Aastha Agarwal

CS-DS

DATE ________

PAGE ________

① ~~CArch.~~ Comp. Architecture	Comp ~~Corg~~ Organisation
→ funct^{n}al. behavior	→ structural relat^{n} (connections of h/w comp)
→ deals with high level design Issue	→ low level design issue
→ fixed	→ depends on CA and then organisat^{n} is decided
→ comprises of logical f^{n}.	→ comprises of phy. units
eg → reg. data types etc	eg circuits, peripherals etc
→ What it does	→ How it does.
→ involves logic	→ involves physical comp.
→ interface b/w h/w & s/w	→ deals with h/w compon.
→ help us to understand the f^{n}alities of system	→ tells us abt how exactly every component is arranged / interconnected

Scanned by TapScanner

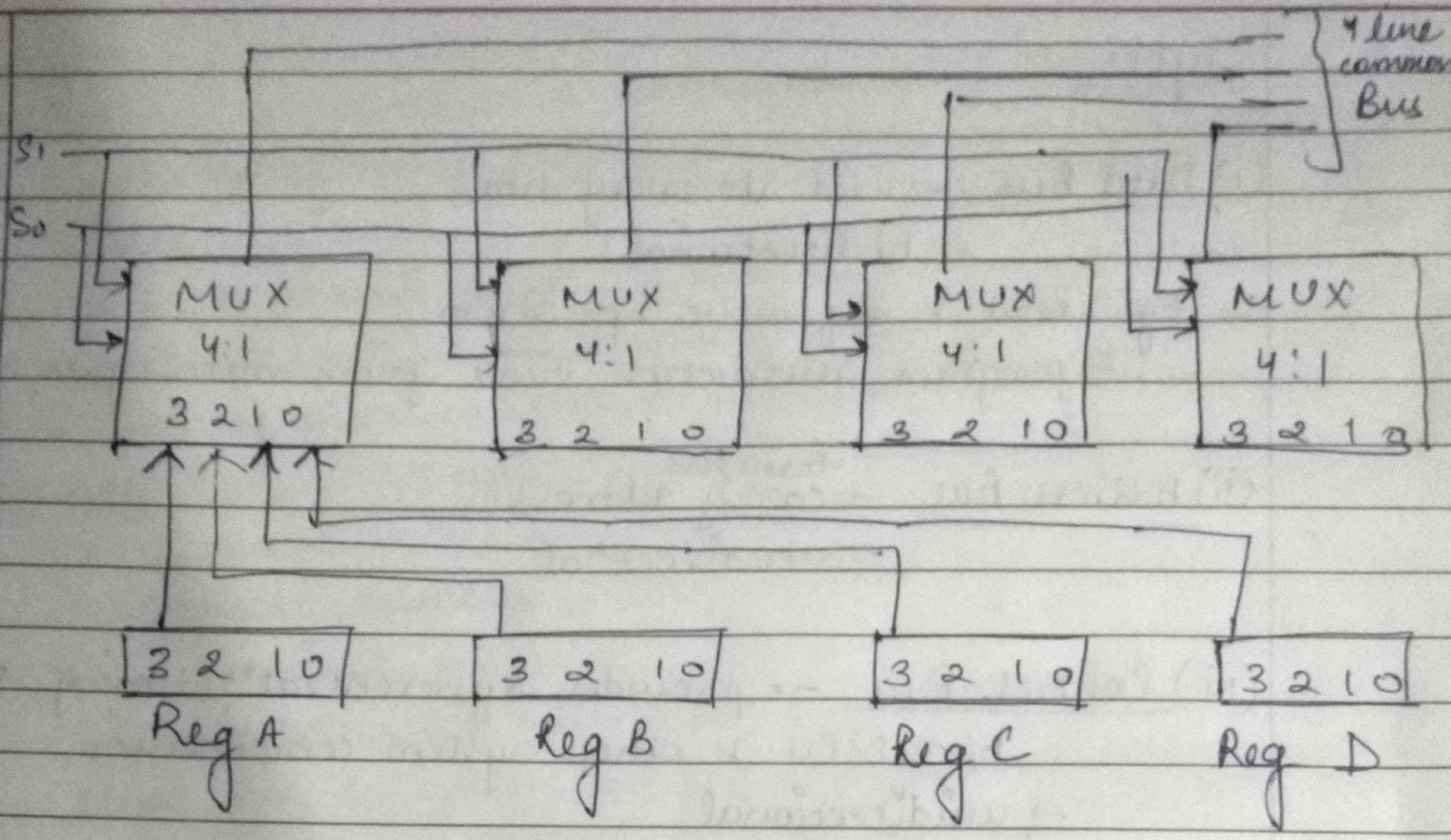
Q2.

In a Loosely coupled systems, all processors can use their local buses simultaneously. Hence, there is a contest for system bus. This is called bus arbitration

- Approaches → ① Centralised Bus Arbitration
  ② Distributed Bus Arbitration

- Types. ① Daisy Chaining Method
  ② Polling Method.
  ③ Independent Request Method.

Q3



ii) $X = A + B * [C * D + E * (F + G)]$

⇒ Zero Address.

Postfix ⟶ ABCD*EFG+*+*+

Zero Address

↳

Push A
Push B
Push C
Push D
MUL

Push E
Pushf
Push G
Add
MUL

Add
MUL
Add
pop·X

Q5 Processor Organisation

The parts of the computer that performs the bulk processing operations is called CPU.

→ CPU is made of 3 major parts.

(i) Registor. → Stores intermediate data used during execut^{n}

(ii) ALU → Performs the required microoperations for execut^{n}.

(iii) CU → Instructs ALU as to which operat^{n} to perform.

