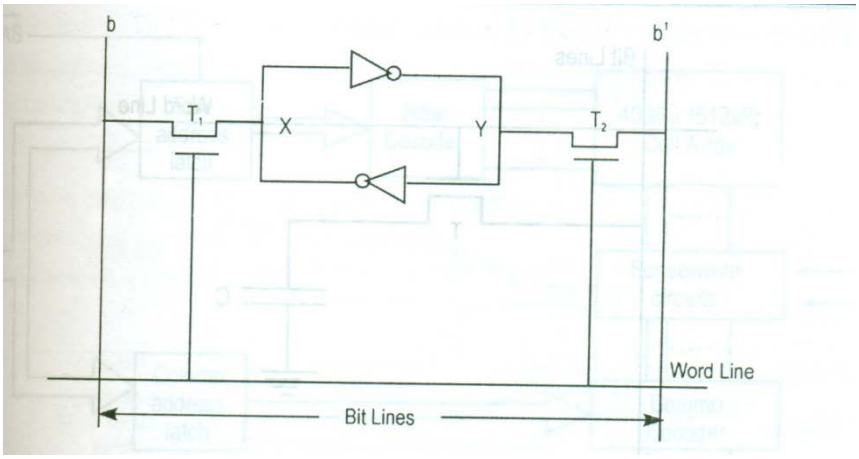


Figure 5.6. Static RAM cell

Write operation

The state of the cell is set by placing the appropriate value on bit line b and its complement on b', then activating the word lineThe required signals on the bit lines are generated by the Sense/Write circuit**CMOS Cell**Transistor pairs form the inverters in the latchThe read/write operation is same as aboveFor example, in state 1, the voltage at point X is maintained high by having transistors T3 and T6 on, while T4 and T5 are off. Thus if T1 and T2 are turned on, bit line b and b' will have high and low signals respectivelyA major advantage of CMOS SRAM's is the low power consumption because the current flows only when the cell is accessed. Otherwise T1, T2 and 1 transistor and each invertor are turned off, so that there is no active path between V supply and groundSRAMS can be accessed very quickly. Access times are of few ns SRAMS are used in applications which requires speed

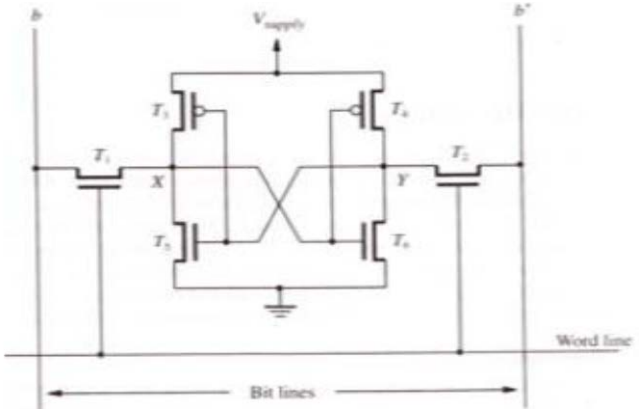


Figure 5.7. An example of a CMOS memory cell

Asynchronous DRAMS

SRAMS are fast, but it costs high. Less expensive RAM's can be implemented if simpler cells are usedHowever, such cells do not retained their state indefinitely; Hence they are called DRAMSInformation is stored in a dynamic memory cell in the form of a charge on a capacitor, and this charge can be maintained only tens of msSince the cell is required to store information for much longer time, the content much be periodically refreshed by restoring the capacitor charge to its full value

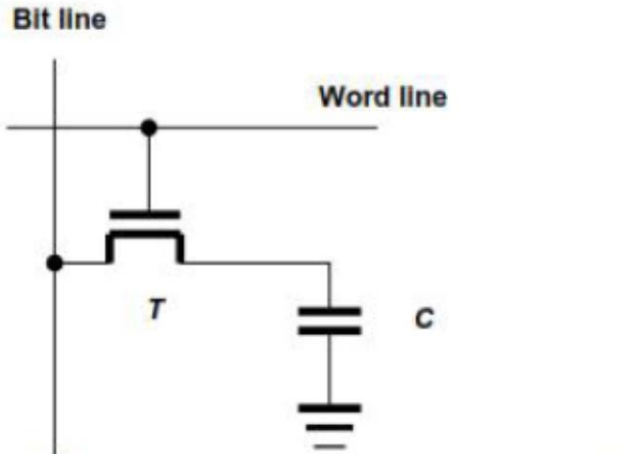


Figure 5.8.A single-transistor dynamic memory cell

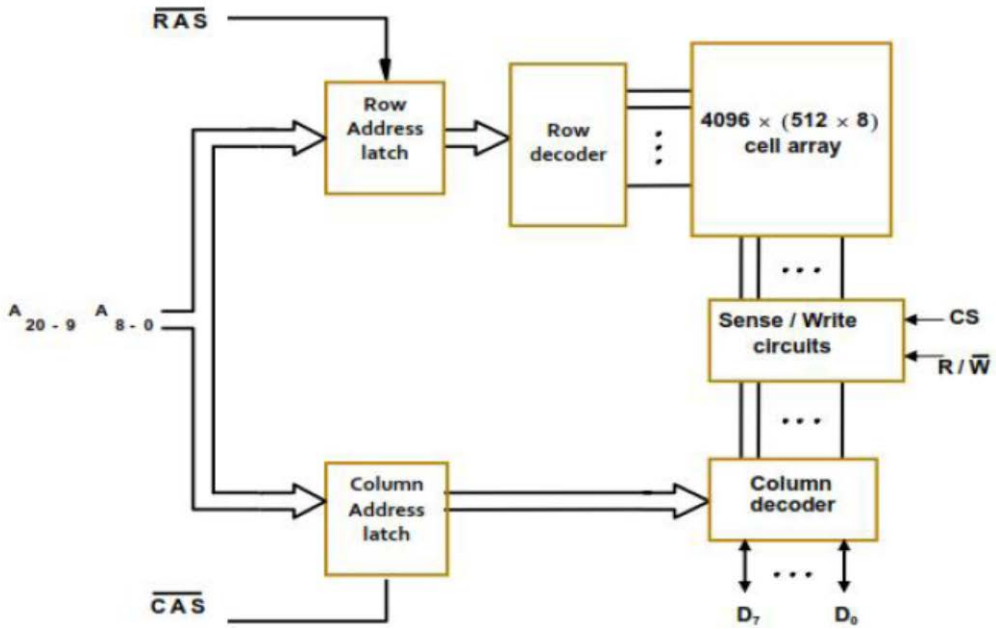


Figure 5.9. Internal Organization of a 2M * 8 Dynamic memory chip

Working Principle

In order to store information in the cell, transistor T is turned on and an appropriate voltage is applied to the bit line, This makes the capacitor to be charged.After the transistor is turned off, the capacitor begins to discharge.During that time, the transistor conducts a tiny amount of current measured in Pico Amperes.Hence, the information stored in the cell, can be retrieved correctly only if it is read before the change on the capacitor drops below some threshold value.During the read operation, the transistor in the selected cell is turned onA sense amplifier detects the charge is above the threshold value. If so, it derives the bit line to a full voltage that represents logic value 1If the sense amplifier detects the charge is below the threshold value, it pulls the bit line to the ground level, which ensures that the capacitor has no charge, represents logic value 0All cells in the selected row are read at the same time, which refreses the content of the row

Synchronous DRAMs More recent development has resulted in DRAMS whose operations are directly synchronized with a clock signal. Such memories are known as Synchronous DRAMS.

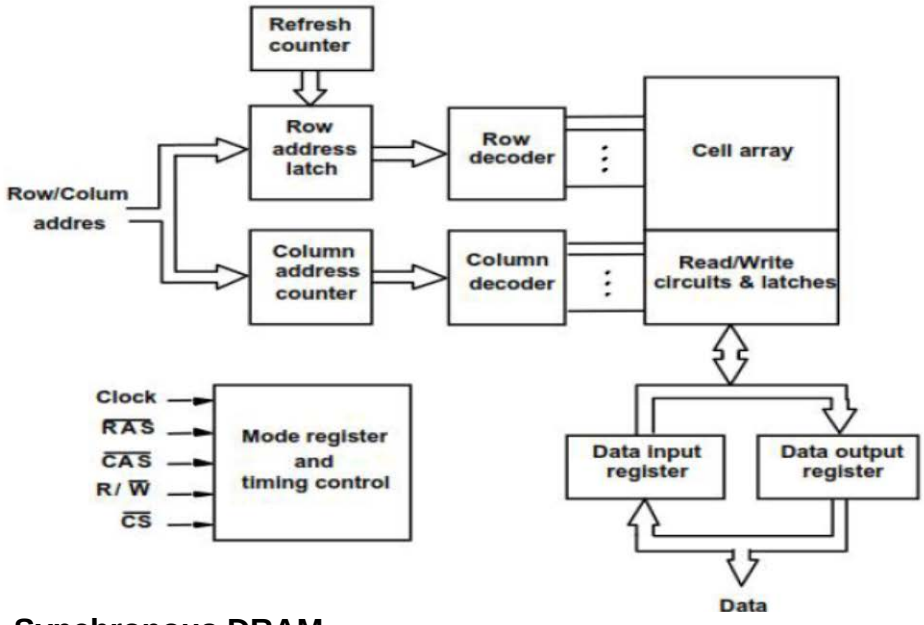


Figure 5.10. Synchronous DRAM

The cell array is the same as in asynchronous DRAMSThe address and data connections are buffered by means of registersThe output of each sense amplifier is connected to a latchA read operation causes the content of all cells in the selected row to be loaded into these latches. But, If only refreshing has to be done, the content of the latches are not changedData held in the latches that correspond to the selected columns are transferred into the data output register.SDRAMS have several different modes of operation, which can be selected by writing control information into a mode register.**Latency and bandwidth**

Transfer b/w the memory and the processor involve single words of data or small blocks of the words. Large blocks, constituting a page of data, are transferred b/w the memory and the disk. The speed and efficiency of these transfers have large a word impact on the performance of a computer system.A good performance is indicated by two parameters latency and bandwidth.The memory latency is used to refer to refer to the amount of the time it takes to transfera word of data to or from the memory. When transferring blocks of data, it is of interest to know how much time is needed to transfer an entire block.

- Since blocks can be variable in size, it is useful to define performance in bits or bytes that can be transferred in one sec. this measure is often referred to as the memory bandwidth.

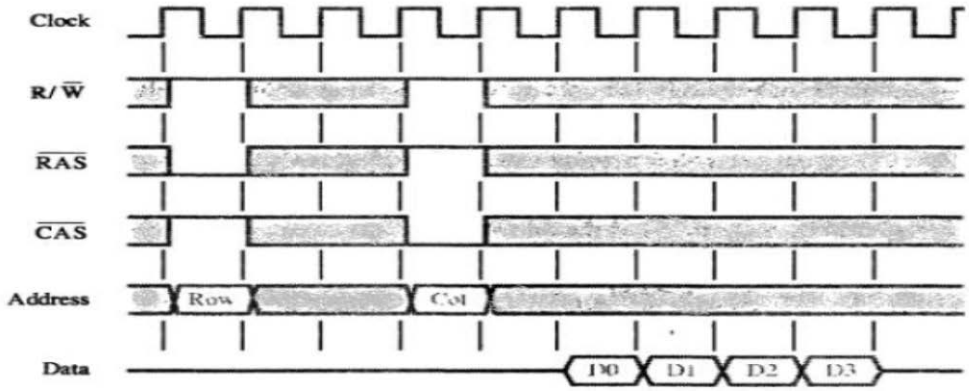


Figure 5.11. Burst read of length 4 in an SDRAM

Double data rate SDRAM Since the SDRAM transfer data on the both edges of the clock, their bandwidth is essentially doubled for long burst transfers. Such devices are known as double-data-rate SDRAMs.To make it possible to access the data at high rate, the cell array is organized in two blanks. Each blanks can be accessed separately. Consecutive words of a given block are stored in differ blanks.Such interleaving of words allows simultaneous access to two that are transferred on successive edges of the clock. DDRSDRAMs and standard SDRAMs are most efficiently used in application where block transfers are prevalent.**Structure of larger memories*Static memory system*** Consider 2Mx32 memory. Each chip has control input called chip select. When I/p is set to l, it enables the chip to accept data from or to place where data on its data line. The data o/p for each chip is of three state type.21 address bits are needed to select 32-bit word in memory. The R/W I/p of all chips are tied together to provide a common R/W control.

