COMPUTER ARCHITECTURE

CST-301

Date: / / Page No.:

Name - Ashutosh Soni

Id - 2018UCP1505

Ans 1 ⇒

(a). 7 and 13

2	7	1
2	3	1
	1	

$7 \rightarrow (00111)_2$.

2	13	1
2	6	0
2	3	1.
	1	

$13 \rightarrow (1101)_2$

Addition.

00111.
01101
10100 ↝ NO overflow.

Substraction.

2's complement of 13 ⇒.

$(01101)_2$.

1+ 1's complement = 10010+1

= 10011.

Now substraction

7 + (-13).

⇒ 00111
+10011
11010

1 | 1010

sign | this represent magnitude

its 2's complement =.

0110 = 6.

= .

Now Ans

⇒ -6

Date: / / Page No.:

(b). -12 and 9.

$(9)_{10} = (01001)_2$.

$(12)_{10} = (01100)_2$.

Addition:

2's complement of 12 =

(01100) → 10100 ~ (-12)

| 1 | 0 1 0 0 |

(-12) + 9.

```
 10100
 01001
 11101
```

| 1 | | 1 | 1 | 0 | 1 |

sign magnitude.

taking 2's complement of 1101

0011 → represent 3.

Substraction:

9 → $(01001)_2$. -9 → (10111).

+12 → $(01100)_2$. -12 → (10110)

(-12) + (-9). ~~10111~~.

```
 10110,
 10111.
~~100 01101~~
 10110
```

one bit excess, (overflow).

Qu1→(b)

continued:-

Advantages of 2's complement method:-

doesnot requige any carry value &
there exists only one bit pattern for
every number. and problem of -0 & +0
is also not there.

Ans 2→. In the programmed I/O mode of
the data transfer each data item is
transferred by an instruction in the
program. The CPU uses a command.
then waits for I/O operation to be
complete.

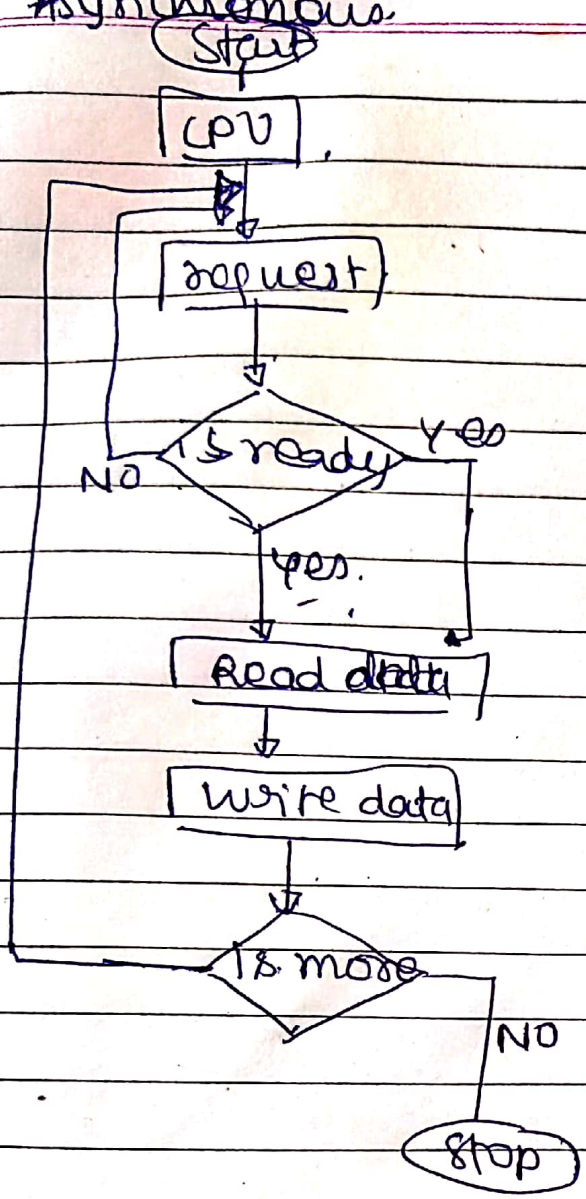
→ As the CPU, faster then I/O. module.
the problem with programmed I/O is.
that if the CPU has to wait a long time
for the I/O module.

