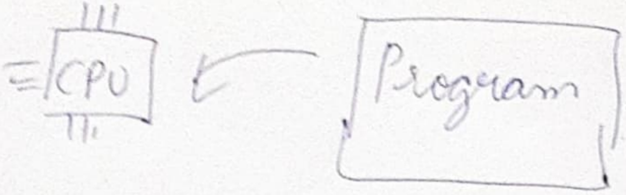
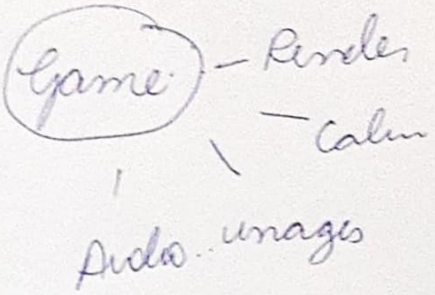
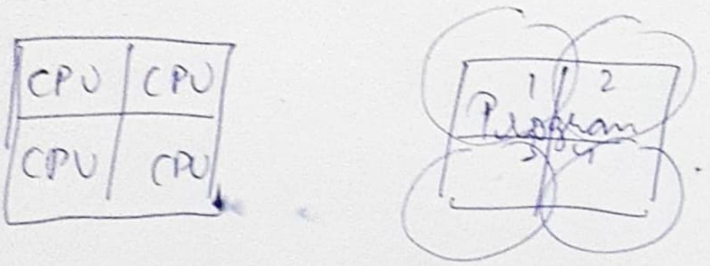
Parallel processing



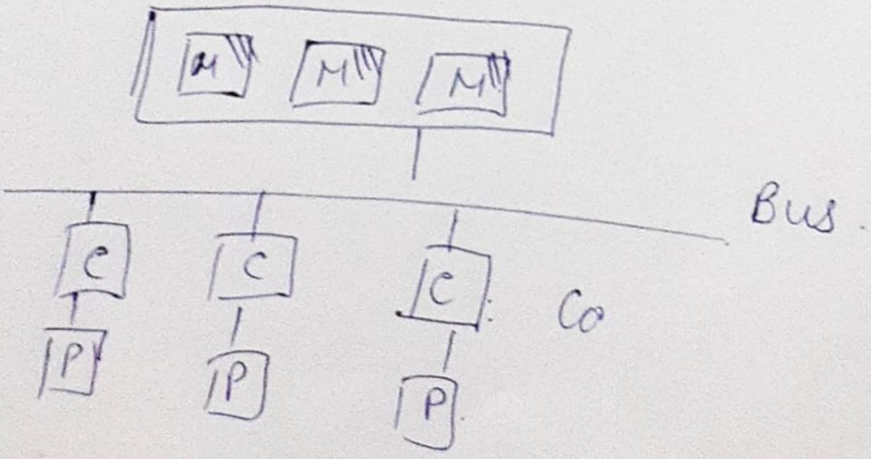


diff for CPU to load big programs.



Types of Organisations:

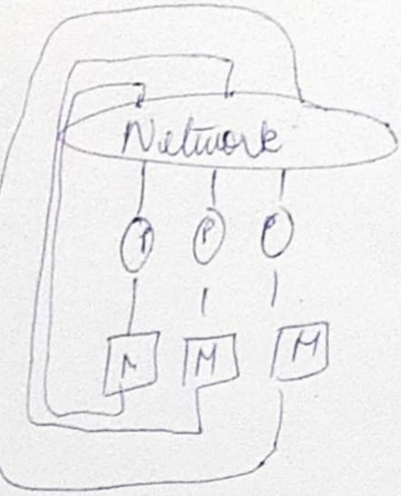
SMP (Symetric Multprocessors).



Cache coherance.
duplication
Memory waste.

solution: NUMA.

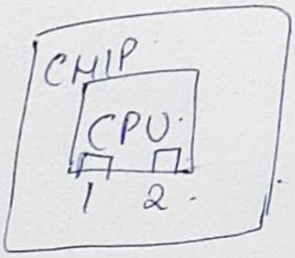
②NUMA - Non uniform Memory location



Fast

Assymetric
(Non uniform).

③. Multi-threading.



2 control units.

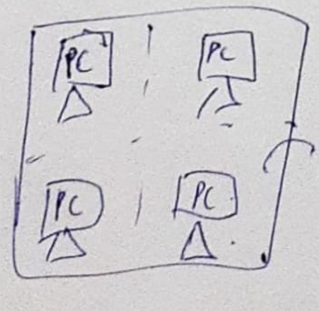
2 instructions performed at same time

CPU=1.

C.U ~ executes instruction.

④ Clustering.

Group of computers.



works whole.

PC.

Appliations

- Astrophysics
- Atmospheric & Ocean modelling
- CAD
- Military appluation
- VLSI design.
- weather forecast.

Inter process Communiation

Means the process to communicate with each other while they are running.
when we send a message we need sender & receiver.

Message passing system

(a) send (destination name, message).
(b) receive (source-name, message)

<u>Design Characteristics</u>.

① Synchronization ③ format of msg
② Addressing ④ Queing discipline

Synchronization

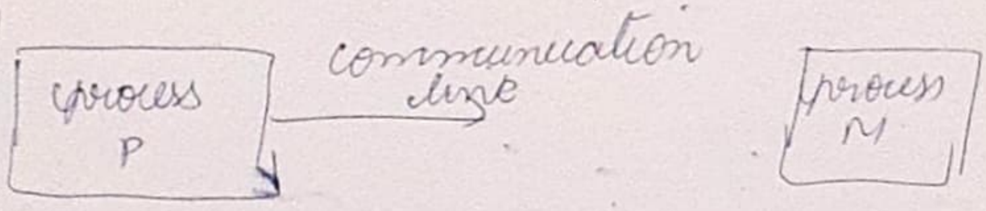
The communication of a msg between 2 processes implies some level of synchronization between the 2 process

Sender & receiver can be blocking or non blocking.

(a) Blocking send, Blocking receive

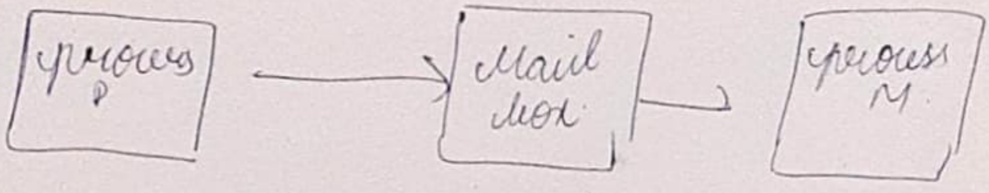
(b).

(1)



Direct Communication.

(2)

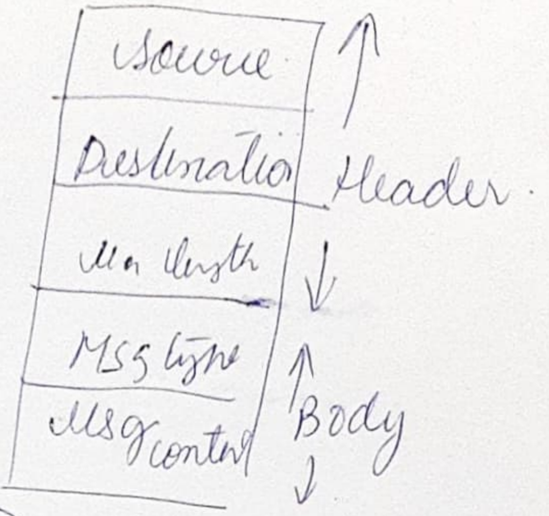


Indirect Communication

(3) Queing /Buffering.

a) zero capacity. b) Bounded Capacity.
c) unbounded capacity.

④ Message format



used to settle a dispute.

Inter processor Arbitration

* Computer system needs buses to facilate the transfer of info b/w its various components

* System bus connects CPU, IOP & Memory.

* Only 1 of CPU, IOP & Memory can be granted to use the bus at a time.

* Hence an appropriate priority resolving mechanism is requried to decide

* Arbitration mechanism is needed to handle multiple requests for bus.

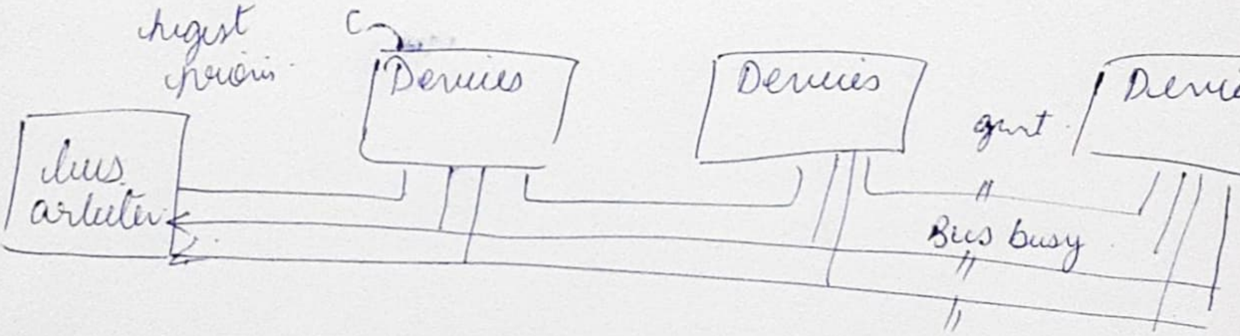
Arbitration techniques

① Static Techniques – { Serial Arbitration
Parallel Arbitration.

② Dynamic Techniques (S/W)

① Static Arbitration Techniques

- * The priorities assigned are fixed.
- * Also called Daisy chain Arbitration.
- * It is obtained by daisy chain connection of bus arbitration circuits.



- * This scheme is because the grant line ch from highest priority to lowest priority.
- * The highest priority device will pass to LP. only if it does not want it.
- * If bus busy line is free, the bus arbiter gives bus grant signal.

Advantages:-

① Simple design

② Less no of control lines.

Interconnection Structures

* A multi processor consists of a set of componets like CPU, memory, I/O processor et that communicates with each other

* The collection of paths connecting the various modules is called interconnection Structure

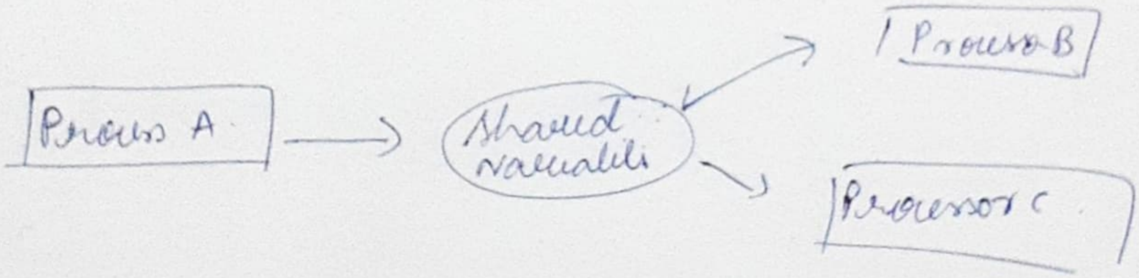
* The physical form for establishing an interconnection network in multi processor system are

* The physical forms are

1) Bus- oriented System.
2) Multi -port memory.
3) Cross - bar connection System.
4) Multi stage Switchup Network.
5) hypercube system.

Inter processor communication with using shared variables

* In a shared memory systems, interprocessor communication is achieved using shared variables. In such systems, the shared variables are stored in common memory which is accessible to all processors.



* While sharing common resources, or shared variables conflict problem may arise

* It is necessary to prevent conflicting use of shared resources by several processors

* This task is done by operating system.

Characteristics of Multi processors

* A multiprocessor system is an interconnection of 2 or more CPUS with memory & input-output

equipment.

* The term "processor" in multi-processor can be either a CPU or IOP.

* Multiprocessors are classified as MIMD systems.

