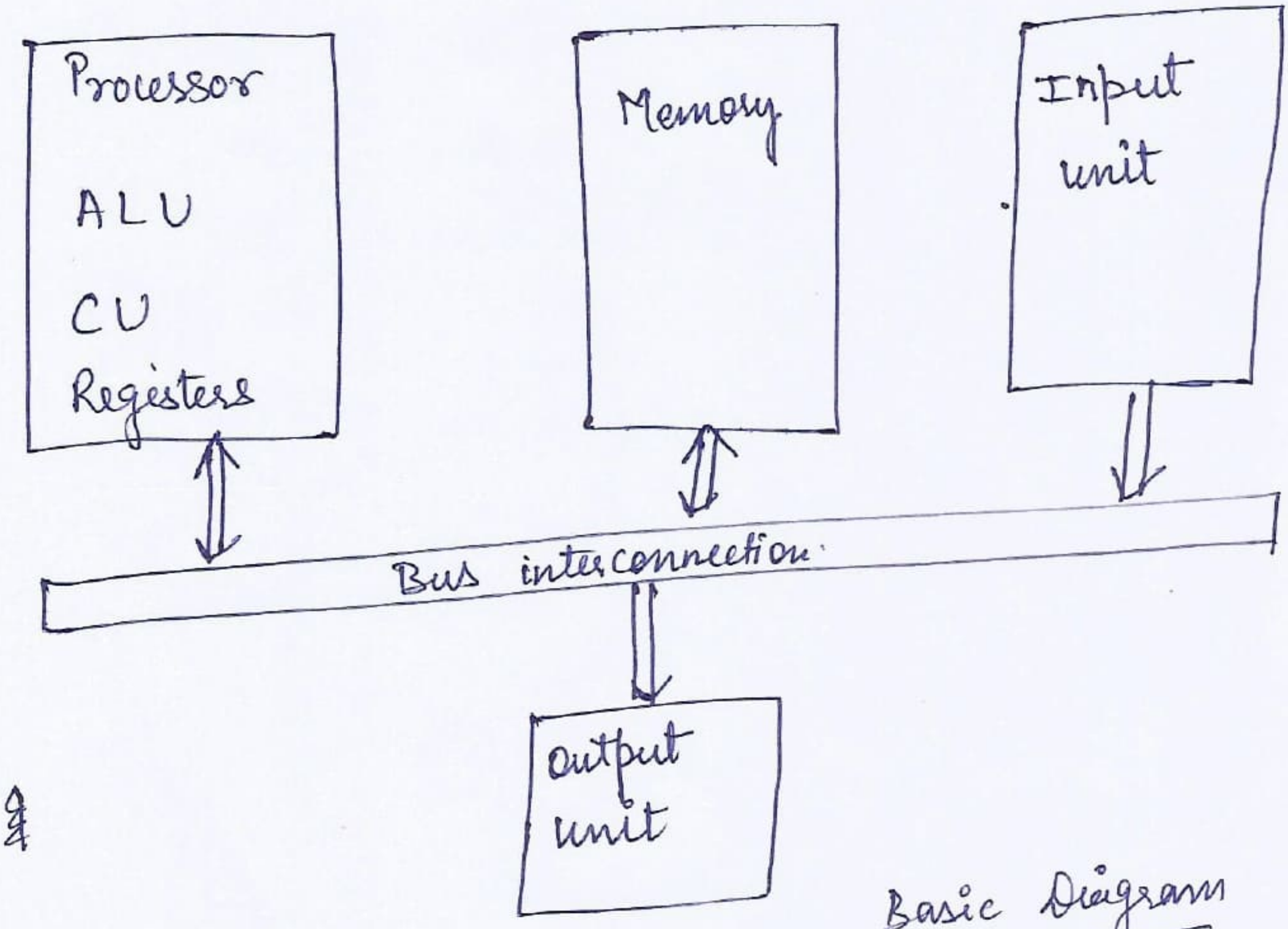
COA Unit-1 Notes

Minakshi

Functional Units and their interconnections:

→ Functional units of a compiler / digital system are:

- → Input unit
- → Output Unit
- → Central Processing Unit (CPU)
- → Memory
- → Bus structure.



Basic Diagram

Note: Theory about these components are given in "Fundamental of COA" Notes and Unit 1 Notes (about bus)

Buses, bus Architecture, types of buses and bus arbitration

Bus:

→ Bus is a communication system that helps data transfer between different modules of the computer.

→ A bus is a communication pathway connecting two or more devices.

→ A bus is a group of electrical lines/wires that carry computer signals/bits.

→ A bus consists of multiple lines. Each line is capable of transmitting signals representing 1 or binary 0.

> A bus that connects major compute components (processor, memory, I/O) is called a system bus.

→ On any bus lines can be classified into three function groups:-

1. Data Bus
2. Address Bus
3. Control Bus.

Data Bus:- Data lines provides path for moving data between components.

- Bidirectional

→ Width of databus is number of lines in data bus.

→ Each line can carry only one bit at a time.

Address Bus: (Unidirectional)

→ The address lines are used to designate (know) the source or destination of the data on the data bus.

For example: if the processor wishes to read or write a memory word, it puts the address of desired word on the address lines.

The width of address bus determines the maximum possible memory capacity of the system.

control Bus: (Bidirectional)

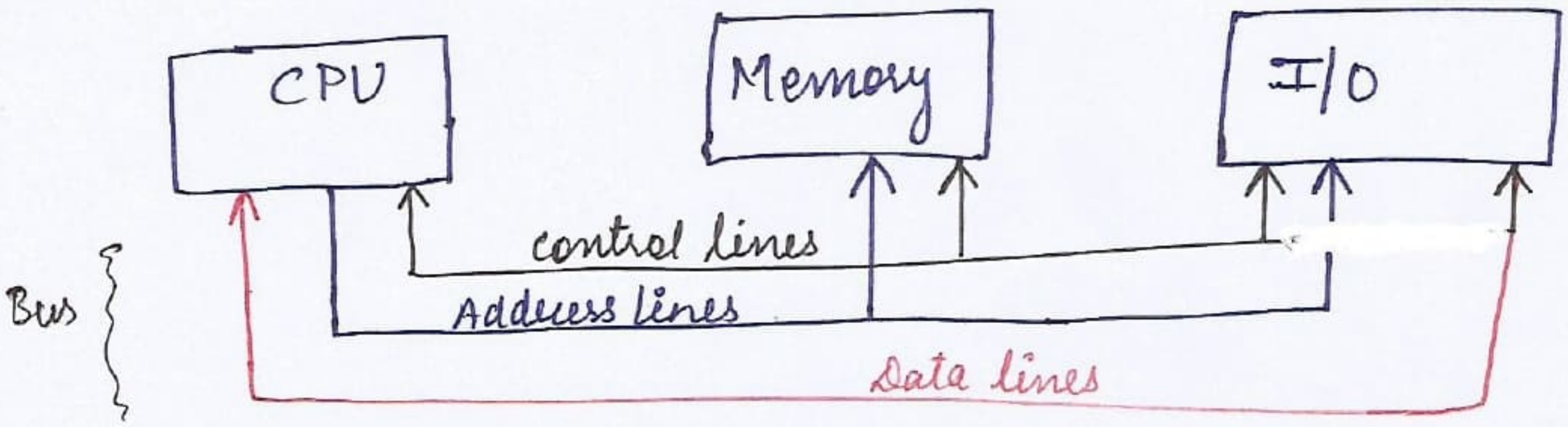
- The control lines are used to control the access to and the use of the data and address lines.
- control signals transmit both command and timming information

Timing signals ⇒ indicate the validity of data and address information

command signals ⇒ specify operations to be performed.

- Typical control lines include:-
  - Memory write
  - Memory Read
  - I/O write
  - I/O Read
  - Transfer Ack
  - Bus Request
  - Bus Grant
  - Interrupt Request
  - Interrupt Ack
  - clock
  - etc.

4.



Bus interconnection scheme.

Types of Buses:

- Dedicated
- Multiplexed

- Dedicated bus line is permanently assigned either to one function or to a physical subset of computer components.
- Example of functional dedication :- the use of separate dedicated address and data lines.
- Multiplexed: Address and data information may be transmitted over the same set of lines using an Address Valid control line.

