UNIT-I

Computer Organization & Instruction

Introduction

* Computer architecture acts as the interface between the hardware and lowest level software

* Computer architecture is defined as study of the Structure, behavior, and design of Computers.

Computer Organization: It refers to the Operational Units and their InterConnections that realize the architectural Specification. It describes the function of and design of the various Units of digital Computer that Store and Process information. The attributes in Computer Organization refers to.

* Control Signals.
* Computer/ Peripheral Interface.
* Memory technology.

Computer Hardware: It consists of electronic circuits displays, magnetic and optical storage media, electromechanical equipment and Communication facilities.

Computer architecture: It is Concerned with the Structure and behavior of the Computer. It includes the Information formats, the Instruction Set and techniques for addressing memory.

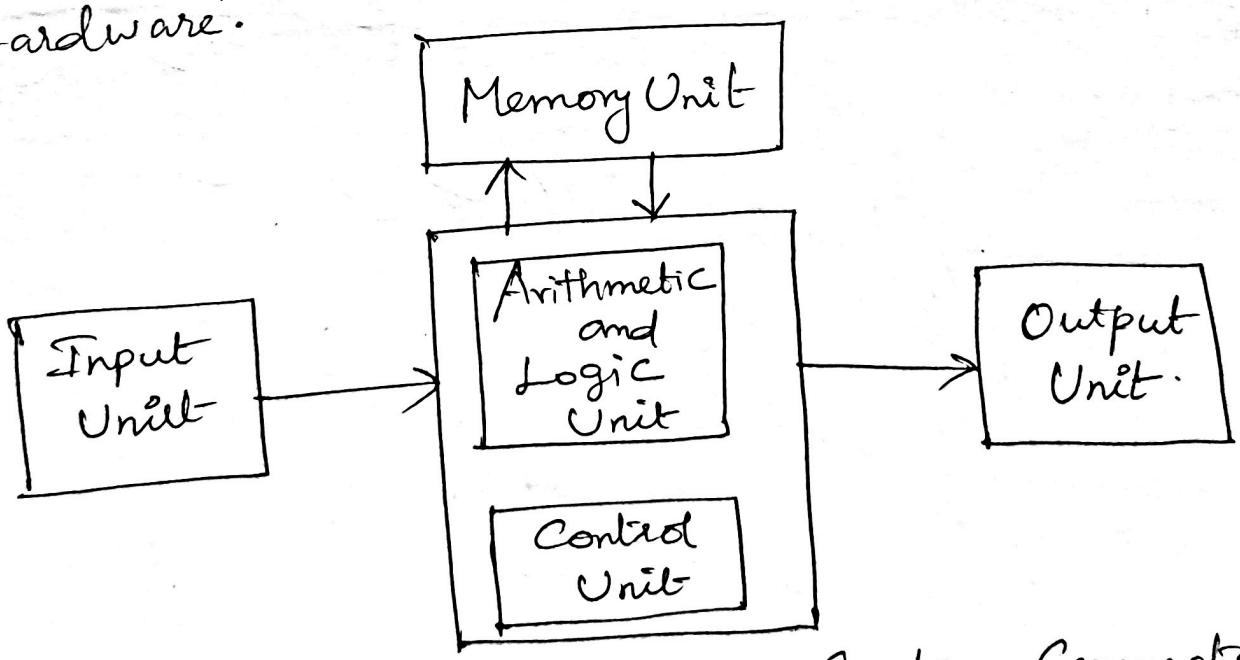
→ The attributes of Computer architecture refers to the

* Instruction Set.
* Data representation.
* I/o Mechanism.
* Addressing techniques.

Basics of Computer System.

A Computer in its simplest form is a fast electronic calculating machine that accepts digitized information from the user, Processes it according to a sequence of instructions and Provide the Processed information to the user.

Hardware: Electronic Components Inter Connected in the Computer System as a whole is referred to as Hardware.



Components of Computer System Connected as Hardware.

Input Unit: * It Provides the data to the Computer System from the Outside. So it links the External Environment with the Computer. It takes the data from the Input devices, Converts it into machine language and then loads it into the Computer Systems.

* Most Commonly Used input devices are keyboard and mouse.

Output Unit * It Provides the results of Computer Process to the Users. ie) It links the computer with the external environment.

* Different Output devices are monitors Printers, Speakers, Headphones etc.

Memory Unit * Memory Unit is Used to Store Programs and data.

* Primary Storage memory and Secondary Storage memory devices form a total memory Unit.

* Primary memory is directly accessible by CPU. Secondary memory is not accessible by CPU.

* Primary memory (main memory) is fast Semi Conductor RAM.

* Secondary Storage memories Such as magnetic tapes, magnetic disk are Used for the Storage of larger amount of data.

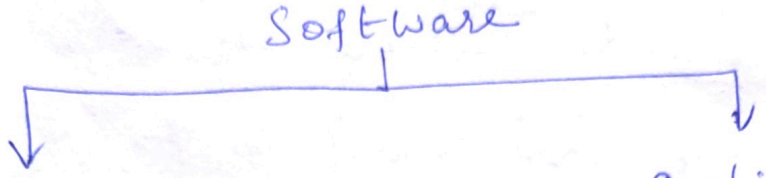
Arithmetic and Logic Unit:

* ALU is responsible for Performing arithmetic and logical operations.
* To perform these operations operands from main memory are brought into internal registers of Processor.
* After Performing Operation the result is either Stored in the register or memory location.

Control Unit

* A Control Unit Co-ordinates and Controls all the activities amongst the functional Units
* Control Unit Transfers data from Storage Unit to arithmetic logic Unit when Calculation need to be Performed.
* The Combine Unit of arithmetic logic Unit and Control Unit is called as Central Processing Unit.

Software: Computer Software is divided into two broad categories: System Software and application software



System Software

Example: Operating System
Device Drivers
Language translators
System utilities

Application Software

Example: Word Processor
Spread Sheet
Image editor
Database Management System

System Software: * It is collection of Programs which is needed in the Creation, Preparation and execution of other program.

* Typical System software includes editors, assemblers, linker, loader, Compiler, debuggers.

Application Software: Allows to perform Specific task on a Computer using the Capabilities of a Computer.

Evolution of Computer Systems

First Generation (Vaccum Tubes)

* First Electronic Computer ENIAC (Electronic Numerical Integrator and Computer)
* It was made up of more than 18000 Vaccum tubes and 1500 relays.
* It was able to perform nearly 5000 additions or Subtractions per second.

* Its data memory consists of 20 accumulators each capable of storing a ten digit decimal number

Features. Weight - 30 tons, area - 15000 Sq.ft.
Power Consumption - 140 KW.

Draw back - It was wired in for specific computation.

Second Generation: (Transistors).

* Transistors are smaller, cheaper and they dissipate less heat ie) low power consumption.
* Greater speed, larger memory capacity and smaller size than first generation.
* Due to addition of a set of index register and arithmetic circuits, CPU can handle both floating point and fixed point operations.
* Introduction of more complex arithmetic and logic units and control unit to support higher level programming languages. for example FORTRAN.
* Magnetic core memories and magnetic drum storage devices were more widely used.

Third Generation. (Integrated Circuits).

* Uses Integrated circuits.
* Integrated circuit technology enabled lower cost, faster processors and developement of memory chips.

* Integrated circuit technology allowed to increase memory size and number of I/O ports.
* Magnetic core memories were replaced by Integrated circuit memories.
* Various techniques were introduced to improve the performance of the Computer. these are.

→ Microprogramming
→ Parallel processing as Multiprocessing & Pipelining.
→ Sharing resources.

* Introduction of IBM System/360 - first planned family of Computer Products.

Feature of IBM 360-370.

* It uses 32 bit or 4 byte word format
* It has 16 general register and 4 double length floating point registers.
* Increasing speed, Increasing number of I/o Port.

VLSI (Very Large Scale Integrated Circuits)

* Uses Large scale Integration and Very Large Scale Integration (VLSI) technology for Computer design.
* VLSI technology made it possible to fabricate an entire CPU, main memory or Similar device with a Single IC that can be mass Produced at Very low cost.

* fourth generation Products are portable notebook Computers, desktop Computers and workstation interconnected by local area network, wide area network and the internet.

ULSI (fifth Generation)

* Ultra Large Scale Integration is the Process of integrating or embedding millions of transistors on a Single Silicon Semiconductor microchip
* It is a Successor to large scale integration (LSI) and very large scale integration.
* Parallel Processing Technique Used.

Performance:

* The performance is an important attribute of a Computer. It is an important Criteria for selection of a Computer.

Defining Performance:

* The Computer User is always interested in reducing the time between the start and Completion of the program or event. (e) reducing the execution time.

* The execution time is also referred to as response time. Reduction in response time increases the throughput. (the total amount of work done in a given time).

* The performance of the Computer is directly related to throughput and hence it is reciprocal of execution time.

$$\text{Performance}_A = \frac{1}{\text{Execution time}_A}$$

This means that for two Computers A and B if the performance of A is greater than the Performance of B, we have.

$$\text{Performance}_A > \text{Performance}_B$$

$$\frac{1}{\text{Execution time}_A} > \frac{1}{\text{Execution time}_B}.$$

$$\text{Execution time}_B > \text{Execution time}_A$$

The execution time on B is longer than that on A if A is faster than B.

→ "A is n times as fast as B" to mean

$$\frac{\text{Performance}_A}{\text{Performance}_B} = n.$$

If A is n time faster than B then the execution time on B is n times longer than it is on A.

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

Measuring Performance:

CPU Time

* CPU execution time or simply CPU time is the time the CPU spends computing for particular task and does not include time spend waiting for I/O or running other programs.

* CPU time can be divided into the CPU time spent in the program called User CPU time and CPU time spent in the operating system performing tasks on behalf of the program, called System CPU time.

Performance Metrics:

* Users and designers often examine performance using different metrics.

CPU execution time for a Program =

CPU clock cycles for a Program x clock cycle time

alternatively, because clock rate and clock cycle time are inverse.

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

This formula makes it clear that the hardware designer can improve performance.

Problem:

If Computer A runs a Program in 10 seconds and Computer B runs the same Program in 25 sec. How much faster is A and B?

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

Thus the performance ratio is

$$\frac{25}{10} = 2.5$$

A is therfore 2.5 times faster than B.

Computer A runs a Program in 12 Seconds with a 3 GHz clock. We have to design a Computer B such that it can run the same Program with in 9 Seconds Determine the clock rate for Computer B. Assume that due to increase in clock rate CPU design of Computer B is effected and it requires 1.2 times as many clock cycles as Computer A for execution this Program.

Soln

Clock rate$_A$ = 3×10^9 cycle/sec.

Cpu time$_A$ = 12 Seconds.

Cpu time$_B$ = 9 Sec.

$$\text{Cpu time}_A = \frac{\text{CPU clock cycles}_A}{\text{clock rate}_A}$$

$$12 \text{ sec} = \frac{\text{CPU clock cycle}_A}{3 \times 10^9 \text{ cycles/sec}}$$

$$\text{CPU clock cycle}_A = 12 \text{ sec} \times 3 \times 10^9 \text{ cycle/sec}$$

$$= 36 \times 10^9 \text{ cycles}$$

Cpu time for Computer B is

$$\text{Cpu time}_B = \frac{\text{CPU clock cycles}_B}{\text{Clock rate}_B} = \frac{1.2 \times \text{CPU clock cycle}_A}{\text{clock rate}_B}$$

$$9 \text{ Seconds} = \frac{1.2 \times 36 \times 10^9 \text{ cycle}}{\text{clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 36 \times 10^9 \text{ cycle}}{9 \text{ sec}} = 4.8 \text{ cycle/sec}$$

$$= 4.8 \text{ GHz}.$$

Power Wall

* Power wall refers to the representational wall signifying the peak power constraint of a system.

* Running a Processor at high clock Speeds allows for better performance. However when Processor runs at a higher Speed, it generates more heat and consumes more power.

Thus when there is an increase clock rate, there is increase in power consumed.

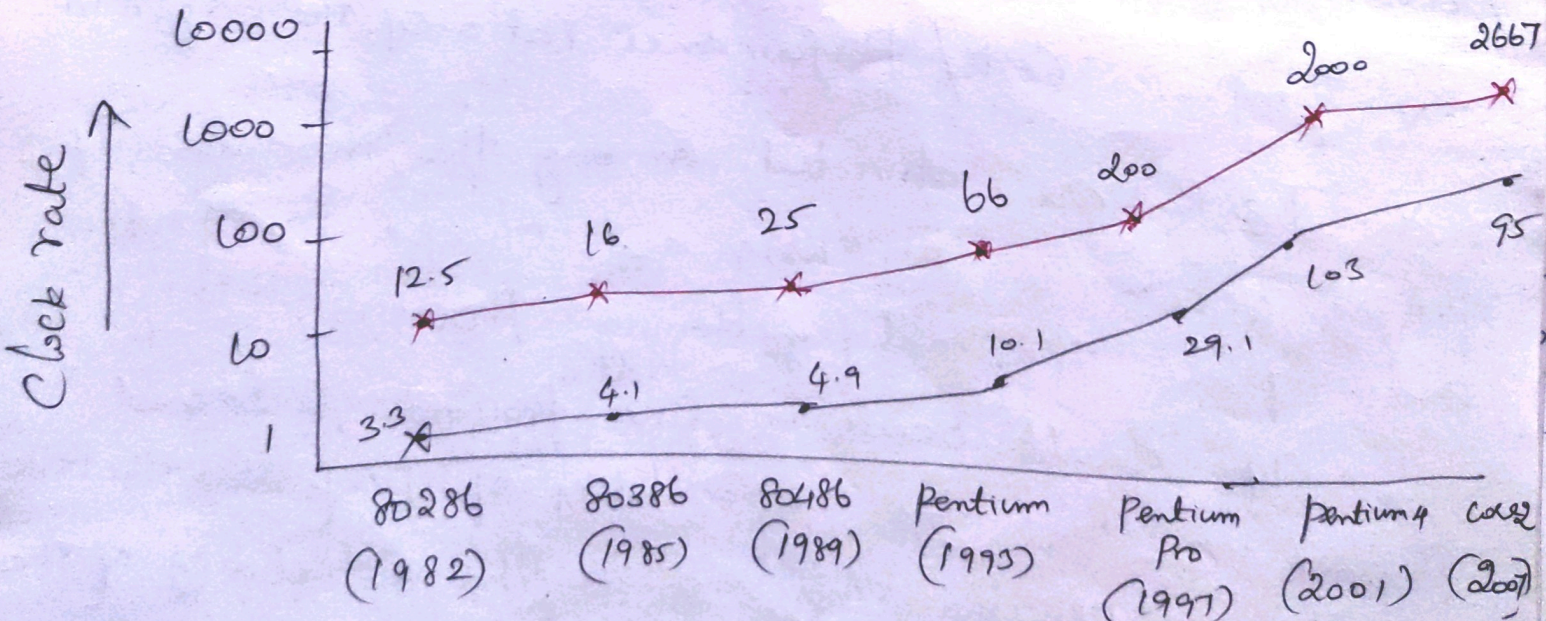
* fig shows the increase in clock rate and power of Intel microprocessors. The graph is flattened in the recent years because there is a practical limit for cooling microprocessors and hence increase in power dissipation.

* Power Consumed by a CPU is given by

Power = Capacitive load x Voltage^2 x frequency

$$P = CV^2f$$

The f is the function of clock rate. The C is function of both the number of transistors connected to an output and technology.



From Uniprocessor to Multiprocessors

→ The Performance of the Computers has drastically increased when the technology has drifted from Uniprocessor Systems to Multiprocessor System. The performance of the processors also increased significantly.

→ Uniprocessor System is a type of architecture that is based on a single Computing Unit. All the Operations were done sequentially on the same Unit

Multiprocessor Systems are based on executing instructions on multiple Computing Units.

→ The multiprocessor architecture is based on Flynn Taxonomy.

Advantages of Multiprocessor

* Improves Cost/Performance ratio of the System
* Tasks are divided among the modules. If failure occurs it is easier and cheaper to find and replace the malfunctioning Processor.
* If fault occurs in one Processor, a second Processor can take the responsibility of performing the task of failure processor. This improves the reliability of the Systems.

(1) Single Instruction, Single Data (SISD)

* This is a Uniprocessor machine. which is Capable of executing a Single instruction operating on a Single data Stream.
* Most Conventional Computers have SISD architecture.
* All the instructions and data to be Processed have to be Stored in Primary memory.

(2) Multiple Instruction, Single Data (MISD).

* An MISD Computing System is a Multiprocessor machine Capable of executing different instruction on different processing elements but all of them Operating on the same dataset.

(3) Single Instruction Multiple Data (SIMD).

* This machine Capable of executing the same instruction on all the CPU's but operating on different data Stream.

(4) Multiple Instruction, Multiple Data (MIMD)

* This is Capable of executing multiple instruction on multiple data sets.
* Each PE in the MIMD model has Separate instruction and data Streams.
* Unlike SIMD and MISD machine, PEs in MIMD machine work asynchronously.

Instruction Types (or) Operation:

A Computer has a Set of instructions that allows the User to formulate any data Processing task.
It can be Categorized into following instruction types:

* Data transfer
* Arithmetic
* logical
* Program Control
* Input/output
* String Manipulation
* Translate Instruction
* Processor Control Instruction.

① Data Transfer Instruction.

Which copy information from one location to another either in the processor's internal register set or in external Main Memory.

Example:

MOVE → Transfer data from the Source location to destination location.

LOAD → Transfer data from a Memory location to CPU register.

STORE → Transfer data from a CPU register to Memory location.

PUSH → Transfer data from the Source to Stack

XCHG → Exchange; Swap the Contents of the Source and destination.

② Arithmetic: which perform Operations on numerical data.

Example:

ADD → Calculate Sum of two operands.

ADC → Add with Carry. Calculate Sum of two Operands and the carry bit.

SUB → Subtract; Calculate difference of two numbers.

MUL → Multiply; Calculate Product of Two Operands.

DIV → Divide.

NEG → Negate. Change Sign of Operand.

INC → Increment ; add 1 to operand.

DEC → Decrement ; Subtract 1 from operand.

③ Logical operation which include Boolean and other non-numerical operations.

Example: NOT → Complement the operand

OR → perform bit wise logical OR of Operand.

AND → perform bit wise logical AND of operand

ROT → Rotate, Shift Operand (Left or Right)

TEST → Test for specified condition. and set flags.

④ Control Transfer Instruction: which change the sequence in which Programs are executed.

(eg). JUMP → Branch ; Enter the Specified address into PC ; Unconditional transfer.

JUMP IF → Branch on Condition, Enter the Specified address into PC only if the Specified Condition is Satisfied; Conditional transfer.

INT → Interrupt ; Creates a Software interrupt.

⑤ Input-Output Instruction. which Cause information to be transferred between the processor or its main memory and external I/O devices

(eg.) IN → Input ; Transfer data from Specified I/O port or device to destination.

OUT → Output ; Transfer the data from Specified Source to I/O port or device.

(6) String Manipulation Instruction.

These manipulate String of byte, word, double etc

eg. MOVS → Move byte or word of string
LODS → Load byte or word of string
CMPS → Compare byte or word of string.

(7) Translate Instruction.

These Convert data from one format to the another.

eg. XLAT → Translate; Convert the given code into another by table lookup.

PACK → Convert the unpacked decimal number into packed decimal.

(8) Processor Control Instruction:

These Control Processor Operations.

eg. HLT → Halt; stop the Program execution.
NO-OP → No operating nothing
CMC → Complement 'Carry' flag.
CLC → clear 'carry' flag.
WAIT → Stops Program execution, resume execution when Condition is Satisfied.

Addressing Modes.

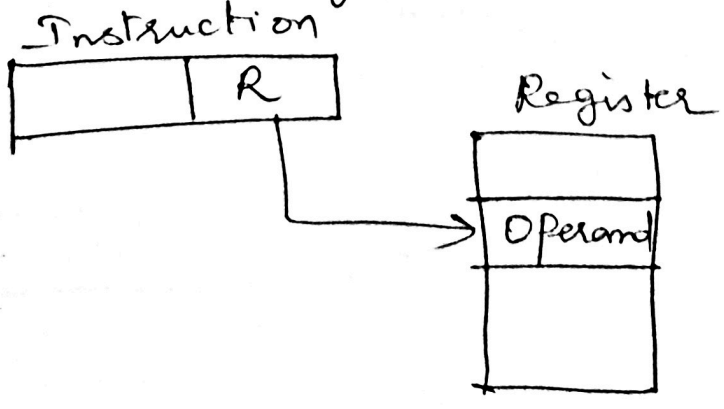
The different ways in which the location of an Operand is specified in an instruction are referred to as addressing modes.

Types of addressing modes are.

1. Register addressing mode.
2. Absolute (or) Direct addressing mode
3. Immediate addressing mode
4. Indirect addressing mode.
5. Register Indirect addressing mode.
6. Displacement addressing mode.
7. Relative addressing mode.
8. Base register addressing mode.
9. Index addressing mode.

(1) Register addressing mode: The Operand is the contents of processor register.

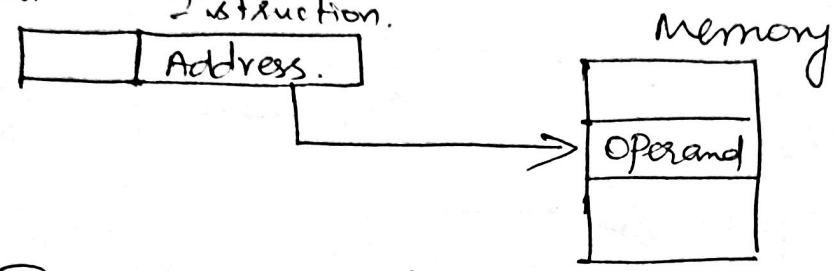
Example: Mov R_2, R_1; This instruction copies the contents of register R_2 to register R_1.



2). Absolute or Direct addressing mode:

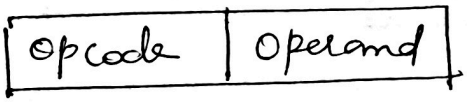
The address of the location of the operand is given explicitly as a part of the Instruction.

Example Mov 2000, This instruction copies the contents of memory location 2000 into the A register.



③ Immediate addressing mode:

In this mode of addressing the operand is a part of the Instruction and is specified in the address field.



Mov #20 A; This instruction is given explicitly in Copies Operand 20 in the register A.

Sign # in front of the value of an Operand is used to indicate that this value is an immediate operand.

④ Indirect addressing mode:

In this addressing mode, the instruction contains the address of memory which refers the address of the operand. It is Indirectly storing the addres of the target location in another memory location.

(5) Register Indirect addressing mode:

The effective address of the operand is the contents of a register or the main memory location whose address is given explicitly in the instruction.

MOV (R0), A; The instruction copies the contents of memory addressed by the contents of register R0 into the register A.

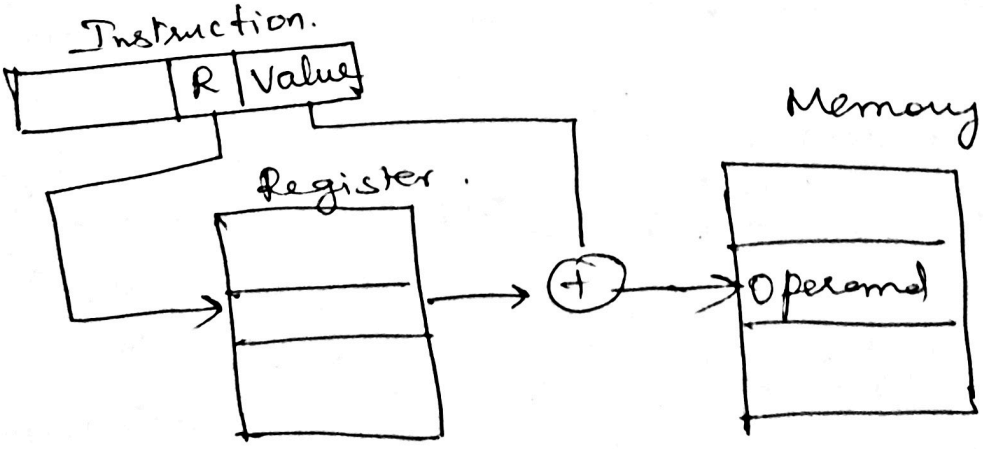
(6) Displacement addressing mode:

→ This addressing mode combines the capabilities of indirect addressing and register indirect addressing

→ In this addressing mode, instruction has two address field: value and referenced register

→ The effective address is compulied by adding contents of referenced register to value

EA = value + (R).



Relative addressing

This addressing mode is Commonly Used to specify the target address in branch instruction for example.

JNZ BACK.

This Instruction Causes Program execution to go to the branch target location identified by the name BACK, if the branch Condition is satisfied.

(8) Base Register addressing

* In this addressing mode, the referenced register Contains the main memory address and address field Contains the displacement.
* Displacement is Usually integer number.

EA = (R) + Displacement.

MOV [R+8], A : This Instruction Copies the Contents of memory whose address is determined by adding the Contents of register R and displacement 8 to the register A.

(9) Index addressing mode:

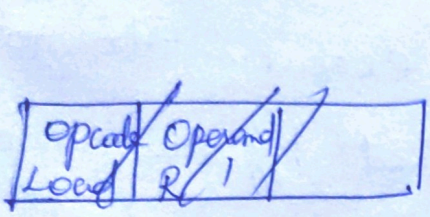
The effective address of the Operand is generated by adding a Constant Value to the Contents of a register.

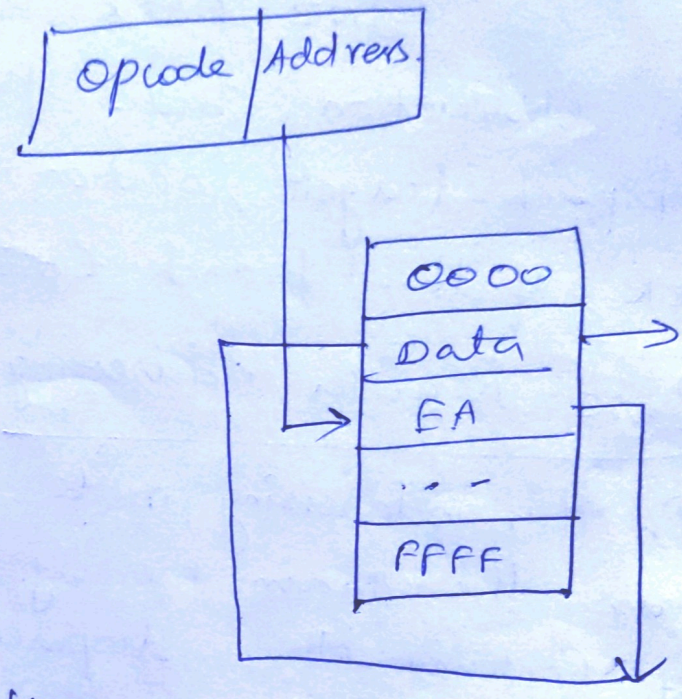
Move 20(R1), R2

The above Instruction loads the Contents of register R2 into the memory location whose address is

Calculated by addition of the Contents of register R1 And Constant Value (offset or displacement) 20.