* The similarity b/w Multi processor & multi computer is that both support concurrent operations

* Multi processing improves reliability of the system.

* The benefit derived from a multi processor organisation is an improved system performance.

* Multi processing can improve performace by decomposing a program into parallel executable tasks.

* Multi processor are classified by the way their memory is organised.

equipment.

* The term "processor" in multi-processor can be either a CPU or IOP.

* Multiprocessors are classified as MIMD systems.

* The similarity b/w Multi processor &e multi computer is that both support concurrent operations

* Multi processing improves reliability of the system.

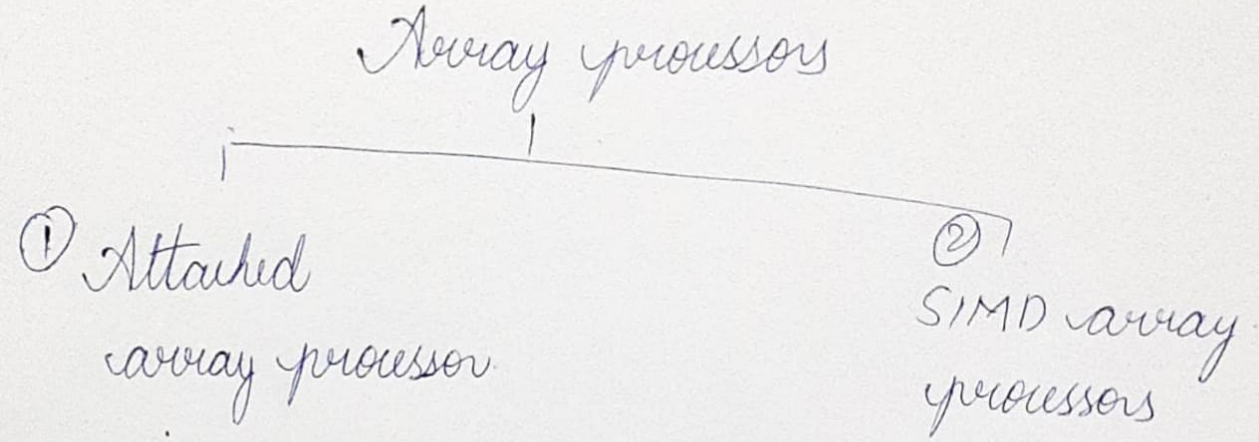
* The benifit derived from a multi processor organisation is an improved system performance

* Multi processing can improve performance by decomposing a program into parallel executable tasks.

* Multi processor are classified by the way their memory is organised.

Array Processing

It performs computations on large arrays of data.

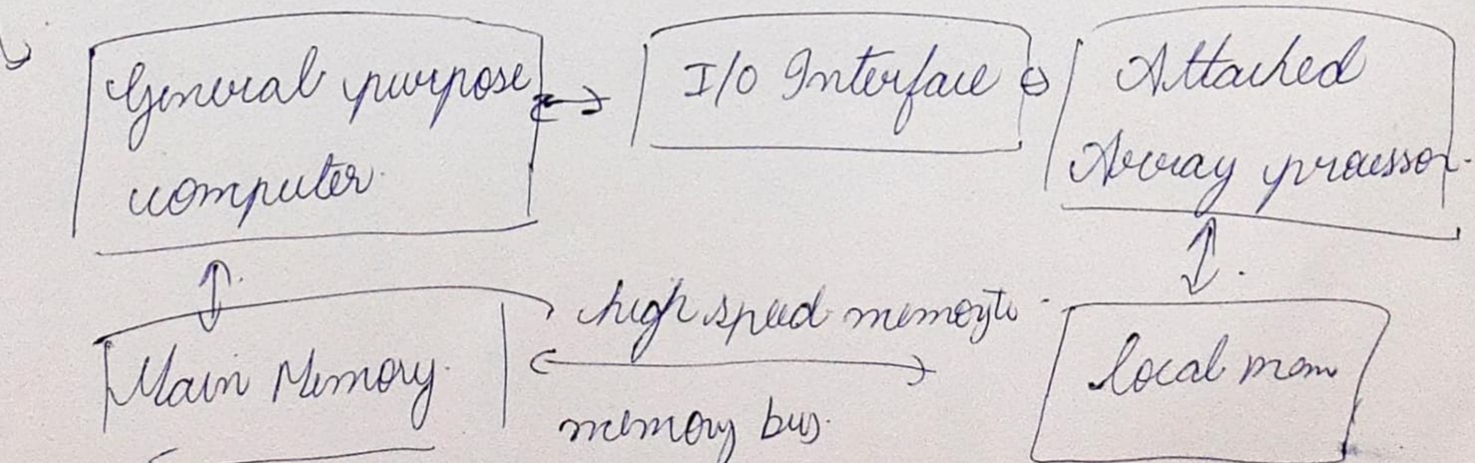


①

* To improve the performance of the host computer in specific numerical computation task. auxillay processor attached to it

② SIMD array processors.

* This is computer with multiple processing.



* Attached array processors has 2 interface.

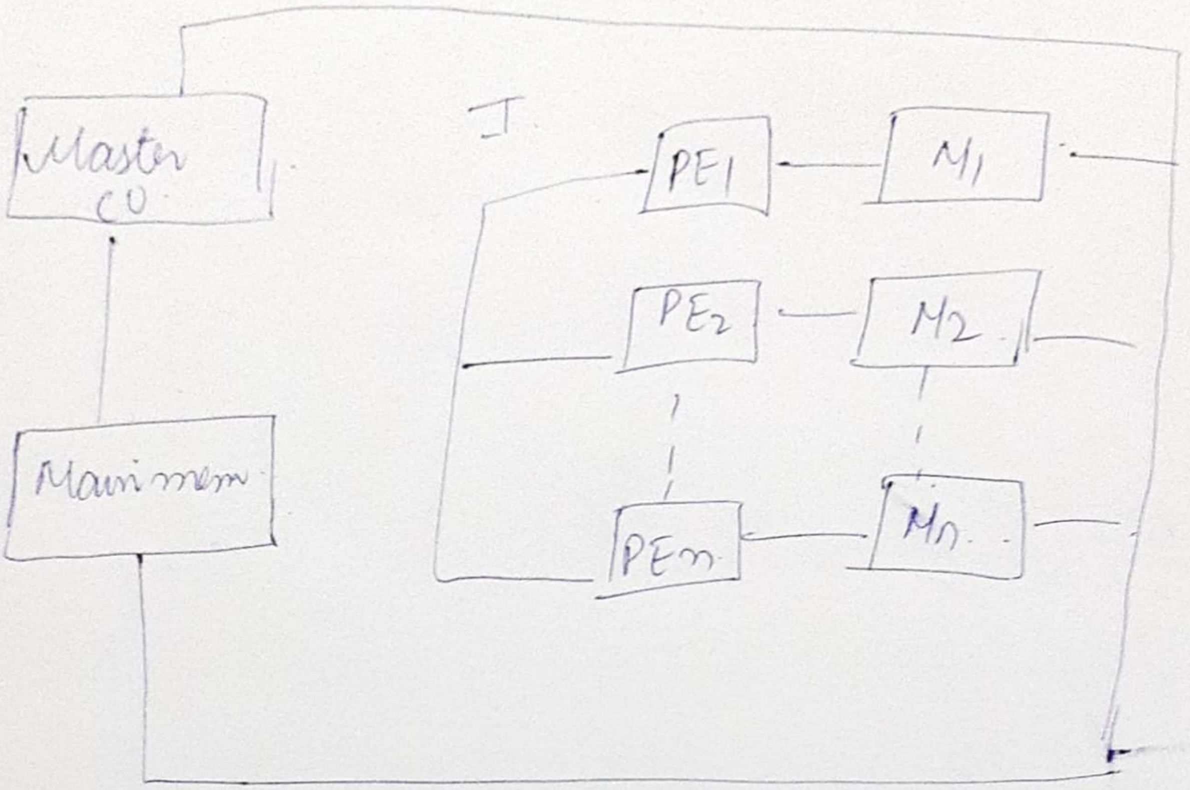
1) An I/O interface to a common bus.
2) Interface with local memory.

* The local memory interconnects main memory.
* Host computer is general purpose.
* Attached processor is back end machine driven by the host computer.
* The array processor is connected through an I/O controller

② SIMD

* This is computer with multiple processing unit operating in parallel.
* Both types of array processors, manipulate vectors but their internal organization is different.

SIMD. Array processors



* SIMD is a computer with multiple processing units operating in parallel.

* The processing units are synchronised to perform the same operation under the control of a common control unit, this provides a simple instruction stream, multiple data stream organisation

* Each PE includes
  - ALU
  - floating pt arithmetic
  - working registers

* Master CU controlls the operations of PEs.

* Main memory is used for storage ~~of~~ the program each PE uses operands stored in local variable.

Main Memory Vs Auxillary Memory

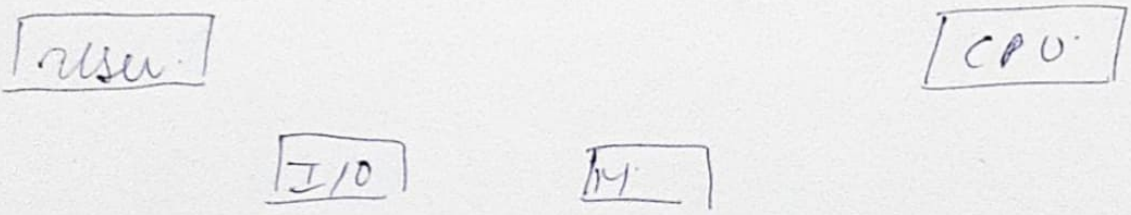
Main Memory	Auxilliay Memory
* Also called - primary memory	* Also called - secondary memory
* It is temporary memory of a computer.	* It is permanent memory of a computer.
* It has fast access time	* slow
* It is smallest in size.	* biggest.
* It is directly communic to CPU	* not directly.
* It is more expensive.	* less expensive.
* Volatile in nature except ROM	* Non volatile.
eg RAM / ROM.	* Eg HDD, ~~CF~~ FDD, pendrive etc

Programmed I/O Interrupt

3 techniques of I/O

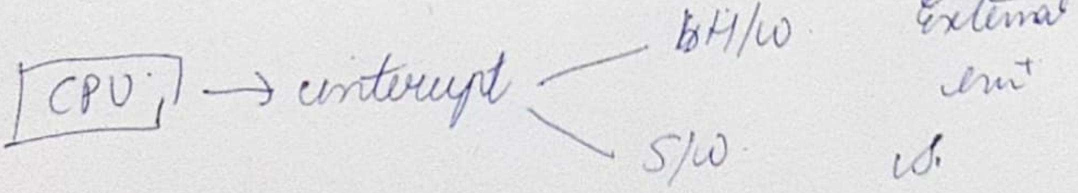
- programmed I/O
- interrupt driven I/O
- DMA

} exchange info b/w user CPU



Programmed I/O

Suspension of process



* There is no provision through which IO can inform to CPU about data transfer

* IO sets its own status and waits.

* CPU runs program periodically and checks

the status of each device one-by-one

* If any device has its status, if so then the CPU performs data transfer for it.

* It works on the principle of polling.

* Time required in programmed I O = .

Time to check status + Data transfer time.

$$\underbrace{\text{CPU reads flag register from I/O.}}_{\text{depends on I O speed.}} + \underbrace{\text{CPU internal checks flag bit.}}_{\text{negligible.}}$$

② Interrupt Initiated I O.

* I O device has a provision to inform to CPU about communication.

* When CPU receives & interrupt.

* It completes execution of current instruction

* Saves the status of current processor.

