Date:- 5.8.24

Computer organization and Architecture

Module-1

Books

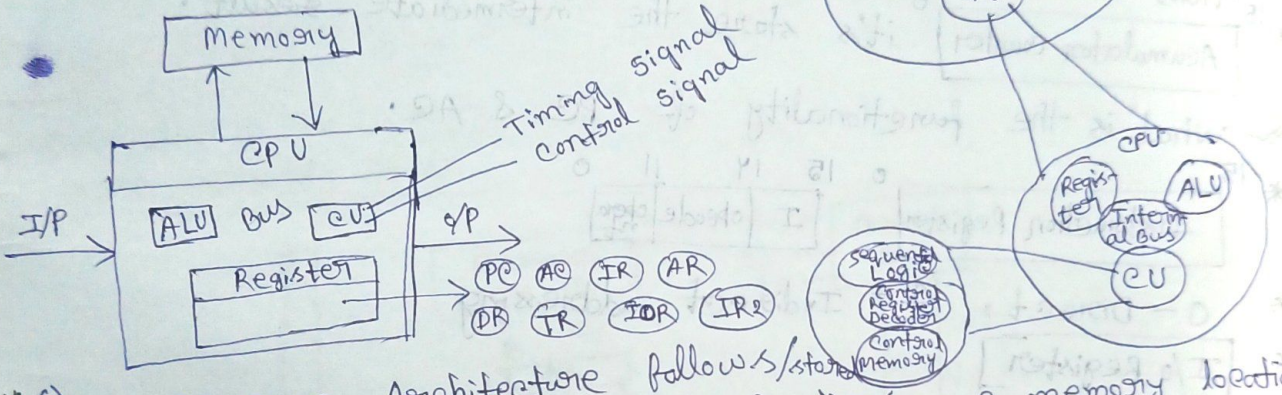
- Morris mano
- Stallings
- Pal choudhury

* Architecture deals with what to do & organization deals with how to do

Structure:- A structure specifies the way in which the components are inter related.

Function:- The operation of each individual component as part of the structure.

Date:- 6.8.24



* (i) Von-Neumann Architecture follows/stores data & instructions in a single location/same memory location.

(ii) It's always works on single microprocessor.

(iii) Fetch-decode-execute cycle ← Principle of Von Neumann

* PC - Programme Counter
IR - Instruction Register
AC - Accumulator Counter
IOR - Input output Register
DR - Data Register
TR - Temporary Register
AR - Address Register

* Von Neumann is also called store programm architecture.

* Store programme control concept
- → Von Neumann Architecture
- → General purpose system
- → Parallel processing

* Whole program's control flow is control unit.

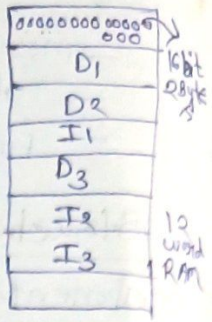
* ~~Parallel processing~~ Three types of Bus - Control, Data & Address.

Date:- 7.8.24

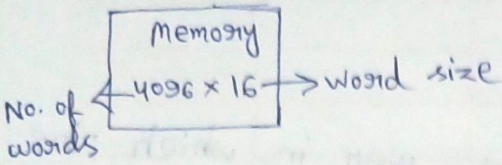
General Purpose Registers

$n = \log_2(8)$
$= \log_2(2^3)$
$= 3\log_2^2 = 3 \times 1 = 3$

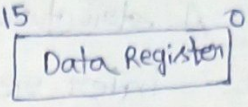
$1024 - 2^{10}$
$2048 - 2^{11}$
$4096 - 2^{12}$



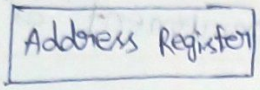
* 1 word = 1 Byte



* Data register only store data.

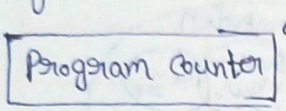


*



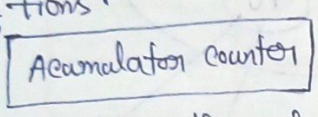
* System can be two types - Word addressable, Byte addressable.

*



special register which it's store the address of the next instructions.

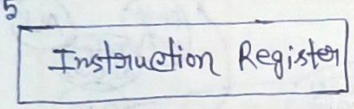
*

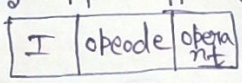


it's store the intermediate result.

Q what is the functionality of PC & AC.