For example:

At the beginning of a data transfer, the address is placed on the bus and Address valid line is activated. At this point The address is then removed from the bus and the same bus are used for subsequent read or write data transfer. (this known as time multiplexing)

5/

Bus Arbitration:

→ More than one module may need control of the bus e.g CPU and DMA controller!

e.g. I/O module may need to read or write directly to the memory without sending the data to the processor.

The process by which multiple requests are recognized and giving priority given to one of them is called arbitration

- Arbitration can be — Centralized / localized
  — Decentralized / Distributed

→ In a centralized scheme, a single hardware device, referred to as a bus controller or arbiter, is responsible for allocating time on the bus. The device may be a separate module or part of the processor.

- In distributed scheme, there is no central controller. Rather, each module contains access control logic and the modules act together to share the bus.

- With both methods of arbitration, the purpose is to designate one device, either the processor or an I/O module, as master. The master may initiate a data transfer (e.g. read or write) with some other device, which acts as slave for this particular exchange.

Timing

Timing Refers to the way in which events are coordinated on the bus.

- Buses use — ① Synchronous Timing
  — ② Asynchronous Timing

Synchronous Timing:

- With synchronous timing the occurrence of the events on the bus is determined by a clock.
- The bus includes a clock line upon which a clock transmits a regular sequence of alternative 0's and 1's of equal duration. A single 1-0 transmission is referred to as a clock cycle or bus cycle and defines a time slot.
- All events starts at the beginning of a clock cycle.

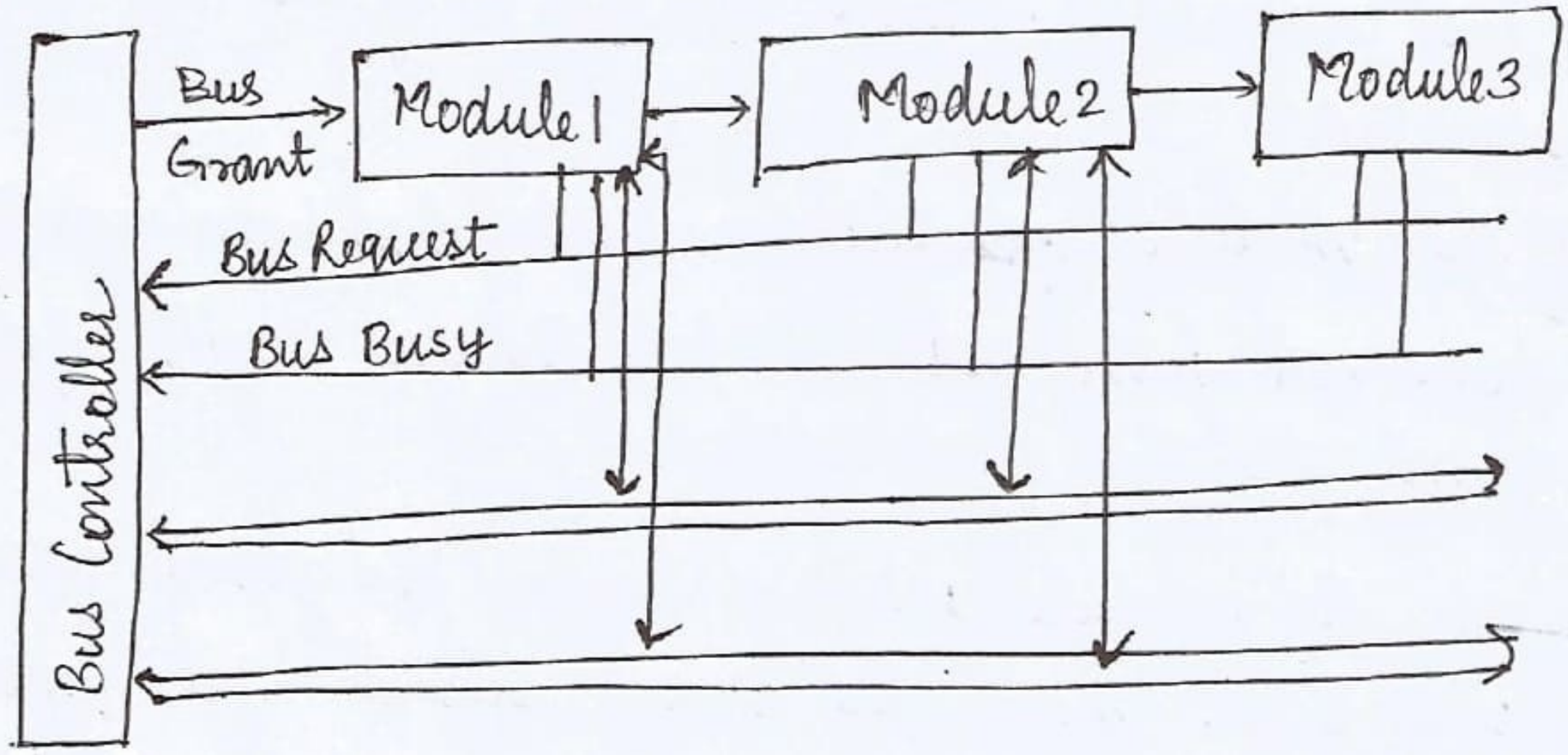
Asynchronous Timing:

→ with asynchronous timing, the occurrence of one event on a bus follows and depends on the occurrence of a previous event.

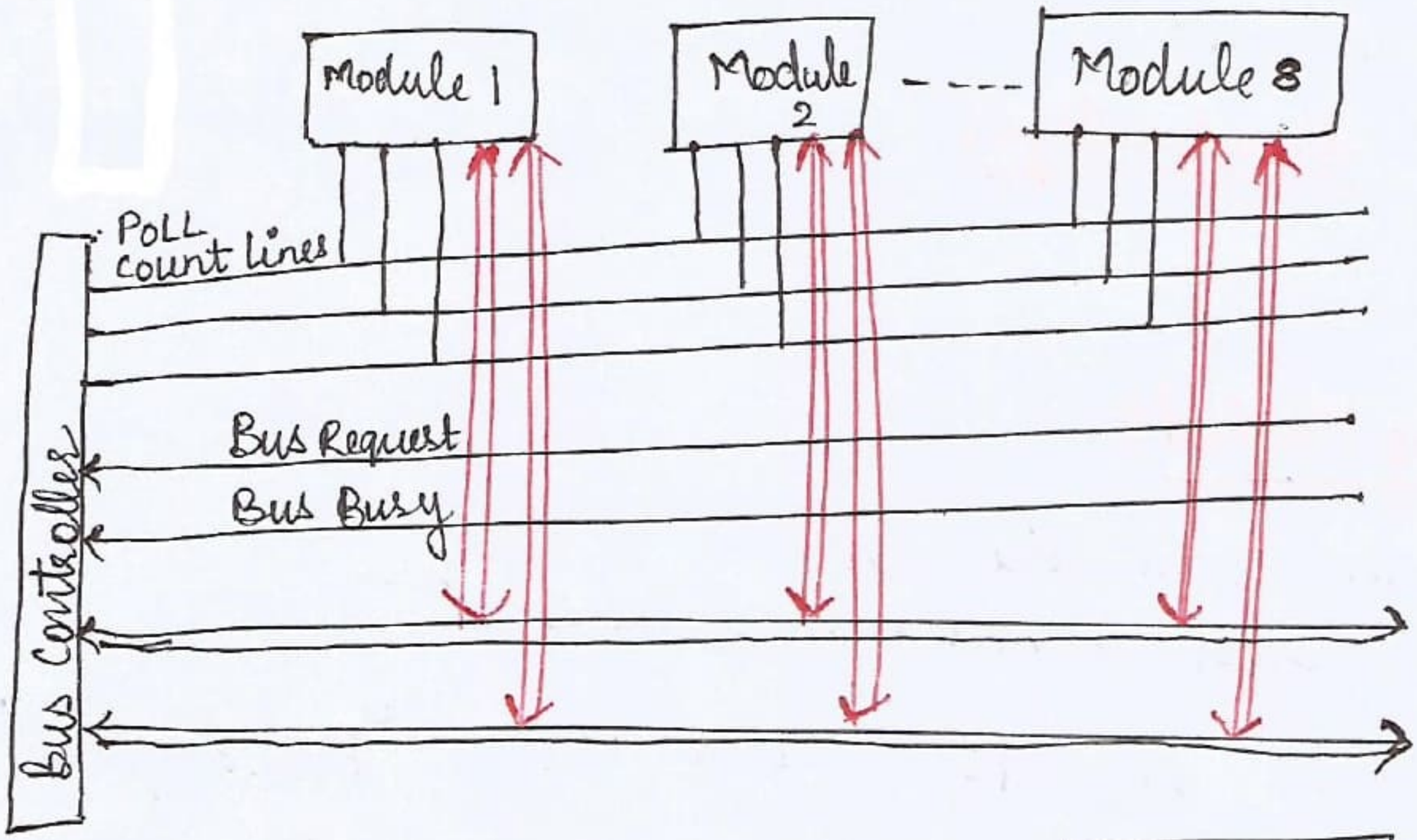
→ No clock.