two types

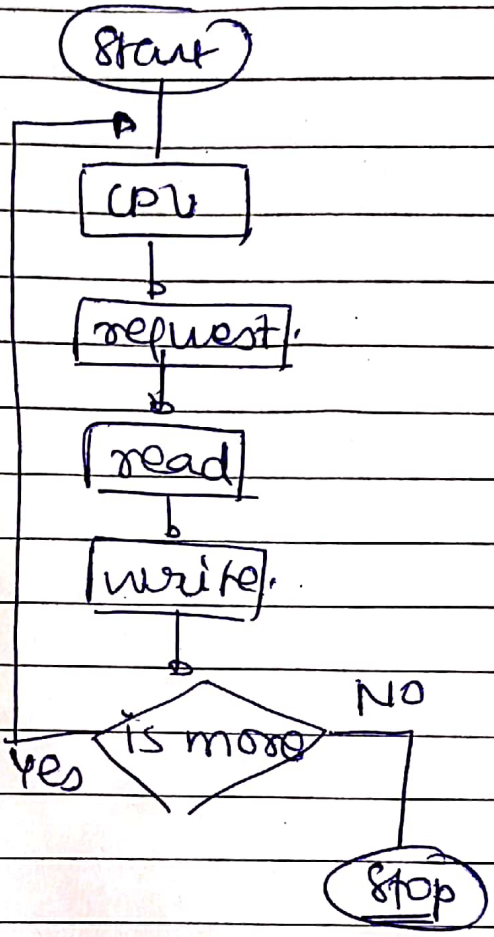
① Asynchronous Data transfer
② Synchronous Data transfer.

Date: / / Page No.:

Asynchronous



Synchronous Data Transfer.