→ Types of Processor Organisation

(i) Single Accumulator → The accumulator registor is used implicitly for processing all instructions of program and storing the resuts into accumulator

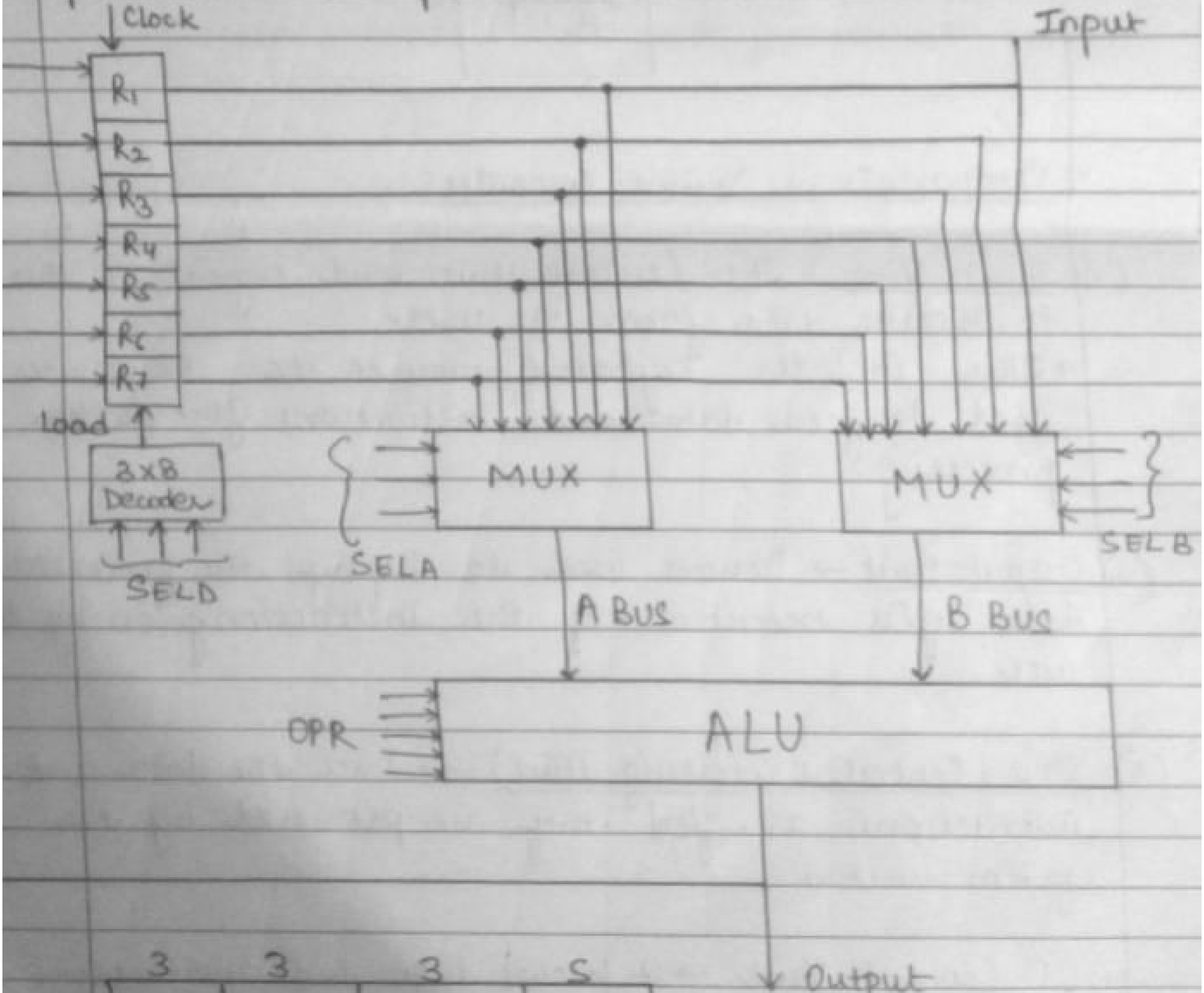
The instruction format used is One Address field

General Register Organisation.

→When a large no. of register is included in the CPU, it is best efficient to connect them through a Common Base Register.

→The register communicates with each other not only for direct data transfer but also while performing various micro operation.

→Hence, it is necessary to provide a common unit that can perform all arithmetic, logic and shift micro-operation in the processor.



3	3	3	5
SELA	SELB	SELD	OPR

(1) REGISTER STACK

A stack that can be placed in a portion of a large memory or it can be organised as a collection of a finite number of memory words or register

- SP → Stack Pointer

↳ It contains binary number whose value is equal to address of word that is currently on top of stack

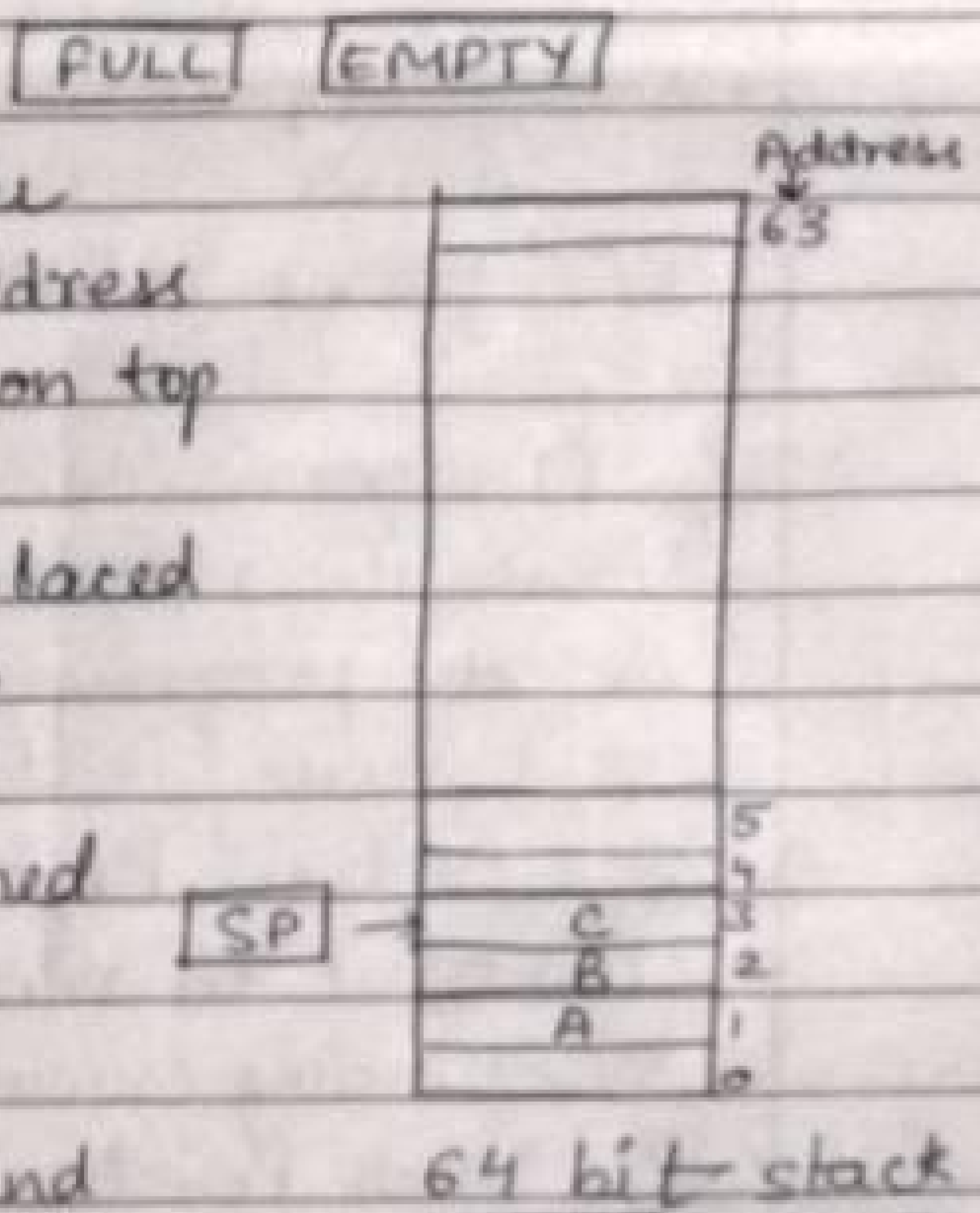
↳ Three items A, B, C are placed on stack where C is on top