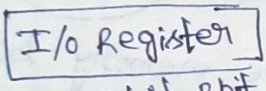
*



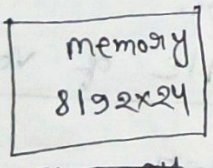


* 0 - Direct, 1 - Indirect addressing

*



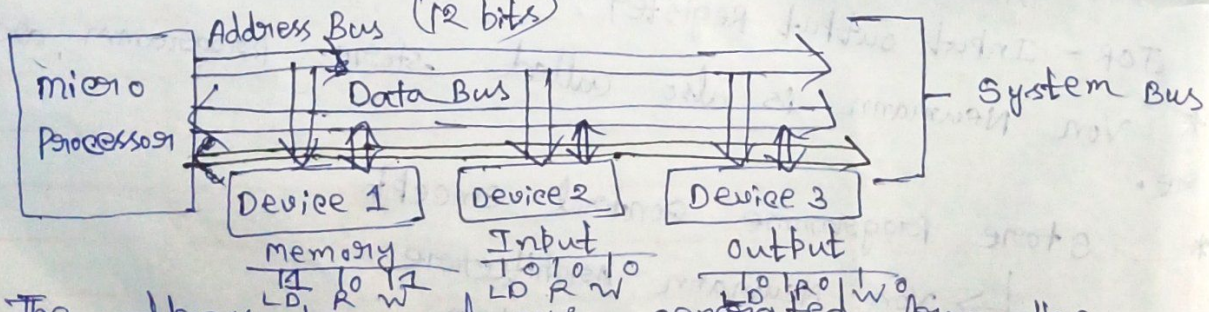
Q



~~8192 x 24~~

Date:- 12.8.24

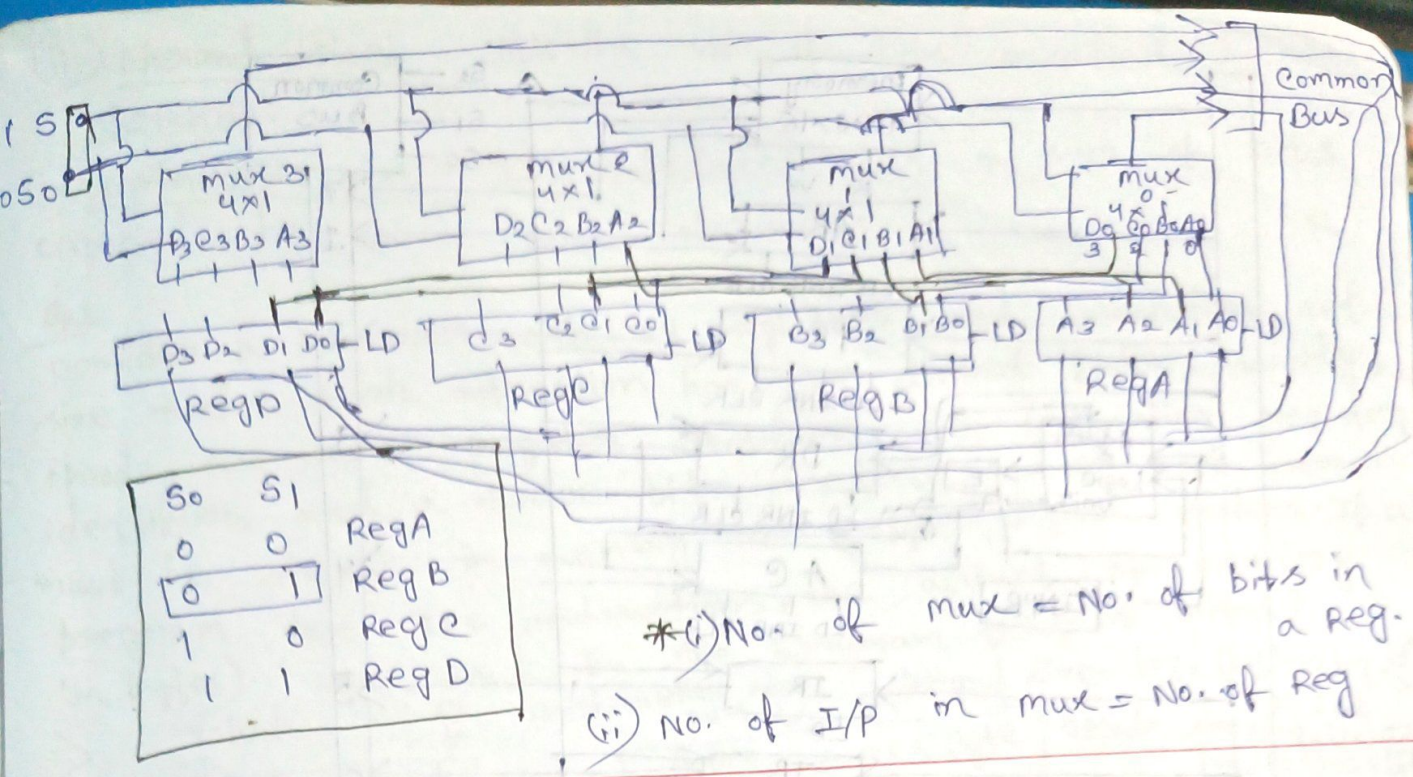
Bus Architecture



* The address bus always generated from the microprocessor.

* Control Bus carries signal or commands from the CPU or control unit.

Common Bus Architecture



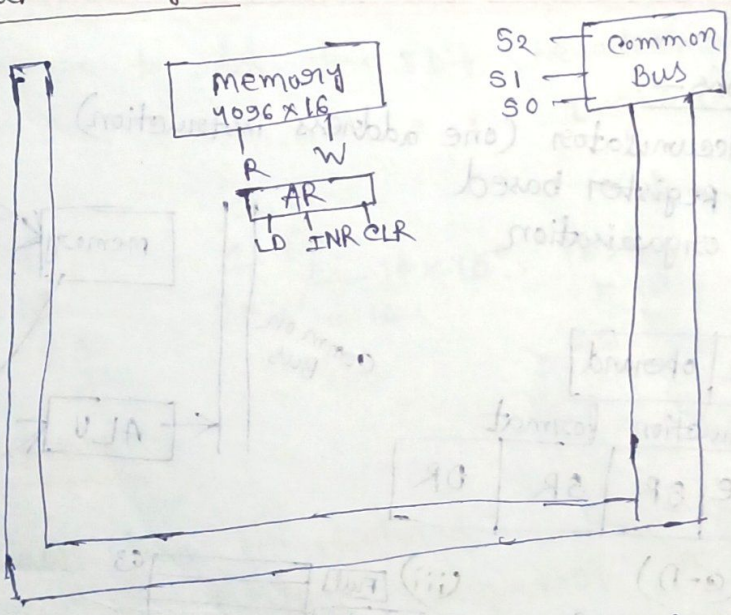
S0	S1	
0	0	RegA
0	1	RegB
1	0	RegC
1	1	RegD

*(i) No. of mux = No. of bits in a Reg.

