Chapter 1

SN	Computer Organization	Computer Architecture
①	It refers to the operational units and their interconnections that realize architectural specifications.	It refers to those attributes of a system that have a direct impact on the logical execution of a program.
②	Attributes include: Control signals, interface between computer and peripherals, memory technologies.	Attributes include: Instruction set, no. of bits used to represent various data types, IO mechanism & memory addressing techniques.
③	It describes how computer does it.	It describes what computer does.
④	It deals with a structural relationship.	It deals with functional behaviour of computer systems.
⑤	It is frequently called microarchitecture.	It is also called ISA.
⑥	While Organization is done on basis of architecture.	While designing a computer system is architecture considered first.
⑦	It involves physical components.	It involves logic.
⑧	It deals with low-level design matters.	It deals with high level design matters.
⑨	It indicates performance.	It indicates hardware.
⑩	It is classified into 3 categories based on number of address field: - Organization of a single Accumulator - Organization of general register - Stack organization.	Different architectural categories found in computer system are: - Von neumann Architecture - Harvard Architecture - Micro Architecture

Functional View of Computer

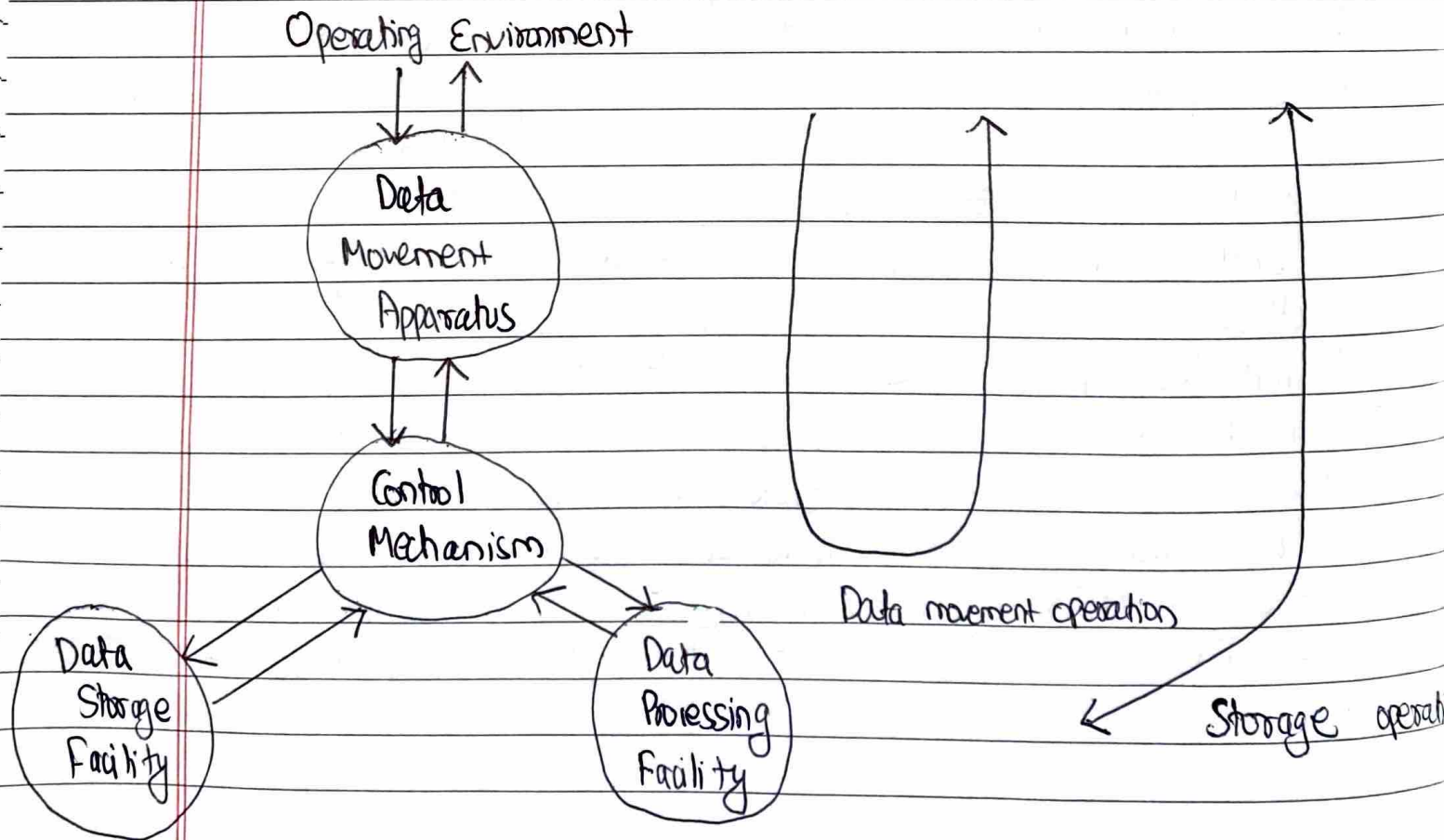
Structure is the way in which components relate to each other.
Function is the operation of individual components as part of the structure.
All computer functions are:

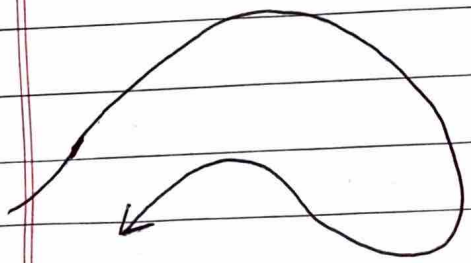
Data processing: Computer must be able to process data which may take a wide variety of forms and the range of processing

Data storage: Computer stores data either temporarily or permanently

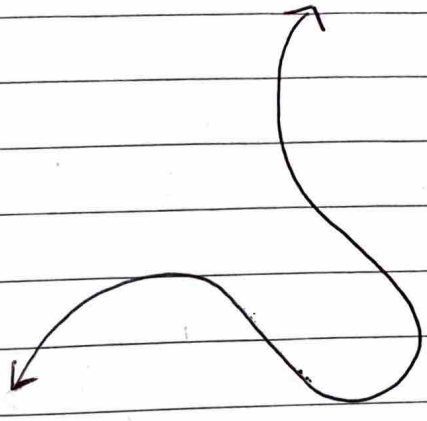
Data movement: Computer must be able to move data between itself and the outside world.

Control: There must be a control of the above three functions.





Processing from/to storage



Processing from storage to IO

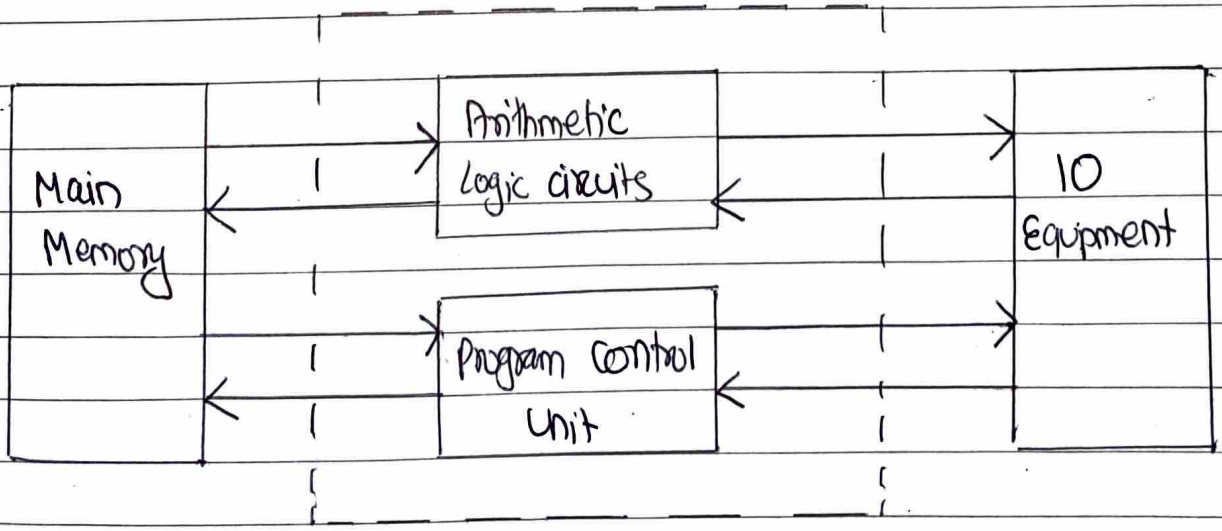
Four main structural components:

- CPU
- Main Memory
- IO
- System Interconnections

CPU structural components:

- CU
- ALU
- Register
- CPU interconnections

Von-Neumann machine / IAS computer



It was designed by the princeton institute for advance studies (IAS).

It consists of:

- Main memory : which stores both data and instruction.
- Arithmetic and logic unit : capable of operating on binary data.
- Program control unit : which interprets the instruction in memory and causes them to be executed.
- IO equipment operated by control unit.

Different registers:

① MBR : Contains a word to be stored in memory or sent to IO unit or recieve.

② MAR : Specifies the address in memory of the word to be written from or read into MBR.

③ IR : Contains 8 bit op-code instruction being executed.

④ IBR : Employed to hold temporarily the right-hand instruction from a word in buf memory.

⑤ PC : Contains address of next instruction-pair to be fetched from memory

⑥ AC and Mutiplier quotient (MQ) : Employed to hold temporarily operand and results of ALU operations.

Operation

IAS operates by repetitively performing an instruction cycle. Instruction cycle contains fetch and execute cycle.

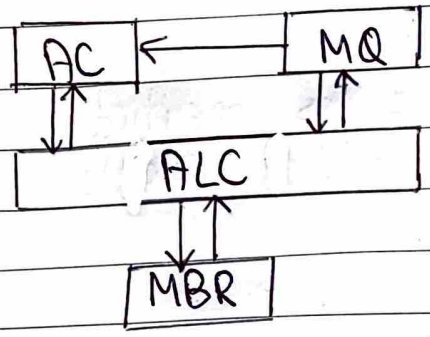
Fetch cycle:
The opcode of next instruction is loaded into IR and address portion is loaded into MAR. This instruction can be taken from IBR, from MBR and then down to the IBR, IR and MAR.

Execute cycle:
Control circuitry interprets the opcode and executes the instructio by sending out appropriate control signals to cause data to be moved or an operation to be moved or perform ALU operations.

IAS has total of 21 instruction that can be grouped as follows

① Data Transfer : between memory and ALU registers or two ALU registers.
② Unconditional branch
③ Conditional branch : branch can be made dependent on a condition.
④ Arithmetic : operations performed by the ALU.
⑤ Address modify : permits address to be computed in ALU and then inserted into instruction stored in memory.

IAS instruction format:
- Memory of IAS consists of 1000 storage locations called words of 40 binary digits.
- Both data and instruction are stored there.
- Number are represented in binary form.
- 1 sign bit, 39 bits value - Number word
- Two 20 bits instruction (8 bits opcode + 12 bits address) - Instruction word



ALU

Addressing Modes

- It specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. It specifies how operands are accessed.

The method of calculating or finding the effective address of operand in the instruction is called addressing mode.

Effective address means the memory address where required operand is located.

The various addressing mode are:

① Implied mode

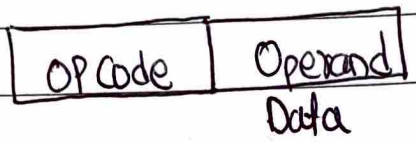
In this mode the operands are specified implicitly in definition of the instruction.

