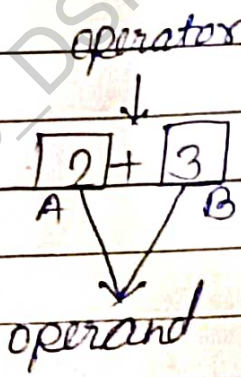
CPU

* Micro-operation :—

↳ The operation executed on values that are stored in registers are called as micro-operation. CPU can perform operations on some value (operand), and these values are stored in register.



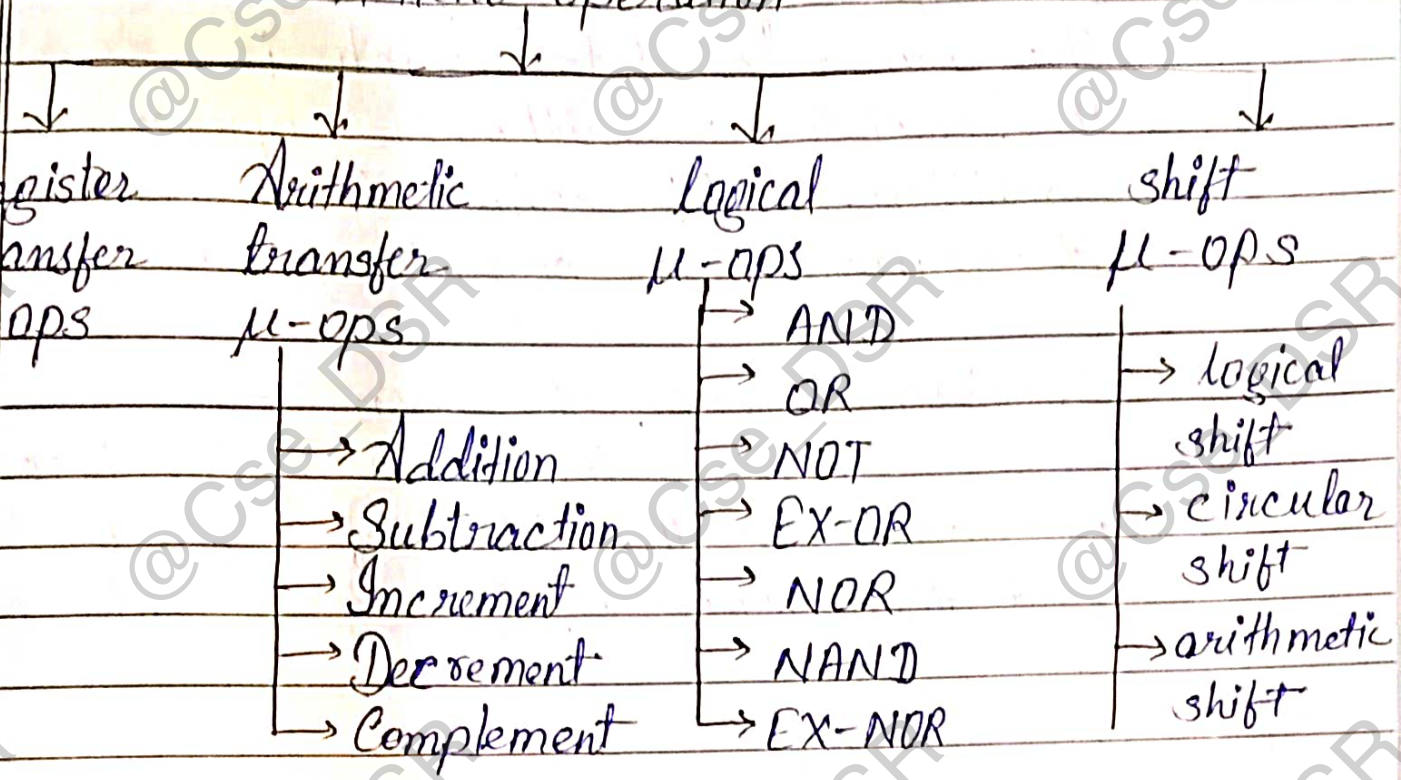
↳ If CPU wants to perform any operation, suppose execution of only one instruction. CPU can not perform one operation in one step, it performs these operations in multiple small-small steps.

↳ Micro-operation is also called as micro-ops (or) μ-ops.

↳ Each micro-operation is executed on one clock-pulse.

↳ In CPU, μ-ops are low-level instruction used in design of complex machine instruction.

* Types of Micro-operation :-



* Register transfer micro-operation :-

→ The operation of information transfer from one register to other register is register transfer μ-ops.

Ex :-

$$R_1 \leftarrow R_2$$

It denotes transfer of data from register R_2 to R_1.

* Arithmetic micro-operation :-

→ We can perform arithmetic operation on the numeric data which is stored in the register. Arithmetic operations are addition, subtraction, increment, decrement, and complement.

→ **Addition micro-operation :-**

In this, the value of register R_1 is added to the value of register R_2 and then sum is transfered into R_3.

$$R_3 \leftarrow R_1 + R_2$$

→ **Subtraction micro-operation :-**

In this micro-operation, the content of register R_2 is subtracted from register R_1 and the result is stored in R_3 register.

$$R_3 \leftarrow R_1 - R_2$$

Another way,

The another way of subtraction is in 2's complement format. In this, 2's of R_2 is added to R_1 and the result will be stored in R_3.

$$R_3 \leftarrow R_1 + (\overline{R_2} + 1)$$

→ Increment micro-operation:—

In this micro-operation, the value inside register R_1 is incremented by 1.

$$R_1 \leftarrow R_1 + 1$$

→ Decrement micro-operation:—

In this micro-operation, the value of register R_1 is decremented by 1.

$$R_1 \leftarrow R_1 - 1$$

→ Complement micro-operation:—

i) 1's complement:—

In this micro-operation, the complement of the value inside register R_1 is taken.

$$R_1 \leftarrow \overline{R_1}$$

ii) 2's complement:—

In this micro-operation, the complement of the value inside the register R_1 is taken and then 1 is added to that value and result will be stored in R_1.

→ This process is also called as "negation".

$$R_1 \leftarrow \overline{R_1} + 1$$

* Logic micro-operation :—

↳ Logic micro-operations are binary micro-operation implemented on the bits stored in register.

↳ These micro-op^n treated each bit independently

i) AND :—

In this μ-op^n, each bit of register R_1 is ANDed with each bit of register R_2 and result will be stored in register R_3.

$$R_3 \leftarrow R_1 \wedge R_2$$

Ex:—

$R_1 = 1001$

$R_2 = 0101$

$R_1 \wedge R_2 = 0001$

ii) OR :—

In this μ-op^n, each bit of register R_1 is ORed with each bit of register R_2 and result will be stored in register R_3.

$$R_3 \leftarrow R_1 \vee R_2$$

$$R_3 \leftarrow R_1 \text{ OR } R_2$$

Ex:—

$R_1 = 1001$

$R_2 = 0101$

$R_1 \vee R_2 = 1101$

iii) NOT :—

In this μ-op^n, the complement of register R_1 will be taken.

$R_1 \leftarrow \overline{R_1}$

$R_1 \leftarrow \sim R_1$

iv) EX-OR :—

In this μ-op^n, each bit of register R_1 is EX-ORed with each bit of register R_2 and result will be stored in R_3.

$R_3 \leftarrow R_1 \oplus R_2$

v) NOR :—

In this μ-op^n, each bit of register R_1 is NORed with each bit of register R_2 and result will be stored in R_3.

vi) NAND:-
In this μ-op^n, each bit of register R_1 is NANDed with each bit of register R_2 and result will be stored in R_3 register.

vii) EX-NOR:-
In this μ-op^n, each bit of register R_1 is EX-NORed with each bit of register R_2 and result will be stored in register R_3.

* Shift micro-operation:-

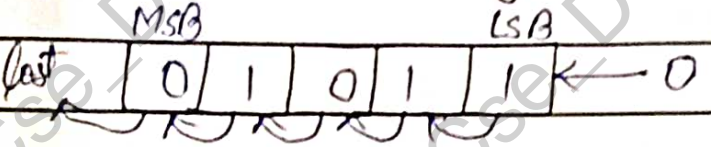
→ The shift μ-op^n allows bits to be moved to the left or right in the register.