Bus Arbitration Techniques:

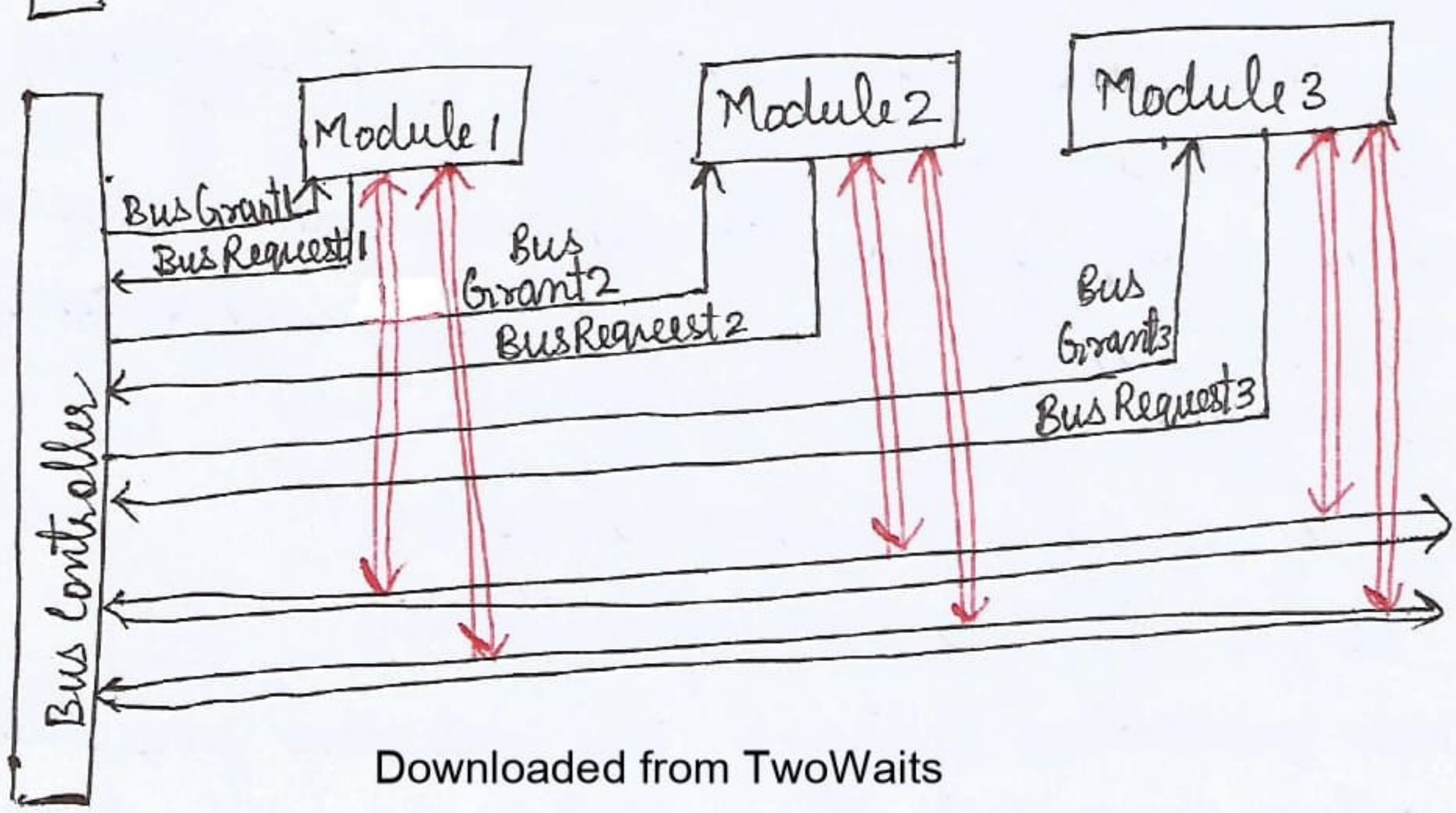
Daisy chain



Polling



Independent Requesting



Daisy-chain Method or Serial Method:

⇒ The request of bus usage can be placed by any module connected.

⇒ The bus controller grants the bus only to the first module connected.

⇒⇒ If the module ~~granted bus~~, who has been granted bus to, ~~does not~~ have not requested the bus, will forward the grant to the next module.

⇒ if the module who is having the grant wants to use bus, makes the busy signal 'set' and use the bus. After using bus, the module unset the busy signal.

Disadvantage: Modules has the fixed Priority.

Polling Method:

Polling count line = n

Max Number of connected modules ⇒ 2^n.

⇒ Modules requests the bus using 'Bus Request' line.

⇒ Bus controller places any one of 2^n sequences on the poll count lines.

⇒ The module, for which polling sequence is allocated by the bus controller, can set busy signal and use the bus.

⇒ Priority is set by the bus controller.

⇒ It is time consuming method.

⇒ Non-productive Polling ⇒ Bus controller generated the polling sequence for the module which did not request the bus

Independent Request Arbitration Technique:

→ In this method, every module has connected to the bus controller by separate 'Bus grant' and 'Bus Request'

⇒ Individual module can request to the bus controller by its own 'Bus Request' line. And, Bus controller can grant the bus to the module which has requested for the bus.

⇒ But this method increases the cost of the system.

⇒ Better Performance.

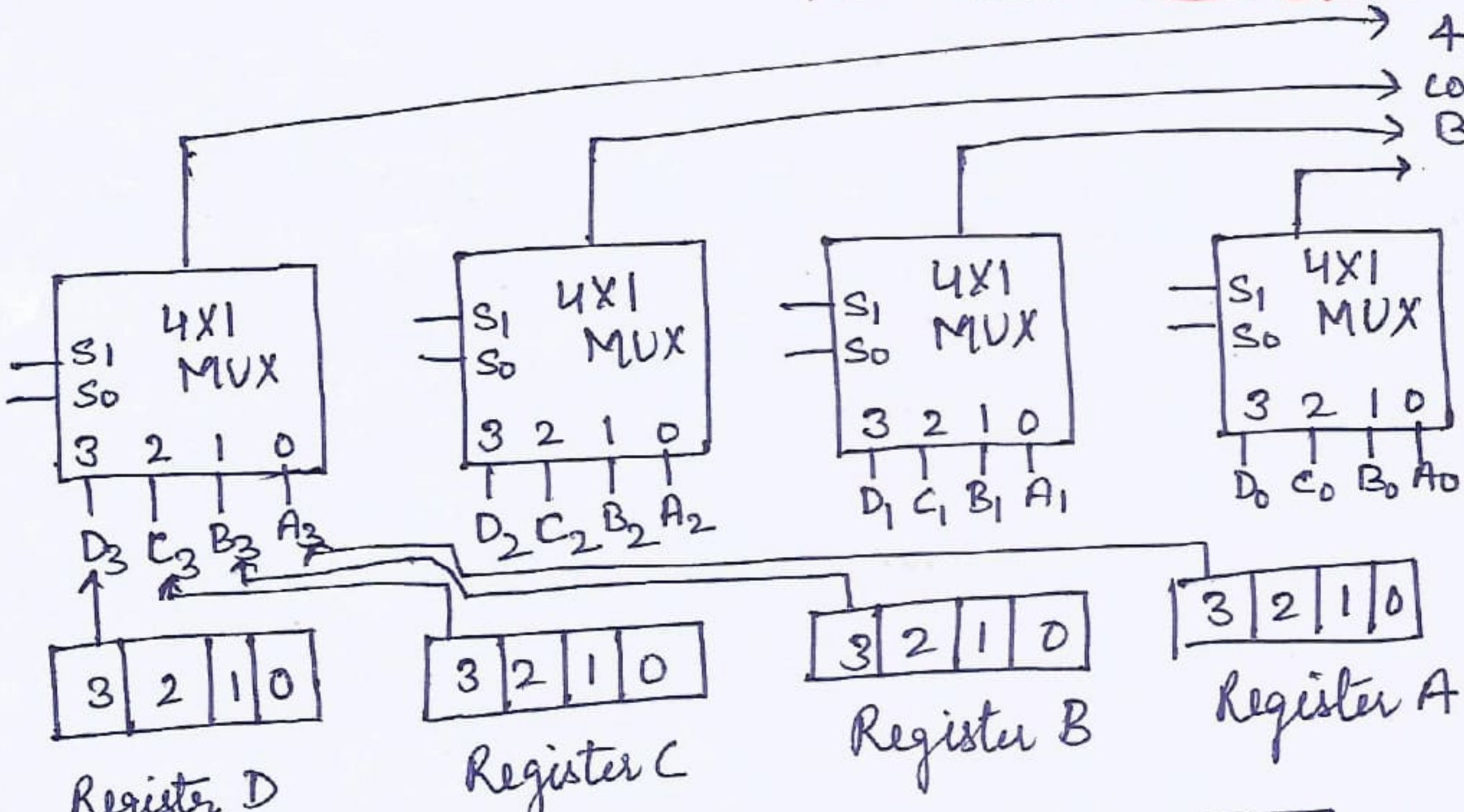
Bus and Memory Transfer:

→ A typical digital computer has many registers.
→ A path must be provided to transfer information from one register to another.
→ common bus structure
  → using Multiplexer.
  → Using Tristate Buffer.

Bus system using Multiplexer:

Example ⇒ For four registers.

Number of multiplexers = Size of register.
Size of multiplexer = Number of registers



Function Table:

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

Memory Transfer:

~~Memory Read: The transfer of information to be stored into memory is~~

MemoryRead: The transfer of information from memory to the outside environment is called a read operation

Read: DR ← M[AR]

DR ⇒ Data Register

M[AR] ⇒ memory location specified by Address Register AR.

Memory Write: The transfer of ~~new~~ new information to be stored into the memory is called a write operation

~~MR~~ Write: M[AR] ← DR

Registers

- Registers are used to store data temporarily

→ The register in CPU performs two roles

- User Visible Register
  - General Purpose Register
  - Data Register
  - Address Register
  - Condition codes
- control and status Register
  - Program Counter
  - Instruction Register
  - MAR
  - MDR

User Visible Registers:

- A user visibe register may be referenced by means of machine language that the processor executes.
- categories – General Purpose
  - Data Register
  - Address Register
  - Condition Codes

General Purpose Registers:

- can be used by the programmer.
- contains operands for any opcode.
- can be used for addressing functions (e.g. Register indirect, displacement).

Data Register: Data Register may be used only to hold the data

Address Register:

Address Register may be devoted to a particular addressing modes. Examples
- Segment Pointer
- Index Register
- Stack Pointer.

- Segment Pointer holds the address of the base of the segment in segmented addressing.

- Index Registers are used for indexed addressing and may be auto indexed.

- Stack Pointer points to the top of the stack. This allows implicit addressing that is push, pop and other stack instructions need not ~~sto~~ contain an explicit stack operand.

condition codes (Also referred to as Flags):

condition codes are the bits set by the processor hardware as the result of operation.

Example:
- positive or negative result
- Zero Result
- Overflow
- etc.

control and status Register:

→ Registers employed to control the operation of processor.
→ Not visible to the user.
→ Four Registers are essential to instruction execution

① Program counter
② Instruction Register
③ Memory Address Register (MAR)
④ Memory Data Register (MDR)

- Program counter contains the address of the instruction to be fetched
- Instruction Register contains the instruction most ~~recently~~ recently fetched.

→ MAR contains the address of a location in the memory.
→ MDR or MBR (Memory Buffer Register) contains a word of the data to be written to memory or the word most recently read.

PSW : Program status word:

Many processors include a set of registers known as PSW, that contain the status information
- condition code and other status information

Sign: contains the sign bit of the result.
Zero: set when the result is zero.
Carry: set if an operation resulted in a carry into or borrow out of a higher order bit.

Equal: set if a logical compare result is equal.

Overflow: used to indicate arithmetic overflow.

Interrupt Enable/disable: used to enable or disable interrupts

supervisor :- indicates whether the processor is executing supervisor or user mode.
Certain previledged instructions can be executed only in supervised mode, and certain memory areas can be accessed only in supervisor mode.

Topic

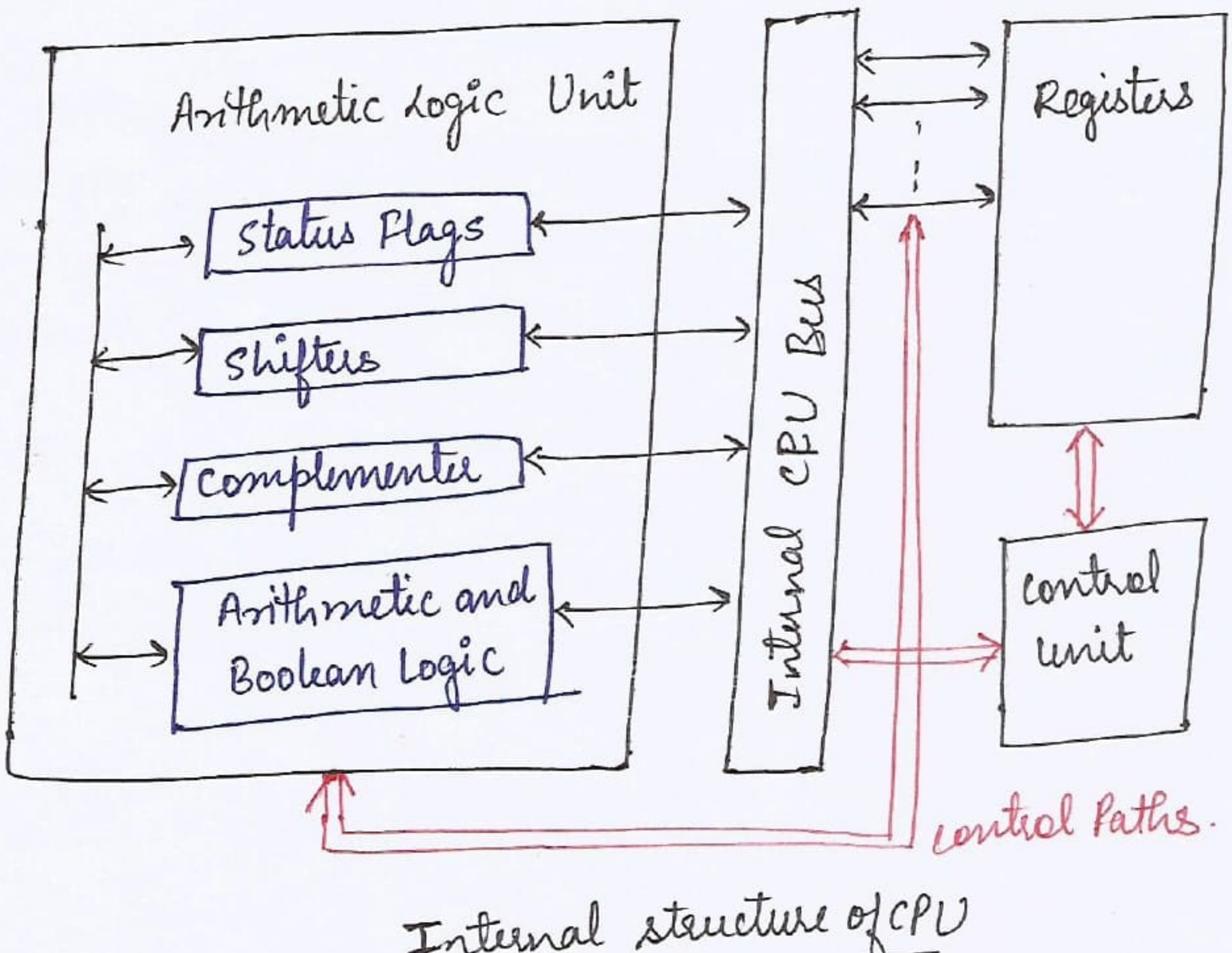
Processor Organization:

Processor contains ALU, CU and Registers

→ ALU ⇒ The ALU (Arithmetic Logic Unit) does the actual computation or processing of the data

→ CU: control unit:- CU controls the movement of data and instruction into and out of the processor and controls the operation of ALU.

