Page No.:

Computer Organisation:

It deals with the hardware involved in the computer. The hardware is of two types:

1. Internal Hardware

   Processor, mother board, daughter boards, RAM, ROM

2. External Hardware

   Keyboard, mouse, printer, display etc.

Architecture:

Architecture deals with the fundamental operation of individual units of computers and also with the flow of information.

UNIT - 1

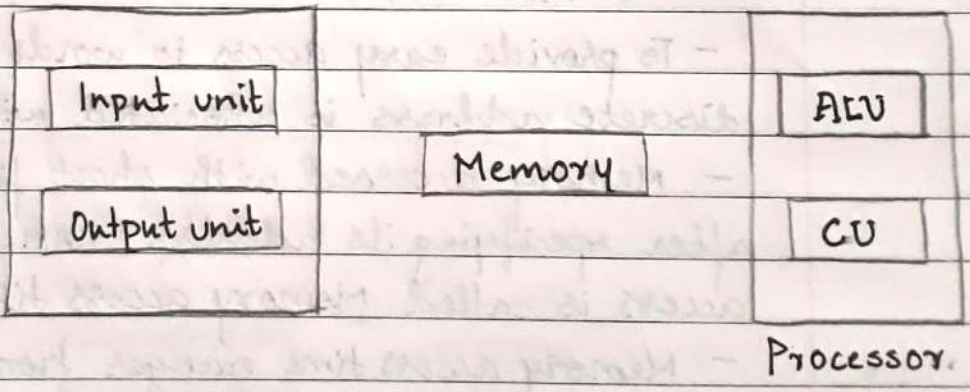
Basic Structures of Computers

* COMPUTER TYPES:

1. Personal computers (Desktop computers)
2. Portable Notebook Computers
3. High Performance Computers: Computation power significantly higher than personal and notebook computers.
4. Main Frame Computers: Faster and greater storage capacity
5. Super Computers: Used for scientific applications such as satellite launching, weather forcasting etc.

* FUNCTIONAL UNITS:

Block Diagram:



Information given as input consists of instructions and data.

Instructions:

- Transfer of information between computer and input and output devices
- Arithmetic and logical operations.

Data:

Any digital information is data.

Ex: Numbers, encoded characters.

Note: Processor is always involved in accepting and transfering data. Processor is always the fastest unit.

Memory Unit:

Memory unit stores information i.e., instructions and data. There are two types of memory :-

1. Primary memory - RAM, ROM
2. secondary memory - Hard disk drive, CD, DVD.

- Primary memory
  - Fastest memory
  - Operates at electronic speed.
  - Programs and data
  - Contains large number semiconductor cells where each cell stores one bit of information (0 or 1).
  - Processed in groups of fixed size - words (powers of 2) (Varies from 8 bits to 64 bits)
  - To provide easy access to words in main memory a discrete address is associated with each word.
  - Memory accessed with short fixed amount of time after specifying its address - RAM. The time taken to access is called Memory access time.
  - Memory access time ranges from 10 to 100 nseconds.

→ Cache memory
  - small fast RAM units
  - limitations: Small capacity, expensive

- Secondary Storage:
  - Magnetic disks, tapes, optical disks.
  - Used when large amount of data need to be stored.

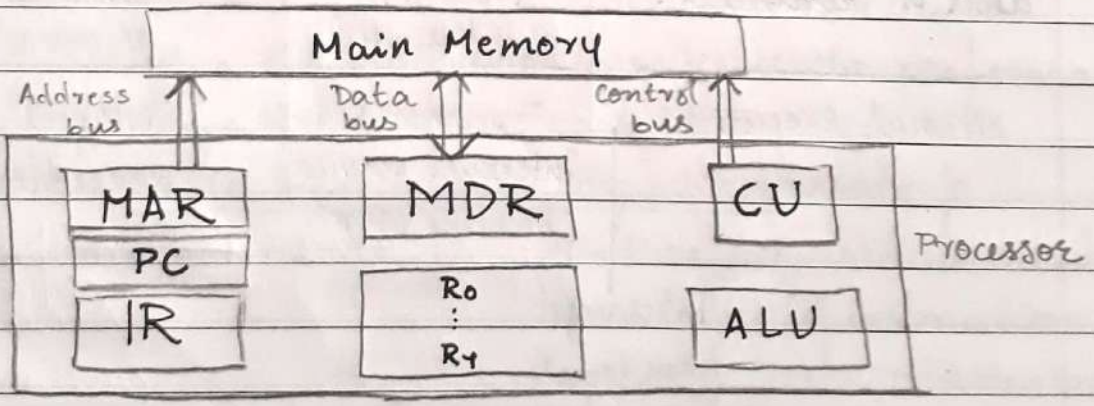
Arithmetic Logical Unit:

- Performs arithmetic and logical operations.
- Processor contains number of high speed registers which are the temporary storage used after arithmetic and logical operations which takes place.
- Each Register can store one word of information.
- Access time is faster than that of main memory (RAM) about 5 to 10 times.
- ALU and CU are usually faster than the other devices connected to the computer system.

* BASIC OPERATIONAL CONCEPTS:

ADD LOCA, R0
(LOCA: source, memory location; R0: destination, processor's general purpose register)

* Connection between Processor or memory:



MAR - Memory address register } Facilitates the communication between processor and memory
MDR - Memory data register }
$R_0, R_1, R_2 \ldots R_4$ - General purpose register
IR - Instruction register - Holds the instruction currently being executed
PC - Program counter
keeps track on execution of program and holds the address of next instruction to be fetched

Fetch → Decode → Execute → Store

* State the operations involved in the execution of instruction:

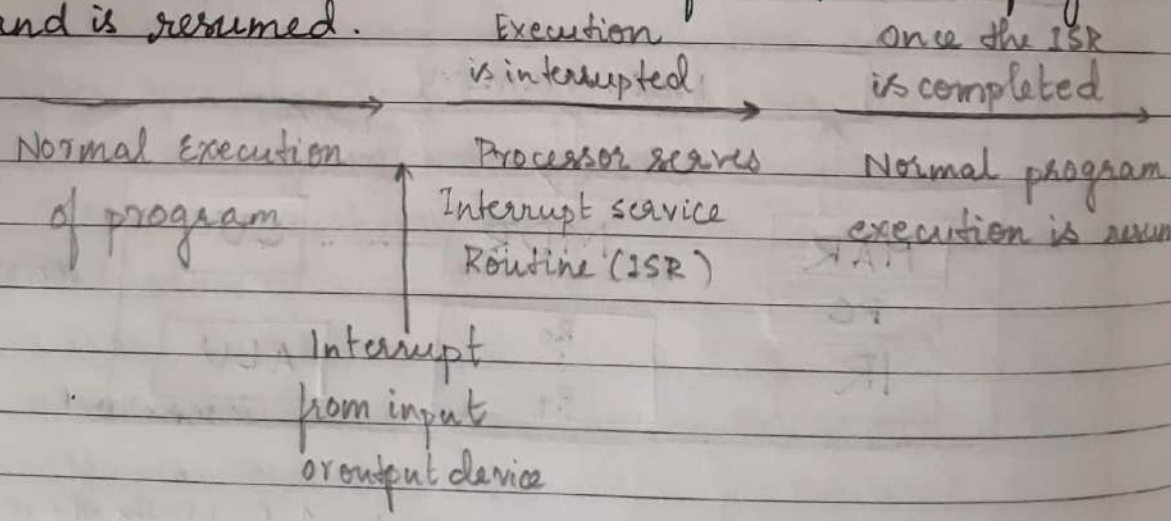
1. ADD R_0, R_1
   → Fetch instruction from memory and placed in IR
   → Decode the contents of IR.
   → Add the contents of R_0 with R_1 and then the result is stored in R_1.

* **INTERRUPT:**

Request from the input or output devices for service by processor.

When there is interrupt from input or output devices then the flow the execution of program is interrupted and the request from interrupt is served first by Interrupt Service Routine (ISR)

Once the interrupt request is served then it gets back to the normal execution of the previous program and is resumed.

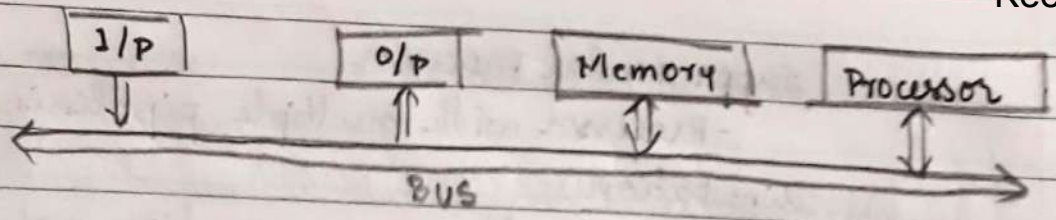


* **BUS STRUCTURE:**

Electrical lines which connects all the units of the computer are called buses. It occurs only powers of two.

Single BUS:

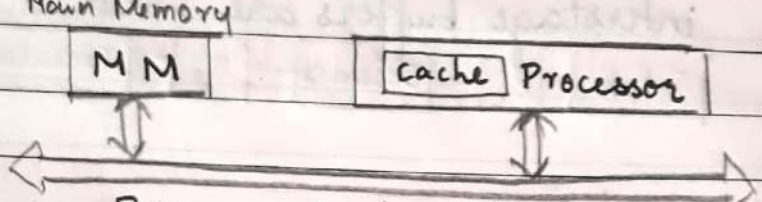
- Bus can be used for only one type of transfer
- Simple and low cost.



* PERFORMANCE:

- Measure of how quickly the computer executes the program.
- Speed is affected by hardware and software.

(1) Hardware : Processor cache



Program instruction are in main memory is copied to cache if repeated execution is required.

For repeated computation the information is placed in cache memory from the main memory since the memory access time is more for main memory.

(2) Processor clock

Processor circuits are controlled by timing signal.
clock rate : $R = 1/P$ where P is clock cycles.
Higher the clock rate greater is the performance.
Ranges from MHz to GHz.

Basic Performance Equation

$$T = \frac{N \times S}{R}$$

T - time required to execute program
N - number of instructions in program
S - steps Fetch - Decode - Execute - Store
R - clock rate

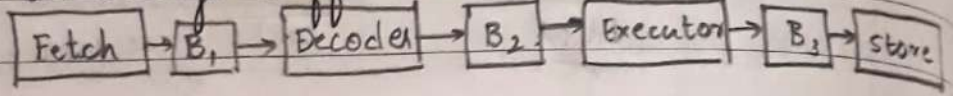
Hence greater the clock rate, lesser is the time taken to execute a program.

(3) Pipelining

More than one piece of hardware working at same time simultaneously is called pipelining. With pipelining the time taken to execute program reduces hence increasing the performance.

Super scalar Processor
- Processor with multiple pipelining.

Limitation
- For simultaneous operation more than one piece of hardware is required hence increase hardware complexity and cost.
- Each step take place at different speed hence not syncronised. To avoid this interstage buffers are used.



(4) Software: Clock Rate:
- Improving the IC technology i.e., by making logic circuit faster.
- Reducing the amount of processing done in each step

Performance Measurement:
SPEC - System Performance Evaluation Corperation.
- Non profit organisation
- Programs - Gaming compiler, database applications.
- Designed computer is compared with respect to the refrence computer.

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

SPEC rating for all the programs

$$\text{Overall SPEC rating} = \left(\prod_{i=1}^{n} \text{SPEC}_i\right)^{1/n}$$ n - number of programs

* MULTI PROCESSORS AND MULTI COMPUTERS:

1. Multiprocessors: - Higher performance at higher cost.
- Executes a number of different application tasks in parallel
- Shared memory (server-client)
- Data cannot be exchanged.
- Complex interconnection of networks

2. Multi computers:
   - Has their own memory.
   - To communicate data with other computers.
   - Message passing systems.

UNIT - 2

Machine Instructions and Programming

$(\underbrace{1101}_{\text{Real Part}} \cdot \underbrace{101}_{\text{Fractional Part}})_2 = 1\times2^3 + 1\times2^2 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-3} = (13.625)_{10}$

- **Binary Addition**

```
  1101
+ 1100
------
 11001
```

Carry Flag = 1

Carry Flag: special register.

* **Overflow:**

- Adding two numbers with same sign.

When MSB is zero it is positive and MSB is one it is negative.

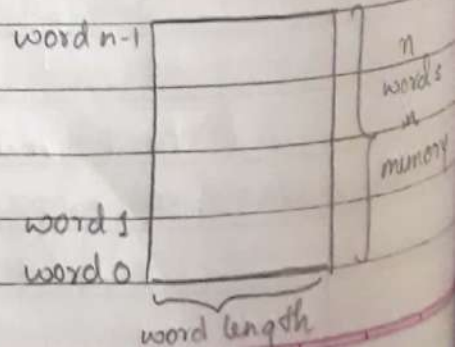
- Carry bit from MSB is not a significant indicater of overflow.
- The sign of the result is same as the added numbers.
- Logical expression to detect overflow.

$$a_{n-1}\, b_{n-1}\, \overline{R_{n-1}} + \overline{a_{n-1}}\, \overline{b_{n-1}}\, R_{n-1}$$

- If it is equal to one then overflow has occured.

* **MEMORY LOCATION AND ADDRESS:**

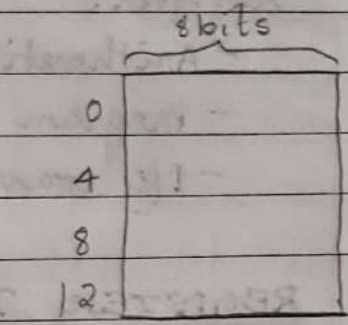
- Memory contains millions of cells, where each cell stores one bit of information.
- It is accessed in groups known as word, usually memory length of 8 bits.
- Address is used to access a particular word in the memory.
- The number of words the processor supports depends on the address bus.



Ex: 10 lines : $2^{10} = 1024 = 1k$. When processor has 10 address lines.

- Greater the processor address lines greater with be the main memory.
- Byte addressability

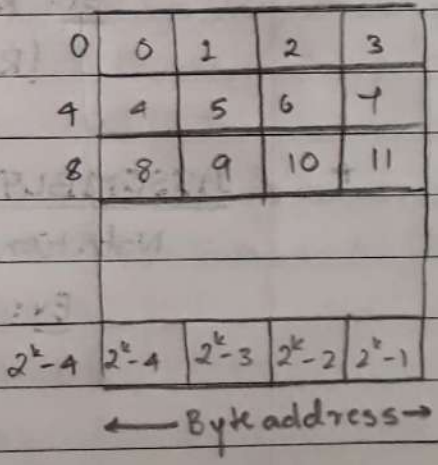
Ex: If word length is 32 bits and the processor word length is 8 bits then the address is in such a way that the successive words are located at 0, 4, 8, 12 and so on.



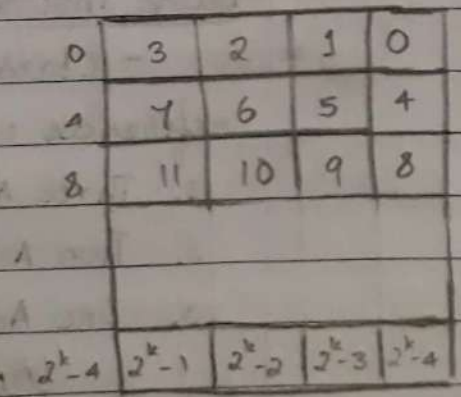
- Data can be read by both MSB or LSB.

* BIG ENDIAN ASSIGNMENTS AND LITTLE ENDIAN ASSIGNMENTS:

Big Endian Assignment — Lower Byte address is used for the most significant byte of the word (Motorola)



Little Endian Assignment — Lower Byte address is used for the least significant byte of the word (Intel)



* INSTRUCTIONS:

- Data is transferred between memory and processor register.

classification
- Arithmetic and logic group of instructions
- Program sequencing and control group
- I/O transfer

* REGISTER TRANSFER NOTATION: (RTN)

- Processor registers are represented by $R_0, R_1, \ldots$
- Address of memory location: LOC, PLACE, MEM etc.
- Input/output Registers: DATA IN, DATA OUT. etc..
- contents of Register/memory are denoted by [.].

Ex: $R_1 \leftarrow [LOCA]$ contents of A to R_1

$[R_3] \leftarrow [R_1] + [R_2]$

* ASSEMBLY LANGUAGE NOTATION:

Notation used to represent Assembly Instructions.

Ex: MOV LOC, R1

ADD R1, R2, R3.

- Basic Instruction Types:

- classified based on the number of address references used.

1. Three Address Instructions : ADD R_0, R_1, R_2
2. Two Address Instructions: ADD A, B
3. One Address Instructions: ADD A (A added to A and stored in A)

   LOAD B (contents of memory location B is loaded into accumulator)

   STORE C (contents of accumulator is stored into memory location C)
4. Zero Address Instructions: Operation on stack memory

   PUSH (Push down the stack)

   POP

* **Instruction Execution and Straight line sequencing:**
  - Assume Instruction length - 32 bit
  - Byte addressable
  - Three instructions in memory starting from address 'i'.

	← 8 bits →	
i	MOV A, R_0	$A \rightarrow R_0$
i+4	MOV B, R_0	$B + R_0 \rightarrow R_0$
i+8	MOV R_0, C	$R_0 \rightarrow C$

Processor Register - Program counter (Holds the address of instructions)
→ The address of first instruction is placed in Program counter
→ Program counter is incremented by 4.

* **Branching:**
Transfer of program from one straight line sequencing to another straight line sequencing.
Conditional : JZ, JC, JNC, JNZ
Unconditional : JMP

JC - Jump on carry
JZ - Jump on zero
JNZ - Jump on non zero
JNC - Jump on no carry

Condition codes - Flag
AC (Auxilary carry) - It is set when there is a carry from lower nibble to higher nibble

```
      (1) ↗AC
1010 1011
1000 1000
0011 0011
```

C (Carry Flag) - It is set when there is a carry from higher nibble.
Z (Zero flag) - It is not set. It is set if an arithmetic operation result is zero.
V (Overflow flag) - It is not set. It indicates an arithmetic overflow has occured in the operation
P (Parity) - Number of ones in the result (Odd parity / Even parity)
Z (Negative Flag) - MSB indicates sign of number. Here MSB is 0, hence it is not set

* **Generating Memory Address:**
  - Flexible way of specifying the address of operand.
  - **Register mode**
    Operand is the contents of processor registers.
    The name of register is specified in instruction.
    Ex: MOVE R1, R2 (content of R_1 → content of R_2)
  - **Direct Mode / Absolute Mode**
    Operand is in the memory location.
    The address of memory location is specified in instruction.
    Ex: MOVE LOC, R1 / MOVE 10, R1 (content of 10 → content of R1)
    10 → data memory location
  - **Immediate Mode**
    Operand is specified explicitly in instruction.
    Ex: MOVE #200, R1 (200 → content of R1)
  - **Indirect Mode**
    Effective address of operand is the contents of register or memory location.
    Ex: MOVE (R1), R_2 (content of 20 → content of R_2) If R1 = 20, then 20 is the data memory location.
    R1 is a pointer operator here.
    Register / memory that contains the address of operand is called a pointer.
  - **Indexed Mode**
    Effective address of operand is generated by adding a constant value (specified in instruction) to the contents of register (offset).
    Ex: ADD 20(R1), R2 If R1 = 1000H, Effective Address = 20 + 1000
    (content of 1020H + content of R_2 becomes content of R_2) Effective Address = 1020H
  - **Additional Modes**
    Auto Increment – Pre Increment: MOVE +(R1), R_2
    – Post Increment: MOVE (R1)+, R_2
    Assuming R1 = 1000H MOVE +(R1), R2 (contents of 1001H → contents of R2)
    MOVE (R1)+, R2 (contents of 1000H → contents of R2)

Auto decrement - Pre decrement : MOVE -(R1),R2
Post decrement : MOVE (R1)-,R2

* **ASSEMBLY LANGUAGE:**

Two / Three / Four letter words are used to represent instructions and these words are called 'mnemonics'.

Ex: ADD A, B ; (A-source, B-destination, ADD mnemonics)
JC label ; (JC-mnemonic, label-Program memory address)
(If there is a carry it jumps to label, if not the next one is executed)

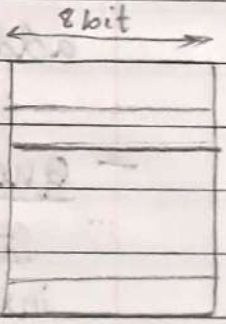
- **Assembler Directives** (pseudo-operations)

-Instructions which are not a part of processor instruction set are called assembler directives.
- instruction to assembler, linker or loader.

