INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR

B. TECH 5TH SEMESTER EXAMINATION, DECEMBER 2021

Microprocessor based design (CS 3101)

Name: Abhirup Mukherjee
Examination Roll no.: 510519109
G-Suite ID: 510519109.abhirup@students.iiests.ac.in

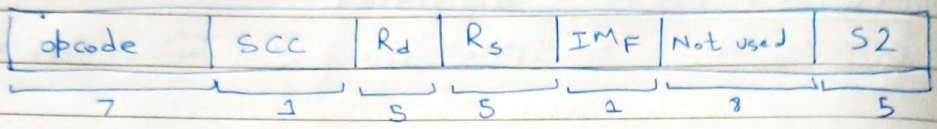
4.5.4.17

Q5) a)

RISC	CISC
→ all instruction takes single machine cycle	→ instructions can take multiple machine cycles.
→ Very Simple instruction set, fixed length instruction	→ Complex instruction set & variable length instruction.
→ Limited Addressing modes	→ Many types of addressing modes present
→ Heavy usage of RAM	→ More efficient usage of RAM
→ needs complex compilers to convert HLL to assembly	→ compiler doesn't need as much complexities with respect to ARM

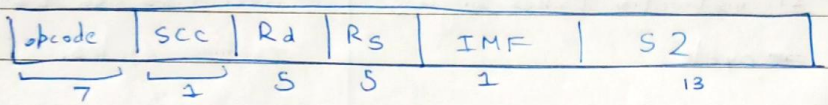
Q5) b) Instruction formats

(i) Register mode.



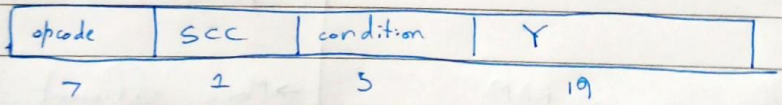
S2 → the register and IMF = 0

(ii) Register Immidiate Mode.



S2 → the immidiate value (operand) & IMF = 1

(iii) PC relative mode.



Y → branch offset

Register Window Scheme

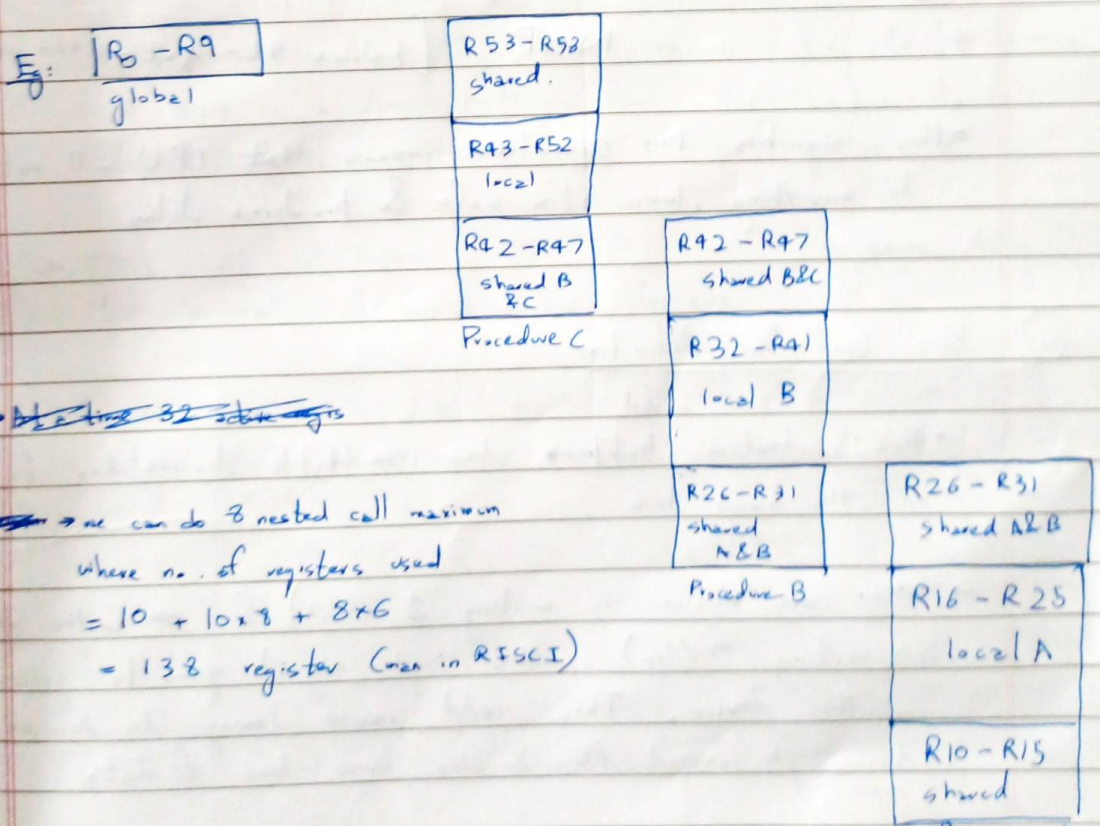
→ reduces parameter passing overhead by sharing ~~some~~ registers between ~~&~~ procedures.