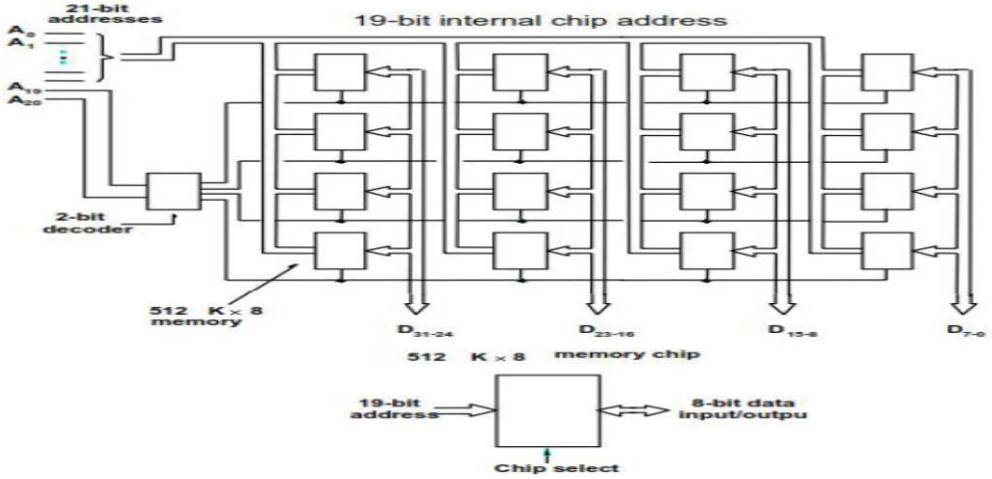


Figure 5.12.Organization of a 2M * 32 memory module using 512K * 8 static memory chips

Dynamic memory system Large memory is built by placing dynamic RAM chips directly on the main system printed circuit board that contains processor, often referred to a mother board; it will occupy an unacceptably large amount of space on mother board. These packing considerations have led to the development of larger memory units known as SIMMs & DIMMs.SIMMs-single In –line memory modules DIMMs-dual In-line memory module.**Memory system controller considerations*Memory controller***To reduce the number of pins, the dynamic memory chips use multiplexed address inputs. **The address is divided into two parts.**The high-order address bits, which select a row in the cell array, are provided first and latched into the memory chip under control of the RAS signal.Then, the low-order address bits, which select a column, are provided on the same address pins and latched using the CAS signal.A typical processor issues all bits of an address at the same time. The required multiplexing of address bits is usually performed by a memory controller circuit, which is interposed between the processor and the dynamic memory as shown in Figure 4.11.The controller accepts a complete address and the R/W signal from the processor, under control of a Request signal which indicates that a memory access operation is needed.The controller then forwards the row and column portions of the address to the memory and generates the RAS and CAS signals. Thus, the controller provides the RAS-CAS timing, in addition to its address multiplexing function. It also sends the R/W and CS signals to the memory. The CS signal is usually active low; hence it is shown as CS in Figure 4.11. Data lines are connected directly between the processor and the memory. Note that the clock signal is needed in SDRAM chips.

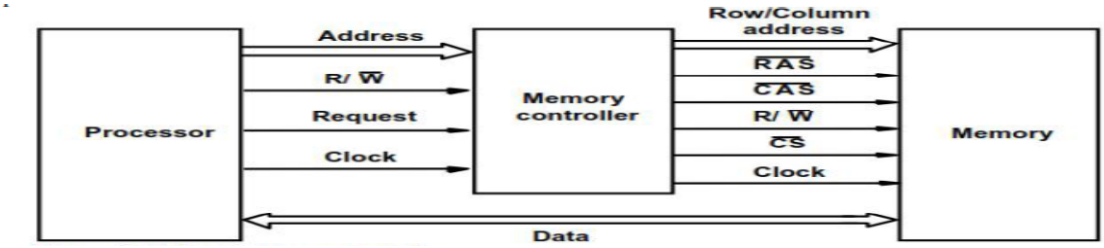


Figure 5.13. Use of memory controller

When used with DRAM chips, which do not have self-refreshing capability, the memory controller has to provide all the information needed to control the refreshing process. It contains a refresh counter that provides successive row addresses.

4. Describe in detail magnetic tape memories and Magnetic hard disk memories. (Or) Serial – Access memories: (or)What are the different secondary storage devices? Elaborate on any one of the devices?(Apr/May 2010) or Explain the organization and accessing of data on a disk.(16) (April/May 2011, May/June 2013)

Serial – Access memories:

Over view

Magnetic hard disks

Magnetic tape memories

Magnetic hard disks

The storage medium in a magnetic-disk system consists of one or more disks mounted on a common spindle. A thin magnetic film is deposited on each disk, usually on both sides. The disks are placed in a rotary drive so that the magnetized surfaces more in close proximity to read/write heads. Each head consists of a magnetic yoke and a magnetizing coil**The functionality of magnetic disks:**Digital information can be stored on the magnetic film by applying current pulses of suitable polarity.The head can be used for reading the stored information.The changes in the magnetic field caused by the film movement induce a voltage in the coil.The polarity of this voltage is monitored by the control circuitry to determine the state.A voltage is induced in the head only at 0-to-1 and 1-to-0 transitions in the bit stream.A long string of 0's or 1's causes as induced voltage only at the beginning and end of the string.To determine the number of consecutive 0s or 1s stored, a clock must provide formation for

synchronization.

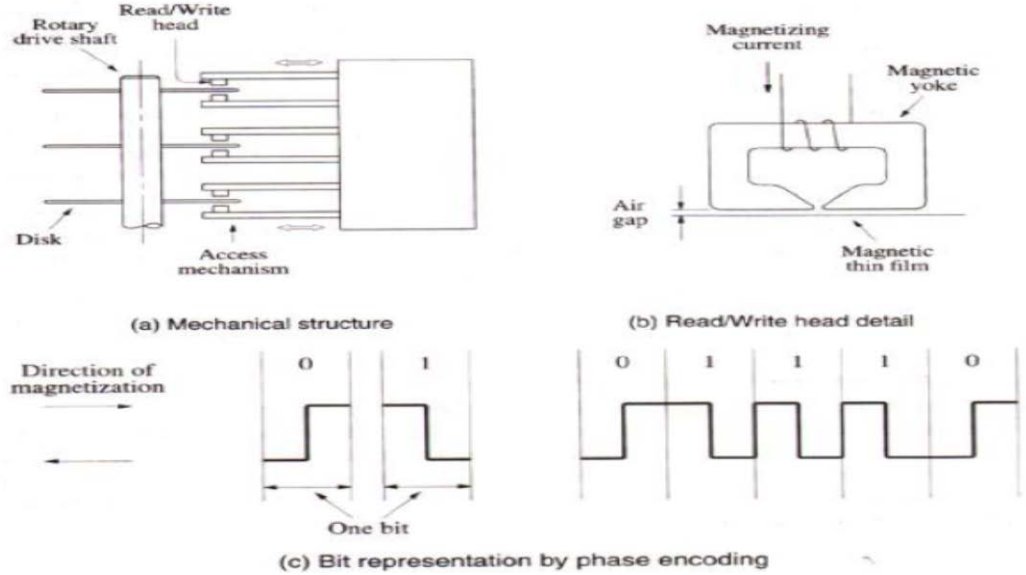


Figure.5.14. Magnetic disk principles

We will now see about the clocking information.

- The modern approach is to combine the clocking information with the data.
- One simple scheme is known as phase encoding or Manchestor encoding.
- In this scheme, changes in magnetization occur for each data bit.
- Change in magnetization is guaranteed at the mid-point of each bit period, thus provide the clocking information.
- The drawback of this coding is its poor bit-storage density because it requires more space.
- Other, more compact codes are developed. They are more efficient and provide better storage density. They also requires more complex circuitry.

Information about read/write head

Read/write heads must be maintained at a very small distance in order to achieve high bit densities and reliable read/write operations.A0ir pressure develops between the disk surface and the head and forces the head away from the surface, while the disks are moving.The force is achieved by the spring that is present for head to be passed towards the surface.The flexible spring connection permits the head to fly at the desired distance away from the surface.**Winchestor Technology**In most modern disk units, the disks and the read/write heads are placed in a scaled, air –filtered enclosure. This approach is known as winchestor technology.Read/write heads can operate closer to the surfaces because dust particles are absent here.If the head are closer to the surface, then the data can be packed more densely along the track**Advantage**Winchestor disks have a larger capacity, compared to unsealed units.Data integrity is greater in scaled units because the medium is not exposed to contaminating elements.

Disk system

The disk system consists of three parts.The assembly of disk platters, usually referred to as the disk.The electromechanical mechanism that spins the disk and moves the read/write heads, it is called the disk drive.The electronic circuitry that controls the operation of the system, which is called the disk controller.

Organization and accessing of data on a disk

The organization of data on a disk is shownEach surface is divided into concentric tracks, and each track is divided into sectors.The set of corresponding tracks on all surfaces of a stack of disks forms a logical cylinderThe data on all tracks of a cylinder can be accessed without moving the read/write heads.The data are accessed by specifying the surface number, the track number, and the sector number.The read and write operations start at sector boundaries.

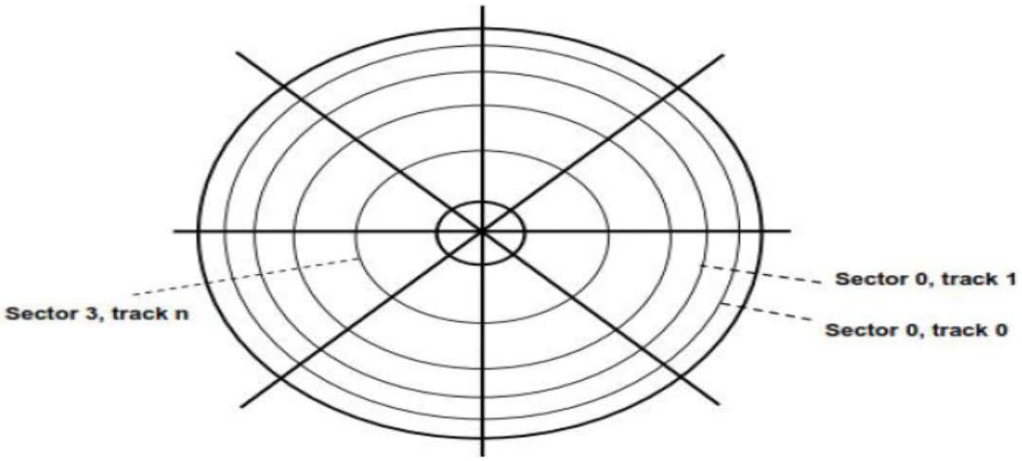


Figure 5.15.Organization of one surface of a disk

Data bits are stored serially on each track.Each sector usually contains 512 bytes of data.
The data are preceded by a sector header that contains identification information used to find the desired sector on the selected trackError-correcting code bits are used to detect and correct errors.To easily distinguish between two consecutive sectors, there is a small intersector gap.**Unformatted disk**An unformatted disk has no information on its track.
The formatting process divides the disk physically into tracks and sectors.The disk controller keeps a record of defective sectors or tracks and excludes them disk.The capacity of the formatted disk is a proper indicator of the storage capacity of the disk.The formatting information takes 15 percent of the total information.The disk comprises the sector headers, the ECC bits and intersector gaps.The figure indicates that each track has the same number of sectors. So all tracks have the same storage capacity. Thus, the stored information is packed more densely on inner tracks than an outer tracks. But, it is possible to increase the storage capacity on outer tracks with more complicated circuits.

Access time

There are two components involved in the time delay between receiving an address and the beginning of the actual data transfer. **Seek time**: Time required to move the read/write head to the proper track. Average time in 5-8ms range.**Rotation delay**: Also called latency time. The amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector passes under the read/write head.**Definition: Disk access time.**The sum of these 2 delays is called disk access time.**Typical Disks**A 3.5-inch (diameter) high capacity, high-data-rate-disk available today have the following parameters.There are 20 data-recording surfaces with 15,000 tracks per surface.There is an average of 400 sectors per track, and each sector contains 512 bytes ofdata.The total capacity of the formatted disk is 20*15000*400*512=60gigabytes.The average seek time is 6ms.The platters rotate at 10,000 revolutions per minute, so that the average latency in 3ms.

Data buffer/cache

A disk drive is connected to the rest of the computer using a standard bus, such as the SCSI bus. The SCSI bus is capable of transferring data at much higher rates than the rate at which data can be read from disk tracks. The difference in transfer rates between the disk and the SCSI bus is to include a data buffer in the disk unit. The requested data are transferred between the disk tracks and the buffer at a rate department on the rotational speed of the disk.The data buffer can also be used to provide a caching mechanism for the disk. Because of this the possible strategy is to begin transferring the contents of the track into the data buffer as soon as the read/write head is positioned over the desired track.

Disk controller

- Operation of a disk drive is controlled by a disk controller circuit.It also provides an interface between the disk drive and the bus that connects it to he rest of the computer system.It may be used to control more than one drive.A disk controller that is connected directly to the processor system bus contains a number of registers that can be read and written by the operating systems.Actually, the DMA transfers are from/to the data buffer, which is implemented as a part of the disk controller module.The OS entail loading the controller's registers with the necessary addressing and control information, typically:

Main memory address: The address of the first main memory location of the block of words involved in the transfer.**Disk controller:** The location of the sector containing the beginning of the desired block of words.**word count:** The number of words in the block to be transferred.**On** the disk drive side, the controller's major functions are:**Seek: C**auses the disk drive to move the read/write head from its current position to the desired track.