DB-Define Byte
DW-Define word
DD-Define Double word
DUP-Duplicate (used to initialise locations and access them)

format: Name Data_type Num DUP

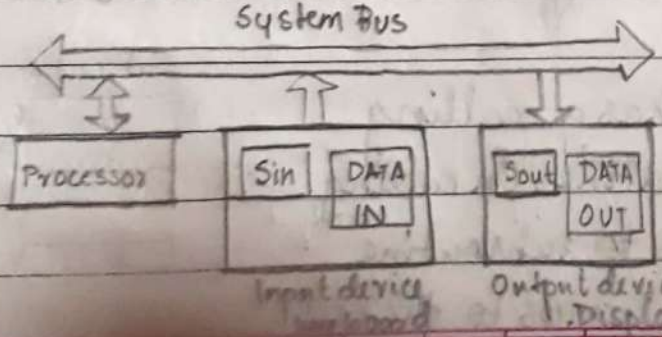
Ex: Table DW 10 DUP(0)
(Now 1-10 are initialised to zero).



EQU-Equate

Ex: N1 EQU 10
(variable N1 is 10 throughout the program)

- **Basic Input Output Operations:**



since the processor operates in very high speed, hence the input and output device need buffers to store data temporarily, i.e., data in and data out.

serial input and serial output

— Sin sets one when a key is pressed which lets the processor know that a key is pressed.

Sout sets one when the output is ready this lets the processor to ~~give display output~~ know that it is ready to accept

Input to processor

— character is stored in DATA IN register as 0's and 1's once the key is pressed. As Sin is set to 1, the processor reads the contents of DATA IN register and sets Sin to

Processor to output

— Processor checks Sout, if Sout is 1 then the processor sends data to DATA OUT register. If Sout is 0, the processor has to wait until it sets to 1 and is ready to accept data.

— Stacks: (Last in First Out)

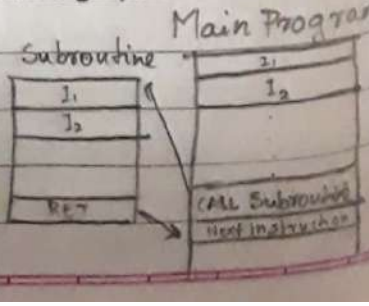
- List of data items or elements which is usually bytes or words.
- There is restriction in accessing data as the elements can be added or removed at one end of the list only.
- stack pointer is a register used to keep track of the address of element of the stack.

— Queue: (First in First Out)

- Elements can be removed from front end and inserted from back end.

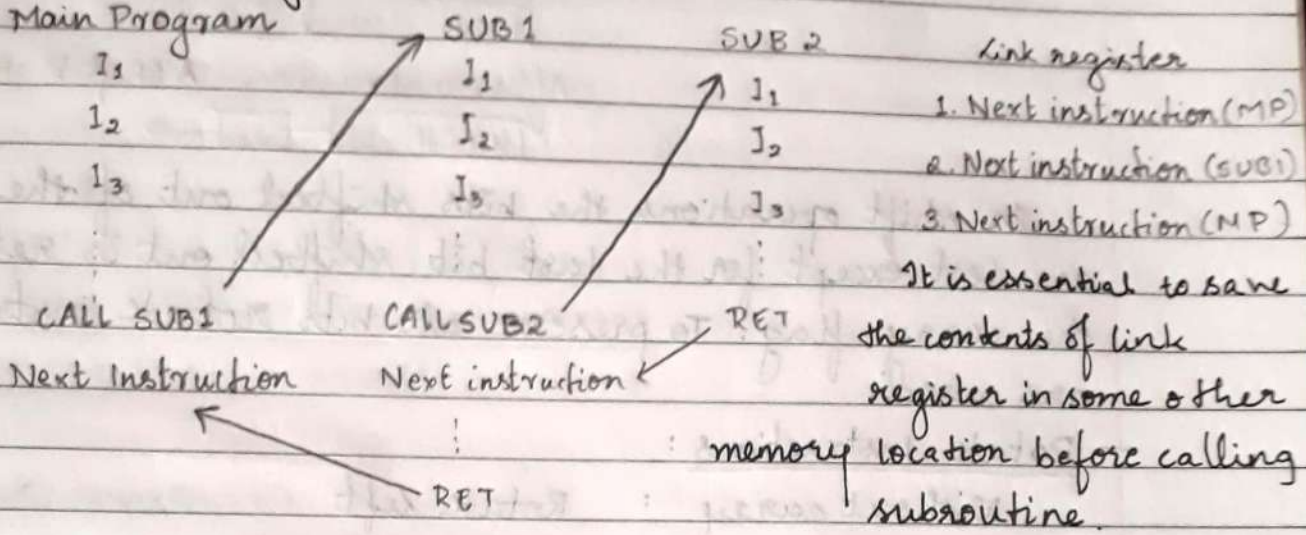
— Subroutines:

- In a given program, if a particular subtask has to be performed on a different data values then subroutines are used.
- Main program has a calling instruction for subroutine. On calling subroutine it jumps to subroutine and once executed it jumps to the next instruction of main program.



- Link register: stores the return address in specific location in memory i.e, the address of the main memory instruction once the subroutine program is executed that has to be executed in main memory.

Subroutine Nesting:



★ Additional Instructions:

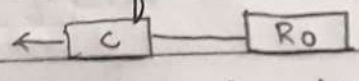
- **Logical Instructions:** AND, OR, NOT

  2's complement: NOT R0 ; 1's complement
  ADD #1, R0 ; 2's complement
  (or)
  NEGATE R0 ; 2's complement

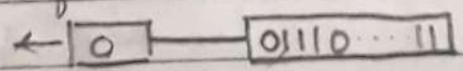
- **Shift Instructions:**

  Logical shift: ⊥ Logical shift left
  LShift L count, Dst

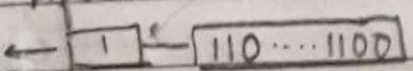
  Used in multiplication



Before instruction

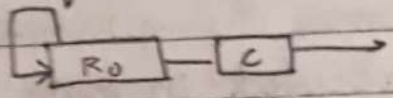


After instruction LShift L #2, R0

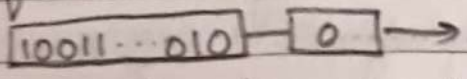


Arithmetic shift: Arithmetic Shift Right

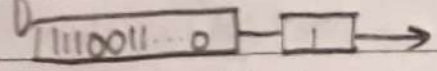
AShift R count, Dst



Before instruction:



After instruction AShift R #2, R0

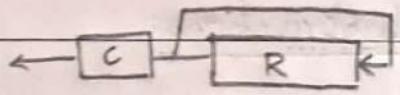


In shift operations the bits shifted out of the operand are lost except for the last bit shifted out is retained in the carry flag. To preserve all bits rotate instructions are used.

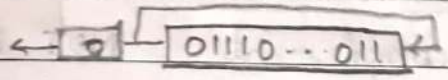
- Rotate Instructions:

Without carry: Rotate left

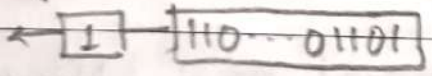
Rotate L #2, R0



Before Instruction

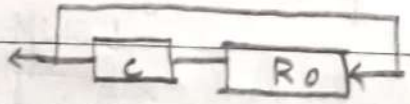


After Instruction

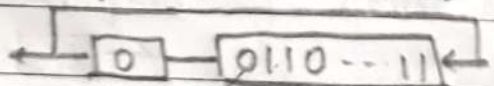


With carry: Rotate left Carry

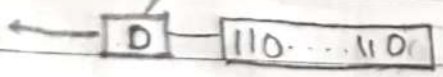
RotateL #2, R0



Before Instruction



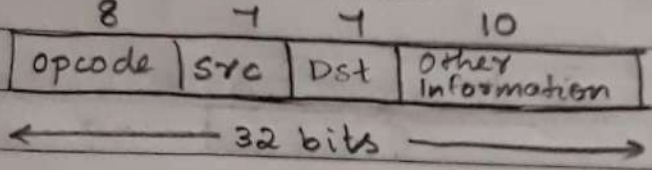
After Instruction



Rotating in the opposite direction gives back the original data

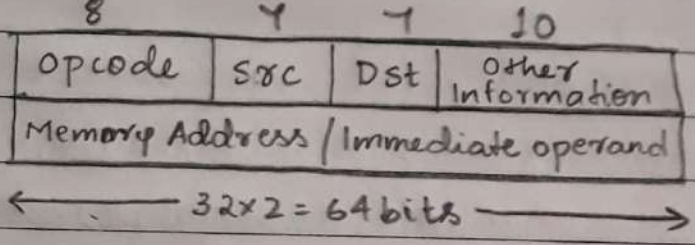
* ENCODING OF MACHINE INSTRUCTION:

- One Word Instruction:

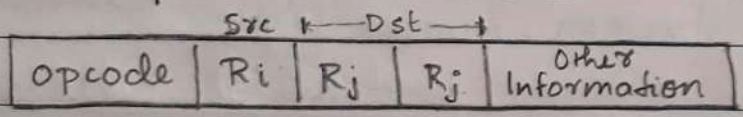


opcode - operational code
Ex: ADD is an operational code for addition

- Two Word Instruction:



- Three Operand Instruction:

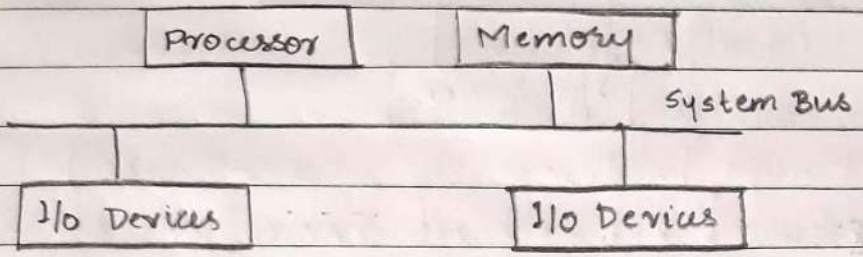


Page No.:

UNIT - 3

Input/Output Organization

* Accessing Input/Output Devices



There are 3 types of Buses:
1. Address bus
2. Data bus
3. Control bus

1. Memory Mapped I/O:

- I/O device and memory shares the same address space.

Ex: MOVE DATAIN, R0

Here DATAIN - Address of input buffer associated with keyboard.

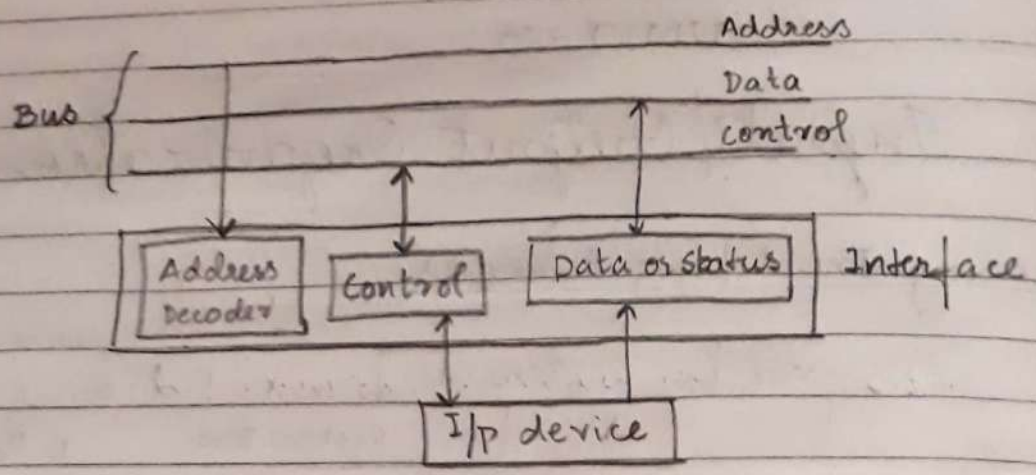
2. I/O Mapped I/O:

- I/O devices and memory have different address space which is provided by the processor. Less memory space for I/O devices and more space for memory.
- Less decoding is required for I/O devices as it has been connected with less address lines.

10 address lines
$\Rightarrow 2^{10}$ memory space

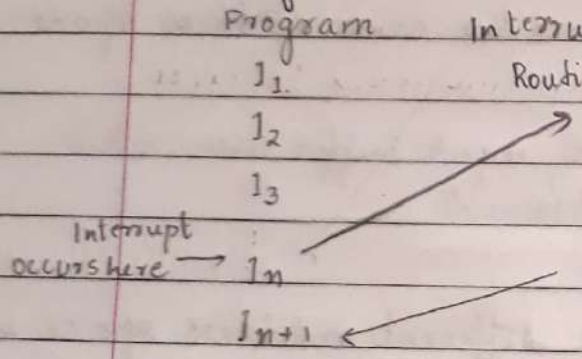
* I/O INTERFACE FOR AN INPUT DEVICE:

- Address decoder - Enables the device to recognise the address
- Data register holds the data being transfered to the processor or from the processor.



★ **INTERRUPT:**

Programmed I/O - Processor repeatedly checks the status bit, if set to 1 then it is that there is an input data.

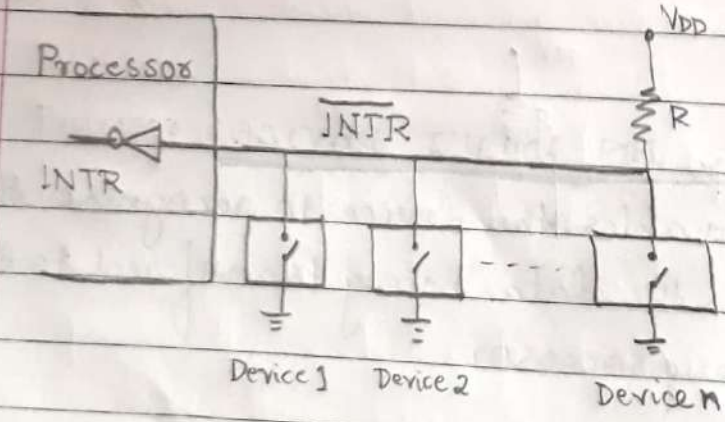


Once an interrupt occurs, first the interrupt service routine is served and later the previous program is continued.

Interrupts:

1. By external signal (Hardware Interrupt)
   Ex: Reset
2. By special instruction (Software Interrupt)
   Ex: Print

— **Interrupt Hardware:**



INTR - Interrupt request line

— When V_{DD} is logic high and the switches are open the current flows through INTR line which is inverted hence the interrupt request is logic low.

— When V_{DD} is logic high and the switch is closed the current flows to the device and further to ground which intern indicates the interrupt request.

To request interrupt, the device closes the switch. Now the interrupt line drops to zero potential which is inverted hence the INTR is high indicating interrupt request by the device.

If INTR1 .... INTRn are inactive

$\overline{INTR} = V_{DD}$ this is inactive

Value of INTR = INTR1 + ... + INTRn

(logical OR of the requests of individual devices)

If INTR1 .... INTRn, any of them are active

$\overline{INTR} = 0$ this is active

Value of INTR = INTR1 + ... + INTRn

- Enabling or Disabling Interrupts:

Maskable Interrupts (under software control)

- Processor does not respond to the interrupt request.

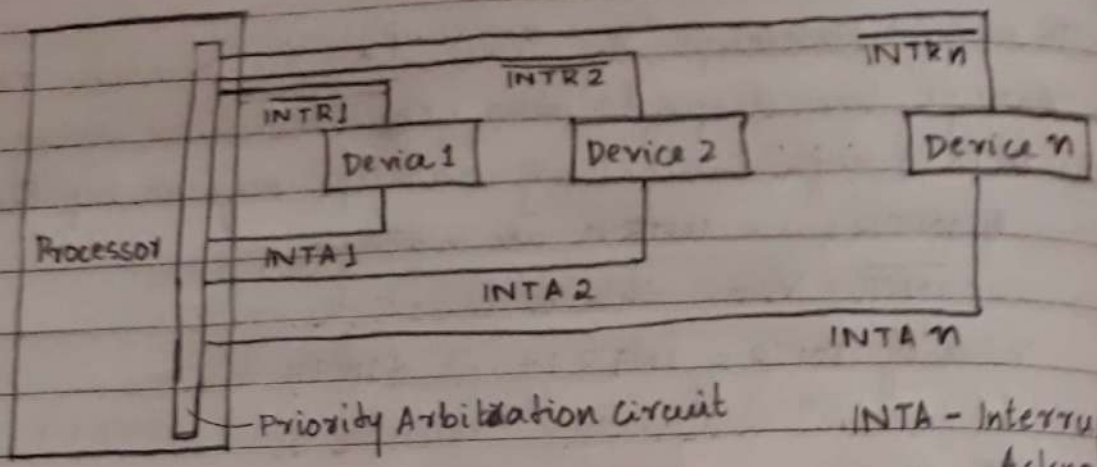
Non Maskable Interrupts (Hardware Interrupt) Ex: Reset

- Cannot be ignored and the processor has to respond to the interrupt request.
- It cannot be masked under software control.

- Handling the Multiple Interrupts:

There are three ways to handle the multiple devices.

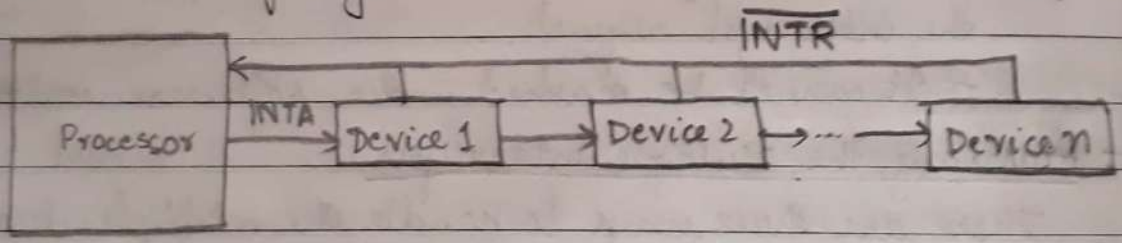
1. Vectoring of Interrupts:
   - External device interrupts the processor
   - Processor executes ISR (Interrupt Service Routine) from predetermined memory location which is the first address of Interrupts Service Routine (ISR) (vector address)
2. Interrupt Nesting:
   - A system of interrupts that allows ISR to be interrupted is interrupt nesting.
3. Interrupt Priority:
   - Interrupt with higher priority will be served first when there are multiple interrupts.



NOTE: In interrupt hardware the processor gets to know there is an interrupt but does not know from which device since only one interrupt request line is used.

In interrupt priority circuit each device has its own priority and is connected by separate lines, with each of them having one request line and one acknowledgement line

For simplifying the circuit: Daisy chain is employed.



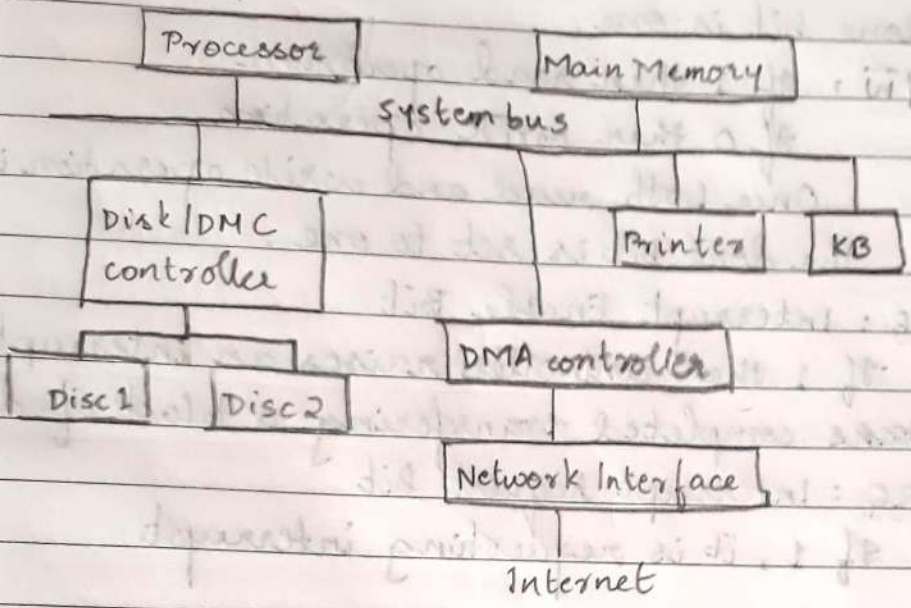
INTR - Common interrupt request line for all devices

INTA - Connected in a daisy chain, signal passes serially through devices. INTA passes through the device which has no interrupt and is blocked by the device which has request.

If there are requests by two devices, then the INTA signal is blocked by the first device and is first served later the INTR is checked again and then INTA signal passes through the first device as it is already served and is then blocked by the next device to be served.

Device electrically closest to the processor is having highest priority.

- Direct Memory Access : (DMA)
  - Data transfer takes place without intervention by processor.
  - DMA controller - it operates under the control of program executed by processor.