↳ SP points at 3

⇒ Operations that can be performed

① Push → Insertion of new element at the top of C.

↳ SP will get incremented and will point to the next address which has the new pushed element.



64 bit stack

② Pop → Deletion of the top most element

↳ SP will get decremented and will point to the previous address having the second last element previously

- FULL → It denotes that the stack is full
  ↳ ~~Also~~ i.e one bit register (FULL) is set to one when stack is full.

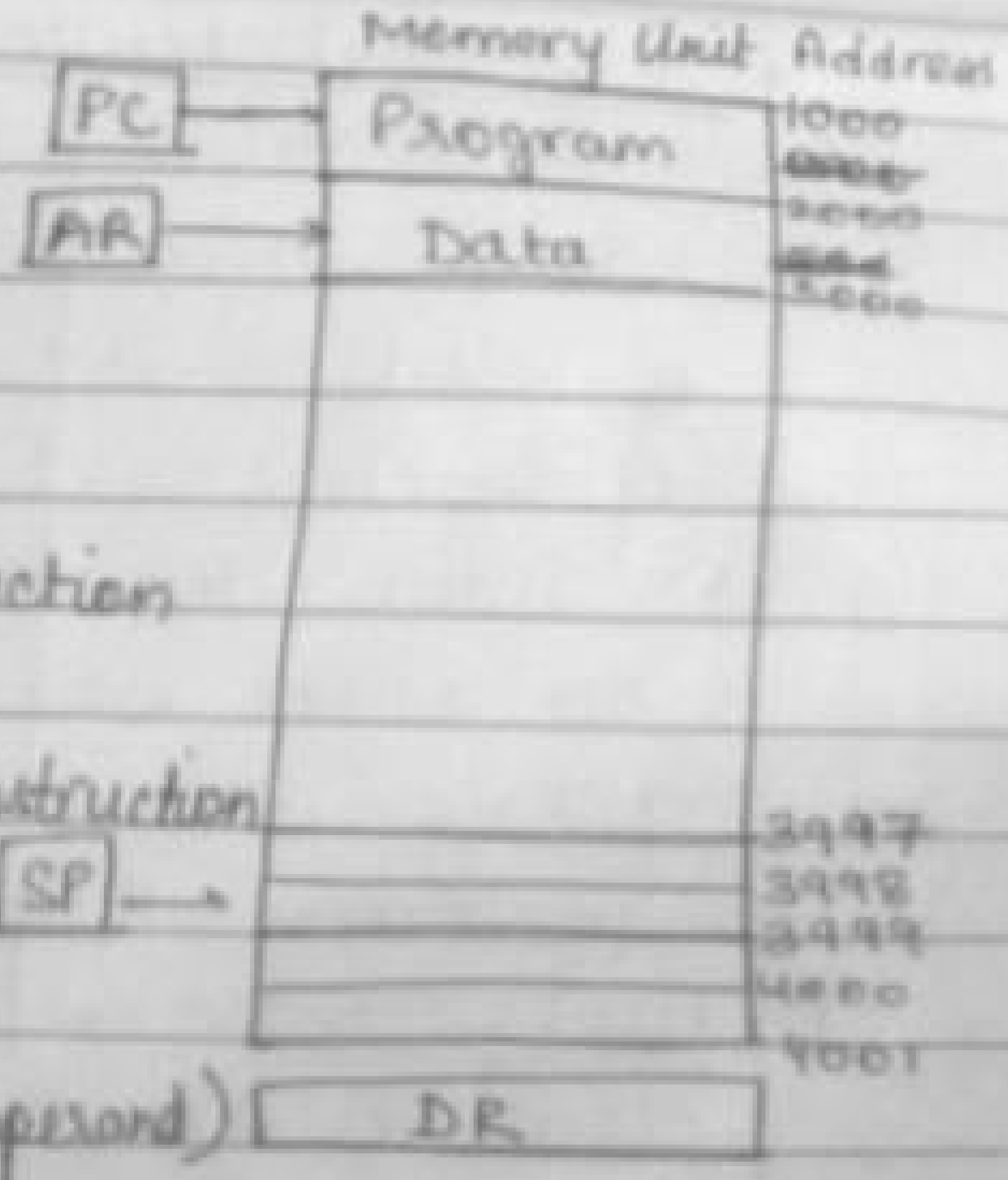
- EMPTY → one bit register EMPTY is set to zero when stack is empty.

(ii) MEMORY STACK

→ A stack which can exist as register stack or can be implemented in RAM Attached to CPU.

→ The stack is implemented by assigning a portion of memory to stack operation and using a processor register as stack pointer.