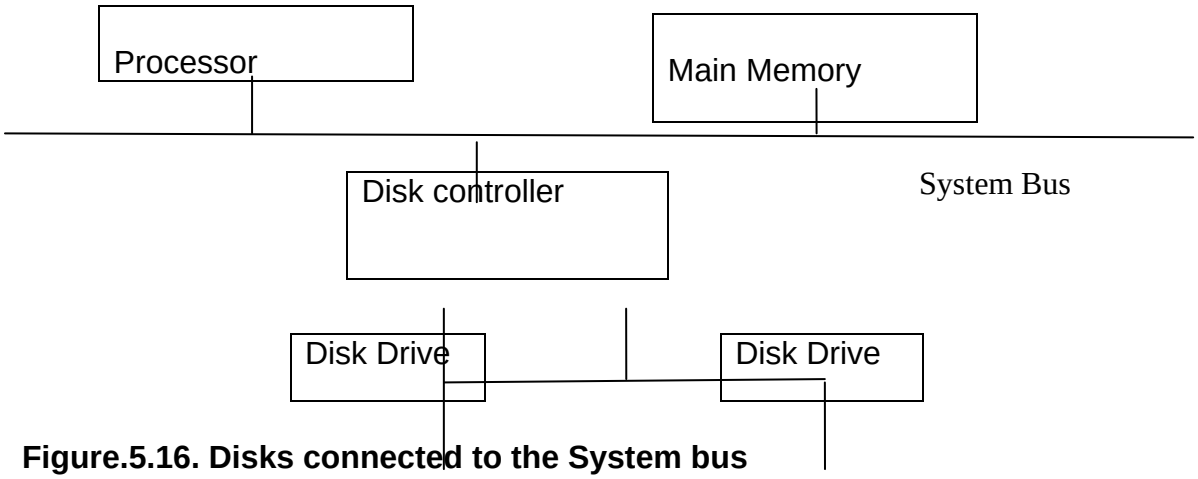


Figure.5.16. Disks connected to the System bus

Read: Initiates a read operation, starting at the address specified in the disk address register.**Write:** Transfers data to the disk, using a control method.**Error checking: C**omputes the error correcting code (ECC) value for the data read from a given sector and compares it with the corresponding ECC value read from the disk.If a disk drive is connected to a bus that uses packetized transfers, then the controller must be capable of handling such transfers.**Magnetic tape memories**The magnetic tape unit is one of the oldest and cheapest terms of main memory. Its main use is to provide hard disk backup storage for a system. Magnetic-tape recording uses the same principle as used in magnetic-disk recording. The main difference is that magnetic film (iron oxide) is deposited on a very thin 0.5 or 0.25 inch wide plastic tape

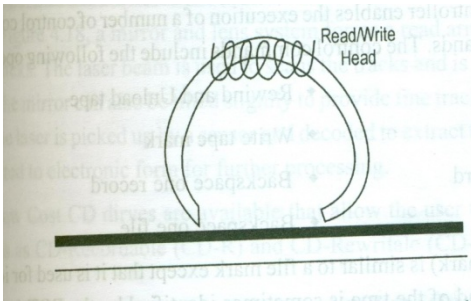


Figure.5.17. Magnetic recording with read/write Head

Date stored on a tape in parallel, longitudinal tracks. Usually 7 or 9 bits are recorded in parallel across the width of the tape perpendicular to the direction of motion. A separate read/write head is provided for each bit position on the tape, so that all bits of characters can be read or written in parallel,. One of the character bits is used as parity bits is used as parity bit.

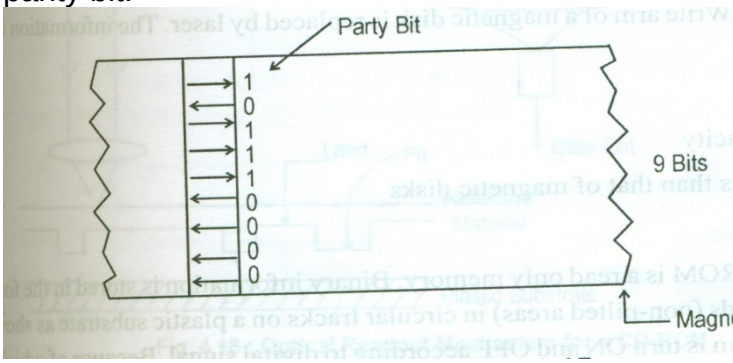


Figure.5.18. Magnetic Recorded Tape

Data on the tape are organized in the form of records separated by gaps.

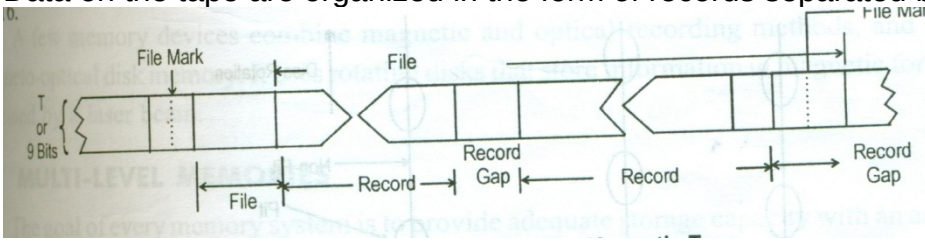


Figure.5.19. Organization of Data on Magnetic Tape

A set of records constitutes a file. A file mark is used to identify the beginning of the file. The file mark is a single or multiple character record, usually preceded by a gap longer than inter record gap.The record (First record) following the file mark can be used as a header file. This allows the user to identify the particular file form number of files recorded on the tape. The magnetic tape drives controller enables the execution of the number of control commands in addition to read or writes commands. The control commands include the **following operations;**Rewind tapeErase tapeForward space one recordForward space on fileRewind and unload tapeWrite tape markThe tape mark (write tape mark) is similar to a file mark except that it is used for identifying the beginning of the tape. The end of the tape is sometimes identified by the EOT (End of the tape) character.

5.Describe cache memory in detail. (or) What is mapping function? What are the ways the cache can be mapped? (or)Explain different types of mapping functions in cache memory. (or) Explain the various mapping techniques associated with cache memories .(NOV/DEC 2011,APRIL/MAY 2011, MAY/JUNE 2012,13,&14)(NOV/DEC-14)

basics :

OVER VIEW:

Cache memory basic concept
Mapping Functions
Replacement Algorithms

Cache memory basic concept

The speed of the main memory is very low in comparison with the speed of the modern processors.The processor cannot spend much time for waiting to access instructions and data in main memory.Therefore it is important to devise a scheme that reduces the time needed to access the necessary information.An efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster than it really is.The effectiveness of the cache mechanism is based on a property of computer programs called **locality of reference**.

- It manifests itself in 2 ways:
  - **Temporal**
  - **Spatial**

- The **Temporal** means that a recently executed instruction is likely to be executed again very soon.
- The **spatial aspect** means that instructions in close proximity to a recently executed instruction are also likely to be executed soon.

f the active segments of a program can be placed in a fast cache memory, then the total execution time can be reduced significantly.The memory control circuitry is designed to take advantage of the property of locality of reference.The term block is referred as a set of contiguous address locations of some size.

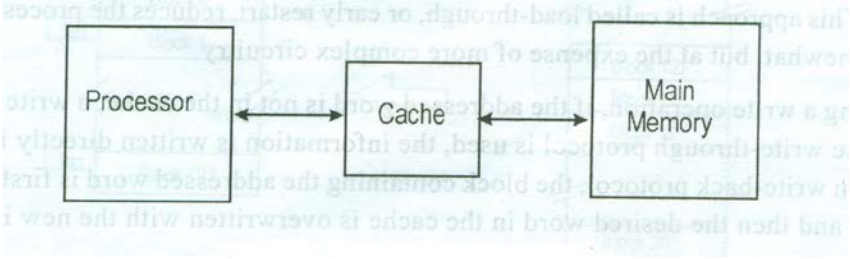


Figure.5.20. Role of Cache Memory

Use of a cache memory

When Read request is received from the processor, the contents of a block of memory words containing the location specified are transferred into the cache one word at a time.
The correspondence between the main memory blocks and the cache is specified by the mapping function.When the cache is full and the referenced memory word Is not in cache, the cache control hardware must decide which block should be removed to create space for the new block that contains the referenced word.The collection of rules for making this decision constitutes the replacement algorithm.The processor does not need to know explicitly about the existence of the cache.It simply issues Read and Write Requests using addresses that refer to locations in the memory.The cache control circuitry determines whether the requested word currently exists in the cache. If it does not, the Read and Write operation is performed on the appropriate cache location. This is known as a **Read or Write hit.**In a Read operation, the main memory is not involved. A write operation can be preceded in 2 ways.**Write-through protocol**: In this, the cache location and the main memory location are updated simultaneously**Write-back or copy back protocol**: In this, the cache location is updated and it is marked with an associated flag bit, often called the

dirty or modified bit. The main memory location is updated later, when the block containing this marked word is to be removed from the cache to make room for a new block.The write-through protocol is simpler, but it results in unnecessary write operations in the main memory when a given cache word is updated several times during its cache residency.The write back protocol may also result in unnecessary write operation because when a cache block is written back to the memory; all words of the block are written back, even if only a single word has been changed while the block was in the cache.When the addressed word in a Read Operation is not in the cache, a read miss occurs; the block of words that contains the requested word is copied from the main memory into the cache. After the entire block is loaded into the cache, the particular word requested is forwarded to the processor.Alternatively, this word may be sent to the processor as soon as it is read from the main memory. This approach is called load-through, or early restart, reduces the processor's waiting period somewhat, but at the expense of more complex circuitry.

During a write operation, if the addressed word is not in the cache, a write miss occurs. Then, if the write-through protocol is used, the information is written directly into the main memory. In write-back protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

Mapping Functions

Over View:

- Direct Mapping
- Associative Mapping
- Set Associative Mapping

Methods for specifying where memory blocks are placed in the cache.

Consider a cache memory of 128 block of 16 words cache, for a total of 2048 word and assume that the main memory is addressable by a 16-bit address. The main memory has 64 k words, which can be viewed, as 4k blocks of 16 word each.

Direct mapping

The simplest way to determine cache locations in which to store memory blocks is the direct-mapping technique.In this technique, block j of the maim memory maps onto block j modulo 128 of the cache. Thus, whenever one of the main memory blocks 0,128,256…. Is loaded in the cache, it is stored in cache block 0. Block 1, 129, 257,… are stored in cache block 1, and so on. Since more than one memory block in mapped onto a given cache block position, contention may arise for that position even when the cache is not full.Contention is resolved by allowing the new block to overwrite the currently resident block. In this case, the replacement algorithm is trivial.Placement of a block in the cache is determined from the memory address. The memory address can be divided into three fields. The low-order 4 bits select one of 16 words in a block. When a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored.The high order 5 bits of the memory address of the block are stored in 5 tag bits associated with its location in the cache. They identify which of the 32 blocks that are mapped into this cache position are currently resident in the cache.As execution proceeds, the 7-bit cache block field of each address generated by the processor points to a particular block location in the cache. The high order 5 bits of the address are compared with the tag

bits associated with that cache location.If they match, then the desired word is in that block of the cache. If there is no match, then the block containing the required word must be first read from the main memory and loaded into the cache. The direct mapping technique is easy to implement, but it is not very flexible.

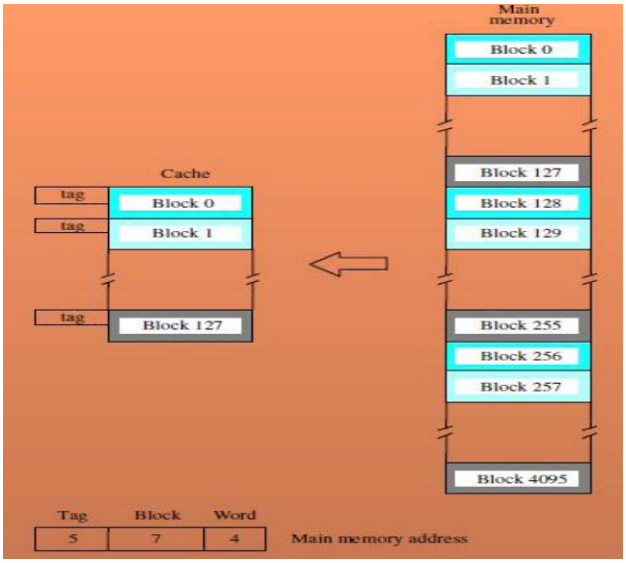


Figure.5.21. Direct mapping

Associative mapping

This mapping method is more flexible, in which a main memory block can be placed into any cache position.In this, 12 tag bits are required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is represent. This is called the associative-mapping technique.A new block that has to be brought into the cache has to replace an existing block only if the cache is full. In this case, an algorithm is needed to select the block to be replaced.