The processor needs to provide the following in order to transfer from one disc to another (Harddisc and DVD):

To initiate block transfer
- Processor sends starting address of block.
- Number of words in block
- Direction of transfer.

On receiving this information DMA controller performs the required operation.

DMA - is a hardware used to transfer data.

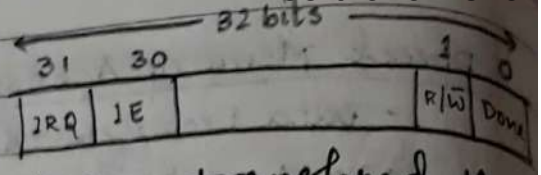
~~Register: First address is stored~~

~~Counter: Number of words to be transfered~~

DMA controller:

1. Starting address register
   - Stores starting address of the block.
2. Counter Register:
   - Counter is set to the number of words to be transfered.

3. DMA control Register

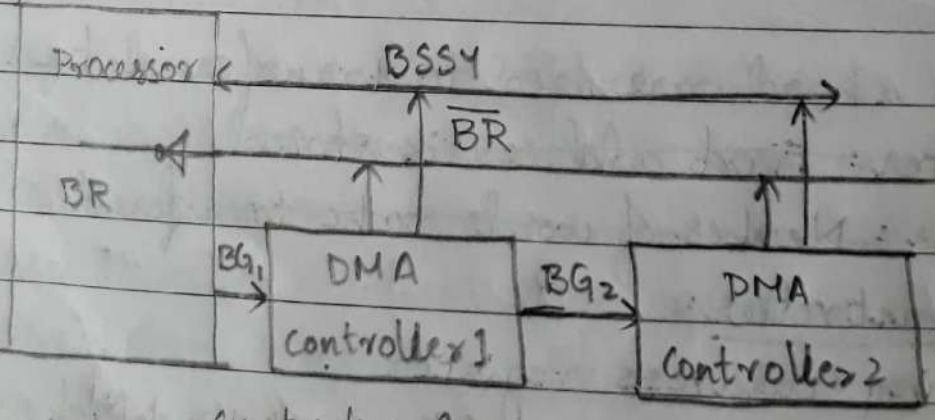


- 32 bit register
- Done bit: When data is being transfered the done bit is zero. Once the data transfer is compl the done bit is one.
- R/$\overline{W}$: If 1 then read operation
  If 0 then write operation
  Once both read and write operation is o the done bit is set to one.
- IE: Interrupt Enable Bit
  If 1 then controller raises an interrupt a has ~~trans~~ completed transfering a block of data
- IRQ: Interrupt Request Bit
  If 1, it is requesting interrupt

★ BUS ARBITRATION:

There are two types of bus arbitration:

1. Centralized Bus Arbitration:
   - Single Bus Master (processor) or DMA controller)
2. Distributed Bus Arbitration
   - All devices waiting to use the bus has equal responsibility to carry out arbiteration. without us processor (central arbitration).



Centralized Bus Arbitration

$\overline{BR}$ - Bus request line

Used to indicate the processor that DMA controller needs to become bus master.

BG - Bus grant signal

When the processor grants the request from DMA controller it generates the grant signal allowing the DMA controller to access the bus.

The Bus grant signal is blocked by the ~~the~~ DMA controller that requested and is passed through the DMA controller which has no request signal.

BSSY - Bus busy line

The DMA controller first checks the Bus busy line
If active - the DMA controller has to wait and cannot activate the bus request as the processor is busy.
If inactive - the DMA controller can generate request as the processor is ready to accept the request.

Centralised bus arbitration is always prefered over Distributed bus arbitration as it leads to clashes between the devices as all the devices have equal responsibility to access the bus.

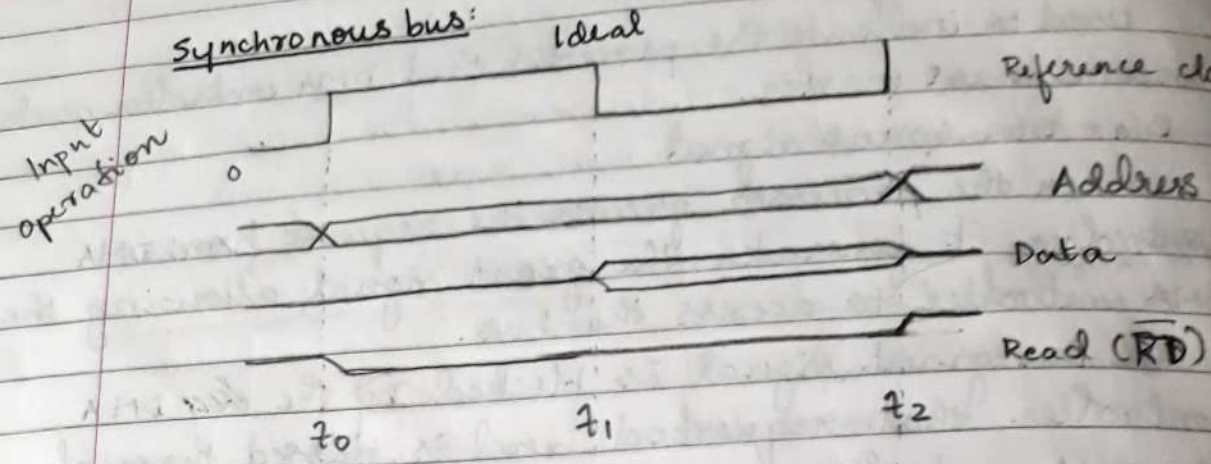
* **BUS:**

Electrical lines connecting processor, memory and input output devices.

* There are two types of buses:-

1. **Synchronous BUS:**
   With reference to clock pulse (Prefered over asynchronous bus)
2. **Asynchronous BUS:**
   Hand shake signals (Master ready - slave accept)
   (reliable but slow)

Synchronous bus:

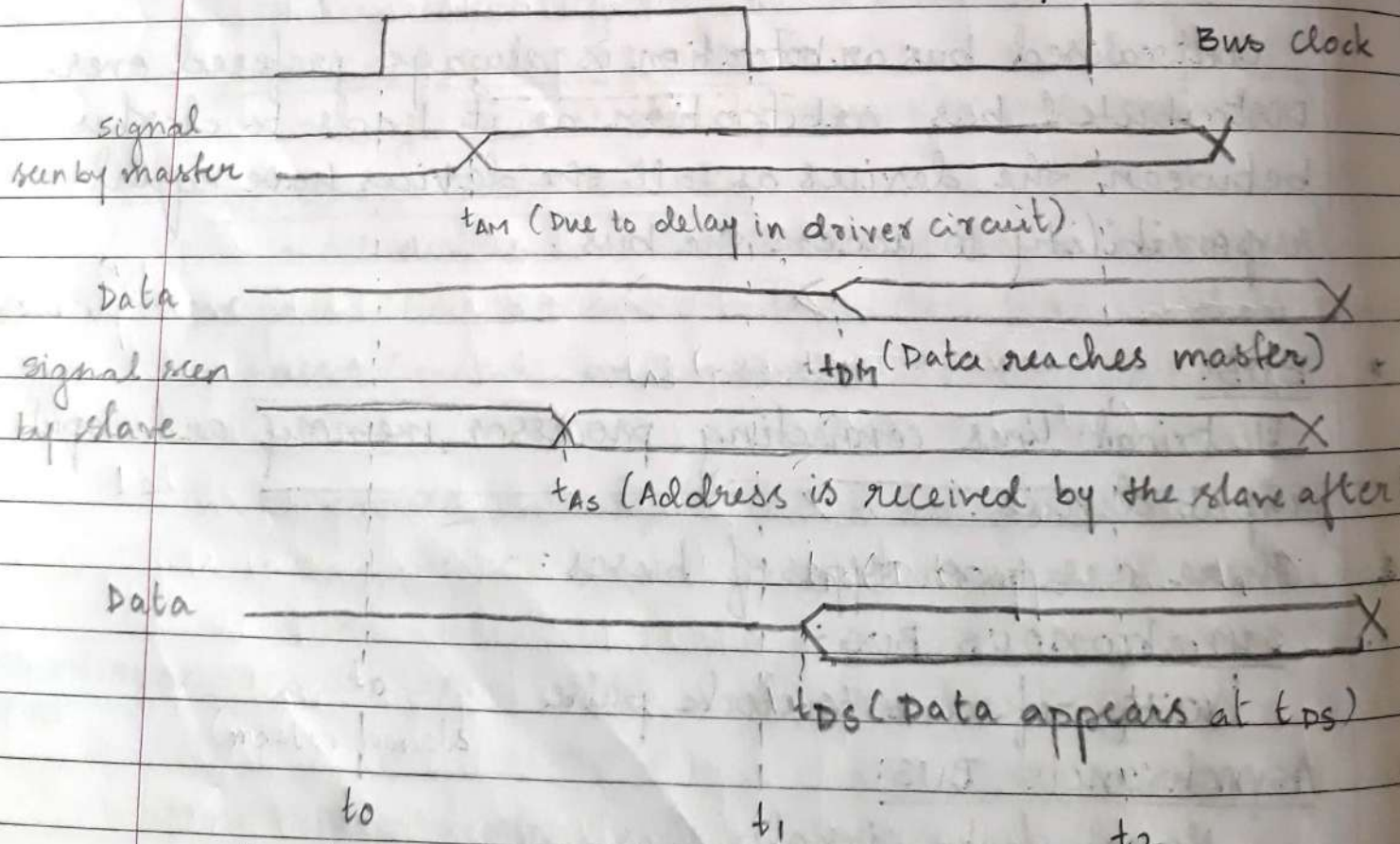


t_0 - processor places address on address line or sets control line to perform input operation.

t_1 - addressed device places data on data bus.

t_2 - processor reads the data from data lines or loads data into input buffer.

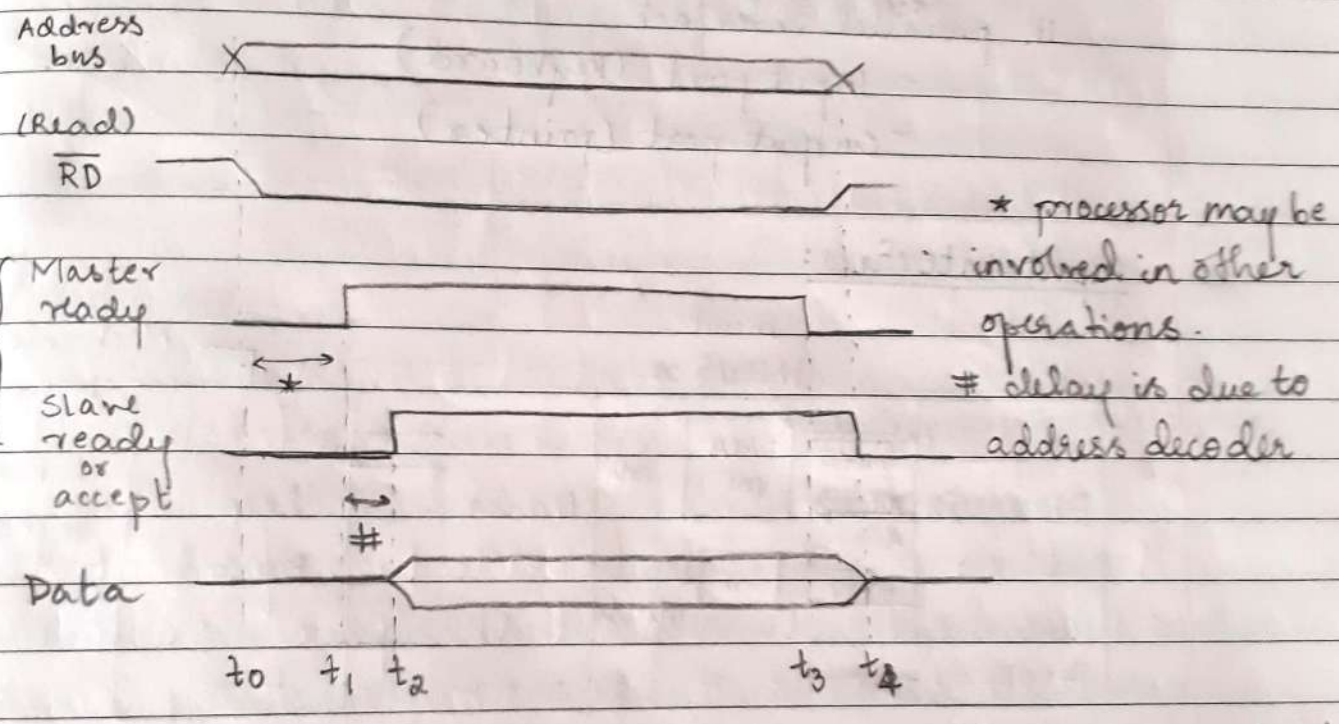
considering propagation delay



Limitations

- Transfer has to be completed with one clock cycle t_2 to to must be selected to accomodate the longest delay on bus slowest.
- There is no way of determining whether the slave is responding to data transfering or not.

Asynchronous Bus:



t_0 - processor places address over address line and sets the control ($\overline{RD}$) to perform input operation.

t_1 - processor informs the input devices that it is ready to accept data.

t_2 - Addressed devise places the data on data bus and informs the processor that it performed the requested operation.

t_3 - processor reads the data and places the data into its input buffer.

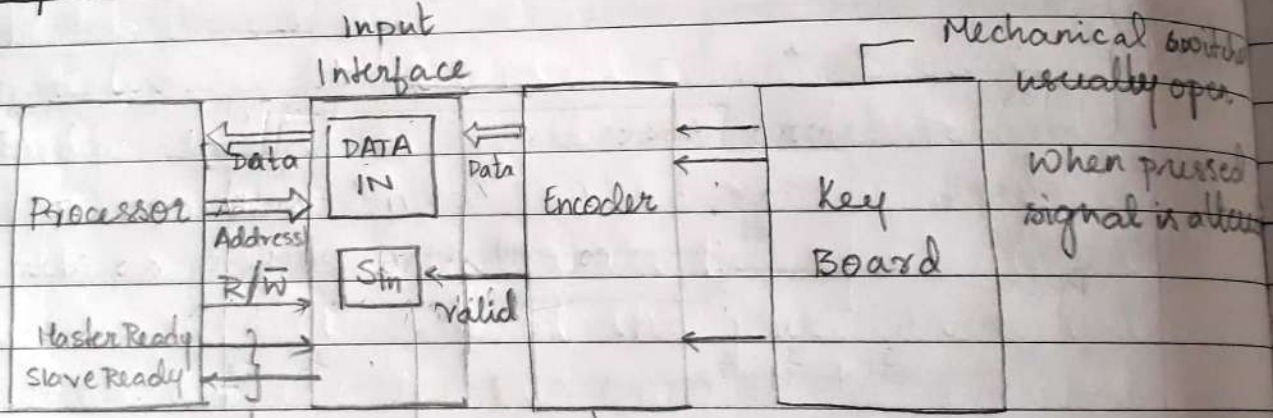
t_4 - the processor withdraws the master ready signal as the input is servered.

limitations
- since the processor is faster than other devices connected if the input device does not respond to the master ready signal by the processor leads to larger delay

★ INTERFACE CIRCUITS:

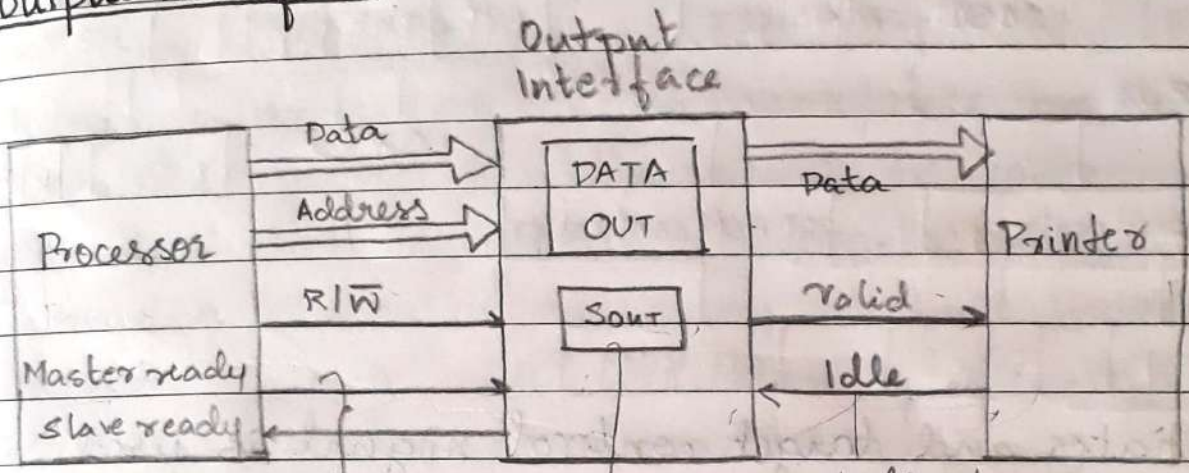
Interfacing can be
i. serial interface
ii. parallel interface
- Input port (keyboard)
- Output port (printer)

Input Interface:



Every key has a unique code based on which the processor gets to know which key is pressed. Once a key is pressed the switch is closed establishing electrical signal. This signal is detected by the encoder and the corresponding ASCII code is generated. The output of encoder is encoded character along with a control signal called valid indicating key is pressed. Now the ASCII code is loaded to DATA IN and SIN is set to 1.

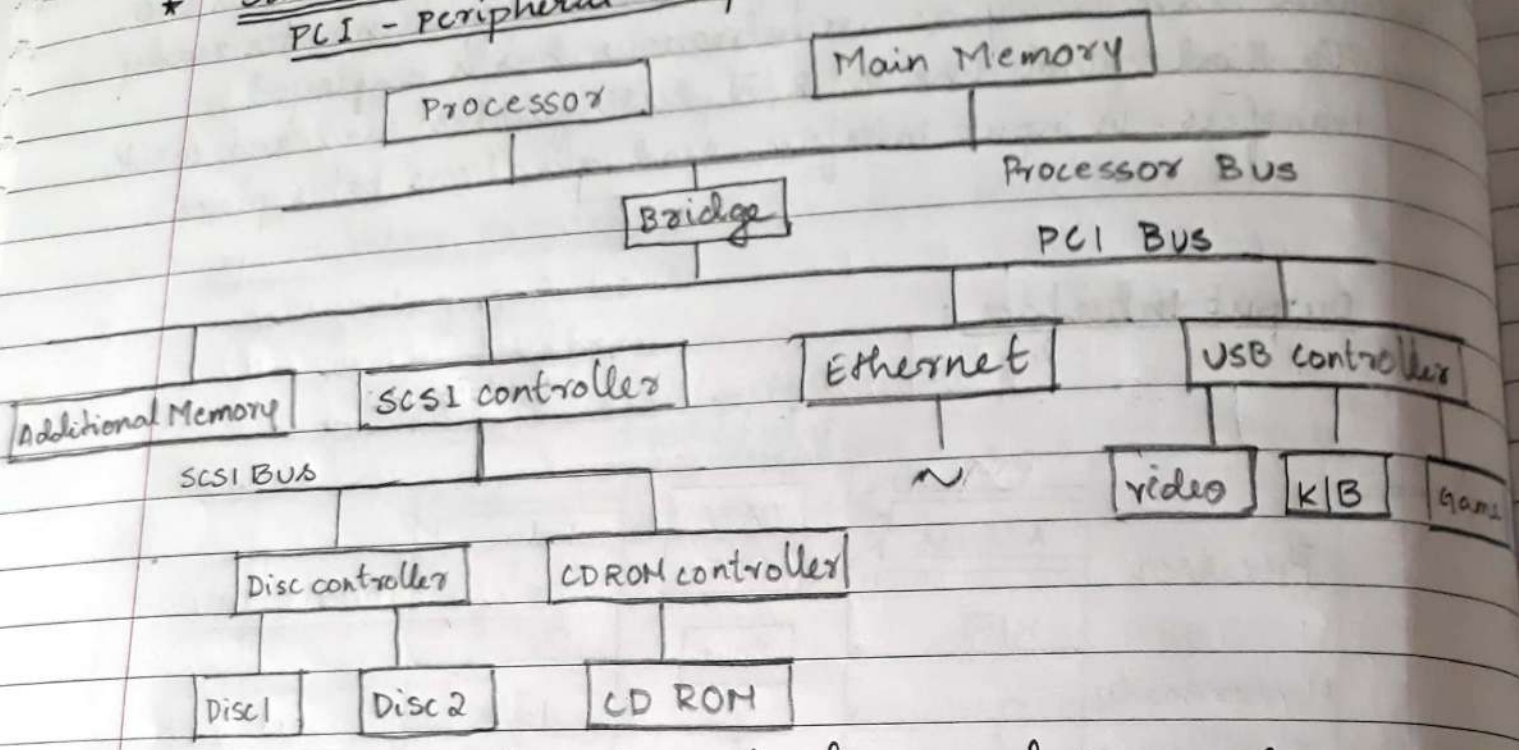
Output Interface:



The printer operates under handshake signals Valid and Idle which is similar to Master ready and Slave ready signals. When printer is ready to accept a character it sends Idle signal. When there is data valid signal is activated.