Advantage: No memory reference

Disadvantage: Limited operand

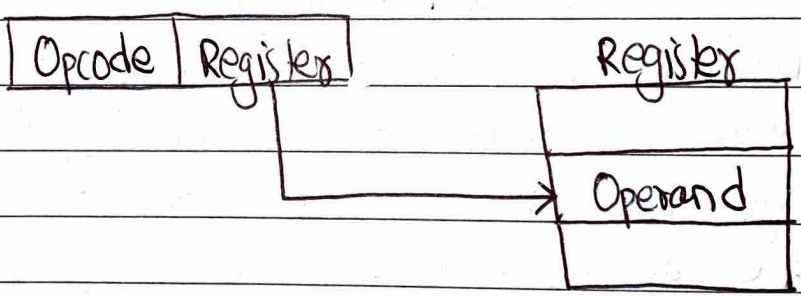
Eg: CMA, DI, EI, CLC

② Immediate Addressing Mode:
In this addressing mode, the operand is specified in the instruction itself ie. there is no any address field to represent operand.



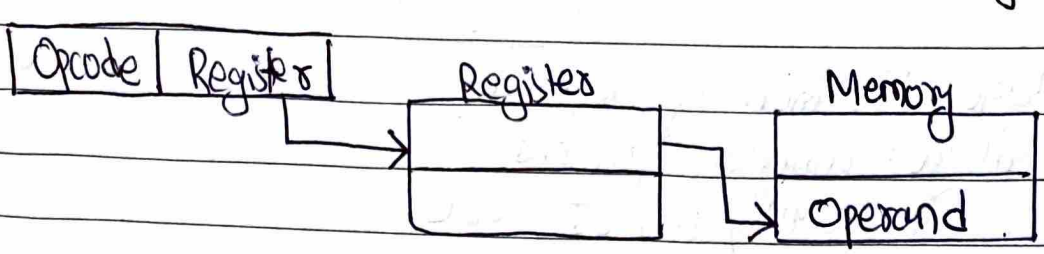
MVI B, 50 H
LXI B, 6000H

③ Register direct Addressing Mode:-
In this mode, operands are in registers that reside within the CPU.
MOV A, B



Advantage: No memory reference
Disadvantage: Limited address space

④ Register Indirect Addressing Mode:
In this mode the instruction specifies a register in the CPU whose contents give the address of operand in memory. In other words, selected register points to the address of operand in memory.



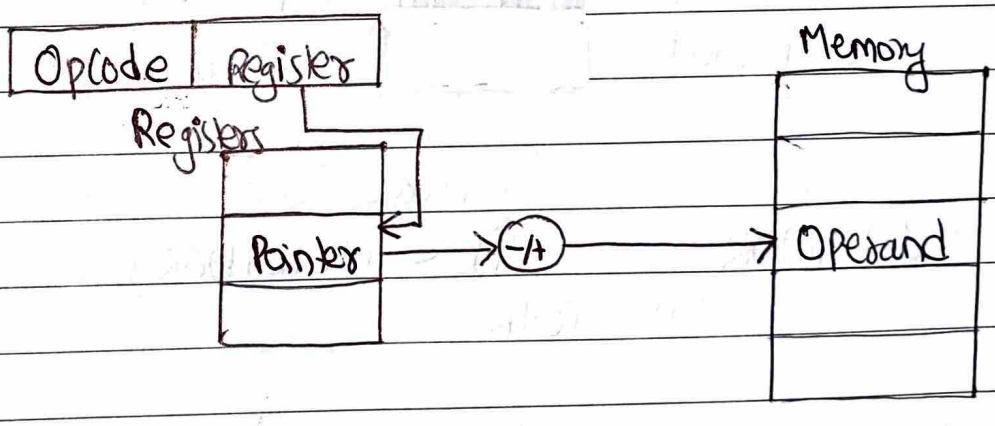
Advantage: Large address space
Disadvantage: extra memory reference