* Branches to service the Interrupt

* Resumes the previous process by taking out of stack

Cache Memory

* The term cache means a safe place for hiding or storing things.
* CM is a small, fast memory which holds copies recently accessed instructions and data.
* When the processor makes a request for memory Reference, the request is first sought in the cache.
* If we get that memory reference which is requested, we call it 'CACHE HIT'. Otherwise CACHE Miss.
* A block of elements are transferred from Main memory to cache memory by expecting that the next requested element will be residing in the. - - -

Associative Memory:

* The time reaquried to find an item in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by address

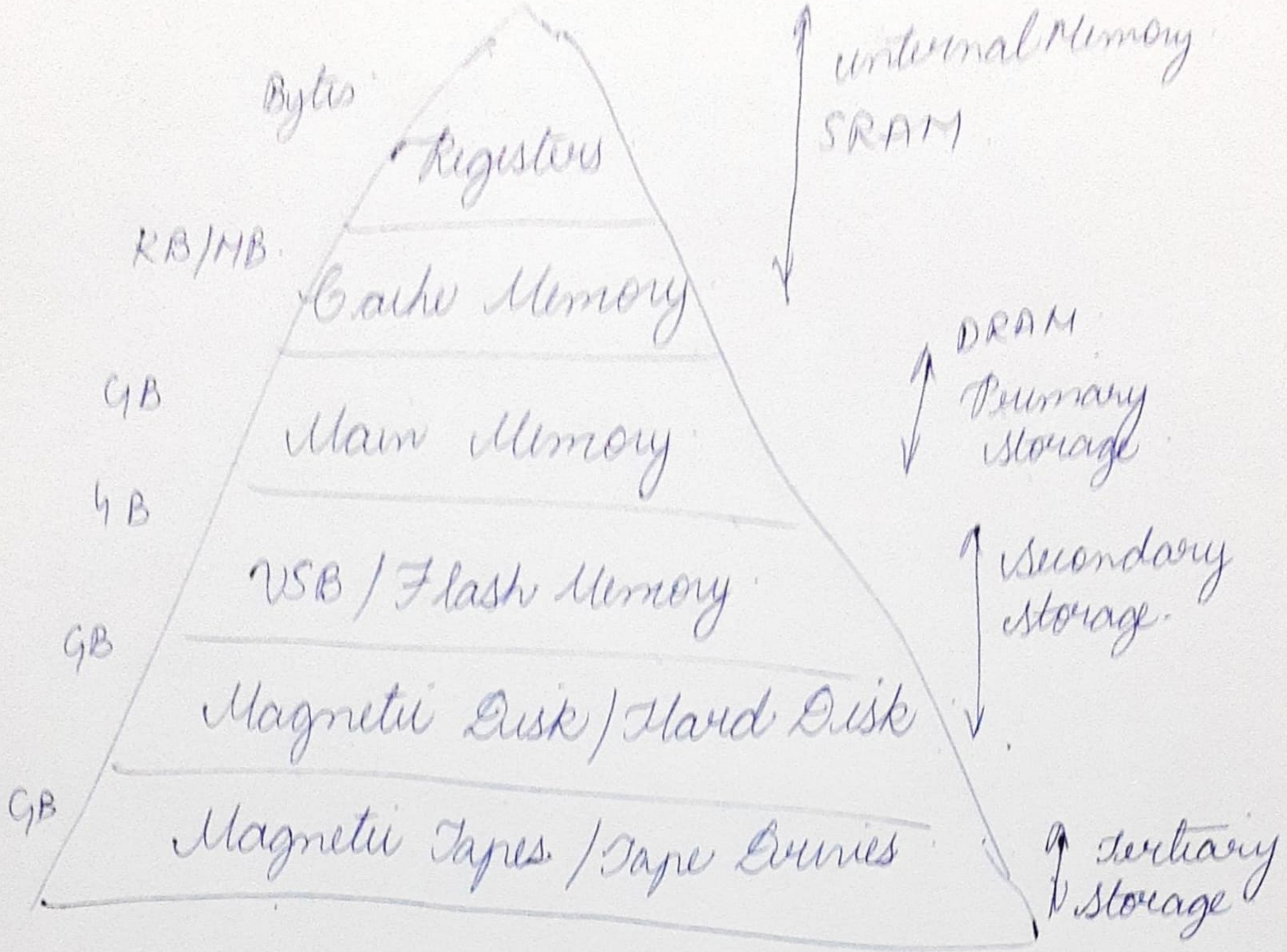
* The memory unit accessed by content is called associative memory or CAM.

* This type of memory is accessed simultaneously and in parallel on the basis of data content rather than specific address or location.

Memory Hierarchy.

* All computers need different kinds of memory.
* Volatile Memory && Non Voltalile memory
* The System combined memories working together is called the Memory Hierarchy

* It can be represented in a pyramid.



Priority Interrupt DMA

* It is a system that establishes a priority over the various sources to determine which condition is to be serviced first when 2 or more requests arrive simultaneously.

* Higher priority levels are assigned to requests which if delayed or interrupted, would have serious consequences.

* Devices with high speed transfer are given high speed priority & slow devices receive low priority

Establishment of priority of Simultaneous Interrupts

Software polling procedure

* One common branch address for all interrupts
* Interrupt program begins at the branch address and palls the interrupt sources in sequence
* The order in which they are tested depends upon the priority of each interrupt.
* The highest priort source is tested first.

H/W

* It functions as overall manager in an interrupt
* Each interrupt source has its own interrupt vector to access its own series.
* No pelling is required
* Can be established by either series or parallel.

Input Output Interface

* It is a hardware circuit used b/w CPU & I/O devices to supervise & synchronize all input & output transfers.

* I/O devices are electromechanical & electro-magnetic while CPU & memory are electric devices. So they have different signals. So conversion of signals is required.

* Data transfer rate of CPU is faster as compared to the I/O devices. So synchronisation is needed.

* Data format of I/O devices are different from word format of CPU & memory.

* Operating modes of I/O devices are different from each other and so each I/O device must be controlled so that it will not disturb the operation of other I/O devices to CPU.

Hardwired Control Unit & Micro-Programmed Control Unit.

Attribute	Hardwired Control Unit.	Microprogrammed CU.
① Speed.	Fast	Slow.
② Cost of Implimentation.	Expensive	Cheaper.
③ Control functions	Impliment in Hardware	Impliment in software.
④ Flexibility.	Not flexible to accomodate new system.	More flexible to accomodate new system
⑤ Ability to handle complex Instruct	Difficult	Easier.

⑥ Ability to support operating system	Very difficult	Easy
⑦ Design process	Complicated	Orderly & Systematic
⑧ Applications	RISC micro processor	CISC micro processor

Micro program Examples

*

3	3	3	2	2	7
F1	F2	F3	CD	BR	(AD)

* The micro Instruction includes 4 fields.

* F1, F2, F3 : These are micro operation fields. Each field is of 3 bits. They specify micro operations for the computer

* CD: This two-bit field selects status bit condition for branching operation. The condition includes zero value in AC, sign bit of AC equal to 1 or 0 etc

* BR: This 2-bit field specifies the type of branch to be used. Branch type includes unconditional branch, branch if zero, branch if negative and so on

* AD: This is an address field which contains a branch address. This field is of seven bits since controll memory has 128 words.

Instruction Formats

* Instructions are categorized into diff formats with respect to the operand fields in the instruction.

ddress

1) Three A. I.
2) Two A. I
3) One A.I.
4) Zero A.I.
5) Risc Instruct.

Three Address Instruction

* Computers with 3-address instruction formats can use each address field to Specify either a processor register or a memory operand

* The program is in assembly language

* The advantage of 3-address format is that it results in Short program when evaluating arithmetic expressions.

* The disadvantage is that, the binary-coded instructions require too many bits to three address.

Data Transfer & Manipullation

* In Computer instructions we have data Transfer instructions, data manipulation instructions and program control instructions.

* In Data transfer instructions, transfer of data from one location to other without changing the binary information content.

* In Data manipulation Instructions, it performs arithmatic logic & shift operations.

* In program Control Instructions, It provides decision making capabilities and change the path taken by the program when executed in the computer

* In Data transfer instructions, the most common data transfers are:-

a) b/w memory and processor registers

b) between processor & registers & I/O.

c) between processor registers themselves.

* Few of the typical data transfer instructions are load, store, move, exchange, Input, Outp

Push and Pop

Program Controll Instructions

* A program control type of instruction, when executed may change the address value in the program counter and cause the flow of control to be altered.

* The change in value of the program counter as a result of execution of a program contro instruction causes a break in the sequence of instruction execution.

* It has Branch, Jump, Skip, Call, Return, Compare & Test

Three Address Instruction

* Computers with 3-address instruction formats can use each address field to specify either a processor register or a memory operand

* The program is in assembly language
* The advantage of 3-address format is that it results in short program when evaluating arithmetic expressions.
* The disadvantage is that, the binary-coded instructions require too many bits to three address.

Two Address Instruction

* Two address Instruction are the most common in commercial computers.

Here again each address field can specify either a processor register or a memory word.

One Address Instruction

* One Address Instruction use an implied accumulator (AC) register for all data manipulation

* For multipliation & division there is a need for a second register

* However, here we will neglect the second register and assume that the AC contains the result of all operations.

Zero Address Instruction

* A stack-organised computer does not use an address field for the instructions ADD & ~~P~~ MUL.

* The PUSH & POP instructions, however need an address field to specify the operand that communicates with the stack

RISC Instruction

* The instruction set of a typical RISC processor is restricted to the use of load & store instruction when communicating between memory & CPU.

* All other instructions are executed within the registers of the CPU without refering to a memory.

* A program for a RISC type CPU consists of a LOAD & STORE instructions that have 1 memory and 1 register address, and computational - type instructions have 3 address with all 3 specifying processor register.

Asychronous Data Transfer

⇓

"At a regular interval"

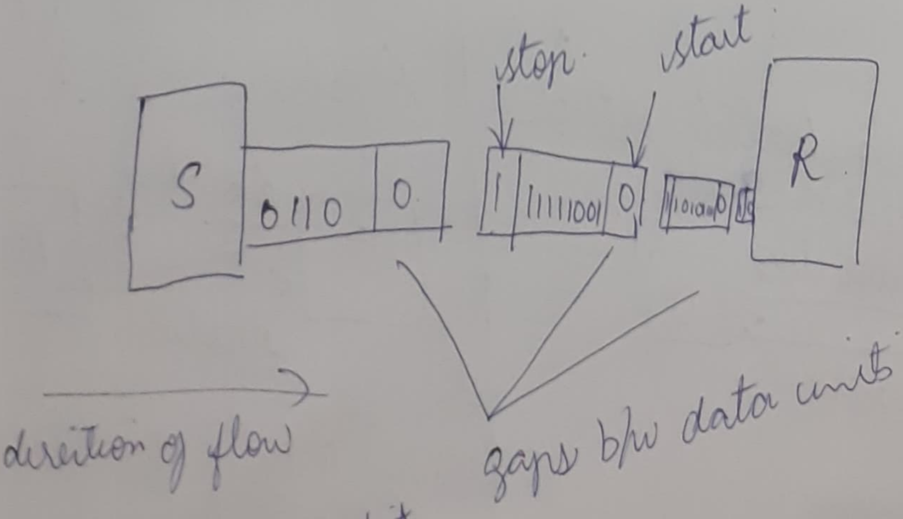
* In the transmitter transmits data bytes at any instant of time.

* Only 1 bytes is sent at a time. There is ideal time b/w 2 data bytes

* Transmitter & receiver operate at diff clock frequencies

* To help receiver start & stop bits are used along with data in middle

* Timing of signals is not imp.



0 ⇒ start bit
1 ⇒ stop bit

Methods used in Asynchronous Data Transfer

1) Strobe control = < source initiated / destination initiated

2) Hand shaking = < source initiated using / destination unit using handshake

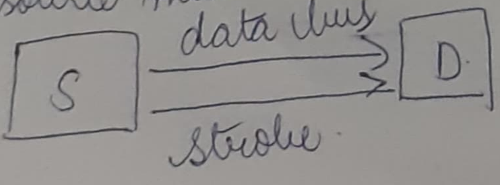
Strobe Control {Asynchronous Data Tra

⇓

* It uses single control signal for each tra
* Source initiated strobe.
* Destination initiated strobe.