→ Memory is partitioned into 3 segments
(i) Programe (ii) Data (iii) Stack

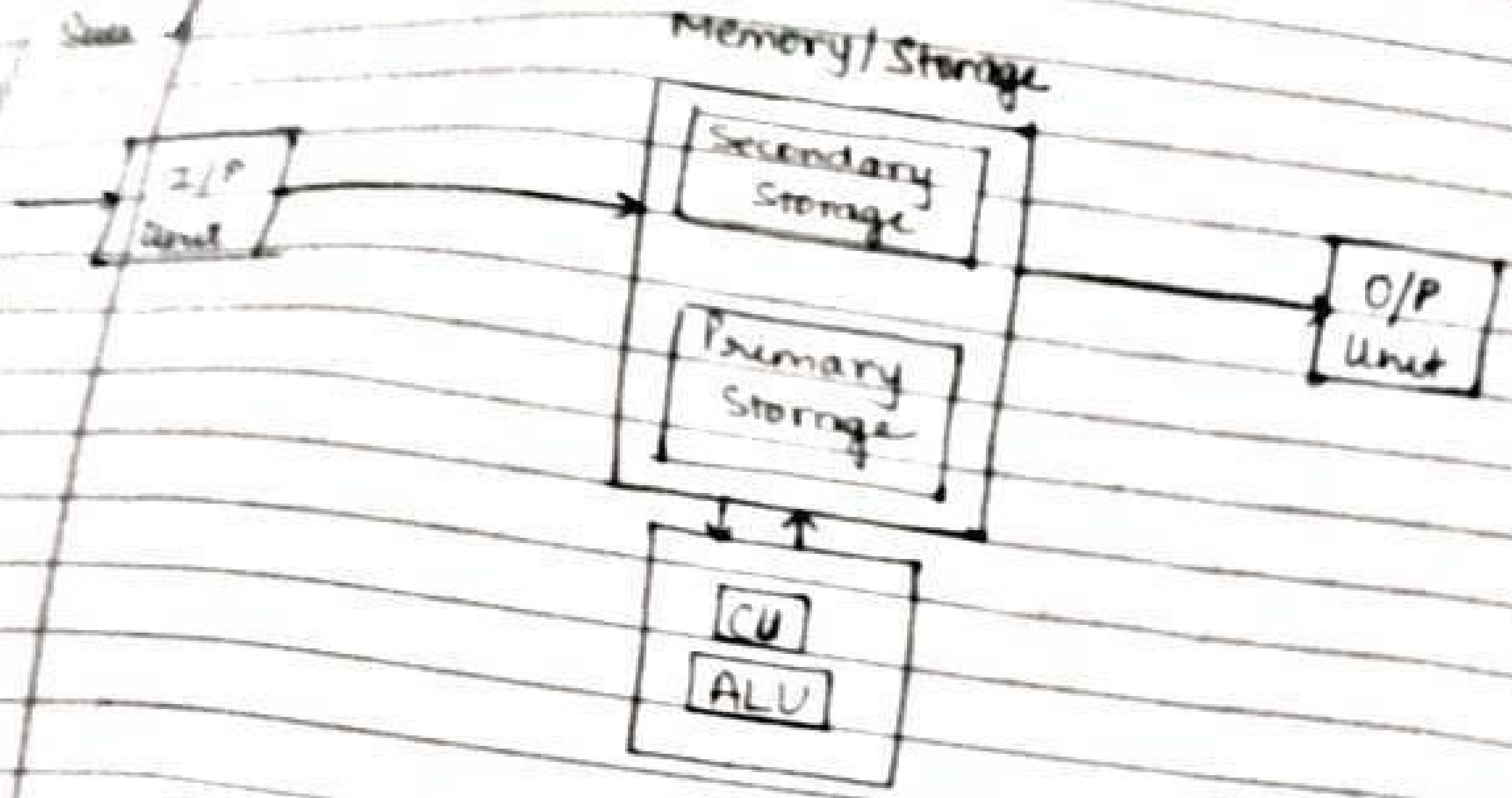


→ PC (Program Counter)
  ↳ Points the address of next instruction in the program
  ↳ Used during fetch phase to read instruction

→ AR (Address Register)
  ↳ Points at an array of data
  ↳ used during execution phase (reads operand)

→ SP (Stack Pointer)
  ↳ Points the top of the stack
  ↳ used to Push and Pop

- Advantage → CPU can refer to memory stack ~~whit~~ without having to specify an address, since the address is always available and automatically updated in SP.



→ <u>Components of Digital computers</u>.

(1) <u>Input Unit</u> → The Central Unit sends signal to this unit to receive data from the user.

→ This includes keyboard, mouses etc. i.e devices used to feed data and instructions for further processing

2) <u>Output Unit</u> → Devices used to display the processed data after execution of the instruction given by the user.

<u>CPU (Central Processing Unit)</u> → Once the data and instructions are fed they are processed by the system unit.

) CU <u>(Control Unit)</u> → It fetches data and instructions from main memory.

→ Interprets the instructions.

→ Controls I/P and O/P Devices.

ii) <u>Arithmetic and Logical Unit (ALU)</u> → All the mathematical and logical operations are carried out in this unit.

(4) <u>Memory</u> → This is the storage device. Data and instruction are stored in memory in the form of 0 and 1 The memory of computer consists of main memory, secondary memory and cache memory.

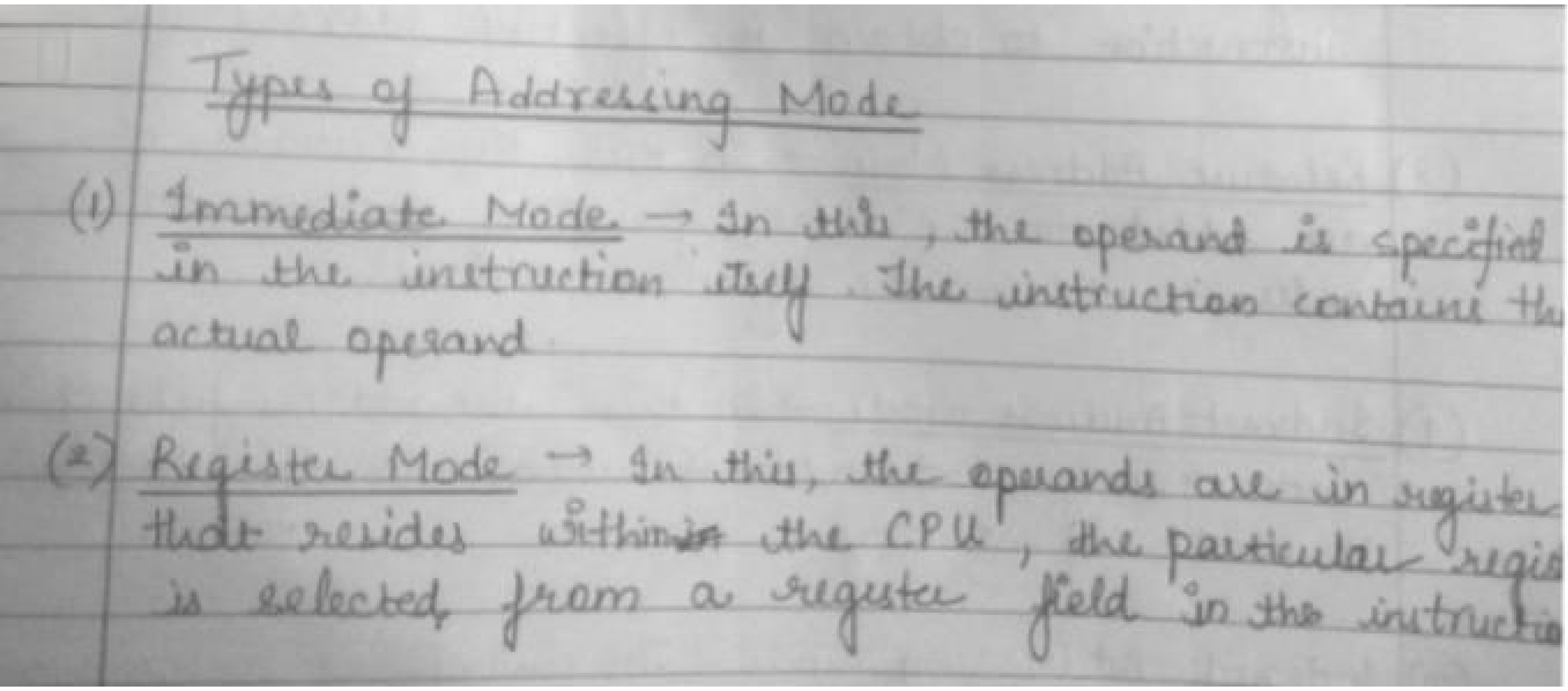
A*B+C*D+E*F)

