Date
DELTA Pg No.

Assignment - 1

(1) $(+42)_{10} = (0101010)_2$

By taking 2's complement

$(-42)_{10} = (1010110)_2$

2	42	0
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1
	0	

$(+13)_{10} = (0001101)_2$

By taking 2's complement

$(-13)_{10} = (1110011)_2$

2	13	1
2	6	0
2	3	1
2	1	1
	0	

(i) $(+42) + (-13)$

$\Rightarrow$ (1) 0 1 0 1 0 1 0
(1) (1)
+ 1 1 1 0 0 1 1
0 0 1 1 1 0 1

Discarding end carry

$(+42)_{10} + (-13)_{10} = (0011101)_2 = (+29)_{10}$

(ii) $(-42) - (-13)$

$\Rightarrow (-42) + (13)$ (1) (1)

$\Rightarrow$ (1) 1 0 1 0 1 1 0
+ 0 0 0 1 1 0 1
1 1 0 0 0 1 1

$\Rightarrow (-42)_{10} + (-13)_{10} = (1100011)_2 \rightarrow$ 2's comp form

(1100011) $\rightarrow$ Taking 2's comp.

$\Rightarrow (0011101)_2 \rightarrow$ Binary int form

$\Rightarrow (-29)_{10}$

Date
DELTA Pg No.

(2) $46.5 = 32 + 8 + 4 + 2 + 0.5$

$= 2^5 + 2^3 + 2^2 + 2^1 + 2^0$

$= (1\,0\,1\,1\,1\,0.1)_2$

0	1 0 1 1 1 0 1 0	0 0 0 0 0 0 0	0 0 0 0 0 1 1 0
↓	↓		↓
1 sign bit	15 bit mantissa		8 bit exponent (+6

(3) (a) $(45)_6 \Rightarrow 4 \times 6^1 + 5 \times 6^0$

$\Rightarrow 24 + 5$

$\Rightarrow (29)_{10}$

9	29	2
9	3	3
	0	

$\Rightarrow (32)_9$

$\Rightarrow (45)_6 = (32)_9$

(b) $(ABC)_{16} \Rightarrow 10 \times 16^2 + 11 \times 16^1 + 12 \times 16^0$

$\Rightarrow 2560 + 176 + 12$

$\Rightarrow (2748)_{10}$

A → 10
B → 11
C → 12

$\Rightarrow (101010111100)_2$

2	2748	0
2	1374	0
2	687	1
2	343	1
2	171	1
2	85	1
2	42	0
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1
	0	

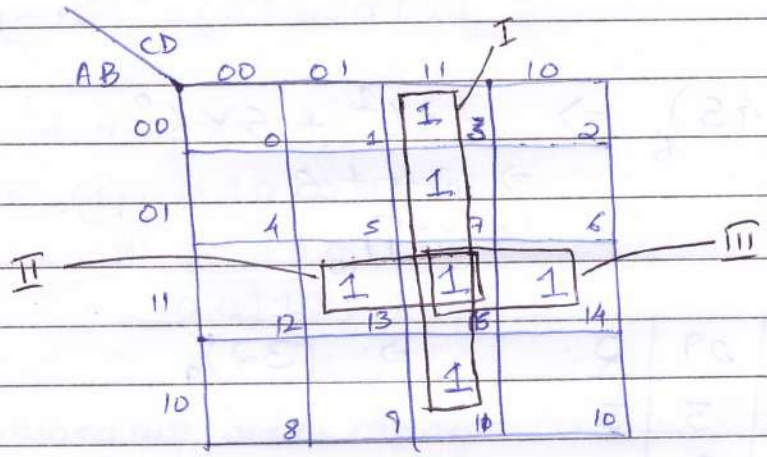
c) $(121121)_3 = (?)_{10}$

$\Rightarrow 1\times3^5 + 2\times3^4 + 1\times3^3 + 1\times3^2 + 2\times3^1 + 1\times3^0$

