realme Shot on realme C35
2023/12/07 15:18

Arithmetic Micro operations

In general, the Arithmetic Micro operations deals with the operations performed on numeric data started in the register

The Arithmetic operations are classified as follows

1. Addition
2. Subtraction
3. 1's compliment
4. 2's compliment
5. Increment of 1
6. Decrement of 1

Addition:-

In addition micro-operations, the value in register R_1 is added to the value in value in register R_2 and then sum is transfered to register R_3.

$$R_3 \leftarrow R_1 + R_2$$

Subtraction:

In subtraction Micro-operation, the contents of register R_2 are subtracted from contents of Register R_1 and then result is transfered into R_3

$$R_3 \leftarrow R_1 - R_2$$

1's compliment:-

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

$$R_1 \leftarrow \bar{R_1}$$

2's compliment:-

In this M-O, the complement of the value inside the register R_2 is taken and then 1 is added to the value then the final result is transfered into register R_2. This process is also known as Negation. It is equivalent to $-R_2$

$$R_2 \leftarrow \bar{R_2} + 1$$

Increment of 1:-

In this M-O, the value inside the R_1 register is increased by 1

$$R_1 \leftarrow R_1 + 1$$

Decrement of 1:

In this M-O, the value inside the register is decreased by 1

$R_1 \leftarrow R_1 - 1$

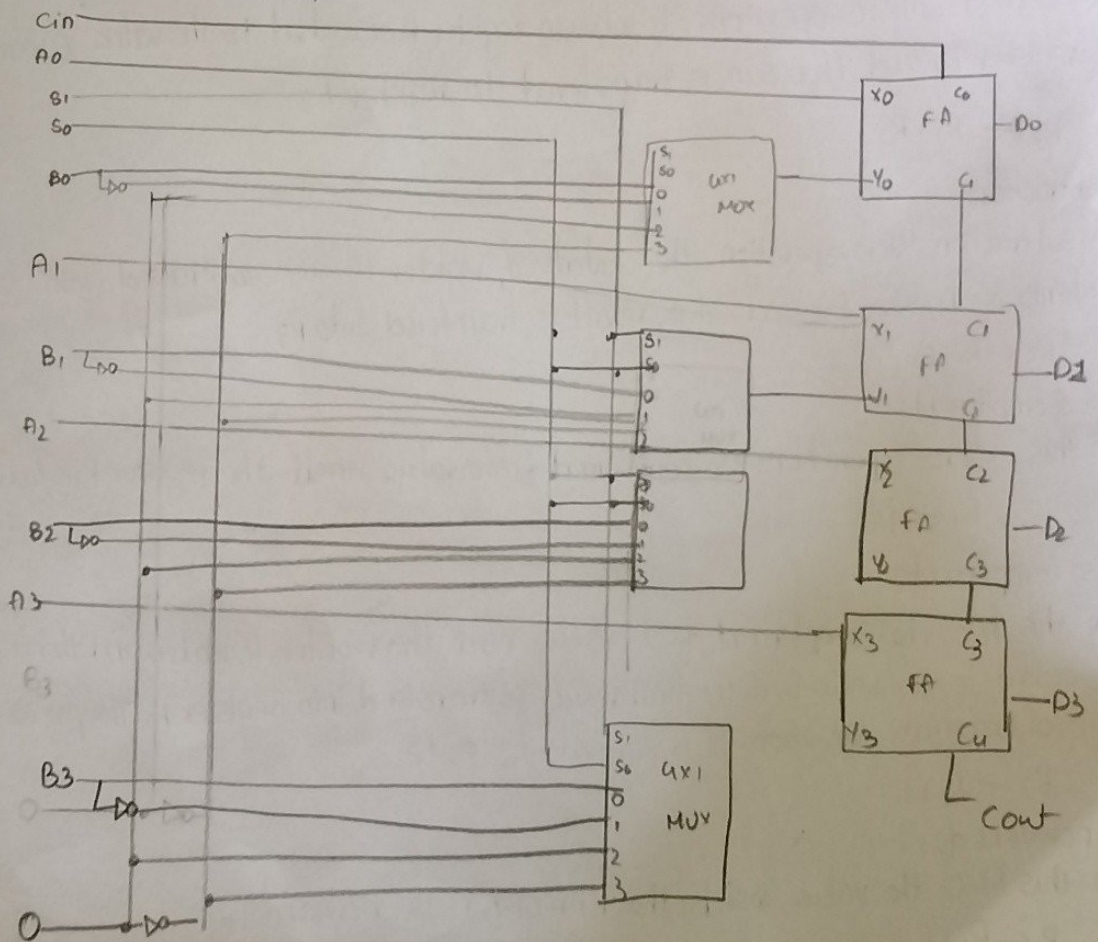
There are several types of arithmetic M-O that can be performed on register data, including:

→ Multiplication
→ Division
→ shift

Arithmetic Circuit:-

The basic component of an arithmetic circuit is the parallel adder 4-bit circuit has 4 full- adder that consists 4bit adder & 4-multiplexors for choosing different operations

output: A + Y + Cin



4-bit arithmetic circuit

Select			Input	Output	Micro-operation
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\bar{B}$	$D = A + \bar{B}$	Sub with borrow
0	1	1	$\bar{B}$	$D = A + \bar{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

Addition:- when $S_1S_0 = 00$

$Y = B$

If $C_{in} = 0$, output $\Rightarrow D = A + B$

If $C_{in} = 1$ output $\Rightarrow D = A + B + 1$

Subtraction

when $S_1S_0 = 01$

$Y = \bar{B}$

If $C_{in} = 0$ Output $\Rightarrow D = A + \bar{B}$

If $C_{in} = 1$ output $\Rightarrow D = A + \bar{B} + 1$

Increment

when $S_1S_0 = 10$

$Y = 0$ [B are neglected]

If $C_{in} = 0$, output $\Rightarrow D = A$

$C_{in} = 1$, output $\Rightarrow D = A + 1$

Decrement when $S_1S_0 = 11$

$Y = 1$

If $C_{in} = 0$ Output $\Rightarrow D = A - 1$

$C_{in} = 1$ Output $D = A$

realme Shot on realme C35
2023/12/07 15:19

Logical Microoperations

Logic operators are binary micro-operations implemented on the bits saved in the registers. These operations treated each bit independently and create them as binary variables

Example:- $P: R_1 \leftarrow R_1 \oplus R_2$

It specifies a logic microoperation to be execute on the individual bits of the registers provides that the control variable $P = 1$

1 0 1 0 content of R_1
1 1 0 0 content of R_2
0 1 1 0 content of R_1 after $P=1$

Exclusive -OR microoperation stated above represent the above logic computation

Special Symbols:-
OR - $\vee$
AND - $\wedge$
's - y's [similar]

In Microoperations '+' $\longrightarrow$ arthmetic plus
In control(Boolean)OP '+' $\rightarrow$ OR operation

$P + Q : R_1 \leftarrow R_2 + R_3 \;,\; R_4 \leftarrow R5 \vee R6$
(OR) (+) (OR)

List of Logic Microoperations:

There are 16 different logic operations that can be Performed with two binary variables

X Y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Sixteen Logic Microoperations

Boolean function	Microoperations	Name
$F0 = 0$	$F \leftarrow 0$	Clear
$F1 = xy$	$F \leftarrow A \wedge B$	AND
$F2 = xy'$	$F \leftarrow A \wedge \bar{B}$	AND
$F3 = x$	$F \leftarrow A$	Transfer A
$F4 = x'y$	$F \leftarrow \bar{A} \wedge B$	AND
$F5 = y$	$F \leftarrow B$	Transfer B
$F6 = x \oplus y$	$F \leftarrow A \oplus B$	exclusive-OR
$F7 = x + y$	$F \leftarrow A \vee B$	OR
$F8 = (x+y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Ex-NOR
$F10 = y'$	$F \leftarrow \bar{B}$	Complement B
$F11 = x + y'$	$F \leftarrow A \vee \bar{B}$	
$F12 = x'$	$F \leftarrow \bar{A}$	Complement A
$F13 = x' + y$	$F \leftarrow \bar{A} \vee B$	
$F14 = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F15 = 1$	$F \leftarrow$ all 1's	Set to all 1's

Hardware Implementation:

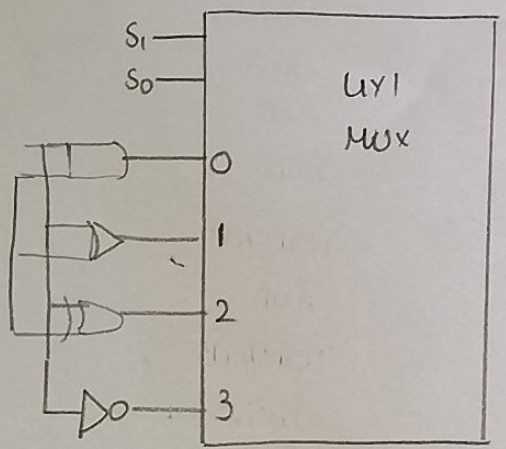
The Hardware implementation of logic microoperations requires that logic gates be inserted.