→ There are three types of μ-op^n:-
i) Logical shift μ-op^n
ii) Circular shift μ-op^n
iii) Arithmetic shift μ-op^n

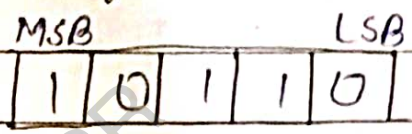
i) Logical shift μ-op^n:-
It shift's the bit either left or right and the empty space will be filled with '0'.

→ It is of two types
i) left shift (shl)
ii) right shift (shr)

i) left shift :—
It shift's each bit to the left direction one-by-one. The empty LSB filled with zero and the MSB will be rejected.



after left shift :—



Representation of left shift :—

$R_1 \leftarrow shl\ R_1$

Register

ii) logical shift right (shr) :—
In this shift μ-op^n, each bits move to right one-by-one and the empty MSB will be filled with zero.

after right shift :—

Representation of right shift :-

R ← shr R
↑ Register

* Arithmetic shift μ-op^n :-

In arithmetic shift μ-op^n, the bits are either shifted left or right one-by-one.

↳ It is of two types :-

i) Arithmetic shift left
ii) Arithmetic shift right

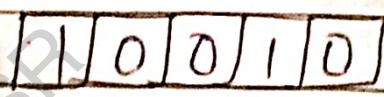
i) Arithmetic shift left :-

In this shift, each bits move to left one-by-one and MSB will be rejected. The empty LSB will be filled with the value of previous LSB.

R ← ashr R

Ex :-

after shifting left :-

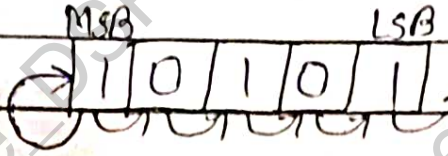


ii) Arithmetic shift right :—

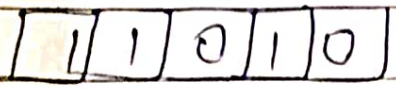
In this shift, each bits move to right one-by-one and LSB will be rejected. The empty MSB bit will be filled with the value of previous MSB.

$R \leftarrow ashr\ R$

Ex :—



after right shift :—



* Circular shift μ-op^n :—

↳ The circular shift circulates the bits in the sequence of register around the both ends without any loss of bit.

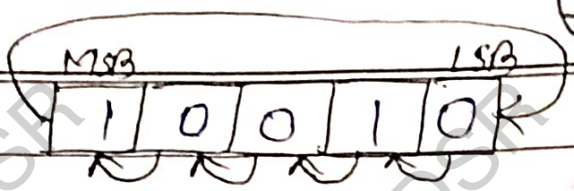
↳ It is of two types :—

i) Circular left shift :—

In this shift, each bits move to the left one-by-one and LSB is filled with MSB.

$R \leftarrow cil\ R$

Ex:-



after cil :-

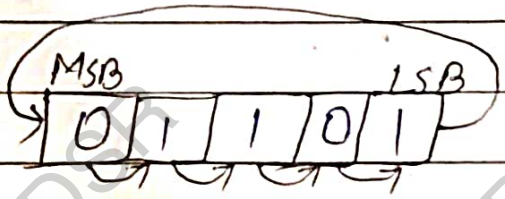
0	0	1	0	1

ii) Circular shift right :-

In this shift, each bits move to the right one-by-one and MSB is filled with LSB.

$R \leftarrow cir\ R$

Ex:-



after cir :-

1	0	1	1	0

* CPU :-

→ CPU is also called as Central Processor, or processor, or μ-processor.

→ It receives instructions from both hardware and software and produces output accordingly.

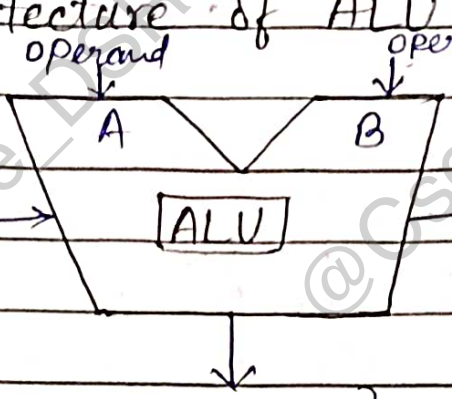
→ It carries all the important functions of computers.

→ It helps input-output devices to communicate with each other.
→ CPU is considered as brain of computer.
→ CPU performs all type of processing operations.
→ It controls all the parts of computer.
→ CPU has three main components :—
i) CU ii) ALU iii) register

i) ALU (Arithmetic logical Unit):—

→ In CPU, an ALU is a combinational digital circuit. That performs arithmetic and logic operations on binary numbers.
→ The control unit supplies the required data from memory from input devices to the ALU. And directs the ALU to perform a specific operation based on instruction.
→ ALU is the calculator portion to the computer.
→ An ALU is a measure component of CPU.
It does all processes related to arithmetic and logic operation, that needs to be done.
→ In some μ-processor, the ALU is divided into AU (Arithmetic Unit) and LU (Logic Unit).
→ ALU is also known as Integer Unit (IU).

* Architecture of ALU



* Opcode :-

↳ When ALU is going to perform the operation, it is described by the operation selection code, that what type of operation an ALU is going to perform.

↳ Opcode is provided by CU.

* Status :-

↳ The result of ALU operation are provided by status signal like carry, overflow, zero etc. are contained by ALU. When ALU completes each operation, the register contain the status output signal.

↳ These signals are stored in register. That led to make them available for further ALU

operation.

* Operand :—

→ The data used as an input is called operand.

* Operations of ALU :—

i) Logic Operation :— This include operations like AND, OR, NOT, NAND, NOR, XOR, XNOR etc.

ii) Arithmetic Operation :— This includes operations like addition, subtraction, multiplication, division etc.

iii) Bit Shifting Operation :— This includes shifting of bit towards left or right.

* Advantages of ALU :—

i) It has capability of performing instructions on a very large set and as high range of accuracy.

ii) Two arithmetic operation in same code like add and multiply or add and subtract is allowed by ALU.

iii) It can combine integer and floating point variables.

iv) In general, it is very fast and it provides result quickly.

v) No memory will be wasted.

vi) They are less expensive and minimize the logic gate requirement.

* Disadvantages of ALU :-

i) With ALU, floating variables have more delay, and the designed controller is not very easy to understand.

ii) In this the concept of pipelining is very complex.

* CU :-

↳ CU is a part of computer CPU, which directs operations of processor.

↳ It controls the operations of all part of computer.

↳ The main objective of CU is to generate the control signal in proper sequence.

* Functions of CU :-

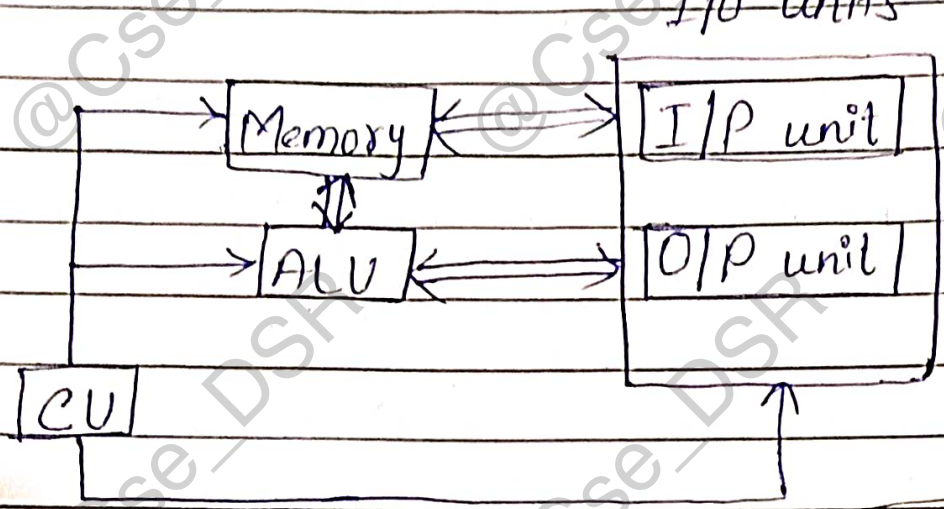
↳ It controls data-flow inside the processor.