(4) Indirect addressing





Logical data:

→ most of the Processors interpret data as a bit, byte, word or double word. These are referred to as units of data.

→ When data is viewed as n-1 bit items of data, each item having the value 0 or 1, it is considered as a logical data.

Logical operation. → logical operators are useful for extracting and inserting groups of bits in a word and it needs to operate on fields of bits within a word or even on individual bits

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

Bitwise Operations [AND]

Bitwise AND Operations leaves a 1 in the result if both bits of the operand are 1 otherwise 0.

MIPS Instruction

Eg: and $s0, $t1, $s3 ; $s0 = $t1 & $s3.

0000 0000 0000 0000 0000 1101 0000 0000 ⇒ $t1

& 0000 0000 0000 0000 0011 1100 0000 0000 ⇒ $s3

0000 0000 0000 0000 0000 1100 0000 0000 ⇒ $s0.

Bitwise operations -OR

Bitwise OR operation places a 1 in the result if either operand bit is a 1.

MIPS Instructions has or & ori

Eg: or $s0, $s1, $s3 # $s0 = $s1 | $s3.

00000000 00000000 0000 1101 00000000 ⇐ $s1

00000000 00000000 00111100 00000000 ⇐ $s3.

00000000 00000000 00111101 00000000 ⇒ $s0.

NOR: A logical bit by bit operation with two Operands that calculate the NOT of the OR of the two operands. That is calculate a 1 only if there is a 0 in both operands.

MIPS instruction set includes NOR (NOT OR) Instruction Instead of NOT Instruction.

nor $t0 $t1, $t2.

$t1 : 0000 0000 0000 0000 0011 1100 0000 0000

$t2 : 0000 0000 0000 0000 0000 0000 0000 0000

$t0 : 1111 1111 1111 1111 1100 0011 1111 1111

nor $t0 $t1 $t2 ≡ # reg $t0 = ~(reg $t1 | reg $t2)

Shift Operation.

→ Another class of operations is called shifts. They move all the bits in a word to the left or right, filling the emptied bits with 0's

→ The actual name of two MIPS shift operations is called Shift left Logical (Sll) and Shift Right logical (Srl).

Example If register $s0 Contained 9_{ten} and the instruction shift left by 4. the new value would be 144.

left Shift by 4.

0000 0000 0000 0000 0000 0000 1001 = 9_{ten}

0000 0000 0000 0000 0000 1001 0000 = 144_{ten}.

Control Operations.

→ Decision making instructions alter the Control flow ie) change the "next" instruction to be executed, like if statement with a go to statement

→ MIPS assembly language includes two decision making instructions like if else and go to called Conditional branch instruction.

Decision Making Instruction.

* The first decision making instruction is

beq register1, register2, L1;

branch if equal statement state that if value of register 1 is equal to the value of register 2, then Processor can changes its sequence to L1. Otherwise it execute the next instruction in sequence.

* Second decision making instruction is

bne register1, register2, L1;

branch if not equal statement - state that if value of register 1 is not equal to the value of register 2. then Processor can change its sequence to L1 otherwise it execute the next instruction in sequence.

* Another kind of branch instruction is Unconditional branch instruction. This instruction says that the Processor always follows the branch without testing for any condition.

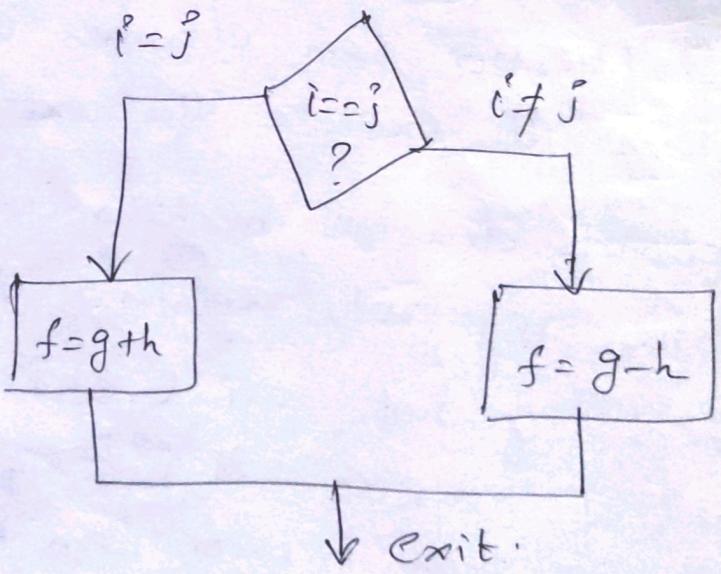
"J" target address → Jump - Instruction allow the Processor to target address.

Compiling if then - else into Conditional Branches.

Example: Compiling 'C' if into MIPS.

In the following code segment f, g, h, i and j are variables. If the five variables f through j Corresponds to the five register $S0 through $S4

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



MIPS Code: first step is Compare for equality (i==j) if equal to True.

```
beq $S3, $S4, True;     # branch i==j
Sub $S0, $S1, $S2;      # f = g-h (false)
j Exit                  # go to exit
```

True : add $s0, $s1, $s2 ; # f = g+h (true)
Exit

Write a program to calculate the arithmetic statement: $Y = (A+B)^*(C+D)$ Using three address, two address, One address and zero address.

soln: Using three address Instructions.

```
ADD  R1, A, B ;   R1 ← M[A] + M[B].
ADD  R2 C, D ;    R2 ← M[C] + M[D].
MUL  Y R1, R2 ;   M[Y] ← R1 * R2.
```

Using two address Instruction.

```
MOV  R1, A ;   R1 ← M[A].
ADD  R1, B ;   R1 ← R1 + M[B].
MOV  R2, C ;   R2 ← M[C].
ADD  R2, D ;   R2 ← R2 + M[D]
MUL  R1, R2 ;  R1 ← R1 * R2
MOV  Y, R1 ;   M[Y] ← R1
```

Using One address Instruction.

```
LOAD  A ;   Ac → M[A].
ADD   B ;   Ac → Ac + M[B].
STORE T ;   M[T] ← Ac
LOAD  C ;   Ac ← M[C].
ADD   D ;   Ac ← Ac + M[D]
MUL   T ;   Ac ← Ac * M[T].
```

STORE Y ; M[Y] ← AC.

Using Zero address Instructions.

```
PUSH  A ;  TOS ← A
PUSH  B ;  TOS ← B
ADD     ;  TOS ← (A+B). - Zero address Instruction
PUSH  C ;  TOS ← C.
PUSH  D ;  TOS ← D.
ADD     ;  TOS ← (C+D) - Zero address Instruction
MUL     ;  TOS ← (C+D) * (A+B)
POP   Y ;  M[Y] ← TOS.
```

In the following Code segment a, b, c, d and e are Variable. If the five variable f through j Correspond to the five registers $S0 through $S4, what is the Compiled MIPS code for Ccode.

Soln

```
bne $S0, $S1, Else.   // go to else if a≠b
add $S2, $S3 $S4      //c = d + e (skipped if a≠b)
j exit                // go to exit
Sub $S2, $S3, $S4     // c = d - e (skipped if a=b)
```

UNIT-II
ARITHMETIC

Introduction.

* The Arithmetic and Logic Unit (ALU) is responsible for performing arithmetic operations such as add, Subtract, division and multiplication and logical operations, such as AND, OR Inverting etc

* Based on number System two basic data types are implemented in the Computer System.

1. fixed point numbers. 2. floating point numbers.

Binary Addition.

When adding two numbers if the Sum of the digits in a given position equals or exceed, then a Carry is propagated.

eg: $5_{10} + 6_{10}$

```
0000 0000 0000 0000 0000 0000 0000 0101
0000 0000 0000 0000 0000 0000 0000 0110
-----------------------------------------
0000 0000 0000 0000 0000 0000 0000 1011
```

$\Rightarrow (11)_{10}$

Binary Subtraction.

Subtraction operation Cannot be Performed directly.

① One can formulate a subtraction algorithm.

② One can negate the subtrahend (ie in a-b the subtrahend is b).

then perform addition (2's Complement).

$$\begin{array}{r} 10011 \\ 101 \\ \hline 01110 \end{array} \quad (-)$$

Ans = $(1110)_2$

0 − 0 = 0
1 − 0 = 1
1 − 1 = 0
10 − 1 = 1

$12_{10} - 5_{10} = 7_{10}$ (Direct Method)

$$\begin{array}{r} 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1100 \\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101 \\ \hline 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0111 \end{array} = (7_{10})$$

Unsigned binary representation

Addition with 2's Complement

5 = 0101
−5 = 1010 + 1 = 1011

$$\begin{array}{r} 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1100 \quad (12) \\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011 \quad (-5) \\ \hline \boxed{1}\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0111 \end{array}$$

↓ Discard the Carry

Fixed Point addition and Subtraction.

Two's Complement addition and Subtraction.

Subtraction can be done in following ways.

① Direct method. ② Using 1's Complement

③ 2's Complement

Perform Add the numbers $(0.5)_{10}$ and $(0.4375)_{10}$ using floating point addition

$(0.5)_{10} = (0.1)_2 = 0.5 \times 2 = 1.0 \Rightarrow (0.1)_2$

Normalizing the above value

$= \boxed{1.0 \times 2^{-1}}$

$0.4375 \Rightarrow$

$0.4375 \times 2 = 0.8750$

$0.8750 \times 2 = 1.7500$

$0.7500 \times 2 = 1.5000$

$1.5000 \times 2 = 1.0000$

$\Rightarrow (0.0111)_2$

Normalizing the above Value $= 1.110 \times 10^{-2}$

Step 1 : The significant of the number with lesser exponent (1.11×10^{-2}) is shifted right until its exponent matches the larger number

$1.11 \times 10^{-2} = \boxed{0.111 \times 2^{-1}}$

Step 2 : Add the significand

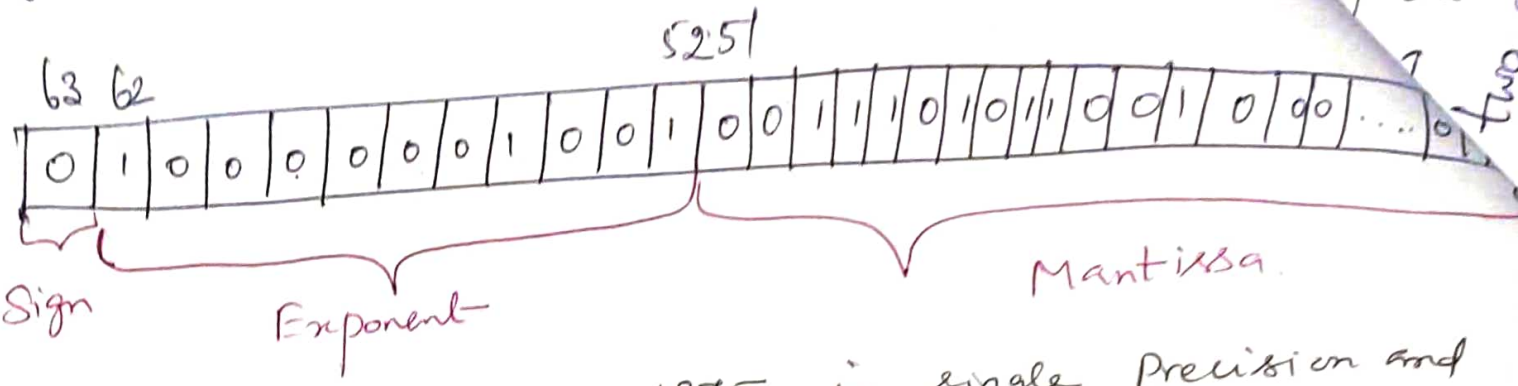
$$\begin{array}{r} 1.0 \times 2^{-1} \\ 0.111 \times 2^{-1} \\ \hline 1.111 \times 2^{-1} \end{array}$$

$$\boxed{\text{Ans} \quad 1.111 \times 2^{-1}}$$

eg. ② Add the numbers 0.5_{10} and -0.4375_{10}

Ans $= 1.000 \times 2^{-4} = 0.0625$

Number in double precision format-



Eg.2 : represent -307.1875_{10} in single Precision and double Precision formats.

Floating Point Arithmetic

Addition and Subtraction.

Consider two floating Point numbers.

$A = m_1 r^{e_1}$ and

$B = m_2 r^{e_2}$ Assume $e_1 \geq e_2$

Procedure:

Step1 : Select the number with a smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents $|e_2 - e_1|$.
for example if numbers are 1.75×10^2 and 6.8×10^4.
then the number 1.75×10^2 is selected and converted to 0.0175×10^4.

Step2 : Set the exponent of the result equal to larger exponent.

Step3 : Perform addition/Subtraction on mantissa and determine the sign of the result.

Step4 : Normalize the result if necessary.

Point Number Representation. (2)

→ Fixed Point / integer numbers are represented in two forms: Signed integer and Unsigned Integer

→ To represent negative numbers various techniques are used because computer does not have provision to represent negative sign. Techniques to represent signed integer numbers are.

1. Sign magnitude representation
2. 1's Complement
3. 2's Complement.

Signed Magnitude Representation.

→ left most bit (sign bit) in the binary number represents sign of the number.

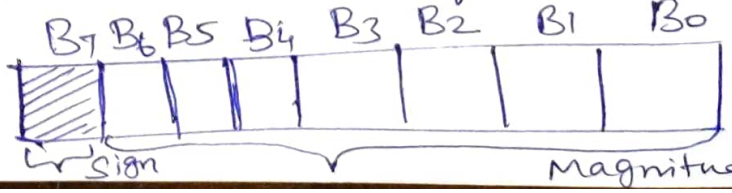
→ Here the most significant bit (MSB) represents sign of the number. If MSB is 1 number is negative and if MSB is 0 number is positive. The remaining bits represent magnitude of the number.

→ examples. $+6 = 0000\ 0110$

$-14 = 1000\ 1110$

$-64 = 1100\ 0000$

→ In Case of Unsigned 8 bit binary numbers the decimal range is 0 to 255. for signed magnitude 8-bit binary numbers the largest magnitude is reduced from 255 to 127



1's Complement Representation

The 1's Complement of a binary number is the number that results when we change all 1's to Zeros and Zeros to ones.

2's Complement Representation.

→ The 2's Complement is the binary number that result when we add +1 to the 1's Complement. It is given as

2's Complement = 1's Complement + 1.

2's Complement form is used to represent negative numbers.

Fixed point Addition and Subtraction

→ We can relate addition and Subtraction operations of numbers by the following relationship.

$(\pm A) - (+B) = (\pm A) + (-B)$

$(\pm A) - (-B) = (\pm A) + (+B)$.

Therefore we can change Subtraction operation to an addition operation by changing the sign of the Subtrahend.

Addition

Rules for addition.

$0+0 = 0$

$0+1 = 1$

$1+0 = 1$

$1+1 = 10_2$

Example Add $(6)_{10}$ to $(7)_{10}$ in binary

	1	1		
	0	1	1	0 $\quad 6_{10}$
	0	1	1	1 $\quad 7_{10}$
1	1	0	1	13_{10}

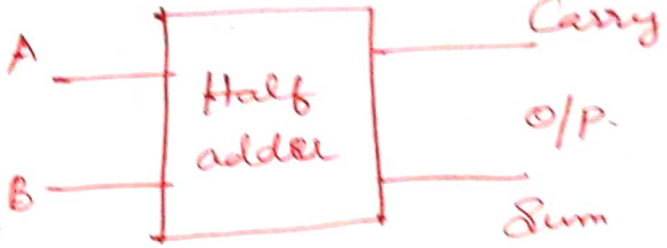
→ The first three operations produce a sum whose length is one digit but when the last operation is performed sum is two digits. The higher significant bit of this result is called a carry and lower significant bit is called sum.

→ The logic circuit which performs this operation is called a half-adder. The circuit which performs addition of three bits

Half adder

→ The half adder operation needs two binary input : augend and addend bits and two binary output : sum and carry.

Input		Output	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Block schematic of half adder

limitations of half adder.

→ In multidigit addition we have to add two bits along with the carry of previous digit addition.

→ Effectively such addition requires addition of three bits. This is not possible with half adder hence half-adders are not used in practice.

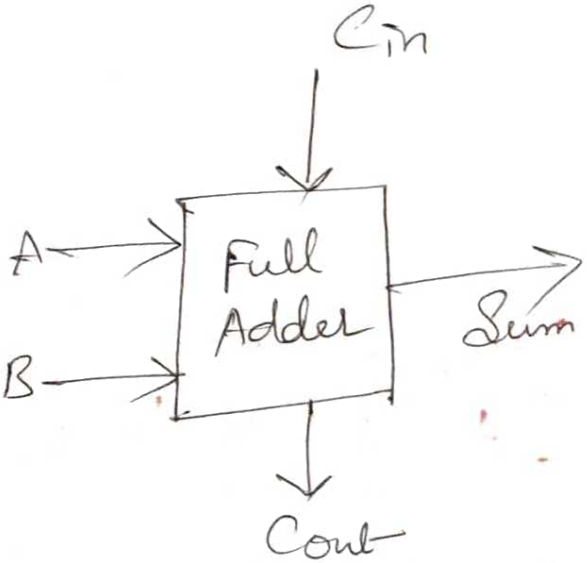
full adder

→ A full-adder is a Combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs.

→ Two of the input Variable, denoted by A and B, represent the two significant bits to be added. The third input Cin represent the carry from the previous lower significant position.

Inputs			Outputs	
A	B	Cin	Carry	Sum.
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth table for full adder.



Block schematic of full adder

Binary Subtraction.

→ The Subtraction Consists of four possible elementary operation

→ In all operations, each Subtrahend bit is Subtracted from the minuend bit. In Case of second operation the minuend bit is smaller than the Subtrahend bit hence 1 is borrowed.

Subtraction Method

① Subtract bits column-wise starting from LSB with borrow if any
② Put the difference at the bottom of same column
③ Take the borrow, if required from the next column

Binary Subtraction Using 1's Complement Method

→ In a 1's Complement Subtraction, negative number is represented in the 1's Complement form and actual addition is performed to get the desired result. for example Operation A−B is Performed Using following steps.

① Take 1's Complement of B.
② Result ← A+1's Complement of B.
③ If Carry is generated then the result is Positive and in the true form. Add Carry to the result to get the final result
④ If Carry is not generated then the result is negative and in the 1's Complement form.

eg. Perform $(15)_{10} - (28)_{10}$. Using 6 bit 1's Complement representation.

$(15)_{10} = (001111)_2$ $\quad (28)_{10} = 011100$

1's Complement of $(15)_{10} = 110000$

```
  1 1  ------> Carry
  0 1 1 1 0 0   (28)
  1 1 0 0 0 0   (15)
-------------------
  1 0 0 1 1 0 0   Result
-------------------
      |_____> 1
-------------------
  0 0 1 1 0 1   final result (13)10.
-------------------
```

$28 = 011100$.

1's Complement of 28 = 100011

binary equ. of $(15)_{10}$ = 001111

1's Comp. of $(28)_{10}$ = 100011

$$\begin{array}{r} (15)_{10} \\ (-28)_{10} \\ \hline (-13)_{10} \end{array}$$

Result = 110010

Verification. 001101 $= (13)_{10}$

Binary Equivalent of $(-13)_{10}$

eg. Perform $(-4)_{10} - (8)_{10}$. Using 1's Complement form

$(-4)_{10} - (-8)_{10} = (-4)_{10} + (8)_{10}$

$(4)_{10} = 0100$

$(-4)_{10} = 1011$ 1's Complement of 4.

```
        1
Sign  [1] 1 0 1 1
extension  0 1 0 0 0
          ---------
          [1] 0 0 0 1 1   =>  00100  Result
```

(5) Subtraction Using 2's Complement Method

→ In a 2's Complement subtraction negative number is represented in the 2's Complement form and actual addition is performed to get the desired result.

for example, operation A−B. is performed Using following Steps.

(1) Take 2's Complement of B.
(2) Result ← A + 2's Complement of B.
(3) If Carry is generated then result is positive and in the true form In this case Carry is ignored.
(4) If Carry is not generated then the result is negative

Eg. Perform $(28)_{10} - (15)_{10}$ Using 6-bit 2's Complement representation.

$(15)_{10}$ = 0 0 1 1 1 1

(1's) 1's Complement of $(15)_{10}$ = 1 1 0 0 0 0

1 1 0 0 0 1

0 1 1 1 0 0 = $(28)_{10}$

1 1 0 0 0 1 = 2's Comp of 15

0 0 1 1 0 1 ⇒ Result: Binary equivalent of $(13)_{10}$.

Pbm

Subtract $(11010)_2 - (10000)_2$ Using 1's Complement and 2's Complement method.

10000

01111 → 1's Complement

1

10000 → 2's Complement

Using 1's Complement method

```
  1 1 1
  1 1 0 1 0
  0 1 1 1 1
-------------
1 0 1 0 0 1
  ↓       1
-------------
  0 1 0 1 0
```

Result is +ve.

Using 2's Complement Method

```
  1 1 0 1 0
  1 0 0 0 0
-------------
X 0 1 0 1 0
-------------
```

Result is true.

↓ ignore Carry.

Fixed Point Multiplication (6)

→ Multiplication of fixed point numbers can be accomplished with addition, subtraction and shift operations.

Unsigned Multiplication

→ Multiplication of unsigned binary integers is handled similarly to the way it is carried out by hand for decimal numbers.

→ In the binary system the partial products are easily defined when the multiplier bit is 0, the partial product is 0 and when the multiplier is 1 the partial product is the multiplicand.

→ The final product is produced by summing the partial products. Before summing, each successive partial product is shifted one position to the left relative to the preceding partial product.

Multiplicand → 1 1 0 1 (13)
Multiplier 1 0 0 1 (9)

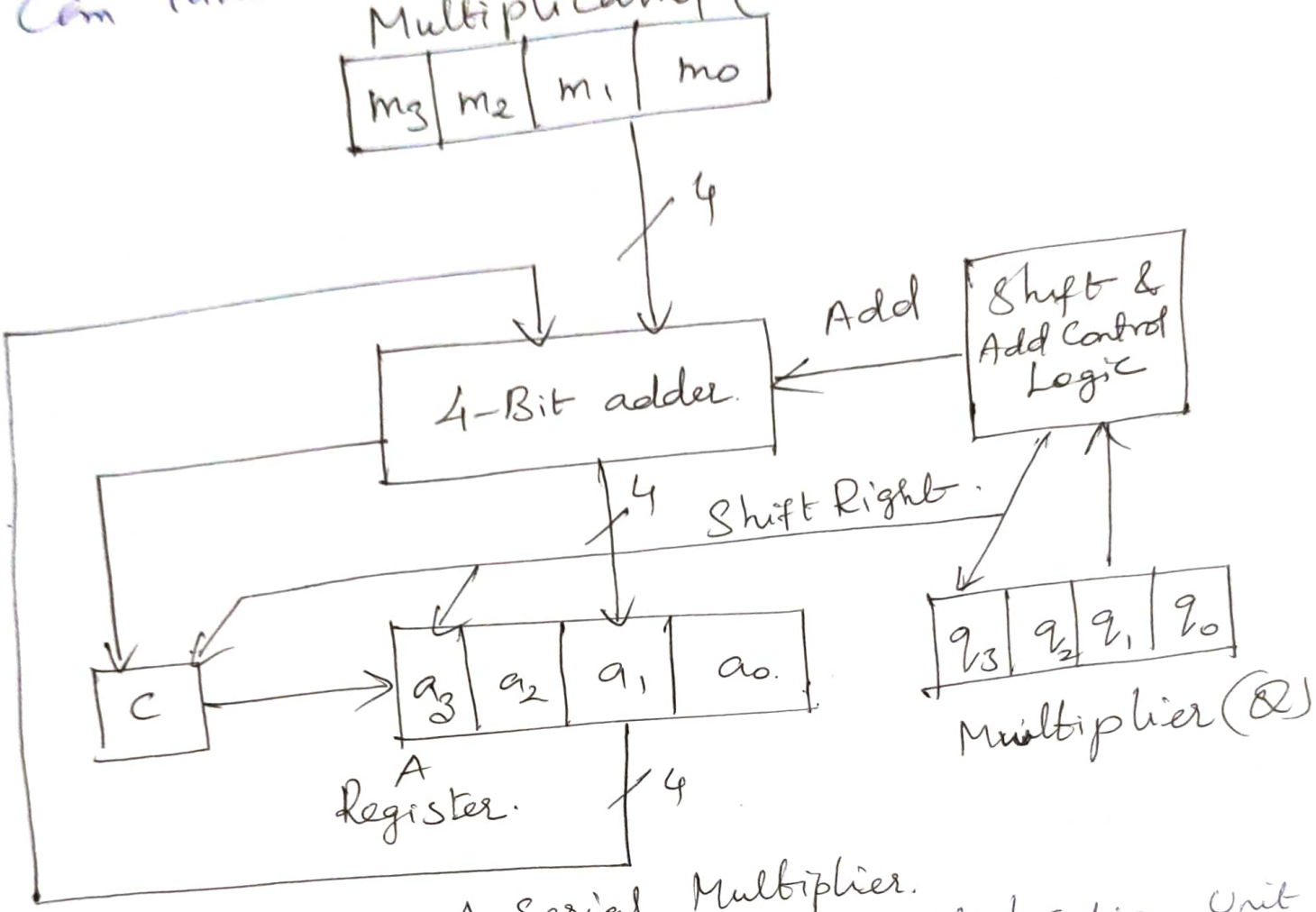
		1	1	0	1		partial products
	0	0	0	0			
0	0	0	0				
1	1	0	1				
1	1	1	0	1	0	1	= 117

→ Take two unsigned n bit numbers are multiplied the result can be large as 2n bits.

→ For example two four bit operands result in an eight bit product.

Hardware Implementation of integer Mul

The hardware implementation of integer mu
Can take a similar form to the manual me



A Serial Multiplier.

→ It shows a layout of a multiplication Unit for four bit numbers in which there is a four bit adder, a Control Unit, three four bit registers and one bit Carry register.

Steps: ① In order to multiply two numbers, the multiplicand is placed in the M-register, the multiplier is placed in the Q-register and the A and C registers are cleared to Zero.

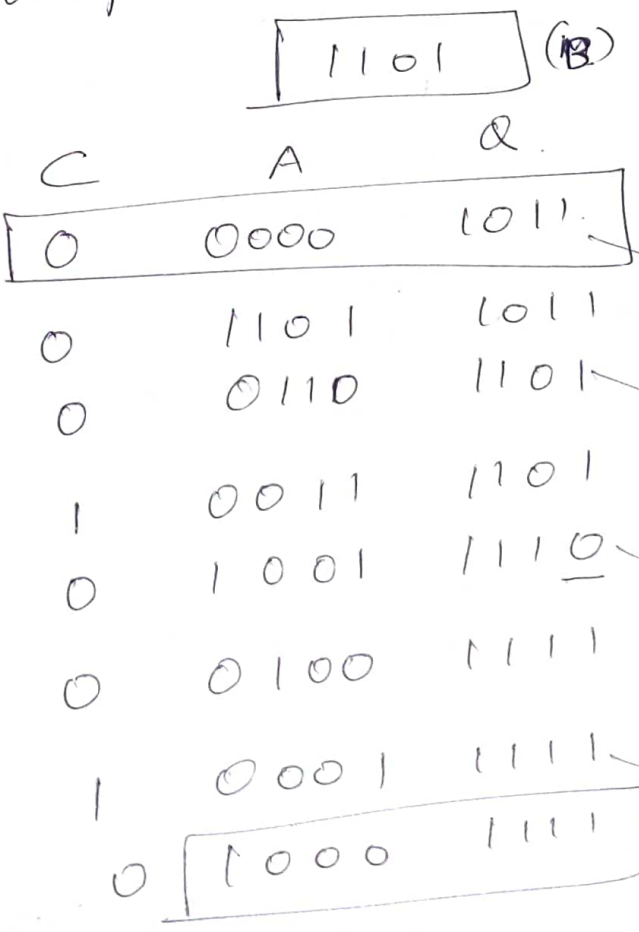
(2) During multiplication the right most bit of the multiplier determines whether the multiplicand is added into the Product at each step.