LDAX B
STAX D

⑤ Auto Increment or Auto Decrement Mode

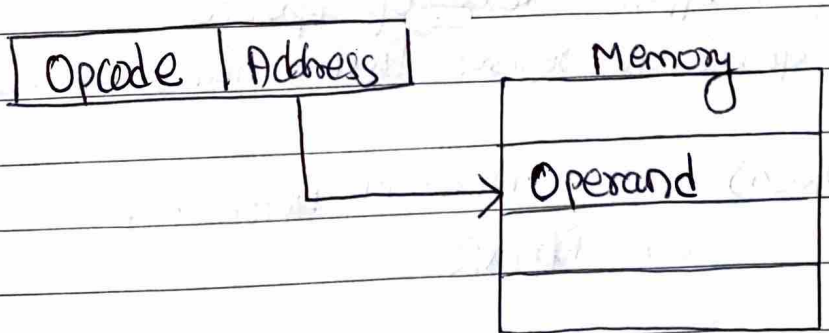
- In auto increment mode, content of CPU register is incremented by 1, which gives the effective address of operand in memory.
- In auto decrement mode, content of CPU register is decremented by which gives the effective address of the operand in memory.

It is similar to Register Indirect mode except that the register is incremented or decremented after/before its value is used to acces memory.



⑥ Direct Addressing Mode

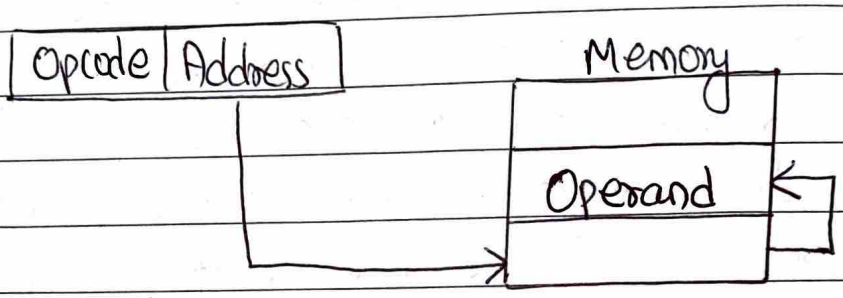
In this mode, the address field of an instruction gives effectiv address of operands.



LDA 4000H.
MOV AX, 5000H

⑦ Indirect Addressing Mode

In this mode, address field of instruction gives the address where the effective address is stored in memory.



LD @ADR AC ← M[M[ADR]]

⑧ Relative Addressing mode

In this mode, content of PC is added to the address part of the instruction in order to obtain effective address.

Opcode	Address

LD $ADR AC ← M[PC+ADR]

EA = PC+A

⑨ Indexed Addressing Mode

In this mode, content of IR is added to the address field of instruction which gives effective address of operand.

IR is a special CPU register that contains an index value.

LD ADR(X) AC ← M[ADR+XR]

EA = A + XR

(10) Base Register Addressing Mode

In this mode, the content of base register is added to the address part of instruction which gives effective address of operand.

LD AR(B) AC ← M[ADR + BR]

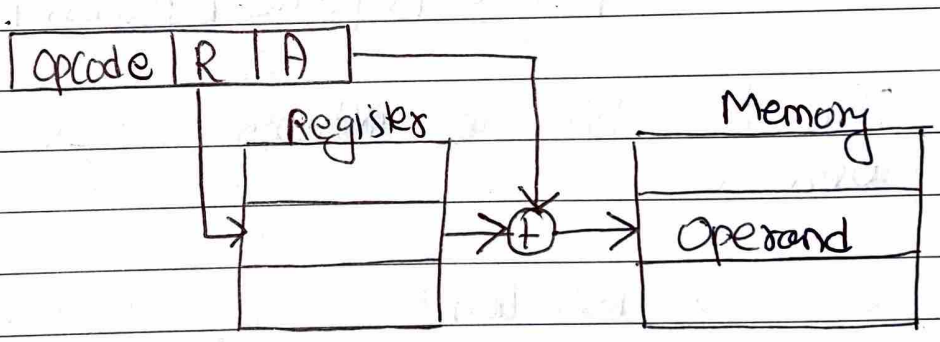
BR is similar to IR.