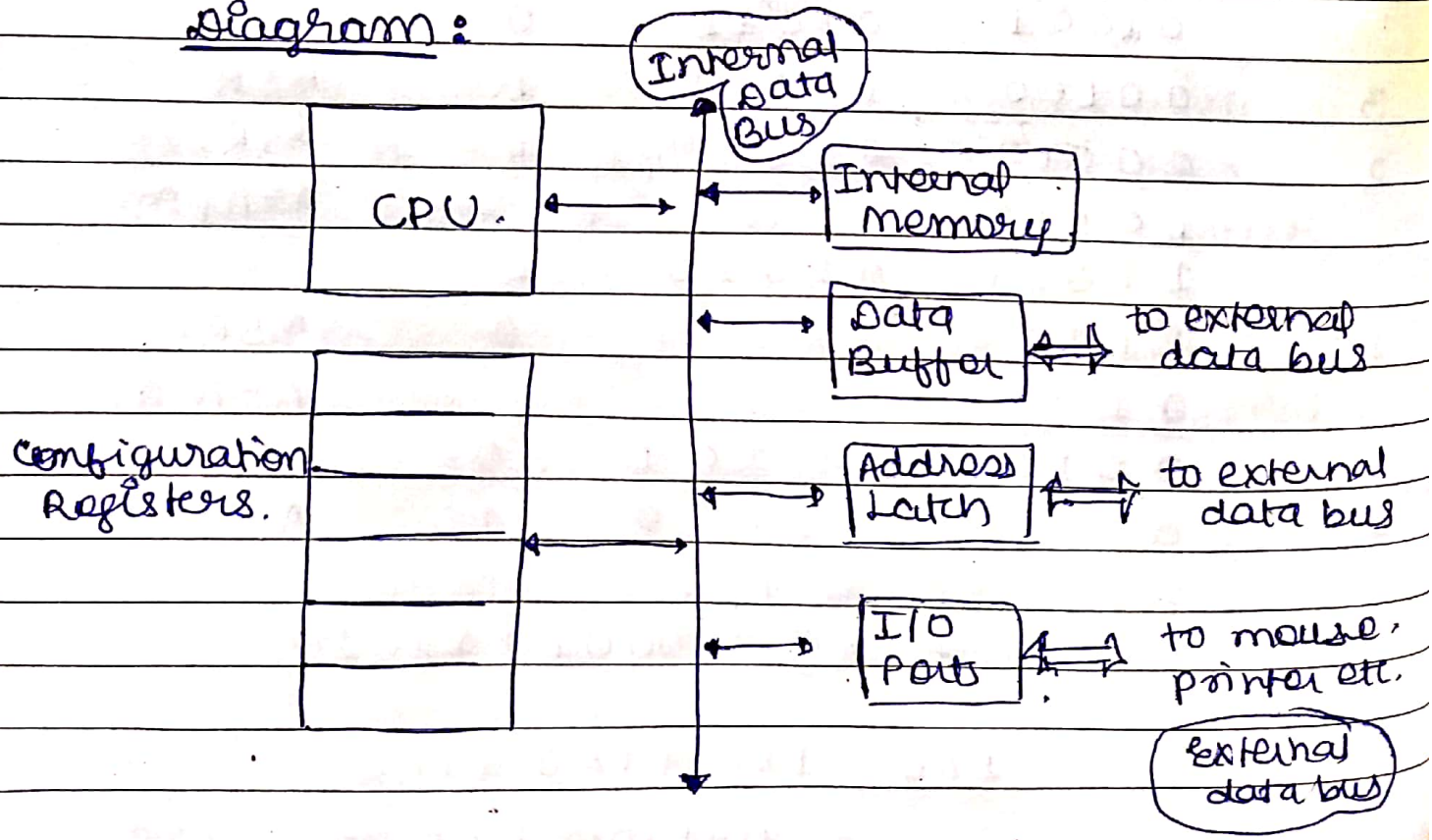
Date: / / Page No.:

Ans 3⇒.

Internal Bus → It is also known as Internal Data Bus. It connects all the internal components of a computer, such as CPU and memory, to the motherboard. Internal data bus are also referred to as local buses, because they are intended to connect local devices. This bus is typically rather quick and is independent of the rest of the computer operations.

External Bus → This bus is made of the electronic pathways that connect the different external devices, such as printer etc to the computer operations.

Diagram:



Date: / / Page No.:

Ans 4 a).

LOAD R2, LOC.

Steps needed mainly:

- Fetch instruction
- Decode instruction.
- Perform ALU operation
- Excess memory.
- load in Register R2 the value stored or pointed by PC.
- Update the program counter.

Their explanation:

first the instruction load is fetched and then instruction is decoded as per norms. then ALU understands that it have to load the content of memory to the register R2. After that it goes to the memory location as given in question that its initially in PC so go to that location & then load the content stored at that memory location in data bus through that bus content is loaded & in register R2. and atlast the PC gets incremented & this instruction got executed.

Date: / / Page No.:

Ans 5⇒.

Memory = 128 MB. = $2^7 \times 2^{20}$ B

1 word = 8 bytes. = 2^3 B.