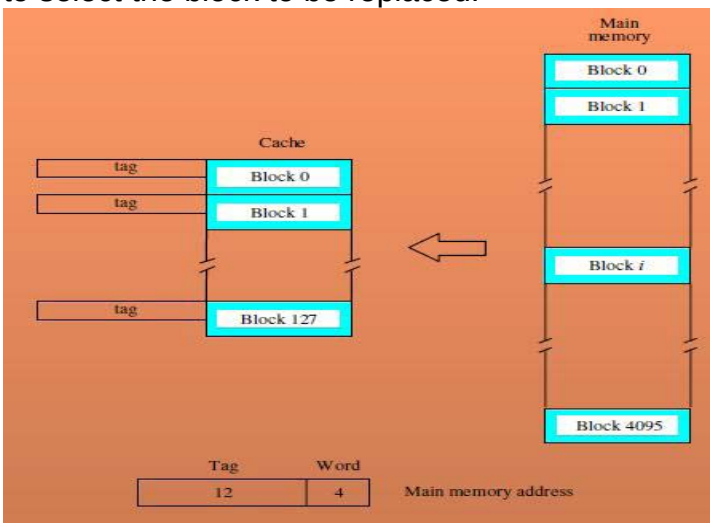
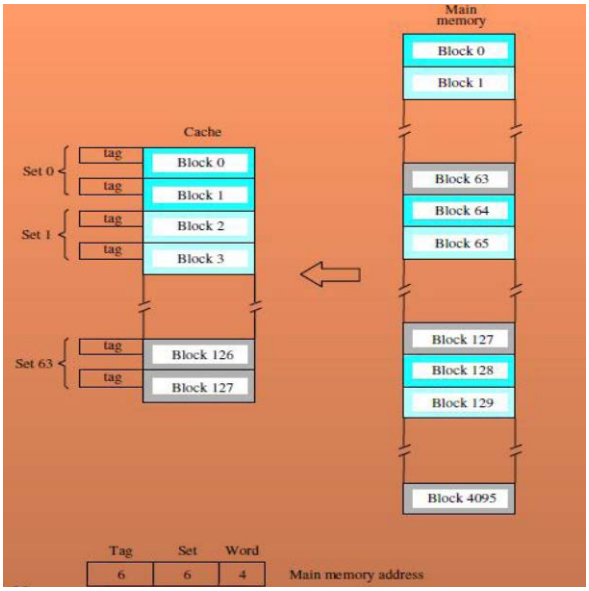


Figure.5.22. Associative mapping

Set-associative mapping

This mapping is the combination of the direct mapping and associative-mapping techniques. Block of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set.
Hence the contention problem of the direct method is eased by having a few choices for block placement.An example of this set-associative mapping technique is shown in the diagram for a cache with two blocks per set.n this case, memory blocks 0, 64, 128, …. 4032 map into cache set 0, and they can occupy either of the two block positions within this set.To have 64 sets, the 6-bit set field of the address detrmines which set of the cache might contain the desired block. The tag field of the address must then be associative compared to the lags of two blocks of the set to check if the desired block is present. This two way associative search is simple to implement.The number of blocks per set is a parameter that can be selected to suit the requirements of a particular computer. A cache that has k blocks per set is referred to as a k-way set associative cache.One more control bit, called the valid bit, must be provided for each block. This bit indicates whether the block contains valid data. The valid bits are all set to 0 when power is initially applied to the system or when the main memory is loaded with new programs and false from the disk. Transfers from the disk to the main memory are carried out by a DMA mechanism. The valid bit of a particular cache block is set to 1 the first time this block is loaded from the main memory.



Figur.5.23. Set-Associative mapping

Whenever a main memory block is updated by a source that bypasses the cache, a check is made to determine whether the block being loaded is currently in the cache. If it is its valid bit is cleared to 0. This ensures that state data will not exist in the cache. A similar difficulty arises when a DMA transfers is made from the main memory to the disk, and the cache uses the write-back protocol. In this case, the data in the memory might not reflect the changes that may have been made in the cached copy. One solution to this problem in to flush the cache by forcing the dirty data to be written back to the memory before the DMA transfer takes place.

Replacement Algorithms:

The algorithm that are used to decide which block is to be replaced with the incoming block are known as page replacement algorithmsThe main objective of cache technology is to keep blocks in the cache that are likely to be used in the near future.However, it is not easy to etermine which blocks are about to be referenced.The property of locality of reference in

programs can be used as a resolving strategy. Since programs usually stay in localized areas for reasonable periods of time, there is a high probability that the blocks that have been referenced recently will be referenced again soon.**The following are the various replacement algorithms that are widely used.**First-in First-out replacement algorithm
Random replacement algorithm
LRU replacement algorithm
Most recently used replacement algorithm
Least recently used replacement algorithm
Most frequently used replacement algorithm
First-in First-out replacement algorithm This replacement algorithm chooses the block that is staying for longest time for replacement.
Advantages:
It is simple
It is easy to implement
Disadvantages
It is not optimal
It is not efficient
LRU replacement algorithm
In a direct-mapped cache, the position of each block is predetermined, hence, no replacement strategy exists. In associative and set-associative caches, when a new block is to be brought into the cache and all the positions that it may occupy are full, the cache controller must decide which of the old blocks to be overwritten.In general, the objective is to keep blocks in the cache that are likely to be reference. However, it is not easy to determine which blocks are to be referenced. Therefore, when a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced. This block is called the least recently used block, and the technique is called the LRU replacement algorithm.Example: Array, which is too large to fit into the cache performance can be improved by introducing a small amount of randomness in deciding which block to replace.Several other replacement algorithms are also used in practice. A reasonable rule would be removed the “oldest” block from a full set when a new block must be brought in.Many replacement algorithms didn’t follow the access of cache, it is generally not as effective as the LRU algorithm. The simplest algorithm is to randomly choose the block to be overwritten . Interestingly enough, the simple algorithm has been found to be quite effective in practice.
Advantages:
The LRU algorithm has been used extensively
It takes locality of reference into consideration
Disadvantages
It becomes expensive as the number of blocks is increased
It is slower in many cases
It is harder to achieve.

6.Discuss the following Measuring and improving cache performance. (NOV/DEC-2011)
Interleaving Or Addressing multiple-module memory systems. (APRIL/MAY 2010)
Hit rate and Miss penalty Pre-fetching
Measuring and improving cache performance

Two key factors in the commercial success of a computer are performance and cost. The objective is the best possible Performance at the lowest cost.The challenge in design alternative is to improve the performance without increasing the cost. A common measure of success is the price/performance ratio.The memory hierarchy shows the best price/performance ratio. Each level of hierarchy plays an important role. The speed and efficiency of data transfer between various level of hierarchy are having great significance.Both is not possible if, both the slow and the fast units are accessed in the same manner, but can be achieved by the parallelism in the organization of the slower unit. An effective way to introduce parallelism is to use an **"interleaved organization.**

Over View:

Memory interleaving
Hit rate and Miss penalty
Caches on the processor chipOther Enhancements
Write Buffer
Perfecting
Lockup-Free cache

Memory interleaving:

- If the main memory of computer is structured as a collection of physical separate modules , each with its own address buffer register(ABR) and data buffer register(DBR), memory address operations may proceed in more than one module at the same time.How individual address are distributed over the modules is critical. Two methods of address layout are there.In the first case, the memory address generated by the processor is decoded as shown in fig. the higher-order k bits name one of n modules, and the lower-order m bits name a particular word in that module.
- When consecutive locations are accessed , when a block of data is transferred to a cache, only one module is involved. At the same time, devices with direct memory access(DMA) ability may be accessing information in other memory modules.

The second way to address the modules is shown in fig. It is called memory interleaving. The low-order k bits of the memory address select a module, and the higher-order m bits name a location within that module.

In this way , consecutive addresses are located in successive modules. This results in faster access to a block of data and higher average utilization of the memory system as a whole.

← K Bits →
-module memory sytems

Interleaving is used to within SDRAM chips to improve the speed of accessing successive word of data.

Hit rate and Miss penalty

- The effective implementation of the memory hierarchy is the success rate in accessing information at various levels of hierarchy.The successful access to data in cache is called a hit. The number of hits is stated as a fraction of all attempted accesses is call the hit rate, and the misses rate is the number of misses stated as a fraction of attempted accesses.High hit rates are essential for high-performance computers well over 0.9.Performance is affected by the miss. The extra time needed to bring the desired information into the catch is called the miss penalty. This penalty makes the processor to stall.In general, the miss penalty is the time needed to bring a block of data from a slower unit to a faster unit. The miss penalty is reduced, if the efficient

mechanisms are implemented.The performance of a computer is affected positively by increased hit rate and negatively by increased miss penalty, the block sizes that are neither very small nor large give the best results.In practice, block sizes in the range of 16 to 128 bytes have been the most popular choices.

Caches on the processor chip

The space on the processor chip is needed for many other functions, this limits the size of the cache that can be accommodated.All high-performance processor chips include some form of cache. Some manufacturers have chosen to implement two separate caches, one for instructions and another for data.**Example:** 68404, Pentium 3 and Pentium 4 processors.**Example:** ARM710T Processor.A combined cache provides better bit rate and it offers greater flexibility in mapping new information into the cache.**Disadvantages:**
Increase parallelism comes at the expense of more complex circuitry.In high-performance processors two levels oh caches are normally used. The L1 caches(s) is on the processor chip. The L2 cache, which is much larger, may be implemented externally.If both L1 and L2 caches are used, the L1 cache should be designed to do fast access. A practical way to speed up access to the cache is to access more than one word simultaneously and let the processor use them one at a time.The average access time experienced by the processor in a system with two levels of caches is $\mathbf{T_{ave}=h_1C_1+(1-h_1)h_2C_2+(1-h_1)(1-h_2)MWhere}$ h_1 is the hit rate in the L_1 cache h_2 is the hit rate in the L_2 cacheC_1 is the time to access information in the L_1 cacheC_2 is the time to access information in the L_2 cache M is the time to access information in the main memory

Write buffer

When a write-through protocol is user, each write operation results in writing new value into the memory.If the processor must wait for the memory function to be completed , the it slowed down by all write requests.It is not necessary for the processor to wait for the write request to be completed.To improve performance , a write buffer can be included for temporary storage of write requests.The processor [places each write request into this buffer and continues execution if the next instruction.The write requests stored in the write buffer are sent to the main memory when it is not having any reading operation.
The write buffer may hold a number of write requests.

5.4.4.2.prefetching

The new data are brought into the cache when they are first needed. A read miss occurs , and the desired data are loaded from the main memory. The processor has to pause until the new data arrive.A special prefetch instruction may be provided in the instruction set of the processor. Executing this instruction causes the addressed data to the loaded into the cache, in the case of read miss.Prefetch instructions can be inserted into a program either programmer or by the compiler.However the overall effect of the software prefetching on performance is positive and it is supported by machine instructions of many processors.Prefetching can also done through hardware . this involves adding circuitry thet attempts to discover a pattern in memory references and then prefetches data according to that.**Example:**Intel's Pentium 4 processor has facilitates for prefetching information into caches using both software and hardware approaches. There are special prefetch instructions that can be included in programs to bring a block of data into the desired level of cache.

Lookup-free cache

- A cache that can support multiple outstanding misses is called lookup-free. Since it can service only one miss at a time, it must include circuitry that keeps track of all outstanding misses.
- They may be done with special registers that hold the pertinent information about these misses.
- A processor that uses a pipelined organization, which overlaps execution of several instructions , a read miss caused by one instruction could stall the execution of other instructions.

7.Describe briefly about virtual memory. (or) Explain about paging? (or) Explain how the logical address translated into real address in segmentation and paging s). ystem? Explain how the virtual address is converted into real address in a paged virtual memory system.(8) Explain the Address Translation in Virtual Memory(8APRIL/MAY 2010, NOV/DEC 2010, MAY/JUNE 2013,NOV/DEC-14)

Virtual memory

Introduction: In modern computers, the physical memory is not so large to accommodate very large programsThe size of main memory is small and the program does not completely fit into the main memory. The parts of it not currently being executed are stored in secondary memoryand transferred to main memory when it is requiredVirtual memory is a technique is used to extend the apparent size of the physical memory.It uses secondary storage such as disks, to extend the size of physical memoryTechniques that automatically move program and data blocks into the physical memory when they are required for execution are called **virtual memory techniques.**

Virtual memory organization:

MMU (Memory Management Unit) controls the virtual memory system. It translates virtual addresses into physical addresses

- A simple method for translating virtual, addresses into physical addresses is to assume that all programs and data are composed of fixed length units called **pages.**

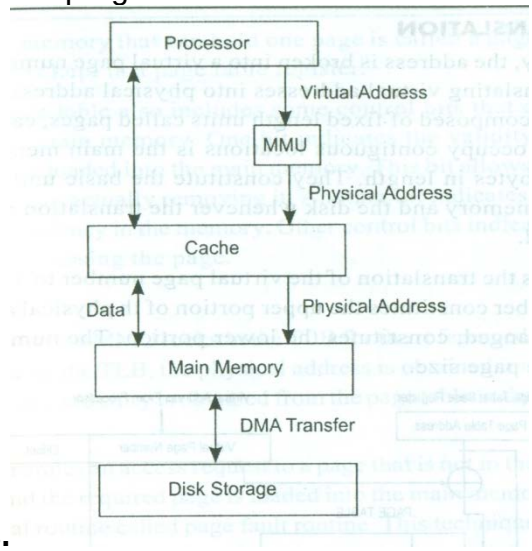
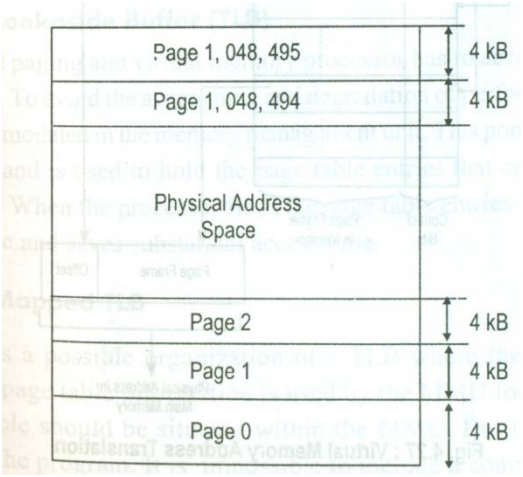


Figure 5.25. Virtual Memory Organazation



Page constitute of the basic unit of information that is moved between the main memory and the disk whenever the page translation mechanism determines the swapping is required.