EA = BR + A

(11) Displacement Addressing Mode

In this mode, it combines the capabilities of direct addresing and register indirect addressing.

Address field of instruction is added to the content of pecific register in CPU.

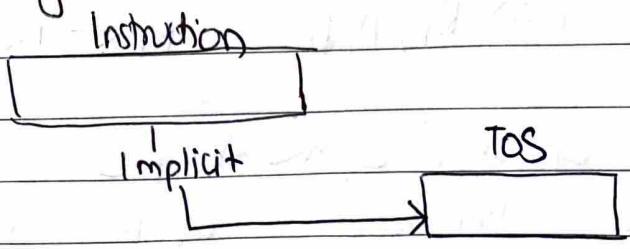


EA = A + (R)

(12) Stack Addressing Mode

Stack is linear array of locations. It is a LIFO data structure.

SP is maintained in register.



EA = TOS

Instruction Address Formats

It defines how instructions are written. An instruction has 3 parts. (bits of instruction are divided into)

Mode bit: It specifies the way the operand or effective address is determined.

Op-code field: It specifies the operation to be performed.

Address field: It designates a memory address or processor register.

The length of instruction varies with the variation of number of addres in an address field.

Mode	Opcode	Address

The four types of instruction on basis of address field they used.

① 3-address Instruction

With this type of instruction, each instruction specifies two operand location and a result location. Each address field can be used to specify either processor register or memory operand. It requires a very complex design to hold 3 address references

ADD R_1, A, B $R_1 \leftarrow M[A] + M[B]$

format:

Op X, Y, Z; $X \leftarrow Y$ Op Z

Advantage: minimizes size of program

Disadvantage: binary coded instruction requires too many bits to specify

② Two-Address Instruction

Each instruction specifies two address in which one must do double duty as both operand and result.
Each address field can be used to specify either register or memory operand.

ADD R_1, B ; $R_1 \leftarrow M[B] + R_1$

format:

Op X, Y ; $X \leftarrow X$ Op Y

Advantage : minimizes size of instruction
Disadvantage : size of program is relatively larger.

③ One-Address Instruction

Each instruction specifies one address as operand. It uses an implied accumulator register for all data manipulation. Accumulator contains one of operand and is used to store the result.

ADD B ; $AC \leftarrow AC + M[B]$

format :

Op X ; $AC \leftarrow AC$ Op X

Advantage: relatively small instruction size
Disadvantage: relatively large program size

④ Zero-Address Instruction

It is used in stack organization computer. It is called "zero" address because of absence of an address field in computational instruction. The PUSH and POP operations however need an address field to specify the operand that communicates with stack.

DIV ; TOS ← TOS DIV (TOS-1)

format:

op ; TOS ← TOS op (TOS-1)

Chapter 2: Register Transfer and Micro operations

Register Transfer Language

Register is the storage device, inside CPU, of data on which microoperations are performed.
The operations executed on data stored in registers are called micro operations.
The transfer of data from one register to another without changing binary bit is called register transfer and corresponding micro-operation is called register transfer micro-operation.

The language, which is used to express the transfer of data among the registers is called RTL. It is symbolic notation used to describe the microoperation transfers among registers.

- Computer Registers are designated by capital letters.
- Information transfer from one register to another is designated in symbolic form by means of a replacement operator.