(3) After the multiplicand is added into the Product the multiplier and A registers are Simultaneously shifted to the right. This has the effect of shifting the multiplicand to the left and exposing the next bit of the multiplier in position q_0.

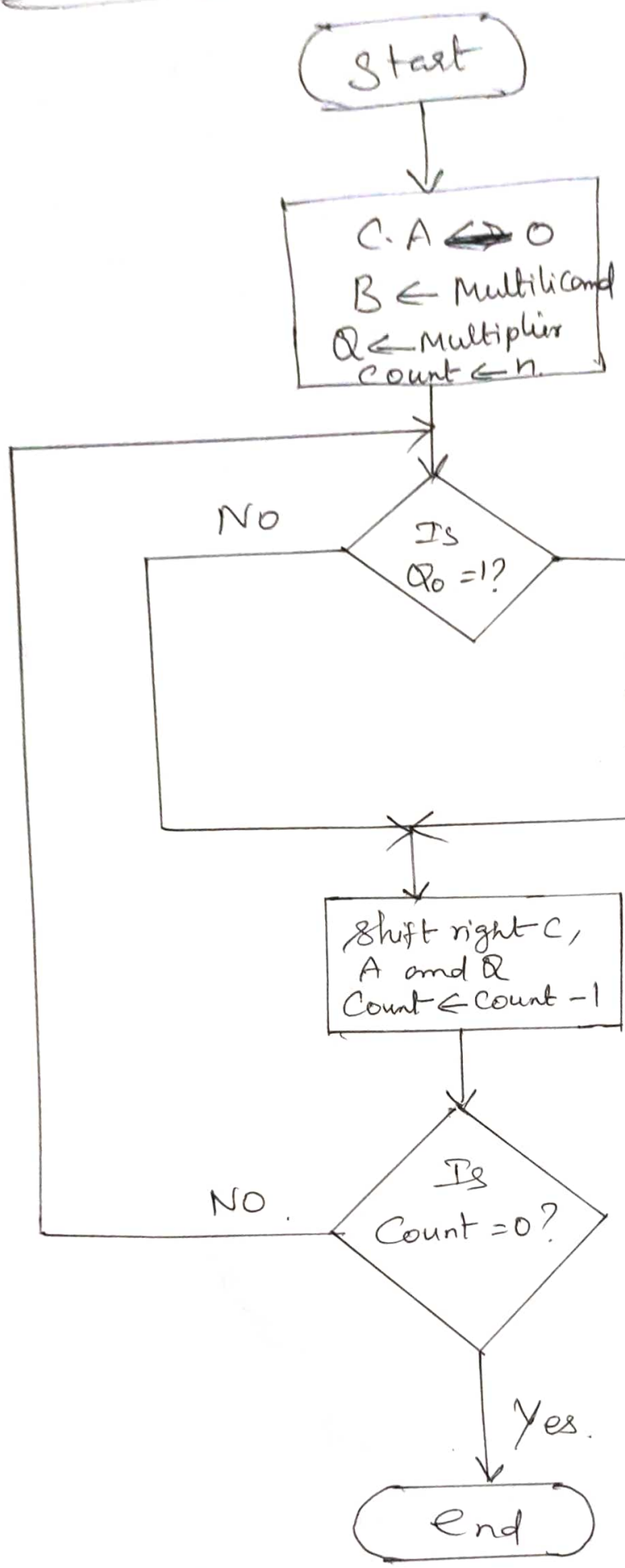
Example Multiplicand (M)

Consider 4-bit multiplier and Multiplicand

Multiplicand = 1101 (B) and Multiplier = 1011 (Q)



Multiplication Operation Steps



→ Bit 0 of multiplier operand (Q0 of Q reg) is checked

→ If bit 0 (Q_0) is One then multiplicand and Partial Product are added and all bit of C, A and Q register are shifted to right one bit. so that C bit goes into A_{n-1}

→ A_0 goes into Q_{n-1} and Q_0 is lost. If bit 0 (Q_0) is 0 then no addition is performed. Only Shift Operation is Carried Out.

→ Step 1 and 2 are repeated n times to get the desired result in the A and Q reg.

flow chart for multiplication Operation

Sequential Version of the Multiplication Algorithm and Hardware

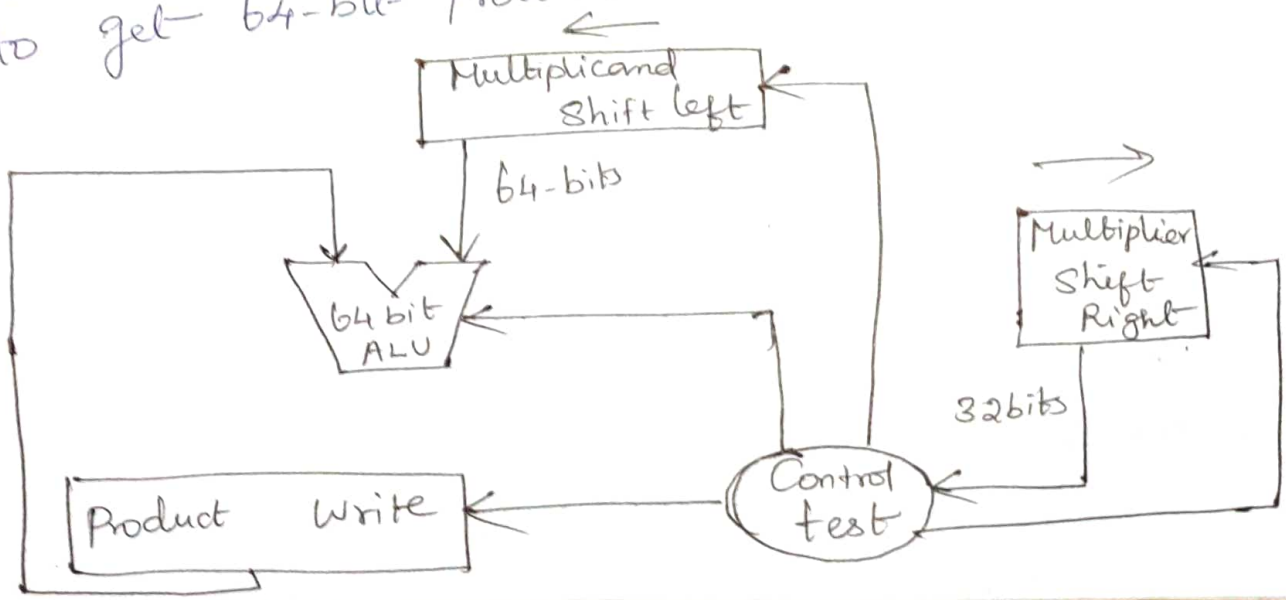
Step 1: The LSB bit of the Multiplier determines whether the multiplicand is added to the Product register.

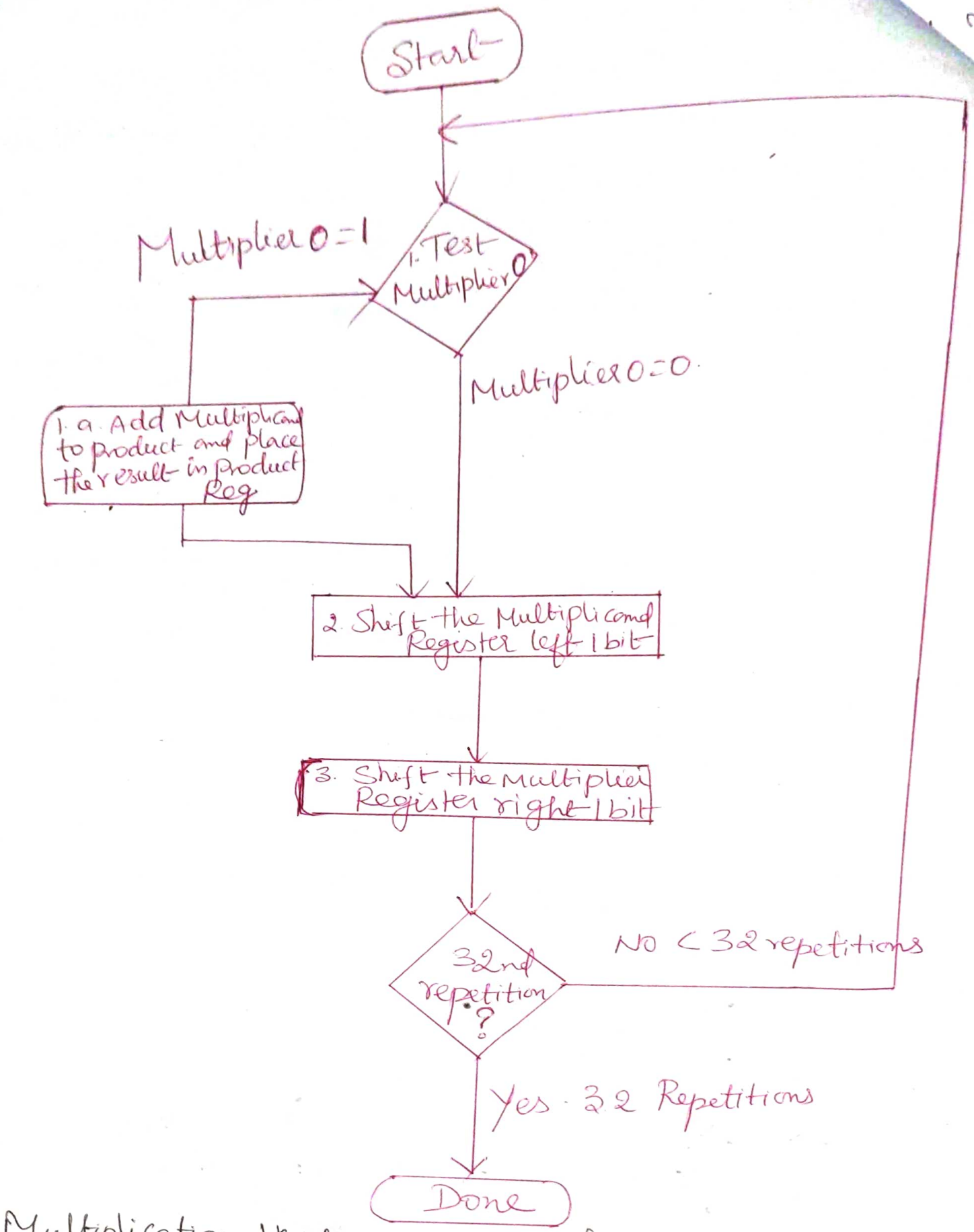
Step 2: Shift Multiplicand by 1 bit left logically that will have a effect of moving the intermediate operands to the left.

Step 3: Shift the Multiplier right logically by 1 bit that will give the next bit of the multiplier to be examined.

Multiplication Hardware

* It is Constructed with 64 bit multiplicand, ALU and Product register and one 32 bit multiplier.
* The 32 bit multiplier is loaded in 32 bit multiplier register, 32 bit multiplicand is loaded in right half of 64 bit multiplicand register and zero in the left half and the 64 bit Product register is initialized to 0.
* The above said procedure is executed for 32 times to get 64-bit Product result.





Multiplication ~~Hardware~~ flow chart.

e For Simplicity assume Use 4-bit Multiply
$4_{10} \times 3_{10} = 12_{10}$ or $0100 \times 0011_2$ Same Concept Can be Extended to 32 bit Size number.

Iteration	Steps.	Multiplier	Multiplicand	Product
0	Initial values	0011	0000 0100	0000 0000
1	1.a.1 → Prod = Prod + MCand	0011	0000 0100	0000 0100
	2. Shif left Mcand	0011	0000 1000	0000 0100
	3. Shift right Multiplier	0001	0000 1000	0000 0100
2.	1.a.1 → Prod = Prod + MCand	0001	0000 1000	0000 1100
	2. Shift left Mcand	0001	0001 0000	0000 1100
	3. Shift Right multiplier	0000	0001 0000	0000 1100
3	1.0 → No-operation	0000	0001 0000	0000 1100
	2. Shift left MCand	0000	0010 0000	0000 1100
	3. Shift Right Multiplier	0000	0010 0000	0000 1100
4.	1.0 → No operation	0000	0010 0000	0000 1100
	2. → Shift left multiplicand	0000	0100 0000	0000 1100
	3. Shift right Multiplier	0000	0100 0000	0000 1100

...ned Division.

→ Division is a reciprocal operation of multiplication. It is similar to the decimal numbers.

→ In Division Process first the bits of dividend are examined from left to right, until the set of bits examined represent a number greater than or equal to divison

→ until this Condition occurs 0's placed in the quotient from left to right.

→ When the Condition is satisfied a 1 is placed in the quotient and divisor is subtracted from the partial dividend. The result is referred to as a partial remainder.

→ from this point onwards the division process is repeated. In each repetition cycle additional bits from the dividend are brought down to the partial remainder until the result is greater than or equal to the divisor and the divisor is subtracted from the result to produce a new partial remainder.

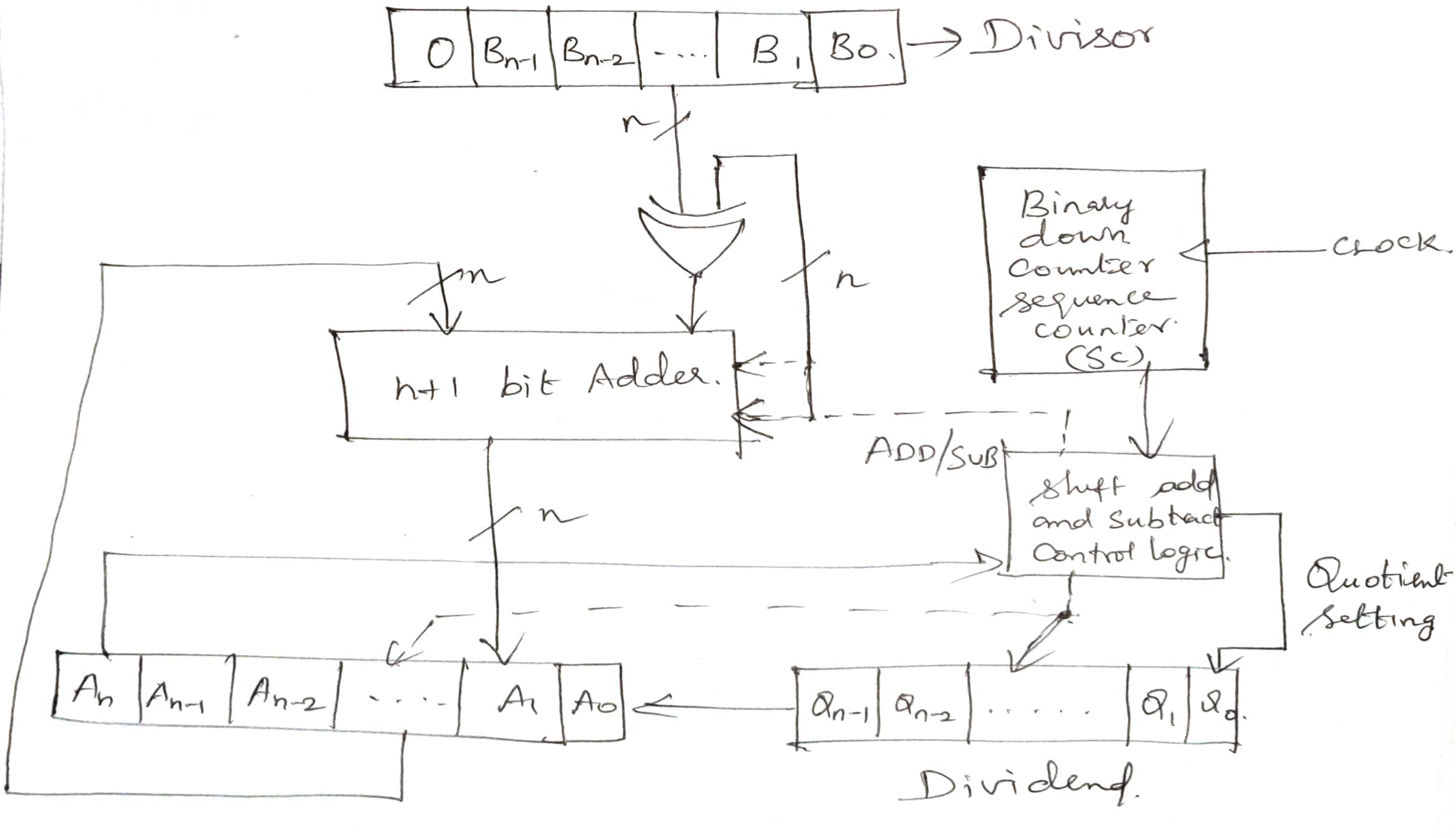
→ The Process Continues Until all the bits of the dividend are brought down and result is still less than the divisor.

110
~~1000~~) ~~110 0 1 0 1 0~~

```
           100100   ← quotient
      ___________
110 ) 11011011      ← Dividend
      110
      ________
      000110
       -110
      ________
        00011       ← Remainder
```

Division Example.

Division Algorithm. (Hardware to implement binary Division)



...on Operation Steps. (9) b

1. Shift A and Q left one binary position
2. Subtract divisor (ie add 2's Complement of divisor(B)) from A and place answer back in A (A ← A−B)
3. If the sign bit of A is 1 Set Q_0 to 0 and add divisor back to A (that is restore A) otherwise Set Q_0 to 1
4. Repeat steps 1, 2, and 3 n times

Example of Division Process Using the Serial ~~adder~~.

Divider.

Divisor (M): [00011] → Initial value.

A	Q	
0000	0111	
00000	1110	→ shift left
11101	1110	→ Subtract M from A
00000	1110	→ Restore A
00000	1110	→ clear q_0.
00001	1100	→ shift left
11110	1100	→ Subtract M from A.
00001	1100	→ Restore A.
00001	1100	→ clear q_0.

00000 — A
11101 — M.
11101

~~Subtract~~

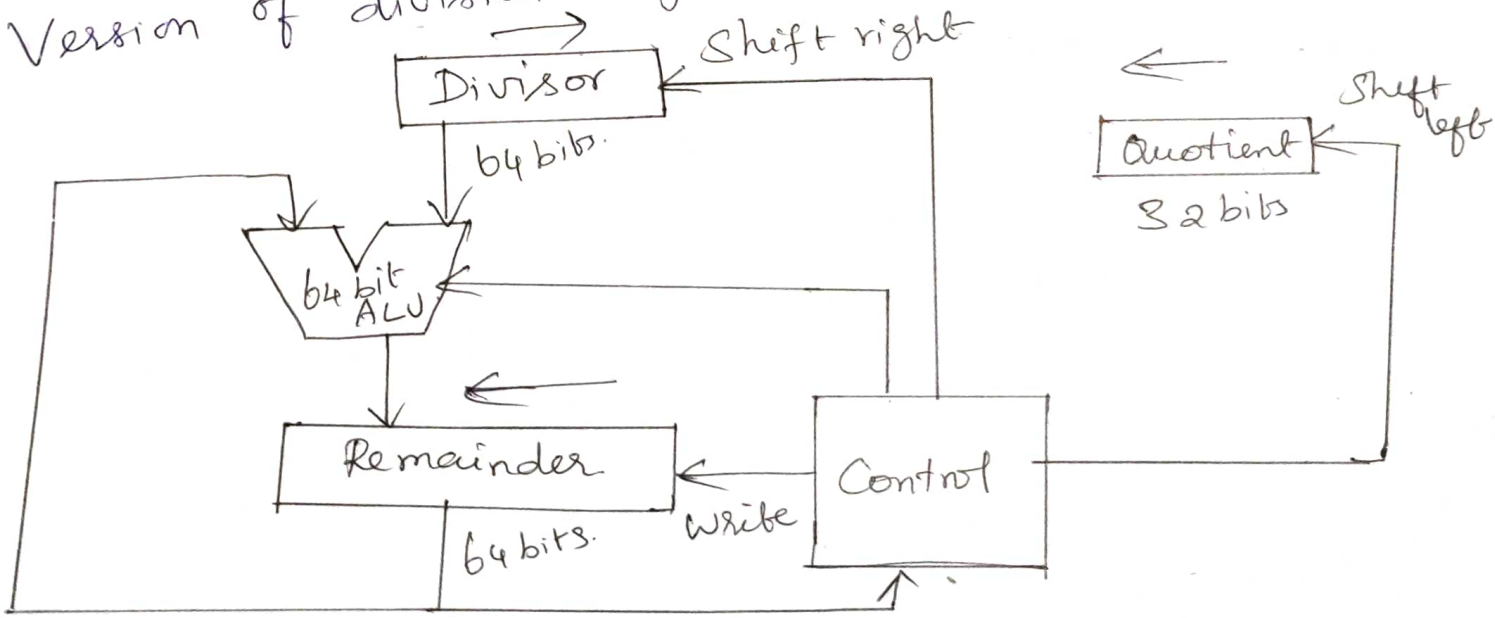
00011	1000	Shift left
00000	1000	Subtract M from A
00000	1001	Set q_0

00001	0010	→ Shift left.
11110	0010	→ Subtract M from A.
00001	0010	→ Restore A
00001	0010	→ clear q_0

Remainder Quotient.

A division Algorithm and Hardware.

The following assumptions are made in the first Version of division algorithm.



Division Hardware.

Divisor register, ALU and Remainder register are all 64 bits wide. with only the quotient register being 32 bits

* The 32 bits divisor starts in the left half of the divisor register and is shifted right 1 bit each iteration.
* 32 bit quotient register is set to 0
* The remainder is initialized with the divident
* Control decides when to shift the divisor and quotient and when to write the new value into the remainder register

Floating Point addition.

Most Computers Uses dedicate hardware ... floating point operations as fast as possible.

① Align the decimal number of the Smallest exponent number that matches the larger exponent. (Shift the Smaller number right until exponent match). ie) exponents of the operands are made equal.

② Add/ Subtract the significands dependending on Sign.

③ Normalise the Sum by adjusting exponent.

④ check for overflow

⑤ Rounding off to appropriate numbers of bits

⑥ Result may need further normalization; then go to Step 3.

⑦ Set the Sign, Sign is determined by addition of the significand.

Example: Perform $9.999_{ten} \times 10^{1} + 1.610_{ten} \times 10^{-1}$

Assume that we can Store only four decimal digits of the significand and two decimal digits of the exponent.

Step 1: Compare the exponent of both the operands If it equal add the two operand (significand). obtained If it is not equal then increases the Smaller exponent to the larger exponent.

$0.1610 \times 10^1 = 0.1610 \times 10^0 = 0.0161 \times 10^1$

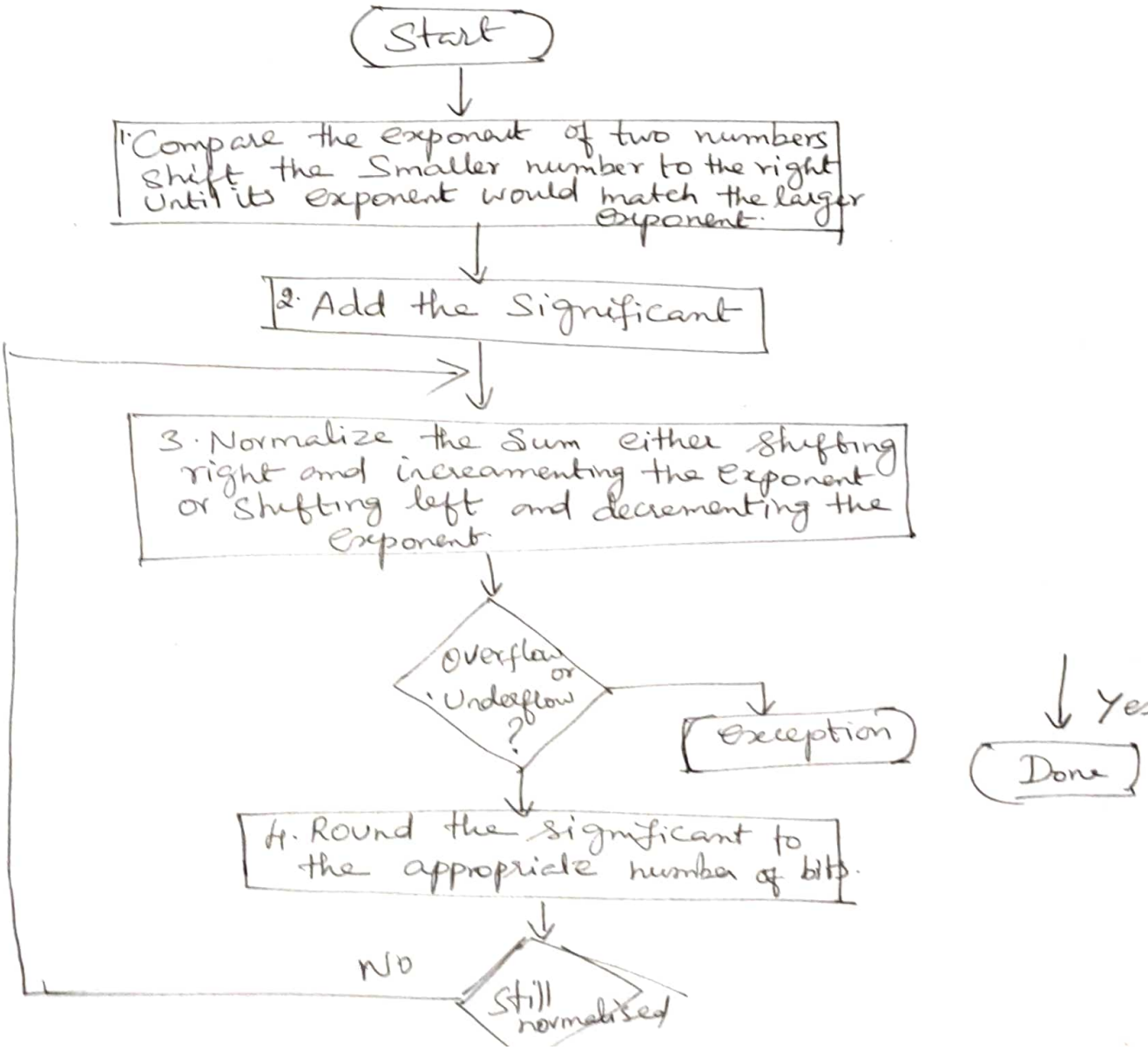
Step2: Now add the significand.

$$\begin{array}{r} 9.999 \\ 0.016 \\ \hline 10.015 \end{array}$$

Step3: This sum is not in normalized scientific notation, so we need to adjust it

$10.015 \times 10^1 = 1.0015 \times 10^2 = 1.0015 \times 10^2$.