→ RISC I has 138 registers out of which 32 are active at a time.

32 active registers → 10 global registers (always in use)
→ 10 register local to procedure
→ 6 register common to "calle" & "called" function

This is actually 12
as a procedure can be "calle" as well as "caller"

Eg:



~~→ At a time 32 active regs~~

~~→~~ → we can do 8 nested call maximum
where no. of registers used
= 10 + 10×8 + 8×6
= 138 registers (max in RISC I)

Q1) i) XRA A

→ This inst means A ← A XOR A

i.e A ← 0

so here Z flag will be set 1, but no change to sign and carry flag

ie Z → 1

S → 0

C_Y → 0

(ii) An instruction (single) which is safe to use for small delay is

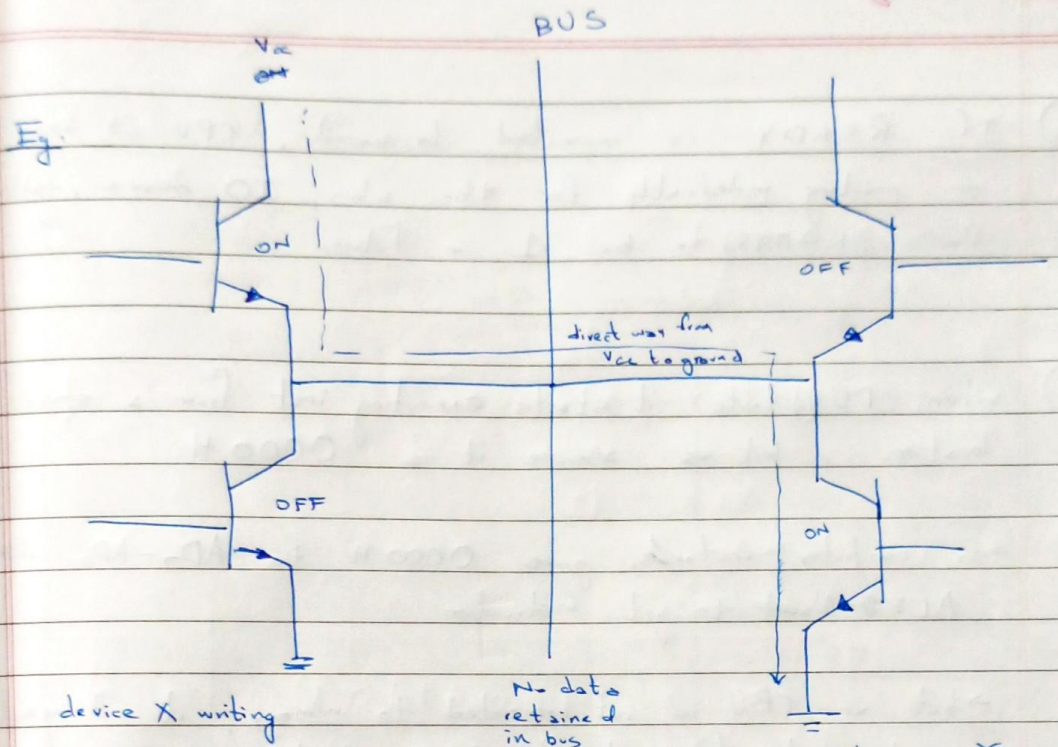
NOP [produces 4t = $\frac{1}{3}$ x 4 = 1.333 μs delay]

→ this signifies No operations, means that CPU will not do any thing, hence it's safe & produces delay

(iii) Bus Contention definition

→ Bus contention could happens when multiple devices try to write on the same bus

→ When one device is writing 1 (high) & another device is writing 0 (low), the high signal goes to ground of another the device. This could cause damage to devices due to high current flow & also cause loss of data



→ when one device is writing 1 & other is writing 0, we see that there is a direct pathway from Vcc to ground, so all signal goes in that way.