There are 16 logic M-O, most computers use only four - AND, OR, XOR & its complement

→ for a logic circuit with n bits, the diagram must be Repeated n times. for $i = 0, 1, 2 \cdots, n-1$

The selection variables are applied to all stages.

one stage of logic Circuit



(logic diagram)

S1 S0	output	operation
0 0	E = A∩B	AND
0 1	E = A∨B	OR
1 0	E = A⊕B	XOR
1 1	E = Ā	compliment

functional table

Shift Micro-operations

Shift micro-operations are those which are used for the serial transfer of information. These are also used in conjunction with arithmetic micro-operation, logic m.o, and other data-processing operations

There are three types of shift Micro-operation

1. Logical shift
2. Arithmetic shift
3. Circular shift

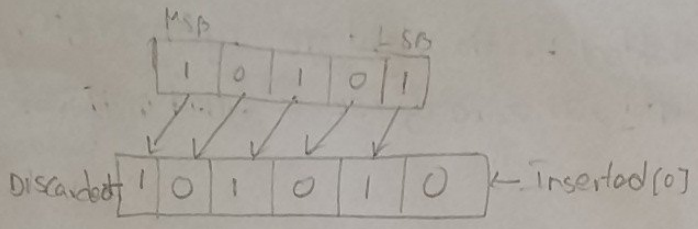
1. Logical shift:

It transfer the 0 zero through the serial input. We use shl and shr for logical shift-left and shift-right M.O and symbols '<<' and '>>' for the logical left and right shift.

Logical Left shift:

In this shift, one position moves each bit to the left one by one. The empty least significant bit (LSB) is filled with zero and most significant bit (MSB) is rejected. It is denoted by (<<)

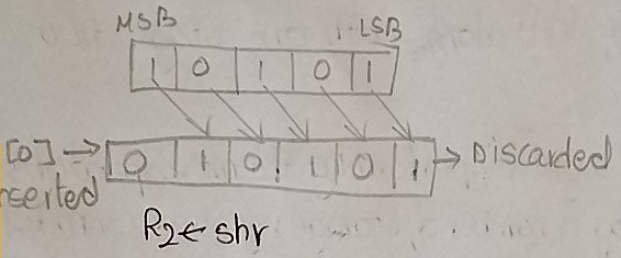
Syntax: <<K



Logical Right Shift

In this shift each bit moves to the right one by one and LSB is rejected and empty MSB (MSB) is filled with zero

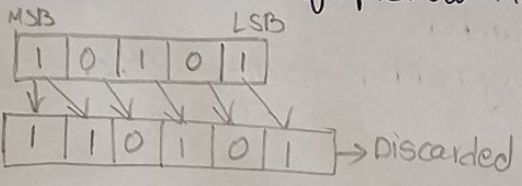
Syntax: >> K



$R_2 \leftarrow shr$

Arithmetic Right shift:

Here, each bit is moved to right one-by-one and LSB is rejected and MSB is filled with value of previous MSB

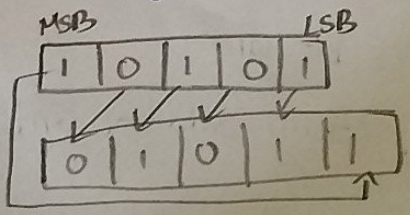


3. Circular shift:

The circular shift circulats the bits in the sequence of register arround both ends without any loss of information. It is also known as circular shift we use symbols cil and cir for left & right

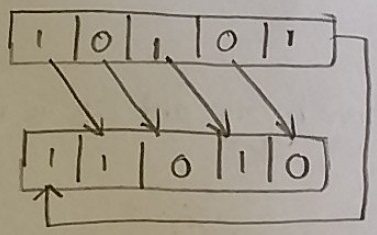
Circular left shift:

Here, each bit in register is shifted to left one by one. After shifting, the LSB becomes empty. So the value of MSB is filled in there



Circular Right shift:

Here each bit in register is shifted to the right one by one. After shifting, the MSB becomes empty, so value of LSB is filled in there.