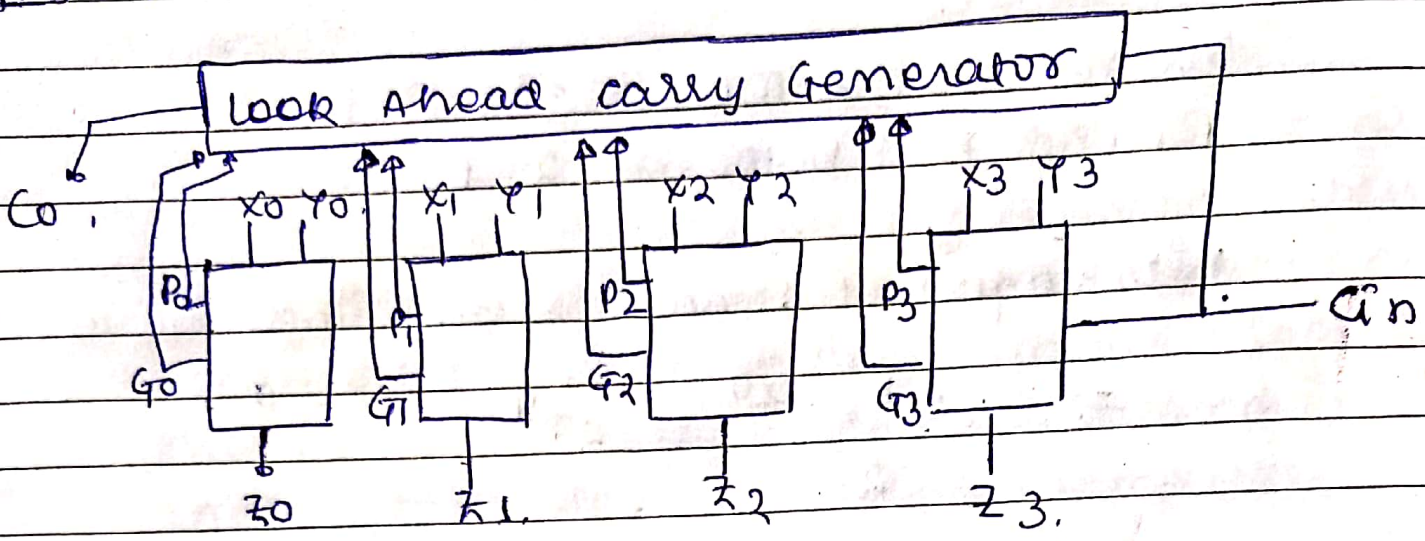
Bits required to address any single word.

⇒. $\frac{2^7 \times 2^{20}}{2^3} = 2^{24}$.

means from here we can say that 24 bits are required to address. single word.

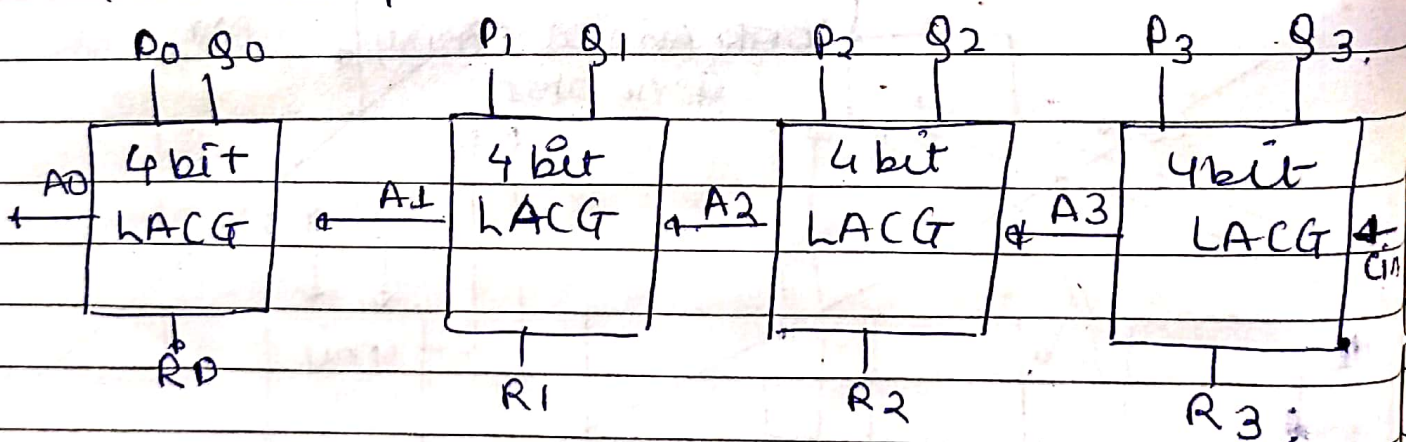
Date: / / Page No.:

Ans 6 ⇒.