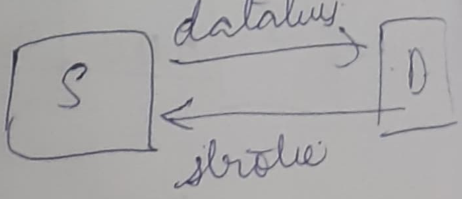
Source Initiated Strobe

source initiated strobe.



timing diagram

Destin....



timing diag

* Source ...
first placed
data on ...

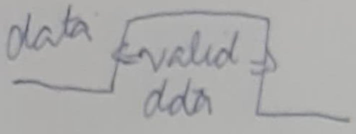
* Source activ...
the strobe
pulse

② Hand...
⇓

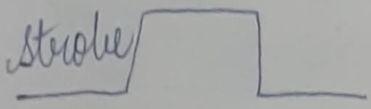
sour...

Han...
are
the

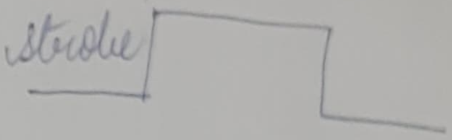
* Source unit is first placed data on bus.



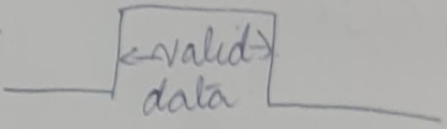
* Source activates the strobe clock pulse



* first destination unit activates the strobe pulse ifo the source to provide data.



②



② Handshaking. {ADT}

⇓

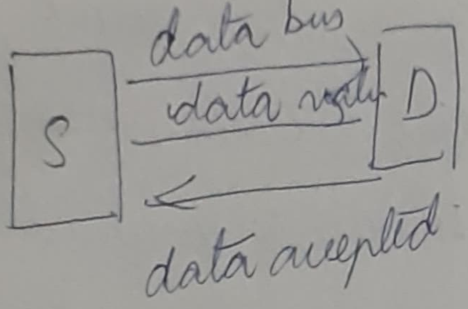
acknowledment

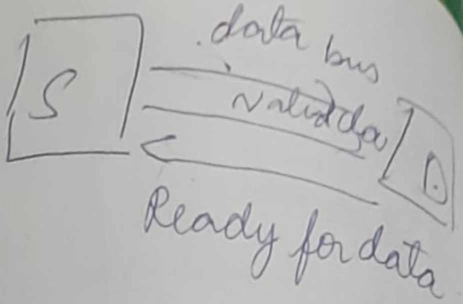
source to destination

⇓

Handshaking signals are used to synchronize the bus activities.

Destination to source





RISC & CISK.

RICS = Reduced

CISC	RISC.
• Load from memory. Arithmetic operations stores in the memory	• Load from memory • Arithmetic operation • stores in memory.
→ 10010001000 → 100111001101 011011001011 11010111111 → 010110101.	→ 1011101001001010 → 0101100101001100 → 01010110 001000
• Code size is small	• Code size is large
• no of cycles are more	• clock cycles r less

• needs littl
ware & y
processin

• uses m

• uses le

• Large n
ions

• Complex
set o

• Large n
mode

E

Pipel

* Arc
* Im

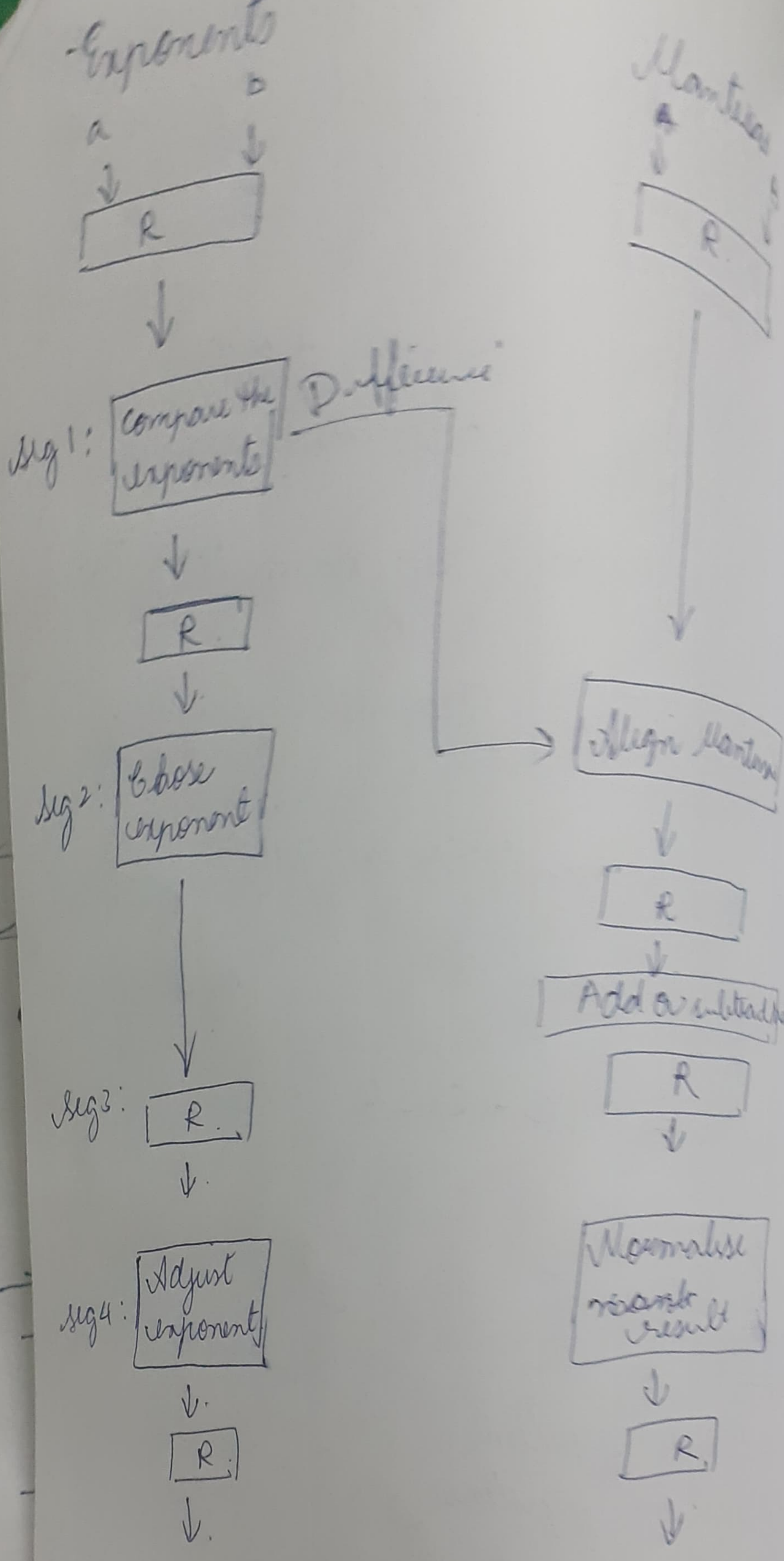
• needs better hard-ware ie powerful processing.	• Less costly hardware would also work.
• Uses more RAM.	• Uses less RAM.
• uses less registers.	• Uses more register
• Large no of instructions	• Less no of Instruction.
• Complex Instruction set computer.	• Reduced Instruction.
• Large no of Adressing mode	• Few no of A.M

EPIC = RISC + CISC.

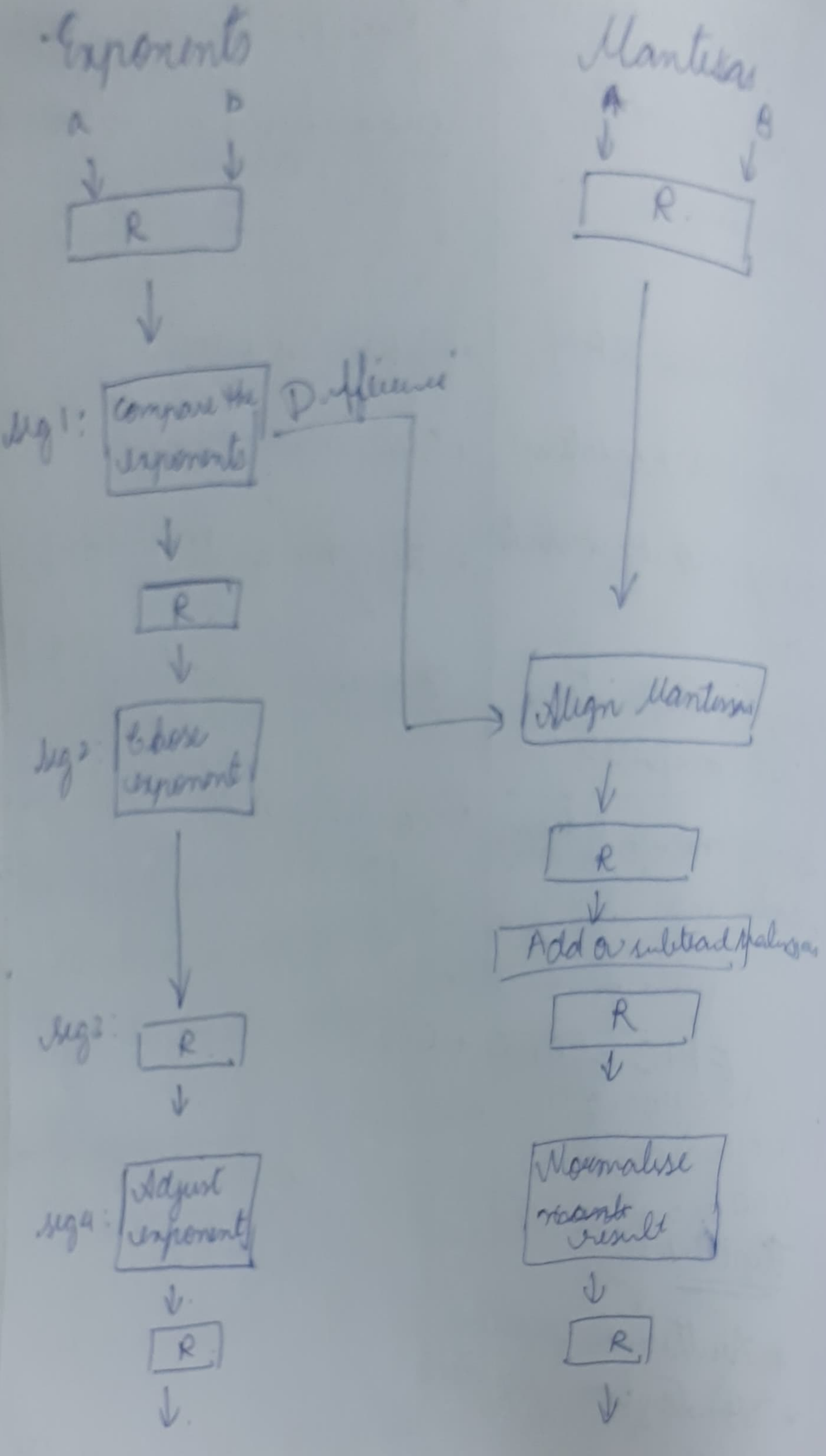
Pipeline

* Arithemeti pipeline
* Instruction pipeline.