→ Registers: minimal internal memory.



Internal structure of CPU

● Processor Organization means how the components of processor are connected and accomplish their tasks.

A processor does the following things.

- Fetch Instruction
- Interpret instruction
- Fetch Data
- Process Data
- Write Data

- Fetch Instruction: The processor reads an instruction from memory.

- Interpret Instruction: The instruction is decoded to determine what action is required.

- Fetch data: The execution of an instruction may require reading data from memory or an I/O module

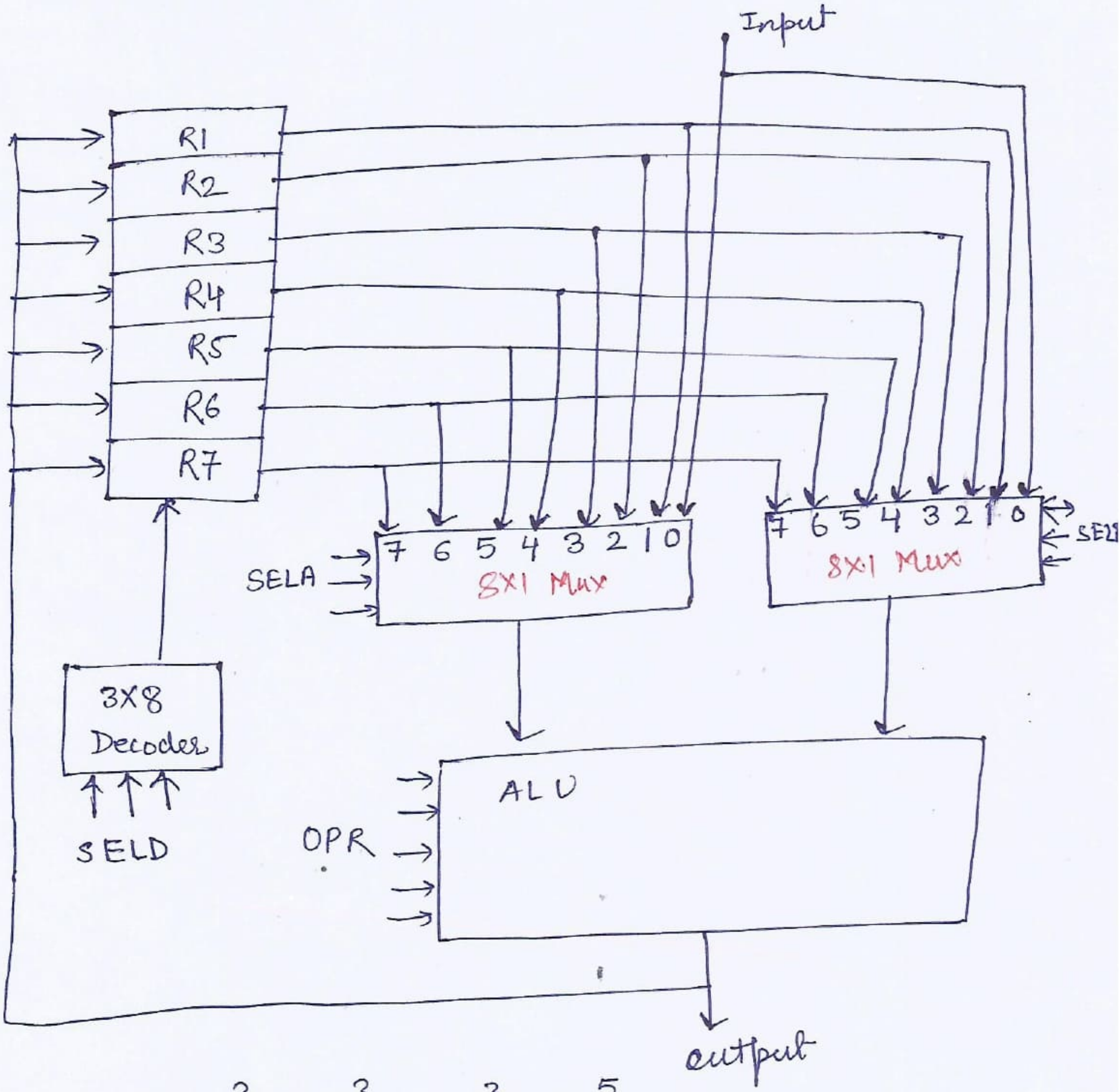
- Process Data:- The execution of an instruction may require performing some arithmetic or logical operation on the data.

- Write Data: The result of an execution may require writing data to memory or I/O module.

Topic

General Register Organization:

→ A bus organization for seven CPU registers.



3	3	3	5
SELA	SELB	SELD	OPR

control word.

Register set with common ALU

- The output of each Register is connected to two multiplexers to form the buses A and B.
- The selection lines in each multiplexer selects one register or input for the particular bus.

→ The buses A and B form the inputs to a common ALU.

→ The operation selected in the ALU determines the arithmetic or logic microoperation that is to be performed.

→ The result is available for output data and goes into the inputs of ~~the~~ all registers. Register for output is selected by a decoder.

⇒ The decoder activates one of the register load inputs.

Register Selection:

Binary Codes	SELA	SELB	SELD
000	input	input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Table of operations performed by ALU

	OPR select	Operation
0	00000	Transfer A
1	00001	Increment A
2	00010	Add A+B
5	00101	Subtract A-B
6	00110	Decrement A
8	01000	AND A, B
10	01010	OR A, B
12	01100	XOR A, B
14	01110	complement A
16	10000	shift Right A
24	11000	shift Left A

This table specifies some selected operants only ⇒ 11

five bits can represent total 32 operation

Topic.

Stack Organization:

→ stores information is LIFO order (Last In First Out)

→ The register that holds theaddress of the stack is called Stack Pointer.

→ Stack Pointer's value always points to the top element.

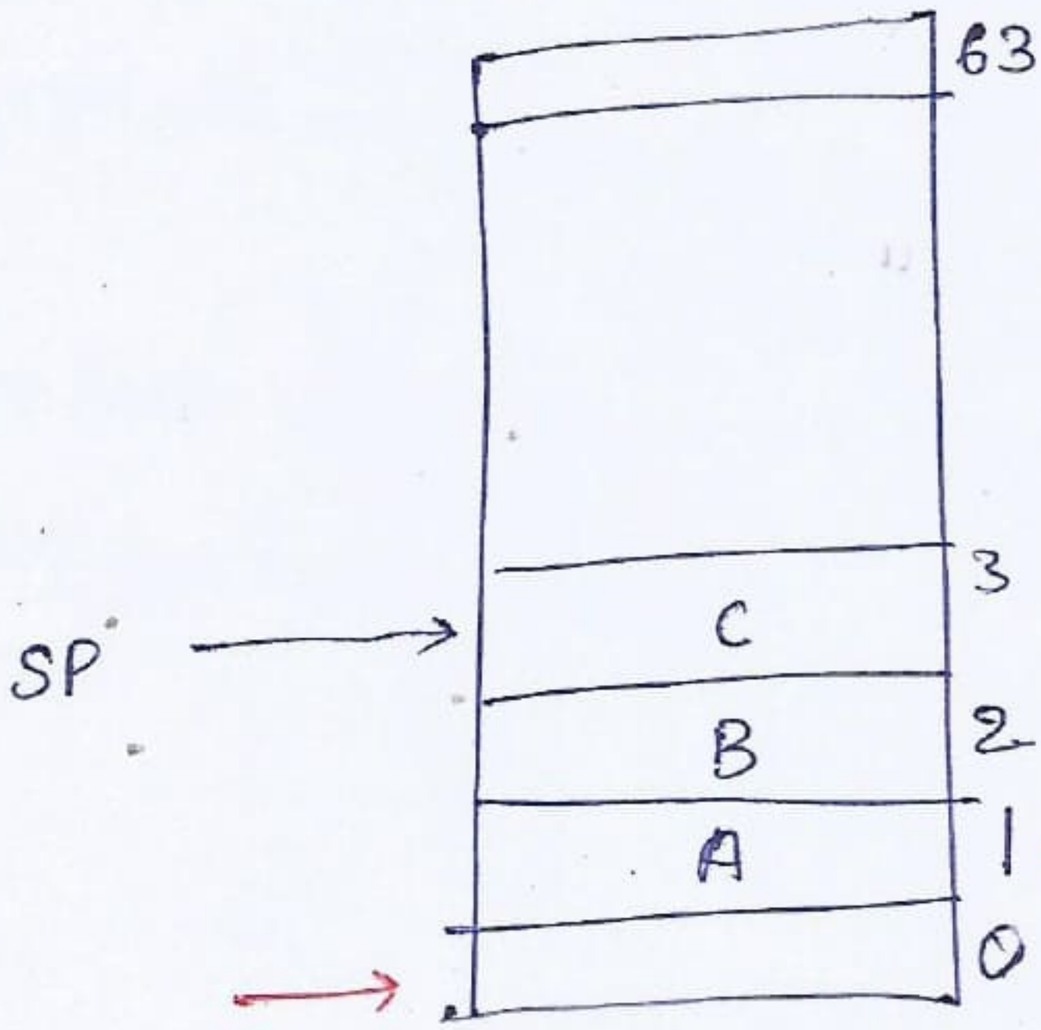
→ Two operations on stack ① Push
② Pop

⇒ Push ⇒ to insert an element in the stack
Pop ⇒ to delete an element foom the stack.

