Introduction

Content :-

1. Types of Computer,
2. Functional Units of digital system and their interconnection
3. Buses
4. Bus Architecture
5. Types of Buses and Bus Arbitration.
6. Register
7. Bus and Memory Transfer.
8. Processor Organization
9. General register organization
10. Stack organization and Addressing Modes.

1. Types of Computers :-

1.1 Fixed Program Computer / Dedicated device / Embedded System –

→ Specific function and couldn't be reprogrammed

→ Calculators, Washing machine : perform specific tasks.

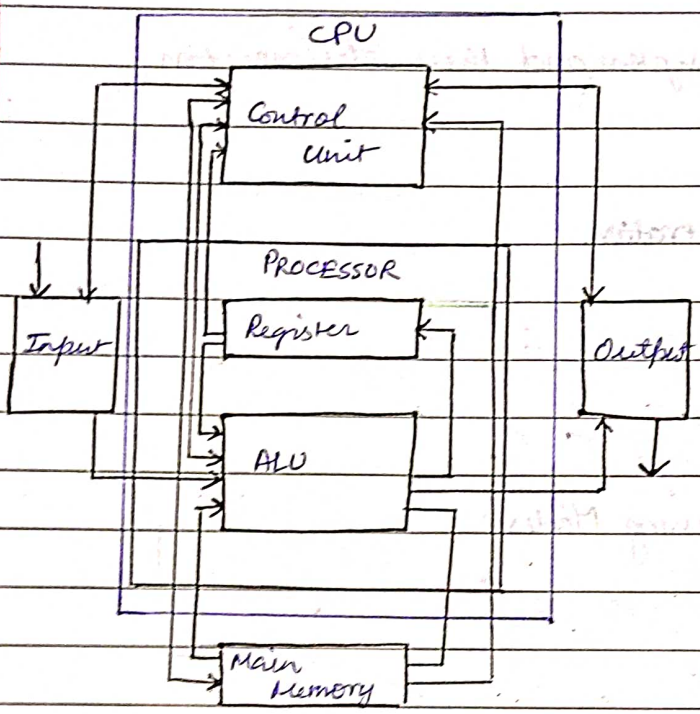
1.2 Stored Program Computer / General Purpose computer / Von Neumann Architecture –

→ These can be programmed to carry out many different tasks.

→ Applications are stored on them.

Stored Program Computers / General Purpose Computer / Von Neumann architecture

Modern Computers work on Stored-program concept by John Von Neumann (stored in computer memory)
Can do anything that a Turing machine can do.



Stored program concept where a programmer writes a program, stores it in memory and then computer executes it, based on requirement program can be changed, so does the functionality.

* Instruction → binary code that controls how a computer performs microoperations.

1. Control Unit :-

→ Generates control signal (to control every other part of CPU)
→ Directs input/output flow
→ Fetches code for instruction.
→ Controls movement of data.

2. Register :-

→ Holds instruction, address storage
→ Holds bit sequence

LD → load
CLR → clear
IN → increment

fast memory, sequence of flip flop

3. ALU :-

→ Arithmetic logic Unit to perform operations like add, subtract etc.

Timing Circuit :-

→ Sequence counter
→ To order certain operations like fetch, decode, execute.
→ Generates timing signals.

Control Unit :-

→ Generates control signals to select a register.

Flags :-

→ One bit information

Bus :-

→ Used to connect different components together.
→ Performs data transfer using multiplexer.

How general operations are performed :-

memory → register → ALU register → memory

$a = b + c$ → memory

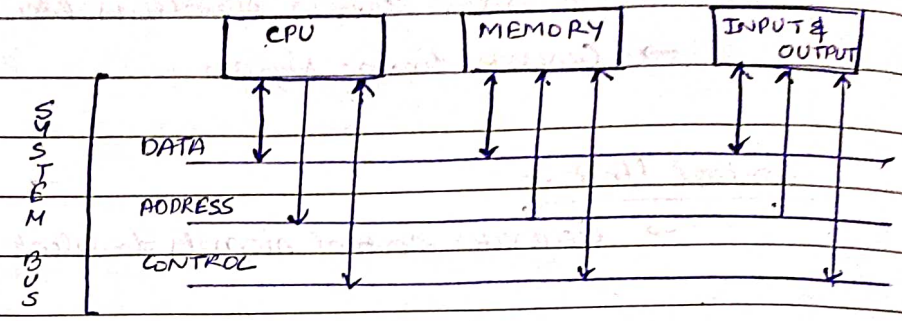
Register: $R_1 \leftarrow m[b]$ // memory address of b to R_1
Register: $R_2 \leftarrow m[c]$ // " " " c to R_2
ALU Register: $R_1 \leftarrow R_1 + R_2$ // perform operations on R_1
memory: $M[a] \leftarrow R_1$ // Place R_1 to memory address a.