4. Explain various functions of ALU along with one stage ALU block diagram?

Instead of having individual registers performing the microoperations directly, computer systems employ a number of storage registers connected to a common operational unit called an arithmetic logic unit (ALU).

Various functional performed by ALU:-

i) Arithmetic operations
- Addition: Adds two binary numbers
- Subtraction: Subtracts one binary number from another
- Multiplication
- Division

ii) Logical operations
- AND, OR, NOT: Execute logical operations on binary inputs
- Bitwise shifts

iii) Comparision operations
iv) Data movement
v) Conditional operations
vi) States - Flag updates

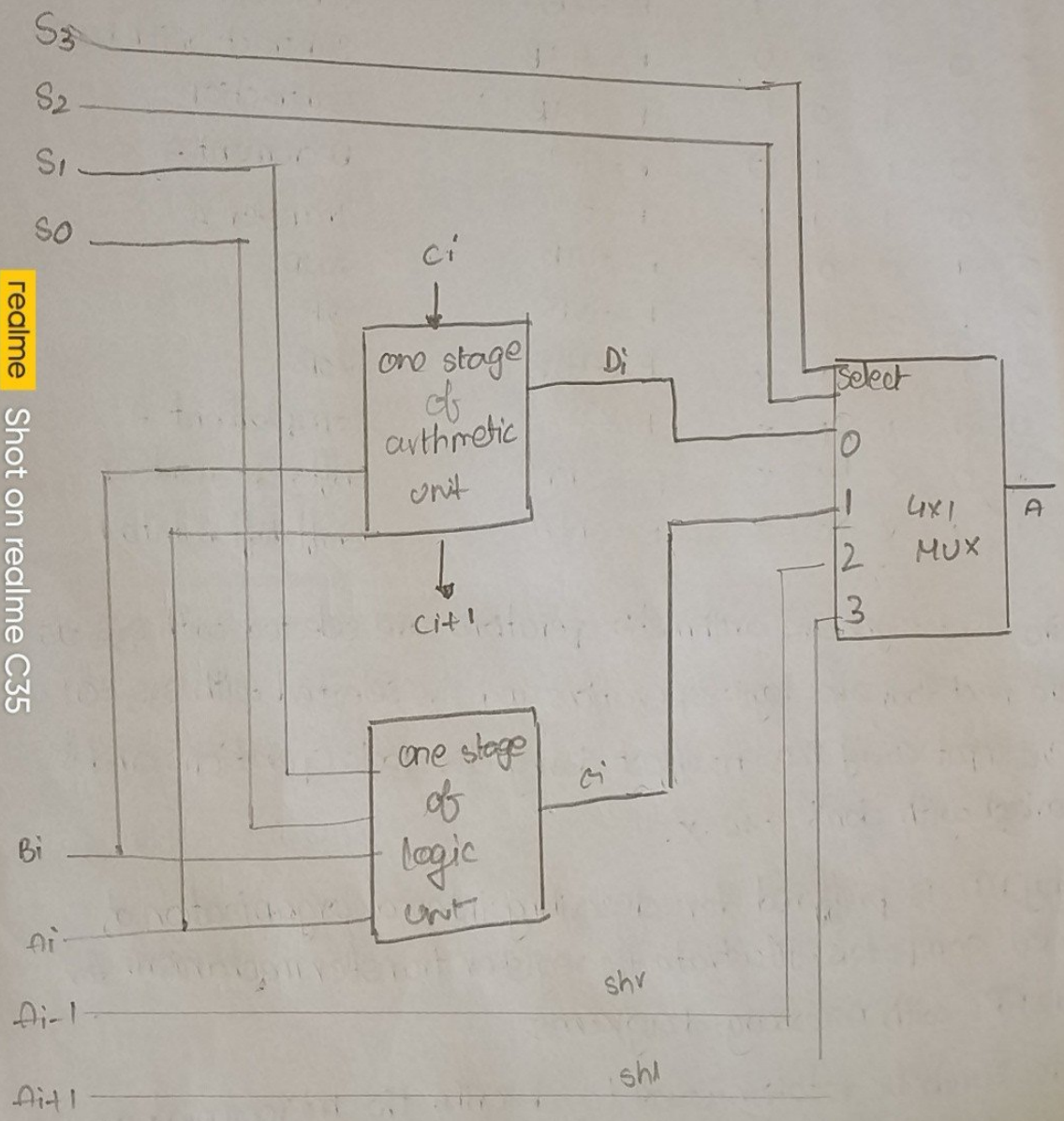
⇒ The arithmetic and logic unit is an 8-bit unit

⇒ It consists of the binary adder to perform addition and subtraction by 2's compliment method

⇒ The ALU performs an operation and the result of the operation is then transfered to a destination register

⇒ The ALU is a combinational circuit so that the entire register transfer operation from the source registers through the ALU and into the destination register can be performed during one clock pulse period.

one stage of arithmetic logic shift unit:



•function table for Arithmetic logic shift unit.