The SOUT flag is set to 1 when the printer is ready to accept data and is cleared to 0 as soon as data from the processor is loaded to DATAOUT.

* Standard I/O Interfaces:
PCI - Peripheral component Interconnect Bus.



- Initiator and target control signal is used.
- Bridge - translates signals and protocols
- SCSI controller - Small computer system Interface
It acts as a DMA controller. To transfer data SCSI controller needs to request the processor to use the bus and can use the bus only if the processor permits.

SCSI bus is used to transfer of data between the Discs and the CDROM.

Disc controller controls the data that is written in to the disc. Where it has to copied is seen by the disc controller.

Similarly CD ROM controller controls the data and where it has to be copied in the CD ROM.

- Ethernet helps for all wired and wireless connectio

Features of PCI Bus:

- PCI is processor independent, ie., there is no different PCI bus for different kinds of processor.
- Low cost
- User friendly.
- Supports single processor aswell as multiple processors.
- Three independent address spaces for memory, I/O, configuration (plug and play capability).

PCI terminology

Master - Initiator : Processor / DMA controller

Addressed device that responds to commands issued by the initiator is called a Target Device.

- complete transfer operation on bus involving address and data along with direction of transfer is called Transactions. Phases are direction of transfer, address and data, number of words etc., with in the transaction.

SCSI Standards:

		Clock Speed		
SCSI 1	Regular	5	5 Mbps	Single
	Fast	10	10 Mbps	
	Fast 20 (Ultra)	20	20 Mbps	
	Fast 40 (Ultra)	40	40 Mbps	
SCSI 3	Fast 80	40	80 Mbps	Double
	Fast 160	80	160 Mbps	

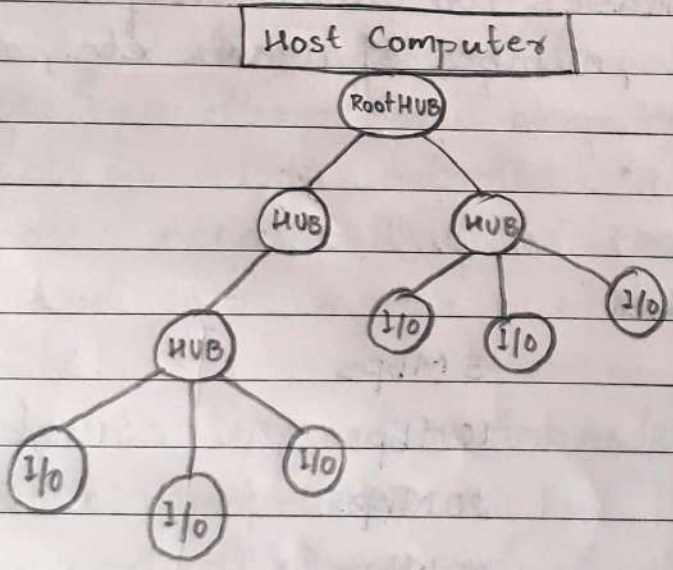
40 Mbps in one direction and 40 Mbps in another direction
similarly 80 Mbps in one direction and 80 Mbps in another.

Universal Serial Bus (USB):

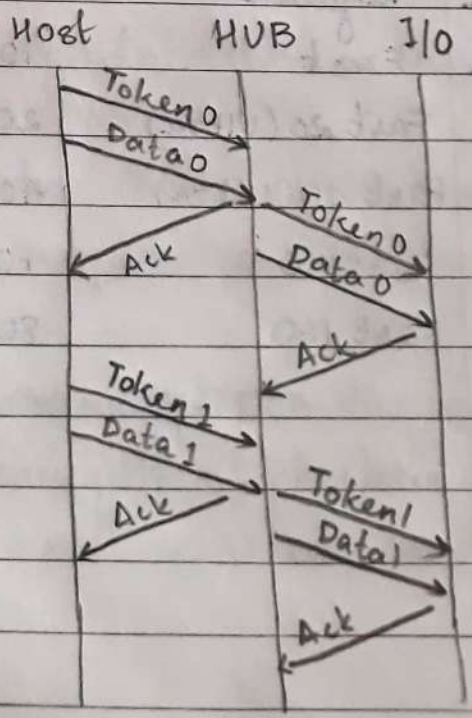
- Objective - To make it possible to add many devices to the computer with out opening computer box.
- Features - Simple connectivity, plug and play. Automatic configuration and no user settings.
- Speed - Supports two high speed data transfer protocol.
  Isochronous: Scanners, Digital cameras 12Mbps.
  (It uses clock signal)
  Asynchronous: keyboard, mouse 1.5Mbps
  (It uses hand shake signal)

USB Architecture:

USB Tree



Reliable transfer as it consists request and acknowledgement.



LOC A contains a
LOC B contains b
LOC c contains c

SQRT - square root
NEG - negative of number
Assuming the processor supports square root.

Program:

```
                                    ; location
Load a, R1                          ; R1 = a
MUL c, R1                           ; R1 = ac
LShift L #2, R1                     ; R1 = 4ac
Load b, R2                          ; R2 = b
MUL b, R2                           ; R2 = b^2
SUB R2, R1, R3                      ; R3 = b^2 - 4ac
                  destination
CALL SQRT                           ; calling subroutine : R3 = √(b^2-4ac)
                                      to compute square root
Load b, R1                          ; R1 = b
NEG R1                              ; R1 = -b
ADD R1, R3, R4                      ; R4 = -b + √(b^2-4ac)
LOAD a, R2                          ; R2 = a
LShift L #1, R2                     ; R2 = 2a
DIV R4, R2, R5                      ; R5 = (-b+√(b^2-4ac)) / 2a
SUB R1, R3, R6                      ; R6 = -b - √(b^2-4ac)
DIV R6, R2, R7                      ; R7 = (-b-√(b^2-4ac)) / 2a
```

LShift L #2
Ex: 0001
=> 0100
multiplied by 4

Timing diagram of DMA

Interface circuits - serial interface

IRDY# processor is ready to accept data (Initiator read

TRDY# Target ready.

FRAME #

DEVSEL # Device select

IDSEL # Initiator device select

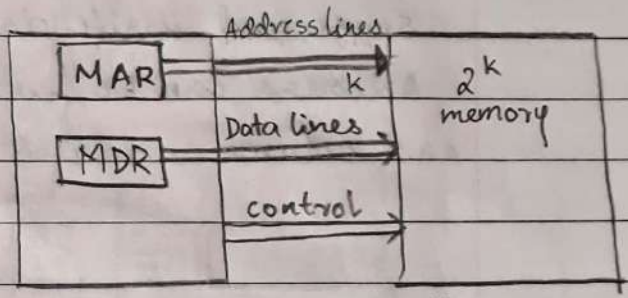
TDSEL # Target device select

UNIT - 4

The Memory System

* **Memory:**

- Maximum number of memory depends on number of address lines used.
  i.e., 20 address lines ⇒ 2^{20} memory locations.
- Byte addressability
  - Little Endian (Intel)
  - Big Endian (Motorola)
- MAR - Memory Address Register
MDR - Memory Data Register are the two registers used to facilitate data.

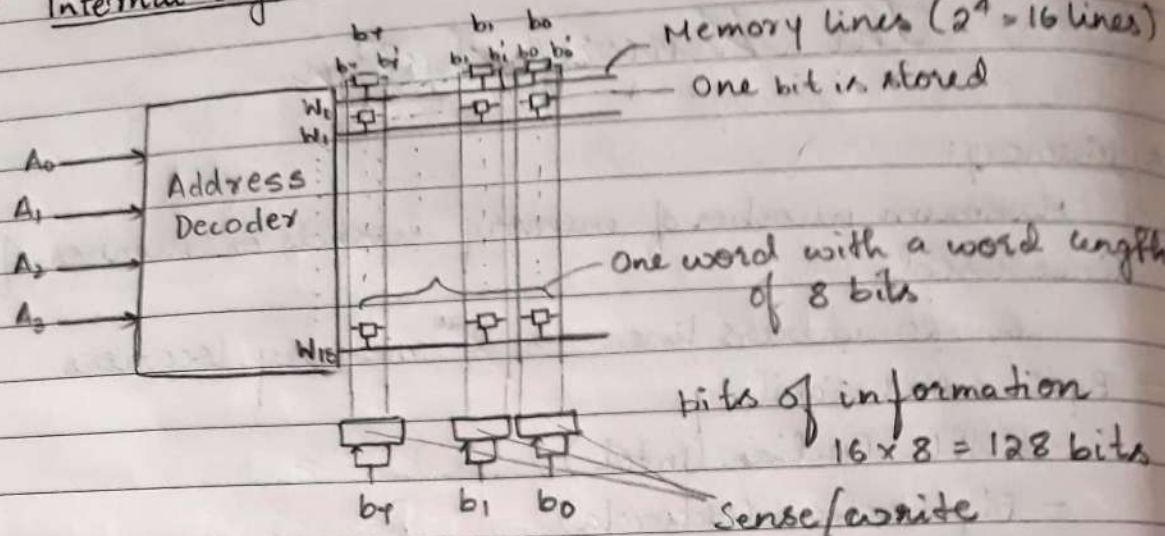


control signal : ~~R/W~~
$\overline{RD}$ (Read) $\overline{WR}$ (Write)

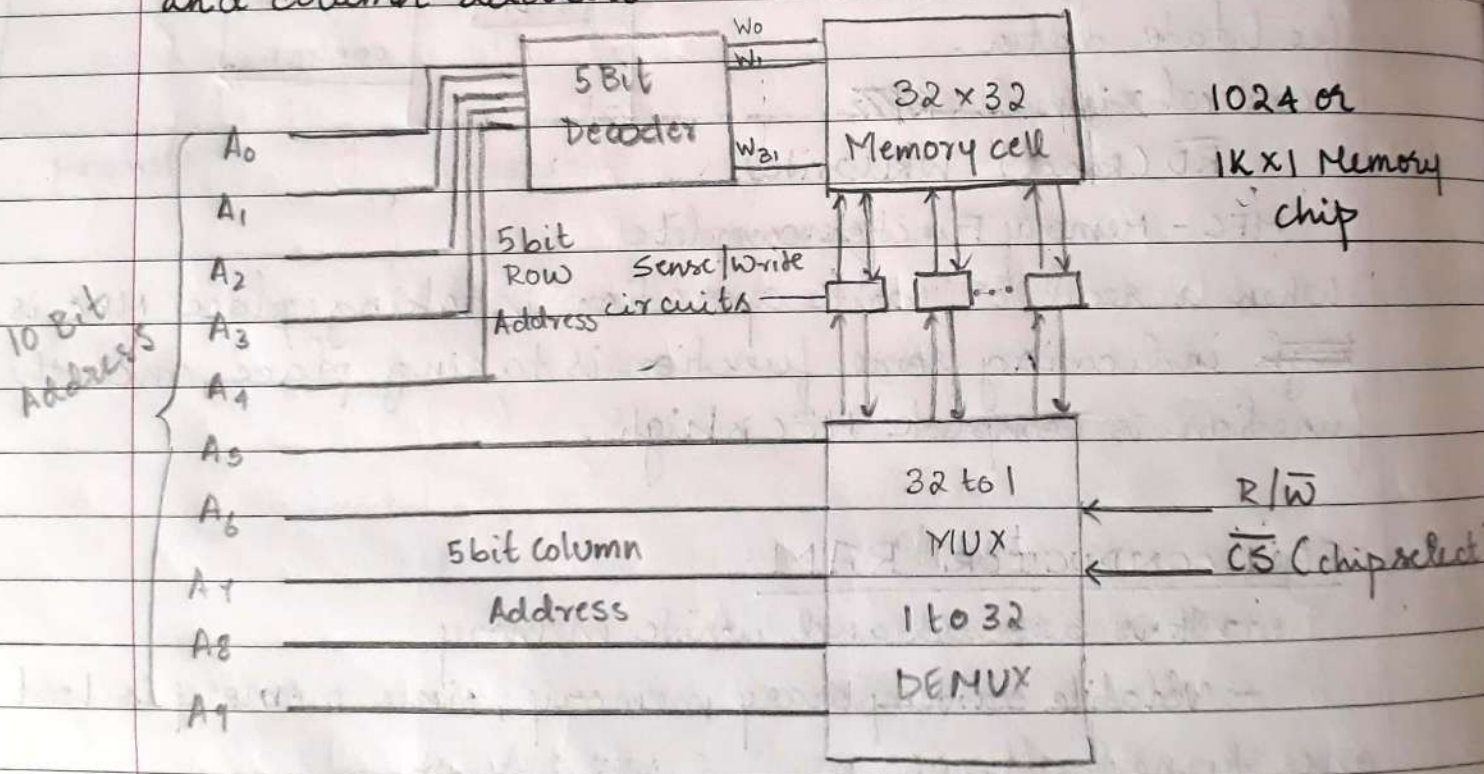
MFC - Memory Function complete
When a read or write operation is taking place MFC is low ~~high~~ indicating some function is taking place, once the function is complete MFC is high.

* **SEMICONDUCTOR RAM:**
  - It is a read and write memory
  - Volatile or temporary memory, since memory is lost once turned off.
  - They are of two types
    1. Static RAM: consists only semiconductor transistor
    2. Dynamic RAM: Data stored as charges thus capacit
    are used in dynamic RAM:

Internal Organisation of a simple RAM:



Sense and write circuits write data into the memory. Address can be further bifercated ~~for~~ into row address and column address.



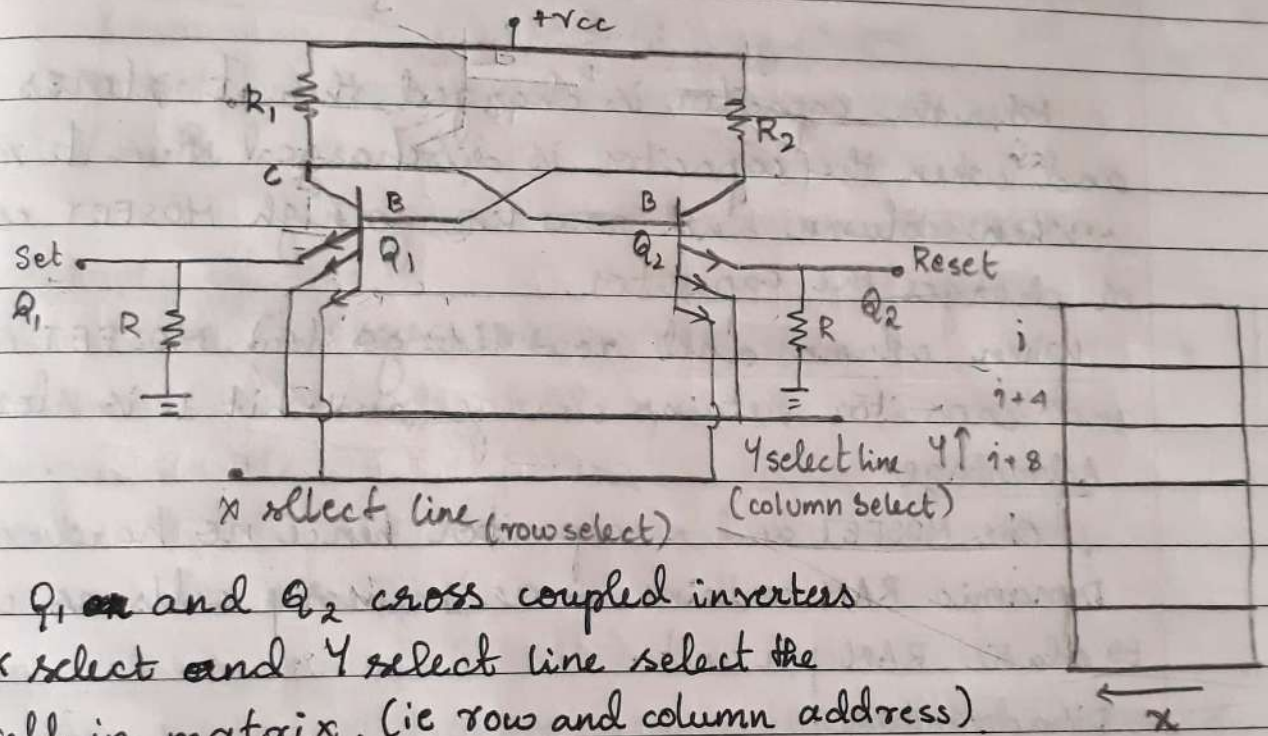
The first 5 address lines is associated to row address and the next 5 address lines is associated to column address

★ Static RAM:

- Electronic circuits consisting of transistors which are capable of retaining states as long as power is applied.
- Transistors can be biplolar junction transistor (BJT) or metal oxide semiconducta field effect transistor (MOSFET)

Transistor - Transistor logic cell (TTL cell)

- Multiemitter transistor



Q_1 and Q_2 cross coupled inverters

X select and Y select line select the cell in matrix. (ie row and column address)

1 is stored in cell when Q_1 is ON and Q_2 is OFF

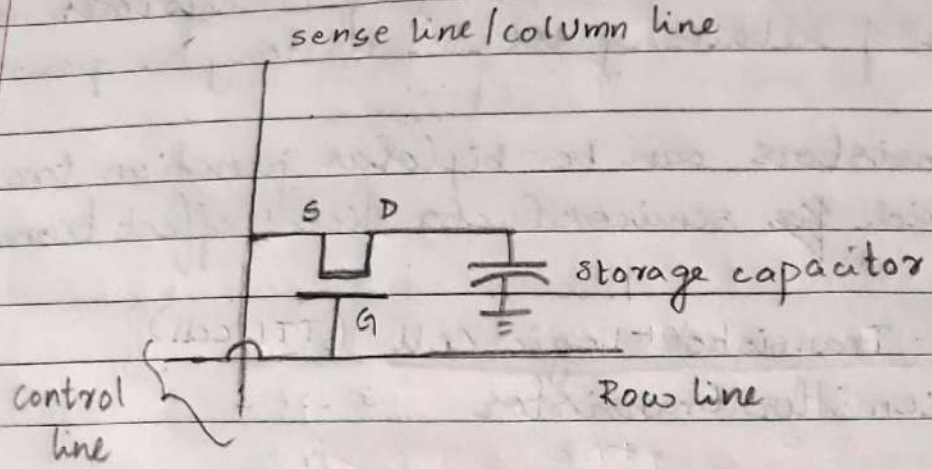
0 is stored in cell when Q_1 is OFF and Q_2 is ON.

Disadvantages

Hardware complexity

No capacitors are used to store data

* Dynamic RAM (DRAM): (Asynchronous)



When the capacitor is charged then it stores 1 and when the capacitor is discharged then it stores 0.

When column and row line go high MOSFET conducts or charges the capacitor.

When column and row line go low MOSFET opens and capacitor retains charge and bit 1 is stored in cell.

Advantage

One MOSFET and a capacitor hence no hardware complexity

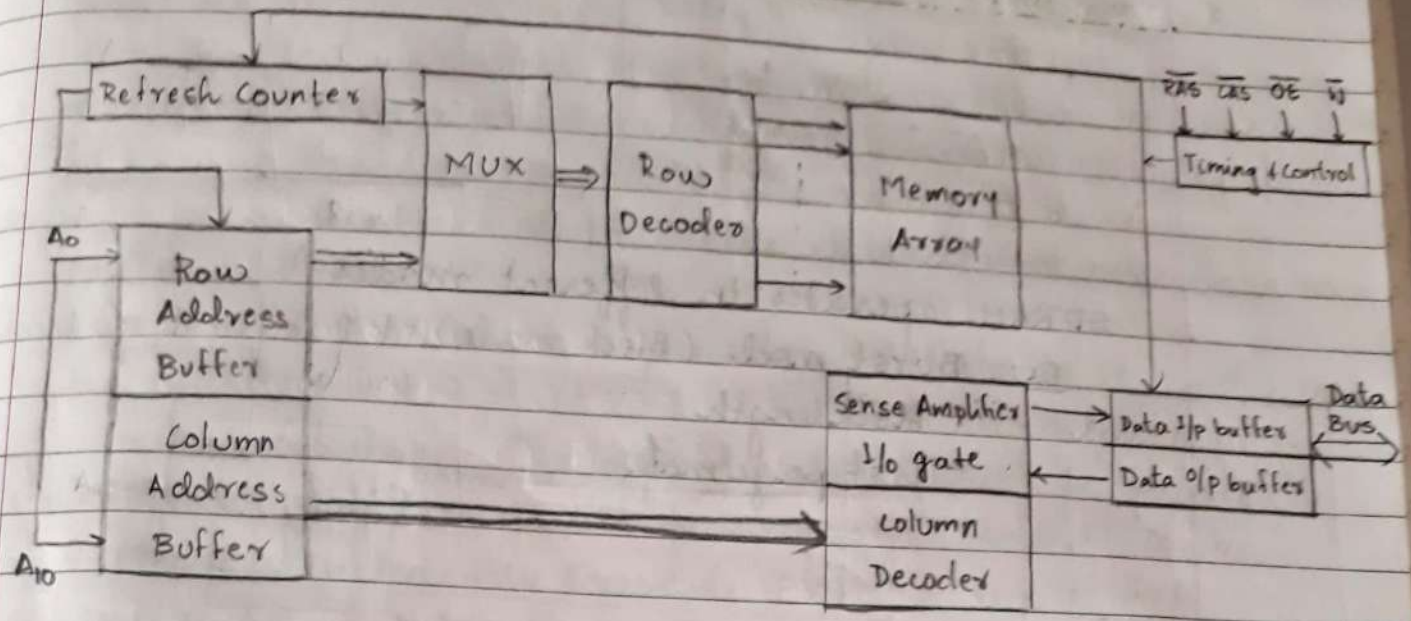
Dynamic RAM contains more memory cells as compared to static RAM per unit.