Step4: Round the number to four digits in the significand to $1.002_{ten} \times 10^2$.



Subword Parallelism.

→ Many Scientific Problems require arithmetic operations on large arrays of numbers these numbers are usually formulated as vectors. A vector is an ordered set of a one-dimensional array of data items.

→ It has been observed that many scientific problems and graphics and audio applications perform the same operation of vectors.

→ Suppose the size of the data item within the vector is 128 bits. With this data, by partioning the carry chains a processor can use parallelism to perform simultaneous operations on short vectors of sixteen 8-bit operands, eight 16-bit operands four 32 bit operands or two 64-bit operands. Such parallelism which occurs within a wide word is known as Subword Parallism.

→ The Subword Parallelism is also classified under the more general name of data level parallelism. It is also called vector or SIMD. (for single Instruction Multiple Data).

→ ARm. Processors have more than 100 instructions in the NEON multimedia Instruction extension to Support Subword Parallelism.

Floating-Point Representation (11)

→ To accommodate very large integers and very small fractions, a computer must be able to represent numbers and operate on them in such a way that the position of the binary point is variable and is automatically adjusted as computation proceeds.

→ In this case the binary point is said to float and the numbers are called floating point numbers.

→ The floating point representation has three field: sign, significant digits and exponent.

→ Let us consider the number 111101.1000110 to be represented in the floating point format.

→ It is important to 111101.1000110 ⟶ 1.11101100110 × 2 (exponent: 5)

→ The string of the significant digits is commonly known as mantissa.

In above example.

Mantissa = 11101100110.

Exponent = 5

IEEE Standard for floating Point Numbers.

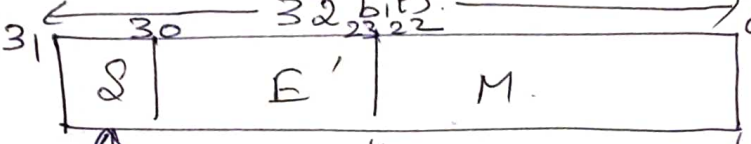
→ The standard for representing floating point numbers in 32 bits and 64 bits have been developed by Institute of Electrical and Electronics Engineers (IEEE) referred to as IEEE 754 standard.

→ 32 bit Standard representation is Single Precision representation. The 32 bits are divided into three field

(field 1) Sign → 1-bit
(field 2) Exponent → 8 bit
(field 3) Mantissa → 23 bits

→ $E' = E$ (Scaling factor) + bias.

Single-Precision



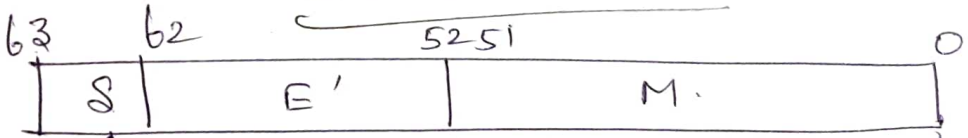
Sign of number
0 signifies.
1 signifies.

8 bit signed exponent in excess -127 representation

23 bit mantissa fraction.

Value represented $= \pm 1.M \times 2^{E'-127}$

Double Precision.



Sign.

11-bit excess -1023 exponent.

52-bit mantissa fraction.

Value Represented $= \pm 1.M \times 2^{E'-1023}$.

→ In 32 bit floating point system (Single Precision) bias is 127. Hence $E' = E$ (scaling factor) + 127. This representation of exponent is also called as the excess 127 format.

$(4EB)_{16}$ → Hex. no

0100 1110 1011 → Binary number.

(12) 1000 – 8, 1001 – 9, 1010 – 10 A, 1011 – 11 B, 1100 – 12 C, 1101 – 13 D, 1110 – 14 E

$(1259)_{10} = (1001110\,1011)_2$

$(0.125)_{10} = (0.001)_2$

Binary number = 1001110 1011 + 0.001.

= 1001110 1011.001

Step 2: Normalised the number. 1001110 1011.001 = 1.001110 1011001 $\times 2^{10}$.

Step 3: Single Precision representation: for a given number $S = 0$, $E = 10$.

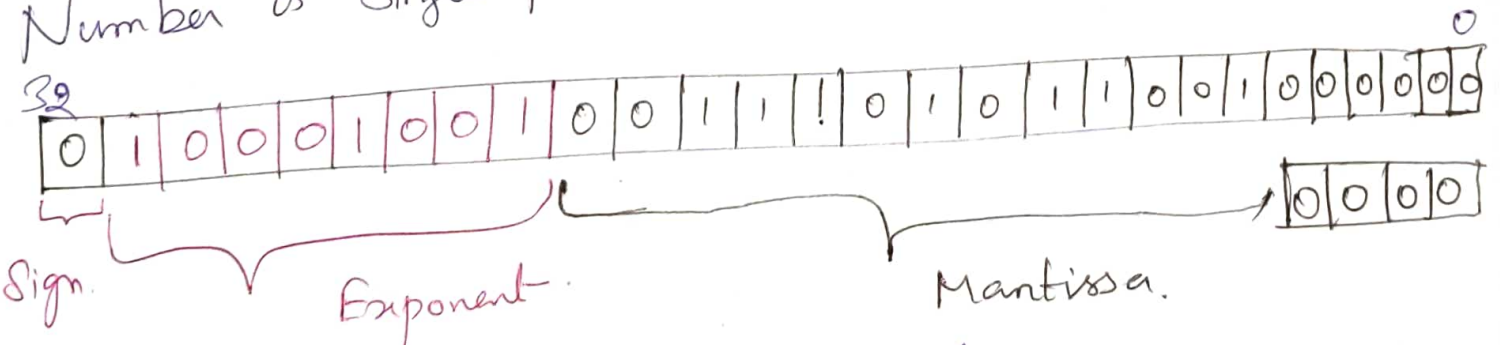
$M = 00111010 11001$

Bias for Single Precision format is 127

$E' = E + 127 = 10 + 127 = 137_{10}$.

$= 1000\,1001_2$

Number is Single Precision format is given as.



Step 4: Double Precision representation

for given number.

$S = 0$, $E = 10$, and $M = 00111010 11001$

Bias for double Precision format is = 1023.

$E' = E + 1023 = 10 + 1023 = 1033_{10}$.

$= 1000000\,1001_2$.

→ The range of E' for normal values in single Precision is $0 < E' < 255$. This means that for the representation the actual exponent E is in the range $-126 \leq E \leq 127$.

→ The 64 bit standard representation is called a double Precision representation. because it occupies Two 32 bit word.

→ 64 bits are divided into three field
field 1 sign → 1 bit
field 2 Exponent → 11 bit
field 3 Mantissa → 52 bits.

→ In the double Precision format value actually Stored in exponent field is given as.

$E' = E + 1023$.

Here bias value is 1023 and hence it is also called Excess 1023 format.

→ Thus range E' for normal values in double Precision is $0 < E' < 2047$. This means that for 64 bit representation the actual exponent E is in range $-1022 \leq E \leq 1023$.

Represent 1259.125_{10} in single Precision and double Precision formats.

Step 1:

16 | 1259 | 11 (LSB)
16 | 78 | 14
4 (MSB)

$(1259)_{10} = (4EB)_{16}$.

fraction.

$0.125 \times 2 = 0.25$
$0.25 \times 2 = 0.50$
$0.50 \times 2 = 1.00$

0.001

Point Multiplication.

Multiplying Decimal Numbers in Scientific Notation

$1.110_{ten} \times 10^{10} \times 9.200 \times 10^{-5}$.
$_{ten}$

Assume that we can store only four digits of the significand and two digits of the exponent.

Step 1: Unlike addition we calculate the exponent of the product by simply adding the exponents of the operands.

New exponent = 10 + (−5) = 5

let's do this with the biased exponents as well to make sure we obtain the same result.

10 + 127 = 137
−5 + 127 = 122.

New exponent = 137 + 122 = 259.

This result is too large for the 8 bit exponent field.

So New Exponent = 137 + 122 − 127
= 259 − 127 = 132 ⇒ (5 + 127)

So to get correct biased sum when we add biased numbers we must subtract the bias from the sum.

Step 2: Next comes the multiplication of the significants.

$$\begin{array}{r} 1.110_{ten} \\ 9.200_{ten} \\ \hline 0000 \\ 0000 \\ 2220 \\ 9990 \\ \hline 10212000_{ten} \end{array}$$

The Product is 10.21200_{ten}.

Assuming that we can keep only three digits to the right of the decimal point.

$10.212_{ten} \times 10^5$.

Step3: This product is unnormalized, so we need to normalize it

$10.212_{ten} \times 10^5 = 1.0212 \times 10^6$.

Step 4: Round of the Product. $1.021_{ten} \times 10^6$.

Step 5: The sign of the Product depends on the sign of the original Operands. If they are both the same, the sign is positive; otherwise it's negative. Hence the product is

$+1.021_{ten} \times 10^6$.

Ex Multiply the numbers 0.5_{ten} and -0.4375_{ten} using floating point multiplication algorithm.

Binary equivalent of $0.5_{ten} = 1.000 \times 2^{-1}$

$-0.4375_{ten} = -1.110 \times 2^{-2}$.

Step 1: Adding the exponents without bias.

$-1 + (-2) = -3$.

or Using biased representation

$(-1+127) + (-2+127) - 127$

$= (-1-2) + (127+127-127) = -3 + 127 = 124$

Step2: multiplying the Significand.

```
  1.000
  1.110
-------
  0000
 1000
1000
1000
-----------
1110000 two
-----------
```

The Product is 1.110000×2^{-3} but we use 4 bits, so it is 1.110×2^{-3}.

Step3: Normalize the Product as per our example

$1.110_{two} \times 2^{-3}$.

Step 4: Rounding the Product makes no change.

$1.110_{two} \times 2^{-3}$.

Step 5: Since the Sign of the Original Operands differ make the Sign of the Product negative.

-1.110×2^{-3}.

Converting to decimal to check our result.

$-1.110 \times 2^{-3} = -0.00111 = \frac{1}{8} + \frac{1}{16} + \frac{1}{32} = 0.21875_{10}$

Advantages of floating Point

* More bits can be stored in the instruction format.
* Floating point has separate registers.

Disadvantages of floating Point

* Floating Point operation requires much more hardware than integer operations.
* Implementing floating point arithmetic is more complex.

Floating Point Arithmetic

• $1011 \times 2^7 + 0.111 \times 2^4$

Addition

$0.1011 \times 2^7 \Rightarrow 0.1011 \times 2^7$

$0.0111 \times 2^4 \Rightarrow 0.0000111 \times 2^7$

0.10111111×2^7

The nearest four bit number is then 0.1100×2^7.

High Performance Multiplication

A number of methods exist for Speeding the Process of multiplication. Two methods are described in this Section.

① first approach gain Performance by Skipping over blocks of 1's which eliminates addition Steps

② Next approach is a Parallel multiplier in which cross product among all pairs of multiplier and multiplicand bits is formed.

Booth Algorithm

* Booth's algorithm is a powerful algorithm for Signed number multiplication, which treats both positive and negative numbers Uniformly

* In this algorithm, multiplier bits are encoded from right to left before the bits are used for getting Partial Products.

* To encode the multiplier bits, two adjacent bits b_i (current bit) and b_{i-1} (previous bit) is examined as follow

1. If i^{th} bit b_i is 0 and $(i-1)^{th}$ bit b_{i-1} is 1 then take b_i as +1 (ie, for (0 1) pair. Multiplicant is added to the partial product.
2. If i^{th} bit b_i is 1 and $(i-1)^{th}$ bit b_{i-1} is 0, then take b_i as -1 (ie for (1,0) pair, Multiplicant is subtracted to the partial Product.
3. If i^{th} bit is 0 and $(i-1)^{th}$ bit b_{i-1} is 0 then take b_i as 0.
4. If i^{th} bit b_i is 1 and $(i-1)^{th}$ bit b_{i-1} is 1, then take b_i as 0 (ie for (0,0) and (1 1) pair, then neither addetion nor subtraction is Performed.
5. When least Significant bit $b_0 = 1$ (0) assume that it had b_{i-1} as 0 (implied Zero), thus take $b_0 = -1$(0)

Multiplier		Version of Multiplicand
Bit i	Bit i-1	Selected by bit i
0	0	0 X M
0	1	+1 X M
1	0	-1 X M
1	1	0 X M.

Booth Multiplier Recoding Table.

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

0 +1 -1 +1 0 -1 0 +1 0 0 -1 +1 -1 +1 0 -1 0

Booth recoding of a Multiplier.

Example Recode the multiplier 101100 for Booth's multiplication.

1 0 1 1 0 0 [0]. implied zero.

−1 +1 0 −1 0 0 Recoded multiplier.

Example Multiply 001100 (+12) and 010011 (+19)

Multiplier : 001100. Multiplicand : 010011

Recoded Multiplier. : 0 +1 0 −1 0 0.

Multiplication:

0 1 0 0 1 1

0 +1 0 −1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 → if multiplier & multiplicand 6 digit add six zeros. = 12

0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 0 1 1 0 1 → 2's Comp of Multiplicand.

0 0 0 0 0 0 0 0 0

0 0 0 1 0 0 1 1

0 0 0 0 0 0 0

1 0 0 0 1 1 1 0 0 1 0 0 (228)

The same algorithm can be applied for negative multiplier and negative multiplicand.

Multiply 01110 (+14) and 11011 (-5).

Multiplier : 1 1 0 1 1 Multiplicand : 0 1 1 1 0

Recoded Multiplier : 0 -1 +1 0 -1

```
        0  1  1  1  0
        0 -1 +1  0 -1
  ---------------------
  1  1  1  1  1  1  0  0  1  0
  0  0  0  0  0  0  0  0  0
  0  0  0  0  1  1  1  0
  1  1  1  0  0  1  0
  0  0  0  0  0  0
  ---------------------
1 1  1  1  0  1  1  1  0  1  0   = (-70)
```

Hardware Implementation of Booth's algorithm.

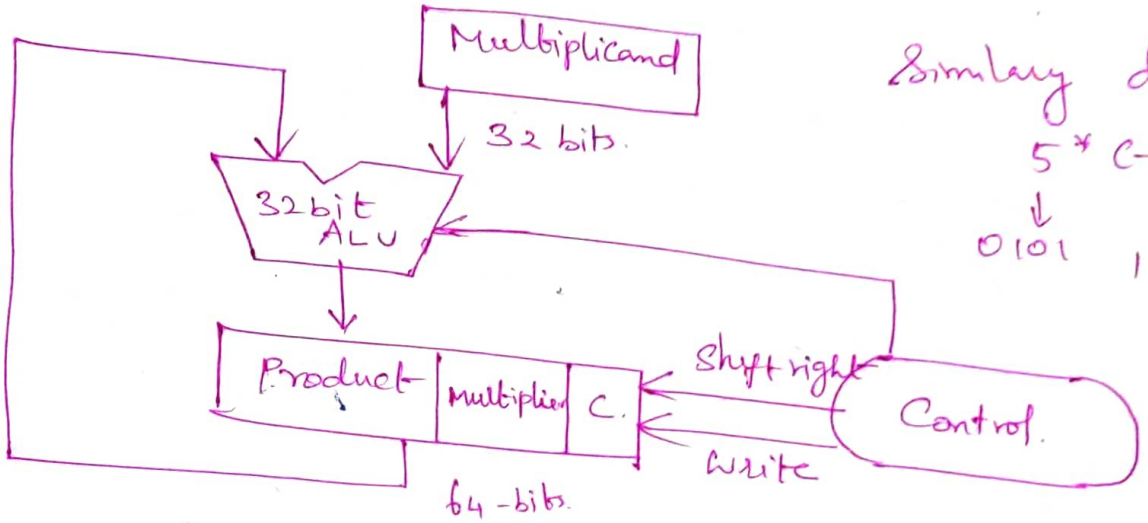
The hardware Consists of 32 bit register M for the multiplicand 64 bit Product register PR and a 1-bit register C, 32 bit ALU and Control. Initially M contains multiplicand, PR Contains multiplier in lower half of register and initialize the upper half of Product register to 0. and C Contains bit 0. The algorithm is the following steps.

Repeat 32 times.

1. If (Q0 C) Pair is
   * 10 : PR = PR - M.
   * 01 : PR = PR + M.
   * 00 and 11 : do nothing.
2. Arithmetic Shift PR right 1 bit (Arithmetic shift Presaves the sign of a two's complement number and it is equivalent to logical shift with sign extension)

The shift Out bit gets into C.

In Booth's algorithm if the multiplicand and multiplier are n-bit two's Complement numbers. the result is Considered as 2n-bit two's Complement Value. The overflow bit C is ignored.



Hardware for Booth's Multiplication

Simlay do
5 * (-4)
↓ ↓
0101 1100

Case 1: Both Positive (5 * 4)

Multiplicand ← 0101 (5) Multiplier ← 0100 (4)

Steps	PR		C	Operation
Step 1	0000	0100	0	Initial
	0000	0010	0	Shift right
Step 2	0000	0001	0	Shift right
Step 3	1011	0001	0	PR ← PR - M
	1101	1000	1	Shift right
step 4	0010	1000	1	PR ← PR + M
	0001	0100	0	Shift right

Result: 0001 0100 +20.

UNIT-III

THE PROCESSOR

Introduction

* MIPS (Microprocessor without Interlocked Pipeline Stages). Is a reduced Instruction Set Computer (RISC) developed by MIPS Technologies.
* Multiple revisions of the MIPS Instruction Set exist, including MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS 32, and MIPS 64. The current revisions are MIPS 32 (for 32 bit implementation) and MIPS 64C for 64-bit implementations).

Characteristics of MIPS

* ALU Operations have 3 register Operands.
* Simple instruction Set.
* Uniform encoding:
  - → All instructions are 32 bit long.
  - → opcode is always in the high order 6 bits.
  - → 3 types of instruction format.

Registers for MIPS

→ MIPS 64 has 32 64-bit general Purpose registers (GPRs). named $R_0, R_1 \ldots R_{31}$.

→ GPR are also sometimes known as Integer register

→ Additionaly there is a Set of 32 floating Point registers (FPRs) named $F_0, F_1 \ldots F_{31}$.

→ Both Single and double Precision floating Point operations (32-bit and 64 bit) are Provided.

→ MIPS also include instructions that operate on two Single Precision Operands in a Single 64-bit floating Point register.

A basic MIPS Implementations

The basic MIPS implementation that includes a subset of the core MIPS instruction set.

* The Memory Reference Instruction load word (lw) and store word (sw).
* The arithmetic-logical Instruction add, sub, AND, OR and slt.
* The instructions branch equal (beq) and jump (j).

Logic Design Convention

→ To design any Computer, we must decide how the hardware logic implementing the Computer will operate and how Computer is clocked.

→ The data path element in MIPS implementation consists of two different types of logic elements.

a) Combinational Element
b) State Element.

Combinational Elements

* Elements that operate on data values and elements that contain State.
* Output depends only current inputs.
* Does not have Internal Storage.

Example AND gate, multiplexer and ALU.

State Elements

* It Contain internal Storage.
* State element has atleast two inputs and one output
* State element also called Sequential elements because O/p depends on current as well as Previous input

* The inputs are data values written into the element and the clock which determines when the data value is written.

Example : 1. D - flip-flop → which has exactly two inputs. (a value and clock) and one o/p

2. Memories and Registers.

Clocking Methodology

* A clocking Methodology defines when signals can be read and when they can be written.

* It is important to specify the timing of reads and writes, because if a signal is written at the same time it is read, the value of the read would correspond to the old value and newly written value or even some mix of the two.

* clocking Methodology is designed to make hardware predictable.

Edge Triggered clocking

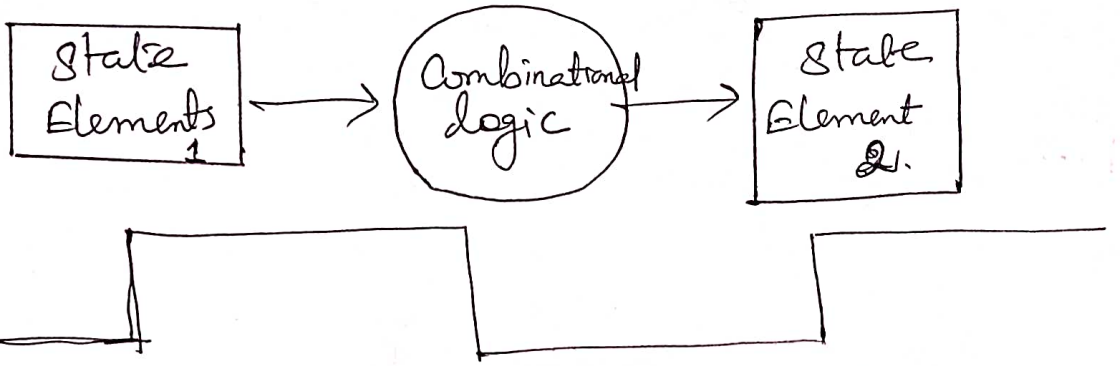
* Edge triggered clocking methodology means any value stored in a sequential logic element are updated only on a clock edge.

* It is a quick transistion from low to high or high to low.

* Because only state elements can store a data value, any collection of combinational logic must have its inputs come from a set of state elements and its o/p written into a set of state elements.

Typical Execution.

* Read contents of some State elements.
* Send values through some combinational logic
* Write results to one or more State elements.
*



Clock cycle.

Combinational logic, State elements and the clock are closely related.

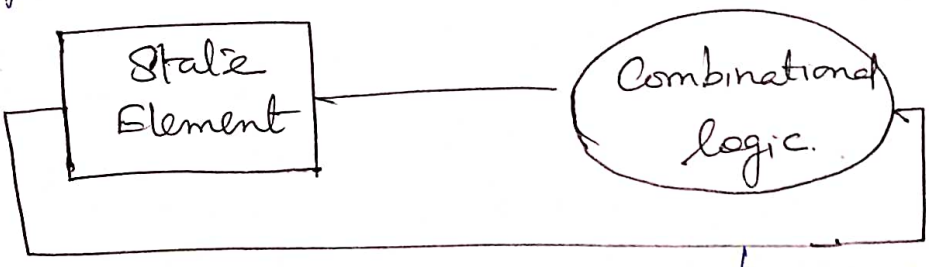
* It shows two State element surrounding a block of Combinational logic, which operates in a single clock cycle, all signals must propagate from State element 1, through Combinational logic and to state element 2. in the time of one clock cycle.
* The time required for the signals to reach State element 2 is called as length of the clock cycle.
* If the State element is not updated on every clock then an explicit write control signal is required.
* Both the clock cycle and write control signal are input and State element is changed only when the write control signal is asserted and a clock edge occurs.
* Control signal is a signal used for multiplexer selection for directing the operation of a functional unit
* Two logical terms are used to represent the signal level.

* The term asserted to indicate a signal that is logically high and assert to specify that a signal should be driven logically high.

* The term deasserted to represent logically low.

* An edge triggered methodology allows us to read the content of register, send the value through some combinational logic and write that register in the same clock cycle.

example.



→ After reading process was done send the value through some combinational logic and write the register in the same clock cycle.

This allows the processor to read the register content send the value through some combinational logic and write that register in same clock under the assumption that the state elements are controlled by implicit clock cycles.

Building Data Path

→ A Data path is a collection of functional units organised in a manner to execute each class of instruction.

→ A data path is a representation of the flow of information (data and instruction) through the CPU implemented using Combinational and Sequential circuitry

→ Single cycle datapath that have Separate instruction and memories because.

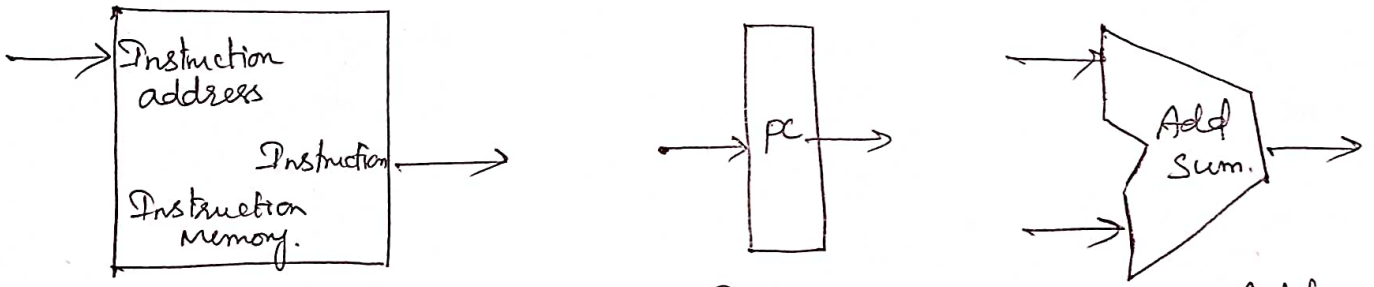
① The format of data and instructions is different in MIPS and hence different memories are needed.

② Having Separate memories is less expensive.

To design a data path first we should examine the major components required to execute MIPS instruction.

Data path elements

→ Data path elements include. 1. Instruction memory.
2. Program Counter 3. Adder, 4. Register file.
5. ALU, 6. Data memory. 7. Sign Extend. 8. Multiplexers.



Instruction Memory Program Counter Adder

Major Components to execute MIPS Instruction

Instruction Memory

→ It is first data path element, memory Unit to store the instructions of a Program and Supply instructions given an address.

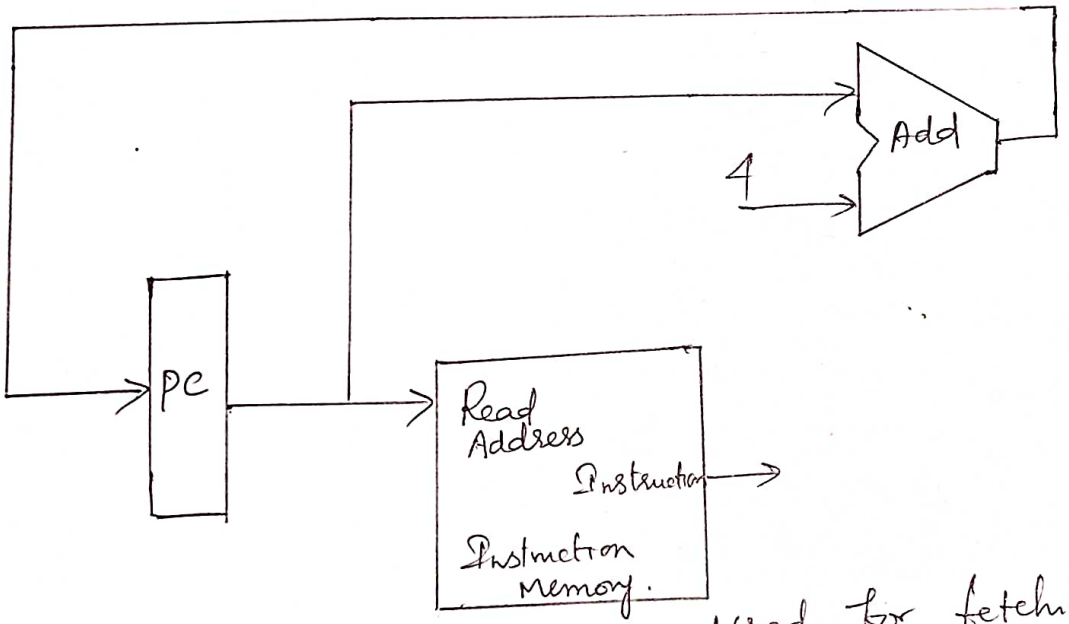
Program Counter

Program counter is used to hold the address of current instruction.