$Z_0 Z_1 Z_2 Z_3$ are 1 bit Sum Obtained.
$X_0 X_1 X_2 X_3$ are 4 bit number.
$Y_0 Y_1 Y_2 Y_3$ are 4 bit number.

Now we will make 16 bit serial adder with 4 bit carry adder.



LACG → Look Ahead Carry generator

Date: / / Page No.:

$P_0P_1P_2P_3$ is a 16 bit number where each of P_0, P_1, P_2, P_3 are 4 bit number.

Similarly $R_0R_1R_2R_3$ is 16 bit result where each R_0, R_1, R_2 and R_3 are 4 bit numbers.

Now,

$$C_3 = x_3y_3 + x_3C_{in} + y_3C_{in}.$$

$$\Rightarrow C_3 = x_3y_3 + C_{in}(x_3 + y_3).$$

$$C_3 = G_3 + P_3C_{in}. \quad \text{where}$$

$$G_3 = x_3y_3$$

$$G_3 = x_3 + y_3.$$

$$C_2 = G_2 + P_2C_{in}$$

$$C_2 = G_2 + P_2G_3 + P_2P_3C_{in}.$$

and $$C_4 = G_1 + P_1C_{in}$$

$$= G_1 + P_1G_2 + P_1P_2G_3 + P_1P_2P_3C_{in}.$$

& $$C_0 = G_0 + P_0C_{in}$$

$$= G_0 + P_0G_1 + P_0P_1G_2 + P_0P_1P_2G_3 + P_0P_1P_2P_3C_{in}.$$

Date: / / Page No.:

Ans=7 ⇒

An array multiplier is a digital combinational circuit used for multiplying two binary numbers by employing an array of full adder & Half Half adder.

for 3 bit,

$$\begin{array}{ccccc} & & b_2 & b_1 & b_0 \\ & & a_2 & a_1 & a_0 \\ \hline & & a_0b_2 & a_0b_1 & a_0b_0 \\ & a_1b_2 & a_1b_1 & a_1b_0 & \\ a_2b_2 & a_2b_2 & a_2b_0 & & \\ \hline c_4 & c_3 & c_2 & c_1 & c_0 \end{array}$$

$c_0 = a_0b_0$

$c_1 = a_0b_1 + a_1b_0$

$c_2 = a_0b_2 + a_1b_1 + a_2b_0$

$c_3 = a_1b_2 + a_2b_1$

$c_4 = a_2b_2$

Now partial product are added with the Help of half adders & their sum goes through fulladder to get the desired result.

- A basic multiplier can be divided into three section. partial product generation, partial products addition & final addition.

partial product addition → Half adder

final addition → full adder

Date: / / Page No.:

Performance:

They are fast. means they have high efficiency & low power consumption because of which they are good.

Booths algorithm implementation.

$(-9) \times (10)$.

$m = -9 = 10111$

$-m = 9 = 01001$

$-10 = 10110$

$10 = 01010$

$n = 5$

n	A	Q	Q$_{n+1}$	Action
5	00000	10110	0	Initialization.
4	00000	01011	0	ASR.
	+01001			A=A-m.
	01001	01011	0	
3	00100	10101	1	ASR
2	00010	01010	1	ASR.
	10111			A=A+m.
	11001	01010	1	
1	11100	10101	0	ASR
	01001			A=A-m.
	00101	10101	0	
0	00010	11010	1	ASR.

Result = AQ = 00010 11010

$= 1 \times 64 + 1 \times 16 + 1 \times 8 + 1 \times 2$

$= 64 + 16 + 8 + 2 = 80$ Ans.