Advantages of virtual memory:

Programs can be larger than physical memoryEntire program need not be in memory

Explain address translation method in virtual memory

Address Translation:

In virtual memory, the address is broken into a virtual **page number** and a **page offset**

A page consists of block of words that occupy contiguous locations in the main memory.Pages commonly ranges from 2k to 16k bytes in length.The physical page number constitutes the upper portion of the physical address, while the page offset, which not changed, constitutes the lower portionThe number of bits in the page offset field decides the page size

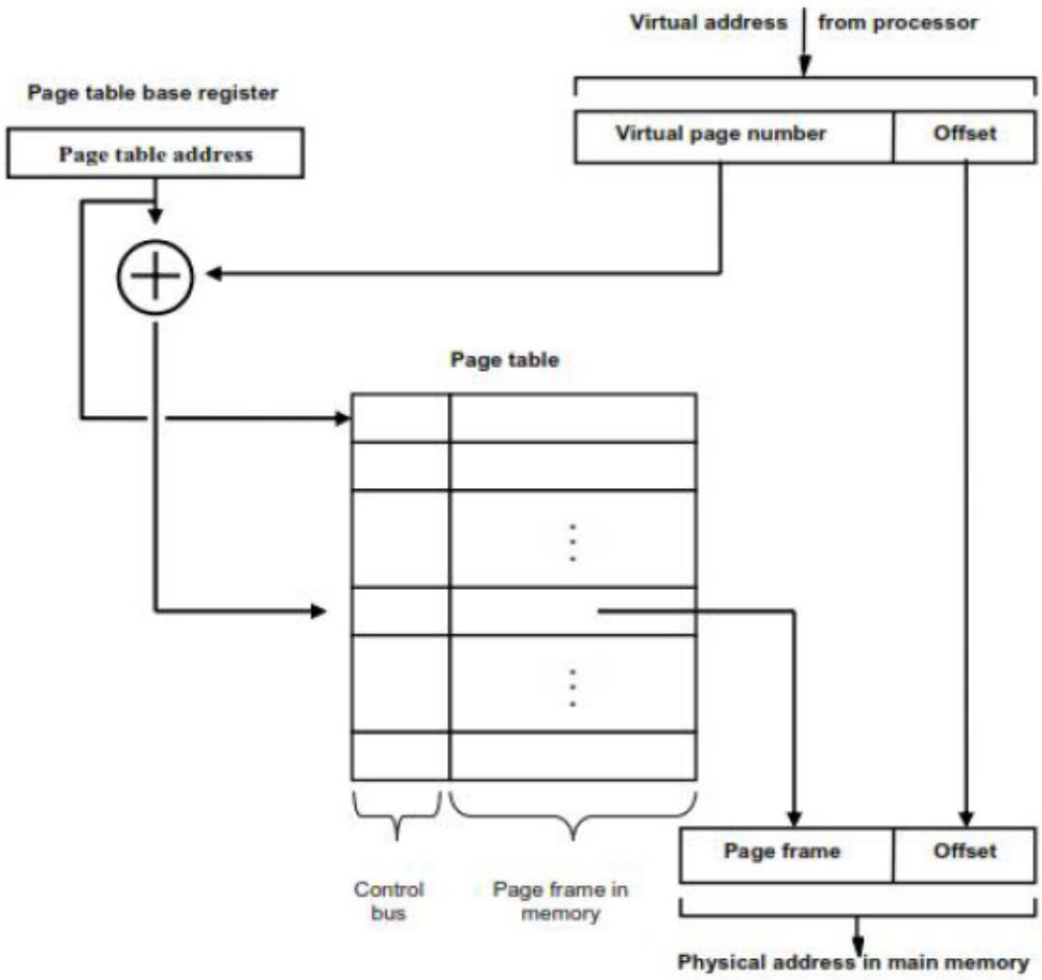


Figure.5.26. Virtual memory addresses Translation:

.Page table

- The page table is used to keep the information about the main memory location of each pageThis information includes the main memory address where the page is stored and the current status of the pageTo obtain the address of the corresponding entry in the page table the virtual page number is added with the contents of page table base register, in which the starting address of the page table is storedThe entry in the page table gives the physical page number, in which offset is added to get the physical address of the main memoryAn area in the main memory that can hold one page is called a **page frame**The starting address of page table is kept in a page table registerEach entry in page table also includes some control bits that describes the status of the page, which it is in the main memoryOne bit indicates the validity of the page, that is, whether the page is actually loaded into the main memoryThe bit allows the operating system to invalidate the page without actually removing itAnother bit indicates various restrictions that may be imposed on accessing the page.

5.5.3.2. Page fault

Given virtual address, the MMU looks in the TLB for the referenced pageIf the page table entry for this page is found in TLB, the physical address is obtained immediately.If there is a

miss in TLB, then the required entry is obtained from the page table in the main memory and the TLB is updated.When a program generates an access request to a page that is not in the memory, a **page fault** is said to have occurred and the required page is loaded into the main memory from the secondary storage by special routine called **page fault routine.**This technique of getting the desired page in the main memory is called **demand paging.**

8. Explain the translation look Aside Buffer. (Nov/Dec 2008)

Translation Look aside Buffer (TLB)

To support demand paging and virtual memory processor has to access page table which is kept in the main memoryTo avoid the access time and degradation of performance, a small portion of the page table is accommodated in the MMU.This portion is called **Translation Look aside Buffer** and is used to hold the page table entries that corresponds to the most recently accessed pagesWhen the processor finds the page table entries in the TLB, it does not have to access the page table and saves substantial access time**Over View:**Associative Mapped TLBPerformance of Demand Paging

Associative Mapped TLB

The page table information is used by the MMU for every read and writes access. So the page table should be situated within the MMU. But the page table may be large as it depends on the program.It is impossible to include a complete page table as the part of processor chip. Therefore the page table is kept in the main memory.However a copy of small portion of the page table can be accommodated with the MMUThis portion consists of the page table entries that corresponds to the recently accessed pagesA small cache, usually called TLB is incorporated into the MMU for this purposeIn addition to control bits and page frame the page table also included the virtual address of the entry

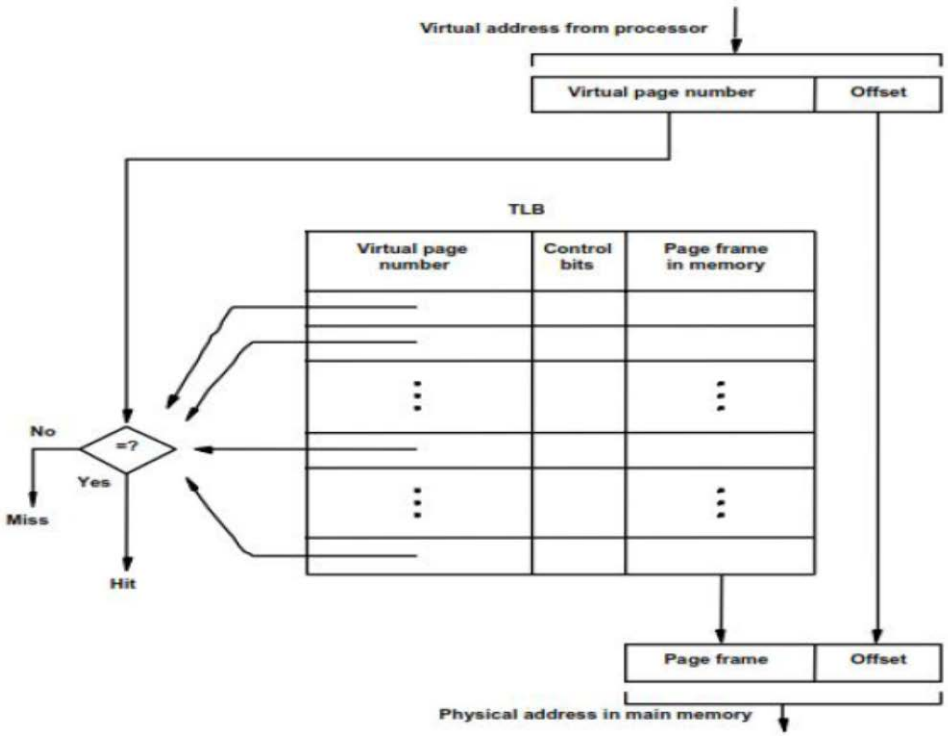


Figure.5.27. Use of an Associative Mapped TLB

Performance of Demand Paging When page fault occurs it is necessary to fetch, decode and execute the instruction which caused the page fault. Demand paging can have a significant effect on the performance of a computer system. As long as we have no page faults, the effective access time is equal to the memory access time. If , however , a page fault occurs, the time required to read the relevant page from disk is added in the total memory access time.e effective address time if average page-fault service time of 20 milliseconds and a memory access time of 80 nanoseconds. Let us assume the probability ofapagefault10%Solution:**Effective access time is givenas**
=(1-0.1)x80+0.1(20milliseconds) =(1-0.1)x80+0.1x20,000,000
=72+2,000,000(nanosconds)
=2,000,072(nanoseconds).

9. Give detail description about the Input/output system.
Input/output system
The main data processing functions of a computer involve its CPU and external memory M. The CPU fetches instructions and data from M, processes them, and eventually stores the results back in M. The other system components – secondary memory, user interface devices, and so on – constitute the input – output (IO) system. **Over View:**Programmed I/O DMA
Interrupts,I/O processors.

6.1.1. Programmed I/O. (Nov/Dec-2010,2012)
Programmed input/output, a method for controlling input/output operations, which is included in most computers. It is particularly useful in small low-speed systems where hardware costs must be minimized.It requires that all input/output operators be executed under the direct control of the CPU, i.e., every data-transfer operation involving an input/output device requires the execution of an instruction by the CPU.Typically, the

transfer is between CPU registers (e.g.), the accumulator and a buffer register connected to the input/output device. The input/output device does not have direct access to main memory.A data transfer from an input/output device to main memory requires the execution of several instructions by the CPU, including an input instruction to transfer a word from the input/output device to the CPU and a store instruction to transfer a word from CPU to main memory. One or two additional instructions may be needed for address computation and data word counting.

Over View:

Input-output addressing
Memory Mapped Input-output
I/O Mapped IO

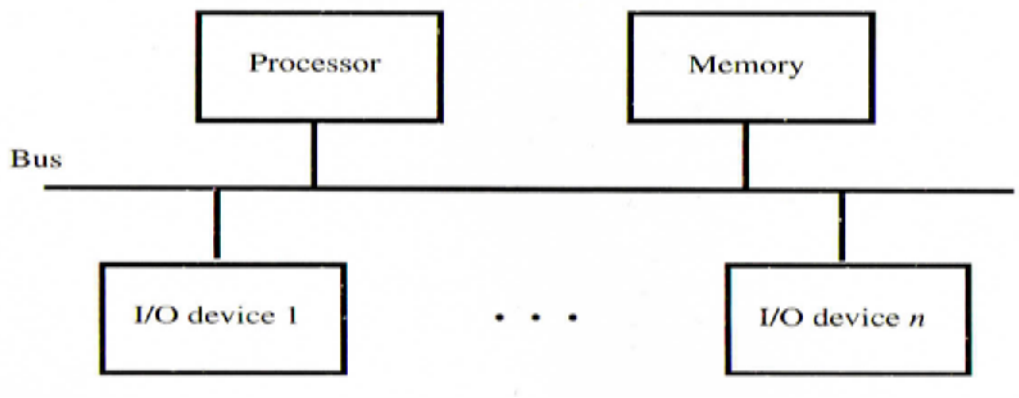
IO instructions

Input / Output Addressing

In systems with programmed input/output, in devices, main memory and the CPU normally communicate via a common shared bus.The address lines of that bus which are used to select main-memory locations can also be used to select input/output devices.Each junction between the system bus and input/output device is called an input/output port and is assigned a unique address.The input/output port includes a data buffer register, thus making if little different from a main memory location with respect to the CPU.