⇒ Two status Full ⇒ set to 1 when stack is full
Empty ⇒ set to 1 when stack is empty

⇒ Stack can be ⇒ collection of memory words or registers

⇒ Example 64 - word register stack



Full
Empty

Full → 1 ⇒ Full
0 ⇒ not full

Empty → 1 Empty
0 not empty

initially SP ← 0
EMPTY ← 1
FULL ← 0

if (Full = 0) ⇒ new item is inserted with push operation
if (EMPTY = 0) ⇒ an item can be deleted from stack with popoperation

Push operation :-

SP ← SP + 1 increment stack Pointer
M[SP] ← DR write item on the top of stack
if (~~Reset~~ SP = 0) then FULL ← 1
EMPTY ← 0

M[SP] denotes the memory word specified by the address presently ~~specify~~ available in the SP.

⇒ The first item is stored in stack at address 1.
The last item is stored in the stack at address 0.

Pop Operation :

A new item is deleted from the stack if stack is not empty

DR ← M[SP] ⇒ Read the data from the top of stack.
SP ← SP - 1 ⇒ Decrement SP
if (SP = 0) then EMPTY ← 1 check if the stack is empty
FULL ← 0 ⇒ Mark the stack not full.

ADDRESSING MODES:

Two issues: ① How is the address of an operand specified?
② How the bits of an instruction organized to define the operand addresses and operation of that instruction.

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in the computer registers or memory words.
The way the operands are choosen during the program execution depends on the addressing mode of the instruction

The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Most Common addressing techniques:

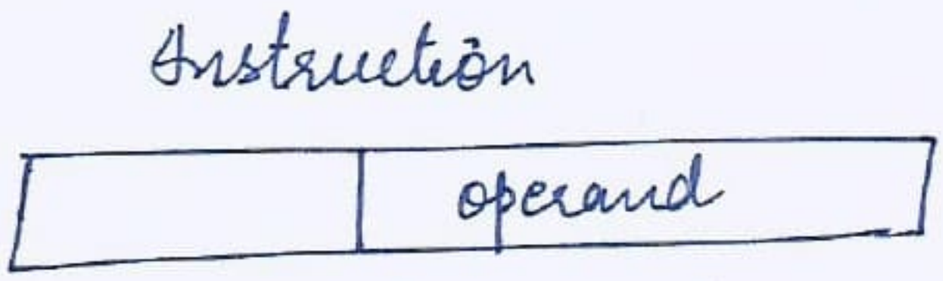
- Immediate Addressing
- Direct Addressing
- Indirect addressing
- Register Addressing
- Register Indirect Addressing
- Displacement → Relative Addressing
  → Base Register Addressing
  → Indexing
- Stack addressing

① Immediate Addressing:

→ In this addressing mode, the operand value is present in the instruction.

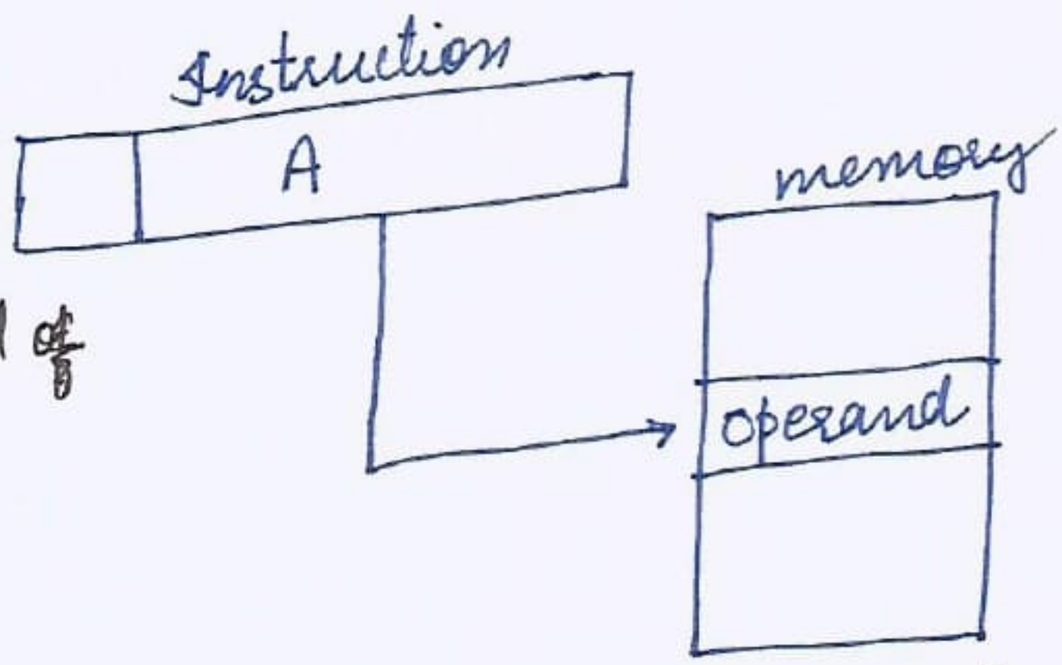
→ Advantage: no memory reference other than the instruction fetch is required to obtain the operand.

→ Disadvantage: Limited operand magnitude.



② Direct Addressing:

In this addressing mode, the address field contains the actual/effective address of the operand.



EA = A

A ⇒ content of an address field of the instruction

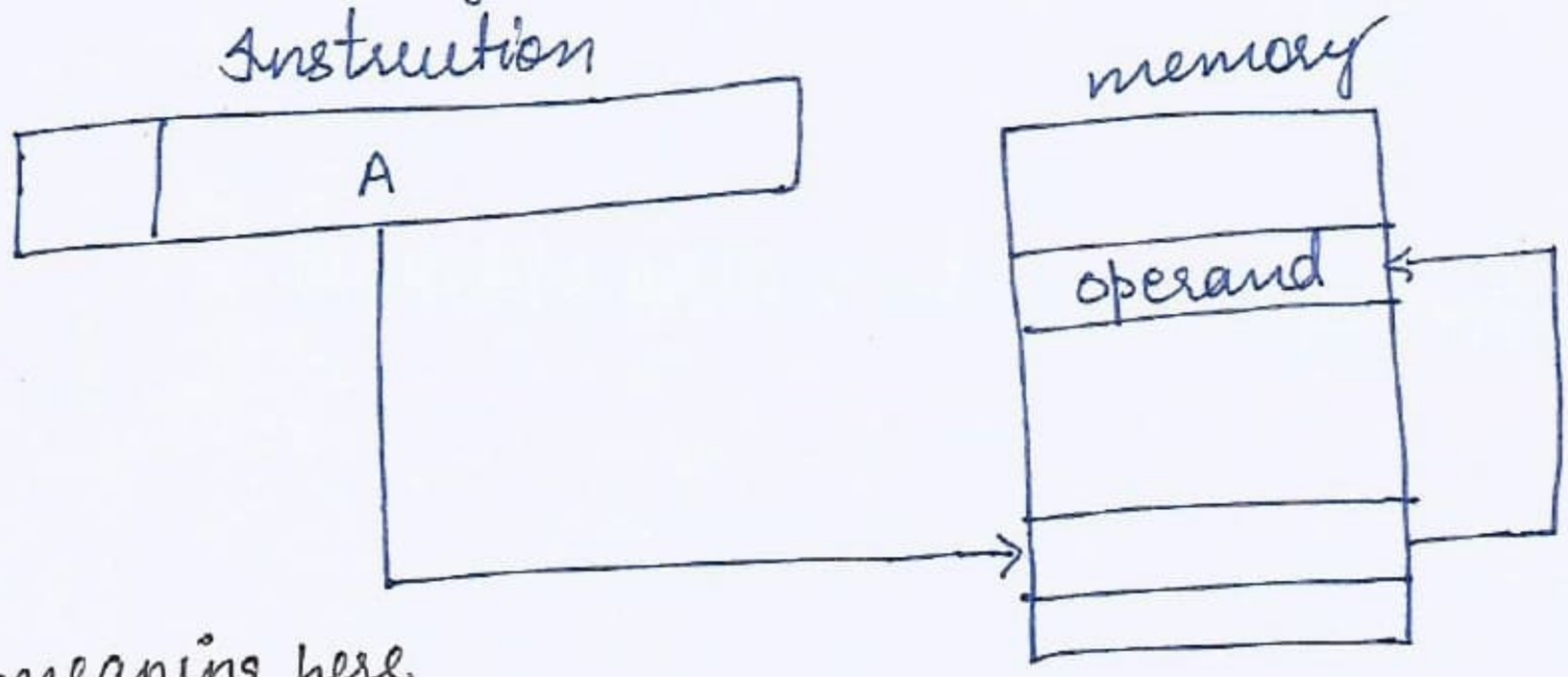
Advantage:

- It requires only one memory reference and no special calculation.
- Simple

Disadvantage: Limited address space.

(3) Indirect Addressing :

In this addressing mode, address field refers to the address of a word in the memory.



EA = (A)

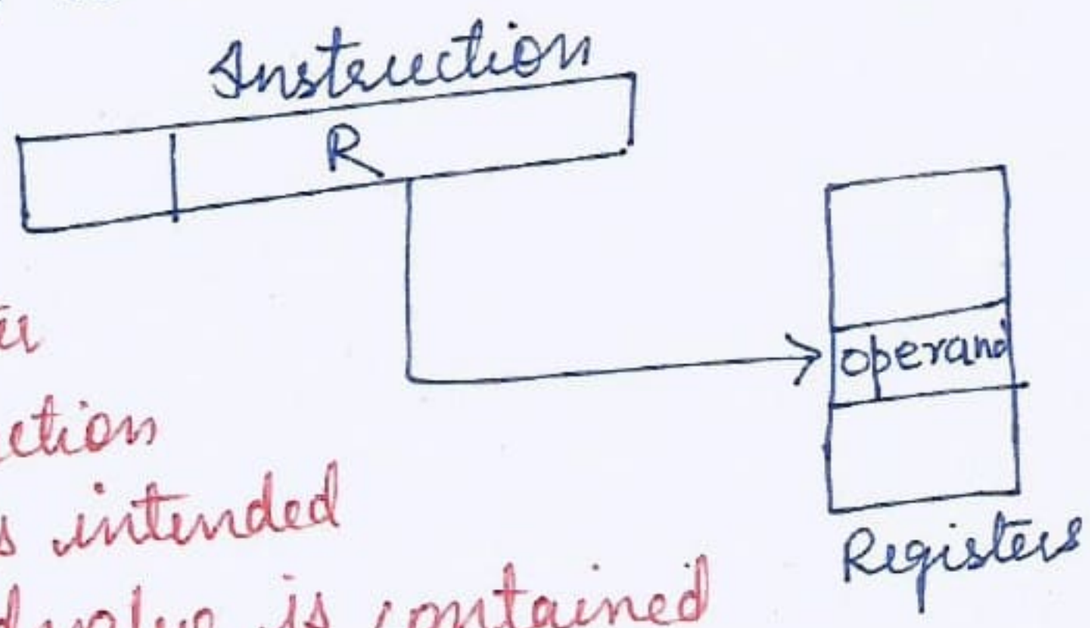
* parantheses meaning here
  ⇒ "content of"
* EA = Actual / effective address of the location containing the referenced operand.

Advantage: Large address space:
→ for a word length of N, an address space of 2^N is now available.

Disadvantage: requires two memory reference
→ one for address of operand.
→ one for to get its value (operand's)

(4) Register Addressing :

Register addressing is similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address.



EA = R

if the content of a register address field in an instruction is 5, then register R5 is intended address, and the operand value is contained in R5.

26/ Advantages:

→ only a small address field in instruction is needed
→ no time consuming memory references.

Disadvantages:

→ very limited register address space.

"Content of the register will be operand" in Register Addressing

(5) Register Indirect Addressing:

- similar to indirect addressing
- But here, the address field of instruction specifies a register whose contents give the address of the operand in the memory.

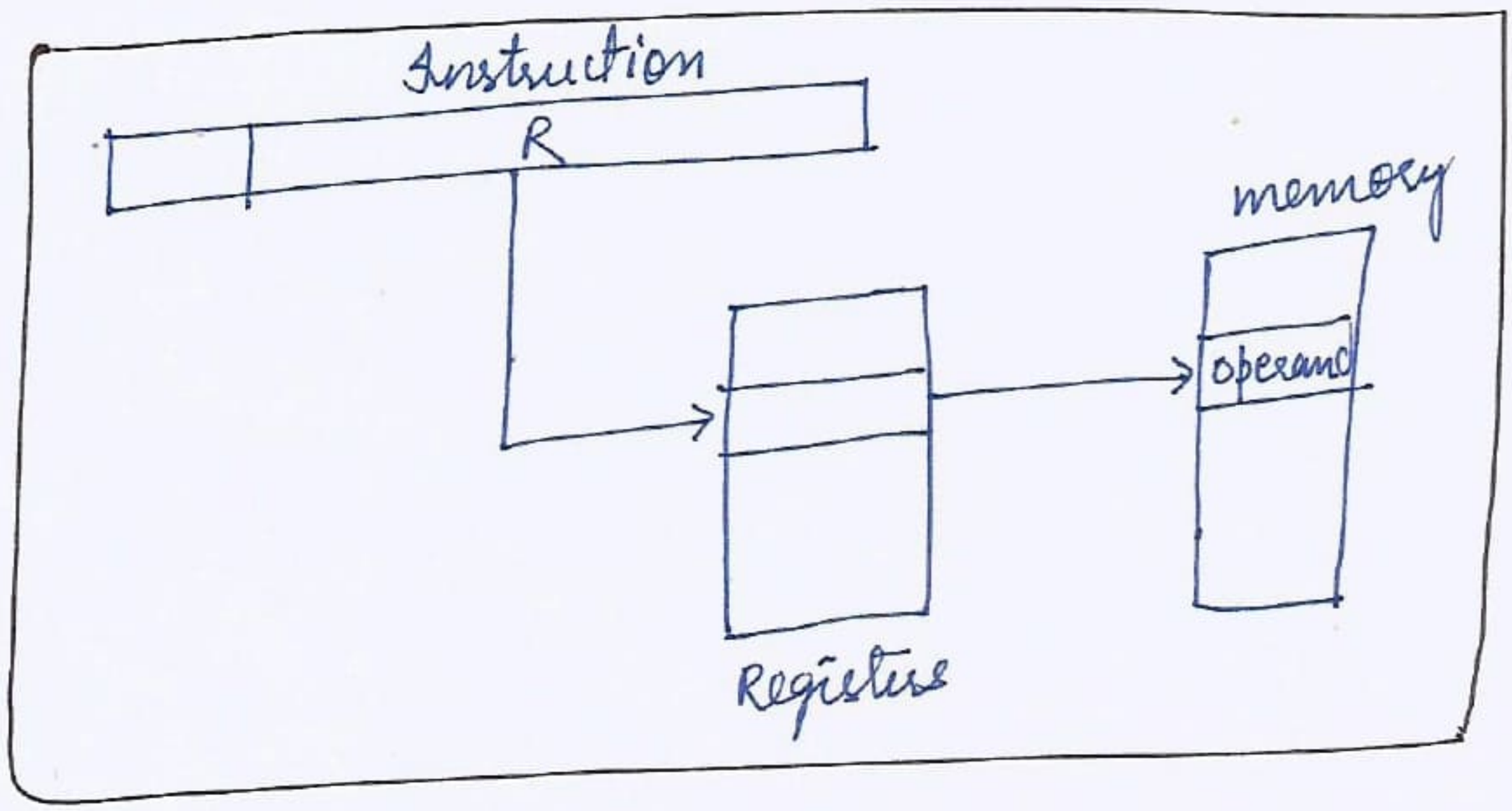
$$EA = (R)$$

Advantage:

→ fewer bits to select a register than a memory address
→ Large address space.