↳ It handles multiple tasks such as fetching, decoding, execution and storing result.

↳ It manages and controls the main memory, ALU, registers, input-output units.

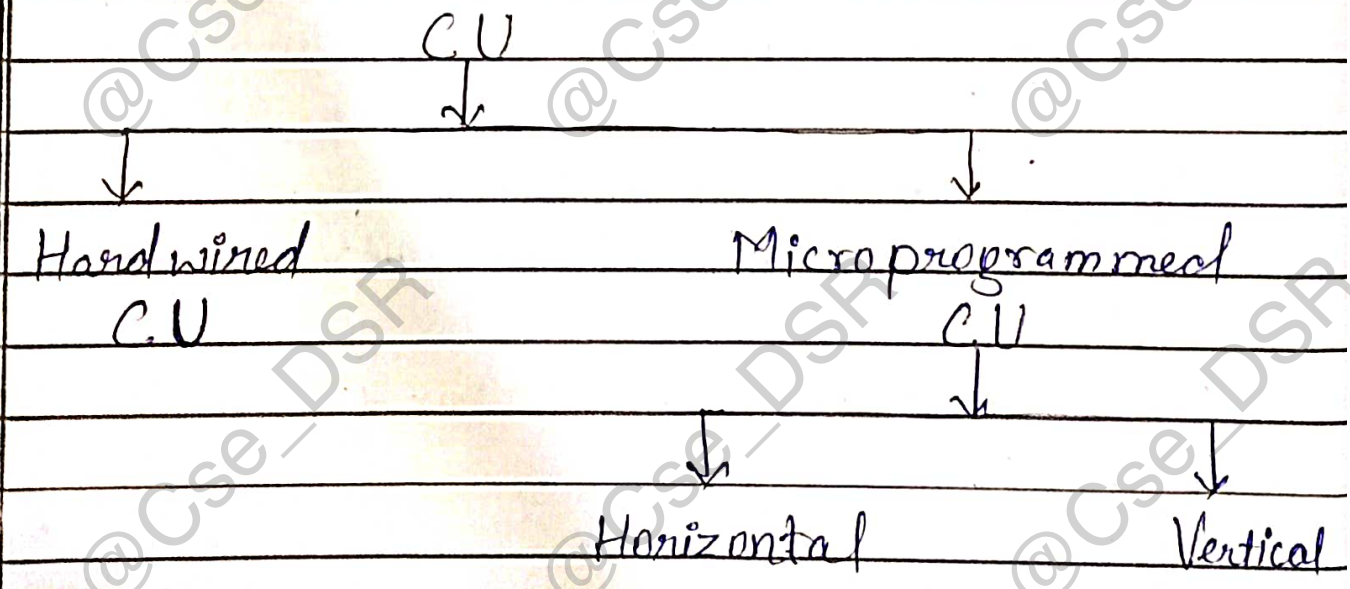
↳ It generates control signals.

Architecture of CU :-



* ⟺ data bus
* ⟶ contrd bus

* Types of CU :-



DATE: __/__/__
PAGE: 61

* Hardwired C.U :-

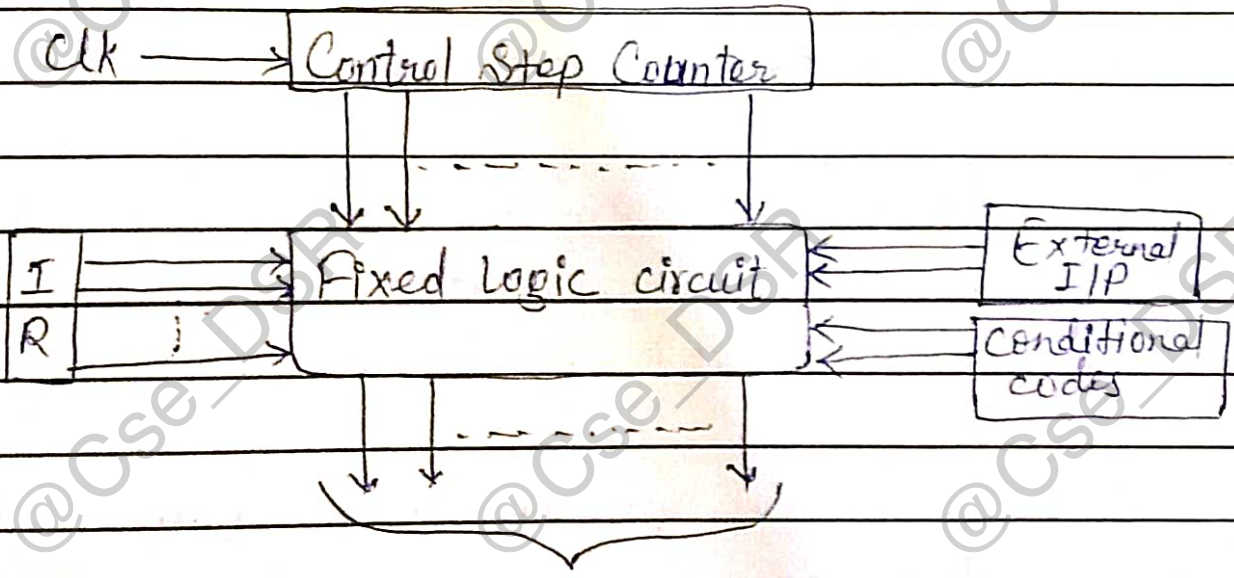
↳ It is implemented with the help of gates, flip-flops, decoder etc in the h/w.

↳ The inputs to C.U are the instruction register, flags, timing signals etc.

↳ This organisation can be very complicated, if we have to make larger control unit.

↳ If the designed has to be modified or changed, all the combinational circuits have to be modified which is very difficult task.

* Architecture of Hardwired C.U :-



i) IR :- An instruction fetched from memory is placed in instruction register (IR).

↳ The fixed logic circuit in the diagram is a combinational circuit made from decoders and encoders.

The decoder decodes the instructions loaded in I.R, and encoders encodes the ~~signal~~ instruction and generate signals.

↳ Control Step Counters stores the address of next coming instruction.

↳ External input represents input control lines connected to fixed logic circuit.

↳ Conditional codes indicates the state of CPU. This includes various flags like carry, overflow etc.

* Advantages of Hardwired C.U :-

↳ Because of the use of hardware combinational circuit to generate signals, hardwired signal is faster.

↳ It depends on number of gates, how much delay can occur in generation of control signals.

↳ It is faster than microprogrammed CU.

* Disadvantages of Hardwired CU:-

↳ The design is complex when we require more signals to be generated.

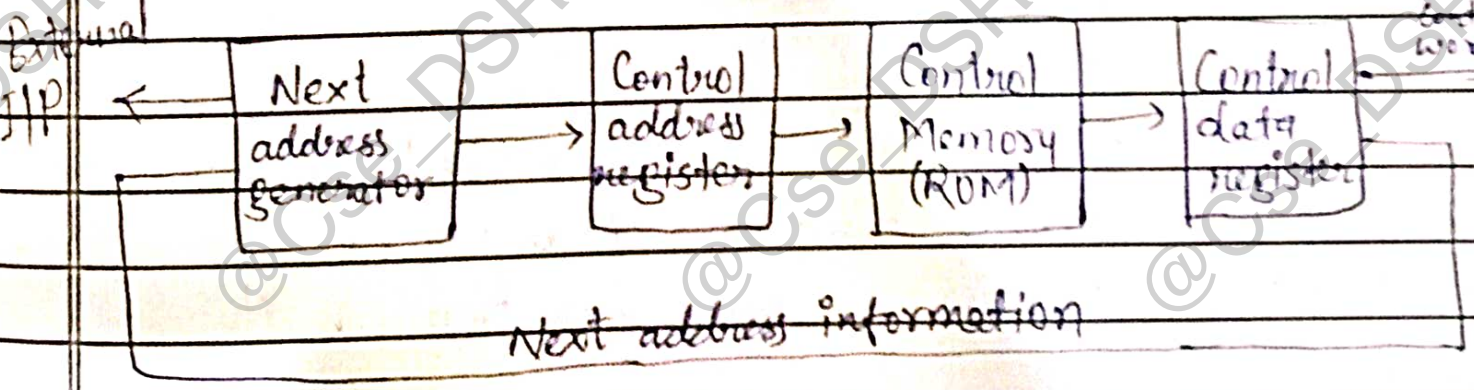
↳ Modification is very difficult and complex.