$\Rightarrow 243 + 162 + 27 + 9 + 6 + 1$

$\Rightarrow (448)_{10}$

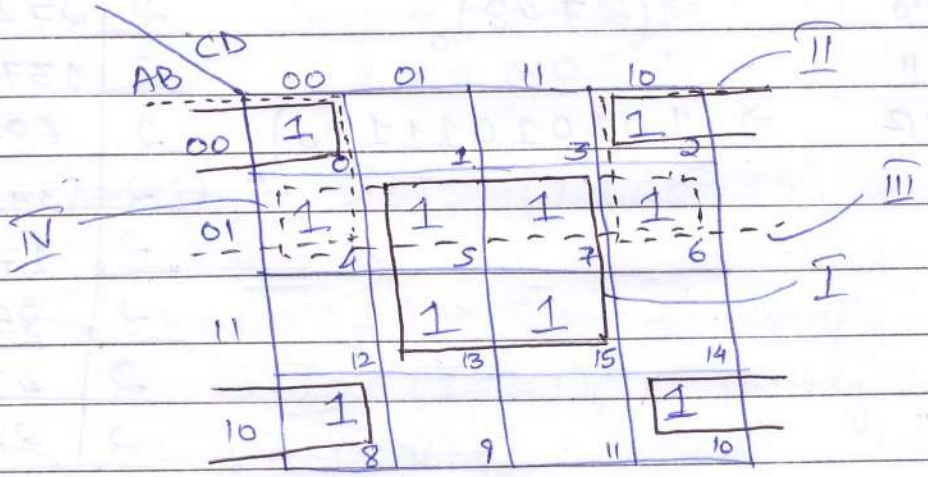
(4) a) $F(A,B,C,D) = \Sigma(3,7,11,13,14,15)$



$F = I + II + III$

$\boxed{F = CD + ABD + ABC}$

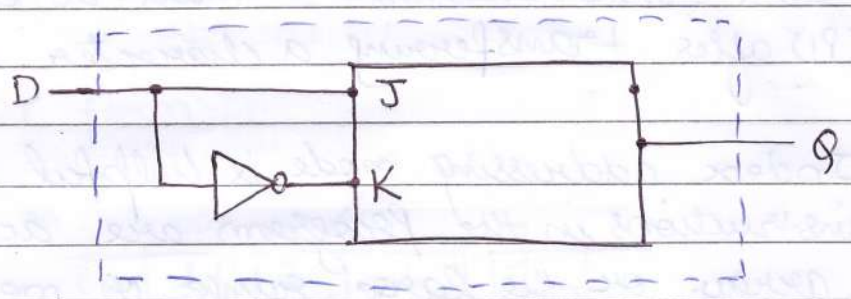
b) $F(A,B,C,D) = \Sigma(0,2,4,5,6,7,8,10,13,15)$



$F = I + II + \boxed{III \text{ or } IV}$

$$F = BD + B'D' + (A'D' \text{ or } A'B)$$

(5)



when $D = 0$; $J = 0$, $K = 1$, $Q \rightarrow 0$

when $D = 1$; $J = 1$, $K = 0$, $Q \rightarrow 1$

(6) A flip-flop is a binary cell capable of storing one bit of information. The stored data can be changed by applying varying inputs

The main drawback of SR flip flop is invalid output when both the inputs are high (1), which is referred to as Invalid state which is not the case in J-K flip-flop where high inputs are well defined.

(7) • The FGI flip-flop is set to 1 after a new character is shifted into INPR. This is done by the I/O operation, not by the control unit This is an example of an asynchronous input event (not synchronized or controlled by the CPU). The FGI flip flop must be cleared after transferring the INPR to AC. This must be done as a micro operation controlled by the CPU, so we must include it in the CU design.

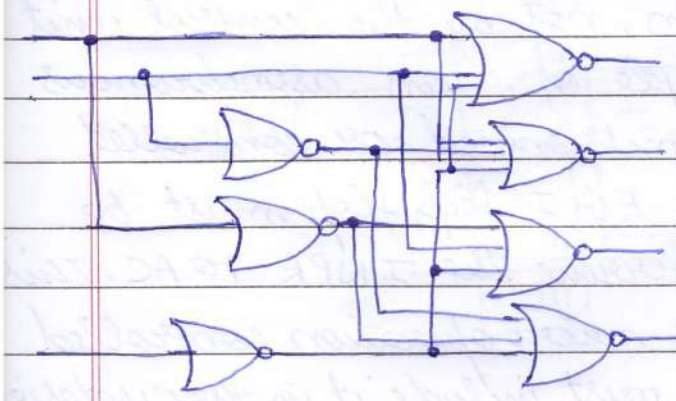
- The FGO flip flop is set to 1 by the I/O interface after the terminal has finished displaying the last character sent. It must be cleared by the CPU after transferring a character into OUTR

→ Index addressing mode is helpful when the instructions in the program are accessing the array or the large range of memory addresses. In this mode, the effective address is generated by adding a constant to the register's content. The content of the register does not change.