(ii) No. of I/P in mux = No. of Reg

Date:- 13.8.24

One Address Instruction
- Basic computer Design



Ex:- $V = (A+B) * (C-D)$

```
LOAD A      AC ← M[A]
ADD B       AC = AC + M[B]
Store T     TR ← AC
LOAD C
SUB D
MUL T
STORE X
```

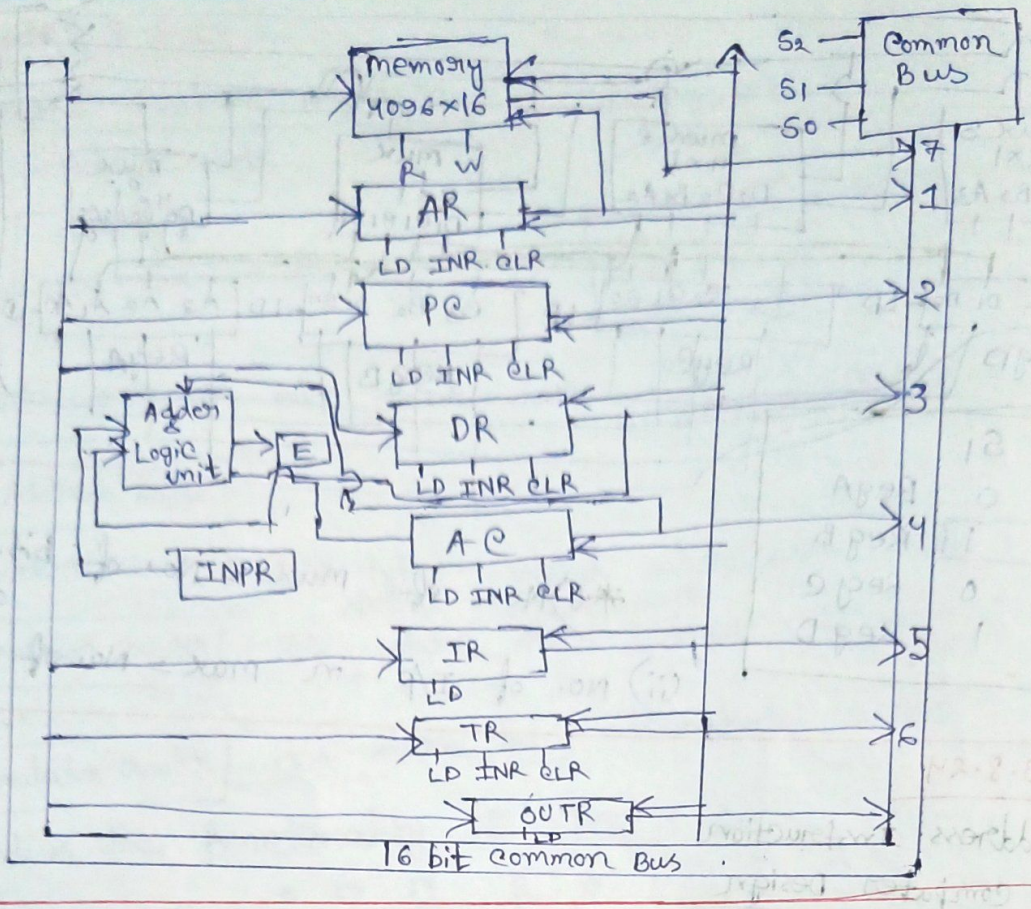
* Absolute Address
Effective "
Relative "

Direct - AC ← AC + [X]

Indirect - AC ← AC + [[X]]

Direct Register - AC ← AC + [R]

Indirect Register - AC ← AC + [[R]]

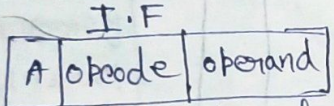


Date:- 14.8.24

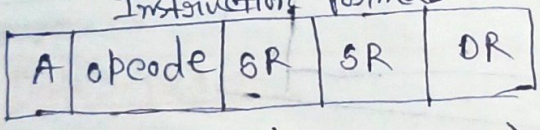
CPU organisations.—

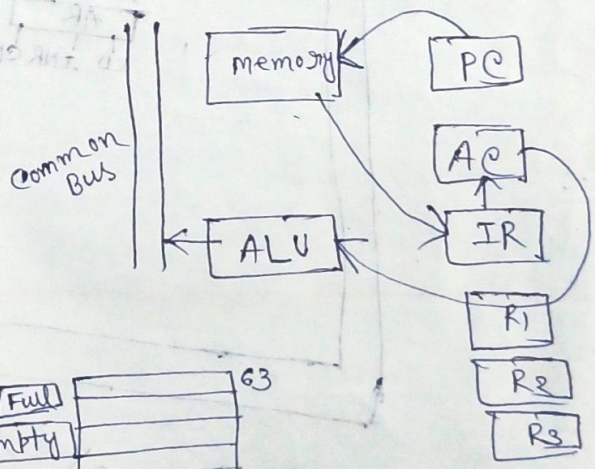
(i) → Single Accumulator (one address instruction)
(ii) → General Register based
(iii) → Stack organisation

(i)



(ii)

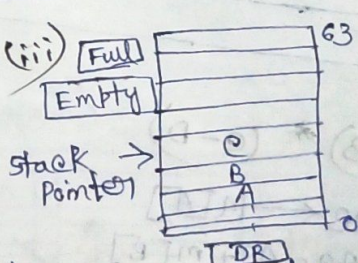




* X = (A+B) * (C-D)

⇒ ADD A, B, R1 . R1 = A+B
SUB C, D, R2 . R2 = C-D
MUL R1, R2, X

(iii)



* Stack organization divided into two parts-
(i) memory stack (ii) Register stack.