Operation select						
S_3	S_2	S_1	S_0	Cin	Operation	function
0	0	0	0	0	F=A	Transfer A
0	0	0	0	1	F=A+1	Increment A
0	0	0	1	0	F=A+B	Addition
0	0	0	1	1	F=A+B+1	Add with carry
0	0	1	0	0	F=A+$\bar{B}$	Subtract with borrow
0	0	1	0	1	F=A+$\bar{B}$+1	subtraction
0	0	1	1	0	F=A-1	Decrement A
0	0	1	1	1	F=A	Transfer A
0	1	0	0	x	F=A∧B	AND
0	1	0	1	x	F=A∨B	OR
0	1	1	0	x	F=A⊕B	XOR
0	1	1	1	x	F=$\bar{A}$	complement A
0	0	x	x	x	F=shr A	shift right A into F
1	1	x	x	x	F=shl A	shift left A into F

⇒ The first eight are arithmetic operations are selected with S_3S_2=00

⇒ The next four are logic operations and are selected with S_3S_2=01

⇒ The input carry has no effect during the logic operations and is marked with don't care x.

5. Why RTL is preffered for describing internal organisation of digital compdters. Illustrate the register transfer mechanism for P: $R_2 \leftarrow R_1$ with necessary diagrams.

A. The symbolic notation used to describe the microoperation transfers among registers is called Register Transfer language (RTL)

⇒ The RTL is often preffered for describing the internal organisation of digital computer because it allows designers to specify operations at register transfer level

⇒ RTL provides a level of abstraction that is closer to the hardware than high-level languages, yet it is more readable and manageable than gate-level descriptions

⇒ It can also be used to facilitate the design process of digital systems

⇒ The internal organization of a digital computer is best defined by:

i) The set of registers it contains and their functions

ii) The sequence of micro operations performed in the registers

iii) The control that initiates the sequence of micro operations.

Register transfer:

The term register transfer means the availability of hardware logic circuit that can perform a stated microoperation and transfer the result of the operation to another register.

⇒ Information transfer from one register to another in symbolic form by means of replacement operator

$R_2 \leftarrow R_1$

⇒ The statement denotes a transfer of the content of register R_1 into register R_2

⇒ The destination register (R_2) has a parallel load capability

⇒ The transfer occur only under a pre determined control pocondition

If ($P=1$) then ($P_2 \leftarrow R_1$)

where P is a control signal generated in the control section

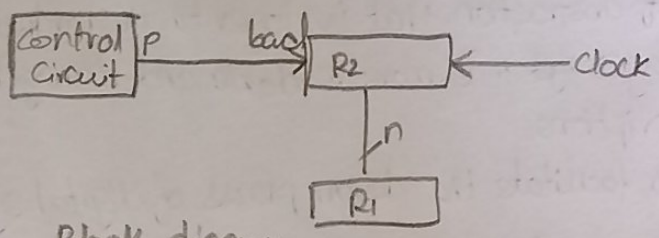
⇒ It is sometimes convenient to separate the control variables by specifying a control function (terminated by colon)

⇒ A control function is a Boolean variable that is equal to 1 or 0

$P: R_2 \leftarrow R_1$

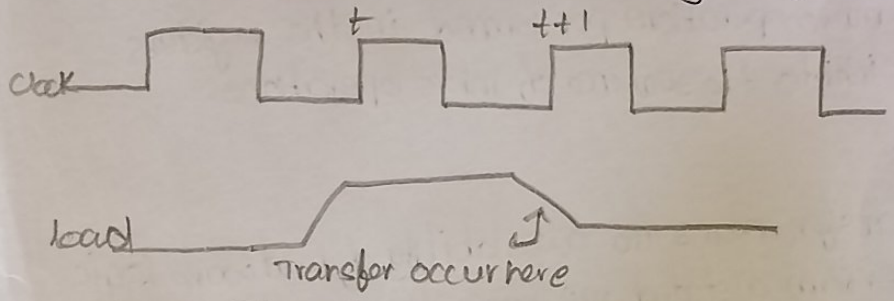
⇒ It symbolize the requirement that the transfer operation be executed by the hardware only if P=1