Adder → It adds two 32 bits input and place the Sum on its output. It is also increment the PC to the address of the next instruction

Fetch operation.

→ The fundamental operation in Instruction Fetch is to send the address in the PC to the instruction memory and obtain the specified instruction, and the increment the PC.



A portion of the datapath used for fetching instruction and incrementing the Program Counter.

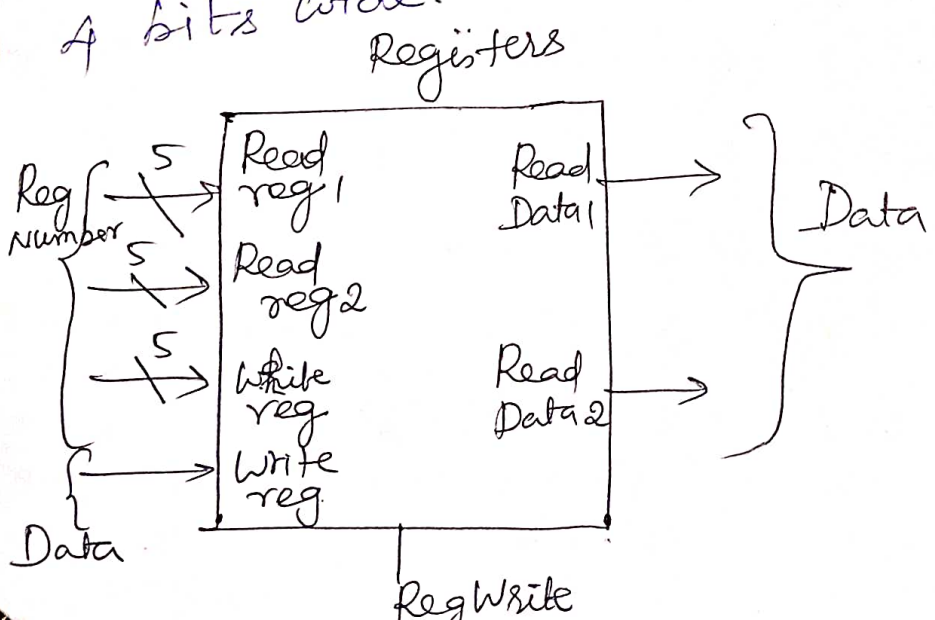
Register file.

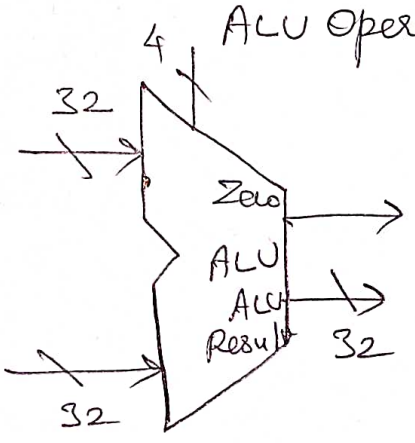
* The Processor's 32 general-purpose registers are stored in a structure called a register file.
* A register file is a collection of registers in which any register can be read or written by specifying the number of the register in the file.

* The register file contains the register state of the machine.
* In addition we will need an ALU to operate on the values read from the registers.
* Because the R-format instructions have three register operands, we will need to read two data words from the register file and write one data word into the register file for each instruction.
* For each data word to be read from the register we need an 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.

ALU → It 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 operation to be performed by the ALU is controlled with the ALU operation signal, which will be 4 bits wide.

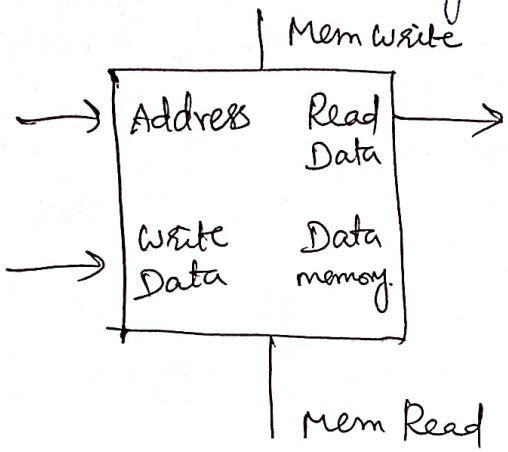




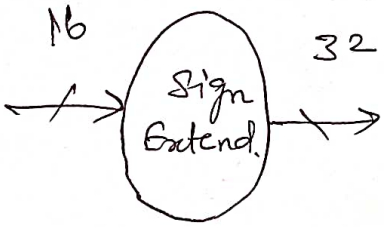
ALU.

Data Memory.

Data Memory must be written on Store instruction hence. it has both read and writes control signals. an address input as well as an input for the data to be written into memory.



Data Memory Unit



Sign extension Unit.

Multiplexors

four multiplexers are used in Simple Data path

Multiplexor 1: Select Second ALU Operand.

It is used to select the second ALU operand It is controlled by ALUSrc control signal. It selects any one of the following two values as the second input to the ALU.

1. Register value for R-type instruction.
2. 32 bit sign extended value for memory instruction.

Multiplexer 2: Select input for Write data.

→ It is used to select the data for write data input of register file. It is controlled by MemtoReg. Control signal. The Two possible inputs are the following and this multiplexer select any one of the following two values.

① Data from memory for a memory instruction.
② ALU result for a R-type instructions.

Multiplexer 8: Selects the next PC value.

It is used to select the next value for the PC. It is controlled by PC Src control signal. The two possible cases are as follows as this multiplexer is used to select any one of the following two values.

1. PC + 4
2. Branch Address.

Multiplexer 4: Selects the destination register.

It is used to select the destination register. It is controlled by RegDst control signal. It select one of the two values.

① Destination comes from rt field of the instruction.

② Destination comes from rd field of the instruction.

Data Path for Branch Instruction.

Branch Target address calculation.

→ 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.

→ The Syntax for the branch instruction 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.

→ Two important things in the definition of branch instructions:

① The instruction set architecture specifies that the base for the branch address Calculation is the address of the instruction following the branch (ie, PC + 4. the address of next instruction.

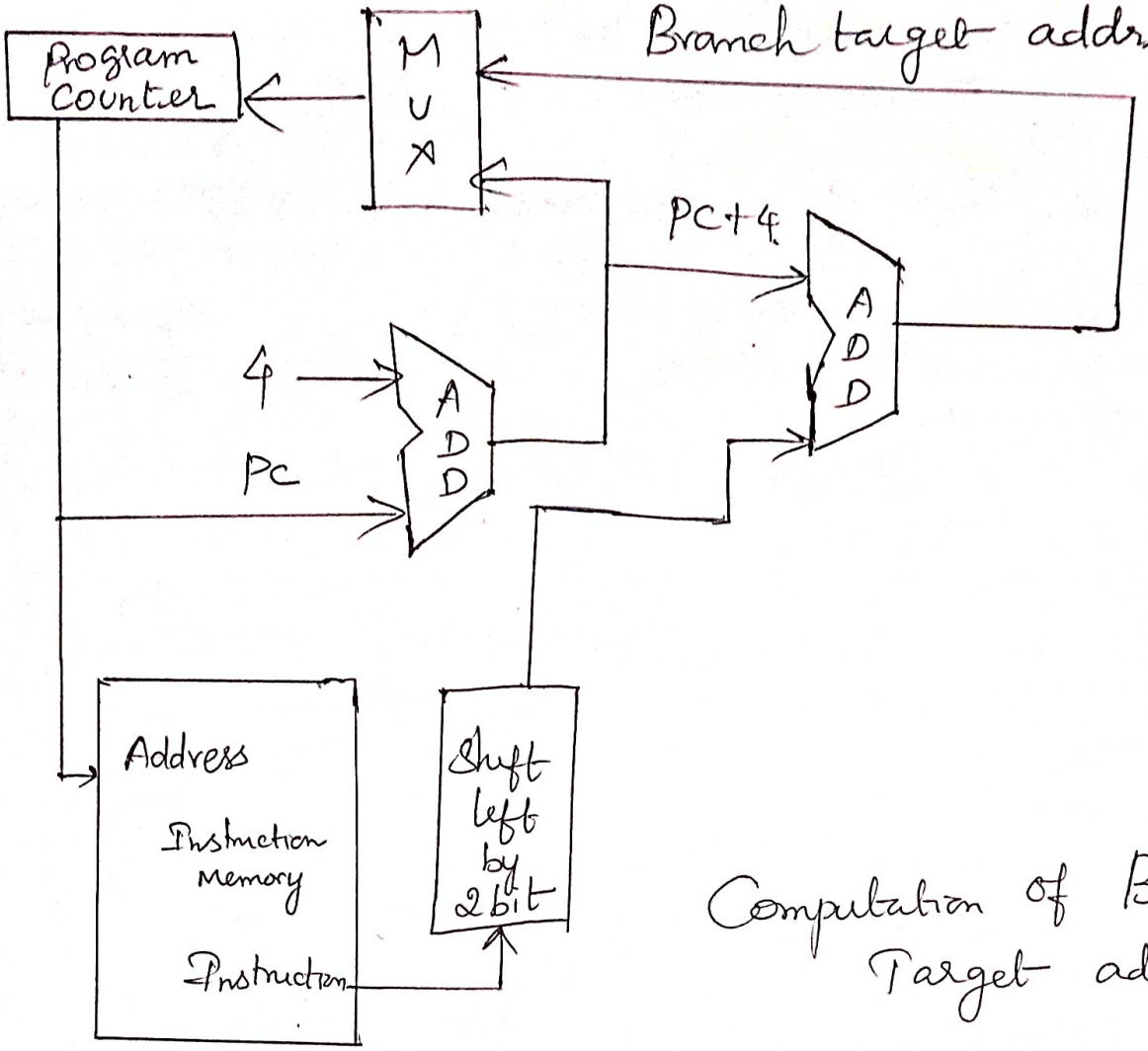
② The MIPS architecture also states that the offset field is shifted left 2 bits so that it is a word offset; this shift increases the effective range of the offset field by a factor of 4. Then to shift the offset field by two.

→ Therefore branch target address is given by

Branch Target address = PC + 4 + offset (shifted left 2 bits)

→ In addition to Computing the branch target address. we must also see whether the two operands are equal or not. If two operands are not equal the next instruction is the instruction that follows sequentially (PC = PC + 4). in this case, we say that the branch is not taken.

→ On other hand if two operands are equal (ie condition is true), the branch target address becomes the new PC, and we say that the branch is taken.



Computation of Branch Target address.

Jump Instruction

→ The Jump Instruction operates by replacing the lower 28 bits of the PC with the lower 26 bits of the instruction shifted left by 2 bits. This shift is accomplished simply by concatenating 00 to the Jump Offset, accomplishes this shift.

Simple Implementation Scheme.

→ Control implementation scheme can be build Using datapath and some implementation method.

→ This Simple Implementation covers load word (lw) store word (sw), branch equal (beq) and arithmetic logical instructions add, sub, and, or, and set on less than.

The ALU Control

The MIPS ALU defines the six 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.
* for load word and store word instructions, we use the ALU to compute the memory address by addition.
* In case of the R type instruction, the ALU needs to perform one of the five actions AND, OR, Subtract, add, or, Set on less than).
* In case of Branch equal the ALU must Perform a Subtraction

We can generate the 4-bit ALU, Control Input Using a small Control Unit that has an Inputs the function field of the Instruction and 2 bit Control field. which we call ALUOp.

ALU Op	Action
00	Load and Store
01	Subtract for beq
10	The operation encoded in the function field
11	

→ Table shows how to set the ALU control input based on the 2-bit ALU op. control and 6-bit function code.

Instruction opcode	ALUOP	Instruction operation	funct field	Desired ALU Action	ALU control I/p
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	Seton less than	0111

→ Here Multiple levels of decoding technique is used.

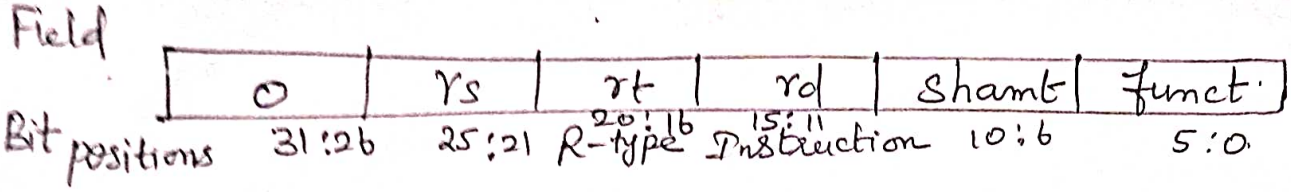
Advantages of Using multiple levels of decoding

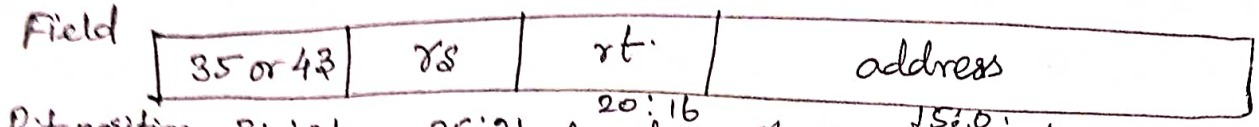
* It reduces the size of the main control unit.
* Use of several smaller control units may also potentially increase the speed of the control unit.

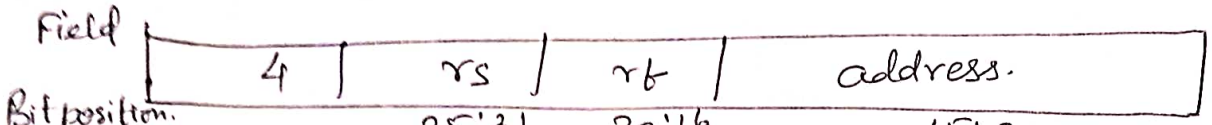
Designing the Main Control Unit

To design a main control unit, first identify the fields of an instruction and control signals needed for the datapath. It is useful to review the format of the three instruction classes.

① R type Instruction ② Branch Instruction ③ Load/Store instruction.







→ The op field also Called the opcode. is always Contained in bits 31:26.

→ The two registers to be read are always Specified by the rs and rt field. at position 25:21 and 20:16. This is true for R type Instruction branch equal and for store.

→ The base register for load and Store Instructions is always in bit position 25:21.

An Overview of Pipelining.

Pipelining

→ Pipelining is an implementation technique in which multiple instructions are executed Simultaneously by overlapping them in execution to Save time and resource and improves the throughput.

→ In a Compulier pipeline each Step in the Pipeline Completes a part of an instruction Different Steps are Completing different instructions in parallel. Each of these Steps is is called a pipeline Stage (or) Pipeline Segment.

→ Pipeling yields a reduction in the average Execution time per instruction.

MIPS instructions classically take five steps:

① Fetch instruction from memory.

② Read registers while decoding the instruction The format of MIPS instructions allows reading and decoding to occur simultaneously.

③ Execute the operation or calculate an address.

④ Access the operand in data memory.

⑤ Write the result into a register.

Single cycle Versus Pipelined Performance.

→ Consider a simple Program segment Consists of eight instruction: load word (lw), Store word (sw), add (add) Subtract (Sub), and (and), or (or), set less than (slt) and branch on equal (beq).

→ Compare the average time between instruction of a single cycle implementation, in which all instruction take 1 clock cycle, to a Pipelined implementation.

Instruction class.	Instruction fetch	Register Read	ALU operation	Data Access	Register Write	Total Time
Load word (lw)	200ps	100ps	200ps	200ps	100ps	800ps
store word (sw)	200ps	100ps	200ps	200ps	100ps	700ps
R-format (add, sub, and or, slt)	200ps	100ps	200ps			500ps
Branch (beq)	200ps	100ps	200ps			500ps

Total time for each instruction Calculated from the time for each component.

Pipeline Speedup:

→ If the stages are perfectly balanced, then the time between instructions on the pipelined processor assuming ideal Condetions - is equal to:

$$\text{Time between instructions}_{\text{pipelined}} = \frac{\text{Time between instruction}_{\text{nonpipeli}}}{\text{Number of Pipe stages}}$$

→. The above formula Suggests that a five Stage pipeline should offer nearly a five fold improvement over the 800 ps nonpipelined time or a 160 ps clock cycle.

→. Pipelining improves performance by increasing instruction throughput, as opposed to decreasing the execution time of an individual instruction.

Pipeline Hazards.

→ There are Situations in pipelining when the next instruction cannot execute in the following clock cycle. This means that it Prevents the next instruction in the instruction stream from executing during its designed clock cycle. This is called Hazard.

→ There are three different types of hazards.

① Structural Hazard ② Data Hazard ③ Control Hazard

① Structural Hazard.

→ First hazard is called a Structural Hazard. It means that the hardware cannot Support the Combination of instructions that we want to execute in the same clock cycle.

→ The Performance of pipelined processor depends on whether the functional units are pipelined and whether they are multiple execution units to allow all possible combination of instructions in the pipeline. If for some combination, pipeline has to be stalled to avoid the resource conflicts then there is a structural 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, I_1 followed by I_2. When this program is executed in a pipeline, the execution of these two instructions can be performed concurrently. In such case the result of I_1 may not be available for the execution of I_2.

→ If the result of I_2 is dependent on the result of I_1 we may get incorrect result if both are executed concurrently. for example A = 10. In the following two operations.

I_1 : $A \leftarrow A + 5$

I_2 : $B \leftarrow A \times 2$.

→ In this case data used in the I_2 depend on the result of I_1.

→ The hazard due to such situation is called data hazard or data dependent hazard.

Solution for data Hazard

Adding extra hardware to retrieve the missing item early from the internal resource is called forwarding or bypassing.

① Operand forwarding (hardware)
② Reordering code (software).

Operand forwarding → It is also called bypassing.
→ It is method of resolving a data hazard by retrieving the missing data from internal buffer rather than waiting for the data to arrive from register or memory.

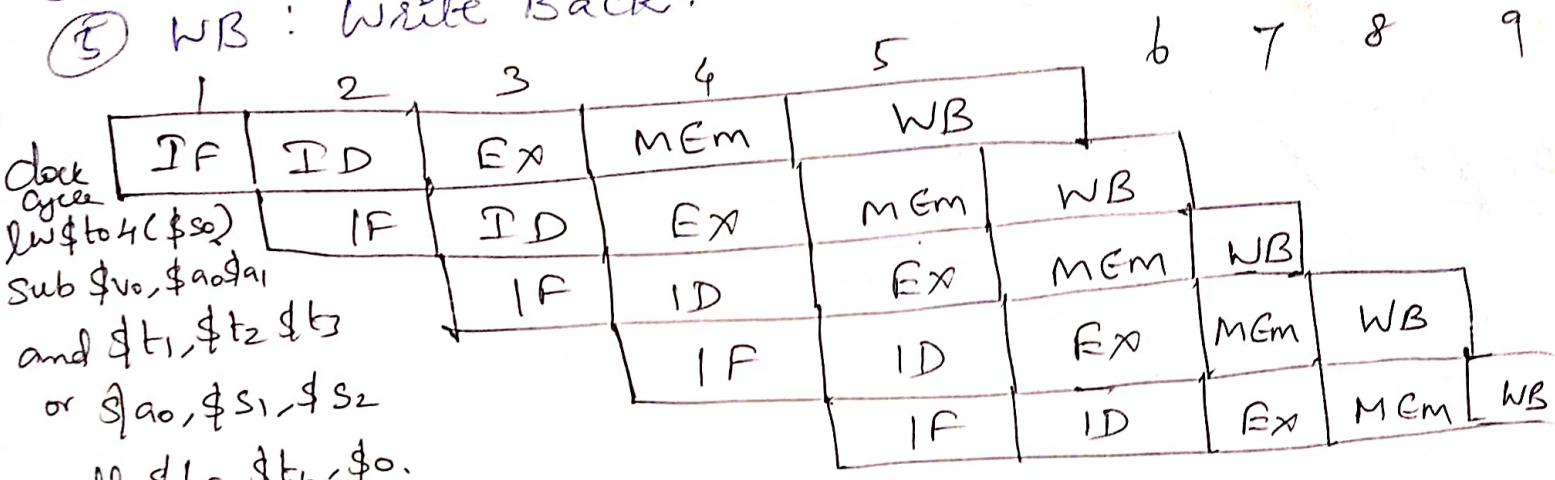
Control Hazard.

→ Control hazard also called branch hazard. arises from the need to make a decision based on the result of one instruction while others are executing.
→ When the proper instruction cannot execute in the proper pipeline clock cycle because the instruction that was fetched is not the one that is needed; that is the flow of instruction addresses is not what the pipeline expected.

Pipelined Data path and Control.

* The goal of pipelining is to allow multiple instructions execute at the same time. we may need to perform several operations in a cycle.
→ Increment the PC and add registers at the same time
→ Fetch one instruction while another one reads or writes data.
* Pipelined data path is divided into five pieces, with each piece named corresponding to a stage of instruction execution.

① IF : Instruction fetch
② ID : Instruction decode and register file
③ Ex : Execution or address calculation
④ MEM : Data memory access.
⑤ WB : Write Back.



Example : Explain the function of a Six segment pipeline and draw a space diagram for a Six segment pipelining showing the time it takes to process eight tasks.

Solution Six stages in the pipelines

① **Fetch Instruction (FI)** : Read the next expected instruction into a buffer.

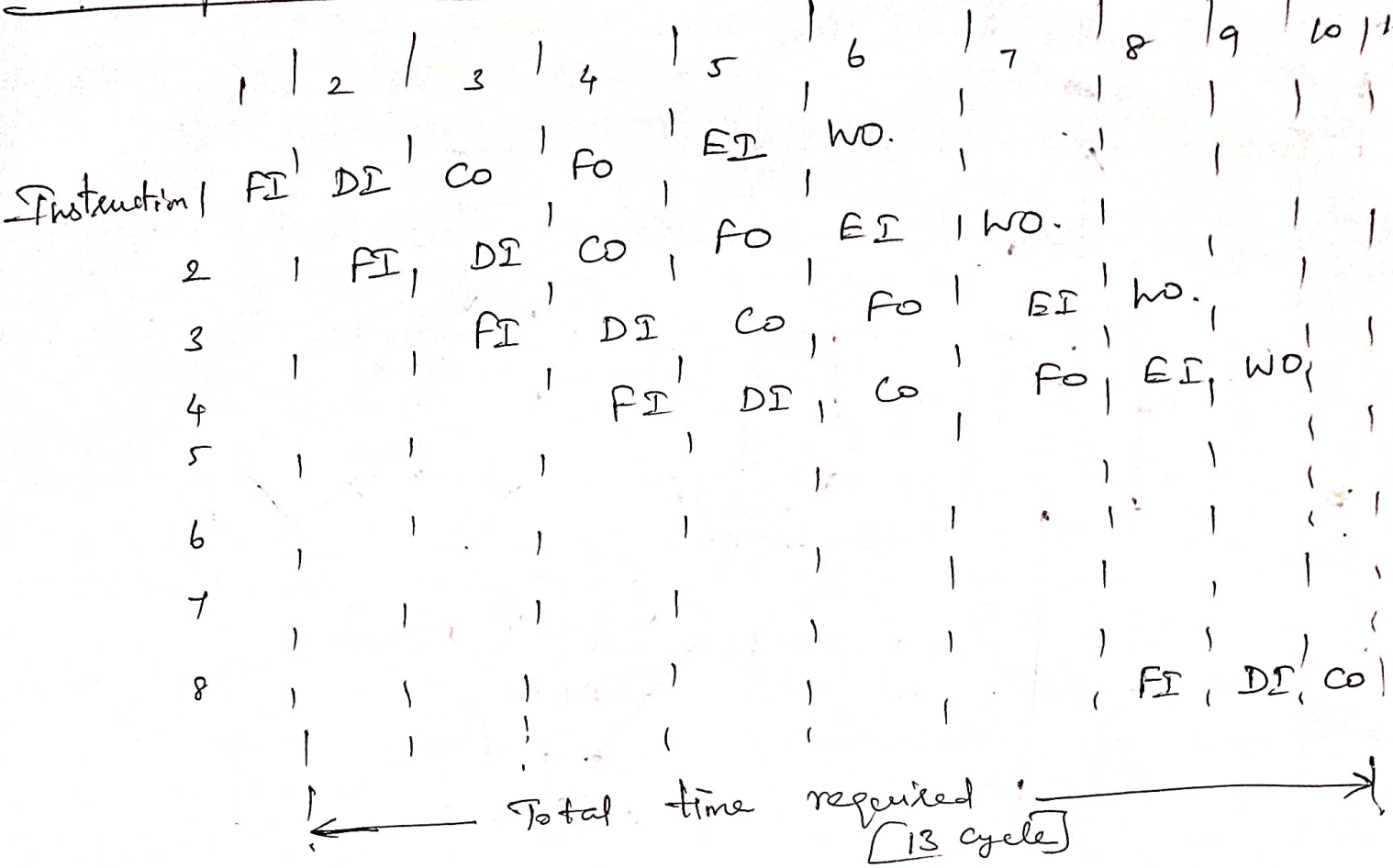
② **Decode Instruction (DI)** : Determine the opcode

③ **Calculate Operands (CO)** : Calculate the effective address of each source operand.

④ **Fetch operands (FO)** : Fetch each operand from memory.

⑤ **Execute Instruction (EI)** Perform the indicated Operation and Store the result, if any in the Specified destination operand location.

Write Operand (WO) store the result in memory.



Data Hazards Forwarding Versus Stalling:

→ Data Hazards arise when an instruction depends on the results of a previous instruction in a way that is exposed by the overlapping of instructions in the pipeline.