Memory-Mapped I/O: Nov/Dec-2010

When I/O devices and the memory share the same address space, the arrangement is called memory-mapped I/O.



- The machine instructions that can access memory are used to transfer data to or from an I/O device.

Figure.6.1. Single Bus structure

For example,

if DATAIN is the address of the input buffer of keyboard, the instruction.

MOVE DATAIN, R0- Reads the data from DATAIN and stores them into processor register R0.Similarly DATAOUT is the address of the output buffer of display unit or printer the instruction.MOVE R0, DATAOUT – sends the data from R0 to location DATAOUT.

I/O Mapped I/O: (Nov/Dec-2010)

Here the I/O devices and the memory have different address space it has special I/O instructions.The advantage of a separate I/O address space is that I/O devices deals with fewer address lines. **I/O Interface:**The hardware required to connect to an I/O device to the bus is given below.**Address Decoder:**It enables the devices to recognize its address when it appears on the address line.**Data Register:**It holds the data being transferred to or from the processor.**Status Register:**It contains information relevant to the operation of the operation of the I/O devices.

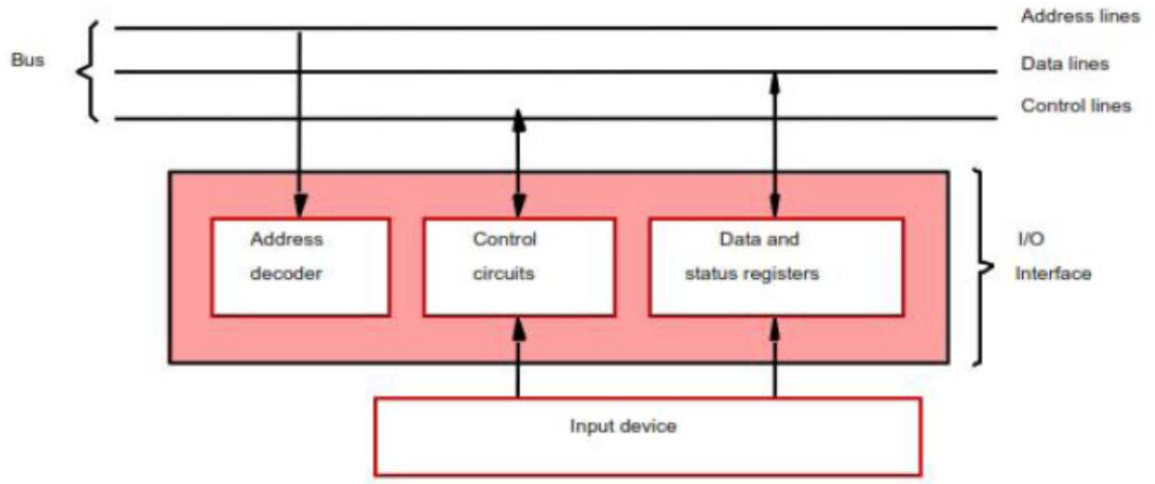


Figure.6.2. I/O interface for an input device

IO INSTRUCTIONS:
Programmed IO can be implemented by a few as two IO instructions for **Example**, the Intel 8085 has two main IO instructions. The INX causes a word to be transferred from IO port X to the 8085's accumulator register.

10.Discuss Direct Memory Access in detail.(Apr/May -2014 ,May/Jun- 2012) (Or) With a neat sketch explain the working principle of DMA.(Nov/Dec-2011)(or) Bus Arbitration techniques in DMA. (Nov/Dec-14)A (Direct Memory Accesses).
To allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor, a **special control unit** is provided. This approach is called **Direct access memory or DMA**.**OVER VIEW:.**MA ControllerDMA RegistersBus Arbitration **DMA Controller:**DMA transfers are performed by a control circuit that is part of input/output device interface. This circuit is DMA controller. The DMA controller performs the functions that would normally be carried out by the processor when accessing the main memory. For each, word transferred, it provides the memory address and all the bus signals that control data transfer. Since it has to transfer blocks of data, the DMA controller must increment the memory address for successive words and keep track of the number of transfers.DMA controller can transfer data without intervention by the processor, its operation must be under the control of a program executed by the processor.To initiate the transfer of a block of words, the processor sends

the starting address, the number of words in the block and the direction of the transfer.When the entire block has been transferred, the controller informs the processor by raising an interrupt signal.

An example of a computer system is given in figure showing how DMA controllers may be used.

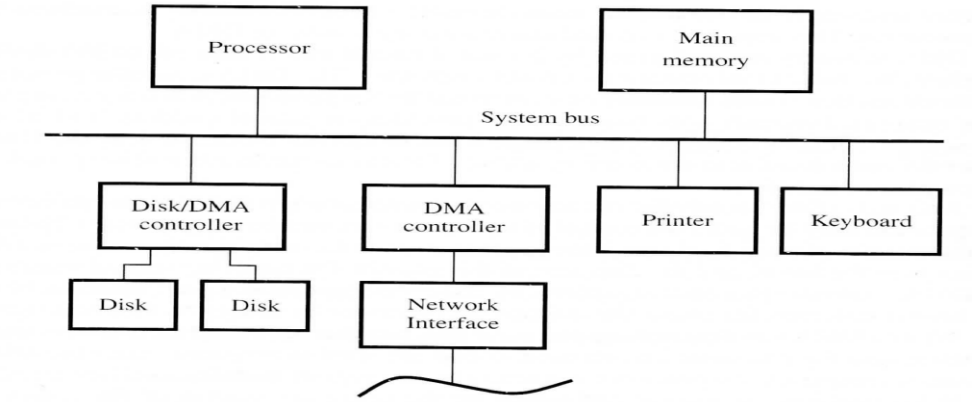


Figure.6.3. Use of DMA controller in a computer system

- A DMA controller connects a high speed network to the computer bus. The disk controller which controls two disk, also has DMA capability and provides two DNA channels.
- It can perform two independent DMA operations.
- To start a DMA transfer of a block of data from the main memory to one of the disks, a program writes the address and word count information in to the resistors of the corresponding channel of the disk controller.

When the DMA operation completed, this is recorded in the control and status register of the DMA channel memory accesses by the DMA controller is given higher priority than the processor requests for memory access.The processor begins most memory access cycles, the DMA controller can be said to steal memory cycles from the processor, this technique is called **cycle staling**.The DMA controller may be given exclusive access to the main memory to transfer a block of data without interruption. This is known as **block or brust mode.**A conflict may arise if both the processor and a DMA controller or two DMA controllers try to use the bus at the same time to access the main memory. To resolve these conflicts an abstraction procedure is implemented on the bus. **DMA Registers**DMA controller registers that are accessed by the processor to initiate transfer operation. There are four types of DMA registers**DMA address register Word count register Control register Status register**DMA address register holds the starting address of the memory block. Word count register contains the number of DMA cycles.

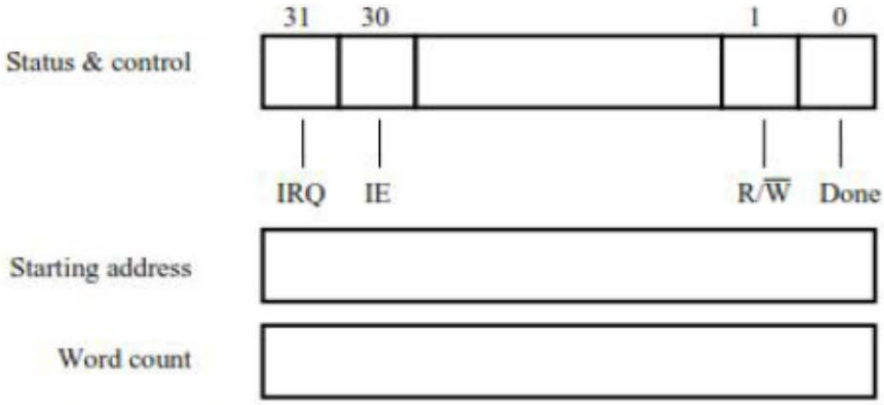


Figure.6.4.Registers in a DMA interface DMA controller connects a high speed network to the computer bus and controls two disks.

Bus Arbitration. The device that is allowed to initiate data transfers on the bus at any given time is called the **bus master**.Bus arbitration is process by which the next devices to become the bus master is selected and bus master ship Is transferred to it. The selection of bus transfer is done on a priority basis.

There are two approaches to bus arbitration

Centralized arbitration

Distributed arbitration

Centralized Arbitration:

In centralized arbitration, a single bus arbiter performs the required arbitration.The bus arbiter may be processor or a separate unit connected to the bus.

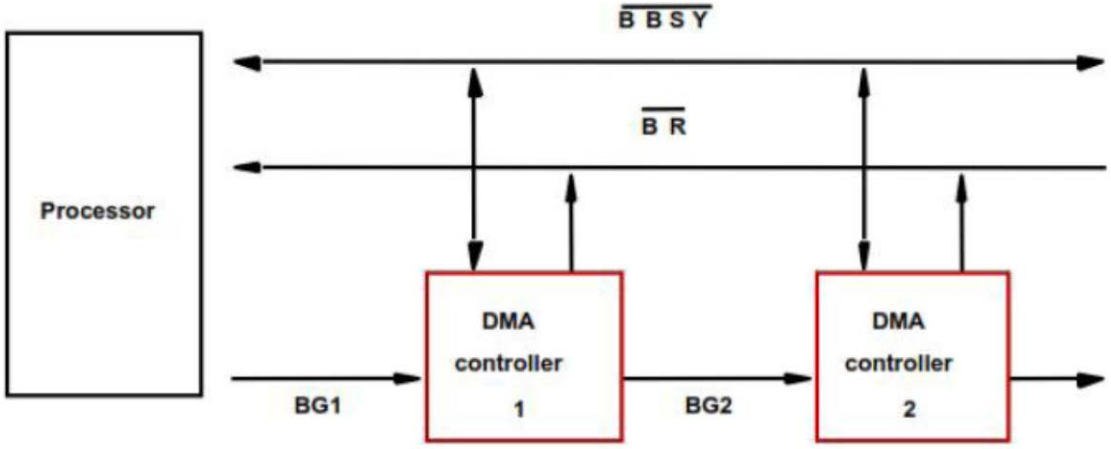


Figure .6.5. A simple arrangement for bus arbitration using a daisy chain

- In this case, the processor is normally, the bus master unless is grants bus masterships to one of the DMA controllers.A DMA controller indicates that it needs to become the bus master by activating the bus request line BR.When the bus request is activated the processor activates the bus grant signal, BG, indicating to the DMA controllers that they may use the bus when it becomes free.This signals is connected to all DMA controllers using a daisy chain arrangement, if DMA controller 1 is requesting the bus it blocks the propagation of the grant signal to other devices, otherwise it passes the grant signal. The current bus master indicates to all devices that is using the bus by activating another line called BUS.Busy, BBSY. Hence after receiving the BUS grant signal, a DMA controller waits for BUS.

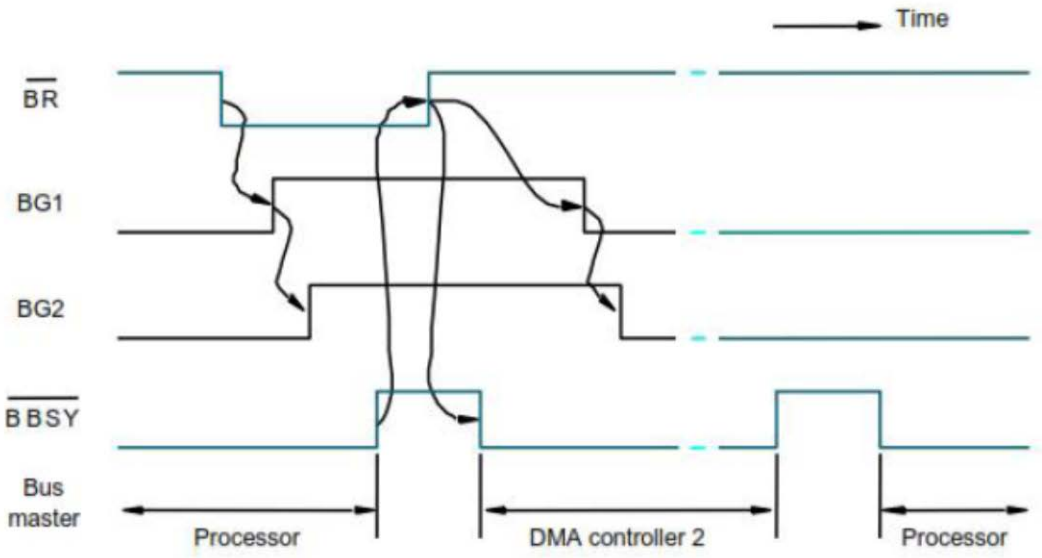


Figure.6.6.Sequence of signals during transfer of bus mastership for the devices

Busy to become inactive, and then get the mastership os the bus. At the time it activates bus. Busy to prevent other devices from using the bus at the same time.**Adavantage**It is simple and cheap.It requires the least number of lines and this number is independent of the number of master in the system.**Disadvantage**The priority of the master is fixed by its physical location.The use of the daisy chain grant signal also limits bus speed.Failure of any one master causes the whole system to fail.