↳ It is expensive.

* Microprogrammed CU:-

↳ The microprogram CU is implemented by using programming approach.

↳ In this CU, the micro-operations are performed by executing of program consisting of micro-instruction.



↳ The next address generator is sometimes called as microprogram sequence as it determines the address sequence.

↳ The control address register specifies the address of micro-instructions.

↳ The control memory is assumed to be wronged within which all control information are permanently store.

↳ The control data register holds the micro-instruction read from memory, after that control signals will be generated.

* Advantage of Micro-program CU.

↳ It is more systematic design of C.U. It is simpler to debug & change.

↳ It can make the design of C.U much simpler.

↳ It is

↳ It is most flexible.

↳ It is used to control functions implemented in software not on hardware.

↳ It carries complex functions easily.

↳ The only disadvantage is it is slower than H/W CU.

Hardwired	Micro-program
i) Technology is circuit based.	i) Technology is software based.
ii) It is implemented through flip-flop, logic gate, encoder, decoder etc.	ii) Micro instruction generate signal to control the execution of instruction.
iii) Instructions are register based.	iii) Instructions are not register based.
iv) ROM is not used.	iv) ROM is used.
v) It is used in RISC (Reduce Instruction Set Computer)	v) It is used in CISC (Computer Instruction Set Computer)
vi) Faster decoding is perform.	vi) Slower decoding is perform.
vii) It is difficult to modify.	vii) It is easy to modify.
viii) It is expensive.	viii) It is inexpensive.

Components of CPU :-

Bus Structure :-

↳ It is the group of wires, which carries information between computers and b/w components of computer.

↳ There are two types of bus structure.

i) Single Bus
ii) Double Bus

i) Single Bus Structure :-

↳ In single bus structure are common bus is used to communicate b/w devices and processor.

↳ Its performance is low and execution process in slower.

↳ Its cost is low.

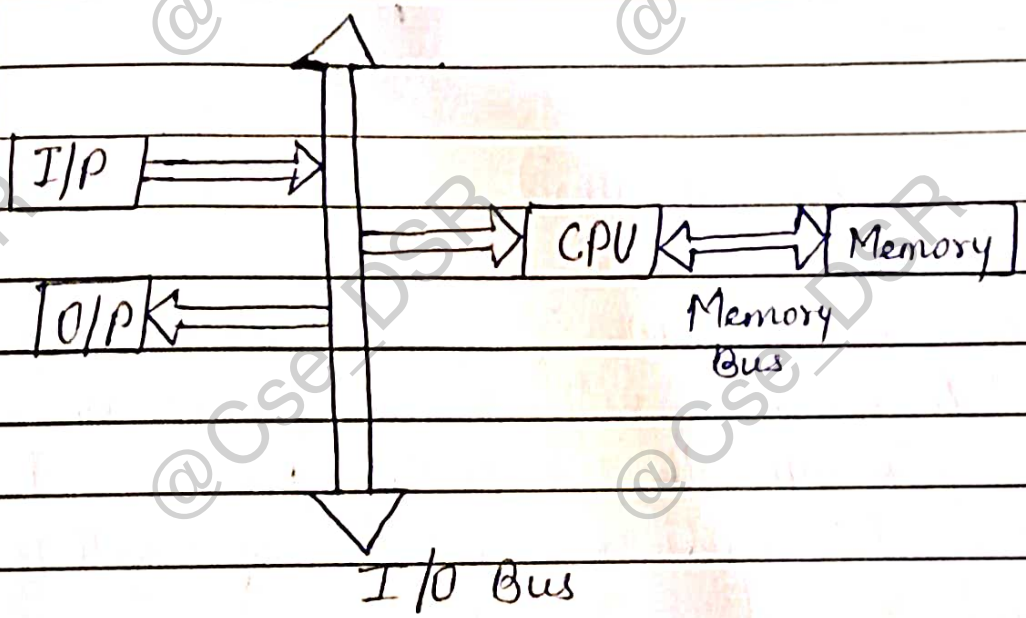
↳ Some bus, in this, use for both data and instruction transfer.

ii) Double Bus Structure :—

↳ In this, one bus is used to fetch instruction, while another bus is used to fetch data require for execution.

↳ Its performance is high and execution process is faster.

↳ It is costlier than single bus structure.

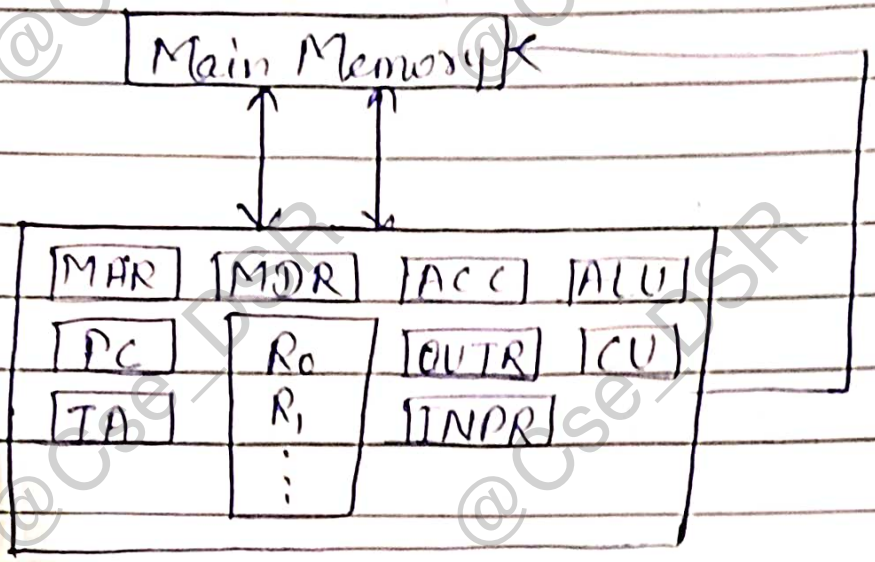


Register :—

↳ Computer registers are high speed storage area.

→ It is group of flip-flops with each flip-flop is capable of store in 1 bit of information.

↳ The main and only purpose of register is fast retrival of data for processing by CPU.



* ACC (Accumulator)

↳ This is the most frequently used register which stores data taken from memory and during the execution process it stores intermediate results of arithmetical and logical operation.

* MAR (Memory Address Register)

↳ It holds the address of the location to accessed from memory.

* MDR (Memory Data Register) :-

→ It contains data to be written or to be read out from the particular location.

* PC (Program Counter) :-

⇒ Program counter is used to keep the track of execution of program. PC point to the address of next instructions to be fetched from main memory where the previous instruction has successfully completed.

* IR (Instruction register) :-

↳ It contains the instruction most recently fetched.