→ This will not happen with both device writing same data 0/1 but in reality we cannot control devices

→ So we use CS line to control who give output to CPU at a particular time.

(iv) If READY is connected to ground, CPU will keep on waiting indefinetly for the slow IO device, waiting for READY to be 1 in future.

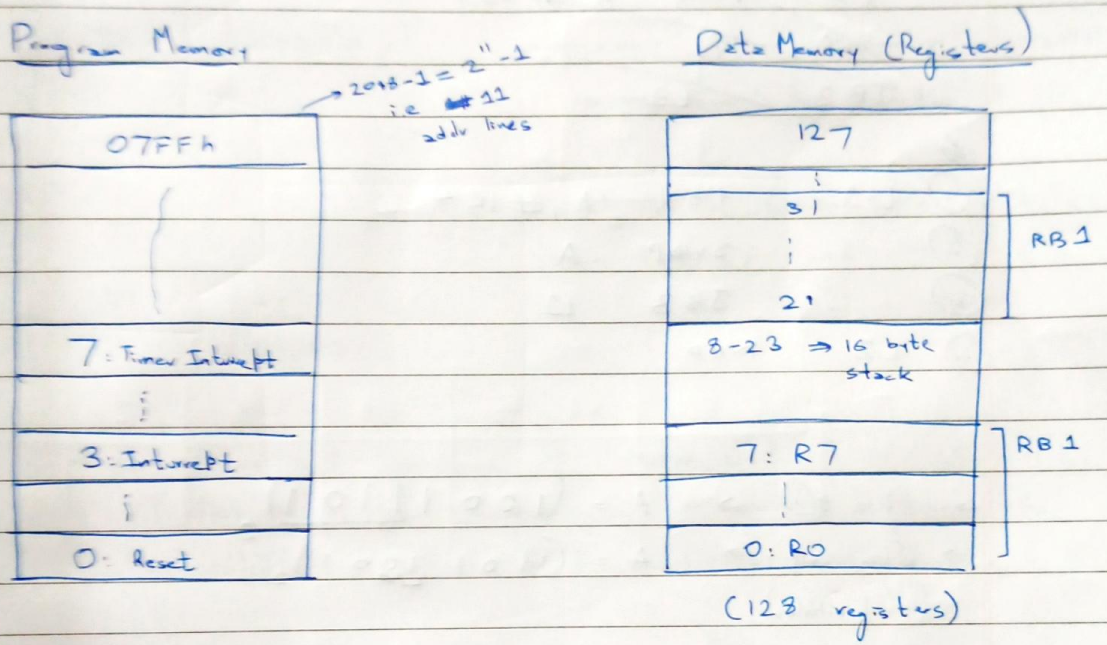
(v) when CPU (re)starts it starts executing inst from a specific location, let us assume it is 0000 H

→ So computer restarts, gives 0000 H to $AD_7 - AD_0$ with ALE = 1 and to get instruction

→ but as CPU is not connected to memory, it will inject wait states to wait for RAM output

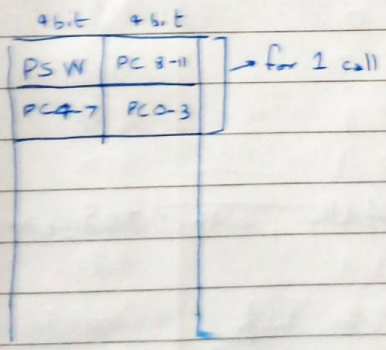
→ This will continue indefinitely as no RAM is connected.

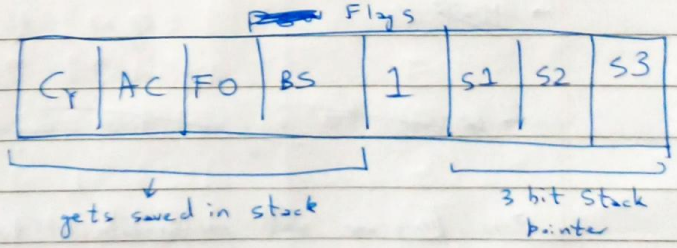
Q6) e) MCS -48 Programming Model



→ only 16 byte stack allowing 8 levels of nesting.

