Indian Institute of Engineering Science and Technology, Shibpur

## B.Tech (CS) 5th Semester Mid-Term Examination, October 2021

Microprocessor Based Design [CS 3101]

[ Answer script header should have the i) Name, Examination Roll No, G-suit-id]

**Name: Abhiroop Mukherjee**

**Enrolment Number: 510519109**

**G-Suite ID: 510519109.abhirup@students.iiests.ac.in**

Answer as many questions as you can. Answer should be brief and to the point. Full

marks is 50. The sum of the total marks exceeding the full marks will be curtailed to 50.

Time allowed is 45 minutes

You may submit machine printed answer script by writing the answers after a question right on this file. For diagrams you may draw them separately and import them at appropriate locations. Don’t forget to import your signature at the end of the answer script.

1. What is a flag? What is a flag register? Explain the use of the flags, namely, Carry and Zero, using

small program segments (2 to 5 lines of appropriate code in assembly).                                       [5]

Ans)

* Flags are a special kind of bit that records the condition of the microprocessor’s calculations.
* Flag Register is a special purpose register which stores all the flag bits.
* Carry Flag (Cy)
  + This flag is set to 1 whenever carry/borrow is generated after addition/subtraction (not internal carry)
  + Else this is 0
  + Eg:
    - MVI A, 0FFH
    - MVI B, 01H
    - ADD B -> A+B will be 100H, this means 00 will remain in A, and Cy will be set 1
* Zero Flag
  + This flag is set 1 if after any Arithmetic operation, result is 0, otherwise this will be 0
  + Example:
    - MVI A, 55H
    - XRA A-> A xor A = 0 -> Z flag set

2. What is bus contention? Explain bus contention with a suitable diagrams.                               [3]

In any uP based design, the CPU is connected to several I/O, memory and other peripheral devices through the buses. The data bus from the CPU is connected to the data bus of the other devices. The CPU address bus (in fact usually the lower order lines) are connected directly to the devices and the higher order lines are use as input to the decoder to generate the chip select signals of the devices. Note that the CPU interact with the devices one at a time. Technically the CPU can write data to multiple devices simultaneously, but the REVERSE is not true. CPU should not try to read multiple devices simultaneously – in that case in a bus line (say bit 0) one device might try writing 1 and another device might write 0. This means that there would be a direct path (low impedance) between the Vcc and Ground through that bus line. This is known as BUS contention and has to be avoided at any cost.

A drawing on a white paper

Description automatically generated with low confidence

3. Draw the timing diagram of the machine cycles (showing the content of the address and data buses in each cycle) needed to execute the following return instruction (OF m/c cycle of RET instruction is 4T). [8]

ORG 3000H

LXI SP, 0A0F0H

SUBX EQU 0F0BH

CALL SUBX

:

SUBX: ; some instructions

RET

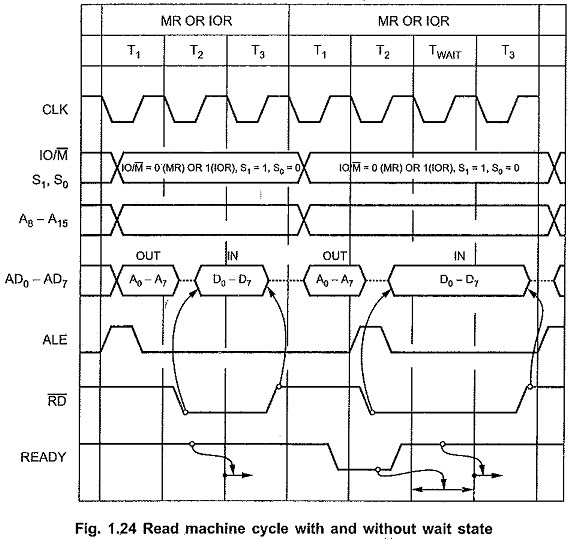
4. What is wait state? Why is it provided? Draw the IF (Instruction fetch) machine cycle with two

wait states.