* General Purpose Register / Temporary register :-

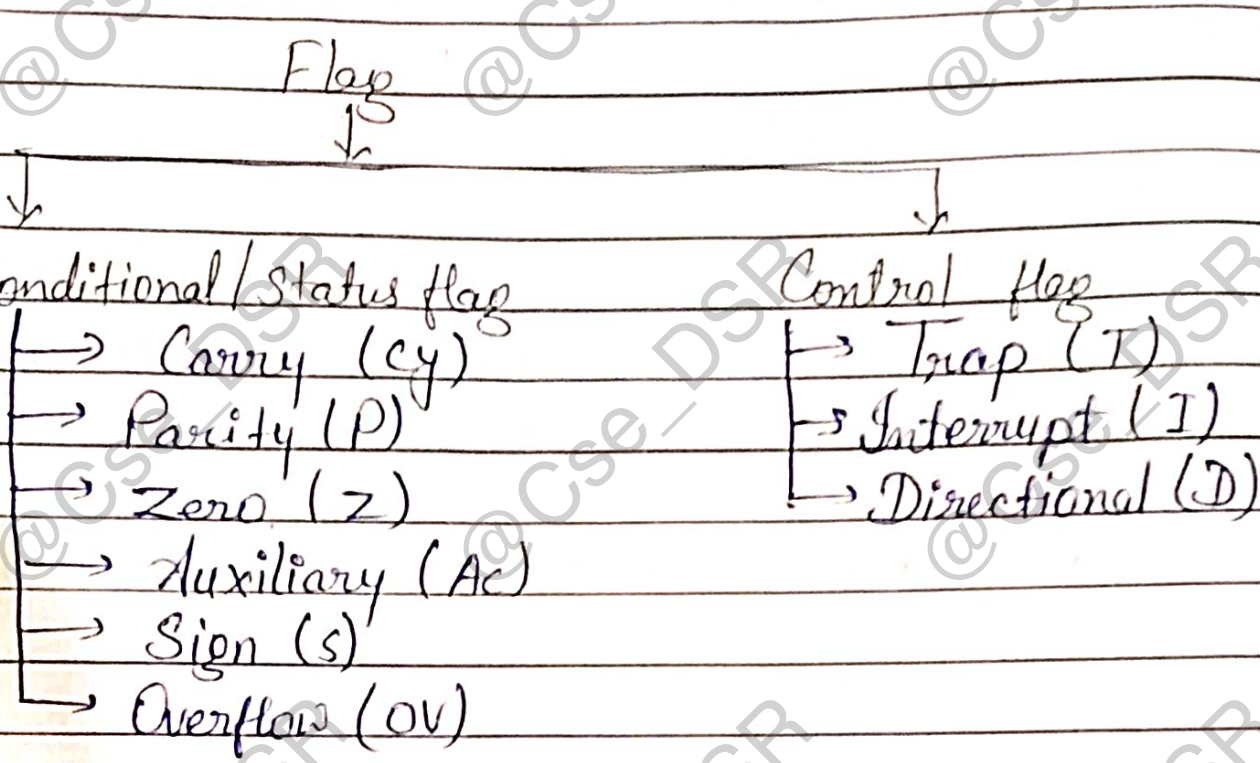
↳ There are numbered as $R_0, R_1, R_2 \dots R_n$ and used to store temporary data during ongoing operation.

* Flags :-

→ A flag is a flip-flop, that means one bit storage space contains either 0 or 1 depending upon

the value of result after the operations.

↳ We can divide flag into two section :-



* Conditional flag :-

↳ These flags are pre-defined with a condition.
↳ Set/Reset is based on the result of ALU.
↳ Set means 1 or true and reset mean 0 or false.
↳ It consists the following flags :-

i) Carry flag :-

↳ When there is an extra bit out of the MSB, it is called carry. If operation

generates carry then this flag is set to 1 otherwise it is set to 0.

ii) Parity flag :—

↳ When result has even number of 1, it will be set to 1 otherwise 0 for odd number of 1.

iii) Zero flag :—

↳ If output of ALU operation is zero then flag 0 is set to 1 otherwise 0 for non-zero values.

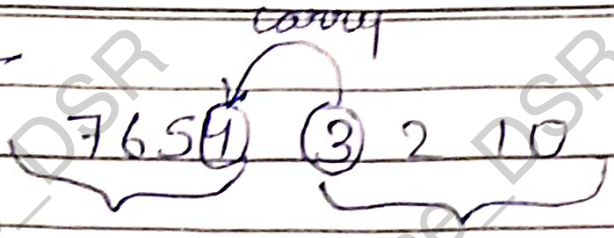
iv) Auxiliary flag :—

→ When arithmetic operation generates carry after the lower nibble and sends it to higher nibble, the AC flag will be 1 otherwise 0 or reset.

→ It is used in BCD (Binary Coded Decimal) arithmetic.

→ This flag is not directly accessible to programmer.

8 bit



v) Sign flag :-

↳ After operation, if MSB is 1 then it indicates the number is negative flag is set to 1, if positive ~~then~~ number then flag is reset.

vi) Overflow flag :-

↳ The overflow flag is set to 1 when the result of signed operation is to large to fit, otherwise it is reset.

Ex:- 1 0 1 1 0 1 0

CY = X
P = 1
Z = 0
AC = X
OV = X
S = 1

DATE: __/__/__
PAGE: _____

* Control Flag :-

↳ The control flag enable or disable certain operation of the processor.

↳ There are three control flag :-

i) Trap flag :-

↳ If flag is set (1) then single step program execution is perform. If it is reset (0), the simple program execution is perform.

ii) Interrupt flag :-

↳ If flag is set (1) then interrupt is enable.
If flag is reset (0) then interrupt is disable.

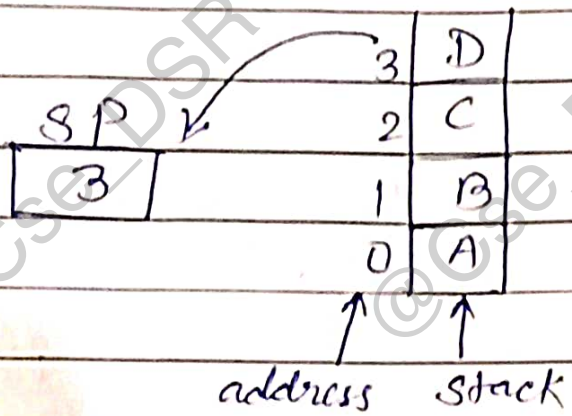
iii) Directional flag :-

↳ If flag is set (1) then access string data from higher memory location to lower memory location (auto decrement). If flag is reset (0) then access string data from lower memory location to higher memory location (auto increment).

DATE : __/__/____
PAGE : _____

* Stack :-

↳ Stack uses LIFO (Last in First out) or FILO (First in Last out) access method. A register is used to store the address of top most element of the stack known as stack pointer



↳ The computer which uses stack data structure for the calculation are stack based CPU org^{n}.

↳ In stack based CPU organisation, ALU operations are performed on stack. Both the operands are always required in the stack. After manipulation, the result is placed in the stack.

↳ The main two operations of stack are PUSH and POP. These operations are performed from the top only.

DATE: __/__/__
PAGE: _____

* PUSH:- This operation inserts operand at the top of stack one by one. And it decreases the size of stack.

* POP:- This operation deletes one operand from the top of stack and it increases the size of stack.

* Advantages of stack:-

i) Efficient computation of complex arithmetic operation.
ii) Execution of instruction is fast because operands are stored in consecutive memory location.

* I/O port:-

↳ Ports are either for input or for output, so it is called I/O port.

↳ Port is the connection point act as an interface between the computer and external devices like printer, modem, keyboard etc.

↳ There are mainly two types of I/O port.
i) internal port
ii) external port

DATE: __/__/__
PAGE: _____

i) Internal port :-

↳ It connects systems motherboard to the internal devices like hard disk, CD drive, internal bluetooth etc.

ii) External port :-

↳ It connects systems motherboard to the external devices like mouse, printer etc.

* Some important types of ports are as follow:

i) Serial port :-

↳ It is used for older computer mouse and external modems.

↳ It transmits data sequentially one bit at a time. It make the slower transfer.
It has 2 versions — i) 9-pin and ii) 25 pin

ii) Parallel port :-

↳ Parallel port can transmit 1 byte (8 bit) data at a time that is parallely.

↳ It is used for scanners and printers. It is 25-pin model.

iii) USB port (Universal Serial Bus) :-

↳ It can connect all kind of external USB devices, such as external hard disk, printer, scanner, mouse, keyboard etc.

DATE: __/__/__
PAGE: _____

↳ In this, data travells 12 mega-bit/sec.

iv) Fire-wire port :-
↳ It connects video, audio, equipments to the computer.
↳ It transfers large amount of data at very fast speed.
↳ Data travells at 400 to 800 mbps.

v) Ethernet port :-
↳ It connects a computer to the network and provides high speed internet.
↳ Data travells at 10 mbps to 1000 mbps depending on network band-width.

* Instruction :-

↳ A statement, that tells a computer to do something is called instruction. Computer instructions are set of machine language instructions that a processor understand and execute.
↳ A computer performs tasks on the basis of instruction provided.

* Types of instruction :-
i) Data transfer instruction
ii) Data manipulation instruction.
iii) Program Control instruction.