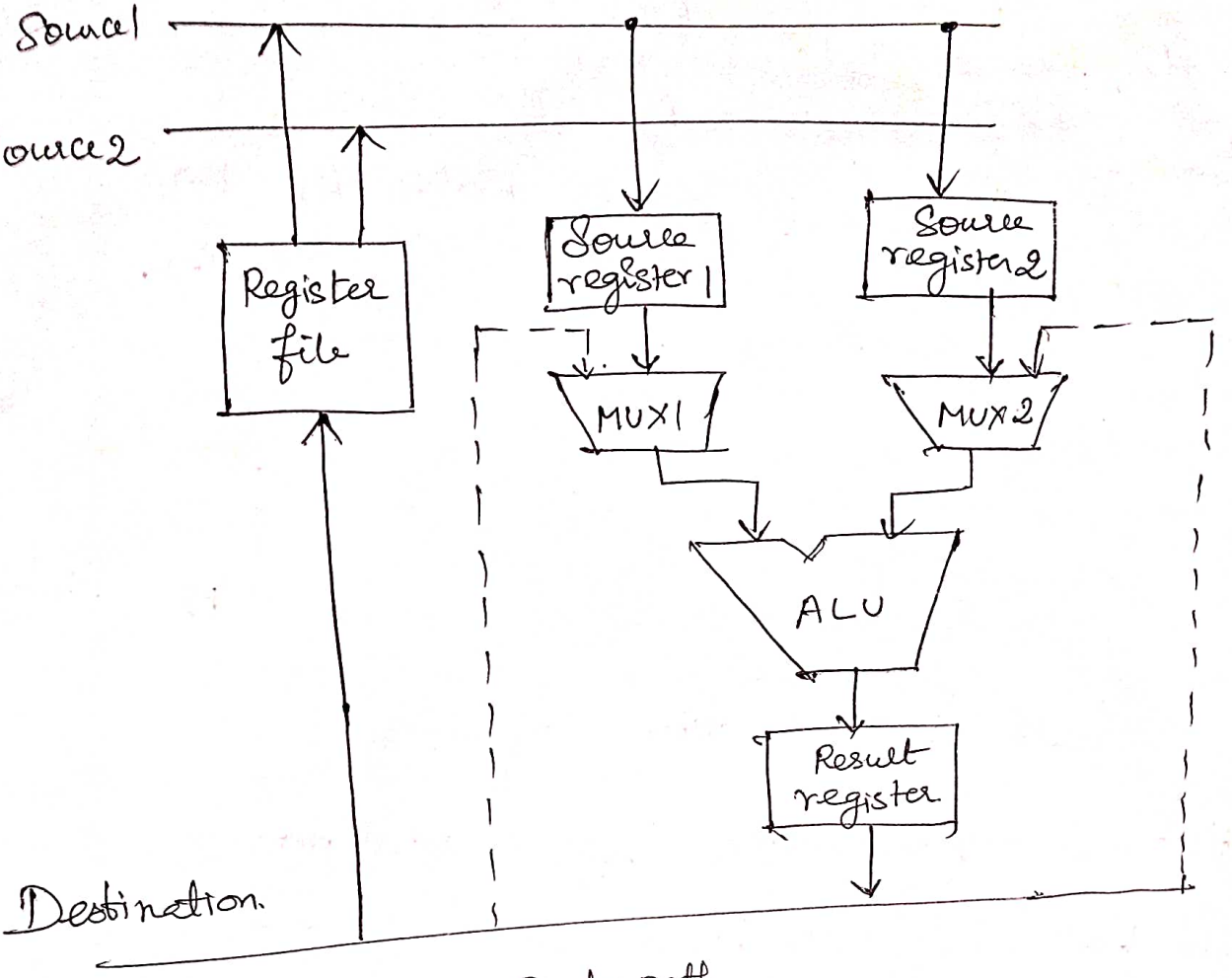
Operand forwarding

→ A simple hardware technique which can handle data hazard is called operand forwarding or register by passing. It is used to avoid the data hazards in pipeline process.

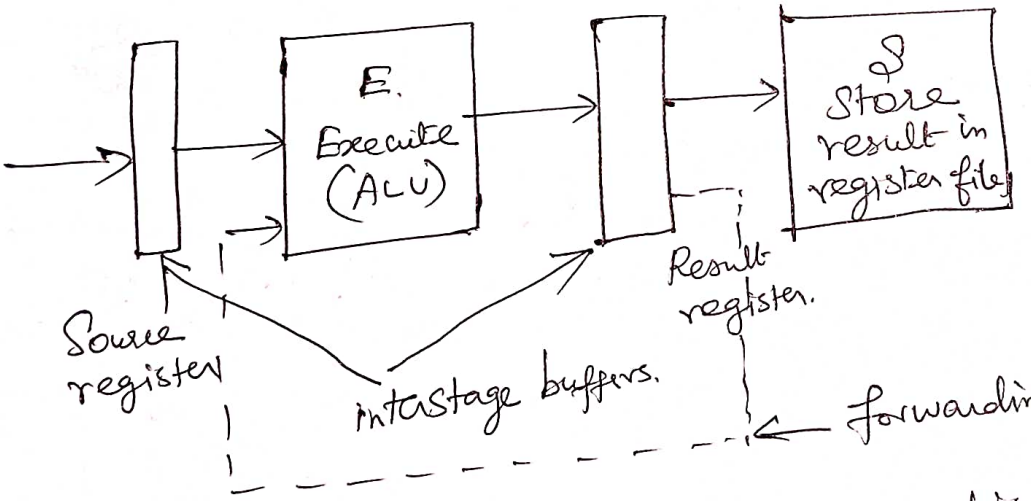
→ In this technique, ALU results are fed back as ALU inputs. When the forwarding logic detects the previous ALU operation has the operand for the current instruction. it forwards ALU output instead of register file.

→ forwarding is

→ It shows a portion of the processor data path involving the ALU and register file.



(a) Data path



Forwarding path in the processor pipeline.

Data Hazards and Stalls.

→ Hazards in pipeline can make the pipeline to stall.

→ Eliminating a hazard often requires that some instructions in the pipeline to be allowed to proceed while others are delayed.

→ when an Instruction is stalled, instructions issued latter than the stalled instruction are stopped. while the ones issued earlier must continue.

→ No new instructions are fetched during the stall.

→ In addition to a forwarding Unit we need a hazard detection Unit.

→ It operates during the ID stage so that it can insert the stall between the load and its use.

Control Hazards.

The next element to go into the pipeline may depend on currently executing instruction or we may have to wait until a stage is completed to determine the next stage.

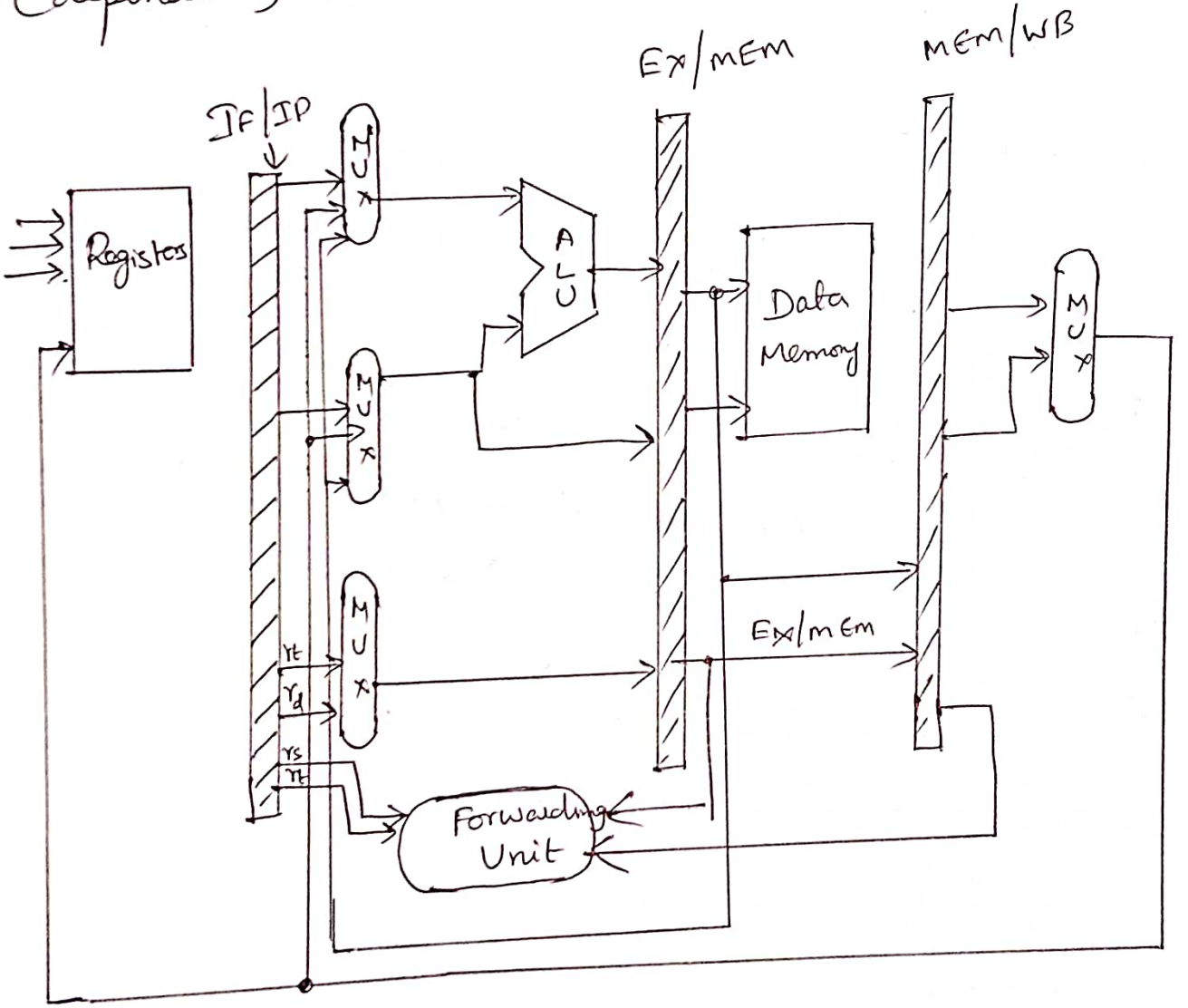
(eg) Branch Instruction.

→ An Instruction must be fetched at every clock cycle to sustain the pipeline, yet in our design the decision about whether to branch doesn't occur until the MEM pipeline stage. This delay in determining the proper instruction to fetch is called Control hazard or branch hazard.

→ The data forwarding mechanism is indicated by dashed lines.

→ The two multiplexers select the data for ALU either from destination bus or from Source 1 and Source 2. registers.

→ When the forwarding logic detects data dependency it forwards ALU output available in the result register Using data forwarding path to the ALU for the next operation. Hence the execution of next (dependent) instruction proceeds without interruption.



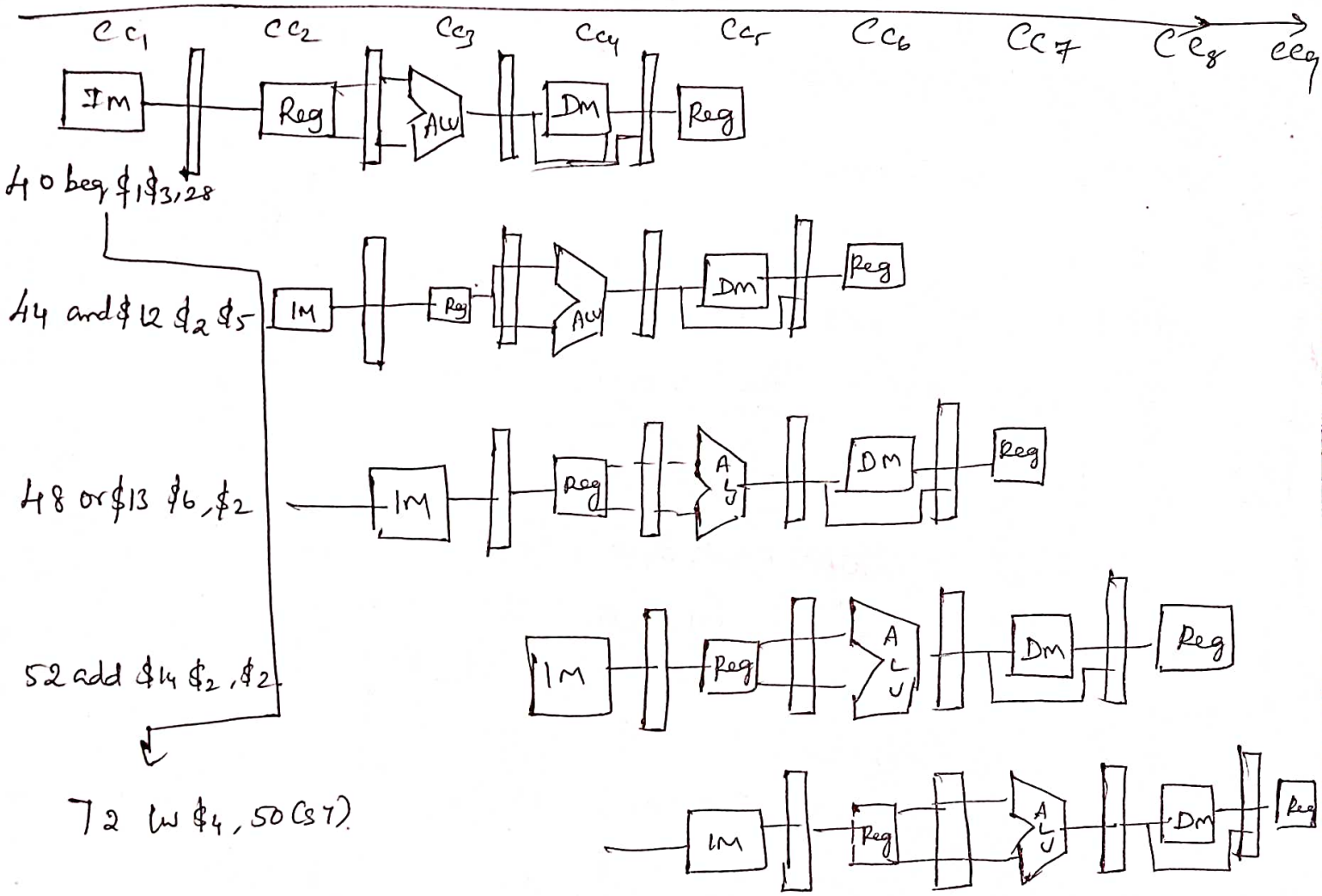
With forwarding.

ALU and Pipeline Registers. before adding forwarding

This section on Control hazard is shorter than the Previous sections on data hazards because:

a) Control hazards are relatively simple to understand.

b) They occur less frequently than data hazards.



The impact of the pipeline on the branch instruction.

Reducing the Delay of Branches.

→ One method to improve branch performance is to reduce the cost of the taken branch.

→ The MIPS architecture was designed to support fast single cycle branches that could be pipelined with a small branch penalty.

→ The designers observed that many branches rely only on simple tests (equality or sign for example) and that such tests do not require a full ALU operation but can be done with at most a few gates.

→ When a more complex branch decision is required a separate instruction that uses an ALU to perform a comparison is required.

→ Moving the branch decision up requires two actions to occur earlier.

(a) Computing the branch target address.

(b) Evaluating the branch decision

Reducing Pipeline Branch Penalties:

→ There are many methods for dealing with the pipeline stalls caused by branch delay. There are two important schemes for reducing pipeline branch penalties.

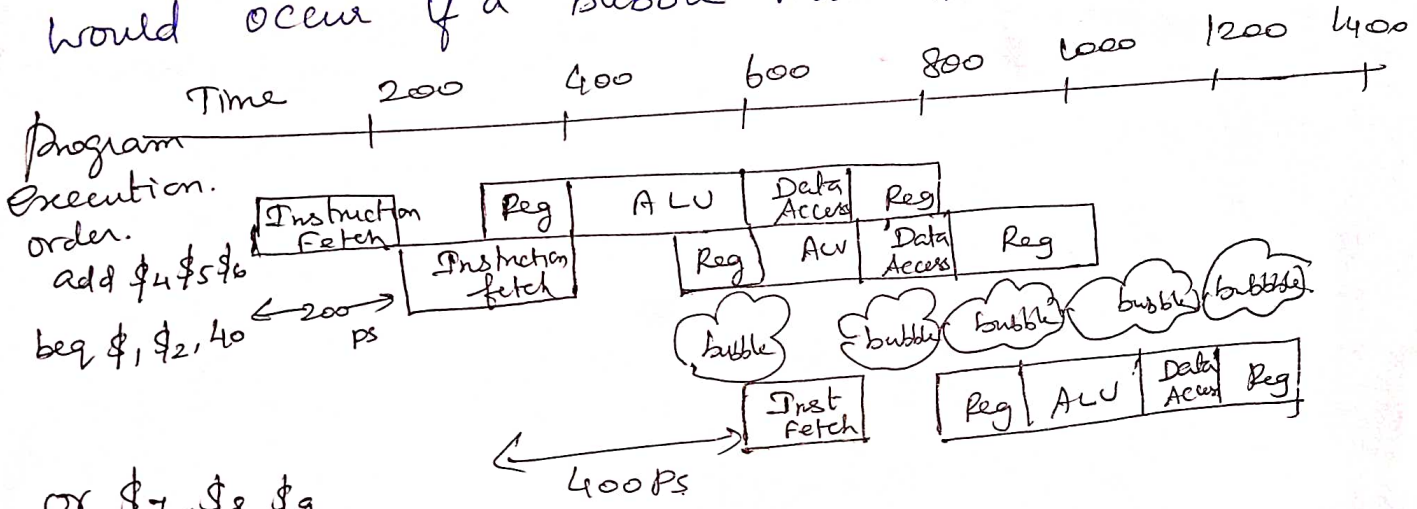
→ The first scheme is the static branch Prediction In this case branches are fixed for each branch during the entire execution. The software can try to minimize the branch penalty using knowledge of the hardware scheme and of branch behavior.

Solution to Control Hazards.

→ Control hazard also called branch hazard, arises from the need to make a decision based on the results of one instruction while others are executing.

When the Proper instruction Cannot execute in the Proper pipeline clock cycle because the instruction that was fetched is not the one that is needed.

Example. This example assumes the Conditional branch is taken and the instruction at the destination of the branch is the OR instruction. There is a one stage pipeline stall, or bubble, after the branch. The effect on performance, however is the same as would occur if a bubble were inserted.



Solution for Control hazard

1. Branch Prediction
2. Delayed slot.

UNIT-IV

Memory and I/O Organization.

Introduction

→ Memories are made up of registers. Each register in the memory is one storage location also called memory location

→ Each memory location is identified by an address. The number of storage locations can vary from a few in some memories to hundred of thousand in others.

→ Each registers consists of storage elements (flipflops or capacitors in semiconductor memories and magnetic domain in magnetic storage). each of which stores one bit of data. A storage element is called a cell.

Characteristics of memory

location → CPU, Internal (main) External (secondary).

Capacity → word size, Number of words.

Unit of Transfer → word, Block.

Access Method → Sequential access, Direct access, Random access, Associative access.

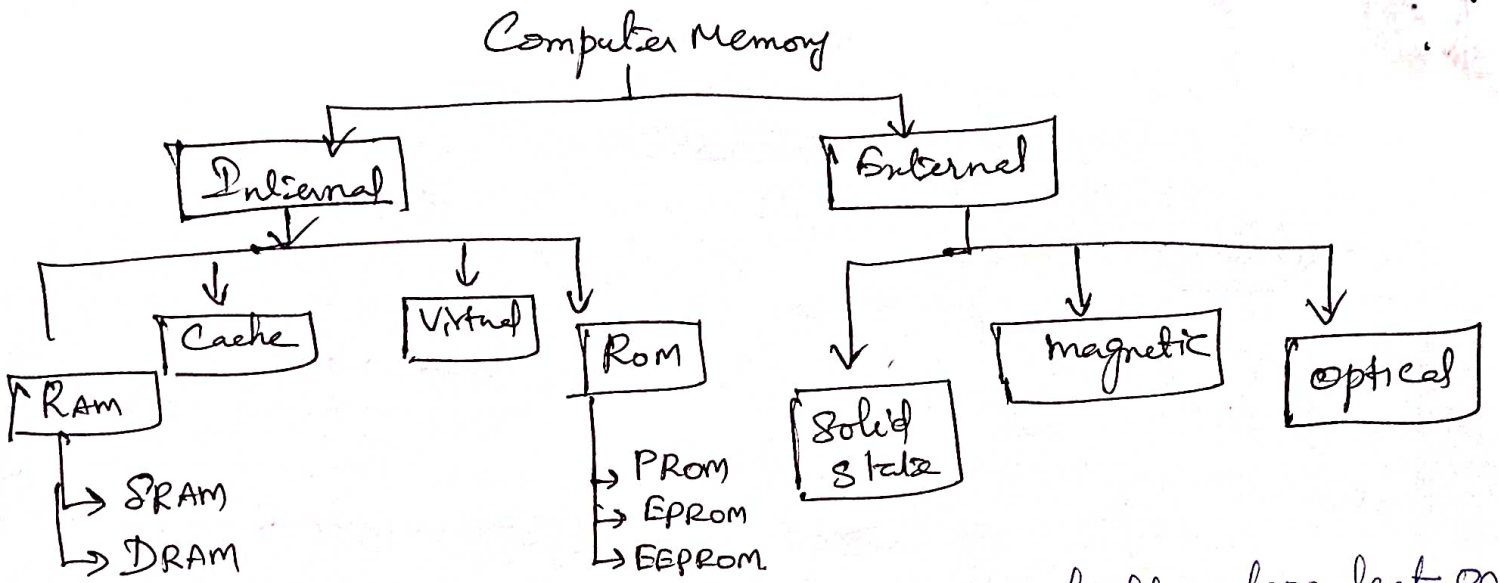
Performance → Access time, Cycle time, Transfer rate

Physical Type → Semiconductor, magnetic surface.

Physical characteristics → Volatile/Nonvoletile.

→. Compulier employs different types of memory. basically it have two types.

1. Internal 2. External Memory.



→ Execution Speed of Programs is highly dependent on the Speed with which instructions and data can be transfer between processor and the memory.

→ Memory would be fast, large and inexpensive. But is impossible to meet all the three requirements Simultaneously.

→ Increased speed and size increased the cost, this Problem is solved by developing another level Structure

→ This can be achieved by Speed → cache.
Virtual memory → Increase apparent size of memory.
Secondary Memory → Produces larger Storage Capacity.

Memory Hierarchy.

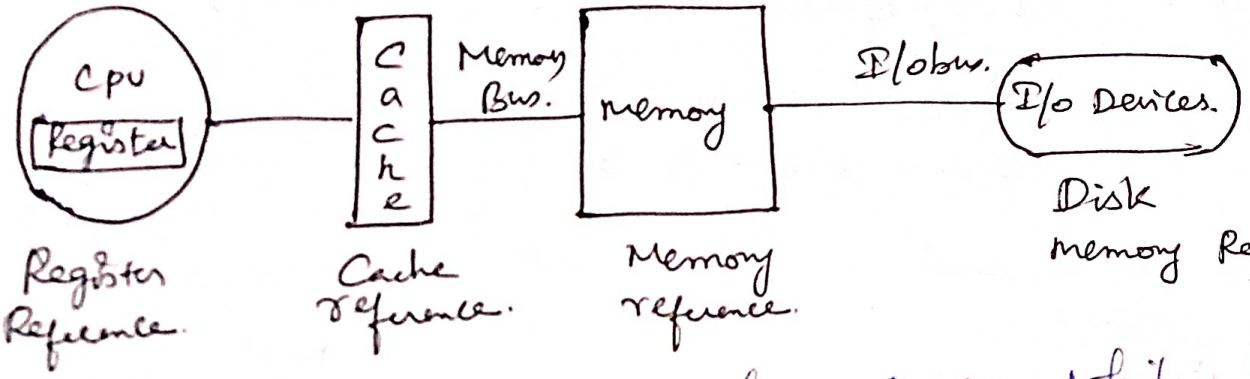
→ An economical solution to that desire is a memory hierarchy which takes advantages of locality and cost performance of memory technologies.

→ Principle of locality States that Program access a relatively small portion of their address space at any instant of time.

→ A memory hierarchy Consists of multiple levels of memory with different speed and sizes.

Faster memories are more expensive per bit than the slower memories.

→ There are three Primary technologies Used in building memory hierarchy..



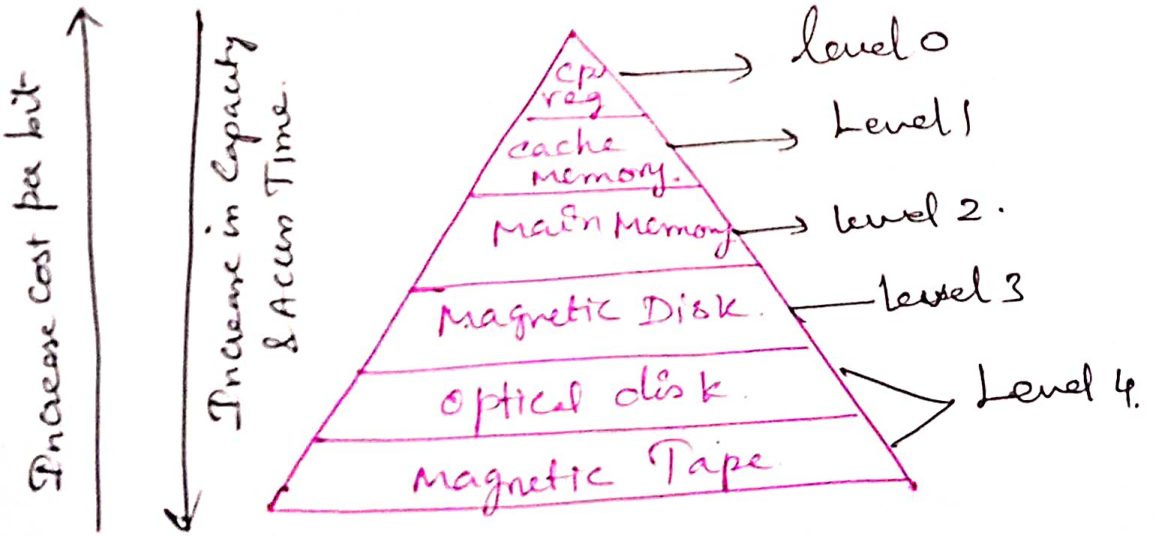
Main Memory is Implemented from DRAM while levels closer to the Processor (Caches) use SRAM (Static Random Access Memory).

→ DRAM is less costly per bit than SRAM.

→ DRAM Uses significantly less area per bit of memory.

and DRAM.

→ Third technology used to Implement the largest and slowest level in the hierarchy is magnetic disk.



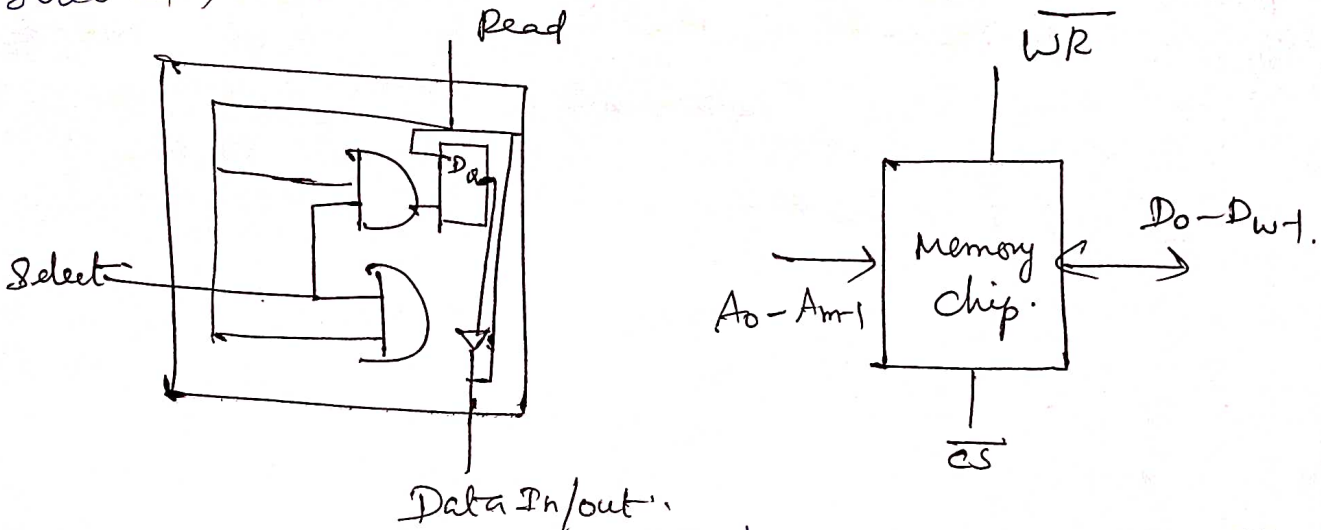
Memory Hierarchy Design.