Symbol	Stack	Expression
(	(	
A	(	A
*	(*	A
B	(*	AB
+	(+	AB*
C	(+	AB*C
*	(+*	AB*C
D	(+	AB*CD*
+	(+	AB*CD*+
E	(+	AB*CD*+E
*	(+*	AB*CD*+E
F	(●+	AB*CD*+EF*●
)		$\boxed{\text{AB*CD*+EF*+}}$

)(A*[B+C*CD+E]/F*(G+H))

(	(	
A	(	A
*	(*	A
[	(*[	A
B	(*[	AB
+	(*[+	AB
C	(*[+	ABC
*	(*[+*	ABC
C	(*[+*	ABCC
D	(*[+*	ABCCD
+	(*[+	ABCCD*+
E	(*[+	ABCCD*+E
]	(*	ABCCD*+E+
/	(*/	ABCCD*+E●+
F	(●/	ABCCD*+E+●*F

*	(●*	ABCCD*+E*F/
(	(*(	ABCCD*+E+*F/
G	(*(	ABCCD*+E+*F/G
+	(*(+	ABCCD*+E+*F/G
H	(*(	ABCCD*+E+*P/GH●
)	(*	ABCCD*+E+*F/GH+
)		ABCCD*+E+*F/GH+*



Types of Addressing Mode

(1) **Immediate Mode** → In this, the operand is specified in the instruction itself. The instruction contains th actual operand.

(2) **Register Mode** → In this, the operands are in register that resides within~~in~~ the CPU, the particular regis is selected from a register field in the instructio

(3) Implied Addressing Mode → In this, the operand a specified implicity in the definition of the Instruc for eg → the instruction compliment accumulato

(4) Register indirect mode → In this, the instruction specifies a register in the CPU whose contents give the address of operands rather than the operand itself.

(5) Direct Address Mode → In this, effective address field of the instruction gives the address where the effectiv address is stored in memory.

(6) Auto-Increment or Auto Decrement → It is similar ~~to~~ to register indirect mode ~~exp~~ except that the register is incremented or decremented after its value is used to access memory

(7) Base Register → In this mode, the contents of a base register is added to the address part of the instruction to obtain the effective address.

(8) Relative Address Mode → In this, the contents of PC are added to the address part of instruction in order to obtain the effective address.

(9) Indirect Address mode → In this, the address field of the instruction gives the address when the effective address is stored in memory.

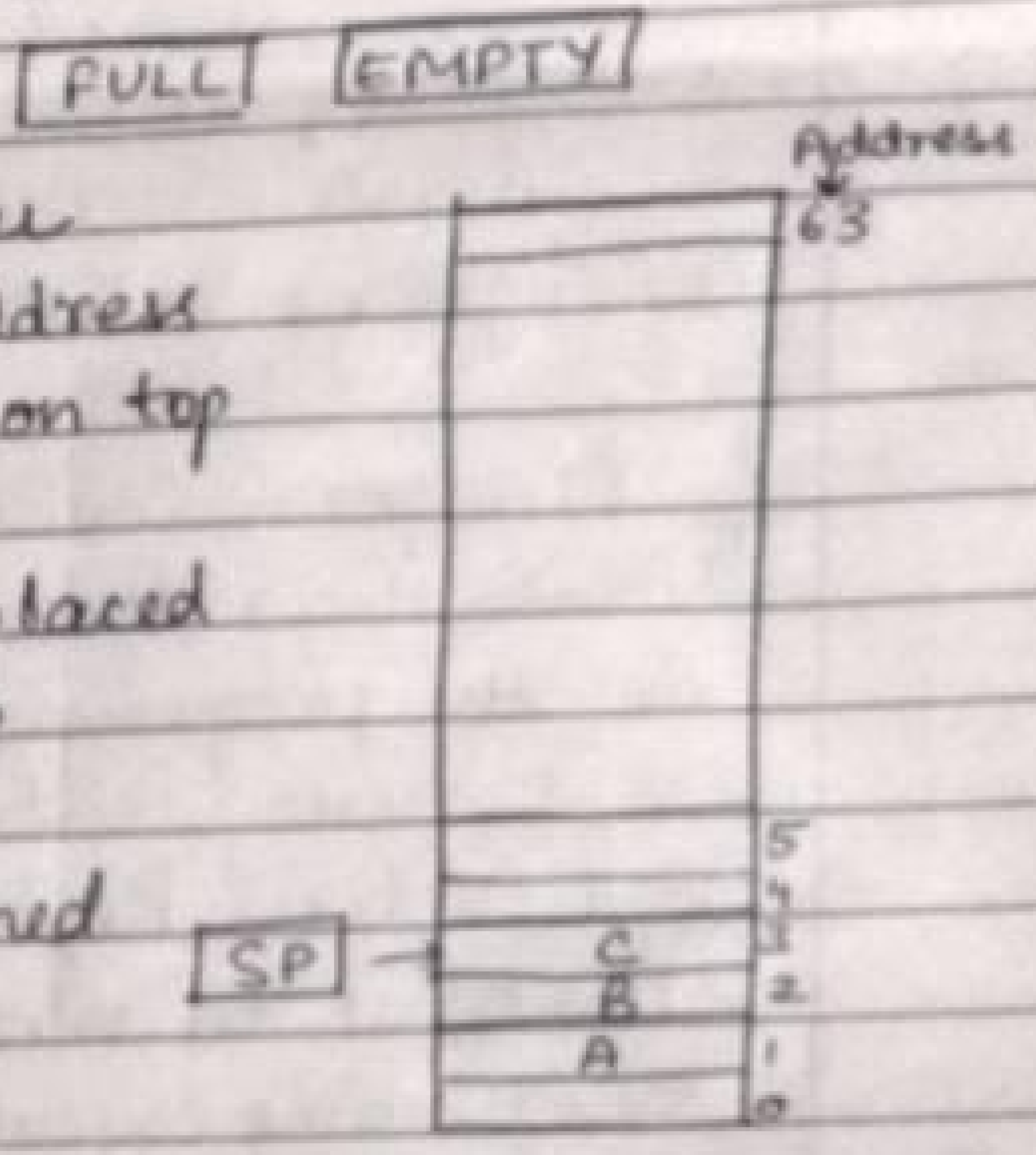
(10) Indexed Addressing mode → The contents of index register is added to the address part of instruction to obtain effective address