Advantage: The index addressing mode provides flexibility to specify memory locations.

(8) The Truth table for the circuit:

E	A	B	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



$D_0 = (A_1 + A_0 + E')' = A_1' A_0' E$

$D_1 = (A_1 + A_0' + E')' = A_1' A_0 E$

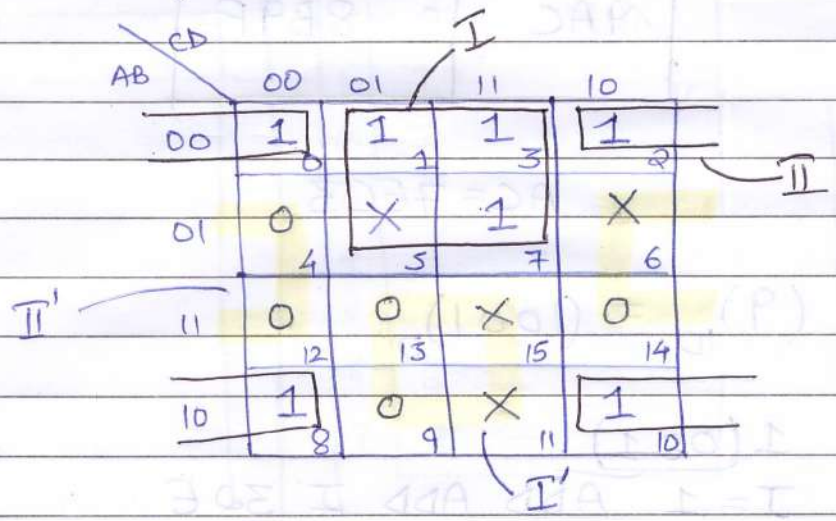
$D_2 = (A_1' + A_0 + E')' = A_1 A_0' E$

$D_3 = (A_1' + A_0' + E')' = A_1 A_0 E$

(9) $F(w,x,y,z) = \Sigma(0,1,2,3,7,8,10)$

$d(w,x,y,z) = \Sigma(5,6,11,15)$

(i) SOP form



$F = I + II$

$\boxed{F = w'z + x'z'}$ → SOP form

(ii) POS form

$F' = I' + II'$

$= xz' + wz$

$(F')' = (xz' + wz)'$

$\boxed{F = (x'+z).(w'+z')}$ → POS form

Date
DELTA Pg No.

10)