Distributed Arbitration:

Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration.

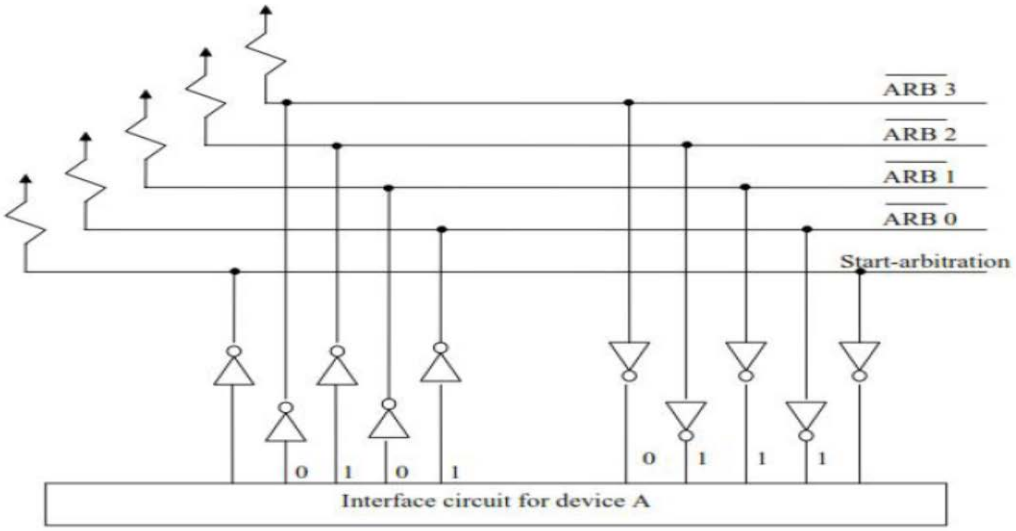


Figure.6.7. A distributed arbitration schem

Distributed arbitration may be implemented in two ways.

Self selection

Collision detection In **self selection scheme**, each device requesting the bus places a code representing its identity on the bus and immediately examines the bus to figure out if it has the highest priority among all devices requesting the bus.The codes assigned to the device are enough for telling whether to requesting device has the highest priority.

For example, with 8-bit bus codes used are

Device 1 – 0 0 0 0 0 0 0 1

Device2 – 0 0 0 0 0 0 1 1

Device 3 – 0 0 0 0 0 1 1 1

Device 8 – 1 1 1 1 1 1 can be assigned to eight devices connected to an 8-bit bus.

In collusion detection the requesting device simply uses the bus, but monitors the bus to see whether multiple transmissions have been mixed up or collide with each other.

11. Explain the Interrupt priority schemes (May/Jun -2012) (or) What are the steps to handle interrupts. (Nov/Dec-2011)

Interrupts:

An interrupt is an event inside a computer system requiring some urgent action by the CPU. In response to the interrupt, the CPU suspends the current program execution and branches in to an Interrupt Service Routine(ISR).

- The ISR is a program that services the interrupt by taking appropriate actions. After the execution of ISR, the CPU returns back to the interrupted program. On returning from ISR, the CPU resumes the old interrupted program as if nothing has taken place. This means the CPU should continue from the place when interruption has occurred and the CPU status at that time should be the same. For this purpose, the condition of the CPU should be saved before taking up ISR. Before returning from ISR, this saved status should be loaded back into the CPU.The processor can perform some useful task while waiting for I/O device to become ready.

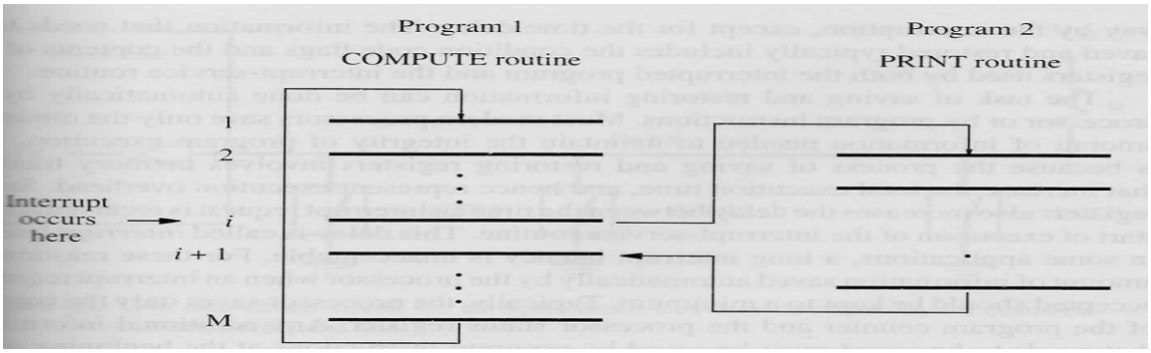


Figure.6.8. Interrupt Service Routine

To allow this to happen, the I/O devices can be allowed to alert the processor when it becomes ready.If can be done by sending a hardware signal called interrupt to the processor. Interrupt Request Line Is used for this purpose.Interrupts eliminates waiting period. The routine serviced in response to an interrupt request is called interrupt service routine. The processor acknowledges the interrupt by interrupt acknowledgement signal.
The interrupt service routine is very similar to sub routine.The task of saving and restoring information can be done automatically by processor or by program instruction.Saving and restoring registers requires memory transfers are ISR overhead.The time between when an interrupt request is received and the start of execution of ISR is called interrupt latency.The routine executed in response to an interrupt request is called the interrupt service routine, which is the PRINT routine is our example.Interrupts bear considerable resemblance to subroutine calls.Assume that an interrupt request arrives during execution of instruction I in figure.The processor first completes execution of instruction I, and then it leads the program counter with address of the first instruction of the interrupt service routine. **Overview:**
Interrupt Hardware
Enabling and Disabling Interrupts
Handling Multiple Devices
Vectored Interrupts
Interrupts Nesting
Interrupt Priority
Controlling Device Requests
o Exceptions
o Use of interrupts in operating System

6.3.3.1. Interrupt Hardware NOV/DEC-2010

- An I/O device requests an interrupt by activating a bus line called **interrupt request**. Computers can request an interrupt. The interrupts are classified as:
  - **Single level interrupt**
  - **Multi Level interrupt**

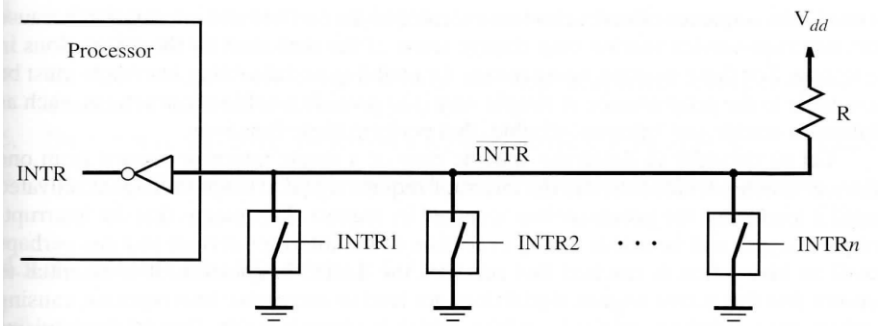


Figure.6.9. Common Interrupt Request Line

A single interrupt request line may be used to serve an I/O device as shown in figure
All devices are connected to interrupt request line via switches to ground. The device to raise interrupt closes the switch.If all interrupt request lines of I/O devices are inactive, then interrupt request line will be equal to V_{dd} . when a device attempts to raise an interrupt by closing the switch, the voltage drops to 0, causing the interrupt request signal INTR, received by the processor to go 1, INTR is logical OR (or) the interrupt request from individual devices.**$INTR = INTR_1 + INTR_2 + ….. + INTR_n$**

Enabling and Disabling Interrupts:

The arrival of an interrupt request from an external device cause the processor to suspend the execution of one program and start the execution of another.A simple way is to provide machine instructions, such as interrupt-enable and interrupt-disable that perform this function.When a device activates the interrupt request signal, it keeps this signal activate until it is serviced by processor.
This ensures that this active request signal does not lead to successive interruptions, causing to enter an infinite loop.

This can be handled by the following ways.

The **first** possibility is to have the processor ignore the interrupt request line until the ISR is completed. This can be done using interrupt disable as the first instruction in ISR and interrupt enable as the last instruction.The **second** option is processor while saving the contents of program counter (pc) and programs status Register(PS) on the stack, it also performs an equivalent of executing an interrupt disable instruction (interrupt enable bit =0), while after completing ISRH is enabled.

The sequence of events in handling interrupts can be summarized as follows.

The device raises an interrupt request.
The processor interrupts the program currently being executed.
Interrupts are disabled by interrupt disable instruction or by setting interrupt enable bit to0The device is informed that is interrupt request has been recognized by interrupt acknowledgement.The ISR Is serviced.Interrupts are enabled and execution of the interrupted program is resumed.

Handling Multiple Devices:

Consider the situation where number of devices capable of initiating interrupts is connected to processor. The devices are operationally independent.
Polling Scheme
Vectored interrupts
Interrupt nesting
Interrupt priority
Daisy chain
Priority Groups

Polling Scheme: It is a simplest scheme to identify the interrupting device. The device that raises the interrupt will set one of the bit d (IRQ) in status register t01.The processor will

poll the devices to find which raised an interrupt.**Disadvantage**In this technique is ti9me spent in interrogating the IRQ bits of all devices.**Vectored interrupts: (NOV/DEC-2010)**
To reduce time involved in polling process, a device requesting an interrupt may identify itself directly to the processor.The code is used to identify the device and it may represent the starting address of the interrupt-service routine for that deviceIf the interrupt produces a call to a predetermined memory location, which is starting address of ISR, then that address is called vectored address and such interrupt are **called vectored interrupts.**

Interrupt Nesting:
For some device, a long delay in responding to an interrupts request may lead to error in the operation of computer.Such interrupts are acknowledged and serviced even though processor is executing an interrupt service routine (ISR) for another device.A system of interrupts that allow an interrupt service routine to be interrupted is known as **nested interrupts.**

Interrupt Priority: : (May/Jun -2012)
When interrupt request arrives from two or more devices simultaneously, the processor has to decide which request should be serviced first. And which one one should be delayed.The processor takes this decision with the help of interrupt priorities. The processor accepts interrupt request having highest priority.

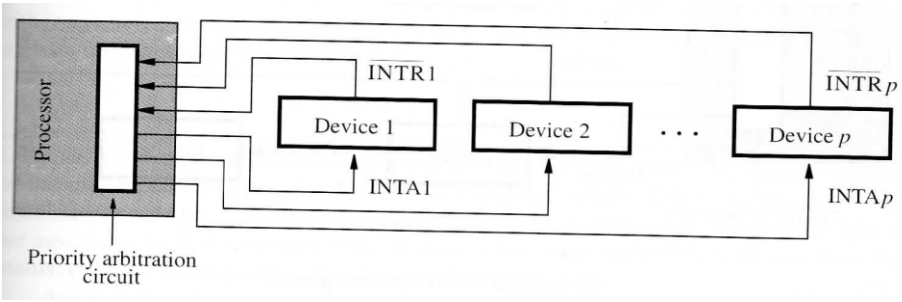


Figure.6.10.

Handling Priority Interrupts-Daisy chain:

Daisy chain :Consider the problem of simultaneous request from two or more devices. The processor has to decide which request to be serviced first.Priority is determined by the order in which devices are polled. In daisy chain scheme, interrupt request line INTR is common to all devices. The interrupt acknowledgement INTA, propagates serially through the devices.

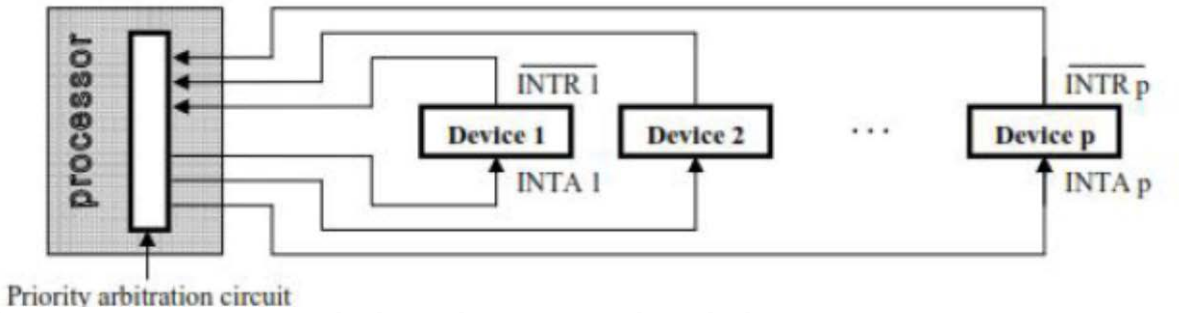


Figure.6.11. Interrupt priority schemes – Daisy chain

When several devices raise an interrupt request, the processor responds by setting INTA line to 1. The signal is received by device 1Device 1 passes the signal on to device 2 only if it does not require any service. If device 1 has a pending request for interrupt, it block S the INTA signal and proceeds to put is device identifying code on the data lines.Therefore, in daisy chain arrangement, the device that is closest to the processor has the highest priority.