Ans

(1) REGISTER STACK

A stack that can be placed in a portion of a large memory or it can be organised as a collection of a finite number of memory words or register

- SP → Stack Pointer
  ↳ It contains binary number whose value is equal to address of word that is currently on top of stack
  ↳ Three items A, B, C are placed on stack where C is on top
  ↳ SP points at 3



64 bit stack

⇒ Operations that can be performed

① Push → Insertion of new element at the top of C.
↳ SP will get incremented and will point to the next address which has the new pushed element.

② Pop → Deletion of the top most element
↳ SP will get decremented and will point to the previous address having the second last element previously

- <u>FULL</u> → It denotes that the stack is full
  ↳ ~~Also~~ i.e one bit register (FULL) is set to one when stack is full.

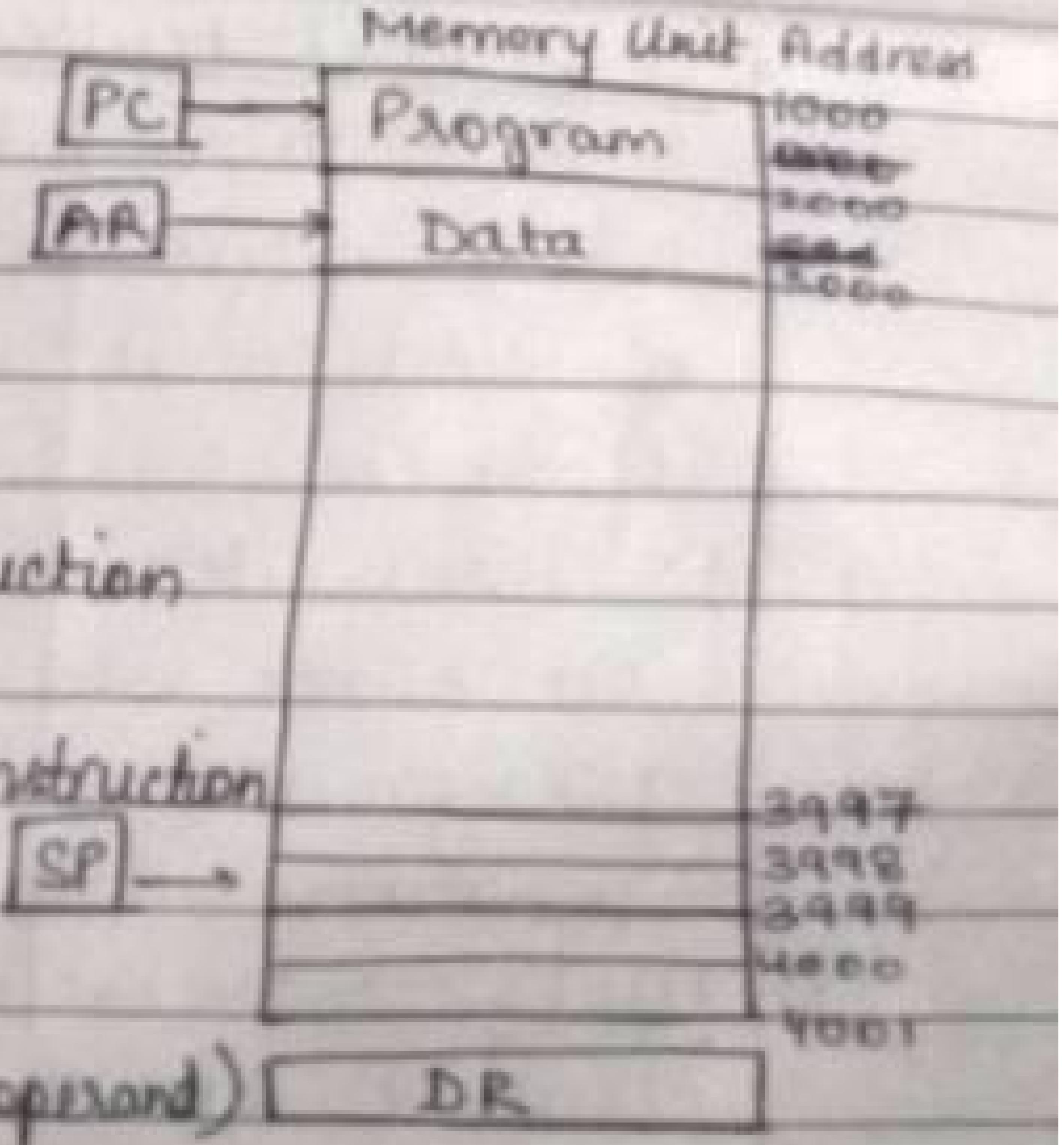
- <u>EMPTY</u> → one bit register EMPTY is set to zero when stack is empty.

(ii) MEMORY STACK

→ A stack which can exist as register stack or can be implemented in RAM Attached to CPU.

→ The stack is implemented by assigning a portion of memory to stack operation and using a processor register as stack pointer.