P.T.O

Buses, Types of Buses, Arbitration :-

Types of Buses :-

→ Address Bus
→ Control Bus
→ Data Bus



1. Address Bus :-

→ Carry address from CPU to memory // I/O devices
→ identify particular location
→ source and distination, address of data, where to store or retreive data.
→ It is uni-directional
→ Capacity can be increased by adding more address lines
→ An address bus that consists of 16 address lines or wires can convey 2^{16} or 64 KB different addresses
→ Selects I/O device.

2. Control Bus :-

Memory Read :- Data from mem. add location on data bus
→ Memory Write :- Data from data bus to memory
I/O Read :- I/O address to data bus
I/O Write :- Data from data bus to I/O address

→ Transfer the control and timing signals from one component to other.
→ CPU uses control Bus to communicate with devices.
→ Bi-directional
→ Sends functional code on the control line.

3. Data Bus :-

→ Used from transfering data or instruction.
→ Bi-directional
→ Each line carries one bit at a time.
→ A 32-bit bus has 32 data lines and can transmit 32 bits of data at a time.

Bus Arbitration :-

→ Method used to decide which device gets access to the common bus when multiple devices request it simultaneously, ensuring data integrity
→ stability provision.

Conflict :-

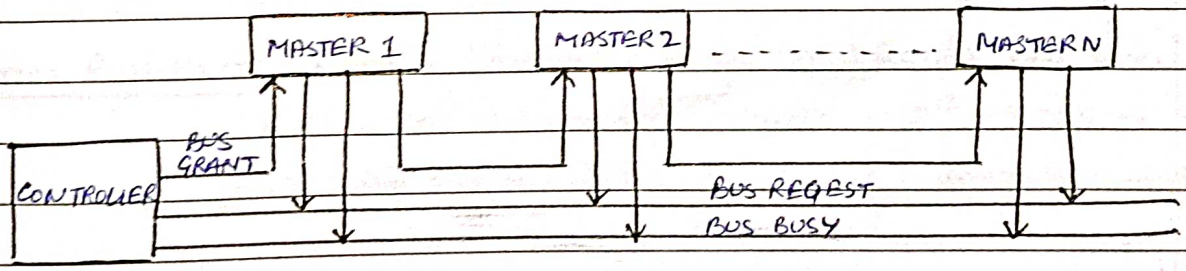
→ Simultaneous access could result in data corruption and system malfunctions, making this mechanism essential for orderly and reliable data transfer

Conflicts can be resolved using few methods :-

→ Daisy Chaining Method
→ Poling Method
→ Fixed Priority or Independent request method

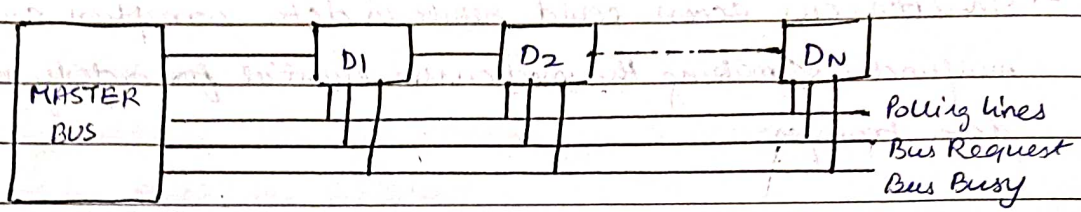
1. Daisy Chaining Method :-

→ simple, cheaper
→ all bus master use common line for making bus request.
→ Propogates to each master serially until the original master who is requesting access.
→ Blocks the propagation of the bus grant signal.
→ Hence requesting module will not receive the grant signal and hence cannot access the bus.