Disadvantage:

- Extra memory Reference.



(6) Displacement Addressing :

This addressing mode combines the capabilities of direct addressing and register indirect addressing.

$$EA = A + (R)$$

- Three most common uses of displacement addressing
  - → Relative addressing
  - → Base-register addressing
  - → Indexing

Relative Addressing :

- Also called PC-relative addressing
- In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

Example: assuming PC = 825

address part in instruction = 24

The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826.

The effective address computation for relative address mode is 826 + 24 = 850.

This is 24 memory locations forward from the address of the next instruction.

- often used with branch type instructions
- saves address bits in the instruction

Effective address = content of Program counter + address in instruction

28/

Base Register Addressing:

In this mode the content of a base register is added to the address part of instruction to get the effective address.

- A base register is assumed to hold the base address and the address part of the instruction gives a displacement relative to this base address.

Effective Address = content of Base Register + address in instruction

Indexing or Indexed Addressing mode:

- In this mode, the content of an index register is added to the address part of an instruction to obtain the effective address.
- The index register is a special CPU register that contains an index value. The address ~~field~~ of field of the instruction defines the beginning address of a data array in the memory. Each operand in the array is stored in the memory relative to the beginning address.

Effective address = content of index Register + address in instruction

(7) Stack Addressing :

- A stack is a linear array of locations.
- A stack is also referred to as a pushdown list or last-in-first-out queue.
- LIFO concept.
- The stack mode of addressing is a form of implied addressing The machine instruction need not include memory reference but implicitly operate on the top of stack.

EA = top of the stack

Advantages: No memory reference

Disadvantage: Limited applicability.

(8) Implied Mode or Implicit mode :

- Operands are specified implicitly
- Example Instruction such as CMA (complement Accumulator)
- In fact all register reference instructions that ~~are~~ use an accumulator are implied mode instructions.
- Zero address instructions in stack organized computer are implied mode instructions

(9) Auto increment or Autodecrement Mode:

30/

The auto increment mode is similar to register indirect mode except that the register (content) is incremented after the execution of the instruction.

Effective address = content of register + address in instruction
Now increment the content of register $R = R+1$

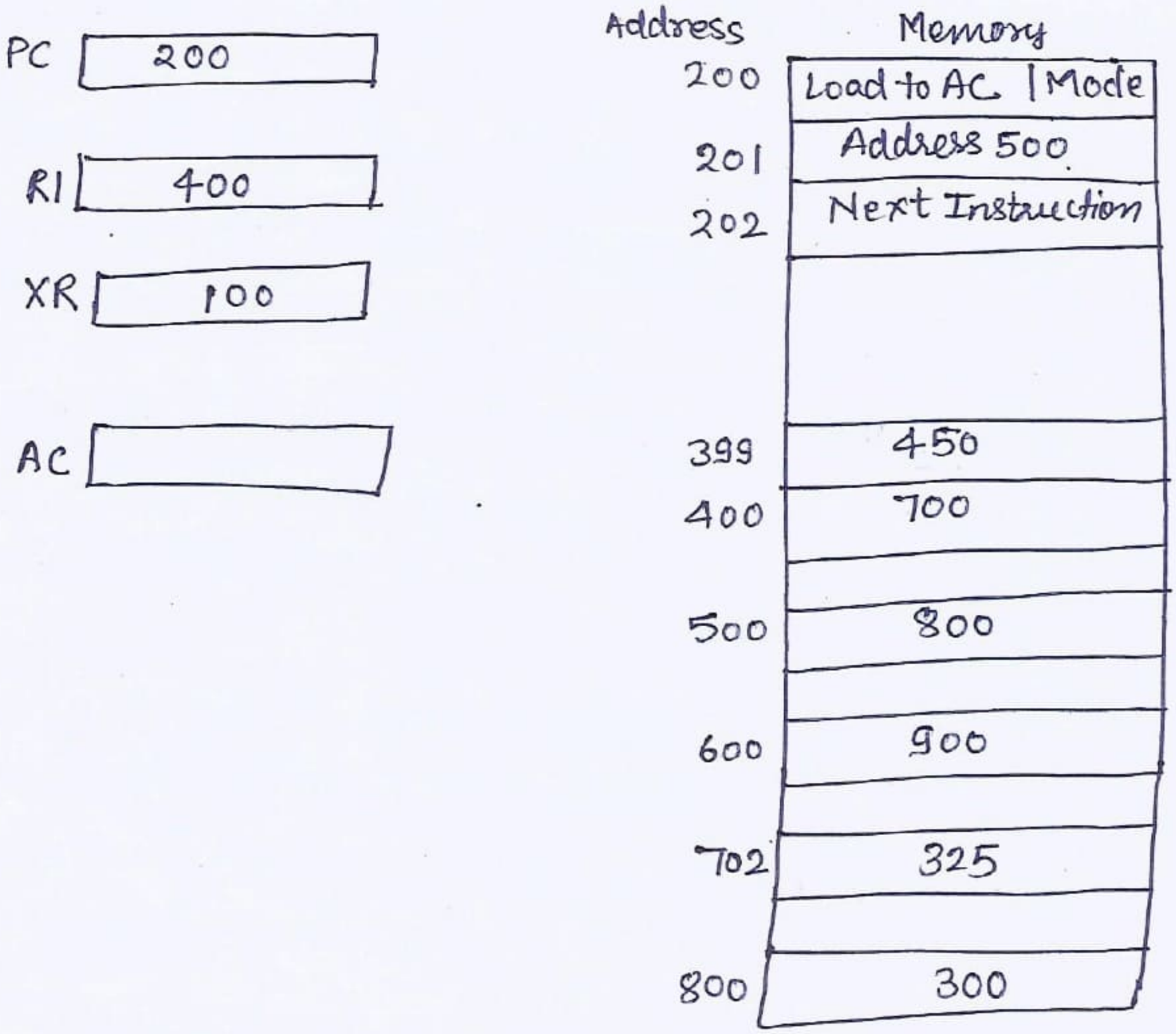
In the auto decrement mode, the content of register is decremented before the execution of instruction.

$R = R-1$
Effective address = content of register + address in instruction

Effective address means ⇒ address of operand

31

Numerical Example for Addressing Modes:



⇒ The two word instruction at address 200 and 201 is a

Load to AC	Mode	Address 500

⇒ The first word of the instruction specifies the operation code and mode, and the second word specifies the address part.

→ PC (Program Counter) has the value 200 for fetching this instruction

→ The content of process register RI is 400 and the content of index register XR is 400.

→ AC (Accumulator) receives the operand after the instruction is executed.

⇒ For each possible mode, we calculate the effective address and the operand that must be loaded into AC.

① Direct mode: ⇒ the effective address is the part of instruction.

Effective Address = 500 Operand ⇒ 800

② Immediate mode: operand is the part of instruction.

operand = 500

EA ⇒ ~~address~~ 201

③ Indirect Mode: ⇒ Effective address is stored at the address part of instruction '500'

Effective Address ⇒ 800 Operand = 300

④ Relative Mode: Effective address ⇒ content of PC + instruction address part.

EA = 202 + 500

= 702

operand = 325

⑤ Index Mode: EA = content of index Register + instruction address part.

= 100 + 500

= 600

operand = 900

⑥ Register Mode:

⇒ There is no effective address. content of the register R1 will be loaded to AC.

Operand = 400

⑦ Register Indirect mode:

Register R1 contains the effective address.

EA = 400 . Operand = 700

⑧ Auto-increment Mode:

same as Register indirect mode, but the content of Register is incremented after the execution of the instruction

EA = 400, Operand = 700 . , R1 = 401

⑨ Auto-Decremented Mode:

Same as Register indirect mode, but the content of Register is decremented before the execution of the instruction

Effective Address = 309 Operand = 450

Table

	Addressing Mode	Effective address	Operand
①	Direct	500	800
②	Immediate	201	500
③	Indirect	800	300
④	Relative	702	325
⑤	Indexed	600	900

	Addressing mode	effective address	operand.
(6)	Register	—	400
(7)	Register indirect	400	700
(8)	Auto increment	400	700
(9)	Auto-decrement	399	450