Ans)

In some applications, speed of memory system and I/O system are not compatible with the microprocessor’s timings. This means that they take longer time to read/write data. in such situations, the microprocessor must confirm whether a peripheral is ready to transfer data or not. If READY pin is high, the peripheral is ready otherwise 8085 enters wait state.

Fig. 1.24 shows the timing diagram for memory read machine cycle with and without wait state.



Concept of Wait States in 8085 continue to be inserted if READY is low. After the wait state, 8085 continues with T3 of the machine cycle. During a wait state the contents of the address bus, the data bus, and the control bus are all held constant.

The wait state then gives an addressed memory or I/O port an extra clock cycle time to output valid data on the data bus. This feature allows to use cheaper memory or I/O devices that have longer access times.

5. Write a subroutine in assembly language that converts an 8-bit binary number to corresponding

Gray code. The subroutine gets the 8-bit binary in Acc register and produces output in Acc. register. Show the calling of this conversion function from the main routine as well.                                [8]

Ans)

STC

CMC

LDA 2050

MOV B,A

RAR

XRA B

STA 3050

HLT

6. Write a subroutine that generates a delay using a 16-bit delay count made available through the

caller calling this subroutine. Also, calculate the maximum delay possible for a clock speed of 3

MHz.

;DELAY: this subroutine produces delay

;in: value in DE pair

DELAY: DCX D

MOV A,E

ORA D

JNZ DELAY

RET

* Maximum delay will occur if DE = FFFF
  + DCX D -> 6t
  + MOV A,E -> 4t
  + ORA D -> 4t
  + JNZ DELAY -> 10t if jump
  + For the last one, no jump -> 7t
  + RET -> 10t
  + So total = (6 + 4 + 4 + 10) \* 0FFFFH – 3 + 10 = (6 + 4 + 4 + 10) \* 65535 – 3 + 10 = 1572847t
  + Now 3MHz t -> 1ms
  + So total time = 1572847 ms

7. Using a standard 3-8 decoder (say 74LS138) where you have 3-inputs, 3 enable lines (active low,

active low, active high) and 8- active low outputs show the decoding to accommodate one 1 x 2K

EPROM, 2 x 2K RAM and 1 input and 1 output port (memory mapped with the same address

A000H). Fold back is allowed.                                                                                                                [8]

So we have been given the following

1. standard 3-8 decoder (74LS138)
2. One 2k x 8 EPROM
3. Two 2k x 8 RAM
4. 2 IO Ports @ address 0A000H (A15 = 1, A13 = 1, all other 0) [Memory Mapped]
   1. As we are doing memory mapped IO, we don’t need to use IO/M’ line

So we can use 3-8 decoder with A15 , A14 , A13 as input line and IO/M’ as enable (If IO/M’ = 0, only then decoder works). Address range of each decoder output will be 64k/23 = 8k, this will bring foldback.

Table

Description automatically generated with medium confidence

8. Explain the assembler directives ORG, EQU, DB, DW and $ with appropriate examples.      [5]

* **DB: Define Byte**
  + **This directive is used for the purpose of allocating and initializing single or multiple data bytes.**
  + **Eg: AREA DB 30H, 52H, 35H**
    - **Memory name AREA has three consecutive locations where 30H, 52H and 35H are to be stored.**



* **ORG**
  + **ORG updates the position from where the Instruction will be written by the Assembler**
  + **It updates the PLC to given data**
  + **Here $ means current value of PLC**
  + **Eg:** ORG $+10H -> next instruction will be written 10 bytes from current PC value
* EQU
  + Define constants
    - Eg: PORTA EQU 30H
* DB
  + Define Byte
    - Used to declare a byte type variable or to store a byte in memory location
    - Eg: myDATA: DB 1,2,3,4,5,6,7,8,9
* DW
  + The DW directive is used to define a variable of type word or to reserve storage location of type word in memory
  + Eg: MULTIPLIER DW 437Ah
    - This declares a variable of type word and named it as MULTIPLIER

Signature: Abhiroop Mukherjee