DATE: __/__/____
PAGE: _____

i) Data transfer instruction :-

↳ Instructions that transfers data from one location (Register / Memory) to another location without changing the data.

↳ Data transfer operation supported by many processes such as LOAD, STORE, MOVE, IN, OUT, PUSH, POP, XCHG.

- LOAD :- Data transfer from memory to register.
- STORE :- Data transfer from register to memory.
- MOVE :- Data transfer from register to register and sometimes b/w register and memory.
- IN :- Transfer data from input devices to register
- OUT :- Transfer data from register to output device
- PUSH :- Gets data from memory or register to the top of stack.
- POP :- Gets data from top of stack to memory or register.
- XCHG :- Exchanges the data between memory and register.

DATE: __/__/__
PAGE: _____

ii) Data manipulation instruction:-

Here, arithmetic and logic instruction exist.

Arithmetic Ex:- ADD, SUB, MUL, DIV, INC, DEC etc.

Logic Ex:- AND, OR, NOT, XOR, SHL, SHR, ROR, ROL etc.

iii) Program Control Instruction:-

→ These are used to change or modify the flow of program.

→ There are diff. type of control instruction:-

i) Unconditional Branch Instruction:-
It causes an unconditional change of execution sequence to a new location. Ex:- JUMP.

ii) Conditional Branch Instruction:-
It causes conditional change of execution sequence to a new location. Ex:- BE (branch if equal (==)), BL (Branch if less than (<)), BG (Branch if greater than (>)), BGE (Branch if greater than equal to (>=)), BLE (Branch if less than equal to (<=)), BNE (≠).

DATE: __/__/__
PAGE: _____

iii) Sub-routines :-
CALL and RET instructions are use to execute sub-routines. CALL is used from main code where sub routine is invoked. And RET is used to return from sub-routine to main code.

(iv) Halting instruction :-

a) NOP :- It means no operation. It cause no change in the processor state. It is used to create small delay in execution of code.

b) HALT :- It brings processor to halt. Halt state occurs when-ever there is no instruction to be executed by CPU. CPU remains in ideal state until it is restarted by any external actions.

v) Interrupt instruction :-
It is mechanism by which an I/O or instruction can suspend the normal execution of the processor and get itself serviced.

a) RESET :- It reset's the processor.

DATE: __/__/__
PAGE: _____

b) TRAP :- It is non-maskable interrupt and it has highest priority.

c) INTR :- It is maskable interrupt and has lowest priority.

* Instruction format :-

→ The way an instruction is written, is called instruction format.

An instruction is a combination groups called field. The fields are
- Mode field
- Operation field
- Address field

Mode	OP code	operand address of OP

Mode :- The mode field specifies how the operand will be located. It is mandatory field.

Operation field (Opcode) :- It's specifies the operation to be perform. It is also mandatory field.
Ex :- ADD, MOV etc.

DATE: __/__/____
PAGE: _____

Address field:— It contain the location of operand or some time operand itself.

There are four types of instruction format

i) Zero address instruction format :—
↳ This instruction format has no address field.
↳ It has shorter instruction.
↳ Stack based CPU uses this instruction format

Mode	OPcode

Ex:— X = (A+B) * (C+D)

```
PUSH A
PUSH B
ADD
PUSH C
PUSH D
ADD
Mul
```

ii) One address instruction:—
↳ This instruction format uses only one address field.
↳ This is used in accumulator based CPU.

DATE: __/__/__
PAGE: _____

↳ One operand is stored in accumulator & other is in register or memory location.

Mode	OPcode	operand add. of ope

Ex:- LOAD A — ACC ← M[A]
ADD B — ACC ← Acc + M[B]
STORE T — M[T] ← ACC
LOAD C — ACC ← M[C]
ADD D — ACC ← ACC + M[D]
MUL T — ACC ← ACC × M[T]
STORE X — M[x] ← ACC

iii) Two Address Instruction :-

↳ It uses two address field.
↳ It is use on general register CPU.
↳ It is most commonly use instruction format.

Mode	OPcode	destination address	source address

↳ Here, address field can also contain operand

iv) Three address instruction :-

↳ This uses three address field. The address field can be register or memory location the program

DATE : __/__/____
PAGE : _____

which is created is much shorter inside.

Mode	Opcode	destination add	source add	source add

↳ Operands can also be use at the place of address.

Ex :- ADD R_1 ; A, B
ADD R_2, C, D
MUL X, R_1, R_2

* Addressing mode :-

↳ The different way in which the location of an operand is specific in instruction are refered to as addressing mode.

It is the method which is use to identific the location of operand.

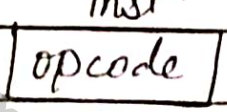
There are different types of addressing.

i) implied / implicit.
ii) immediate
iii) Direct
iv) Indirect
v) Register direct
vi) Register indirect
vii) Indexed

DATE: __/__/__
PAGE: _____

* Types of address mode :-

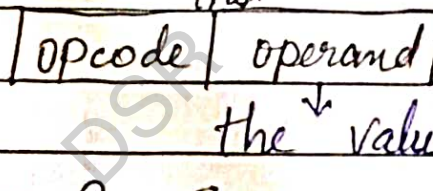
i) Implied / Implicit :- Operands are specified implicitly in the defination of instruction.



Ex:- Zero address instruction (stack operation)

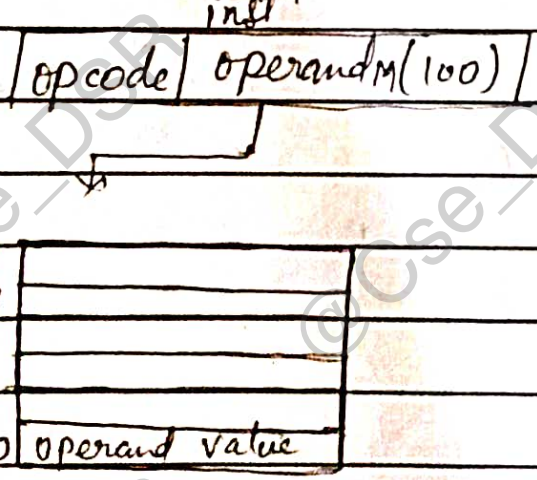
ii) Immediate :- In this, operand value is present in the instruction.

→ designed like one address instruction.



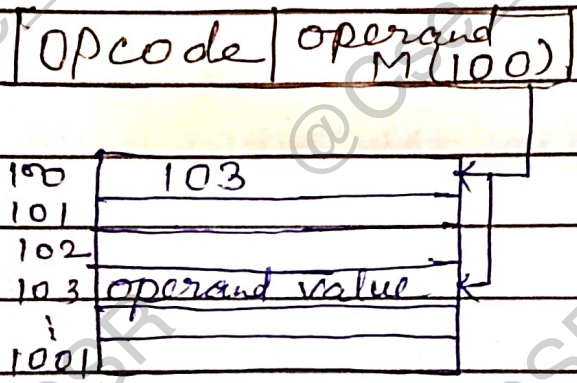
Ex:- MOV R_1, 5

iii) Direct :- In this address field contains the effective address of operand.



Ex :- MOV R, , M[100]

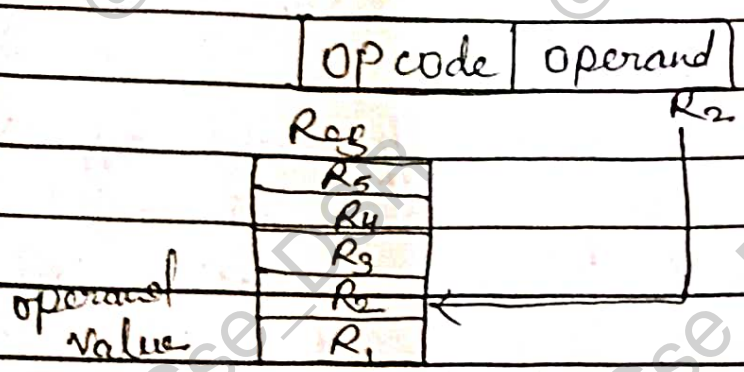
iv) Indirect :- In this address field refers to the address of operand in the memory.