Memory chip Organization.

RAM → Random Access Memory.

→ Term Random means that any memory location can be accessed in the same amount of time.

→ fig represent the memory elements as D flip flop with additional control to allow the cell to be selected, read and written.



→ This topic mainly focus on how RAM cells are organized in to chips and how this organization implements the random access property.

→ m bit address, having lines numbered from 0 to m-1 is applied to pin A0 - Am-1 while asserting chip select (CS) and either WR (for reading data from the chip).

→ The over bar on CS and WR indicates that the chip is selected when CS=0 and that write operation will occur when WR=0.

→ When reading data from the chip after time period t_{AA} the W bit data word appears on data line D0-Dw-1

→ Consider that a 64 M x 1 chip has 26 address lines ($C64M = 2^{26}$). This means that a Conventional decoder would need 2^{26}, 26 input and AND gates.

Mapping function.

A particular block of main memory can be brought to a particular block of cache memory.

→ Mapping functions are used to map a particular block of main memory to a particular block of cache.

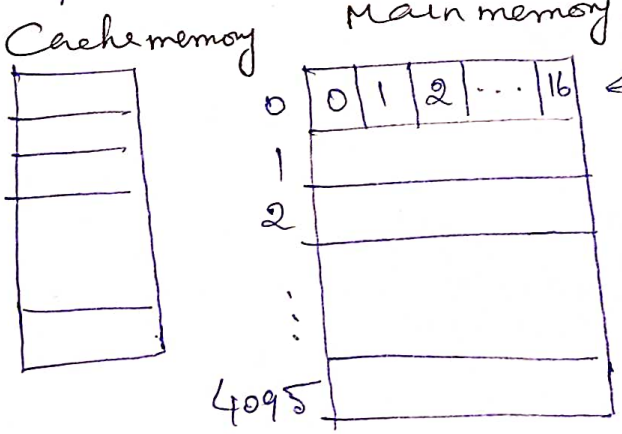
→ This mapping functions is used to transfer the block from main memory to cache memory.

Three types of mapping function.

1. Direct mapping.
2. Associative mapping
3. Set associative mapping.

Direct mapping

A particular block of main memory can be brought to a particular block of cache memory.

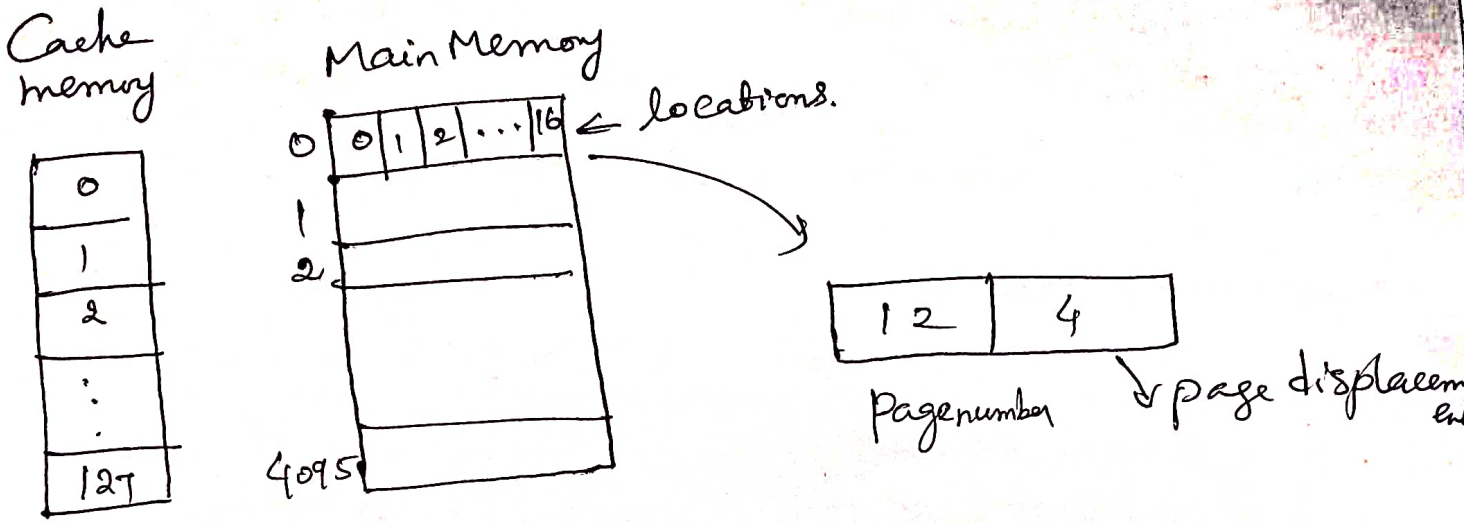


4096 x 16.

$= 65536 = 2^{(16)}$

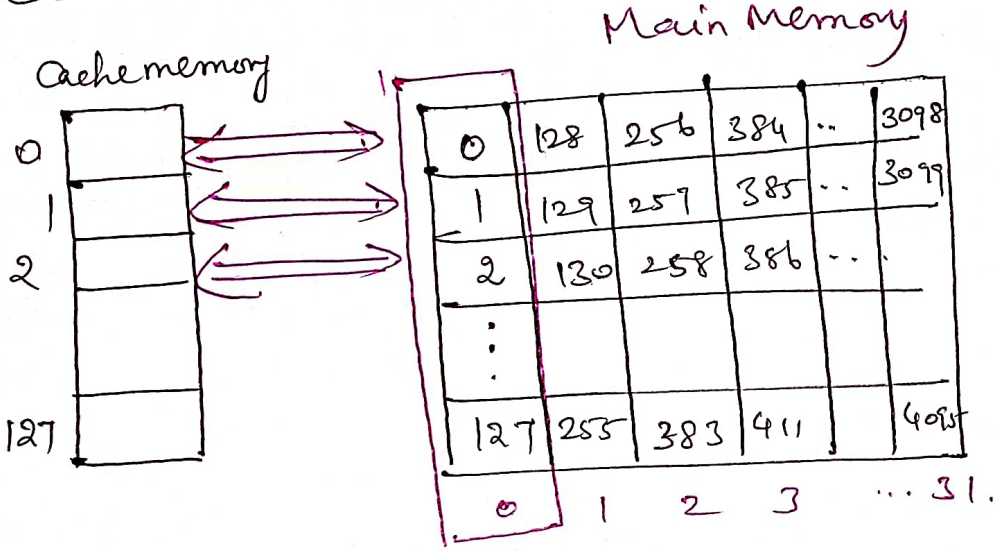
P = 4096

each page have 16 location



C = 128 P = 4096.

Each Cache frame size equal to each block size of Main memory.

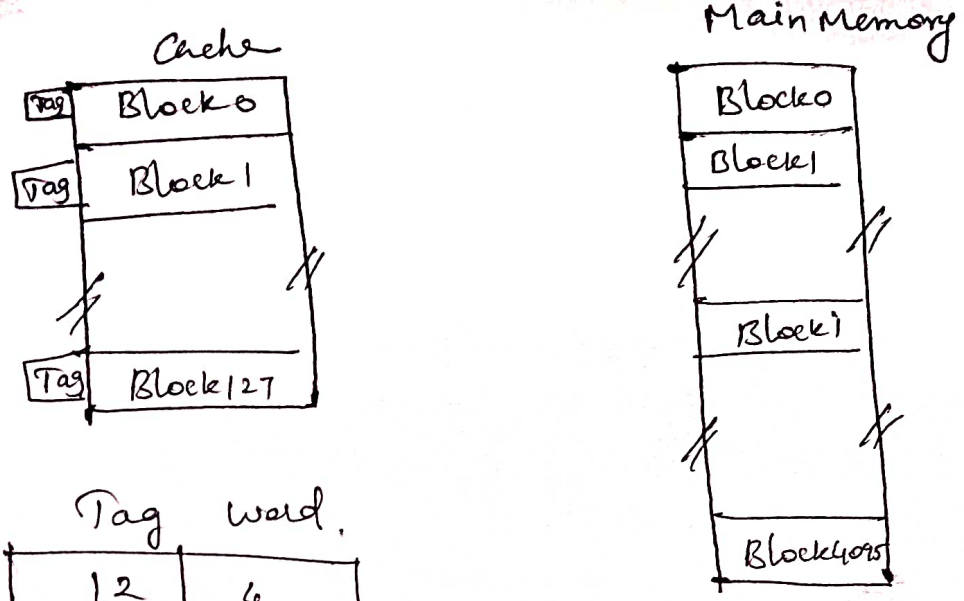


Associative Mapping

→ In this mapping function any block of main memory Can potentially reside in any Cache block position.

→ This is much more flexible mapping method

→ 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 present. This is Called the associative mapping technique.



Associative mapped Cache.

Advantage. → Space in the cache can be Used efficiently

→ it gives full freedom in choosing a cache location to load memory block.

disadvantage.

→ cost of associative mapping cache is higher than Direct mapping Cache.

Set associative mapping

→ This mapping is a combination of the direct mapping and associative mapping Techniques.

→ Blocks of the cache are grouped into sets and mapping allows a block of the main memory to reside in any block of a specific set.

Virtual Memory Organization

→ The term Virtual memory refers to something which appears to be present but actually it is not

→ The virtual memory technique allows User to use more memory for a program than real memory of a Computer.

→ So Virtual memory is the Concept that gives the illusion to the User that they will have main memory equal to the capacity of secondary storage media.

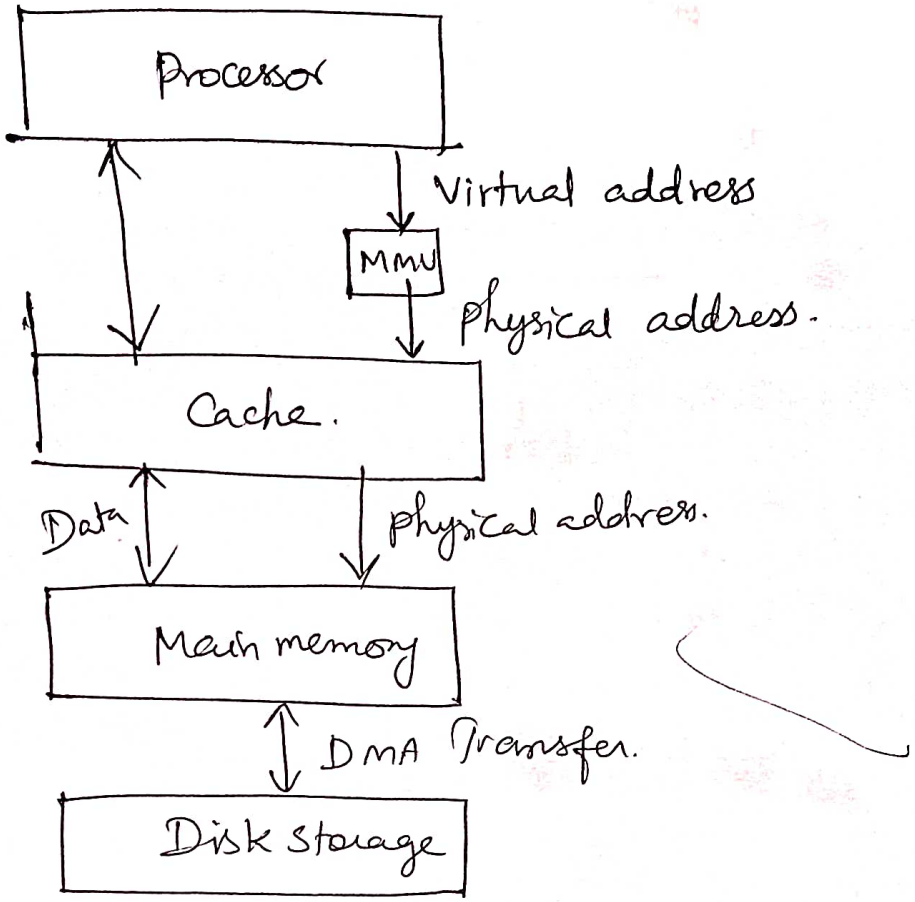
Need of Virtual memory

→ The program is stored in the secondary memory the memory management Unit (MMU) transfer the currently needed part of the program from the secondary memory to the main memory for execution

→ The segments which are currently being executed are kept in the main memory and remaining segments are stored in the secondary storage devices. such as magnetic disk.

→ If an executing program needs a segment which is not currently in the main memory. the required segment is copied from the secondary storage device.

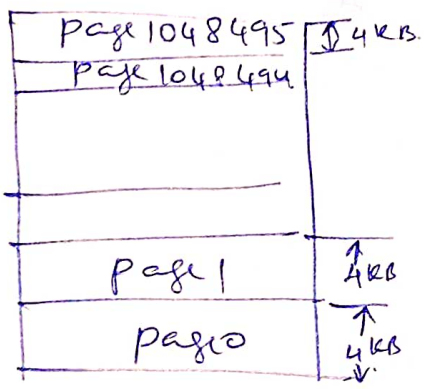
→ In modern Computers. the operating system moves program and data automatically b/w main memory and secondary storage.



Virtual memory Organization.

Paging: The memory Management Unit (MMU) Controls the virtual memory system. It translates virtual address into physical addresses.

→ A simple method for translating virtual address into physical addresses is to assume that all programs and data are composed of fixed length Unit called pages.



Bus It is a Common path way that Connects a number of devices.

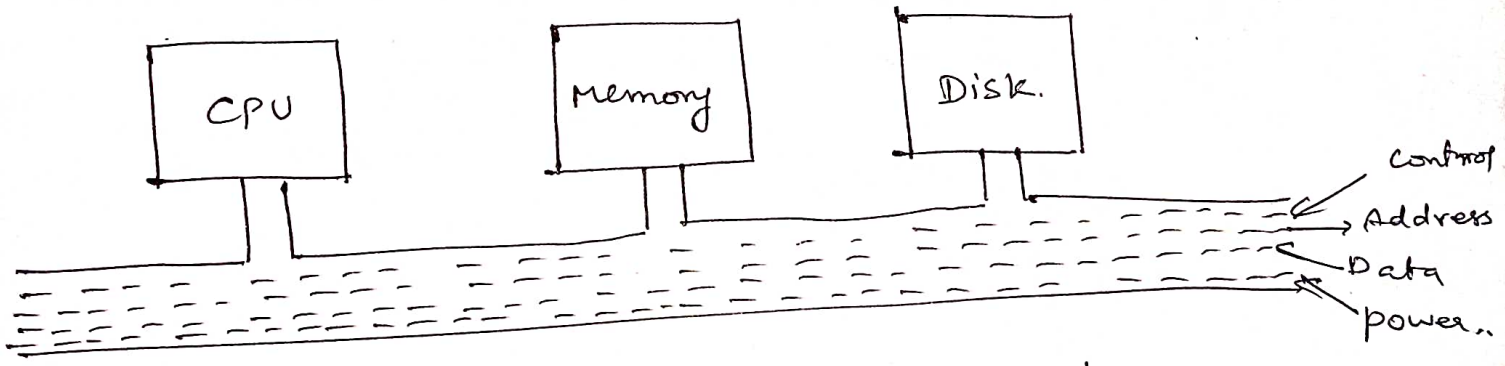
→ example of a bus can be found on the mothe boa on the personal Computer.

→ motherboard Contains Integrated IC such as cpu chips and memory chip board traces (wires) tha Connect the chip, a number of buses for chip and devices that need to Communicate with each other and bridges that Connect the buses.

Bus structure, Protocol and Control

→ A bus Consist of the physical parts like Connectors. and wires. and a bus protocol.

→ The wires Can be Partitioned into Separate groups. of Control, address data and power



Simplified illustration of a bus.

→ Single bus has few power lines such as GND at 0V. Positive voltage at +3.3V. positive and negative voltages at +5V and −5V.

→ All devices. Simultaneously listen but normally only One device receives.

Address Translation

→ In virtual memory the address is broken into a virtual page number and page offset.

→ A simple method for translating virtual addresses into physical addresses is to assume that all program and data are composed of fixed length units called pages.

→ Pages commonly range from 2k to 16k. bytes in length.

Parallel Bus architecture

A Computer System may Contain many Components that need to Communicate with each other In worst Case Scenario. all N Components (CPU, graphics processor, CD ROM, magnetic disk. etc) may need to simultaneously Communicate with every other Components in which $N \times (N-1)/2$. links are needed for N. Components.

Example: Just N=10. Components. CPU, graphics processor, a few memory modules, a few network Interface Cards CDROM) with a 64 bit data path.

→ The number of wires needed to support this is $\frac{10 \times 9}{2} \times 64 = 2880$. which is enormous Degree of Complexity.

→ Only one device can be bus master at any given time and the remaining devices are slaves

Advantages of Bus

To eliminate the need for connecting every device with every other device. Which avoids the wiring complexity.

disadvantages

① Slowdown introduced by the master/slave Configuration.

Types of Bus

A bus can be classified into two types.

(a) Synchronous Bus
(b) Asynchronous Bus.

Internal Communication Methodologies

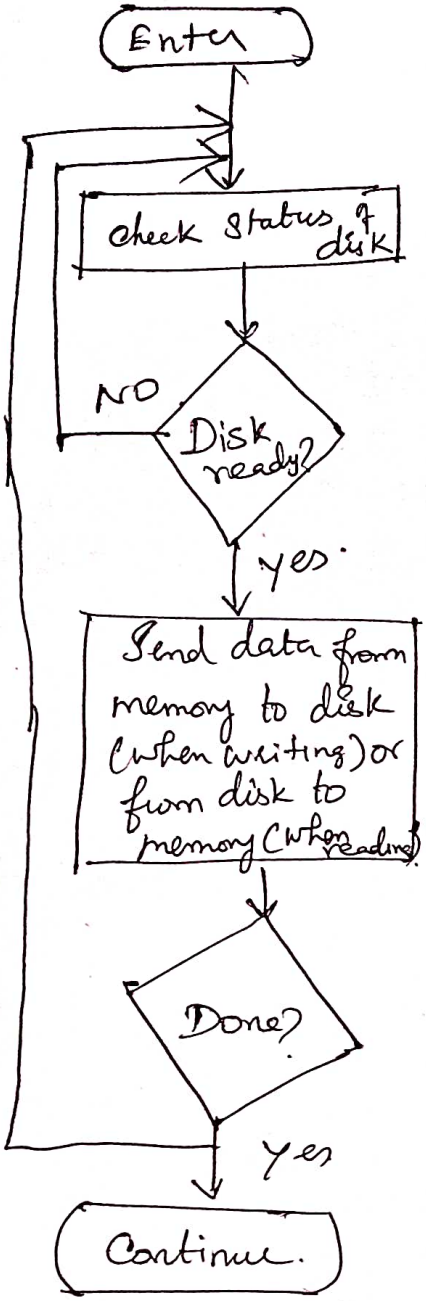
→ Computer systems have a wide range of Communication tasks.

→ The cpu must Communicate with memory and I/O devices such as keyboard, disk drives and network interfaces. There are three different types of data transfer schemes are used.

(i) Programmed I/o. (ii) Interrupt I/o. (iii) Direct Memory Access. (DMA).

Programmed I/o (polling)

→ Consider reading a block of data from a disk. In Programmed I/o, the cpu polls each device to see if it needs servicing.



→ CPU first checks the status of disk.

→ If the disk is not ready to be read or written, then the process loops back and checks the status continuously until the disk is ready.

→ When the disk is finally ready, then a transfer of data is made between the disk and CPU.

Advantages

→ Simple to Implement
→ Very little hardware support.

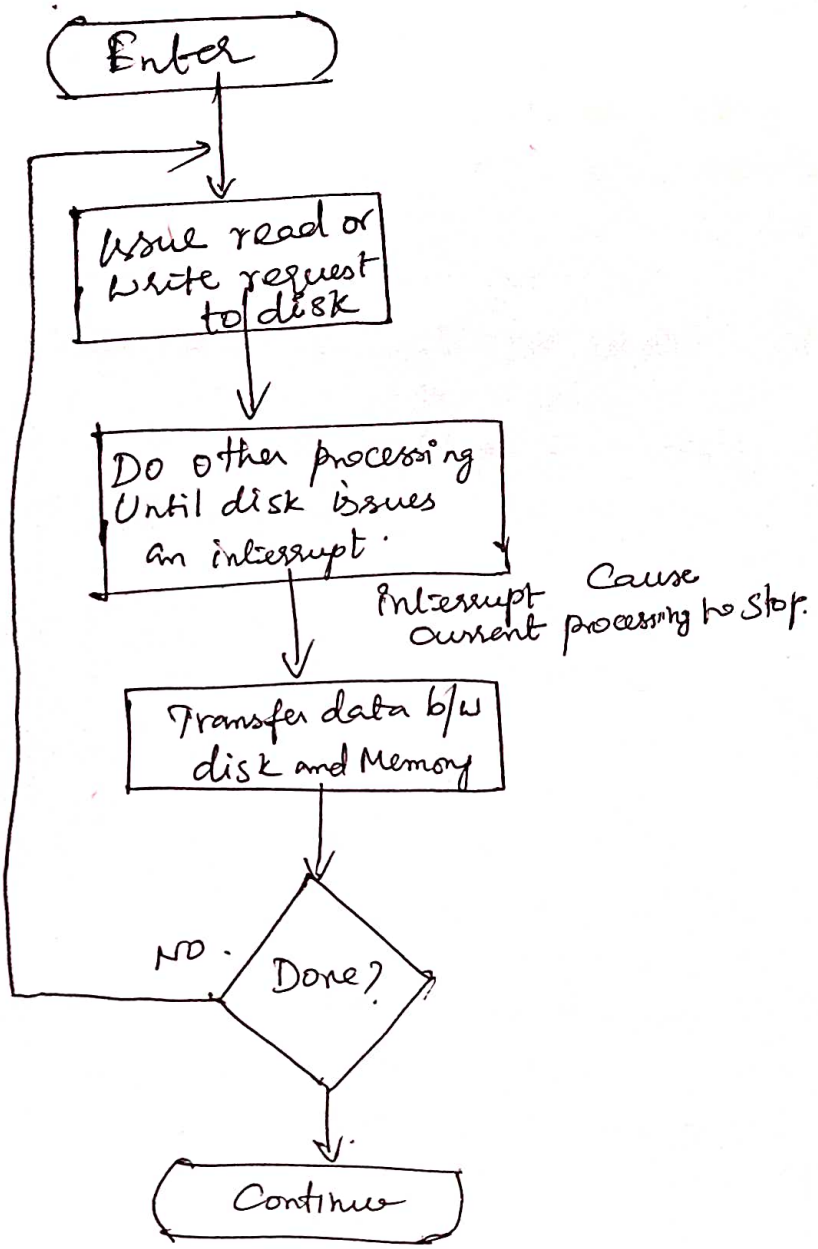
Disadvantages

- CPU is kept busy for checking the status of the slower I/O device.
busy waiting.

Interrupt Driven I/O

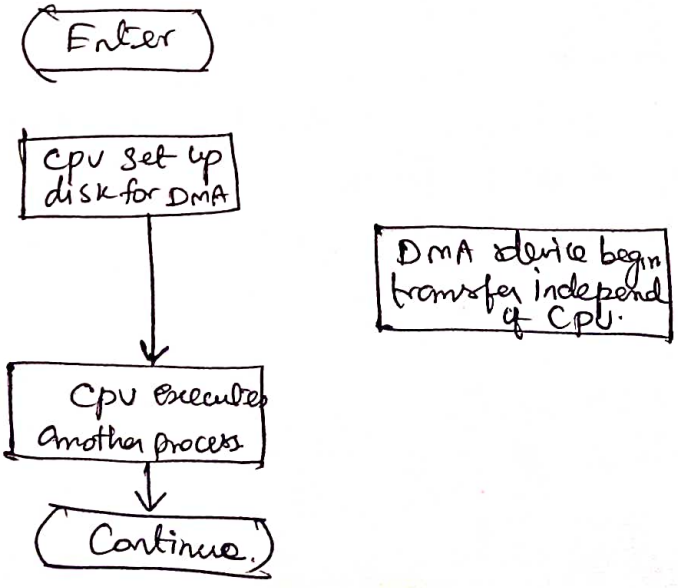
→ With Interrupt driven I/O the CPU does not access a device until it needs servicing and so it does not get caught up in busy wait.

→ In Interrupt driven I/O, the device request service through a special interrupt request line that goes directly to the CPU.



Direct Memory Access

→ Interrupt driven I/O frees the CPU Until the device requires services. the CPU is still responsible for making actual data transfer.



→ A direct memory Acces (DMA) device can transfer data directly to and from memory rather than using the cpu as an intermediary and can thus relieve congestion on the system bus.

→ DMA services are usually provided by a DMA controller, which is itself a specialized processor whose specialty is transferring data directly to or from I/O devices and memory.

Serial Bus architecture

→ Serial communication (bit by bit) have significant advantages over parallel communication.

→ Serial interconnects requires higher bit rates to compensate for the lack of parallelism.

→ it consumes less power, less space, and cost less money.

Serial Communication Approaches

RS232

→ The first serial interconnection method is not a bus at all. but a simple point to point interconnect known as recommended standard 232C. or RS-232

→ Rs 232 standard commonly uses 9 pins and 25 pin connectors.

→ Pin 7 of 25 pin connector is for signal ground pin 2 send data and pin 3 receives data.

Universal Serial bus

→ USB and IEEE 1394 (fire wire) are two group of Standard for Interconnecting peripheral devices.

→ It initially Supported data transfer rates of 12 Mbps with as many as 127 devices Connected to a Single Host Controller through special hub devices.

→ The Standard has evolved to USB 2.0. which Supports data rates upto 480 Mbps. USB is typically used for mice, keyboard, modems and other devices.

Mass Storage.

→ Registers and RAM requires Continuous power to retain their Stored data, whereas media Such as magnetic taps and magnetic disks retain information indefinitely after the power is removed. which is known as indefinite Pasistence.

→ This type of Storage is Said to be non-volatile.

Magnetic Disks

→ A magnetic disks is a device Storing Information that Supports a large Storage density and a relatively fast access time.