Arithmetic Pipeline



Arithmetic pipeline



① Compare the exponents

② Align the mantissas

③ Add or subtract

④ Normalise result

$x = 0.9504 \times 10^{3}$

$y = 0.8200 \times 10^{2}$

$x = 0.9504 \times 10^{3}$

$y = 0.820 \times 10^{3}$

$z = 1.0324 \times 10^{3}$

$z = 0.10324 \times 10^{4}$

Data Hazard

* when either the source or the destination operands of an instruction are not available at the time expected in a pipeline

* The ~~x~~ As a result, the pipeline is stalled, we say such a situation as data Hazard

* Consider a program with 2 instructions I_1 & I_2

* When this program is executed in a pipeline, the execution of these instructions can be performed concurrently

* In such a case the result of I_1 may not be available for the execution of I_2

* If the result of I_2 d
on the result of I_1, we
get incorrect results if both
are executed concurrently.

* Assume A=10.

I_1 : A ← A+5

I_2 : A ← A×2

* when these 2 operations are performed one after the other we get result 30.

* But if they are performed concurrently it leads to incorrect resut as in this case the val of I_2 depends on the result of I_1

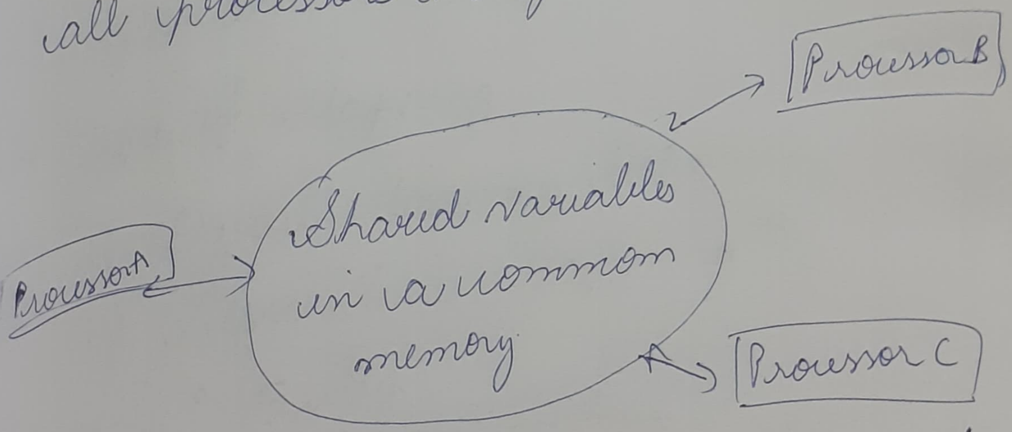
* The hazard due to such situation is called as data Hazard or data dependent Hazard.

* To avoid such incorrect results we have to execute dependent instruction one after the other -

Interprocessor Communi
using Shared Variables

* In a shared memory system, interprocessor communication is achieved using shared variables.

* In such systems, the shared variables are stored in common memory which is accessible to all processors in system.



* While sharing common resources/ Shared variables, conflict problem may arise

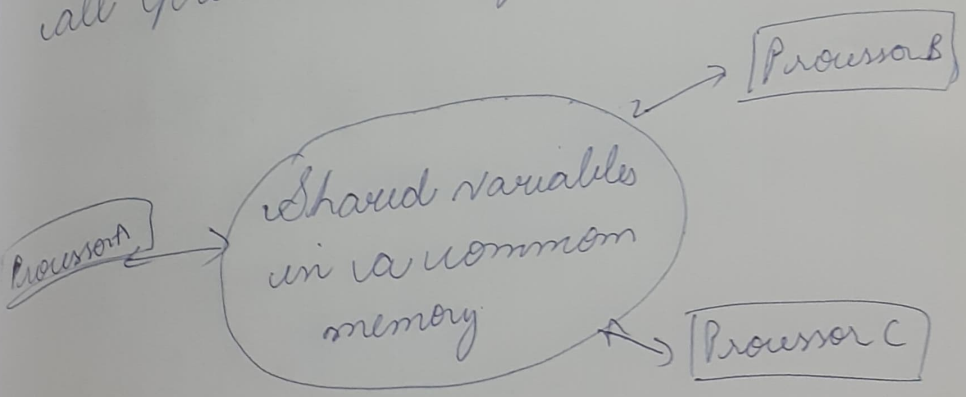
* It is necessary to prevent conflicting use of shared resources by several processors

* This task is done by operating system

Interprocessor Communication using Shared Variables.

* In a shared memory system, interprocessor communication is achieved using shared variables.

* In such systems, the shared variables are stored in common memory which is accessible to all processors in system.



* While sharing common resources/ Shared variables, conflict problem may arise

* It is necessary to prevent conflicting use of shared resources by several processors

* This task is done by operating system

Cache Mapping Techniques

* It tells us which word of the main memory will be placed in which location of the cache memory.

* Basic idea : Mapping b/w the Cache Addresses & Main Memory addresses referin to same unit of information.

* Types of Mapping
  1) Direct Mapping technique.
  2) Associative Mapping "
     - Fully
     - Set Associative.

Direct Mapping

* It is simplest mapping technique
* In this technique, each block from

the main memory has only one possible location in the cache organisation

main mem — location cache org

Associative Mapping (Fully Associative ")

* In this technique, a main memory block has only can be placed into any cache block position

* As there is no fixed block, the memory address has only 2 fields: word & tag.

* The set associative mapping is a combination of both direct & associative mapping.

* It contains several groups of direct mapped blocks that operate as several direct mapped caches in parallel.

* A block of data from any page in the main memory can go into a particular block location of any direct - mapped cache

* The req address comparisions depend on the no. of direct mapped cachs in the cache systems.

* These comparisions are always less than the comparisions req in the fully associative mapping.

A comp 4K :-

word bits: Each block contains 64 words., ∴ to identify each word, we must have (2^6-64) six bits. reserved for it.

$$\text{Tot no of block} = \frac{\text{Tot words in cache}}{\text{words per block}}$$

$$= \frac{4096}{64} = 64.$$

Tot no. of sets = $\frac{\text{no. of blocks}}{\text{blocks per set}} = \frac{64}{4} = 16$

To identify each set ($2^4 = 16$) four bits are used.

Tag bits: Tot no of words in main memory.

$= 65536 \times 64 = 4194304 = 2^{22}$.

Main memory address is 22 bits wide.

Tag bits = 22 − set bits − word bits.

$= 22 - 4 - 6 = 12$.

Main Memory address =

Tag	Set	word
12	4	6

Typical DMA Controller 10M

- DMA - Direct Memory Access
- It is a hardware controlled data transfer.
- DMA controller is used to carry out data transfer.
- During data transfer, data is not routed through processor.
- To perform DMA operation the basic blocks req in a DMA channel/ controller are shown in diagram.
- DMA controller communicals with CPU via the data bus & control lines.
- The registers in DMA are selected by the CPU through address bus.
- The RD & WR inputs are bi-directional.
- When the BG input is 0, CPU can communicate with DMA registers through the data bus to read & write

the DMA registers RD & WR Input signals for DMA

* when BG = 1, The CPU has relinquished the buses & bus DMA can communicate directly with the memory.

* DMA consists of Data count, data register, address register & control logic.

* Data counter register stores the no. which gives the no. data transfers to be done in one DMA cycle. It is automatically decremented after each word transfer.

* Data registers act as buffer whereas address register initially holds the starting address of the device.

* After each transfer, data counter is tested for 0.

* when the data count reaches '0', the DMA transfer halts.

* The DMA controller is normally provided with an interrupt capability

Various Addressing Modes 10M

* Implied Mode

* Immediate Mode

3) Register Mode

4) Register Indirect Mode

5) Auto Increment / Auto decrement mode

6) Direct Address Mode

7) Indirect Address Mode

8) Relative Addressing mode.

9) Indexed Addressing Mode.

10) Base Registor Address.

The diff ways in which source operand is denoted in an instru

1).

1) Register Addressing Mode-

* The operand is the contents of processor register.

* The name of register is specified in the instruction

* Ex: MOV R1, R2:

- The instruction copies the contents of R_2 to R_1.

2) Direct Addressing Mode.

* The address of the location of operand is given explicitly as a part of instruction

* Ex: MOV. A, 2000:

- This instruction copies the contents of the memory location 2000 into the A register

3) Immediate Addressing Mode:-

The operand is given explicitly in

the instruction

Ex:- MOV A,#20.

This instruction copies operand 20 in the register A.

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

4) Indirect Addressing Mode:-

The instruction contains the address of the memory which refers to the address of the operand.

5) Register

Immediate Addressing

initialize a variable.

no additional memory

Operand is mentioned explicitly

eg MOV R_0, 200.

CPU register

Initialized value

Direct Addressing

address of memory location is given explicitly

contents at the memory is provided

eg ROV R_1, x.

memory [content: 10].

CPU register

$P(\overset{n}{\underset{i=1}{\cap}} Ai) \geq \sum_{i=1}^{n} n(Ai)-(n-1)$

Mathemetic induction method is used to prove this

For any event, E∈S, $0 \leq P(E) \leq 1$.

Consider two events ($n=3$) then

$D(A_1 \cup A_2) \leq 1$

∴ $A_1 \cup A_2$ is an event

By using addition Theorem.

$P(A_1)+P(A_2)-P(A_1 \cap A_2) \leq 1$.

$P(A_1)+P(A_2) \leq 1+P(A_1 \cap A_2)$

we can write the above eq^n as

$P(A_1 \cap A_2) \geq P(A_1)+P(A_2)-1$

The statement is true for $n=2$.

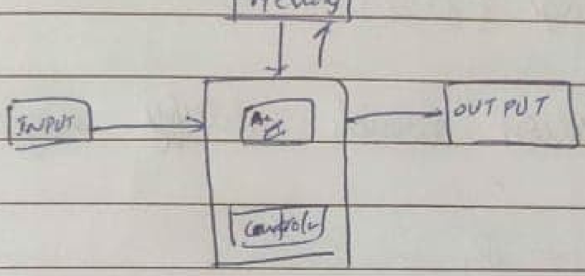
Let us assume that the statement is true for $n=k$ i.e $P(\overset{k}{\underset{i=1}{\cap}} A_1) \geq \sum_{i=1}$

UNIT-1

COA

Q list four basics function of the CPU?

Ans: The digital computer consist of five functionally independent unit : input, memory, arithmetic and logic, output and control unit

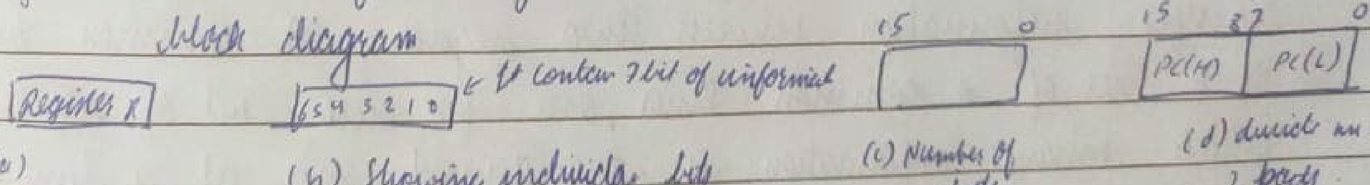


Register transfer ~~language~~ :-

A register is a collection of flip flop where every flip-flop contain 1-bit of information. 3 flip flop contain 3 bit of memory.

Register show in four ways:-

Block diagram



(a)

(b) Showing individual bits

(c) Number of bits

(d) divide in 2 parts.

Register transfer transfer the content of R_1 register to R_2 register

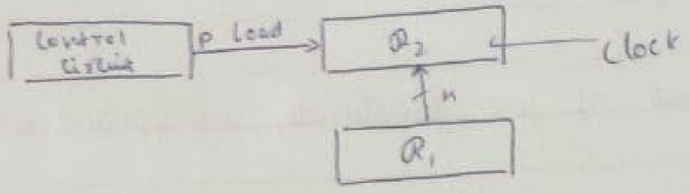
~~$R_1 \leftarrow R_2$~~ $R_2 \leftarrow R_1$ (replacement operation)

if $(P=1)$ then $(R_2 \leftarrow R_1)$

$P: R_2 \leftarrow R_1$

transfer from R_1 to R_2 when $P=1$

Block diagram



$R_2 \Leftarrow R_1$

Register transfer language (RTL)

A symbolic representation which specify register transfer microoperation

$$R_2 \leftarrow R_1$$

if $(P=1)$ then $R_2 \leftarrow R_1$

$P: R_2 \leftarrow R_1$

* Common bus:-

In multiple register configuration a common bus system is used to transfer information between two register. A common bus consist of a set of common lines, one for each bit of register, through which binary information is transformed one at a time.