→ Memory is partitioned into 3 segments
(i) Programe (ii) Data (iii) Stack



→ PC (Program Counter)
↳ Points the address of next instruction in the program
↳ Used during fetch phase to read instruction

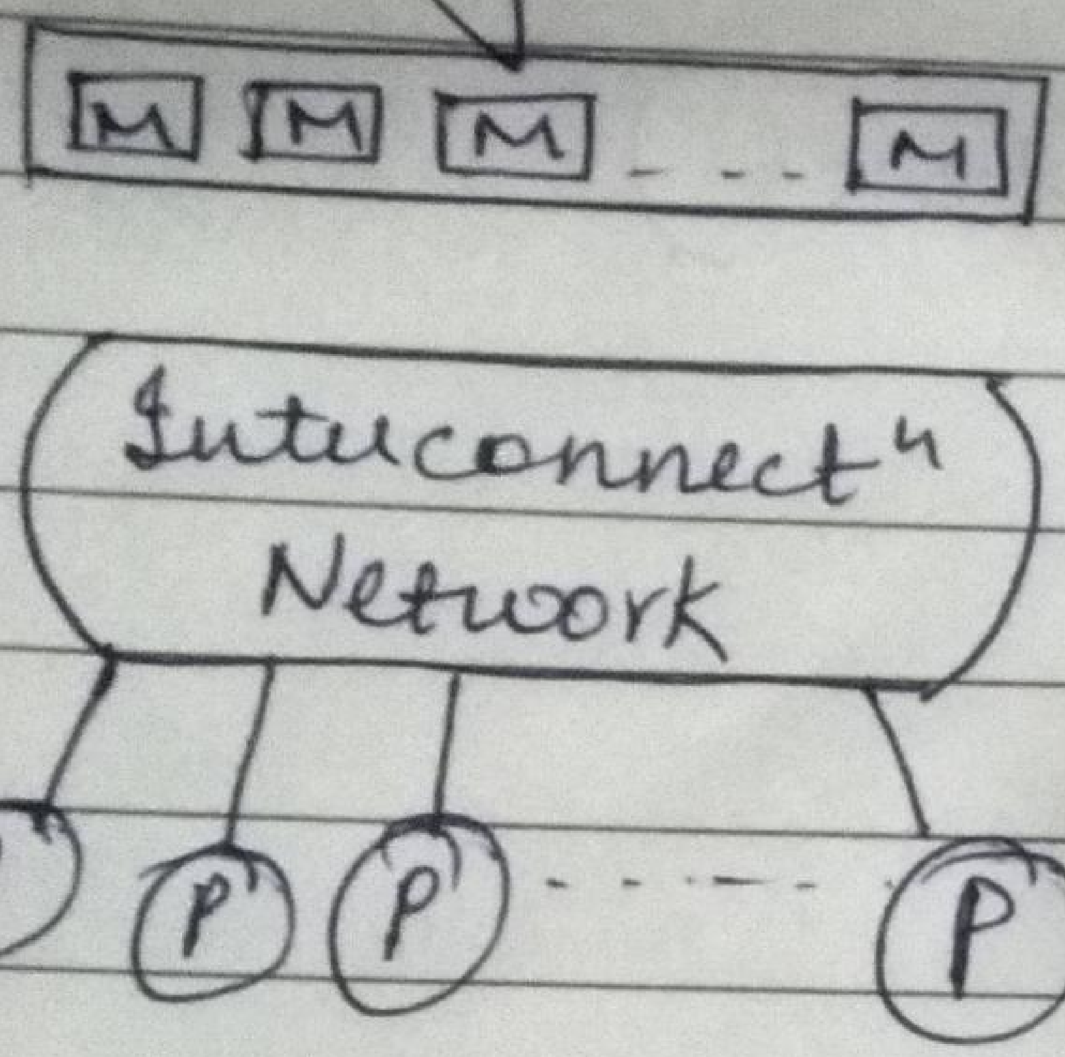
→ AR (Address Register)
↳ Points at an array of data
↳ used during execution phase (reads operand)

→ SP (Stack Pointer)
↳ Points the top of the stack
↳ used to Push and Pop

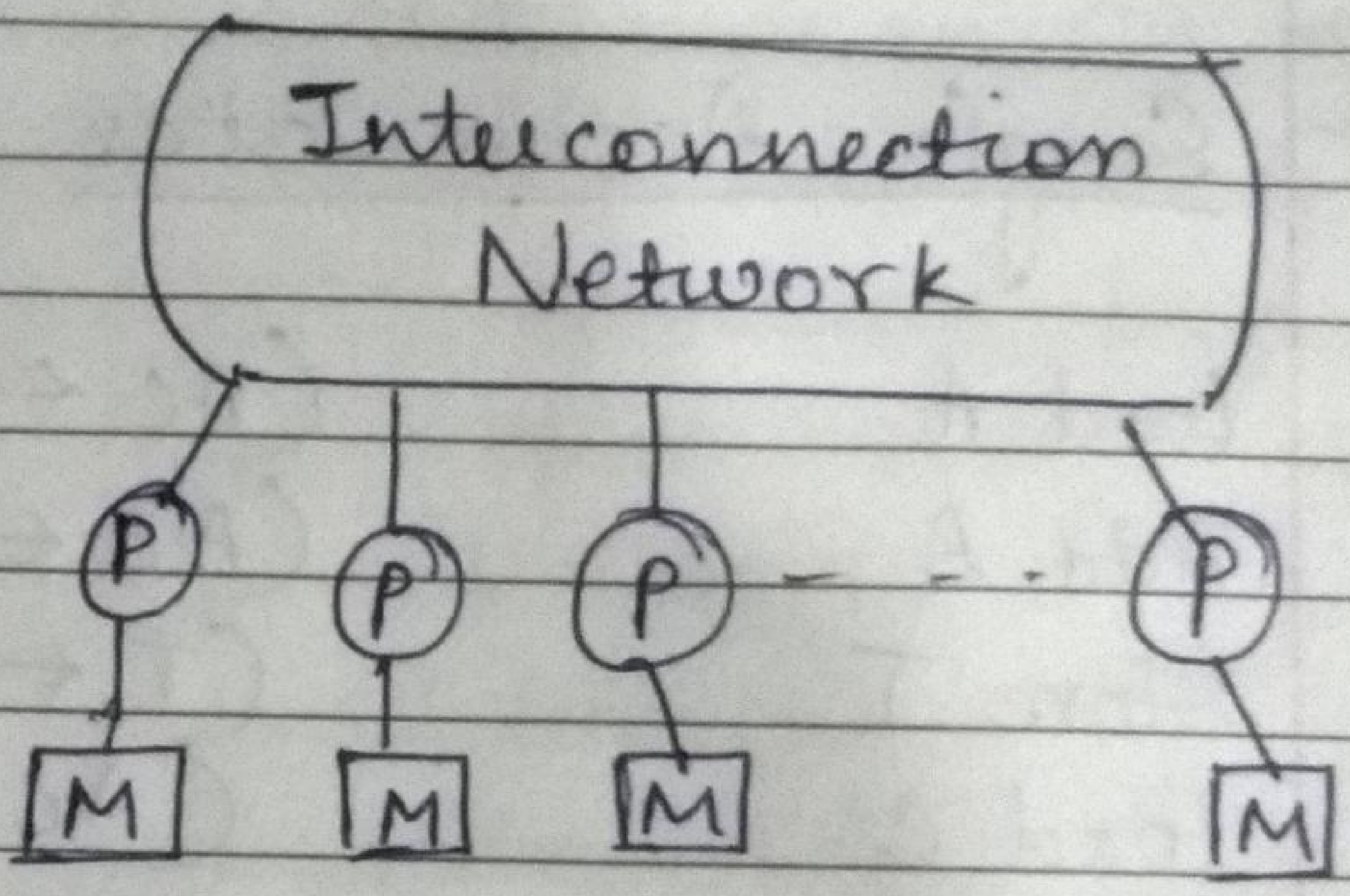
- Advantage → CPU can refer to memory stack ~~whit~~ without having to specify an address, since the address is always available and automatically updated in SP.