Disadvantage:

Refreshing of charge is required since the capacitor cannot retain charge for long.

SRAM	DRAM
- Less cells per unit area.	- More cells per unit area.
- Less access time, hence faster	- More access time, relatively slower
- consists of Flip Flop to store one bit of Information.	- Data is stored as charge in capacitor.
- Refreshing circuit is not required.	- Refreshing circuit is required
- Cost is more	- cost is relatively less.

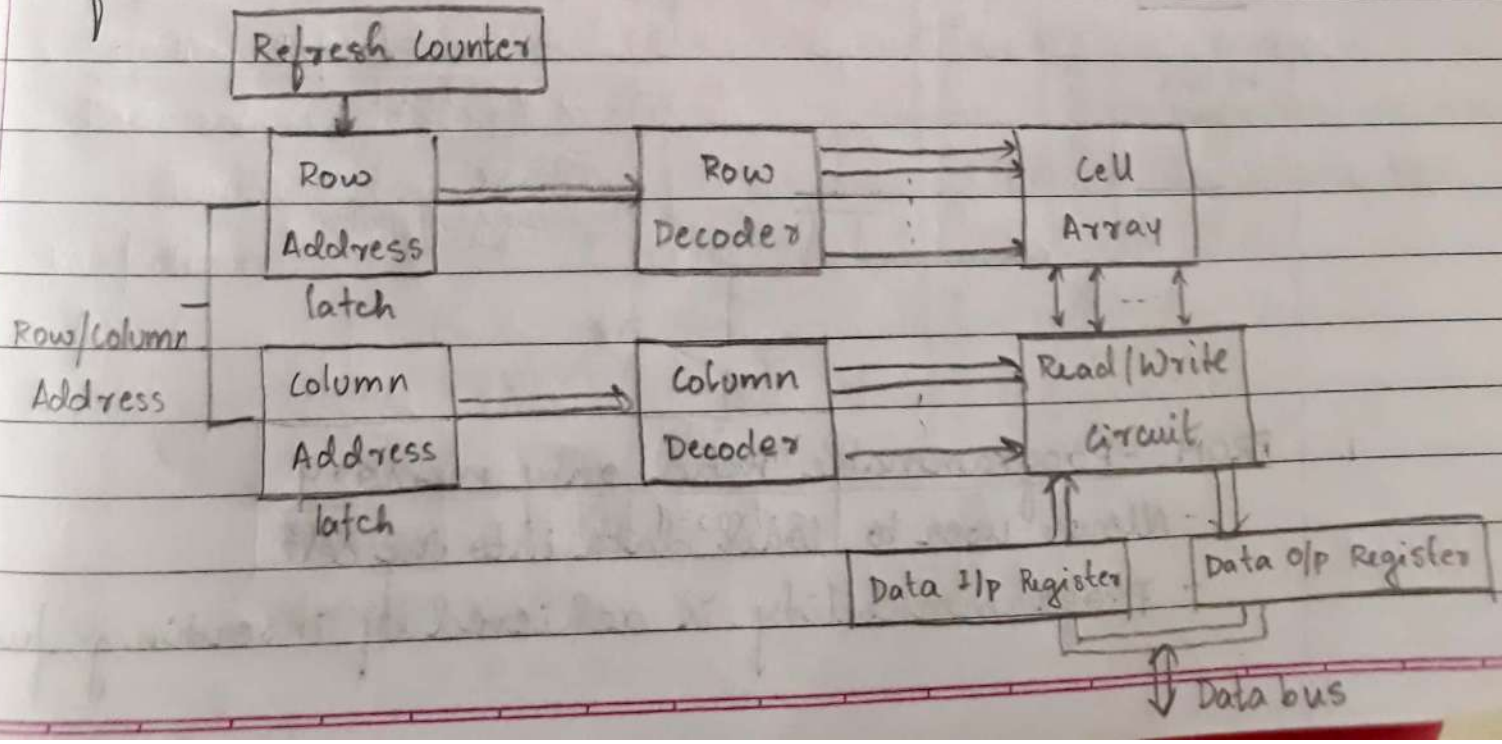
DRAM organisation:

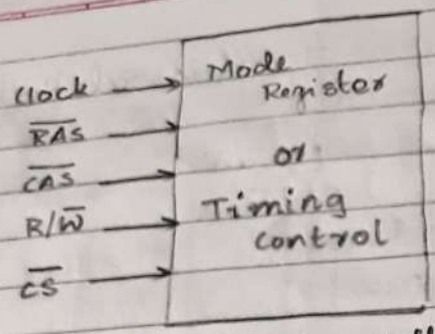


control signals
Row address select ($\overline{RAS}$)
Column address select ($\overline{CAS}$)
Output Enable ($\overline{OE}$)
write ($\overline{W}$)

* Synchronous DRAM : (SDRAM)

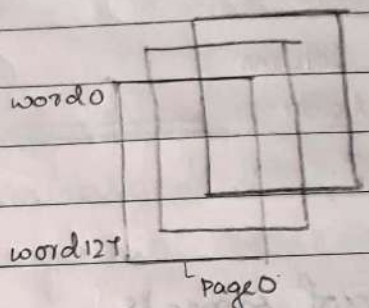
- operation of SDRAM is directly synchonized with reference clock.





SDRAM operates in different modes

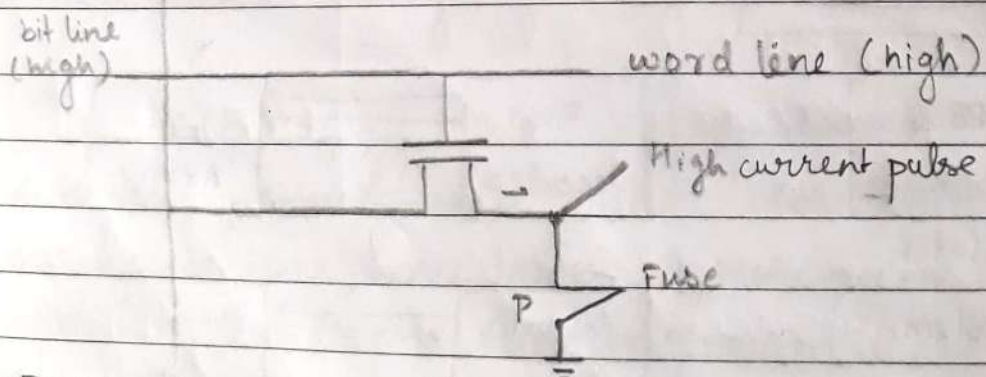
Ex: Burst mode (Block mode) - to transfer large blocks of data
- block length
- fast page mode



★ Double Data Rate SDRAM: (DDRSDRAM):

- ~~DDRSDR~~ The data is transfered at bothe rising and falling edge of the clock signal, i.e., the memory devise accesses cell array on both edges of clock.
- speed is double to that of SDRAM.
- Bandwidth is double to that of SDRAM.

★ ROM:



1. PROM - Programmable Read only memory
   - Allows user to load data into the cell.
   - Programmability is achieved by inserting fuse at

point P. Before it is programmed memory cells contains only 0's.

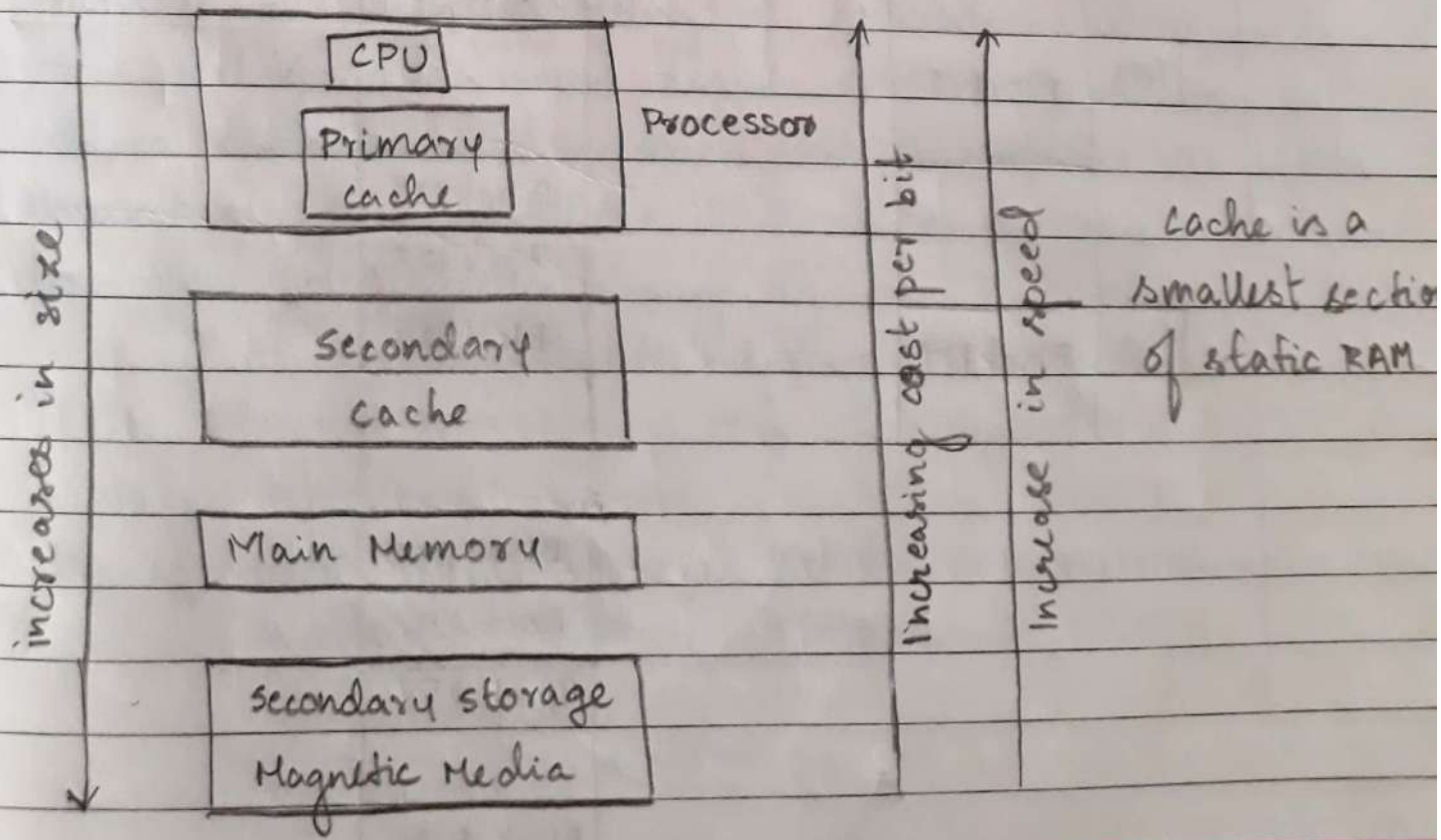
- The user can insert 1 at required location by burning the fuses using current pulses.

2. EPROM : Erasable PROM:

- Allows stored data to be erased and new data to be loaded.
- Special transistor: it permits the data to be written into the memory.
- Disadvantage: The memory chips must be physically removed from the circuit for reprogramming.

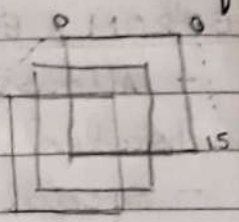
3. EEPROM: Electrically Erasable PROM:

- Data can be rewritten electrically without having to physically remove the memory chip.
- Disadvantage: Different voltage levels are required for erasing, writing and reading the data, so complexity increases.
- Flash memory - high density, low cost, less power



* Mapping Functions:
- Methods to specify where memory blocks are placed in cache memory.

Ex: Assuming a cache consisting of 128 blocks of 16 words each. Hence the size of cache is $128 \times 16 = 2048$ words
$\Rightarrow [1024 \times 2] = 2k$ words

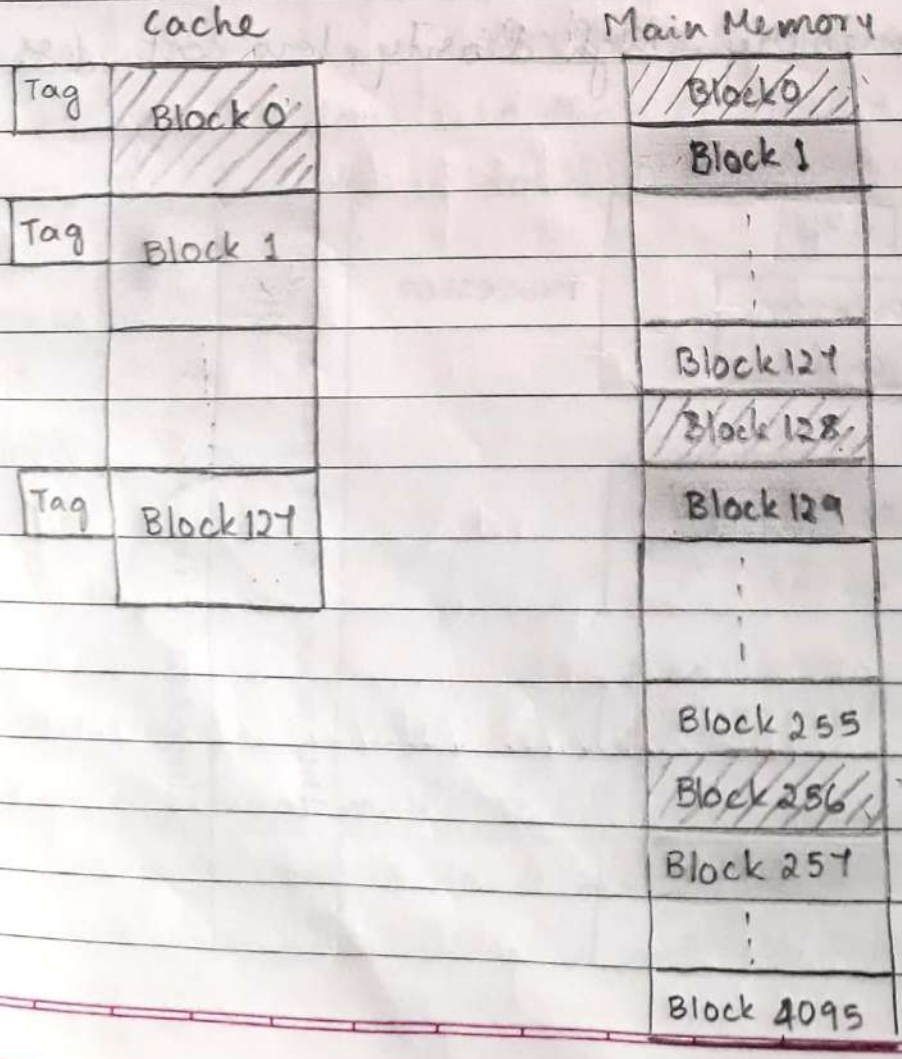


Main memory is address by 16 bit address $\Rightarrow 2^{16} = 65536 = 64k$ words

1. (Direct Mapping)

Each block from main memory has only one possible location in cache.

Block J of main memory maps onto Block J modulo 128 of cache whenever one of blocks of main memory, 0, 128, 256.... it is stored in Block 0 of cache
Blocks 1, 129, 257 $\Rightarrow$ cache Block 1 and so on.



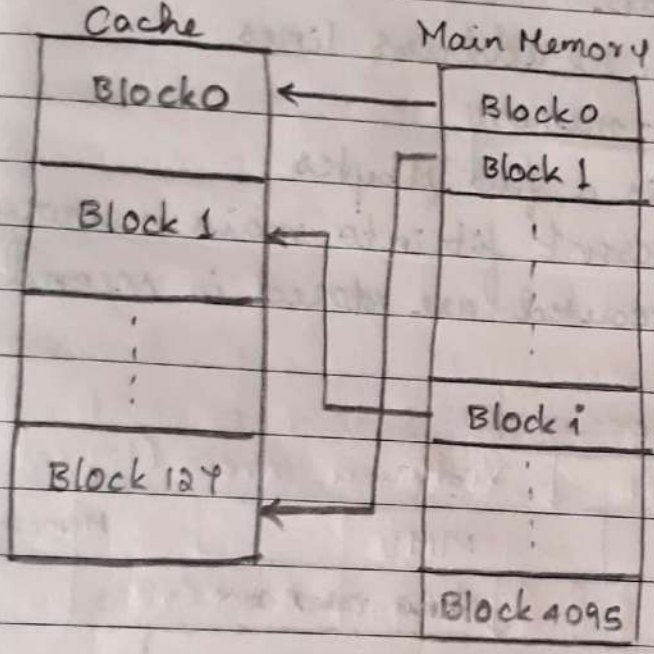
Each block contains 16 words and there are 128 blocks in cache.
Address field

Tag	Block	Words
5 bits	7 bits	4 bits
	$128 = 2^7$	$16 = 2^4$

Main memory = 16 bits

2. (Associate Mapping)

Main memory block can be placed into any cache position.



Address Field
Main memory = 16 bits

Tag	Word
12 bits	4 bits
	$16 = 2^4$

Block field is zero as any block can be placed in any position in the cache.

Q: consider a cache consisting of 256 blocks of 16 words each for a total of 4096 words. Assume that main memory is addressed by 16 bit address and it contains 4k blocks. How many bits are there in each of the tag, block and word fits for i. direct mapping
ii. associate mapping.

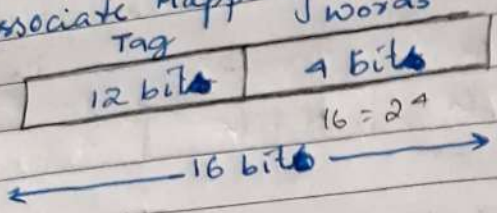
i. Direct Mapping

Tag	Block	Words
4 bits	8 bits	4 bits
	$256 = 2^8$	$16 = 2^4$

Main memory - 16bits

← 16 bits →

ii. Associate Mapping



Main memory - 16 bits

* Virtual Mapping:

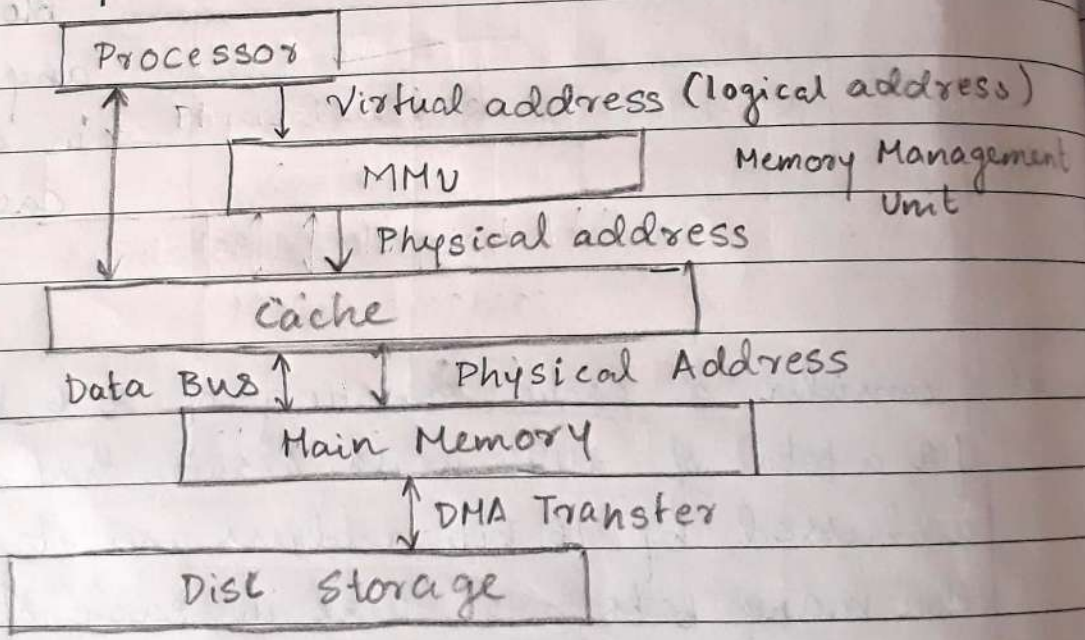
Physical main memory is not as long as address spanned by processor.