→ every call saves PC bits & some flags in stack





Code

```
L1 : MOV  A, #157
SWA  A
JB3  L1
```

```
(i)   L1 : MOV  A, #157
(ii)       SWAP A
(iii)      JB3  L1
(iv)  L2 : NOP
```

→ in line (i) → $A = (10011101)_2$

→ after swp → $A = (11011001)_2$
(swap nibbles)

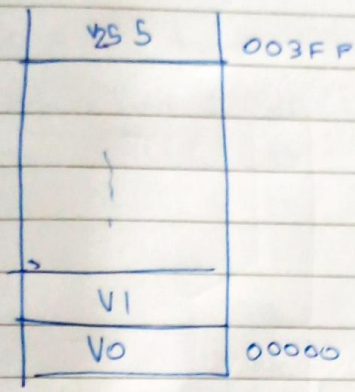
→ JB3 L1 → ~~No jump~~ Jump L1
(jump to L1 if bit 3 is 1)

→ ~~NOP~~

→ ~~So the program runs sequentially, with JB3 L1 failing~~

→ So program will continously run in loop

Q6) b) X86 vector Table



→ memory address 00000 - 003FF is reserved for vector table

→ each vector takes 4 bytes, two for IP and 2 for CS

→ so if we get some V_i addr → we first multiply if by 4 to get corresponding CS_i & IP_i then we calculate actual ISR address using $CS_i \times 16 + IP_i$

Eg INT 23 H

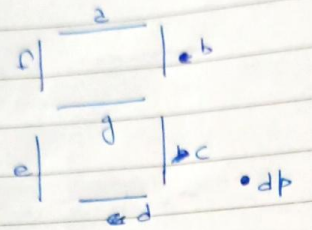
(i) multiply $23 \times 4 \rightarrow 8CH = (140)_{10}$

(ii) now refer V_{140} to get CS_{140} & IP_{140}

(iii) now get ISR address by → $CS_{140} \times 16 + IP_{140}$

(iv) execute the inst. in ISR address.

Q7) consider a 7 segment display as follows.



with given

bit 7	bit 6	bit 5	bit 4	3	2	1	0
a	b	c	dp	d	e	f	g

→ so first let see what each letter will required.

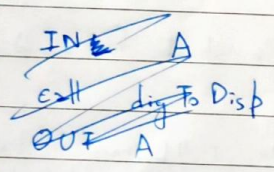
digit	display code in binary	Hex representation
0 (0)	1 1 1 0 1 1 1 0	EF
1 (1)	0 1 1 0 0 0 0 0	60
2 (2)	1 1 0 0 1 1 0 1	CD
3 (3)	1 1 1 0 1 0 0 1	E9
4 (4)	0 1 1 0 0 0 1 1	63
5 (5)	1 0 1 0 1 0 1 1	AB
6 (6)	1 0 1 0 1 1 1 1	AF
7 (7)	1 1 1 0 0 0 0 0	E0
8 (8)	1 1 1 0 1 1 1 1	EF
9 (9)	1 1 1 0 0 0 1 1	E3
A (A)	1 1 1 0 0 1 1 0	E6
B (b)	0 0 1 0 1 1 1 1	2F
C (C)	0 0 0 0 1 1 0 1	0d
D (d)	0 1 1 0 1 1 0 1	6D
E (E)	1 0 0 0 1 1 1 1	8F
F (F)	1 0 0 0 0 1 1 1	87

so I) 8085 code.

```
# ORG   9000
displayTable :   db   0EFH
                 db   60H
                 db   0CDH
                 :
                 db   6DH
                 db   7FH
                 db   67H

# ORG   0000
; convert digit 2 display code (value to be printed is in A)

digToDisp :   push   h      ; saving in stack for avoid loss data
              push   b
              lxi    h, displayTable.  ; HL now in table start
              mvi    b, 0
              mov    c, a   ; BC = A value.
              dad    B      ; HL ← HL + BC ↔ HL now points
                            ; to display code.
              mov    A, M
              POP    b
              POP    h
              RET           ; result in A

# PORT EQU   80
; driver code.
        IN   PORT
        call   digToDisp
        OUT   PORT
```

(II) MCS-48 code.

```
ORG   Page 3
disp Table:   db   0EFH
              db   60H
              ;
              ;
              db   8FH
              db   87H
; value already in A
digToDisp:    movp3  a, @a
              ret

; caller
```