Note:- (i) Instruction set size- It tells the total no of instruction define in the processor. (246)

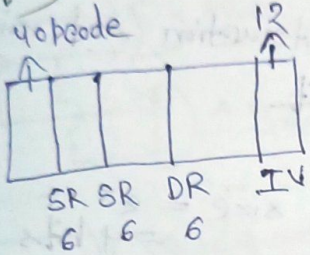
(ii) opcode size:- It's the no. of bits occupied by the opcode which is calculated by taking $\log_2$ of instruction set size. ($\log_2(246)$)

(iii) Operand size – It's the no. of bits occupied by the operands.

(iv) Instruction size – It's calculated as a sum of bits occupied by opcode and operands.

Q.1 Consider a processor with 64 registers and instruction set size twelve. Each instruction has five distinct fields, namely, opcode, two source register identifiers, one destination register identifier, and a twelve-bit immediate value. Each instruction must be stored in memory in a byte-aligned fashion. If a program has 100 instructions, the amount of memory (in bytes) consumed by the program text is what?

⇒



Instruction size = 34 bits

$\frac{3400}{8} = 425$ Byte

To convert bit to byte we've to divide by 8

operand size = $\log_2(64) \approx \log_2(6$

opcode size = $\log_2(12) \approx 6$ bit
$\log_2(24) \approx 4$ bi

LAB

① Write a program to add two 8 bit Hexadecimal No.

② " " " " " " " " Decimal No.

17
7 × 16⁰ = 7
1 × 16¹ = 16
= 23

1A
A 10
10 × 16⁰ = 10 × 1
= 10
1 × 16¹ = 16
= 26

23
26
49
16 | 49 – 1
3

∴ 31 value of Accumulator

Date:–19-8-24

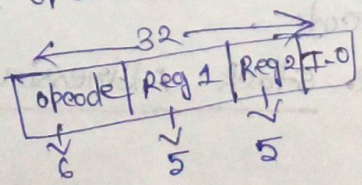
Q.2 A processor has 40 distinct instructions and 24 general purpose register. A 32 bit instruction word has an opcode, 2 register operands and an immediate operand. Calculate the no. of bits available for the immediate operand field?

⇒ 40 distinct instructions → $\log_2 40 = 6$ bits
24 general purpose register → $\log_2 24 = 5$ bits

Opcode – 6 bits
Register operand – 10 bits
Immediate operand – x bits

Opcode + Register operand + Immediate operand = 32

$6 + 10 + x = 32$
$16 + x = 32$
$x = 32 - 16 = 16$ bits



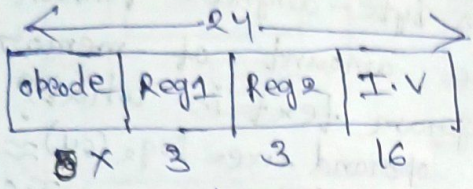
Q.3 A processor with 8 registers uses a 24 bit instruction the instruction word containts an opcode, 2 register identifiers and 16 bit immediate value. Determine the instruction set size of the processor.

⇒ 8 registers

operand size = $\log_2 8 \approx \log_2(2^3) \approx 3$ bit

24 bit instructions

~~opcode size = $\log_2(24) \approx \log_2(e5) \approx 5$ bit~~



∴ opcode + Reg1 + Reg2 + I.V = Instruction set size

~~Instruction set size = 27 bit~~

∴ opcode size $x + 3 + 3 + 16 = 24$

$x + 22 = 24$

$x = 24 - 22$

$= 2$ bit

∴ Instruction set size $= 2^2$

$= 4$ bits

* Addressing modes

- Implied/Implicit Addressing mode
- Stack addressing mode
- Immediate Addressing mode
- Direct " "
- Indirect " "
- Register Direct " "
- Register Indirect " "
- * Relative " "
- Indexed " "
- Base Register " "
- ~~Ortho~~Auto Increment " "
- Auto Decrement " "

Date:- 20.8.24

~~Implicit AM~~ :-

1. The definition of the instruction itself specify the operands implicitly. It's also called 0 address instruction.

Ex:- 'CMA' - Compliment Accumulator. where Accumulator is an operand which is specified implicitly in the opcode.

2. Stack Addressing mode:-

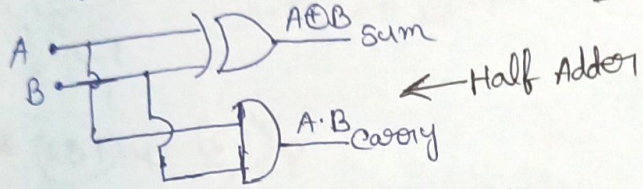
Date:- 21.8.24 Module-2

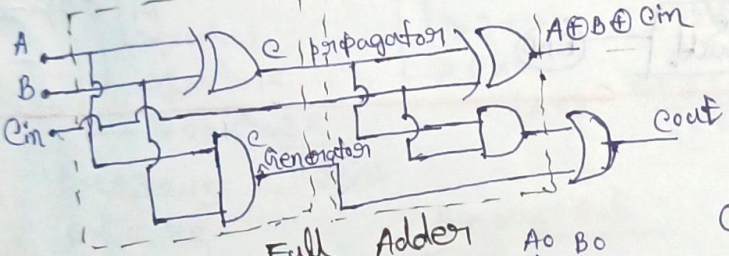
Half Adder

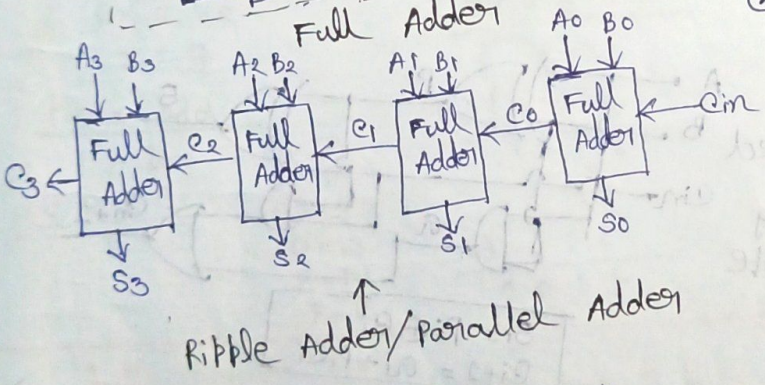
A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum = $A'B + AB' = A \oplus B$