The common bus scheme can be implemented in two ways

- using Multiplexers
- using tri-state bus buffers.

(a) using multiplexer

The number of multiplexer depend on number of register or size of multiplexer is also depend on size of register

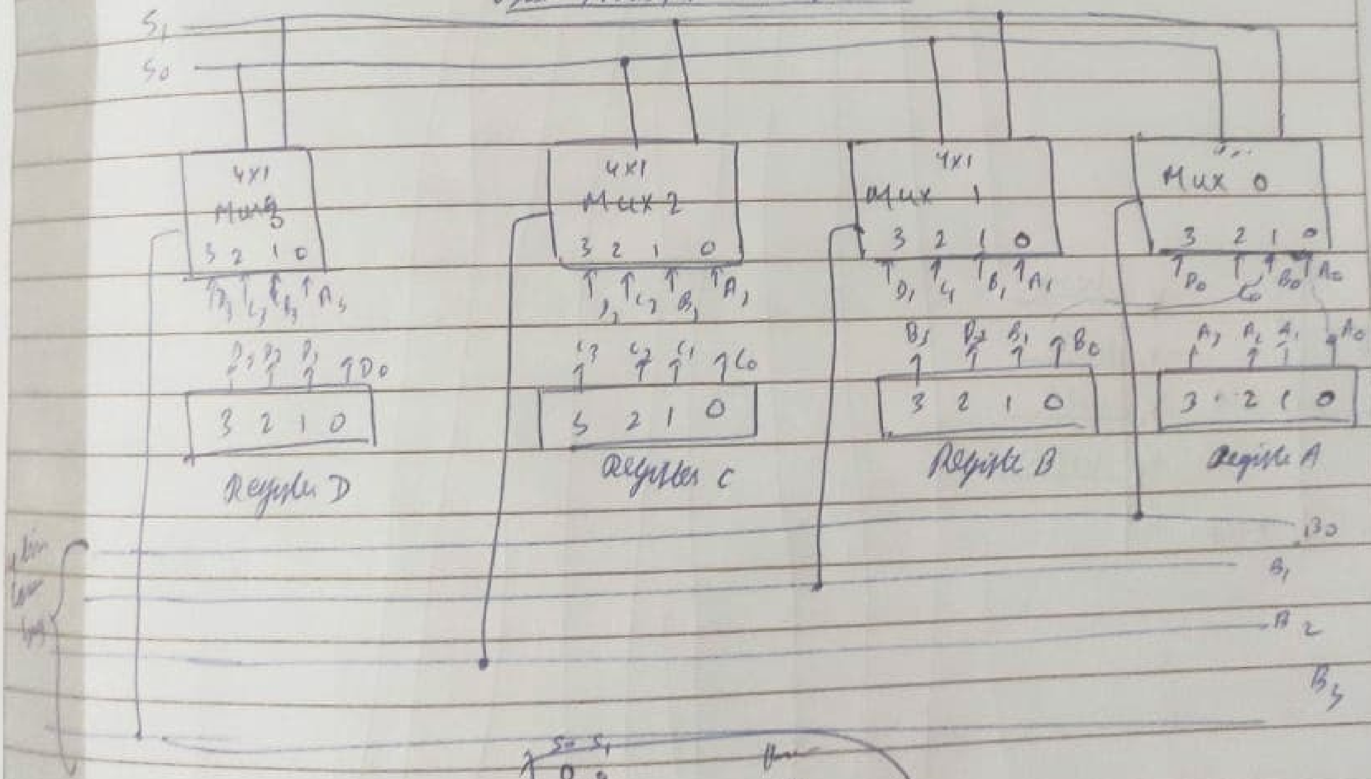
n → 2^n I/p's → 1 output

select one input line and give output at one corresponding

$\boxed{S_0, S_1}$ → are two selection line

2 → 2^2 i/p's → 1 output

Bus transfer for 4 register



S0 S1	
0 0	0
0 1	1
1 0	2
1 1	3

Read

IR ← M[AR]

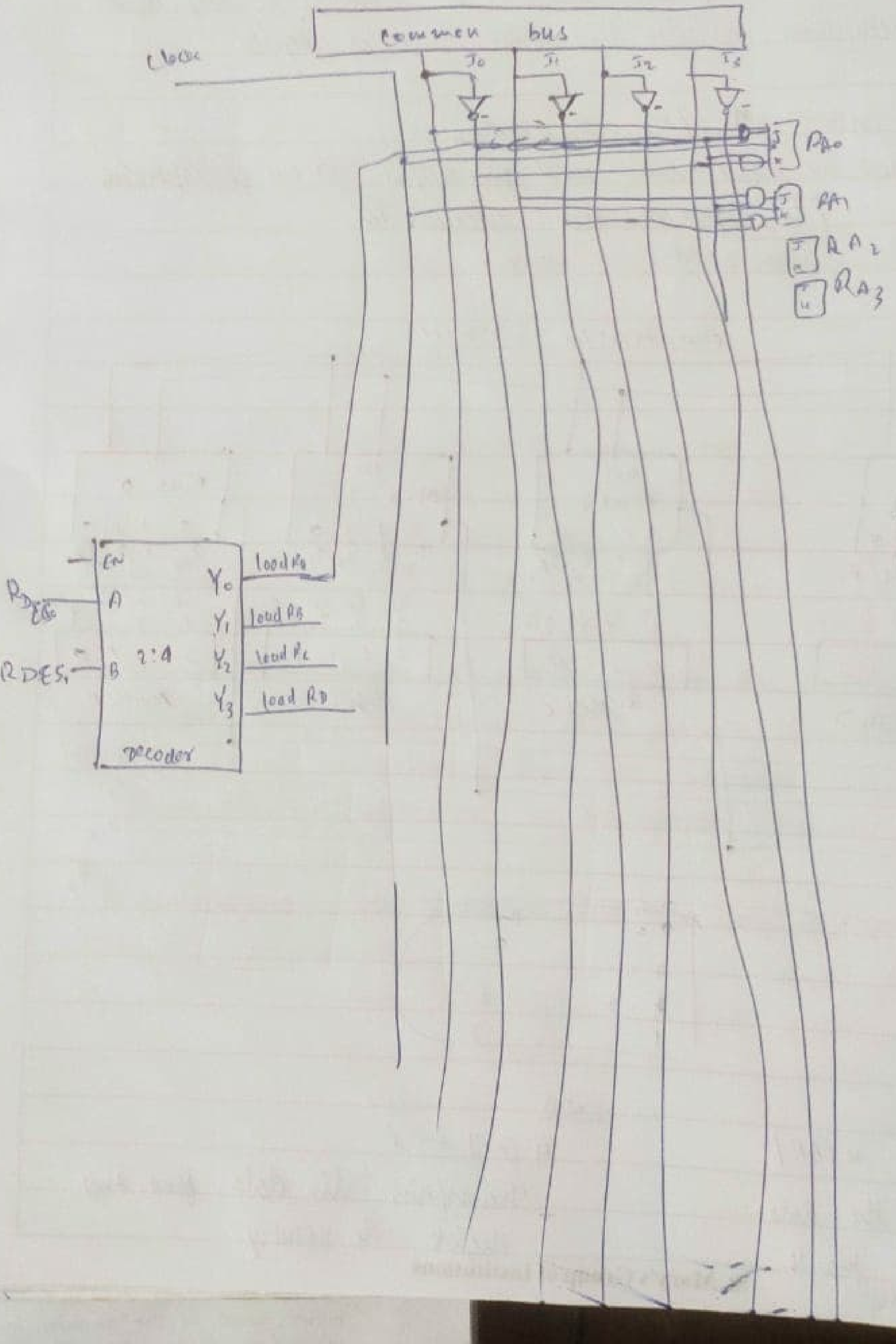
Transfering the data from memory ~~[illegible]~~ to any [illegible]

write

M[AR] ← R

Transfering the data from any Register to memory.

Q. Design a circuit for parallel load operation into one of the 4-bit register from a bus Mention clearly control / selection bits and selection logic for JK flip flop

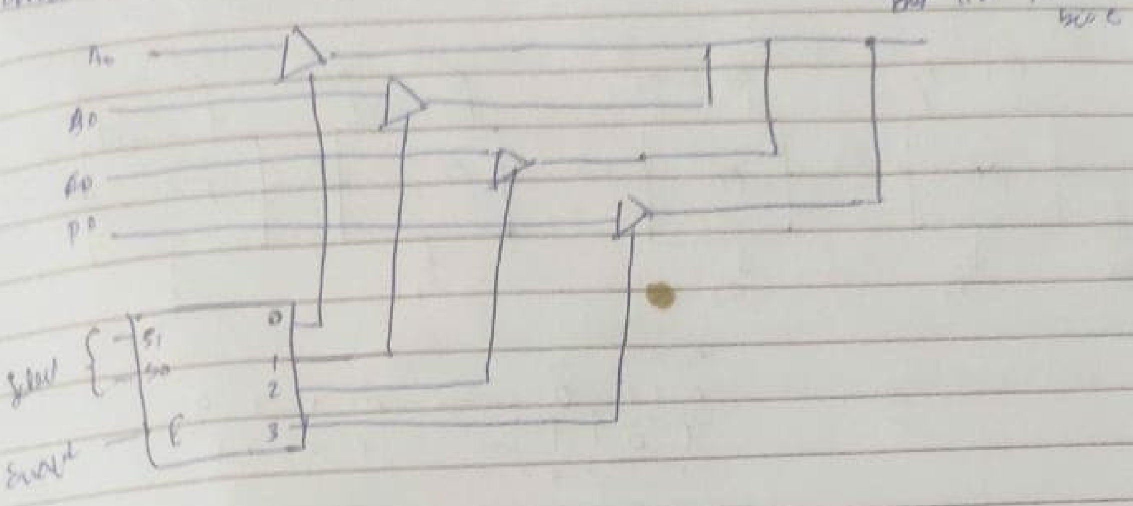


10,11,12 → transferring data Register to any Reg -

26,

Date :

(b) Three state bus buffer:



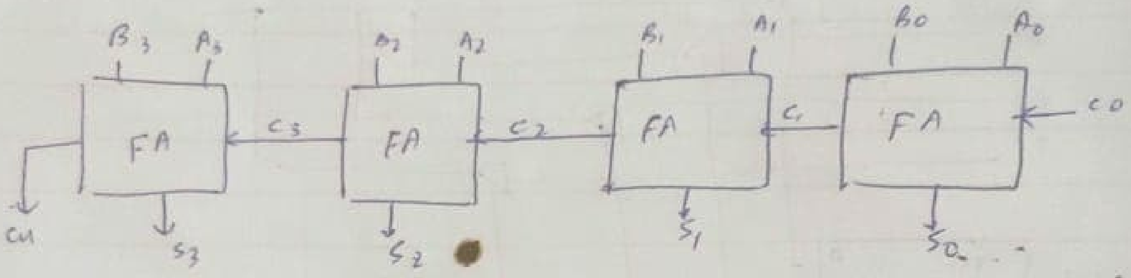
Microoperation

(*) Arithmetic Micro operation:

There is mainly seven type of arithmetic microoperation.