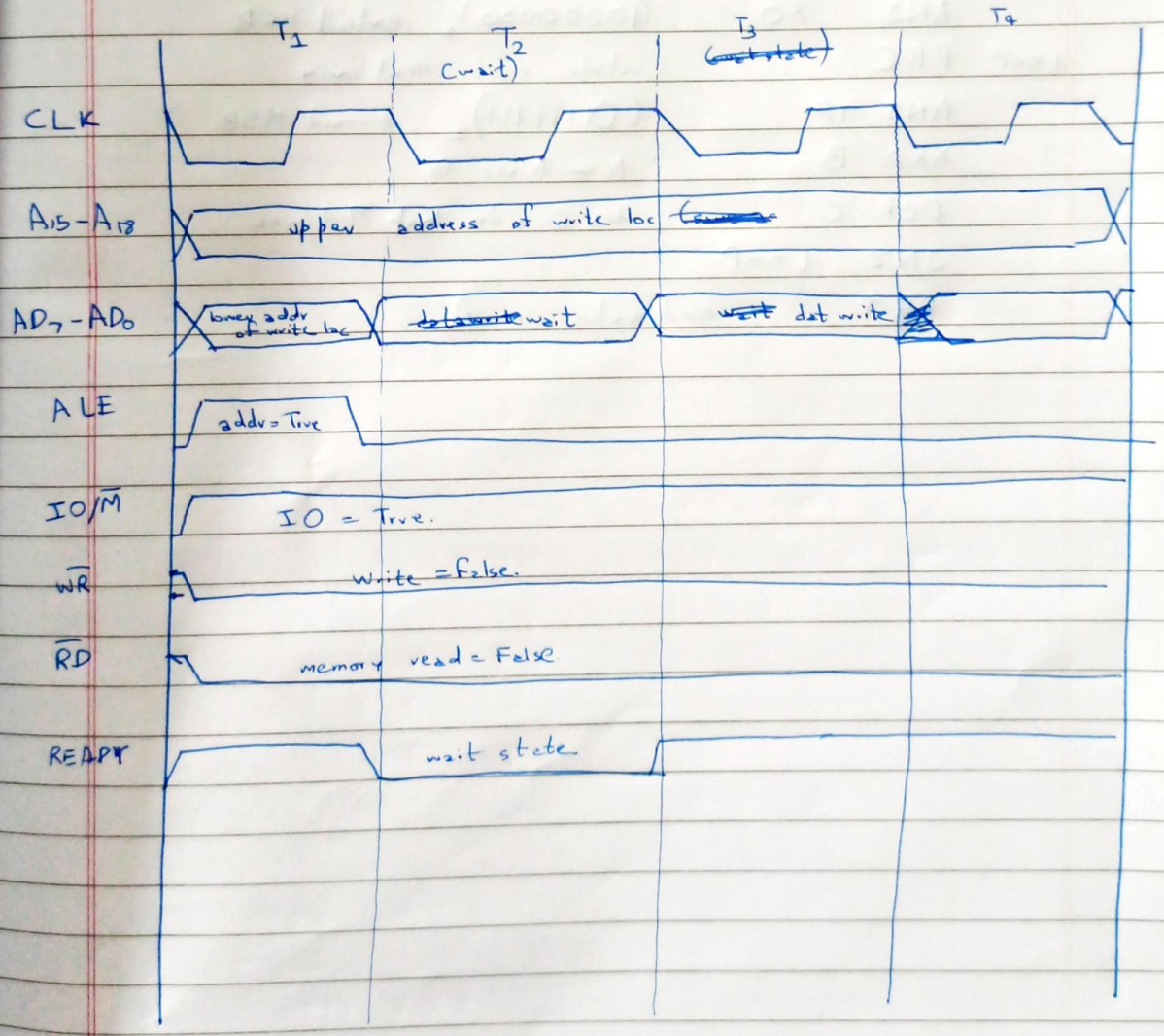

```
# PORT   equ   80H

       IN    PORT
       call  digToDisp
       OUT   PORT
```

Q2) a) Wait State

→ wait states is a state of CPU, where CPU does nothing and waits for slow IO device or memory

Eg: when READY signal is 0, CPU injects wait states in ~~it~~ it's execution to wait for slow IO device to give ready signal

1



→ IO W is similar to Memory write, jus IO/M is 1 instead of WR

Q2) b) 8085 program to convert gray to binary

~~LDA 2000~~

~~#~~ ~~NOT~~ GRAYNUM EQU A5 ; some number.

```
GrayToBn: MVI A, GRAYNUM
          MVI C, 07H
          MOV B, A
          ANI 80      ; (10000000)2 , extract MSB
LOOP:     RRC         ; rotate right without carry
          ANI 7F      ; (01111111)2 , discard MSB
          XRA B       ; A ← A xor B
          DCR C       ; dcr C, do loop 7 times.
          JNZ LOOP
          RET         ; end value in A
```