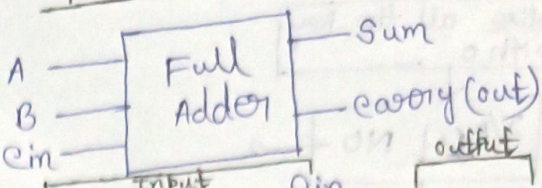
carry = $A \cdot B$







Full Adder



Input			output	
A	B	Cin	Sum	carry (out)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K-map

Cin \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}in$	0	1	3	2
Cin	4	5	7	6

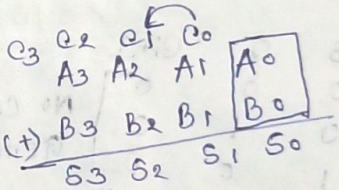
Sum = $\bar{A}\bar{B}Cin + \bar{A}B\bar{C}in + AB\bar{C}in + A\bar{B}C$

$= \bar{A}(\bar{B}C + B\bar{C}in) + A(BCin + \bar{B}\cdot\bar{C}$

$= \bar{A}(B \oplus Cin) + A(B \odot C)$

$= A \oplus B \oplus C$

Cout = $ACin + BCin + AB$

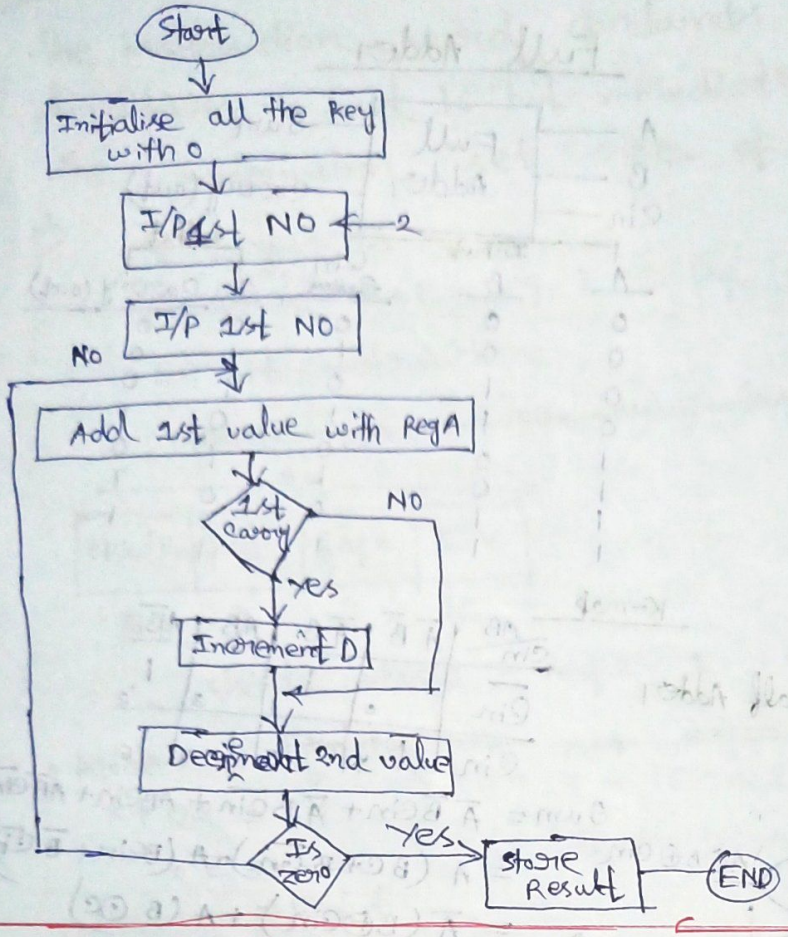


* Drawback of Half Adder - (i) It can only take 2 input.
(ii) It also performed in a positional manner.

* Drawback of full Adder - (i) It can include carry but it also add in a positional manner. If there is multibit number then the 1 bit can't be added.

* Ripple Adder - (i) It can only perform addition.
(ii) There is a propogation delay.

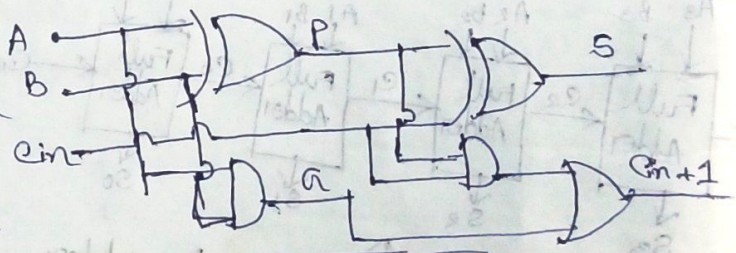
LAB



Date:- 2.9.24

Carry Look-ahead Adder

A	B	Ci	Ci+1	
0	0	0	0	No carry generated
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry propgrate
1	0	0	0	
1	0	1	1	
1	1	0	1	carry generate
1	1	1	1	