(1) Add	$R_3 \leftarrow R_1 + R_2$	Content of R_1 and R_2 are added and result is transferred. to R_3
(2) Subtract	$R_3 \leftarrow R_1 - R_2$	Content of R_2 are subtracted from content of R_1 and result is transferred to R_3
(3) 1's complement	$R_1 \longrightarrow \bar{R}_1$	complement the content of R_1
(4) 2's complem.	$R_1 \rightarrow \bar{R}_1 + 1$	complement the content of R_1 and add 1 in it
(5) 2's complement Subtraction	~~R_3~~ $R_3 \rightarrow R_1 + \bar{R}_2 + 1$	Add R_1 and R_2 the 2's complement of R_2
(6) Increment	$R_1 \leftarrow R_1 + 1$	Increment the content of R_1 by 1
(7) Decrement	$R_1 \overset{\leftarrow}{\ominus} R_1 - 1$	Decrement the content of R_1 by 1

4 bit binary Adder



Ex:

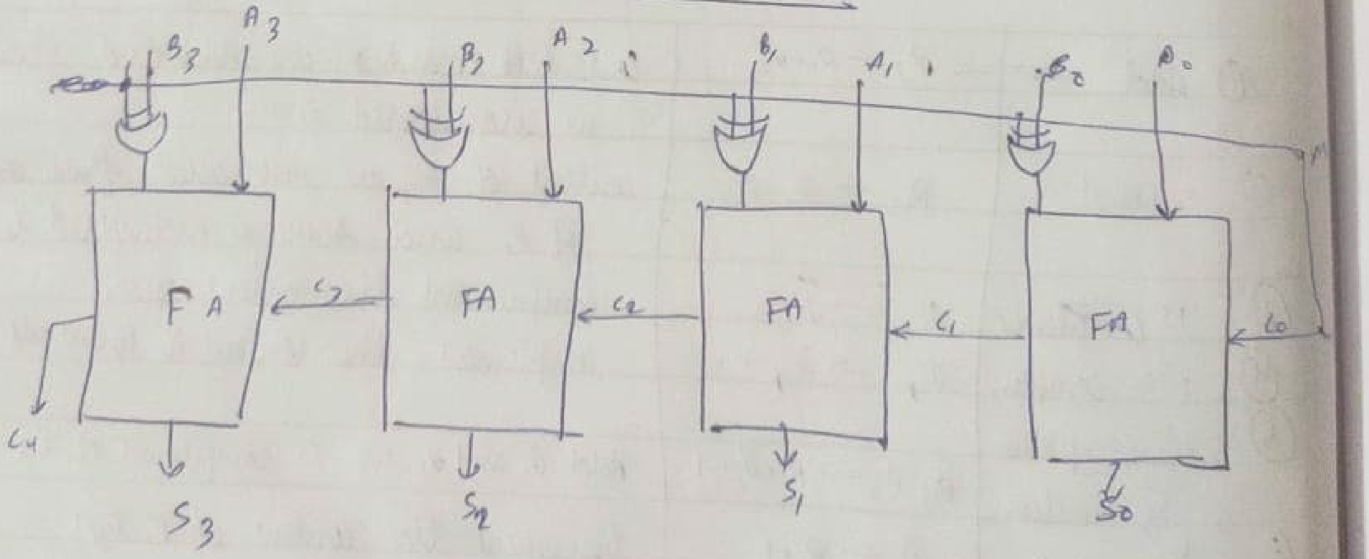
A3 A2 A1 A0
0 1 1 0 — 0 1 1 0
B3 B2 B1 B0
1 0 1 1 — 1 0 1 1
1 1 0 0

0 0 0 1

$D = a + \bar{b} + 1$

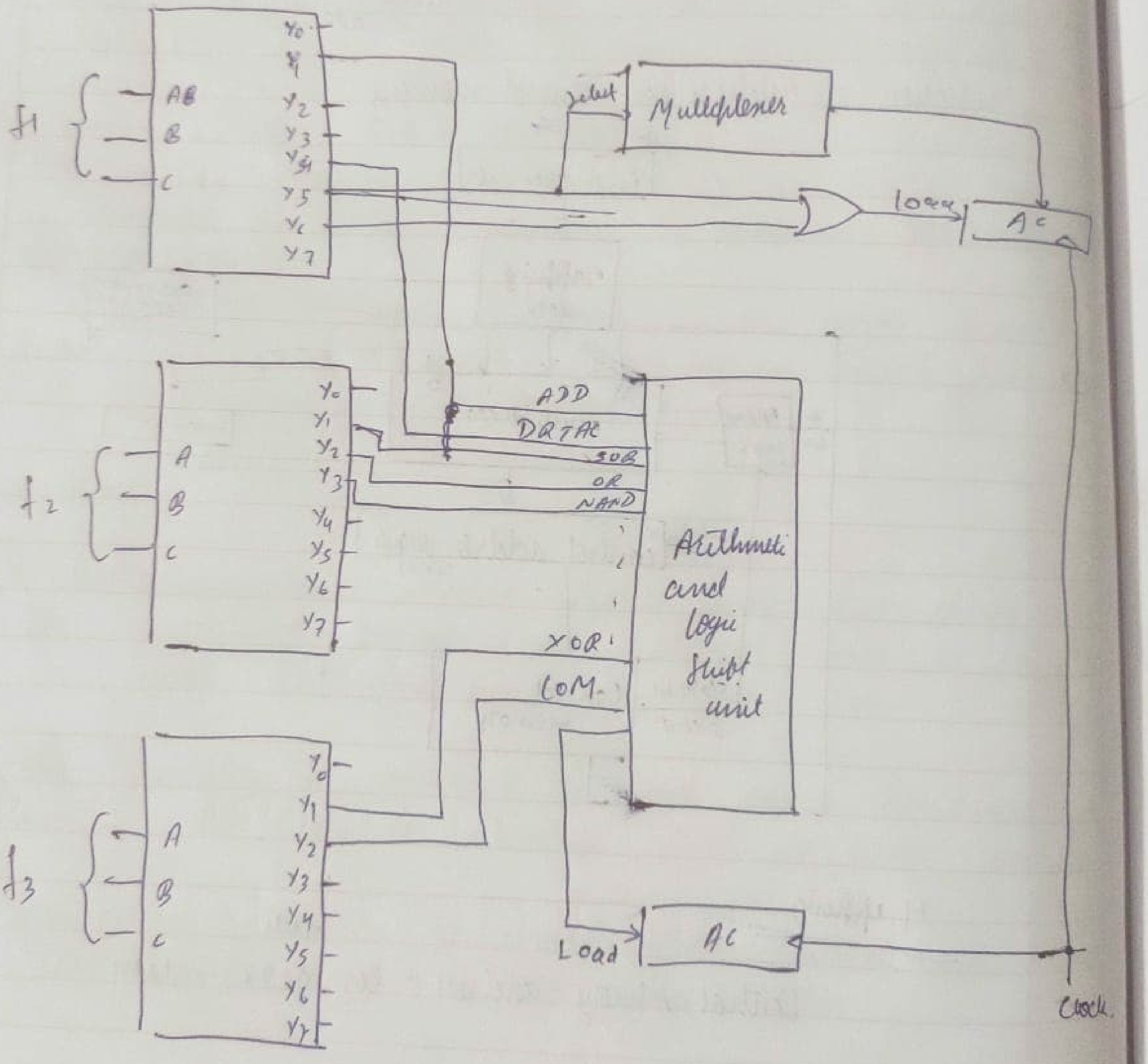
2's complement

4 bit binary Adder / subtracti



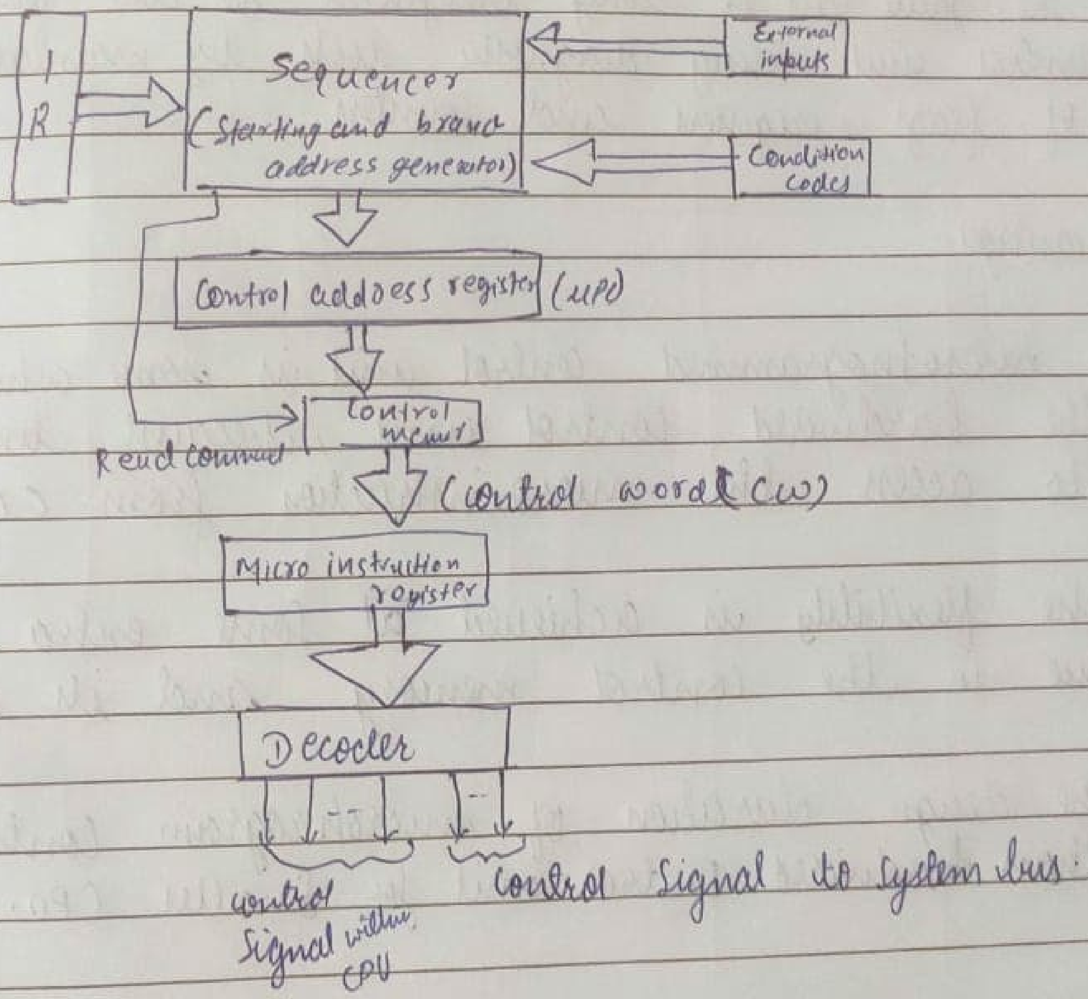
-: Design of control unit :-

Q. Microoperation fetch:-



→ Number of times the control unit is required to test the status of the condition code or external input to choose between alternative courses of action. In such situation, microprogramed control use conditional branch micro-instruction. In addition to the branch address, the instruction specify which of the external inputs, condition codes, or possibly bits of the instruction register should be checked as a condition for branching to take place

Code


I
R
Sequencer
(Starting and branch address generator)
External inputs
Condition codes
Control address register (uPC)
Control memory
Read control
(control word (CW))
Micro instruction register
Decoder
control signal within CPU
Control signal to system bus.

dvantages :-

→ It simplifies the design of control unit. That it is both cheeper and less error prone to ~~to~~ implement.

→ Control function are implemented in software rather than hardware.

UNIT:-2

Control Memory

① CPU ② I/O ③ Memory

↓

ALU, registers, control unit

a) Microprogrammed Control unit

It consist of control memory, control address register, micro instruction register and microprogram sequencer.

The components of control unit work together as follows:

→ The control address register (μPC) holds the address of the next microoperation to be read. Every time a new instruction is to be loaded in to the IR, the output of the block labelled "starting address ~~register~~ generator" is loaded into the μPC.