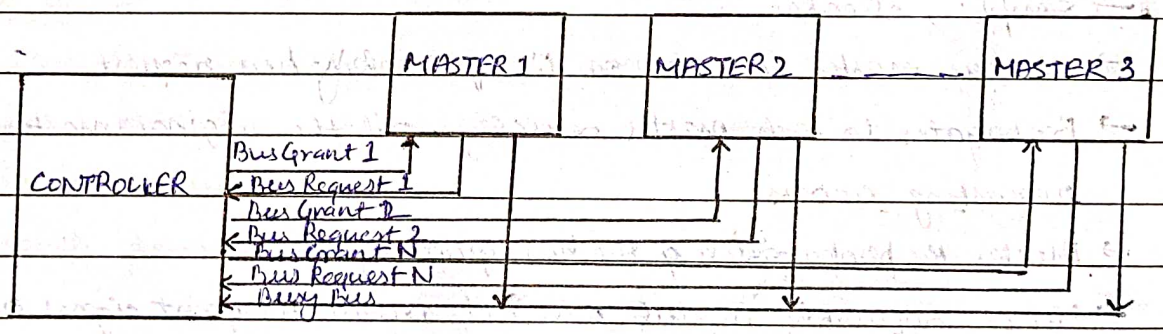
2. Polling Method :-

→ Generates unique address for each masters based on priority
→ The controller cycles through the generates addresses. When a master recognizes its own addresses, it activates a busy signal and gains access to the bus for data transfer.



3- Fixed Priority or Independent Request Method :-

→ Each master has a seperate pair of bus grant lines and each pair has priority assigned to it.
→ The built-in priority decoder within the controller selects the highest priority request and asserts the corresponding bus grant signal.



Register, bus and memory transfer and Processor Organization :-

Processor Organization :-

Registers :-

→ High speed storage areas in CPU
→ The data processed by the CPU are fetched from register

* **Accumulator** → A register, or a memory location used to store the immediate results of arithmetic and logical operations.

Types of Register :-

DR : 16 bits → Data Register : Holds memory operand
AC : 16 bits → Accumulator : Processor register
AR : 12 bits → Address Register : Holds address for memory
IR : 16 bits → Instruction Register : Holds instruction code
PC : 12 bits → Program Counter : Holds address of instruction
TR : 16 bits → Temporary Register : Holds temporary data
INPR : 8 bits → Input Register : Holds input character
OUTR : 8 bits → Output Register : Holds output character.

1. **Data Register** → used to store data that is to be transferred to memory or which are fetched from memory.

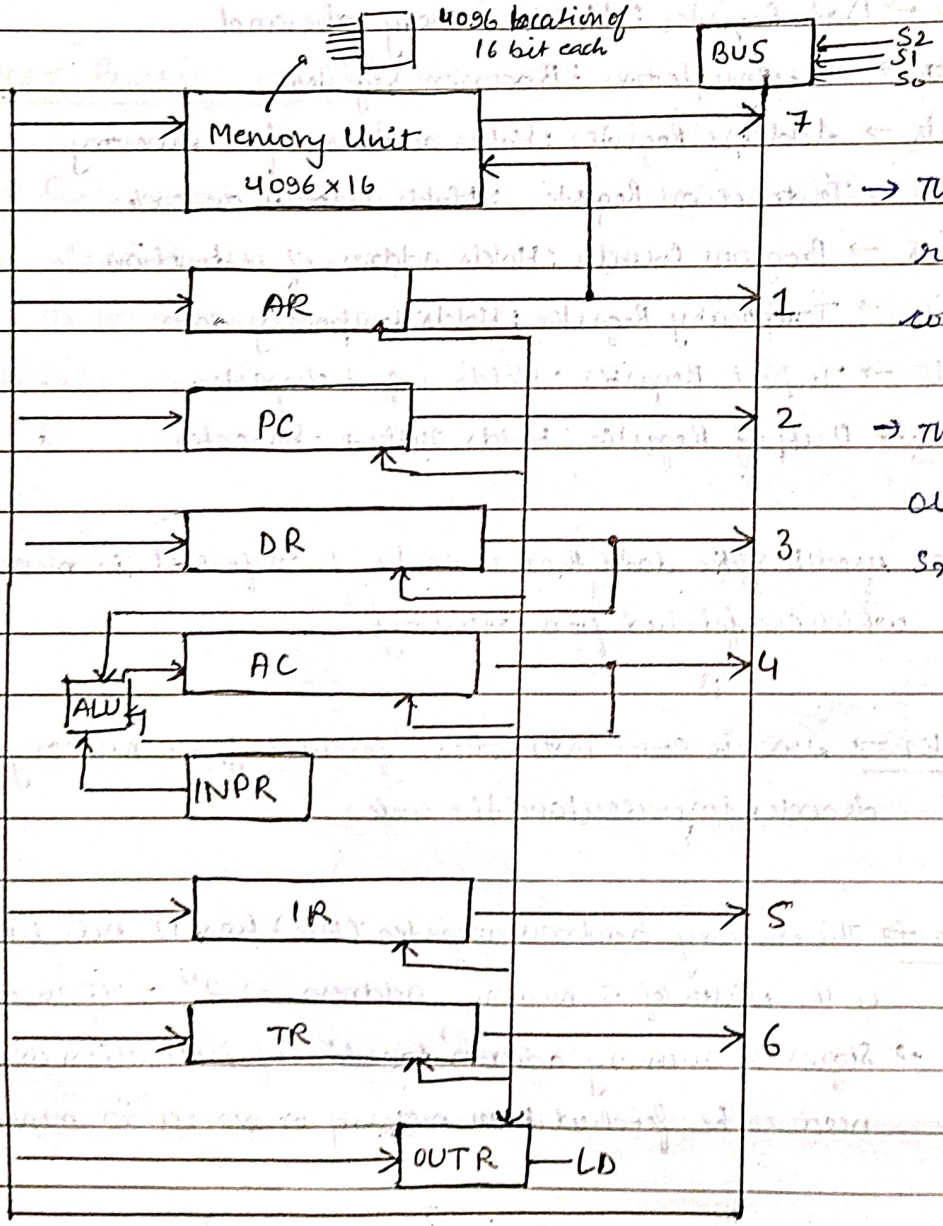
2. **Instruction Register** → used to store instruction fetched from memory and use decoder to understand the code