$S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_i C_i$

$i=0 \quad C_1 = G_0 + P_0 C_0 = (A_0 \cdot B_0) + (A_0 \oplus B_0) C_0$

$= A_0 B_0 + A_0 C_0 \oplus B_0 C_0$

$i=1 \quad C_2 = G_1 + P_1 C_1 = (A_1 \cdot B_1) + (A_1 \oplus B_1) C_1$

$= G_1 + P_1 (G_0 + P_0 C_0) \quad (A_0 B_0 + A_0 C_0 \oplus B_0 C_0)$

$= G_1 + P_1 G_0 + P_0 P_1 C_0 = (A_1 \cdot B_1) + (A_1 \oplus B_1)$

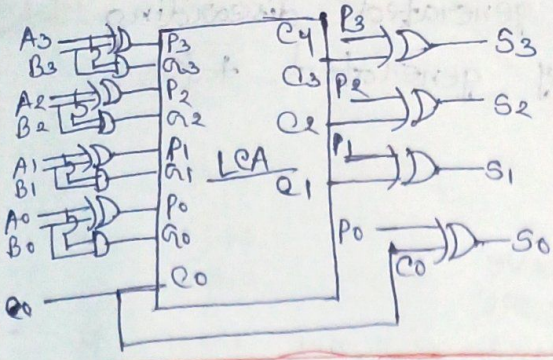
$i=2 \quad C_3 = G_2 + P_2 C_2 = (A_2 \cdot B_2) + (A_2 \oplus B_2) \cdot C_2$

$= (A_2 \cdot B_2) + (A_2 \oplus B_2)((A_1 \cdot B_1) + (A \oplus B_1)(A_0 B_0 + A_0 C_0 + B_0 C_0))$

$C_3 = G_2 + P_2 C_2$

$= G_2 + P_2 (G_1 + P_1 G_0 + P_0 P_1 C_0)$

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$



Date:- 4.9.24

Number System Conversion

Decimal to Binary

$(12.264)_{10}$

12 → $0.264 \times 2 = 0.528$

$0.528 \times 2 = 1.056$

$0.56 \times 2 = 0.112$

2 | 12 – 0
2 | 6 – 0
2 | 3 – 1
1

Binary to Decimal

$(1101.11)_2$

$1 \times 2^0 = 1$
$0 \times 2^1 = 0$
$1 \times 2^2 = 4$
$1 \times 2^3 = 8$
13.75

$11 \rightarrow 1 \cdot 2^{-2}$
$1 \cdot 2^{-1}$
$= 0.5 + 0.25$
$= 0.75$

* $(231)_4 = (63)_?$

$(231)_4$
$1 \times 4^0 = 1$
$3 \times 4^1 = 12$
$2 \times 4^2 = 32$
$(45)_{10}$

$6x + 3 = 45$
$6x = 42$
$x = 7$

63
$3 \times x^0$
$6 \times x^1$
$= 6x + 3$

Octal Add

4567
(+) 2357
(7 1 3)$_8$

Binary Addition

1110
1101
11011

Hexa Add

1AB
(+) 2BC
469

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

Binary Sub

2 3 4 (10)
(–) 1 4 6
0 8 8

1 1 0 1
(+) 1 0 1 0
0 0 1 1

* 1000000
(–) 010111
001011

0 A D (16)
(–) 0 B 9
0 F 4

LAB

① write a program to divide two 8 bit numbers?

Date:- 9.9.24

1's compliment

1 1 0 1 1 0
(+) 0 1 1 1 0 1
1 | 0 1 0 0 1 1
1
0 1 0 1 0 0

Process:- (i) First identify the minuend & substrahead.
(ii) Find 1's compliment of the substrahead.
(iii) Now add the minuend and 1's compliment of the substrahead.
(iv) If carry generated add the carry with the rest of the result. If no carry generated that would be the final result.

2's compliment

1 1 0 1 1 0
(+) 0 1 1 1 1 0
*0101100

10110
(-)11010
11100

Process :- (iv) If carry generated, discarding the value, if no carry generated take the 2's compliment.

Unsigned

000	-0
001	-1
010	-2
011	-3
100	-4
101	-5
110	-6
111	-7

Range- 0 to $2^n - 1$

Sign bit (msb)

000	+0
001	+1
010	+2
011	+3
100	-0
101	-1
110	-2
111	-3

-3 | +3

0 - +ve
1 - -ve

Range - $-2^{n-1}+1$ to $+2^{n-1}-1$

$= -2^2+1$

$= -4+1$

$= -3$ to $+4-1$

$= -3$ to $+3$

Sign	2's	Decimal
0000	0000	+0
0001	0001	+1
0010	0010	+2
0011	0011	+3
0100	0100	+4
0101	0101	+5
0110	0110	+6
0111	0111	+7
1000	1000	-8
1001	0111	-7
1010	0110	-6
1011	0101	-5
1100	0100	-4
1101	0011	-3
1110	0010	-2
1111	0001	-1

Range = -2^{n-1} to $+2^{n-1}-1$

-8 | 0 | +7

* -8 -7 -6 -5 -4 -3 -2 -1 0 +1+ 2 +3 +4 +5 +6 +7

(+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8) (+8)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Date:-18.9.24