Ex:- MOV R1, M[1000]
MOV R2, M[2000]

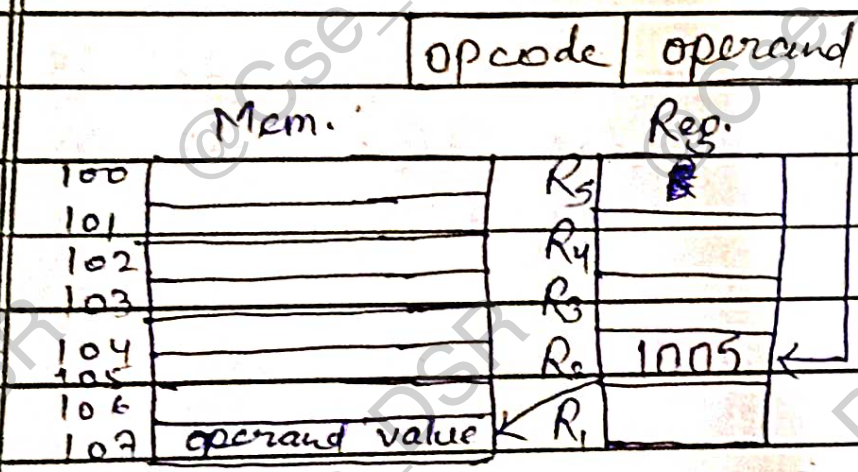
DATE: __/__/__
PAGE: _____

v) Register direct:- It is similar to direct addressing mode. The only difference is that the address field refers to the register rather than memory.



Ex:- MOV R_1, R_2

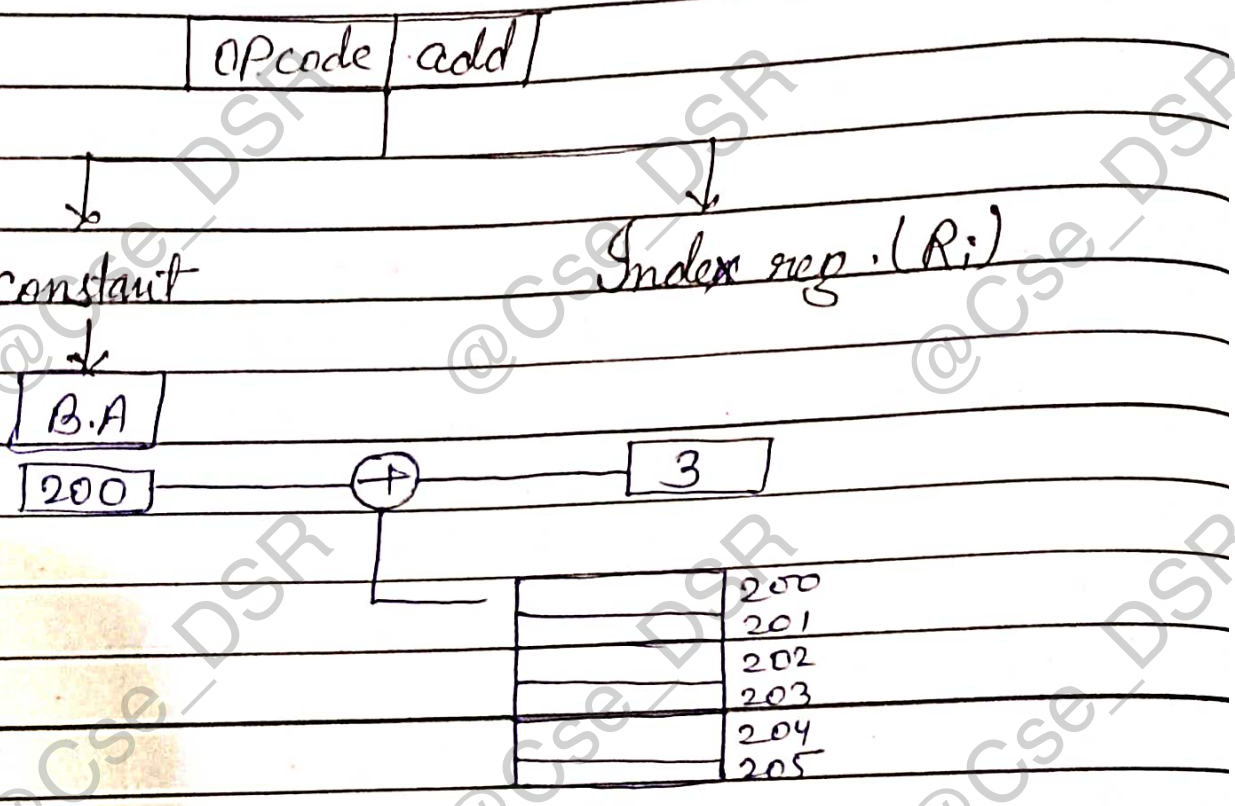
vi) Register Indirect:- In this the address field refers to the register that certain the address of operand in memory.



Ex:- R_1, R_2

vii b) Indexed :— In this the effective address is obtained by adding constant to the content of index register.

{ EA = Add, field value + [Ri] }



CPU organisation

CPU organisation is of three types :—

i) General register org^n
ii) Accumulator org^n
iii) Stack org^n

i) General register org^n :— When we use multiple general purpos

DATE: __/__/__
PAGE: _____

register in the CPU org^n then this org^n is known as general register org^n.

↳ It reduces the size of program on computer.

↳ In this org^n, the CPU uses two address or three address instruction format.

General register org^n is of two types :—

(i) Register-memory refence architecture :—
(CPU with less register)

↳ In this source, '1' is always required in register and source '2' will present in memory. ALU will be performed on both memory and register data.

↳ In this two address instruction format is used.

(ii) Register-register refence architecture :—

↳ In this org^n, all the source should be present in register only. ALU operation are performed only on the register data. After manupulation, the result is also placed in register only.

↳ Two address & three address instruction format both are used in this.

DATE : __/__/__
PAGE : _____

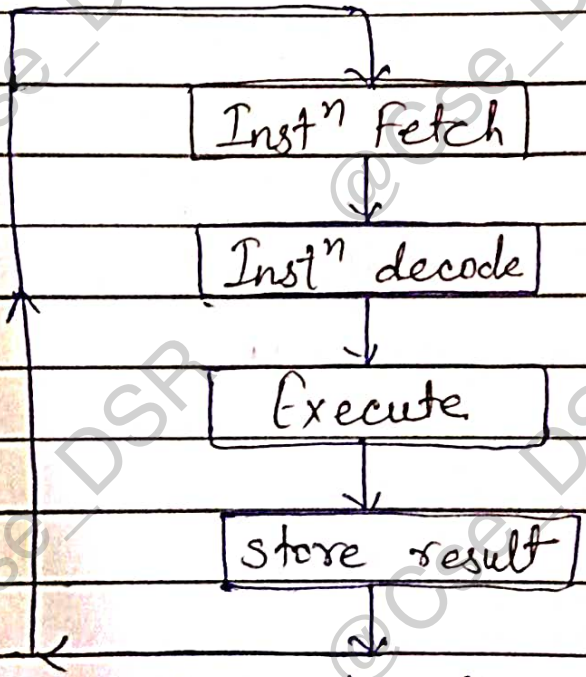
* Instruction cycle :-

↳ The instruction cycle is also known as instruction execution cycle, fetch decode-execute cycle or fetch-execute cycle.

The instruction cycle is cycle that the CPU follows from bootup until the computer has shutdown.

It is having mainly three stage.

(i) Fetch stage
(ii) decode state
(iii) Execute state



In CPU, the instruction cycle is executed sequencially. Each instruction is processed before start of next instruction.

DATE: __/__/____
PAGE: _____

* Inst^{n} fetch / Fetch state :—

↳ The address of next instruction is store in program counter then the inst^{n} is fetch from memory address where actual inst^{n} is store.

↳ At the end of fetch operation, the program counter points to the next instruction that will be executed.

* Decode state :—

↳ In this state, the inst^{n} is decoded by control unit in order to find the sequence of operation.

* Execute state :—

↳ The control unit of CPU passes the decoded information as control signals to the relevent functional unit to perform the action.

* Store state :—

↳ In this state, the result of operation is written to the memory. It is also called write operation.

DATE: __/__/__
PAGE: _____

↳ After processing all the state next inst^n will be taken and this cycle is repeated every time when inst^n execution is needed.