Ex: Processor with 32 address lines

2^{32} = 4Gb - memory

If Main memory is a few Mbytes.

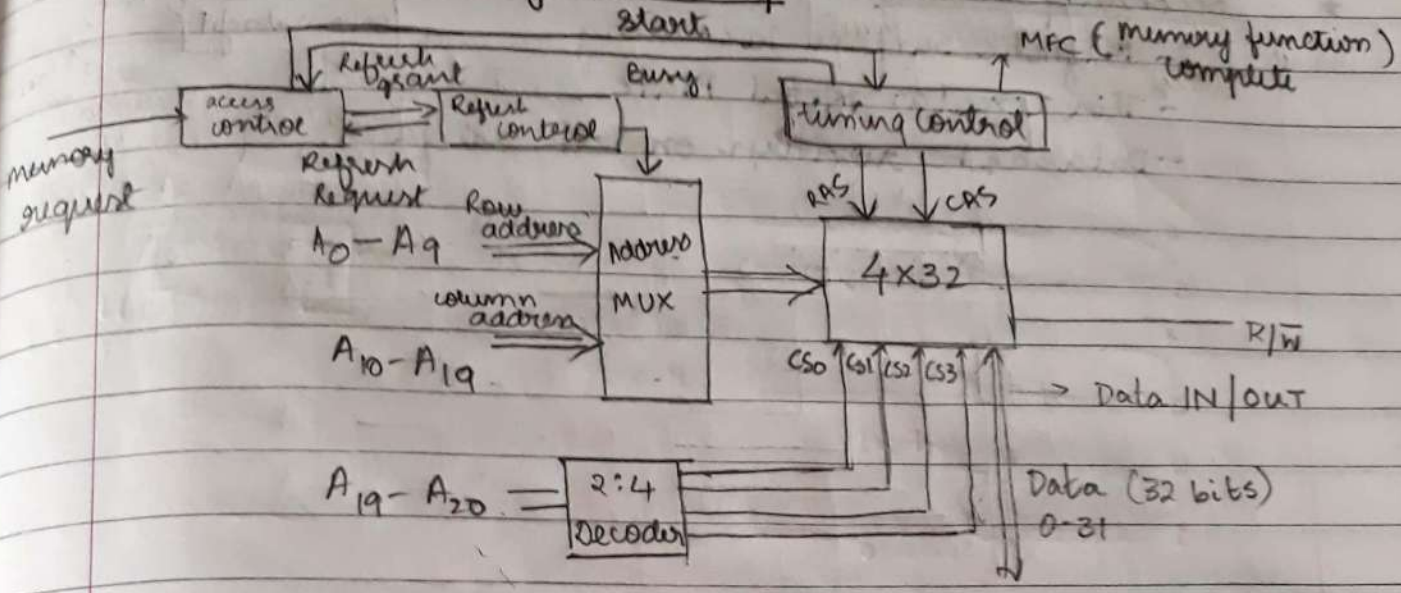
when program doesnot fit into main memory parts of it not currently executed are stored in secondary storage.



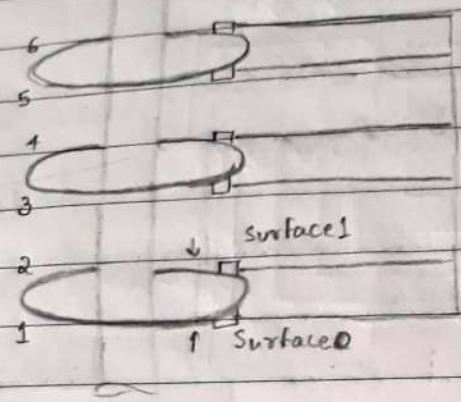
Virtual address - binary address that the processor issues this is converted into physical address by MMU. This is known as Virtual Memory technique.

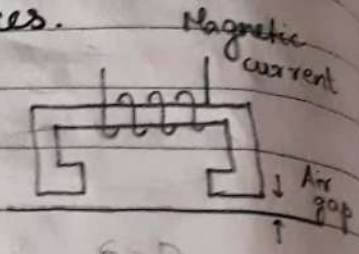
MMU - translator converts virtual address into physical address.

* **Structure of Larger Memory:**



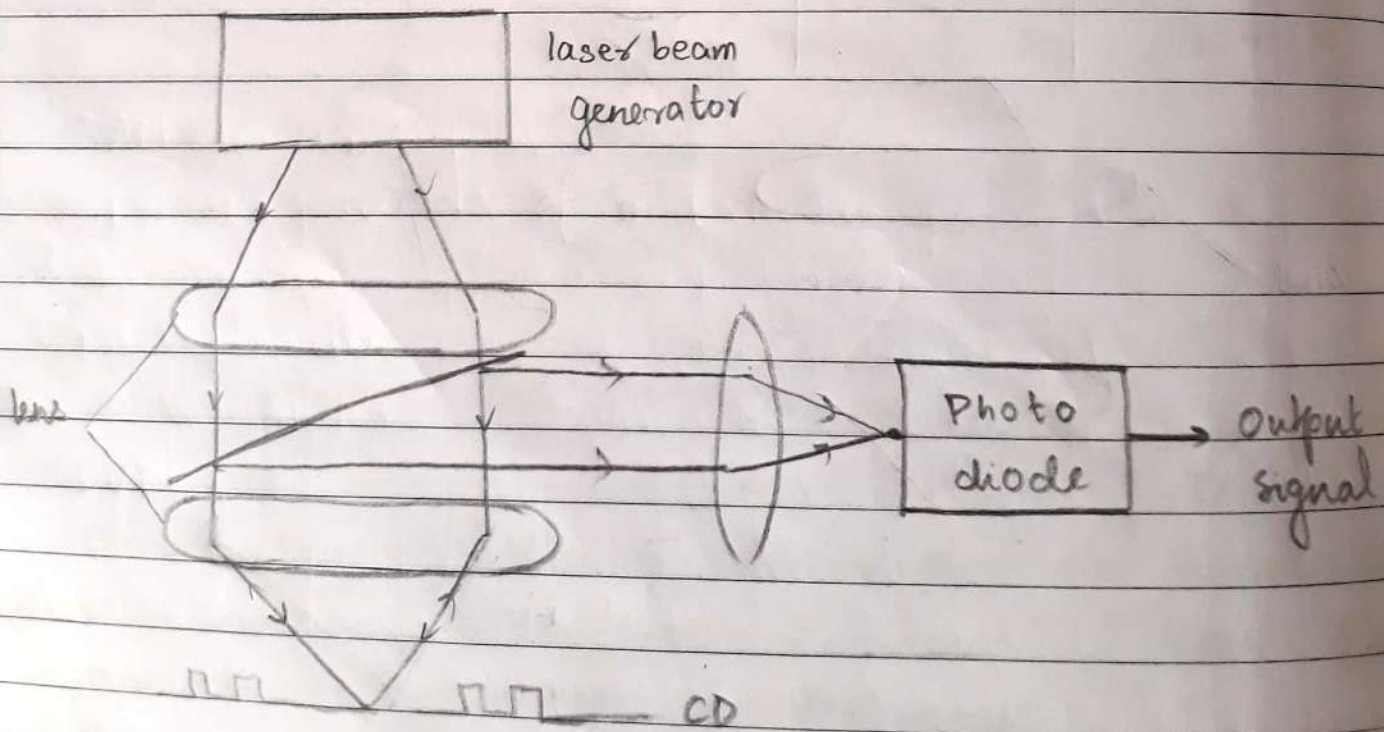
* **Secondary Storage:**
  - Magnetic Surface Recording
  - Thin Circular Metal Plate
  - Data can be written on both surfaces.





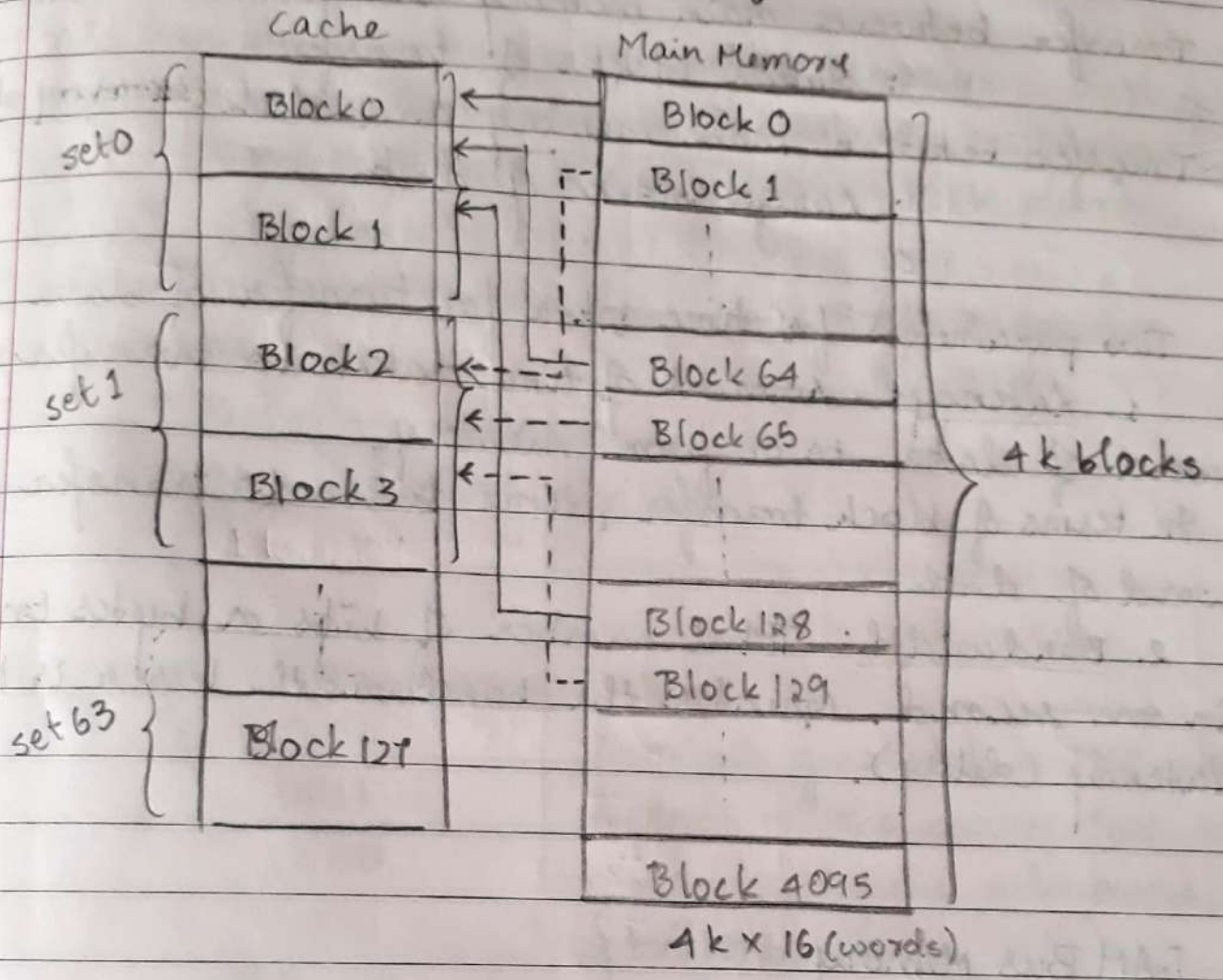
Optical Disk: CD ROM

- Surface is coated with highly reflective ~~method~~ material.
- Laser is used to write data into disk and is also used to read data.



3. (Set Associate Mapping)

combination of direct mapping as well as associate mapping - set associate mapping.

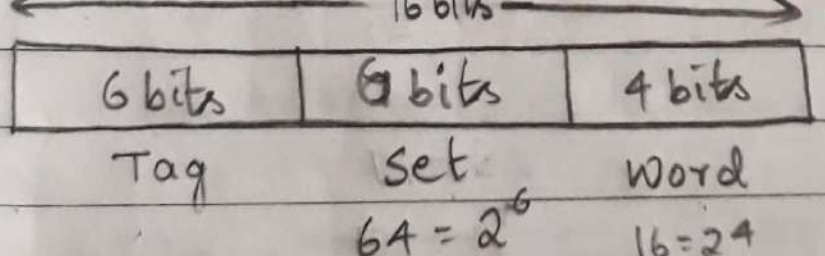


In cache two blocks together form a set.

⇒ Block 0, 64, 128 ..... of the main memory → set 0.

⇒ Block 1, 65, 129 ..... of the main memory → set 1.

Main memory address



* Performance of Memory:
Performance measure

- Transfer between main memory and processor
  : Small blocks of transfer
- Transfer between main memory and disk (secondary storage)
  : Large blocks of data

Two parameters for time taken for transfer of data

1. Latency: Amount of time taken to transfer a word of data to or from memory.
In terms of block transfer, time taken to transfer first word of data.

2. Bandwidth: Total number of bits or bytes transferred in one second. Greater the bandwidth lesser is the latency (delay).

RAM Bus Memory

Hit rate: percentage of instruction present in the cache during repeated computation.

Miss penalty: when the instruction is not present in the cache, it take more time to read from main memory

DVD

Page No.:

UNIT - 5

Arithmetic

- Addition and Subtraction of signed numbers : Fast Adders.
- Multiplication of unsigned numbers (+ve numbers)
  - * Combinational circuit (Braun Multiplier)
  - * Sequential circuit
- Signed number multiplication - Booth Algorithm.
- Division.
  - * Restoring division
  - * Non-restoring division

* Design of Fast Adders.

Ex:
```
  0101
  0011
 ------
  1000
  ↑  ↑
 MSB LSB
```

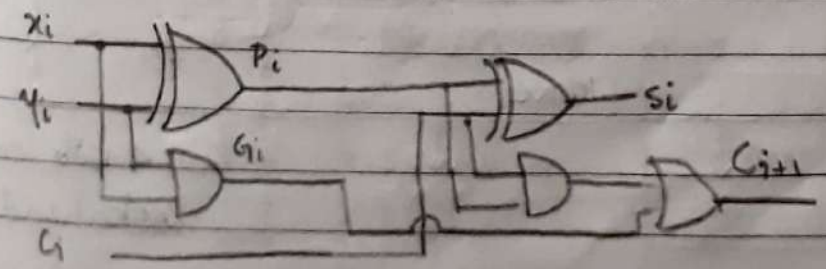
Sum bit generated in MSB position depends on the carry that was generated by the additions in previous positions.

- Carry lookahead Adders:

Method of speeding up addition process by eliminating interstage carry delay.

P_i - carry propagator
G_i - carry generator

Full Adder.



$G_i = x_i y_i$ $\quad P_i = x_i \oplus y_i$

$S_i = P_i \oplus C_i$ $\quad C_{i+1} = G_i + P_i C_i$

$i = 0$ to 3 (4 bits)

Carry

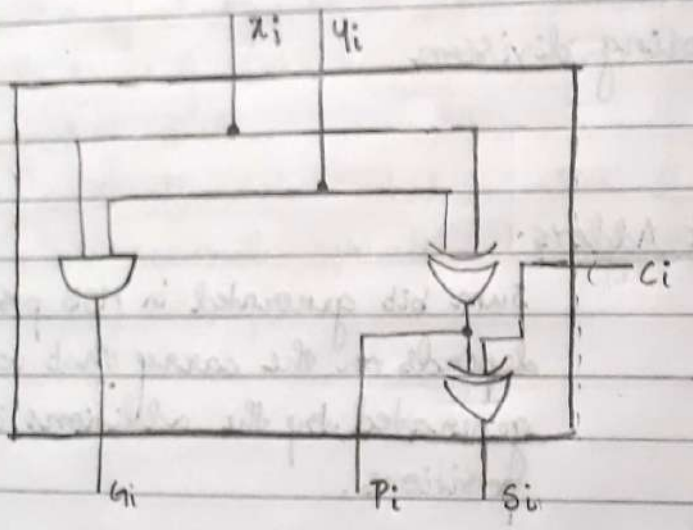
$C_1 = G_0 + P_0 C_0$ — (1)

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$ — (2) From (1)

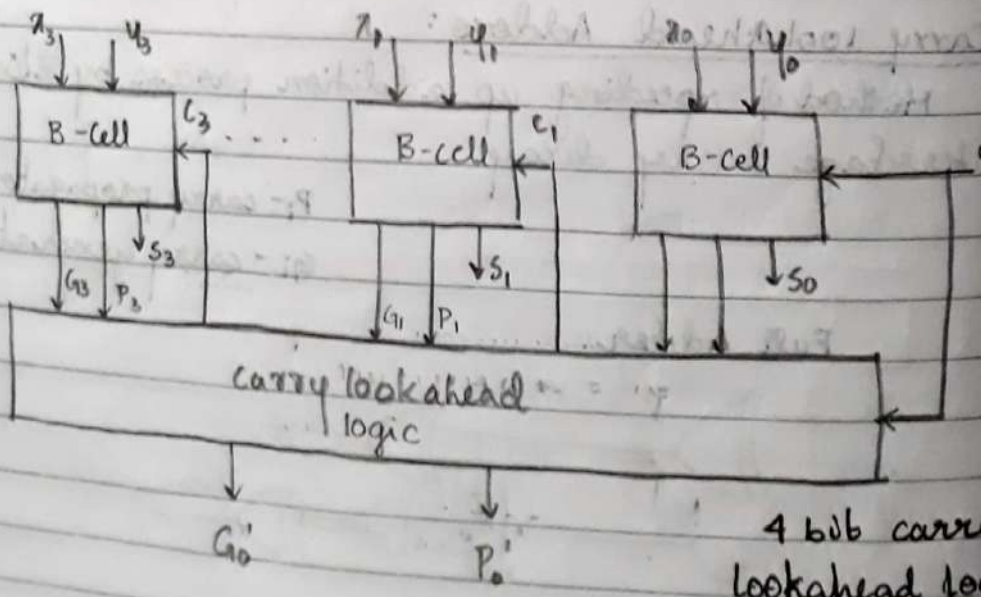
$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$ — (3) From

$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$ — (4) From

Hence the number of gates keeps on increasing the delay increases reducing the performance



Bit-stage cell (B-cell)



4 bit carry lookahead logic

Multiplication of unsigned numbers:

Multiplier : Q
Multiplicand : M
Product : P

Algorithm:

If the multiplier bit is 1 (LSB), then the multiplicand are entered in appropriate position to be added to partial product.
If the multiplier bit is 0 (LSB), then zeros are entered.