Booth's Algorithm

-5 ← multiplicand ← B

4 ← multiplier ← Q

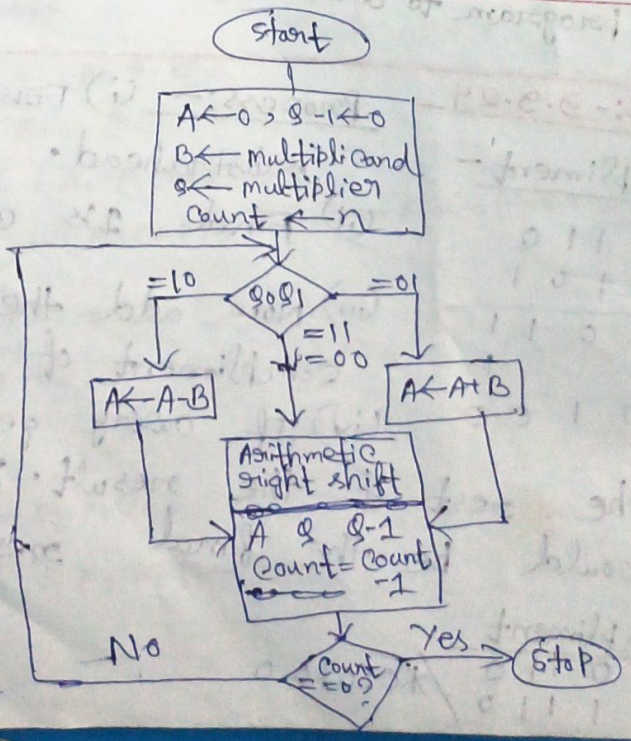
Multiplication of (-5)×4

(-5) = 0101

$(1011)_2$ ← (-5) ← B

$(4) = (0100)_2$ ← Q

Count → 4



Steps	A	Q	Q-1	operation
	0000	0100	0	Initialization
1	0000	0010	0	Right Shift
	0000	0001	0	Right shift
3	0101	0001	0	A ← A − B
	0010	1000	1	Right Shift
4	1101	1000	1	A ← A+B
	1110	1100	0	Right Shift

A − B
= A + (−B)
= 0 + (−(−5))
= +5 → 0101

A + B
= 0010
1011
1101

00010100
11101100
(−20)

* multiplication of 7×3

⇒ 7 = 0111 ← B
3 = 0011 ← Q

Steps	A	Q	Q-1	operation
	0000	0011	0	Initialization
	1001	0011	0	A ← A − B
1	1100	1001	1	Right Shift
				Right Shift
2	1110	0100	1	A ← A+B
	0101	0100	1	Right Shift
3	0010	1010	0	Right Shift
4	0001	0101	0	

(+) 0000
1001
1001

1110
0111
1 0001
+1
0110

Date:-11.9.24

Floating Point Representation

- Fixed Number → Integer 1., 4., 5.
  → Fraction 0.62, 0.5, 0.97

Radix Point
1.256 = 125.6 × 10^{-2}
= .01256 × 10^{2}

- Floating Number

S - Sign bit
m - mantissa
E - Exponent

11010 : 101110
= 11.010101110 × 2^{3} (mantissa)
= 11010101110.0 × $2^{(5)}$ ← Exponent

S	E	m
0	11	01010111

* 101.1 = 0.1011 × 2^{3}

0	0011	10110
S	E	m
(1)	(4)	(5)

S = 1 bit
m = 5 bit
E = 4 bit

Normalization :-

(i) Explicit : $(-1)^{S} \times 0.m \times 2^{E}$
(ii) Implicit : $(-1)^{S} \times 1.m \times 2^{E}$

Q .00110 = ~~0.0110~~ 00.110 × 2^{-2}

S = 1 bit
m = 5 bit
E = 4 bit

0	1110	11000
S	E	m

2 = 0010
1110

Backward mechanism

$(-1)^{0} \times 0.10110 \times 2^{0011}$
$= 1 \times 0.10110 \times 2^{3}$
$= 101.180$

Q $(-5.875)_{10}$
⇒ 101.111
$= 0.101111 \times 2^{3}$

1	0011	10111
S	E	m

Implicit

$(-5.875)_{10}$
$= 101.111 = 1.01111 \times 2^{2}$

1	0010	011111
S (1)	E (4)	m (m)

Date:-12.9.24

$\Rightarrow (-1)^S \times 0.M \times 2^E$

$= (-1)^1 \times 0.10111 \times 2^3$

$= (-) 101.11$

$= 5.75$

Implicit Conversion

-5.875

$= 101.111$

$= 1.01111 \times 2^2$

Biasing

(i) Explicit $= (-1)^S \times 0.M \times 2^{E-B}$

(ii) Implicit $= (-1)^S \times 1.M \times 2^{E-B}$

Bias

$-(8)\ -7\ -6\ -5\ -4\ -3\ -2\ -1\ 0\ +1\ +2\ +3\ +4\ +5+$

$6 \quad +7$

$.00101 = 00.101 \times 2^{-2}$

0	0110	10100

$(-1)^0 \times 0.101\ 00 \times 2^{6-8}$

$= 0.10100 \times 2^{-2}$

$= .00101 00$

101.11

$= .10111 \times 2^3$

$3+8 \quad -2+8$

$= 11 \quad = 6$

1011

Q $(0101)_2$

$\Rightarrow 0.101 \times 2^{-1}$

0	0111	10100
S	E	M

Q $(101.101)_2$

$\Rightarrow$

-2^{n-1} to $2^{n-1}-1$ $(n=4)$

$= -2^{4-1}$ to $2^4-1 \quad = -8$ to $+7$

Bias $= 8$

$E = -1+8 = 7 \quad = 0111$

IEEE754

Name - Binary-16 Common Name - Half Presision

Significant Bit-11 $(5+M)$ Exponent bit-5 Exponent Bias-15

Representation

S	E	M
1	5	10

(ii) Binary 32 - Single Presision - 24 - 8 - 127 -

S	E	M
1	8	23

(iii) Binary 64 - Double Presision - 53 - 11 - 1023 -

S	E	M
1	11	52

(iv) Binary 128 - Quadrable Presision - 113 - 15 - 16383

S	E	M
1	15	112

(v) Binary 256 - Octuple Presision - 237 - 19 - 262143

S	E	M
1	19	236

Date:-17.9.24

Single Presision

S(1)	E(8)	M(23)
0 or 1	0000 0000 (E=0)	000...0 (M=0)

Representation

± 0

0 or 1	1111 1111 (E = 255)	000...0 (m = 0)	± ∞
0 or 1	1 ≤ E ≤ 254	xxx... (m ≠ 0)	Implicit Normal Form $(-1)^S \times 1.M \times 2^{E-127}$
0 or 1	E = 0	m ≠ 0	Fractional Form $(-1)^S \times 0.M \times 2^{-126}$
0 or 1	E = 255	m ≠ 0	NAN