3. **Address Register** → The memory address register (MAR) has 12 bits since this is the width of a memory address $\Rightarrow 2^{12} = 4096$ or 4KB
→ Stores the memory address location of instruction or data that need to be fetched from memory or stored in memory.

4- Program Counter → Holds address of next instruction to be read from memory after the current instruction is executed.
→ Goes through a counting sequence and causes the computer to read sequential instruction stored in memory.

5- Input Register → Receives 8 bit character from input device
→ passes to AIR, Then accumulator and then to memory.

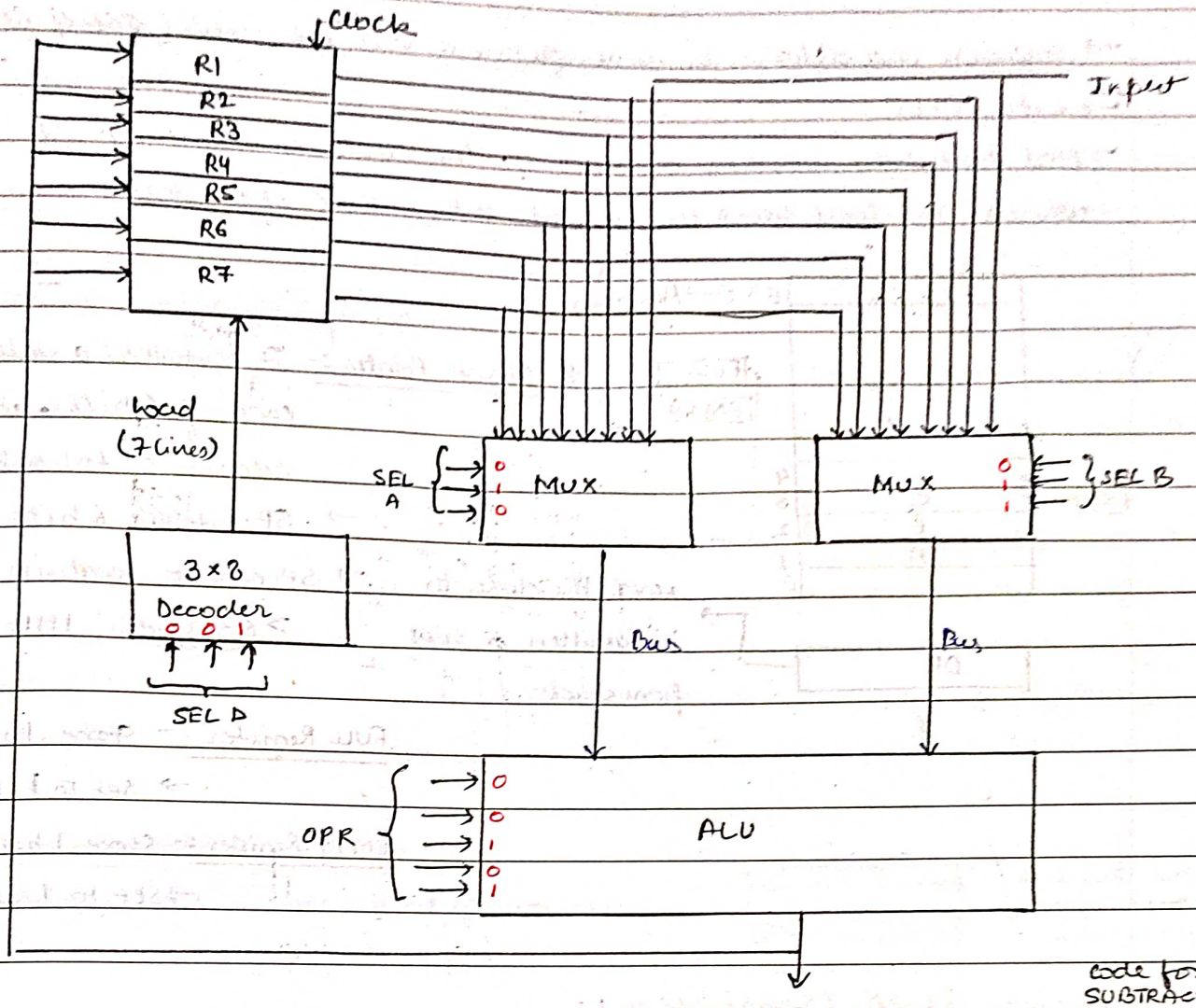
6- Output Register → Holds an 8 bit character for an output device, screen, printer