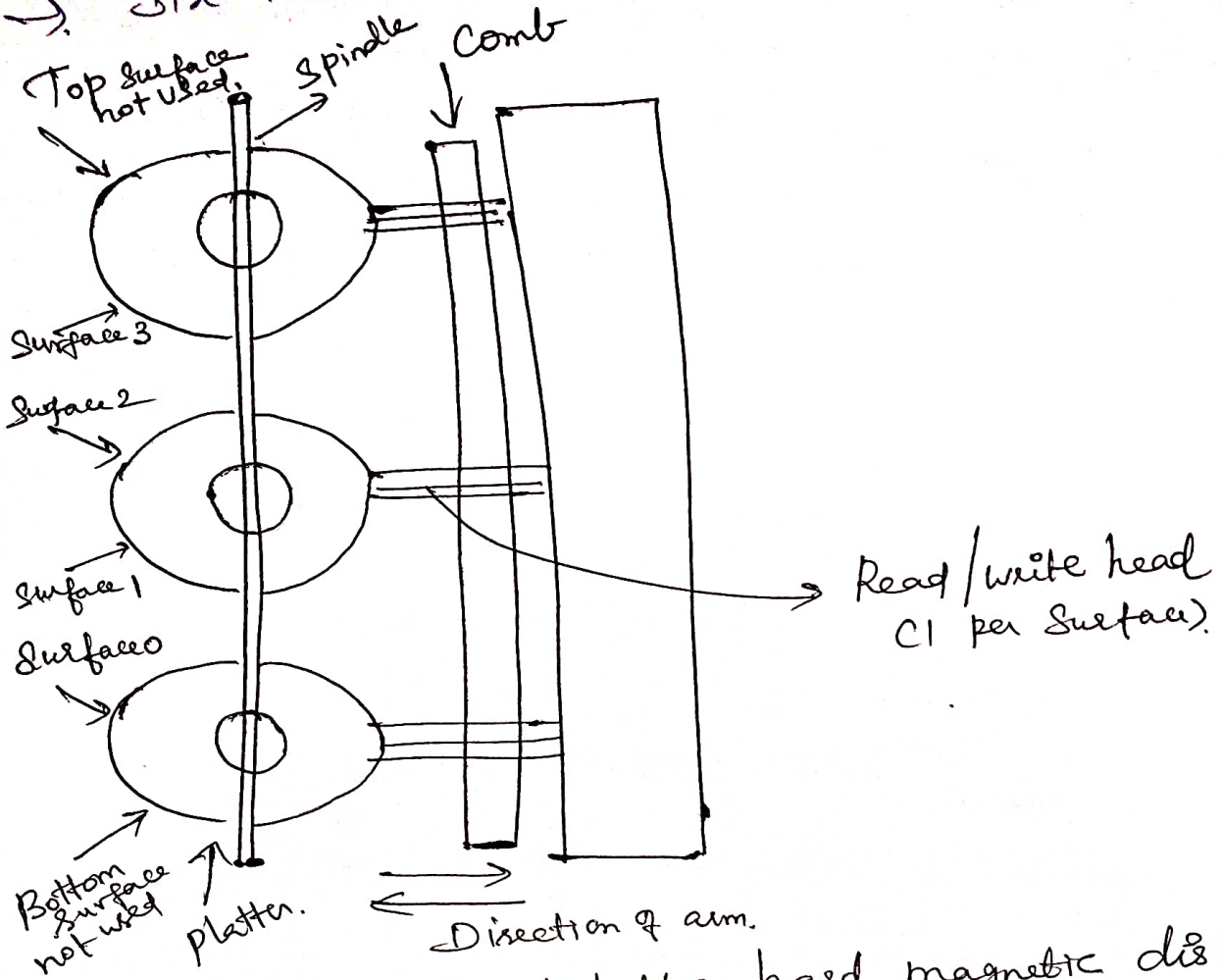
→ A moving head magnetic disk is Composed of a Stack of one or more platters that are spaced close together and are connected via Spindle.

→ The most common Size for desktop personal Computer disks is 3.5" diameter

→ laptop hard disk drives use 2.5" diameter or Smaller

→ A single head is dedicated to each surface.

→ Six heads are used in the example shown.



Multiplatter hard magnetic disk.

→ The heads are attached to a common arm (also known as comb) that moves in and out to reach different portions of the surfaces.

→ In a hard disk drive, the platters rotate at a constant speed of typically 5400 to 1500 magnetic revolutions per minute (RPM).

Magnetic Tape:

→ The magnetic tape unit typically has a single read/write head, but may have separate heads for reading and writing.

Input devices.

Keyboard :→ A keyboard is basically a board of keys Keyboard is one of the Primary input devices used with a computer.

→. This keyboard layout is known as "QWERTY" layout, which gets its name from the first six letters across in the upper left hand corner of the keyboard.

→. A keyboard layout using ECMA-23 Standard is shown.

→. When a character is typed, a bit pattern is created that is transmitted to a host Computer.

→. ECMA-23 keyboards include additional of function keys (in row F, for example). and the addition of special keys such as tab, delete and Carriage return.

Tablets

→ A digitizing tablet is an input device that consists of a flat surface and either a pen based stylus or puck.

→. Tablet has an embedded two dimensional mesh of wires that detects an induced current created by the puck as it is moved about the tablet.

→ The tablet transmits x-y (horizontal-vertical) positions and the state of the button on the puck (or stylus) either continuously, or for an event such a key click or a element. depending on the control method.

Touch Sensitive Pen-Based display.

Unlike a tablet which Uses an active matrix that Senses an electromagnetic field induced by a Stylus or Puck. Pen based Personal digital assistants (PDA) Uses a Passive matrix in which. the pen can be anything that induces Pressure on the Screen.

→ Two transparent Conducting layers are placed on the Screen. separated by a Spacer dots. When the User applies Pressure to the top layer as with a Stylus or Simply a finger the top and bottom layers make Contact. The Induced voltage at the edges Varies according to the Position of the Stylus.

Joysticks → A joystick indicates horizontal and Vertical Position by the distance a rod that Point from the base is moved.

→ joysticks are Commonly Used in Video and for indicating Position in graphics Systems.

Output devices

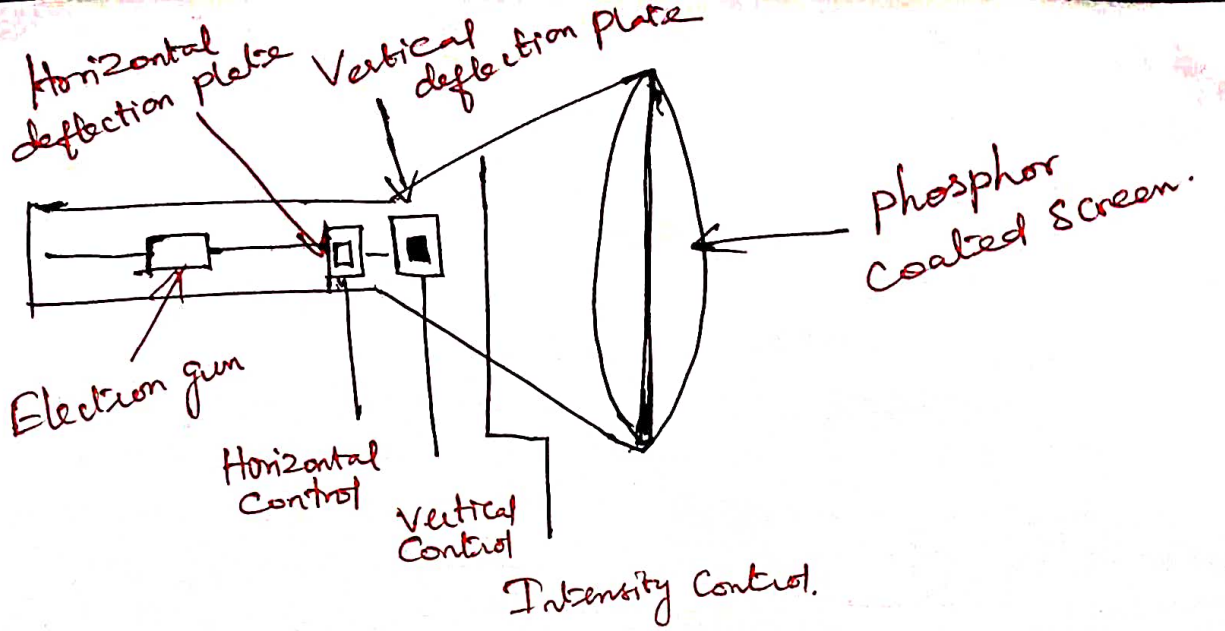
Three Common Output devices : laser Printer, Video display and LCD Panel.

Laser Printer. → A Laser Printer Consists of a charged drum in which a laser discharges Selected areas according to a bit-mapped representation of a page to be Printed.

→ As the drum advances for each scan line the charged areas pick up electrostatically sensitive toner powder.
→ The drum continues to advance and toner is transferred to the paper which is heated to fix the toner on the page.

Video displays

→ The display devices are known as O/p devices. The most commonly used output devices in a graphics video monitor.
→ The operations of most video monitors are based on the standard cathod ray tube (CRT).
→ A video display, or monitor consists of a luminescent display device such as a cathode ray tube. (CRT).
→ In a CRT, vertical and horizontal deflection plates steer an electron beam that sweeps the display screen in a raster fashion.
→. A configuration for a CRT is shown in fig. An electron gun generates a stream of electrons that is imaged onto a phosphor-coated screen at positions controlled by voltages on the vertical and horizontal deflection plates. Electrons are negatively charged and so a positive voltage on the grid accelerates electrons towards the screen and negative voltage repels electrons away from the screen.



CRT with a single electron gun.

→ The colour Produced on the Screen is determined by the characteristics of the phosphor.

→ For a colour CRT there are three different phosphor types (red, green, blue) that are interleaved in a regular pattern, and three guns. which Produce three beams that are Simultaneously deflected.

Liquid Crystal Displays (LCD's)

→ We always use devices made up of Liquid Crystal Displays (LCD) like Computers. digital watches. also DVD and CD players.

→ Liquid Crystal Displays (LCD) have emerged as a Prevalent choice of computer monitors. LCDs are Small and light weight when compared with tube monitors, Consume less power and dissipate less heat and are more shock resistant.

Unit 5

Advanced Computer Architecture

One method of improving the performance of a Processor is to increase the clock rate. This will work upto a point because higher the clock rate consumes more power

→ An alternate approach is to increase the number of processors and decompose and distribute a single program onto the processors. This approach is known as Parallel Processing. In parallel processing a number of processors work collectively in parallel on a common problem.

Parallel Processing architectures and challenges.

At first the performance metrics for parallel processing is explained followed by parallel processing challenges and then specific types of parallel architectures are discussed.

Measuring Performance

A parallel architecture can be characterized in terms of three parameters.

1. The number of processing elements.
2. The interconnection networks among the PEs.
3. The organization of the memory.

→ **Parallel time** . It is the absolute time needed for a program to execute on a parallel processor.

→ **Speed up** It is the ratio of the time for a program to execute on a sequential processor to the time for that same program to execute on a parallel processor.

③ **Efficiency** It is the ratio of speed up to the number of Processors used.

④ **Throughput** It is a measure of how much Computation is achieved over time.

Parallel Processing Challenges.

High Performance can mean high throughput for independent task. Utilizing multiple Processors by running independent Programs simultaneously is called as task level parallelism. or Process level parallelism.

→ A single Program that runs on multiple Processors simultaneously is called as Parallel Processing Program.

→ Some of the challenges of Parallel Processing are

* Parallel Programming
* Complex algorithm.
* Memory issues.
* Communication costs.
* Load Bbalancing.

→. Some Problems doa not have work efficient Parallel. algorithms that exhibit massive parallblism.

That is we simply donot know how to solve these Problems with a large number of parallel execution Units without significantly increasing the Computation Complexity.

→ Some Problems have known parallel algorithms with sufficient parallelism. but questionable numerical stability.

→ Some applications that have parallel algorithms are Overwhelmed by Catastrophic load imbalance due to highly non Uniform data distribution.

Hardware Multi-threading.

→ Multithreading is first introduced to exploit thread level parallelism within a processor. Hardware multithreading allows multiple threads to share the functional units of a single processor in an overlapping fashion.

→ It tries to utilize the hardware resources efficiently To allow this sharing, the processor must duplicate the independent state of each thread.

→ Hardware multithreading increases the utilization of a processor by switching to another thread when one thread is stalled. Thread is a light weight process.

→ All multithreading approaches use similar resources partioning system. Although there are similarities in the multithreading implementations, there are still some difference Two major differences are.

(1) Thread scheduling policy.
(2) Pipeline partioning.

→ Based on above criteria there are three main approaches to multithreading.

(1) Fine grained multithreading (FGMT)
(2) Coarse grained Multithreading [CGMT]
(3) Simultaneous multithreading [SMT].

Fine-grained Multi threading.

→ Fine grained multithreading switches between threads after every instruction. It makes interleaved execution of multiple threads at the same time. This interleaving is done in round robin fashion on every clock cycle.

advantage → It hides throughput losses due to short and long stalls.
→ Instructions from other threads can be executed when one thread stalls.

Disadvantage
→ It slows down the execution of individual threads.
→ thread that is ready to execute without stalls will be delayed by instructions from other threads.

Coarse-Grained Multithreading

Coarse grained multithreading is an alternative to fine grained multithreading. Coarse-grained multithreading won't switch out the executing thread until it reaches a situation that triggers a switch. This situation occurs when the instruction execution reaches either a long latency operation or an explicit additional switch operation.

advantages → Faster, Reduces the penalty of high cost stalls
→ Less likely slow down the execution of an individual thread.

Disadvantage → Difficult to overcome throughput losses due to shorter stalls.
→ When a stall occurs the pipeline must be emptied.

Simultaneous Multithreading (SMT)

→ Hardware multithreading that uses the resources of multiple issues, dynamically scheduled pipeline processor is called Simultaneous Multithreading. Here both thread level and instruction level parallelisms are used.

Advantage → More functional unit parallelism available than single thread.
→ Register renaming and dynamic scheduling are used.
→ It does not switch resources for every cycle.

Multicore and Shared Memory Multiprocessors

Hardware multi-threading improved the efficiency of processors at modest cost, but the difficulty is how to run old programs in parallel hardware.

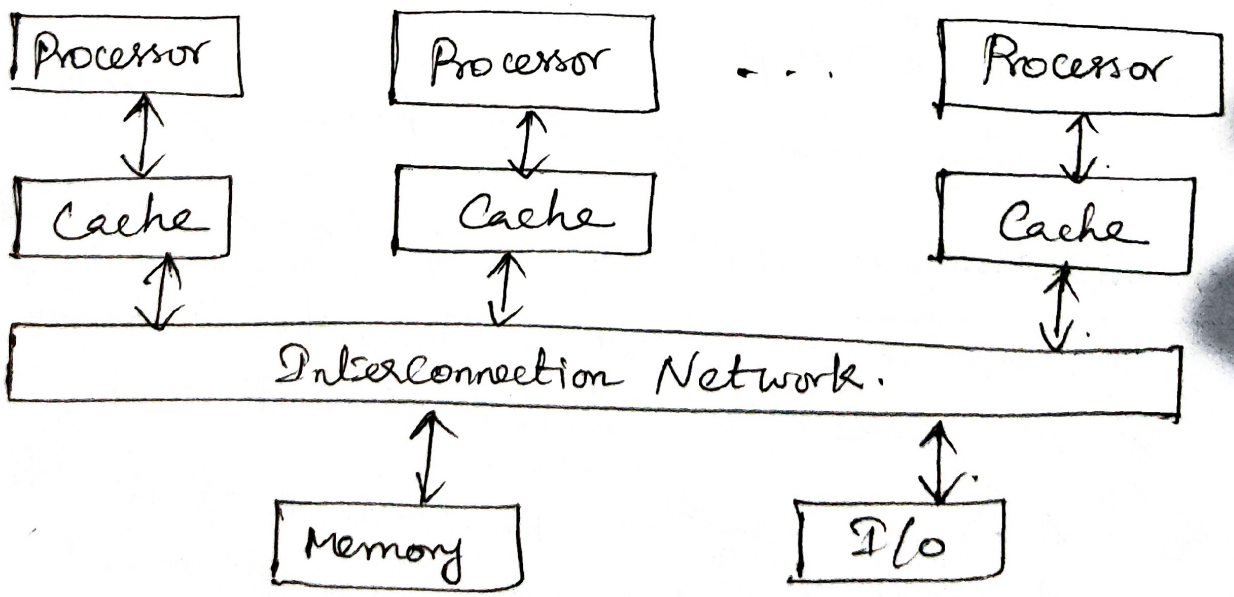
There are two solutions:

(i) Shared Memory Multiprocessor (SMP) Provides single address space to all processors. In this case, Program need not to Concern about where they are run. they may be executed in parallel. In this approach, all variable of the program must be made available at any time to any processor.

(ii) clusters and other message passing multiprocessors. Separate address space is reserved for each processor. here Sharing must be Explicit.

Shared Memory Multiprocessor.

→



Organization of Shared memory multiprocessor

→ A parallel processor with single physical address space is called Shared Memory Multiprocessor.

→ Processors communicate through shared variable in memory. All the processors can access any memory location via loads and stores. fig shows the organization of SMP.

→ There are two types of single address space multiprocessors.

(i) Uniform Memory Access. (UMA)

Latency to a word in memory does not depend on which processor asks for it. That is the time taken to access a word is uniform for all processors.

(ii) Non Uniform Memory Access. (NUMA)

→ Some memory access are much faster than others depending on which processor asks for which word.

→ Here the main memory is divided and attached to different microprocessors or to different memory controllers on the same chip.

→ Programming challenges are harder for NUMA than UMA

Synchronization. The process of co-ordinating the behavior of two or more processes which may be running on different processors are called Synchronization.

Lock A synchronization mechanism that allows access to data to only one processor at a time is called lock. Other processor must wait until the original processor unlocks the variable.

Introduction to Graphics Processing Units

→. The main reason for Improving graphics Processing was the compulier game industry. The fast growing game market encouraged many companies to make increasing investments in developing faster graphics hardware.

→ The graphics Processing Improves at a faster rate than general purpose Processing in mainstream microprocessors. As the graphics Processors increased in power they earned the name Graphics Processing Unit or GPU to distinguish themselves from CPUs

→ Anyone can buy a GPU, today which hundreds of Parallel floating Point Units at reasonable cost. It makes high performance Computing more accessible.

GPU Vs CPU

There are afew key Cheracteristics as to how GPU vary from CPU's

1. GPUs are accelerators that Supplement a CPU, So they need be able to perform all the tasks of a CPU. This role allows them to dedicate all their resources to graphics. do not

2. The GPU Problems sizes are typically hundred of megabytes to giga bytes where as The CPU Problem Size are ranging from hundreds of gigabytes to terabytes.

An Introduction to the NVIDIA GPU Architecture.

→ C inspired Programming languages are used to write Programs directly for the GPU's.

→ Thousands of CUDA Threads are Combined together to utilize the various Styles of Parallelism within a GPU Multithreading, MIMD, SIMD and instruction-level Parallelism These threads are blocked together and executed in groups of 32 at a time.

→ Multicore Computers with SIMD instruction extensions have similarities with GPU's

→ SIMD Processors are also similar to Vector Processors. The multiple SIMD processors in GPU act as independent MIMD Cores just as many Vector computer have multiple Vector Processors.

→ The biggest difference is multithreading which is fundamental to GPU and missing from most Vector Processors.

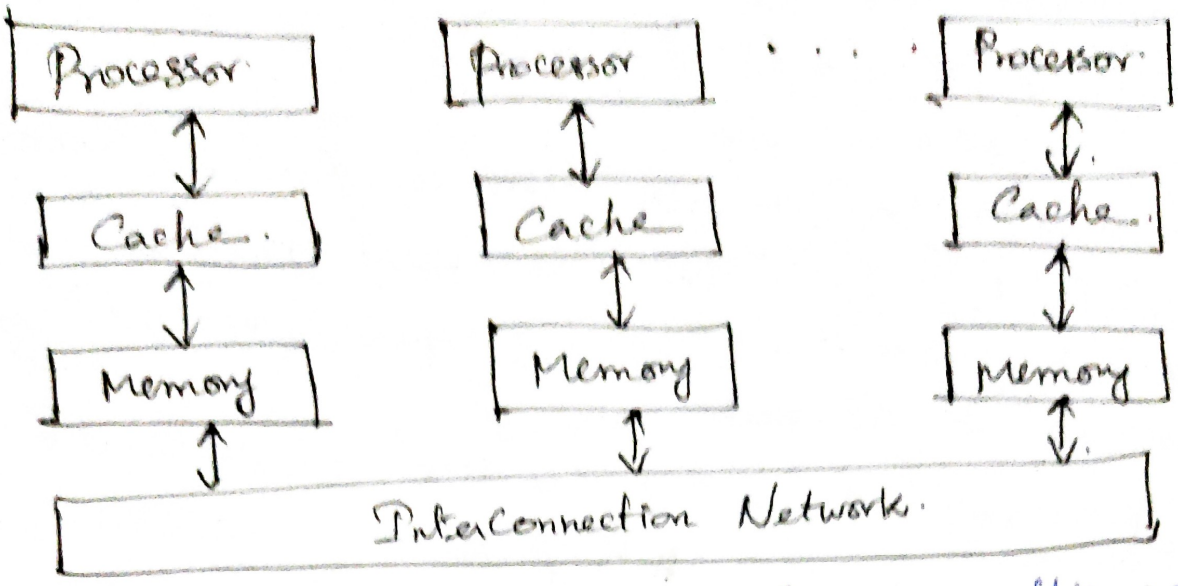
Clusters and Warehouse Scale Computers

cluster and other message Passing Multiprocessors.

→ An alternative approach to Sharing an address Space is for the Processors to have their own Private physical address Space. figure shows the organization of a multiprocessors with multiple Private address Space. In this each Processor have their own Private physical address Space.

→ In this approach the multiProcessor Communicate via Explicit message passing scheme, hence the name message passing multiprocessors, Communications b/w multiple Processors is done by explicitly Sending and receiving information.

→ There are routine to Send and receive message. Send message routine is Used to Send message to other Processors in machines with Private memories. Receive message routine is Used to receive messages from another Processors in machine with Private memories.



Organization of message passing multiprocessor.

Drawbacks of clusters.

① The cost of administering a cluster of n machine is about the same as the cost of administering n independent machine. But the cost of administering a shared memory multiprocessor with n processors is about the same as administering a single machine.

② The processors in a cluster are connected using the I/O interconnect of each computer. But the cores in a multiprocessor are usually connected on the memory interconnect of the computer.

③ The overhead in the division of memory, A cluster of n machine has n independent memories and n copies of the operating systems are needed.

Warehouse scale Computers Vs Servers.

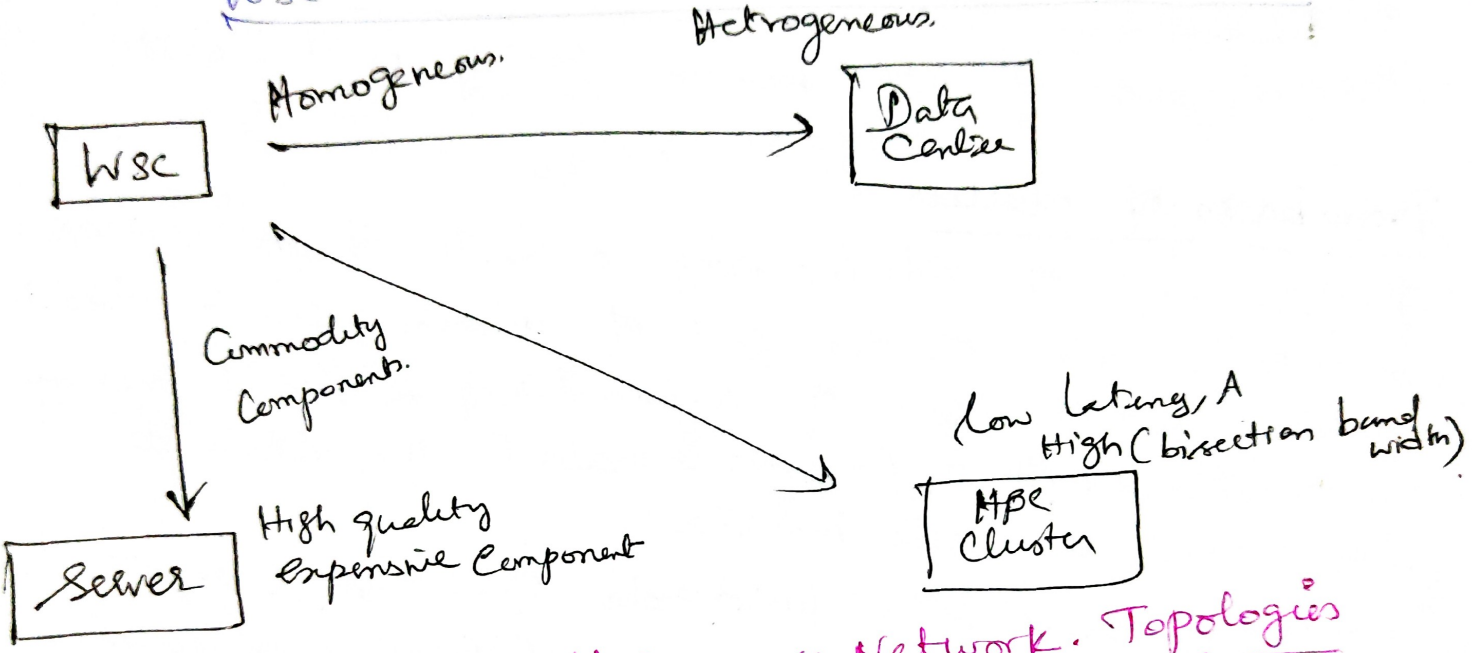
WSC have three major distinctions with servers.

1. More computational parallelism is not important
   → Most jobs are totally independent
   → Many independent jobs can proceed in parallel with little need for communication or synchronization

② Operational Cost Count:
→ WSC have longer lifetimes ; so that the Operational Costs increase.

③ Scale and the Opportunities / Problems Associated with Scale.
To Construct a single WSC, you must purchase 100,000 Servers along with the Supporting infrastructure, which means volume discounts.

WSC Vs Data Center Vs HPC. Cluster Vs Servica



Introduction to Multiprocessor Network Topologies

→ Multicore chips require on chip networks to connect cores together and clusters require local area networks to connect servers together. The performance metrics used to analyze the topologies are, network bandwidth and bisection bandwidth.

→ Network bandwidth is defined as the peak transfer rate of a network. It refers to the speed of a single link or the collective transfer rate of all links in the network.

→ Bisection bandwidth is defined as the bandwidth between two equal parts of a multiprocessor. This measure is for a worst case split of the multiprocessor.

→ There are number of different topologies available but only a few have been used in commercial parallel Processors.

Multistage network topologies

→ An alternative to placing a Processor at every node in a network is to leave only the switch at some of these nodes.

→ The switches are smaller than Processor-memory switch nodes and thus may be packed more densely there by lessening distance and increasing performance.

→ Such networks are frequently called multistage networks to reflect the multiple steps that a message may travel.

→ Simply a network that supplies a small switch at each node is called as multistage network.

→ A network that allows any node to communicate with any other node in one pass through the network is called as Cross bar network.

① Cross bar
* The crossbar uses n^2 switches where n is the number of Processors.
* Here Cross bar uses 64 switches versus 12 switch boxes.
* It supports any combination of message b/w Processors.

② Omega Network
* The Omega network uses $2n \log_2 n$ of the large switch boxes.
* Here Omega uses 48 switches.
* It uses less hardware than the cross bar network.
* It does not support any combination of messages between processors.