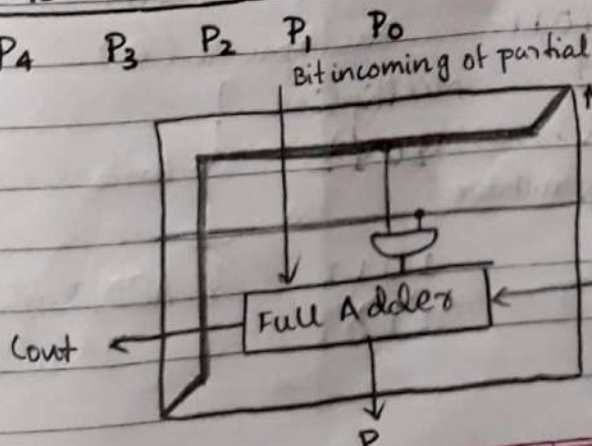
Ex: $(13)_{10} = 1101$: Multiplicand (M)
$(11)_{10} = 1011$: Multiplier (Q)

```
  1101
  1101
 0000
1101
--------
(10001111)2 = (143)10
```

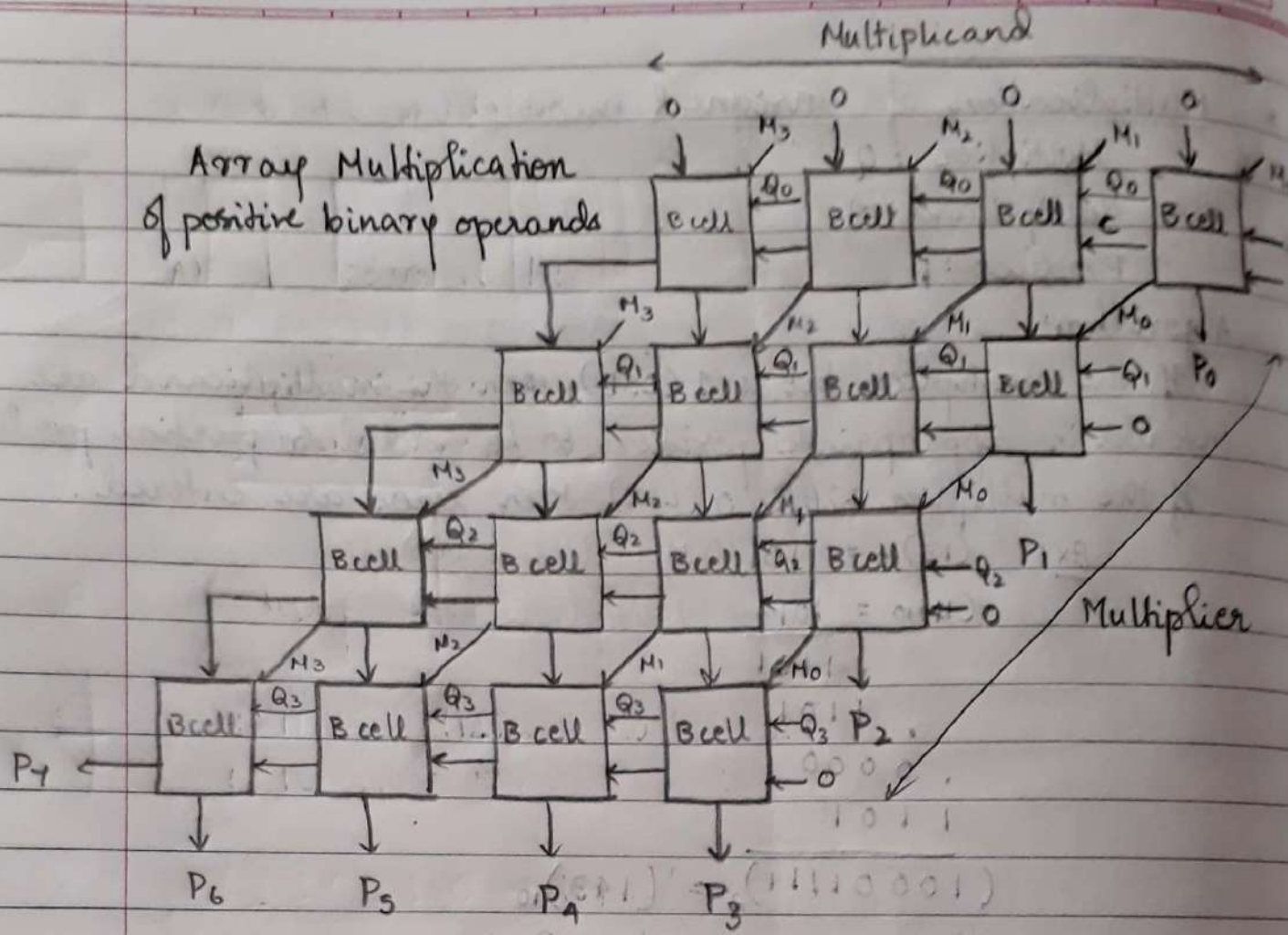
Multiplicand M: $M_3\ M_2\ M_1\ M_0$
Multiplier Q: $Q_3\ Q_2\ Q_1\ Q_0$

$M_3\ M_2\ M_1\ M_0$
$Q_3\ Q_2\ Q_1\ Q_0$
$M_3Q_0\ \ M_2Q_0\ \ M_1Q_0\ \ M_0Q_0$
$M_3Q_1\ \ M_2Q_1\ \ M_1Q_1\ \ M_0Q_1$
$M_3Q_2\ \ M_2Q_2\ \ M_1Q_2\ \ M_0Q_2$
$M_3Q_3\ \ M_2Q_3\ \ M_1Q_3\ \ M_0Q_3$
$P_7\ \ P_6\ \ P_5\ \ P_4\ \ P_3\ \ P_2\ \ P_1\ \ P_0$

two 4 bit numbers are multiplied hence product is 8 bit number.



Bit stage cell (B-cell)



* Sequential circuit for Binary Multiplication:

Algorithm:

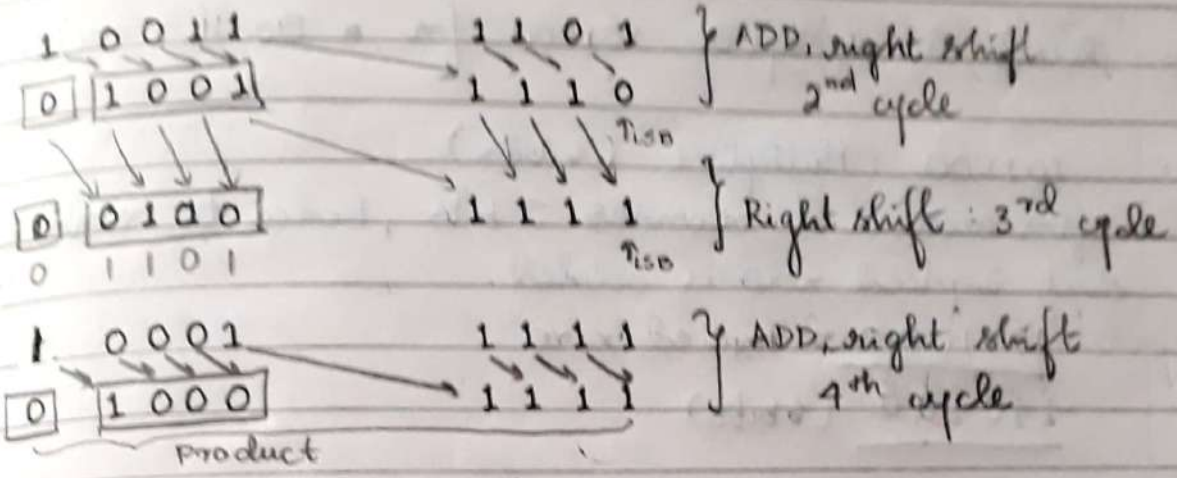
LSB of multiplier (Q) is checked, if LSB (Q_0) is 1 Multiplicand (M) and partial products are added and all bits are shifted (right).

If LSB of multiplier is 0, no addition but all bits are right shifted.

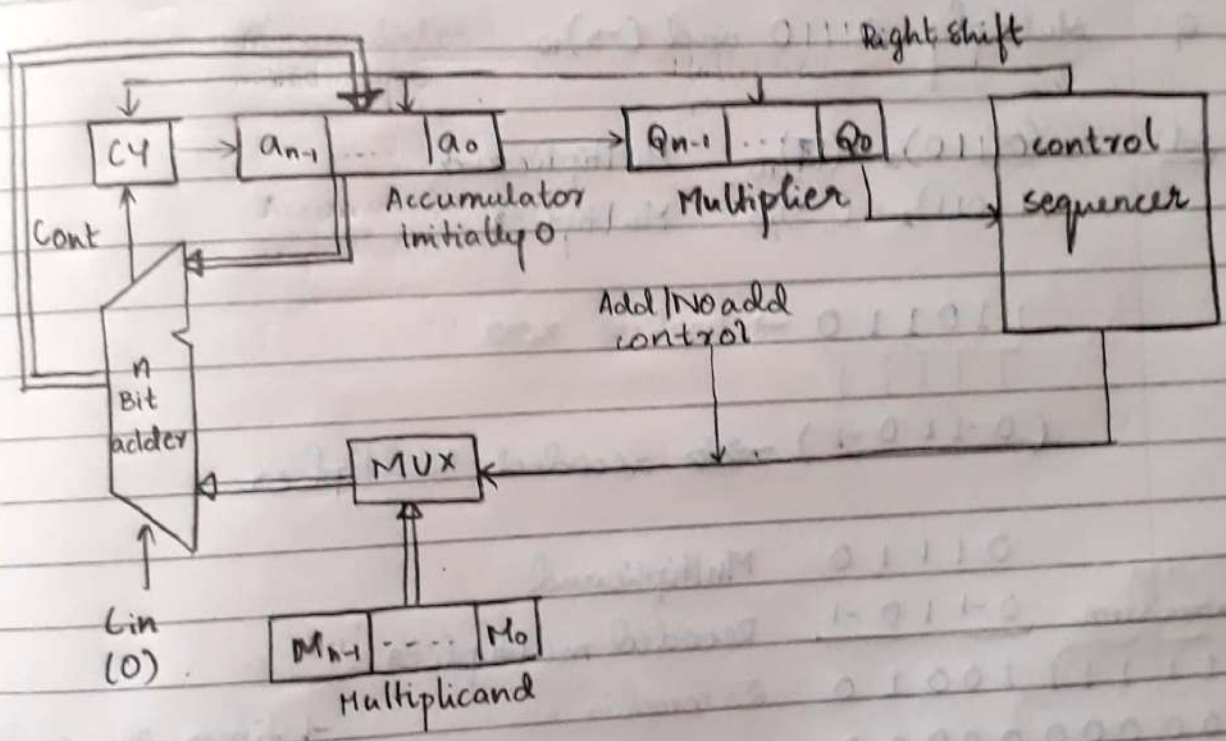
Ex: $(13)_2$ = 1101 : Multiplicand (M)

$(11)_2$ = 1011 : Multiplier (Q)

Multiplicand	Multiplier	
1101	1011	Initial configuration
C4 | 0 | 0000 accumulator	↑LSB	
0 1101	1011	ADD, Right shift 1st cycle
0 | 0110	1101	
+ 0 1101	↑LSB	



$(1000\ 1111)_2 = (143)_{10}$



* Multiplication of signed numbers: Booth's Algorithm

-1 times the multiplicand is selected when moving from 0 to 1. (2's compliment of multiplicand is added).

+1 times the multiplicand is selected when moving from 1 to 0.

0 times the multiplicand is selected when moving from 1 to 1. or 0 to 0.

Q: Recode the multiplier 101100 for a Booth's multiplication.

Sol: 101100 : Multiplier (6 bits)
After recoded it becomes 5 bits, hence to avoid that implied zero is added.

1011000 → Implied zero
-1 +1 0 -1 0 0 (6 bits)

Q: Multiply 01110 and $(-5)_{10}$

Sol: $(01110)_2 = (14)_{10}$ Multiplicand
$(11011)_2 = (-5)_{10}$ Multiplier.

1 1 0 1 1 0 → Implied zero

(0 -1 1 0 -1) recoded multiplier

	0 1 1 1 0	Multiplicand
sign extension	0 -1 1 0 -1	Recoded multiplier.
1 1 1 1 1	1 0 0 1 0	2's compliment
0 0 0 0 0	0 0 0 0 0	0 times the multiplicand
0 0 0 0 1	1 1 0	1 times the multiplicand
sign extension 1 1	1 0 0 1 0	2's compliment
+ 0 0	0 0 0 0	
CY=1	1 1 1 0 1 1 1 0 1 0	

-1 times the multiplicand
01110
10001 1's compliment
10010 2's compliment

$(1110111010)_2 = (-70)_{10}$

Q: Given: Multiplier 001100
Multiplicand 010011
Perform multiplication using Booth's algorithm

Sol: $(001100)_2 = (12)_{10}$

$(010011)_2 = (19)_{10}$

0 0 1 1 0 0 0 → Implied zero

0 1 0 -1 0 0 → recoded multiplier

```
                0 1 0 0 1 1   - Multiplicand
              0 1 0 -1 0 0    - recoded multiplier
------------------------------
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 0 1 1 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 1
+ 0 0 0 0 0 0 0 0
1] 0 0 0 0 1 1 1 0 0 1 0 0
```

Sign extension

CY (neglected)

Multiplicand
010011
101100 - 1's compliment
101101 - 2's compliment

$(11100100)_2 = (228)_{10} = (12)_{10} \times (19)_{10}$

★ Division:

1. Non restoring division

If the sign of the accumulator (A) is 0, shift A and Q left by one bit and subtract divisor from A.

If the sign of the accumulator (A) is 1, shift A and Q left by one bit and add divisor with A.

If the sign of A is 0, set q_0 to 1.

If the sign of A is 1, set q_0 to 0.

Ex: $(1000)_2 = (8)_{10}$; dividend
$(0011)_2 = (3)_{10}$; divisor

Divisor : 0011
1's comp: 1100
2's comp: 1101

	Accumulator	Dividend	
Initially	0 0 0 0 0	1 0 0 0	
	0 0 0 1 1		
Shift left	0 0 0 0 1	0 0 0 □	Cycle 1
Subtract	1 1 1 0 1		
Set $q_0 \rightarrow 0$	(1) 1 1 1 0	0 0 0 [0]	
Shift left	1 1 1 0 0	0 0 [0] □	Cycle 2
Add	0 0 0 1 1		
Set $q_0 = 0$	(1) 1 1 1 1	0 0 [0] [0]	
Shift left	1 1 1 1 0	0 [0] [0] □	Cycle 3
Add	0 0 0 1 1		
Set $q_0 = 1$	(0) 0 0 0 1	0 [0] [0] [1]	
Shift left	0 0 0 1 0	[0] [0] [1] □	Cycle 4
Subtract	1 1 1 0 1		
Set q_0 =	(1) 1 1 1 1	[0] [0] [1] [0]	

Quotient

To restore remainder

Add	1 1 1 1 1	Accumulator
	0 0 0 1 1	Divisor
	0 0 0 1 0	

Remainder

2. Restoring Division:

Algorithm

shift A and Q left one bit

Subtract M from A, place result in A

If sign of A is 1 set q_0 to 0, add M back to A

If sign of A is 0 set q_0 to 1.

Ex:

Ex: $(1000)_2 = (8)_{10}$: dividend

$(0011)_2 = (3)_{10}$: divisor

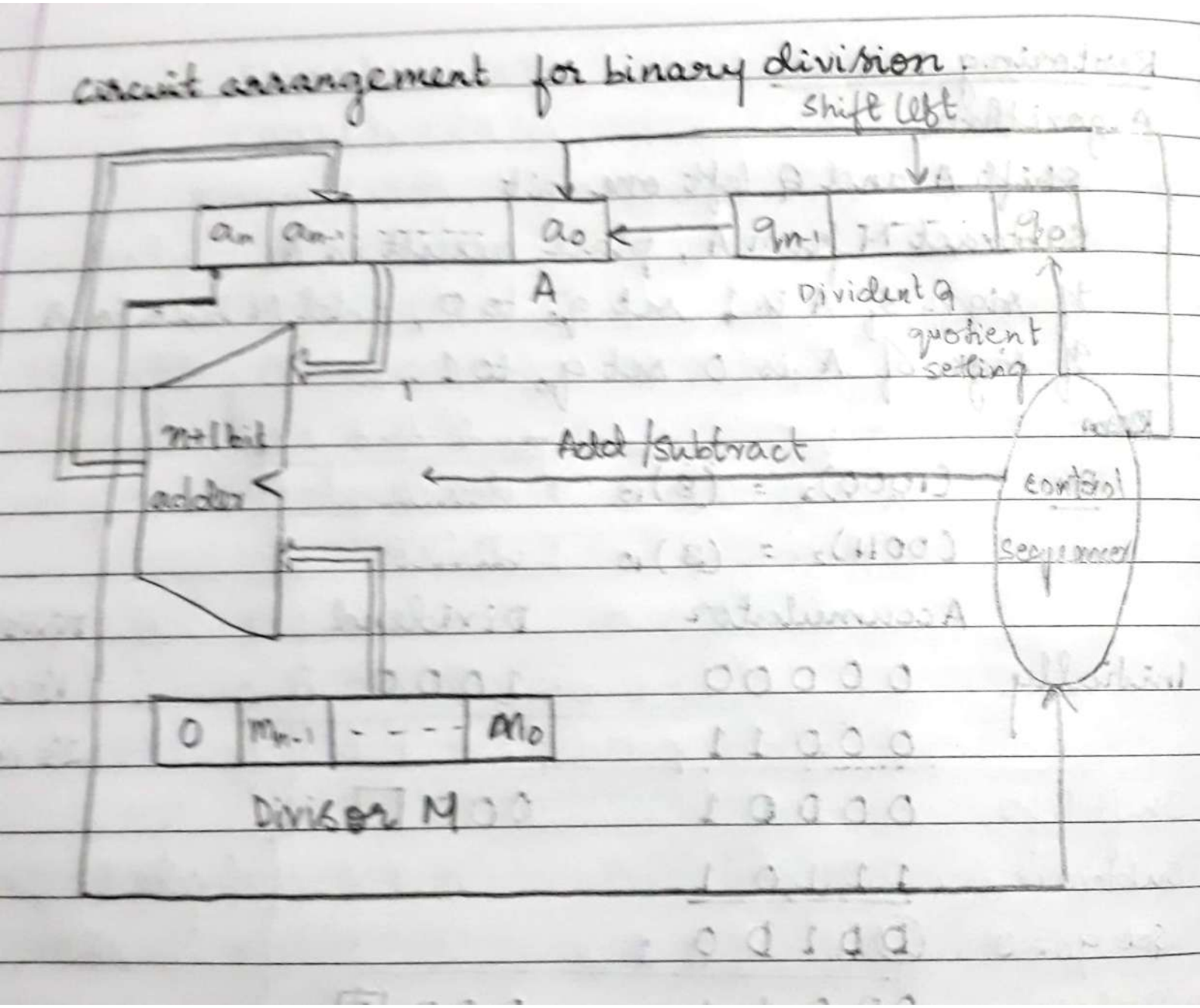
Divisor : 00011

1's comp = 11100

2's comp = 11101

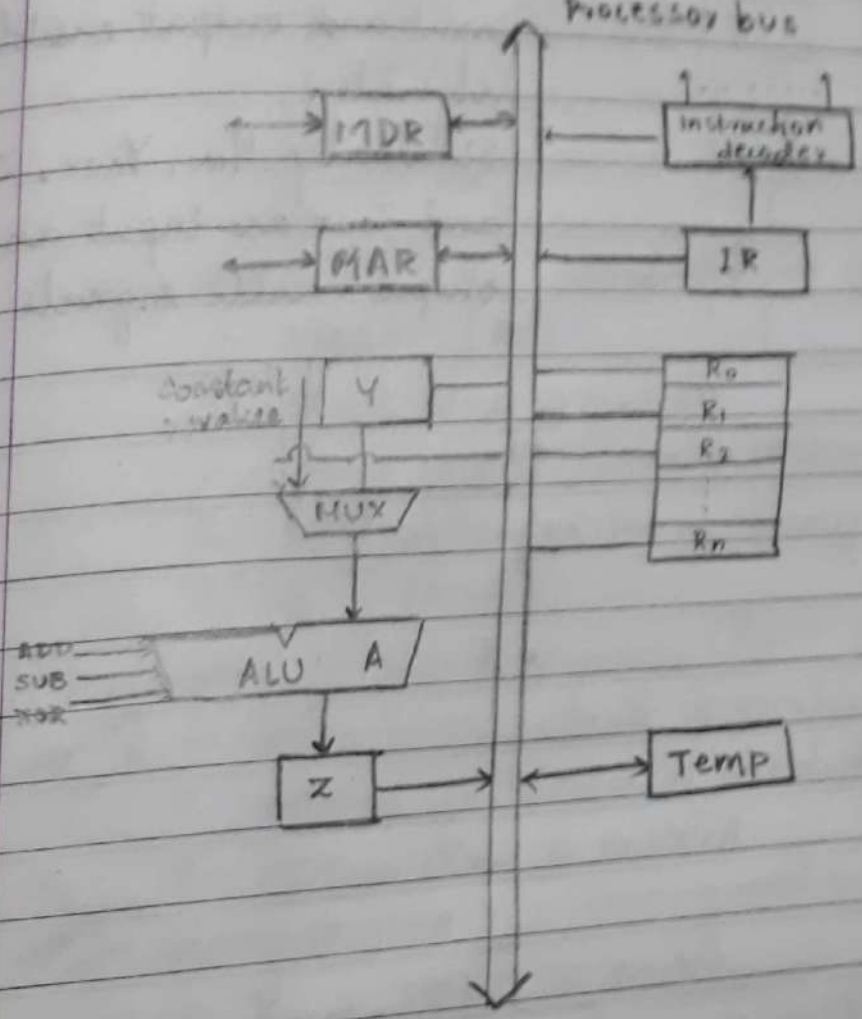
	Accumulator	Dividend	
Initially	0 0 0 0 0	1 0 0 0	
	0 0 0 1 1		
Shift left	0 0 0 0 1	0 0 0 □	
Subtract	1 1 1 0 1		Cycle 1
Set q_0 = 0	(1) 1 1 1 0		
Restore	0 0 0 1 1		
	0 0 0 0 1	0 0 0 [0]	
Shift left	0 0 0 1 0	0 0 [0] □	
Subtract	1 1 1 0 1		Cycle 2
Set q_0 = 0	(1) 1 1 1 1		
Restore	0 0 0 1 1		
	0 0 0 1 0	0 0 [0] [0]	
shift left	0 0 1 0 0	0 [0] [0] □	
Subtract	1 1 1 0 1		Cycle 3
Set q_0 = 1	(0) 0 0 0 1		
	0 0 0 0 1	0 [0] [0] [1]	
shift left	0 0 0 1 0	[0] [0] [1] □	
Subtract	1 1 1 0 1		Cycle 4
Set q_0 = 0	(1) 1 1 1 1		
Restore	0 0 0 1 1		
	0 0 0 1 0	[0] [0] [1] [0]	
	remainder	quotient	