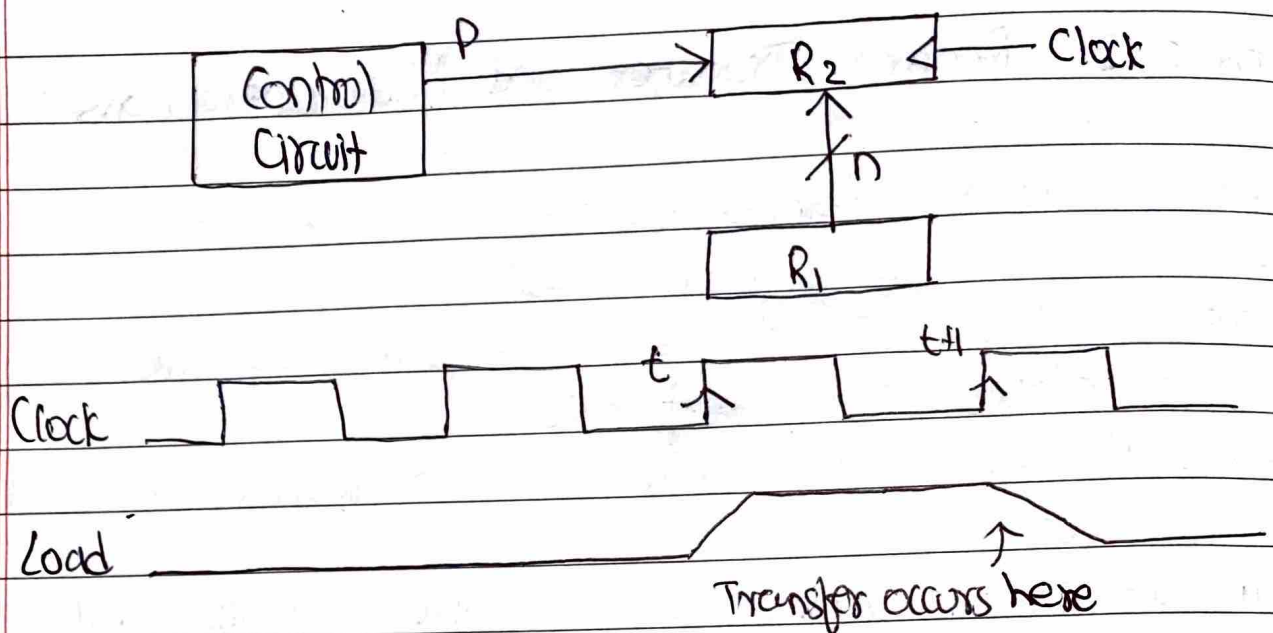
$$R_2 \leftarrow R_1$$

- If there is predetermined control condition like,

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

Then we can write statement as,

$P: R_2 \leftarrow R_1$ where P is control signal usually a control function.



It is assumed that all transfers occur during a clock edge transition. If P becomes active at t, transfer occurs at t+1 clock time. A comma is used to separate two or more operations that are executed at the same time.

T: $R_2 \leftarrow R_1$, $R_1 \leftarrow R_2$, it denotes when T=

Symbol	Description	Example
Letters (numerals)	Denotes a register	MAR, R_2
Parentheses	Denotes a part of register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	$R_2 \leftarrow R_1$
Comma ,	Separate two microoperations	$R_2 \leftarrow R_1$, $R1 \leftarrow R2$

Example, RTL of fetch cycle:

T1: MAR ← PC

T2: MBR ← [MAR]

T3: IR ← MBR

T4: unspecified, PC ← PC+1

The operations performed within this single unit of time are called micro operations. The n outputs of register R_1 are connected to n inputs of register R_2, register R_2 has a load input that is activated by control variable P.

Bus and Memory Transfer

A typical digital computer has many registers, and paths must be provided to transfer information from one register to another.

The number of wires will be exessive if separate lines are used between each registers. A more efficient scheme for transferring information between registers in a multiple register configuration is a common bus system.

A bus structure consists a set of common lines, one for each bit of a register, through ~~with~~ which binary information is transferred one at a time

Control signals determine which register is selected by the bus during each particular register transfer.

In general, a bus system will multiplex k registers of n bits each to produce an n line common bus. The number of multiplexers needed to construct the bus is equal to n, the number of bits in each register.

The transfer of data from a bus into one of many registers can be accomplished by connecting bus lines to inputs of all registers and activating load control of register selected.