* RISC & CISC Architecture of µ-processor:-

① RISC (Reduced Inst^n Set Computer):-

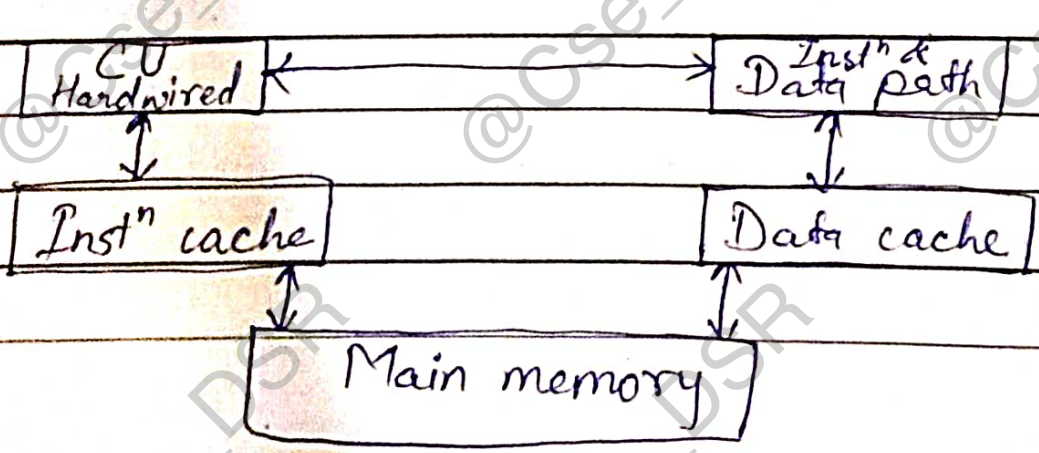
↳ It is a µ-processor architecture which a simple collection of highly customized set of instructions. It is built to minimize the inst^n execution time by optimizing & limiting the no. of instructions.

↳ In this, each inst^n cycle requires only one clock cycle and each clock cycle contains 3 parameters - fetch, decode and execute.

↳ RISC uses simple addressing modes and fixed length instruction.

↳ Pipelining technique is used.

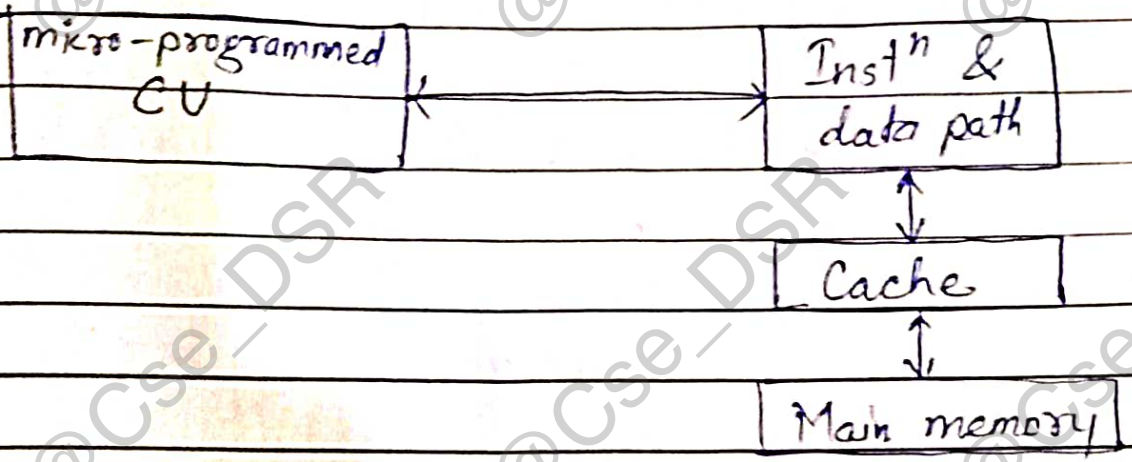
Architecture:-



DATE: __/__/__
PAGE: _____

(ii) CISC (Complex Inst^n Set Computer) :—

→ It has large collection of complex instructions that range from simple to very complex.
It uses inst^n format of different length.
In this micro-programming requires assembly language that is easier to implement.
CISC uses complex addressing modes.
There is no pipelining or sometimes very less pipelining.



* Difference b/w RISC & CISC

RISC	CISC
→ Compact inst^n set.	→ large inst^n set.
→ It is hardwired CU.	→ It is micro-programmed CU.
→ It has simple inst^n decoding.	→ It has complex inst^n decoding.

DATE : __/__/____
PAGE : _____

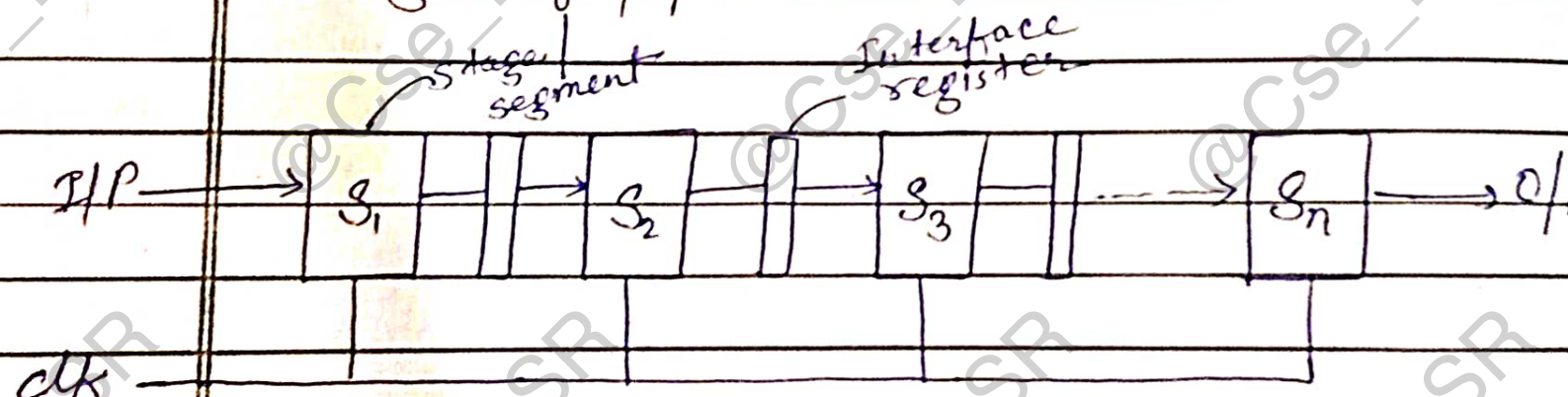
↳ Pipelining is simple.	↳ Pipelining is difficult.
↳ Inst^n format of same length.	↳ Inst^n format of variable length.
↳ Simple addressing modes are used.	↳ Complex addressing modes are used.

* Pipelining :— Pipelining is a process of arrangement of h/w elements of the CPU such that its overall performance is increased.

↳ Simultaneous execution of more than one inst^n takes place in pipelined processor.

↳ In pipelining multiple instructions are overlapped in execution.

* Design of pipeline :—



↳ In a pipelined processor, a pipeline has two ends, the i/p end and o/p end between these ends. There are multiple stages/segments such that o/p of one stage

DATE: __/__/__
PAGE: _____

is connected to i/p of next stage & each stage performs a specific op".

↳ Interface registers are used to hold the intermediate o/p between two stages. These interface registers are also called latch or buffer.

↳ All the stages in the pipelining along with the interface registers are controlled by a common clock.

* Pipeline stages:—

RISC processor has 5 stage inst" pipeline to execute all inst"s.

Stage 1: Inst" Fetch: (IF)

In this stage the CPU reads inst" from the address in the memory, whose value is present in the program counter.

Stage 2: (Inst" decode): (ID)

In this stage, inst" is decoded and the register file is accessed to get values from the registers used in the inst".

Stage 3: (Memory Access): EO

In this stage, memory operands are read and written from/to the memory that

is present in the inst^{n}.

Stage 4: (Inst^{n} Execute): Ex

In this stage, ALU op^{n} are performe

Stage 5: (Write Back): WB

In this stage, computed/ fetched value is written back to the registers present in the inst^{n}s.