Circuit arrangement for binary division :



UNIT - 6

Basic Processing Unit and Embedded Systems



Single bus architecture

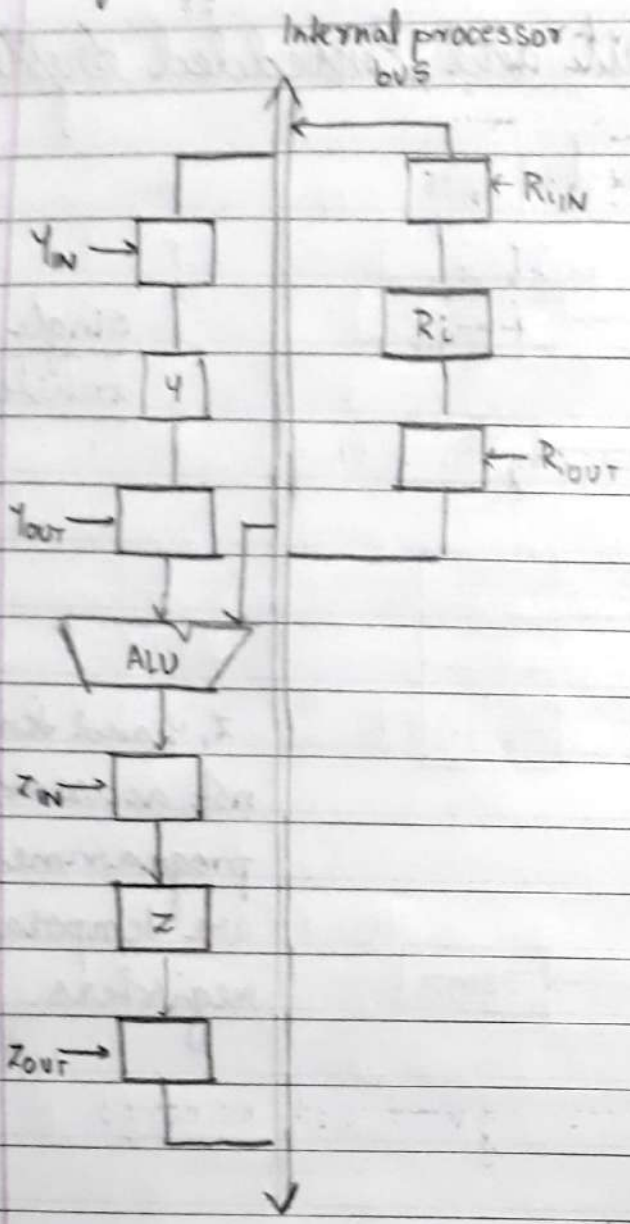
Z, Y and temp are not accessable to the programmer. as they are temporary registers.

Demerits:

Time consuming as a single bus is used.

* **Microoperations:**

* Register Transfers:



$R_{i_{IN}}$ and $R_{i_{OUT}}$ are input and output enable signals.
Similarly Y_{IN}, Y_{OUT}, Z_{IN} and Z_{OUT} are input and output enable signals.

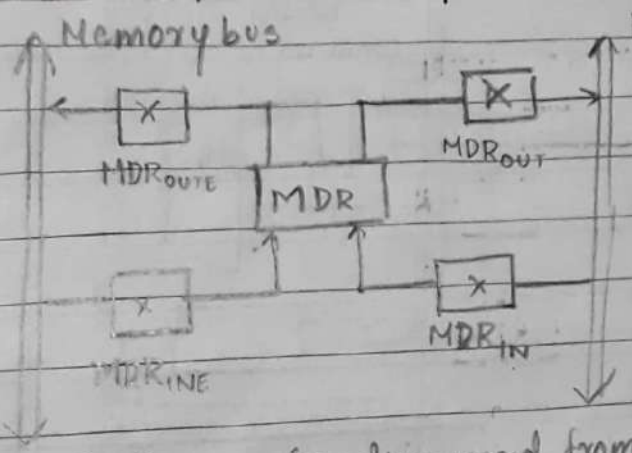
* Performing an arithmetic or logical operation:

Ex: ADD, R_1, R_2, R_3

1. $R_{1_{OUT}}$ and Y_{IN} is enabled
2. $R_{2_{OUT}}$ and Select Y, ADD, then Z_{IN} is enabled.
3. $R_{3_{IN}}$ and Z_{OUT} is enabled.

} Micro operations

* Fetching a word from memory:



Ex: MOVE (R1), R_2 (Reading word from the memory)

1. MAR ← [R_1]
2. Start read operation
3. Wait for MFC (Memory function complete) response of memory.
4. MDR is loaded from memory bus
5. R_2 ← [MDR]

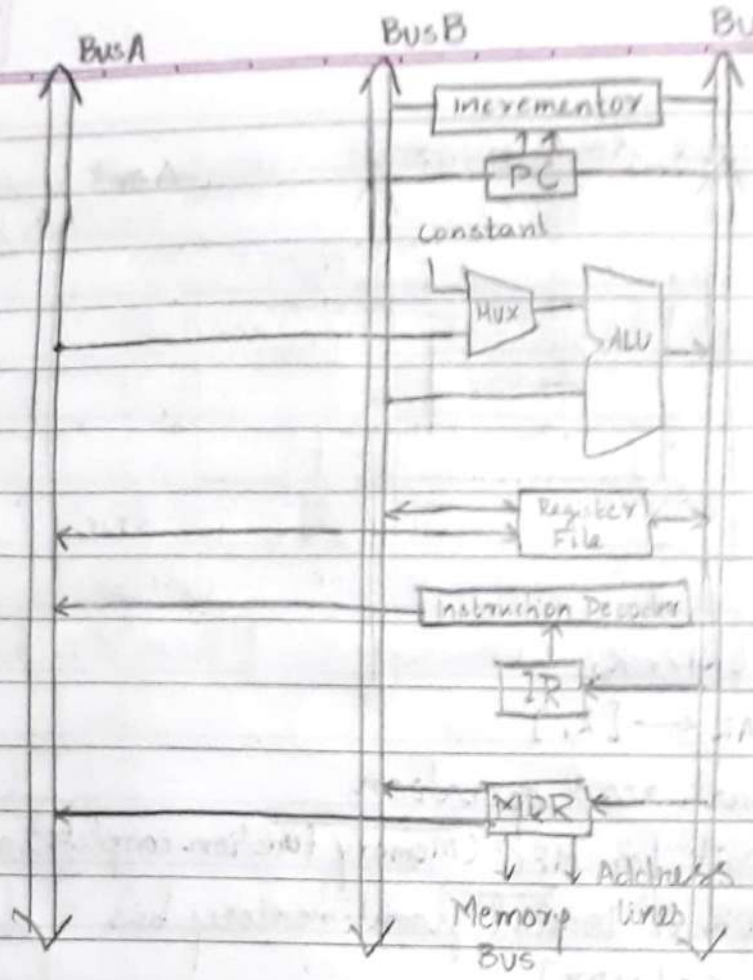
In terms of control sequency (Memory unit Operation)

Memory read
1. R_{1OUT}, MAR_{IN}, Read is enabled.
2. $MDR_{INEnable}$, WMFC is enabled
3. MDR_{OUT}, R_{2IN} is enabled

Memory write — MOVE R_2, (R_1) (storing word to the memory)
1. R_{2OUT}, MAR_{IN} is enable.
2. MDR_{IN}, Write, R_{2OUT} is enabled
3. MDR_{OUT}, WMFC

* Multiple Bus Organisation:

3 bus architecture

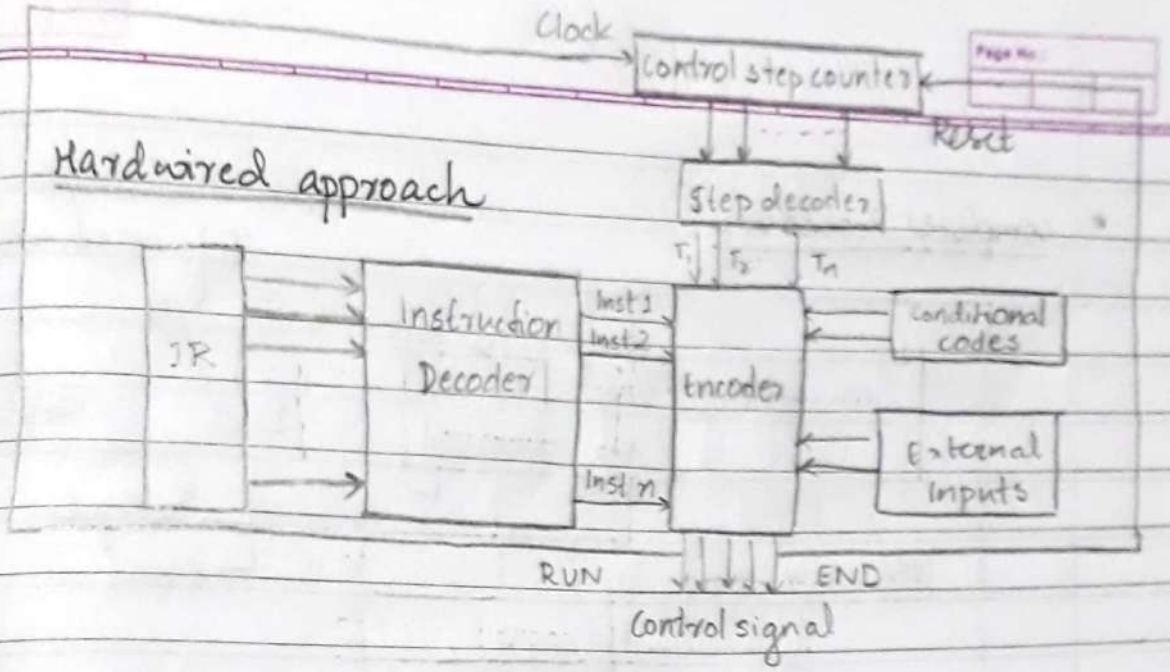


Here same bus is used to fetch instruction as well as to fetch data (limitations)

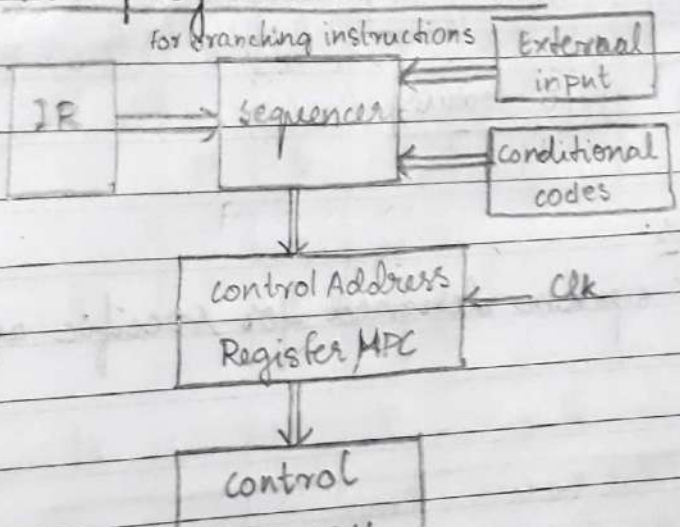
Incrementor increments the program counter to increment the address (i.e, i+4, i+8 ...). This avoids the involvement of ALU to increment the program counter increasing the performance.

* Generating control signals:
  1. Hardwired approach
     Better performance
  2. Microprogrammed control
     simple circuit, but low performance.

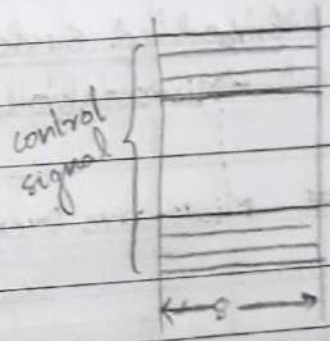
Hardwired approach



Microprogrammed control

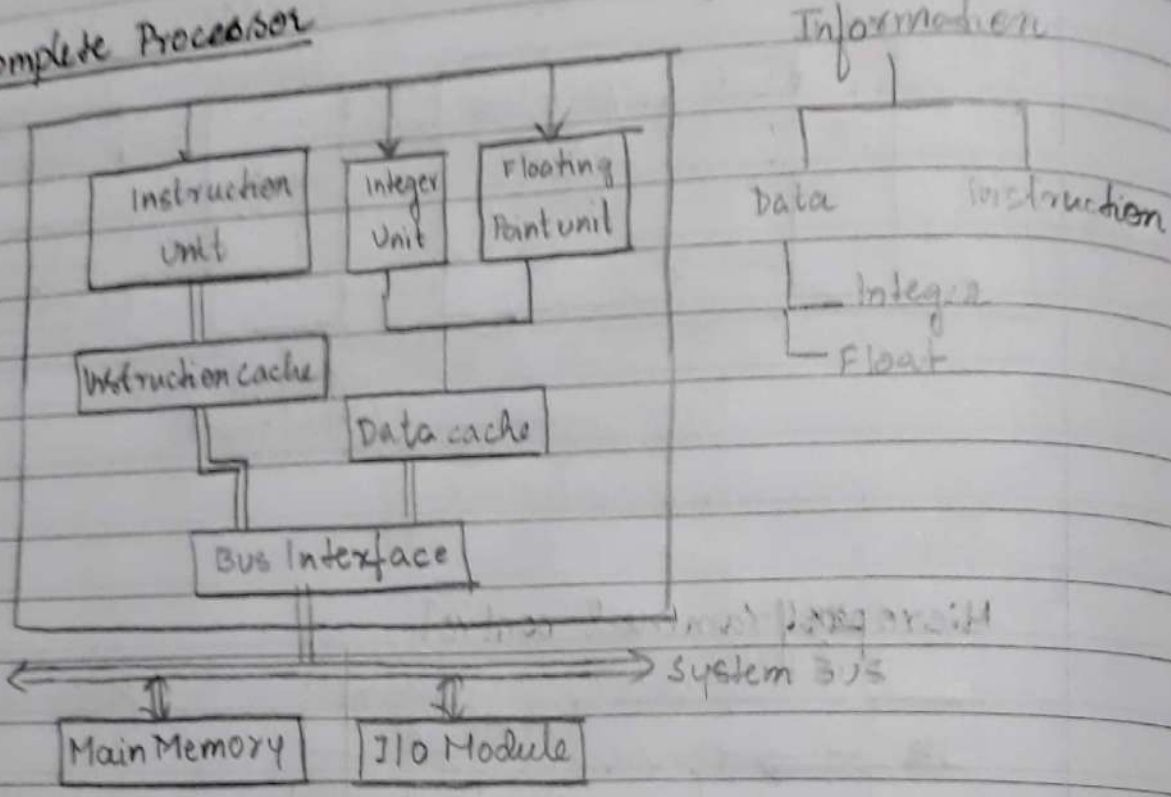


MPC - Microprogram Counter



Off Chip (not on processor)
Hence requires access time.

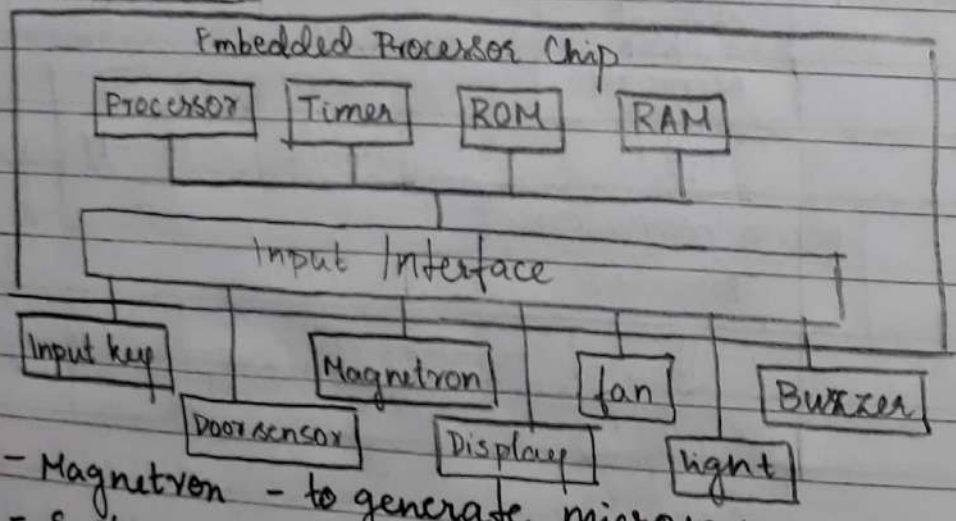
* Complete Processor



* Embedded Systems:

-Microcontroller system designed for specific application

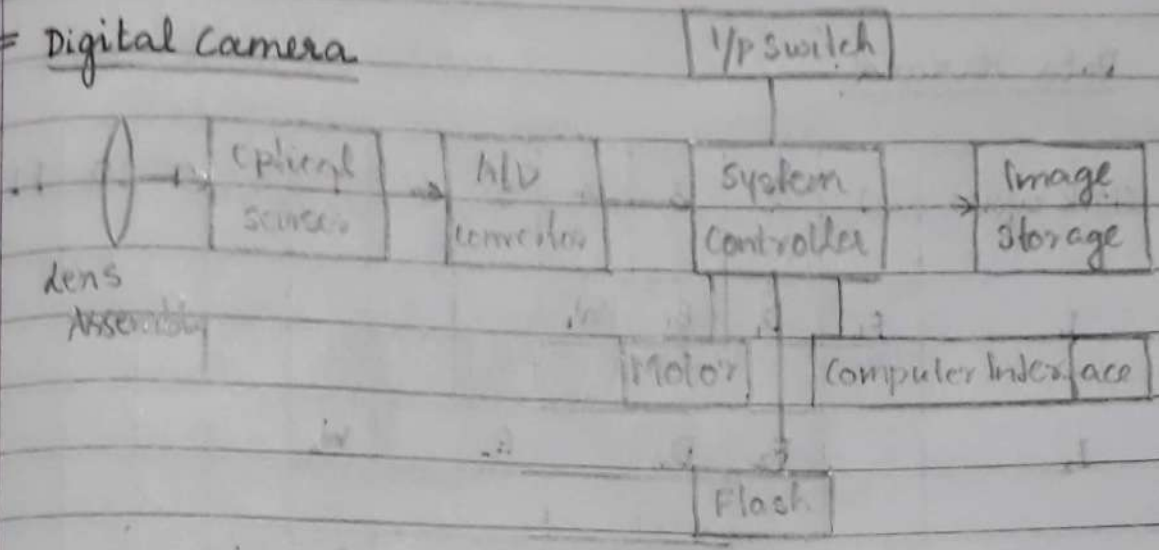
= Microwave Oven



- Magnetron - to generate microwave
- Switches
- display
- light
- exhaust fan
- door sensor

Magnetron is switched off once the door is open operated by OS present in ROM

= Digital Camera



★ Pipelining:

clock cycle	1	2	3	4	5	6	7	8	9
I_1	F	D	E	S					
I_2		F	D	E	S				
I_3			F	D	E	S			
I_4				F	D	E	S		

Ideal condition

stalled - delay in each step.

If more cycles are taken to carry out each step leads to haza

Data Hazard:

Clock cycles	1	2	3	4	5	6	7	8	9	10	11
I_1	F_1	D_1	E_1	W_1							
I_2		F_2	D_2	E_2 (Execution takes more time that usual)			W_2				
I_3			F_3	D_3			E_3	W_3			
I_4				F_4			D_4 (control signal is not issued until execution of previous instru	E_4	W_4		
I_5							F_5	D_5	E_5	W_5	

Control Hazards:

Control hazards are related to cache.

Cache miss: If the data is not present in the cache the the processor needs to access main memory and more it to cache and later carry out the execution.

Clock cycles	1	2	3	4	5	6	7	8	9	10
I_1	F_1	D_1	E_1	W_1						
		cache miss (more time to access data)								
I_2		F_2				D_2	E_2	W_2		
I_3						F_3	D_3	E_3	W_3	

Structural Hazard:

This occurs when two instructions require the given hardware at the same time.

clock cycles	1	2	3	4	5	6	7
I_1	F_1	D_1	E_1	W_1			
					Delay		
I_2		F_2	D_2	E_2	M_2	W_2	