- Transfer from R_1 to R_2 when P=1



Block diagram

→ It depicts the transfer from R_1 to R_2
→ The letter n coill be used to indicate any no. of bits for registration



Timing diagram

⇒ P is activated in the control section by the rising edge of a clock pulse at time t

⇒ The next positive transition of the clock at time t+1 finds the load input active and the data inputs of R_2 are then loaded into the register in parallel

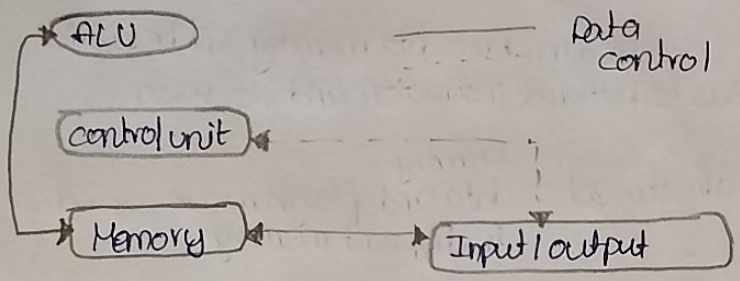
⇒ Even though the control condition such as P becomes active just after time t, the actual transfer does not occurs until the register is triggered by the next position transition of the clock at time t+1

- Basic symbols for Register Transfer

Symbols	Description	Examples
Letter (and numerals)	Denotes a register	MAR, R_2
Paranthesis ( )	Denotes a part of a register	$R_2(0,7)$, Re(L)
Arrow ←	Denotes transfer of information	$R_2 \leftarrow R_1$
comma ,	Sepparate two micro operations	$R_2 \leftarrow R_1$ $R_1 \leftarrow R_2$

6. Von - Neumann vs harvard architecture

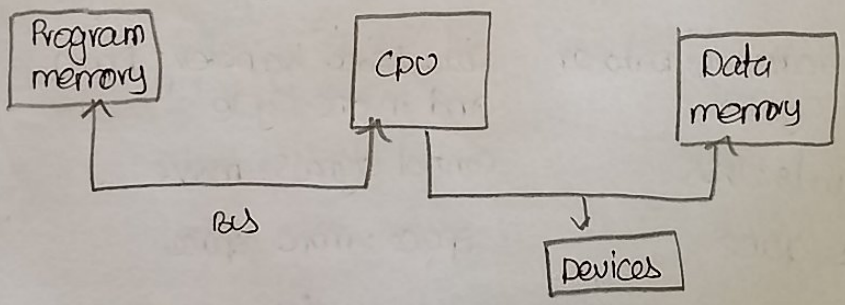
A. Von-Neumann Architecture :



Von Neumann Architecture merges data and instruction memory sharing a bus for both, which can create a bottle neck due to limited band width

Harvard Architecture:

Hardvard Architecture, on the other hand, seperates instruction and data memory using distinct buses, which can enhance Performance but requires more complex design



Von-Neumann architecture	Hardvard architecture
content of the memory if organized and all installed memory can be used	Two memories with two Buses allow parallel access to data access and instructions
One bus is simpler for the Contol unit design	Control unit for two buses is more complicated and more expensive
computer with one bus is cheaper	Program can't write itself
Error in a program can rewrite instruction and crash program execution	Both memories can use different sizes

Development of the control unit is cheaper and faster	Development of complicated control unit needs more time
Data and instruction is accessed in the same way. One Bus is a bottleneck	Free data memory can be used for instruction and vice versa
Memory: data and program are stored in same memory	Memory data and program are stored in different memory
Memory type: It has only RAM for data & code	Memory Type: It has only RAM for Data & ROM for code
Buses: common bus for address & Data/code	Buses: seperate bus for address & Data/code
Program Execution: code is serially executed and takes more cycle	Program execution: code is executed in parallel with data so it takes less cycle.
Data/code Transfer: Data or code in one cycle	Data/code Transfer: Data and in one cycle
control signals: less	Control signals: more
space: less space	space: more space
cost: less	cost: costly.