Memory Transfer

The transfer of information from a memory word to outside environment is called a read operation. The transfer of new information to be stored into the memory is called a write operation.

A memory word will be symbolized by letter M. It is necessary to specify address of M when writing memory transfer operations. It is done by enclosing the address in square brackets following letter M.

Read: DR ← M[AR]

This causes a transfer of information into DR from the memory word M selected by address in AR.

Shift Micro Operations

The operations on data in registers are called microoperations. The elementary operation performed during one clock pulse on the information stored in one or more registers is called micro operation.

RTL can be used to describe the microoperations, are usually classified into 4 categories:

- Register transfer (transfer with/without control)
- Arithmetic (+, -, *, --, +(1), -(1))
- Logic (∧, ∨, ⊕, NOT)
- Shift

The operations that changes adjacent bit position of binary values stored in register is known as shift micro operation. They are used for serial data transfer. They are used in conjuction with arithmetic, logical, and other processing operations. The contents of register can be shifted to left or right. They are mainly classified into 3 types:

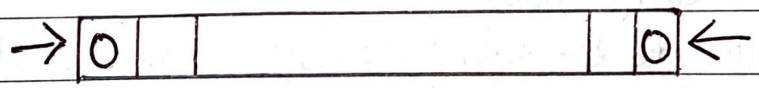
① Logical shift

A logical shift transfer by 0 through serial input, It can be defined by (RTL):

R ← shl R ; Shift -left

R ← shr R ; shift -right

When data values on register be shifted right or left by value of 0 (as serial input).



② Circular shift

A circular shift rotates the bit from one end of register to another end of the register. It can be defined in RTL by:-

R ← cil R ; circular shift -left register

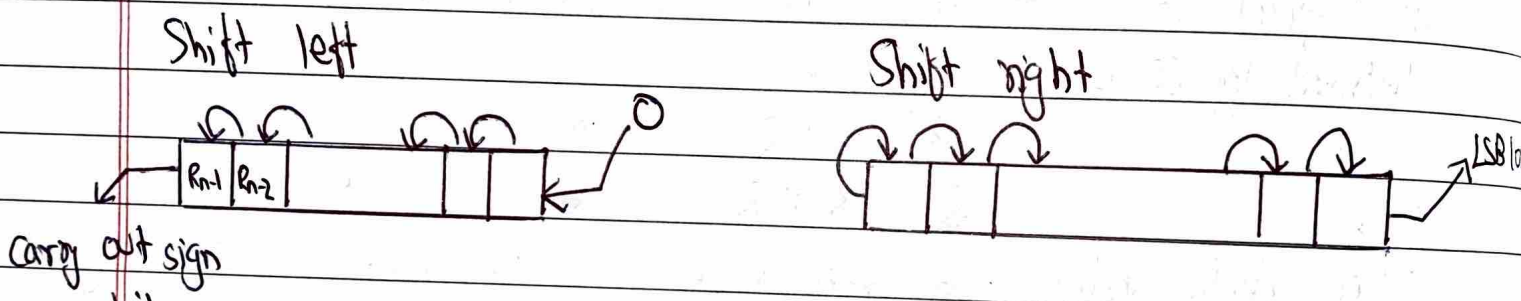
R ← cir R ; circular shift -right register

When data values of register shifted towards right or left direction without loss of any bit i.e LSB is replaced by MSB and vice versa.

③ Arithmetic Shift

It shifts signed-binary number left or right.
For shift left the content of register is multiplied by 2 whereas
For shift right the content of register is divided by 2.
It must leave sign bit unchanged.

R ← ashl R ; arithmetic shift-left
R ← ashr R ; arithmetic shift-right



Overflow case during ASL:

If R_{n-1} bit changes ~~bit~~ sign after shift, sign reversal occurs in result. This ASL must be checked for overflow: an overflow occurs if $R_{n-1} \neq R_{n-2}$ before ASL operation.