Double Precision

S (1)	E (11)	m (52)	Representation
0 or 1	000 0000 0000 (E = 0)	000...0 (m = 0)	± 0
0 or 1	111 1111 1111 (E = 2047)	000...0 (m = 0)	± ∞
0 or 1	1 ≤ E ≤ 2046	xxx... (m ≠ 0)	Implicit Normal Form $(-1)^S \times 1.M \times 2^{E-1023}$
0 or 1	E = 0	m ≠ 0	Fractional Form $(-1)^S \times 0.M \times 2^{-1022}$
0 or 1	E = 2047	m ≠ 0	NAN

Q Consider a 32 bit register which stores floating point numbers in a IEEE single precision format. Find the decimal value of the following 32 bit - 01000001111100...0

⇒ | S | E | m |
|---|---|---|
| 0 | 10000011 | 11100...0 |

$(-1)^0 \times 1.110\ldots0 \times 2^{131-127}$

$= 11100.000\ldots0$

$= (28)_{16}$

Q | S | E | m |
|---|---|---|
| 0 | 11111111 | 00...0 |

E = 11111111 = NAN

= 255

~~$(-1)^0 \times 1.110\ldots0 \times 2^{255-127}$~~

~~$= 1.110\ldots0 \times 2^{128}$~~

Q Represent the numbers $(116)_{10}$ in IEE754 single and double precision format.

⇒ $(116)_{10} = 6 \times 10^0 = 6$

$1 \times 10^1 = 10$

$1 \times 10^2 = 100$

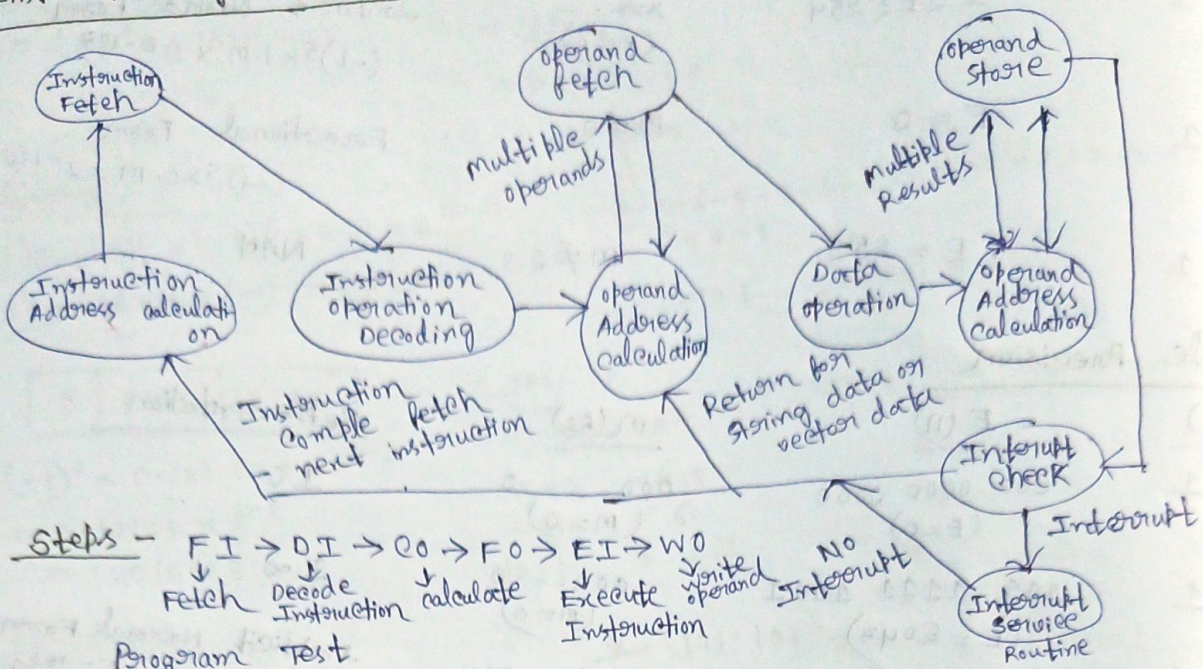
116 — 1110100

$1110100 \times 10^{6+127}$

S	E	m
0	10000101	11010000...00

Date:- 18.9.24

Instruction cycle State Diagram



Steps - FI → DI → CO → FO → EI → WO
Fetch, Decode Instruction, calculate, Execute Instruction, write operand

Program Test

Instruction cycle — Instruction cycle

FI DI CO

micro operation

* RISC - Reduced Instruction set Computer
* CISC - Complex " " "

* RISC → It reduce the cycles per instruction at the cost of number of instructions per program.

* CISC:- The CISC approach attempts to minimize the number of instruction per program but at the cost of an increase in the number of cycles per instruction.

$$* \text{ CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{Instruction}}{\text{program}} \times \frac{\text{cycles}}{\text{Instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

Q Difference between RISC and CISC.

⇒

RISC	CISC
(i) RISC is a reduced instruction set.	(i) CISC is a complex instruction set.
(ii) The number of instructions are less as compared to CISC.	(ii) The number of instructions are more as compared to RISC.
(iii) The addressing modes are less.	(iii) The addressing modes are more.
(iv) It worked in a fixed instruction format.	(iv) It worked in a variable instruction format.
(v) The RISC consumes less power.	(v) The CISC consumes more power.
(vi) The RISC processors are highly pipelined.	(vi) The CISC processors are less pipelined.
(vii) It optimizes the performance by focusing on software.	(vii) It optimizes the performance by focusing on hardware.

(viii) Requires more RAM.	(viii) Requires less RAM.
(ix) They are inexpensive.	(ix) They are relatively expensive.
(x) Examples of RISC chips include SPARC, POWER PC.	(x) Examples of CISC include Intel architecture, AMD.