→ When address is available in control address register, the sequencer issues READ command to the control memory

→ After issue of READ Command, the word from the address location is read into the microinstruction register

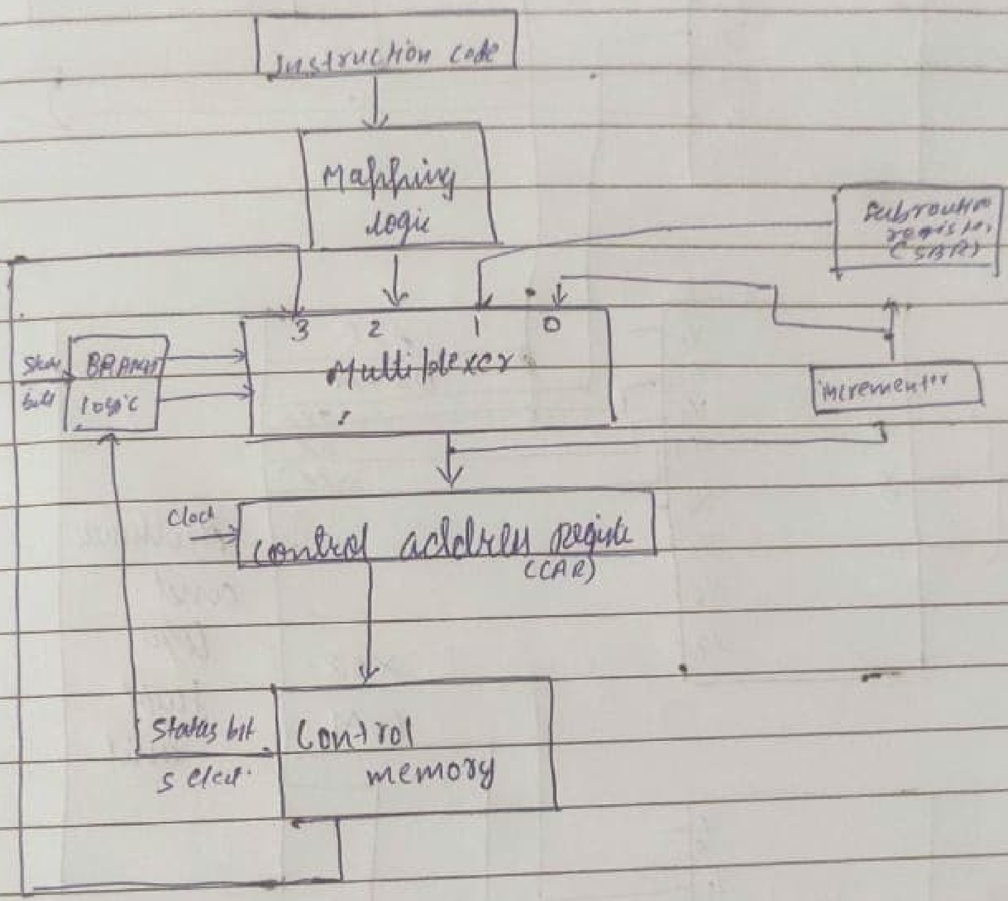
→ The μPC is then automatically incremented by the clock, causing successive microinstruction to be read from the control memory

→ The content of the microinstruction register generates control signal which are delivered to various parts of the processor in the correct sequence.

Address Sequencing :-

To generate the next micro instruction address

Selection of address for control memory .



Mapping :-

Control memory address = $\log_2 \overset{7 \text{ bits}}{4096}$ = 12 bit

0 0 x x x x x x x 0 0 0

opcode

1011 | Address

0 | xxxx | 0 0

0 1011 00 x

Date :

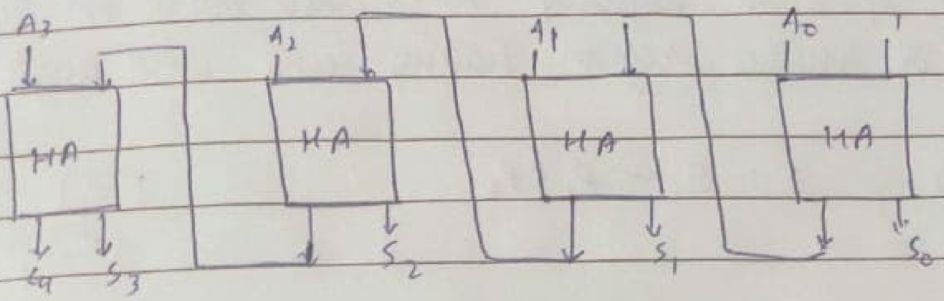
Page No.

Truth table for 16 functions of 2 variable

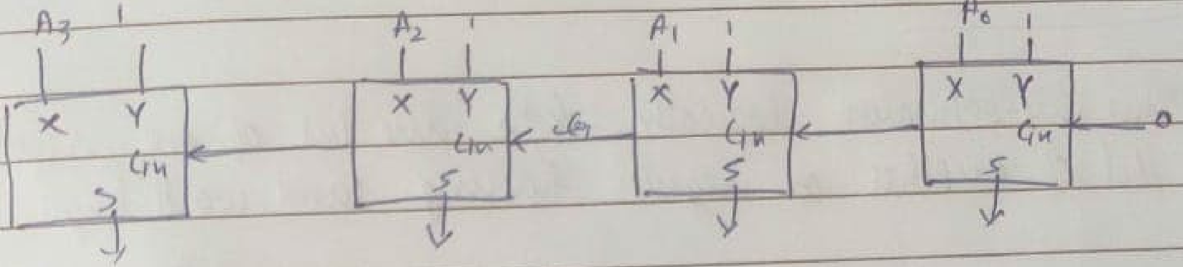
x	y	f0	f1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean function	Microoperation	Nam of operati.
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = AB$	$F \leftarrow A \wedge B$	AND
$F_2 = A\bar{B}$	$F \leftarrow A \wedge \bar{B}$	AND with second operand complemen
$F_3 = A$	$F \leftarrow A$	Transfer A
$F_4 = \bar{A}B$	$F \leftarrow \bar{A} \wedge B$	AND with first oper comple.
$F_5 = B$	$F \leftarrow B$	Transfer B
$F_6 = \overset{A \oplus B}{\cancel{A \oplus B}}$	$F \leftarrow A \oplus B$	Exclusive OR
$F_7 = A + B$	$F \leftarrow A \vee B$	OR
$F_8 = (\overline{A+B})$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = \overline{A \oplus B}$	$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F_{10} = \bar{B}$	$F \leftarrow \bar{B}$	Complement B
$F_{11} = A + \bar{B}$	$F \leftarrow A \vee \bar{B}$	OR operant with se
$F_{12} = \bar{A}$	$F \leftarrow \bar{A}$	Complement A
$F_{13} = \bar{A} + B$	$F \leftarrow \bar{A} \vee B$	OR with first --
$F_{14} = \overline{AB}$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

Incrementor / Decrementor



If the number to be incremented is 1111, the output carry C4 is generated and we get 0000 at s0 through s3



A-1 = A+2's complement of 1

2's compliment of 1 = 1's compliment of 1(0001 + 1) = 1110+1
= 1111

-> The design process is orderly and systematic.

-> More flexible, can be changed to accomodate new system specification or to correct the design errors quickly and cheaply.

-> Complex function such as floating point arithmetic can be realised efficien

-> The new or modified instruction set of CPU can be easily implemented by simply rewriting or modifying the contents of Control memory.

-> The fault can be easily diagnosed in the micro program control unit using diagnostic tools by maintaining the contents of flags, registers and counters.

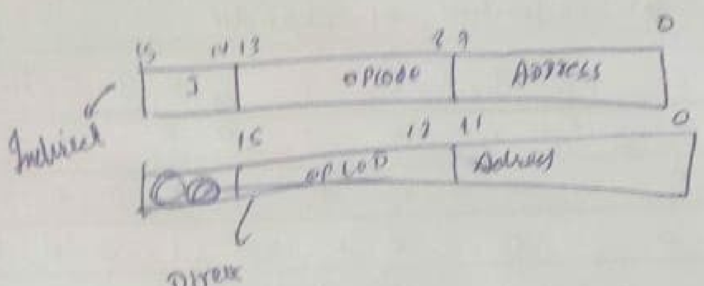
Disadvantages

-> A microprogrammed control unit is some what slower than the hardwired control unit, because time is required to access the microinstruction from CM.

-> The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

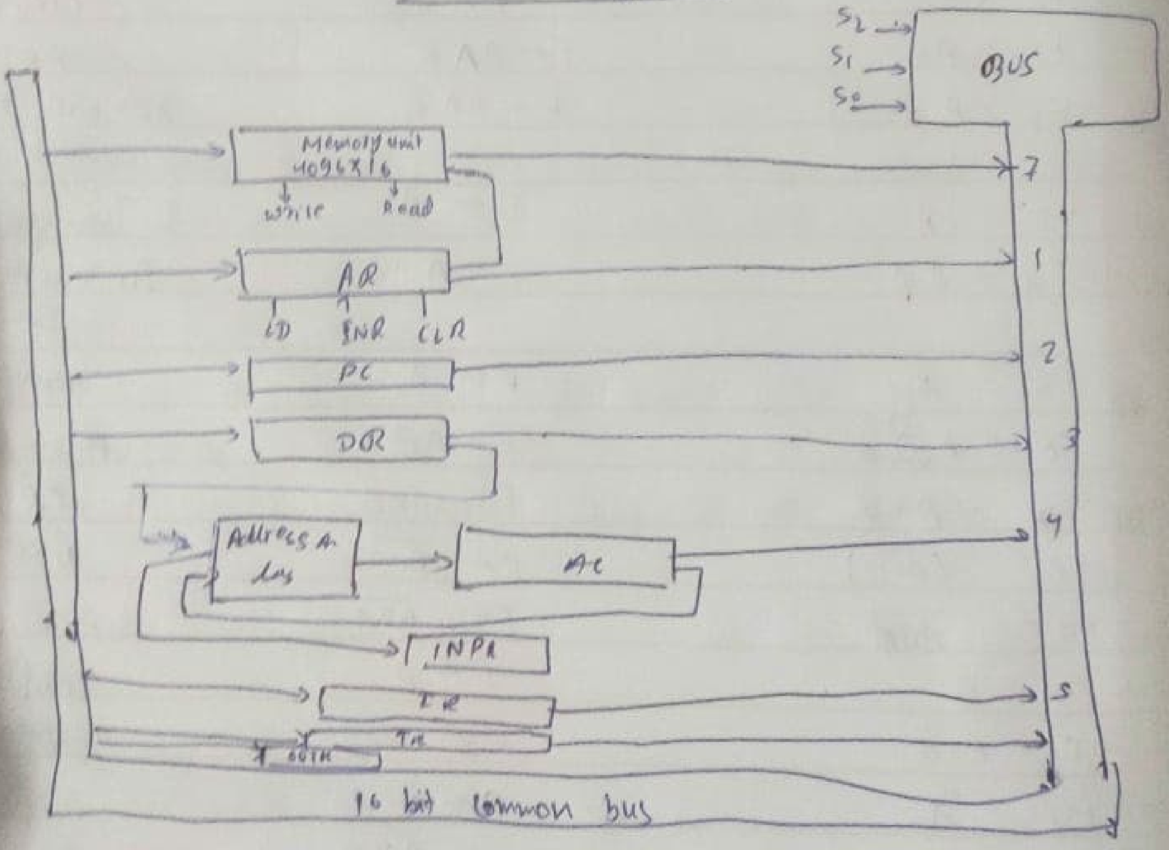
The design duration of microprogram control unit is more than hardwired control unit for smaller CPU.

Instruction Code



The address of an operand is called effective address.

Computer Register



Date :

Regi

① Data R
Memo

② Addr
Memory

③ Accu
Reg

④ Inst

⑤ Prog

⑥ Tem

⑦ In

⑧ O

①