Memory

3AF	932E
32E	09AC
9AC	8B9F

AC = 7EC3

(a) $(9)_{10} = (1001)_2$

1(001)
I = 1 ADD ADD I 32E

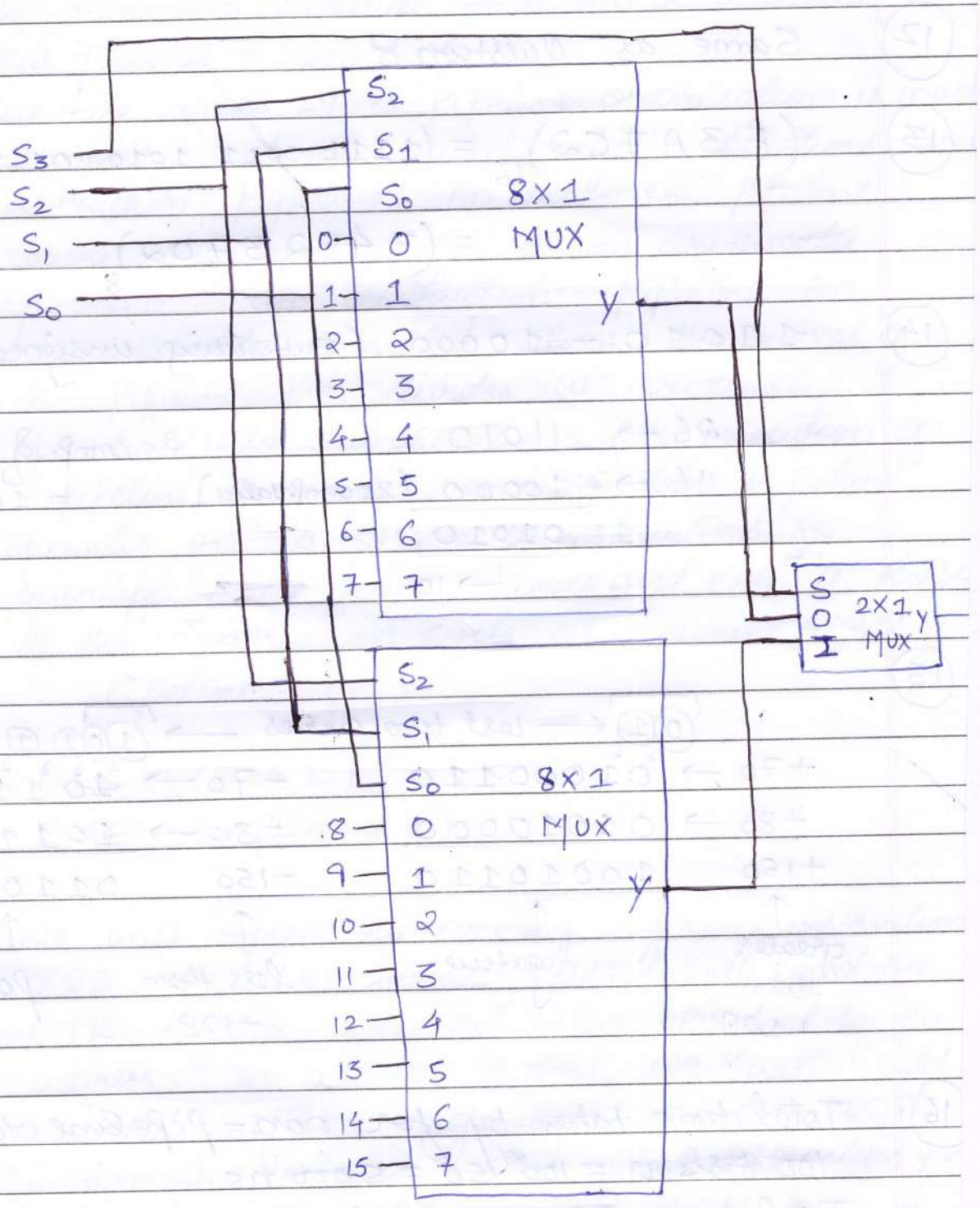
(b) AC = 7EC3 (ADD) ~~DR~~
DR = 8B9F
0A62

7EC3 = 0111 1110 1100 0011
8B9F = 1000 1011 1001 1111
(1) 0000 1010 0110 0010
↓
0A62

(c) PC = 3AF + 1 = 3B0
AR = 7AC
DR = 8B9F
AC = 0A62

IR = 932E
E = 1
I = 1
SC = 000D

11


S3
S2
S1
S0
S2
S1
S0
8X1
MUX
0
1
2
3
4
5
6
7
y
S2
S1
S0
8X1
MUX
8
9
10
11
12
13
14
15
y
S
0
I
2X1
MUX
y

(12) Same as Question 8.

(13) $(F3A7C2)_{16} = (1111\ 0011\ 1010\ 0111\ 1100\ 0010)_2$

$= (74723702)_8$

(14) 11010 − 10000 (Sub. using unsigned numbers)

26 → 11010 — 2's comp. of 10000

16 → + 10000 (2's comp. value) = 10000

1 01010

↳ 10

(15)

(0)(1) ← Last two carries → (1)(0)(1)(1)

+70 → 01000110 −70 → 10111010

+80 → 01010000 −80 → 10110000

+150 10010110 −150 01101010

↑ greater than +127 — ↑ negative — ↑ less than −128 — ↑ positive

(16) Total time taken by for non-pipeline to complete 100 task is = 100 * 50 = 5000 ns.

Total time taken by pipeline configuration to complete 100 task is = (100 + 6 − 1) * 10 = 1050 ns