Q10 MIMD → Multiple Instruction Multiple Data Streams

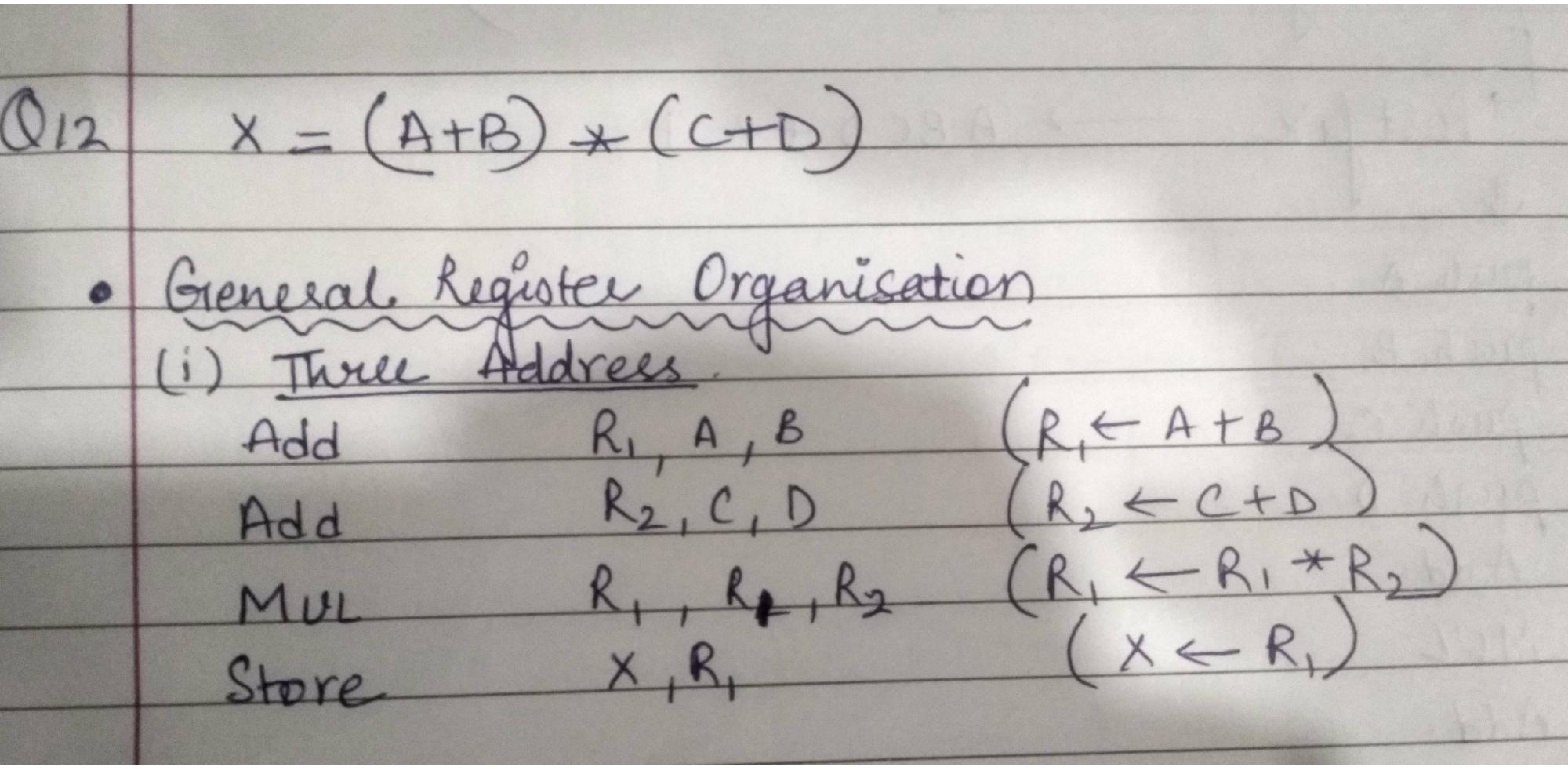
→ It includes parallel architecture made of multiple processor and multiple memory modules linked via some interconnection network. They fall into two broad types including shared memory or message passing.



Shared Memory MIMD Architecture



Message Passing MIMD Architecture



Q12 $X = (A+B) * (C+D)$

- General Register Organisation

(i) Three Address

```
Add     R1, A, B       (R1 ← A + B)
Add     R2, C, D       (R2 ← C + D)
MUL     R1, R1, R2     (R1 ← R1 * R2)
Store   X, R1          (X ← R1)
```

ii) Two Address

Load	R_1, A	$(R_1 \leftarrow A)$
Add	R_1, B	$(R_1 \leftarrow R_1 + B)$
Load	R_2, C	$(R_2 \leftarrow C)$
Add	R_2, D	$(R_2 \leftarrow R_2 + D)$
MUL	R_1, R_2	$(R_1 \leftarrow R_1 * R_2)$
Store	X, R_1	$(X \leftarrow R_1)$

* **Single Accumulator** (1 Address)

Load A	$(AC \leftarrow A)$
Add B	$(AC \leftarrow AC + B)$
Store T	$(T \leftarrow AC)$
Load C	$(AC \leftarrow C)$
Add D	$(AC \leftarrow AC + D)$
Mul T	$(AC \leftarrow AC * T)$
Store X	$(X \leftarrow AC)$

* **Stack Organisation** (zero Address)

↳ Postfix ⟶ ABCD+*+

⇓

push A

push B

push C

push D

Add

MUL

Add.

Q13 Various Registers in CO

i) Accumulator :- It is most often utilized register, and it is used to store info taken from ~~m~~ memory.

ii) Memory Address Register :- Address location of memory is stored in this register to be accessed later

iii) Memory Data Register : All the info that is supposed to be written or information that is supposed to be read from a certain memory address is stored here

iv) Program Counter : Used to store address ~~to be~~ of the next instruction which is to be executed.

v) Instruction Register : It holds the information about to be executed.