→ The outputs of seven registers are connected to common bus.

→ The number along the output of DR is 3 so $S_2S_1S_0 = 011$

General Register Organization :-



code for SUBTRACT

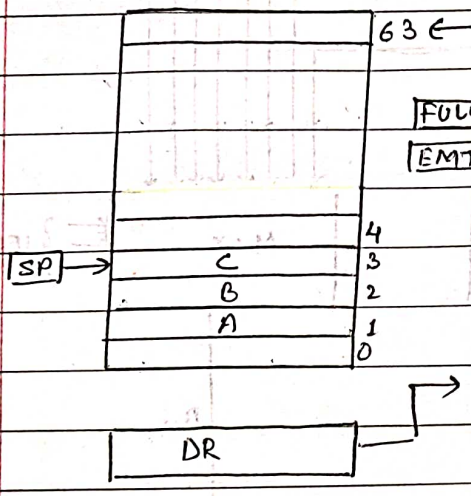
Microoperation	SEL A	SEL B	SEL D	OPR	Control Word
$R_1 \leftarrow R_2 - R_3$	R_2	R_3	R_1	SUB	010 011 001 00101

* Multiplexer → circuit that receives binary information for one of the 2^n input data lines and direct to a single output line (many to one)

* Decoder → circuit that converts n inputs to 2^n outputs (one-to many)

Stack Organization :-

→ Addition and deletion is done from one end i.e. TOS (Top of Stack).
→ LIFO, FILO
→ Most frequently accessible element in the stack is the top most element, whereas the least accessible element is the bottom of the stack.



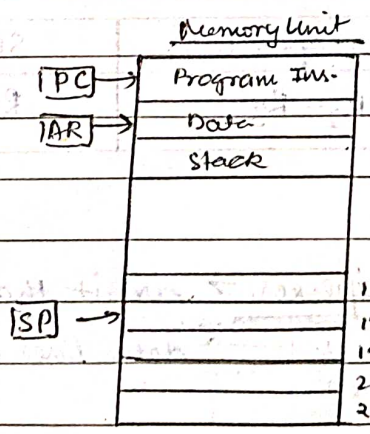
* Stack Pointer :- It contains a value in binary each of 6 bits, which is the address of top of stack
→ SP contains 6 bits
→ SP cannot contain a value >63, i.e. 11111 (0-63) = 64

FULL Register :- Store 1 bit information
→ Set to 1 when full

EMTY Register :- Store 1 bit information
→ Set to 1 when Emtfy

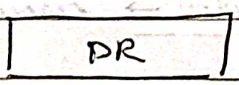
Memory Stack Organization :-

→ LIFO
→ executes using memory and stack pointer.
→ Registers used PC, AR, SP linked to a bus.
→ Stack grows with decreasing addresses
→ Two registers set upper (1000) and lower (2001)



SO/MSO :

→ Stack organization is the structure and management of memory as stack.
→ Memory Stack Org. is a part of computer memory for managing function calls.



Addressing Modes :-

→ way in which the operand of an instruction is specified.
→ Effective Address :- Final address where operand is stored.