Priority groups:

In this scheme, devices are organized in groups, and each group is connected at a different priority level within a group, the devices are connected in a daisy chain.

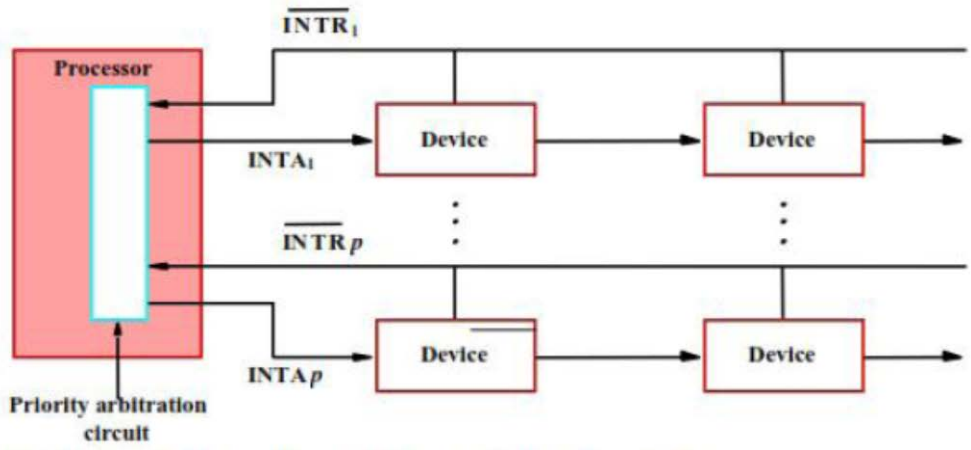


Figure.6.11. Interrupt priority schemes Arrangement of priority groups

6.3.3.5 Controlling device request:

I/O device generate an interrupt request when it is ready for data transfer, while handing interrupt requests it is requested to ensure that interrupts request are generated only by those I/O devices that are being used by a given program.

There are two mechanism used to control device request.

- **One is at the device end**
- **And the other at processor end**

- At the device end, an interrupt enable bit in the control and status register. The programmer is allowed to set or reset this interrupt-enable bit.

- It the processor end, either an interrupt-enable in the program status register (PS) or a priority structure determines whether the given request will be accepted.

Exceptions:

An interrupt is an event that cause the execution of one program to be suspended and begins execution of another program . Exception is referred to any event that causes **an interruption**. I/O interrupt is an example of exception**Faul**Faults are exceptions that are detected and serviced before the execution of the faulting instructionExample: Page fault in virtual memory system.**Traps**Traps are exceptions that are reported immediately after the execution of the instructions which causes the problem.Example: op-code field of instruction may not correspond to any level instruction.**Aborts:**These are exceptions which do not permit the precise location of the instructions causing the exception to be determined. They report severe error-hardware error. **Use Of Interrupts In Operating System:**The operating system is responsible for coordinating all activities within a computer. It makes extensive use of interrupts to perform input/output operations and communicate with and control the execution o user programs. The interrupt mechanism enables the operating system to assign priorities, switch from one user program to another, implement security and protection features and coordinate input/output activitiMultitasking is a mode of operation in which a processor executes several user programs at the same time. Each program runs for a short period called a time slice, then another program runs for its time slice and so on.

Operating system routines:

a) OS Initialization, services and scheduler

OSINIT	set interrupt vectors. Time-Slice-Clock-SHEDULER Software interrupt-OSSERVICES Keyboard interrupt-IO Data.
OSSERVICES call	Examine stack to determine requested operation appropriate routine.
SCHEDULER	Save program state. Select a runnable process. Restore saved context to of new process. Push new values for PS and PC on stack. Return from interrupt.Input/output routines
IOINIT	Set process status to Blocked.Initialize memory Buffer address pointer and counter.Call device driven to initialize device an enable interrupts in the device interface.IODATAPoll devices to determine source of interrupts call appropriate f character = CR, then{ set END = 1; Disable interrupts} lse set END=0 Return from subroutine. At the time the operating system is started, an initialization routine, called OSINIT is executed.

OSINIT loads the starting address of a routine called SCHEDULER in the interrupt vector corresponding to the timer interrupt. Hence at the end of each time slice, the timer interrupt causes this routine to be executed.The current state of execution of program is called a process. A process can be in one of the three states.**Running** -the program is currently being executed.**Runable** – The program is ready for execution but is waiting to be selected

by the scheduler.**Blocked** – the program is not to be ready to resume execution for some reason. It may be waiting for completion of an input/output operation.Assume program A is in the Running state during a given time slice. At the end of the time slice, the times interrupt the execution of this program and start the execution of SCHEDULER. The information saved, which is called the program state, include register content, the program, counter, and the processor status word.

13. Explain the IO processors.(Nov/Dec-2013)(or)) Explain in detail about any two Standard Input and Output Interface required to connect the I/O device to the bus. (NOV/DEC-14)

PROCESSORS:

The IO processor (IOP) is a logical intention of the IO control methods. In systems with programmed IO, peripheral devices are controlled by the CPU.

The DMA concepts extend to the IO devices limited control over data transfers to and from main memory. An IOP has the additional ability to execute certain instructions, which give it more complete control over IO OPERATIONS.An IOP, like CPU, is an instruction set processor, but it usually has a more restricted instruction set than the CPU, IOPs are primarily communications links between IO devices and main-memory hence the use of the term" channel" for IO.IOPs have also been called peripheral processing units to emphasize their subsidiary role with respect to the CPU.**Over view:**IO Instruction typesIOP Organization **IO Instructions Types**:In a computer system with IOPs, the CPU does not normally execute IO data transfer instructions. Such instructions are contained is IO programs stored in main memory and are fetched and executed by the IOPs.The CPU executes a small number of IO constructions which allow it IO system. The IO instructions executed by the IOP are primarily associated with data transfer. A typical IOP instruction has:READ (WRITE)- a block of n words from device X to memory region Y.The IOP is provided with direct access to main memory and so can control the system bus when that bus is not required by the CPU.An IOP can execute a sequence of data transfer operations involving different regions of main memory and different IO devices, without CPU intervention.Other instruction type such as arithmetic, logical and branch may be included in the IOPs instruction set to facilitate the calculation of complex addresses, IO devices priorities etc.A third category of IO instructions includes those executed by specific IO devices. These instructions control functions such as REWIND SEEK ADDRESS OR PRINT LINE.Instructions are this type are fetched by the IOP and transmitted as data to the appropriate IO device.

Format diagram

Figure (a)

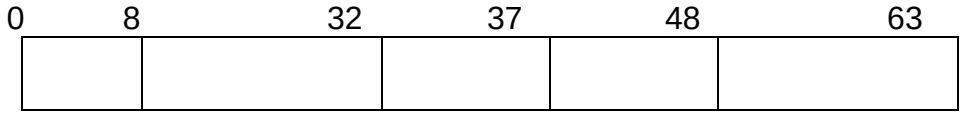


Figure (b)

Figure.6.12. formats of system/360 IO instructions executed (a) by a CPU and (b) by an IOP (channel)

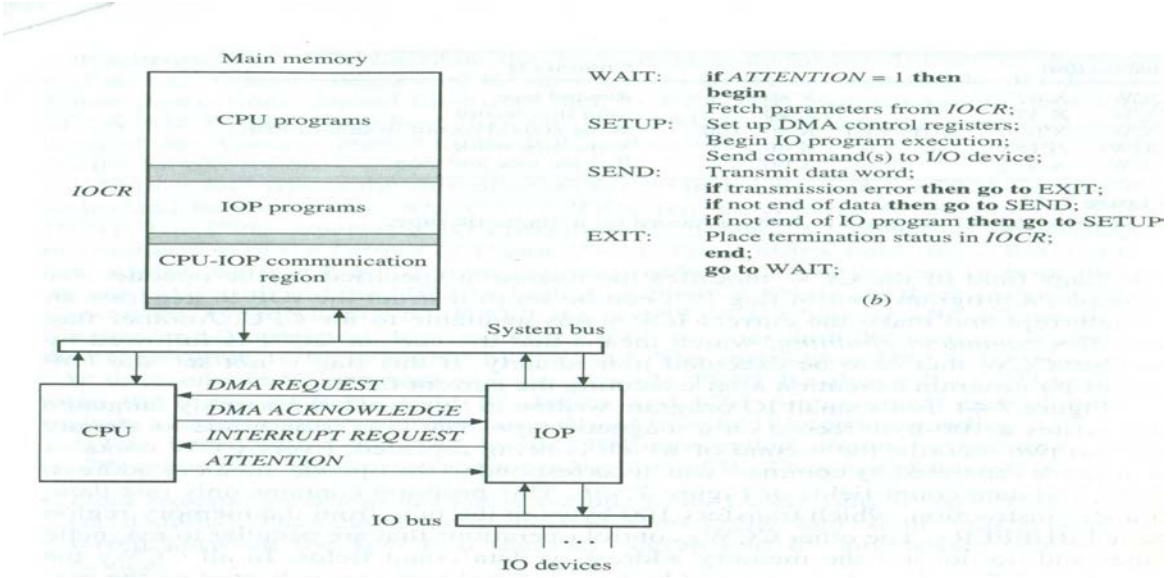
The CPU supervises IO operations by means of a small set of privileged IO instructions with the format of fig. The address field specifies a based register B and a displacement D, which identify both the IO device to be used and the IOP to which it is attached. There are three major instructions of this type.START IOHALT IOTEST IO**START IO** is used to initiate an IO operation. It provides the IOP it names with the main memory addresses of the IO program to be executed by the IOP.**HALT IO** causes the IOP to terminate IO program execution.**TEST IO** allows the cpu to determine the status of the named IO device and IOP.The IO instructions executed by IOP are called channel command words(CCWS) and have the format shown in fig.There are 3 main type.**Data transfer instructions** including input (read), output(write) and sense. These CCWS causes the number of bytes specified in the data count field to be transferred between the specified main memory.**Branch instructions** which cause the IOP to fetch the next CCW from the specified memory address rather than the next sequential locations. This is a simple unconditional jump within IO program.**IO device control instructions**. These instructions are transmitted directly to the IO device and are used to specify functions peculiar to that device which do not involve data transfers.The opcode of data transfer instruction may be transmitted directly to the IO device as the command byte during a device selection process. If the IO device requires additional control information it can be supplied with it via an output data transfer. The flags field of the CCW is used to modify or extend the operation specified by its opcode. Another flag specifies "**command changing**", which means that the current CCW is followed by another CCW is to be executed immediately. If this flag is not set, the IOP causes IO program execution after executing the current CCW.

IOP ORGANIZATION:

The structure of a representative system containing an IOP appears in fig. The IOP and CPU share access to a common main memory M via the system bus. M contains separate CPU programs for execution by the CPU and the IOP; It also contains a communication region IOCR used for passing information in the form of messages between the 2 processors.The CPU can place there the parameters of an IO task, eg. The address of the IO program to be executed, and the identity of the devices to be used. The CPU and IOP also communicate more directly via special control lines. Standard DMA or bus grand/acknowledge lines are used for arbitration of the system bus between the 2 processors. The CPU can attract the IOP program whose specifications have been placed in the IOCR communication area.In an essentially similar fashion, the IOP attracts the CPUs attention by activating one or more INTERRUPT REQUEST lines, causing the CPU to execute an interrupt-service routine that responds to the IOP by, for instance, defining a new IO program for the IOP to execute.An IO operation begins when the CPU encounters a START IO instruction whose address and an attention in fig. This causes the CPU to transmit the IO device address and an attention signal to the specified IOP. The IOP then fetches the channel address word CAW previously placed in a memory location 72, which is a part of the CPU-IOP communication region. The CAW contains the absolute 24-bit starting address of the IO program to be executed by the IOP, as well as a memory protection key.

- The IOP next initializes the specified IO device for the IO operation by carrying out an IO device selection procedure via a standard communication protocol over the S/360-370 IO bus.**Representative system containing an**

IOP



The CPU can maintain direct control over the IO operation by periodically executing TEST **Figure (a)**

Figure.6.13.Computer containing an IOP: (a) System organization and (b) CPU – IOP interactionIO. This causes the IOP to construct a channel status word CSW, which it stores at memory location 64. The CPU can fetch and examine it.

Interrupt request can be masked by an interrupt mask register which forms part of the CPU's program status word register PSW.