~~the maximum~~ S

Thus, speed up ratio will be 5000/1050 = 4.76

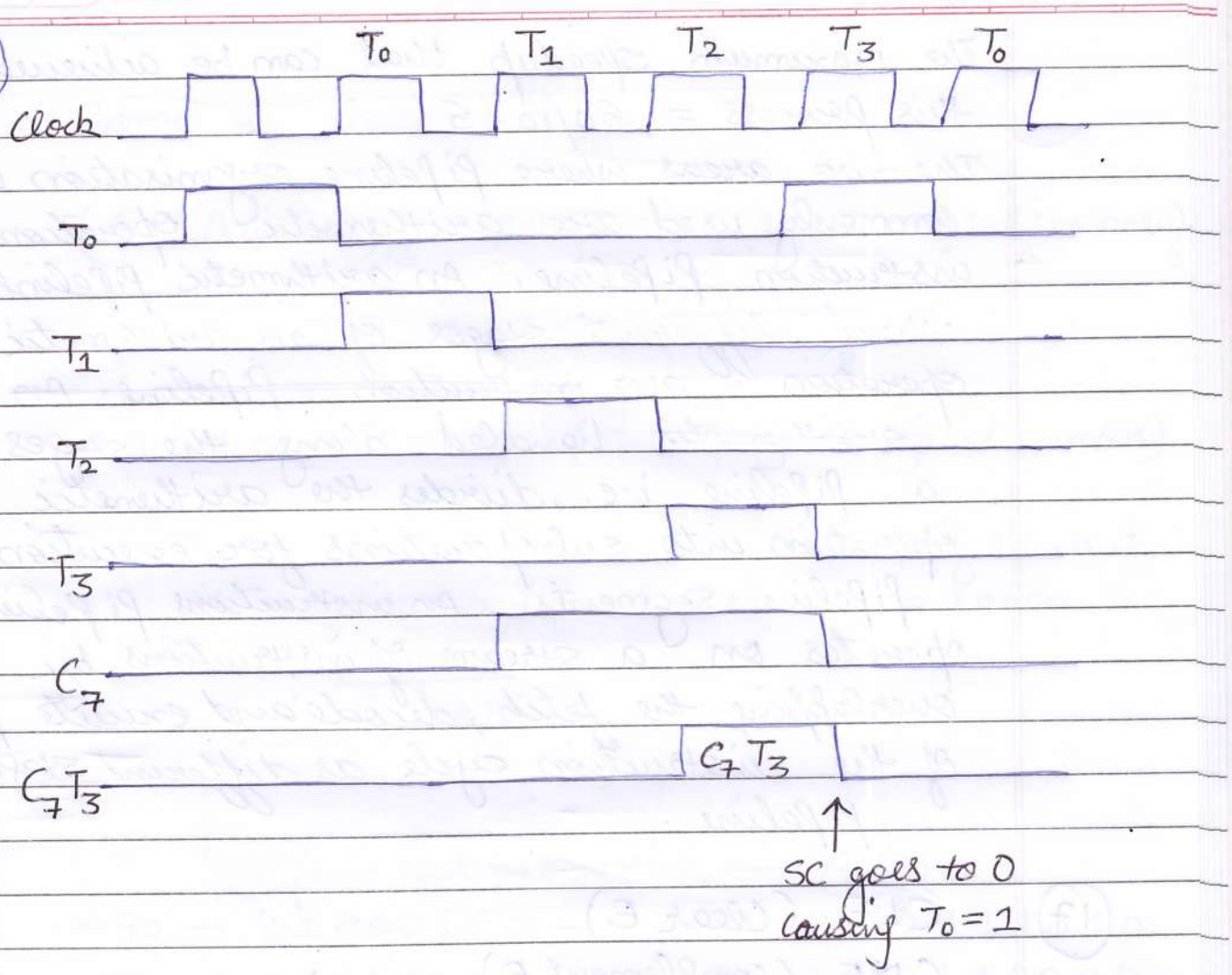
The maximum speedup that can be achieved for this process = 50/10 = 5

The two areas where pipeline organisation is most commonly used are arithmetic operations and instruction pipeline. An arithmetic pipeline where different stages of an arithmetic operation are ~~instruction~~ pipeline. An ~~arithmetic~~ handled along the stages of a pipeline i.e., divides the arithmetic operation into suboperations for execution of pipeline segments. An instructions pipeline operates on a stream of instructions by overlapping the fetch, decode and execute phases of the instruction cycle as different stages of pipeline.

(17) CLE (Clear E)
CME (complement E)

(18) We will replace the memory reference instruction ISZ with [LDC address], which will load the CTR register with the value specified by the address. In addition to that, we will add a new register reference instruction. ICSZ: (increament CTR and skip next instruction if zero) By using the instruction, we don't have to load the memory word into DR, then incrementing DR, and then check it whether it is zero or not, and finally at T6 we will load the value of DR into memory or increment PC. However, by using ICSZ we will execute this instruction at only one clock cycle at T4.

19



20 Effective address

a) Direct: 400
b) Immediate: 301
c) Relative: 301 + 400 = 702
d) Reg. Indirect: 200
e) Indexed: 200 + 400 = 600