Types of Addressing Modes :-

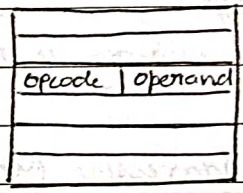
1. Immediate addressing modes
2. Direct AM
3. Indirect AM
4. Implied AM
5. Register AM
6. Register Indirect AM
7. Base Register AM
8. Index AM
9. Relative AM.

* Opcode ⇒ Operation code like ADD, SUB, MULTIPY

* Operand ⇒ variable that contains a specific value.

1. Immediate Addressing Mode :-

→ operand is itself a part of instruction.
→ Fast, no memory reference

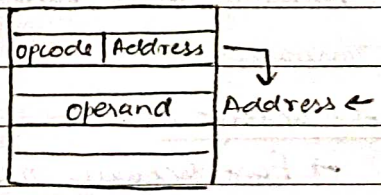


* used when required data is directly moved to required register or memory.

Disadvantages →
→ Cannot be used for variables whose values are unknown at the time of program writing.

2. Direct Mode Addressing (Absolute Address Mode) :-

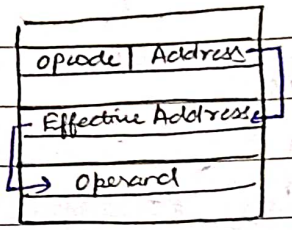
→ instruction contain address of memory location where data is present. (effective Address)
→ Only one memory reference operation is required to access data.



Advantage :-
→ Can be used for variables whose values are unknown.
→ can be used to access global variables.

Disadvantage :-
→ Slow from immediate mode
→ will fail in large calculations, limited variables

3. **Indirect Addressing Mode :-**



→ Store the address of effective address.
→ effective address access the operand.
→ 1st reference to access effective address, 2nd reference to access data.

Advantage :-
→ No limitation on no. of variables.

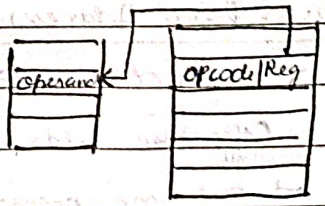
Disadvantage :-
→ Slow as memory is referred more than one time.

4. **Implied Addressing Mode :-**

The IAM itself specify the operands implicitly (without directly applying)

5. **Register Mode Addressing :-**

→ Variables are stored in register of CPU instead of memory. in the instruction will give the register numbers.



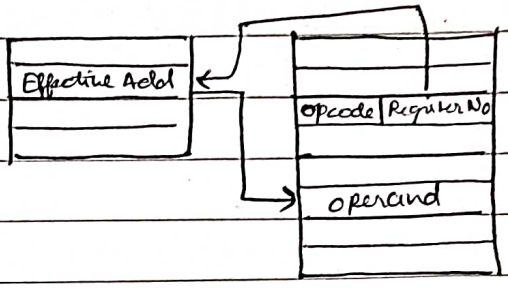
Advantage :-
→ Fast because register access time is less.
→ No. of register is less, so bits required to specify a register is also less.

Disadvantage :-

→ No of register is less so few very few variables can be used.

6. Register Indirect Mode Addressing :-

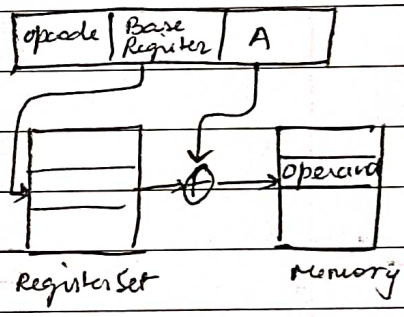
→ Same as indirect addressing mode.



7. Base Register (Offset) Mode Addressing :-

→ Effective addressing of operand is obtained by adding the content of BR (Base register) with AP (Address part of instruction.

Effective Addressing = Content of Base Register
+
Address Part of Instruction

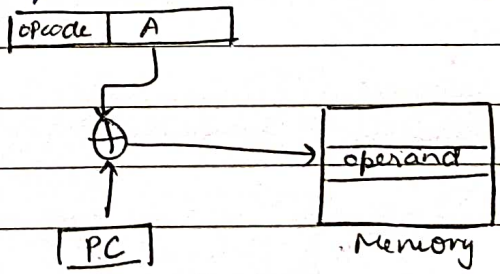


8. Index Mode Addressing :-

→ Same as base address bet instead of Base register we use index register.

9. Relative Addressing Mode :-

→ EA is obtained by adding the content of Program counter with address part of instruction.



Introduction Complete