CS F241 - MICROPROCESSOR PROGRAMMING AND INTERFACING

DESIGN ASSIGNMENT

Group No. : 41

Question No. : 27

Made By-

Ankit Roy	[2016A7PS118P]
Neethu Mariya Joy	[2016A7PS119P]
Nikhil Aggarwal	[2016A7PS120P]
Adit Shastri	[2016A7PS121P]

Problem Statement-

Design a microprocessor based **EPROM Programmer** to program **2716** and **2764**.The EPROM can be programmed by applying 25V at VPP and 5V at OE pin. Initially all data of EPROM will be 1's and the user should make the bits zero selectively. Before the EPROM location is programmed it must be checked for whether it is empty (data in location must be FFH if the location is empty) The 8- bit parallel data is applied to the data pins of EPROM. The address for the EPROM is to be provided. To program the address of each location to be programmed should be stable for 45ms.When address and data are stable, a 40ms active high pulse is applied to CE input. After the EPROM is programmed, IC number is to be displayed on LCD as "27xy programmed".

Assumptions Made-

- Due to limitation of the screen size of the LCD, the outputs will be shown as "27xy PROG".
- We are only using a 12-stage binary counter for convenience. In the case of programming 2764, after 2^12, counter will start again from zero and the circuit will work the same.
- The frequency of clock input is 200Hz and time for 1 clock is 5ms.

- The data on the data lines is FFh initially.

List of components used:

- IC 2716 - 2k EPROM
- IC 2764 - 8k EPROM
- IC 8253 - Programmable interval timer
- IC 8255 - Programmable peripheral interface
- 8086 - Intel x86 microprocessor
- 74HC4040 - 12 stage binary counter
- 74HCT138 - 3:8 decoder
- LM020L - LCD
- 74LS245 - Bidirectional Buffer

Memory Mapping:

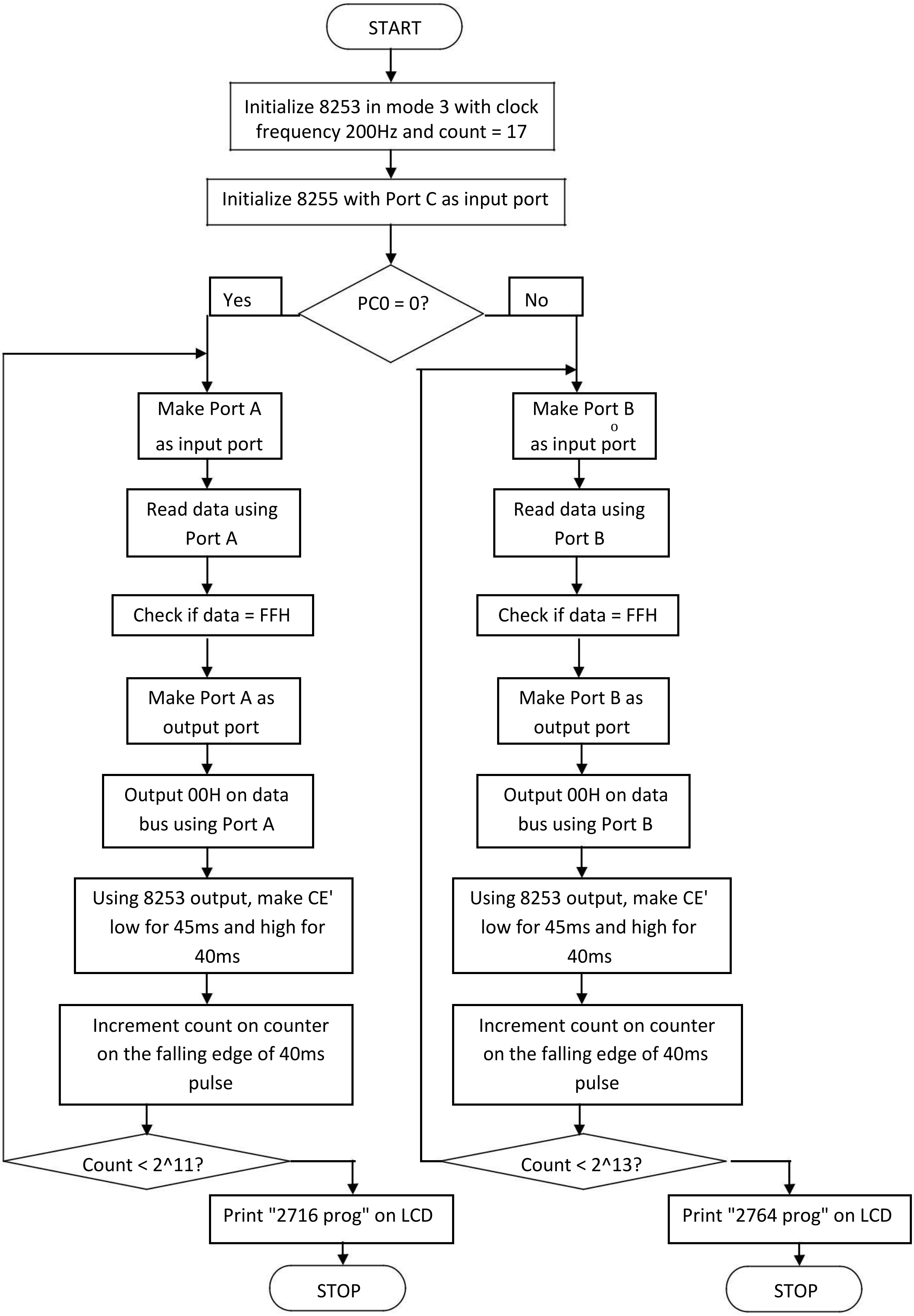
- 2716: 2000H-27FFH
- 2764: 3000H-4FFFH
- 8255: 0010H-0016H(used for interfacing LCD)
- 8255: 0008H-000EH(used for interfacing ROM)
- 8253: 0000H-0006H

Simulation:

- .asm File is created to code the above Problem Statement.
- Then, the coded file is compiled using emu Compiler.
- The generated .com file is simulated in PROTEUS Version 8.1.

NOTE: Since we are using Proteus 8.1, we can’t create .dsn file but, .pdsprj format from Proteus. So, it is requested to view our files in Proteus 8.1.

Flowchart of the software-


START
Initialize 8253 in mode 3 with clock frequency 200Hz and count = 17
Initialize 8255 with Port C as input port
PC0 = 0?
Yes
No
Make Port A as input port
Make Port B as input port
o
Read data using Port A
Read data using Port B
Check if data = FFH
Check if data = FFH
Make Port A as output port
Make Port B as output port
Output 00H on data bus using Port A
Output 00H on data bus using Port B
Using 8253 output, make CE' low for 45ms and high for 40ms
Using 8253 output, make CE' low for 45ms and high for 40ms
Increment count on counter on the falling edge of 40ms pulse
Increment count on counter on the falling edge of 40ms pulse
Count < 2^11?
Count < 2^13?
Print "2716 prog" on LCD
Print "2764 prog" on LCD
STOP
STOP

Assembly Language Code for the project-

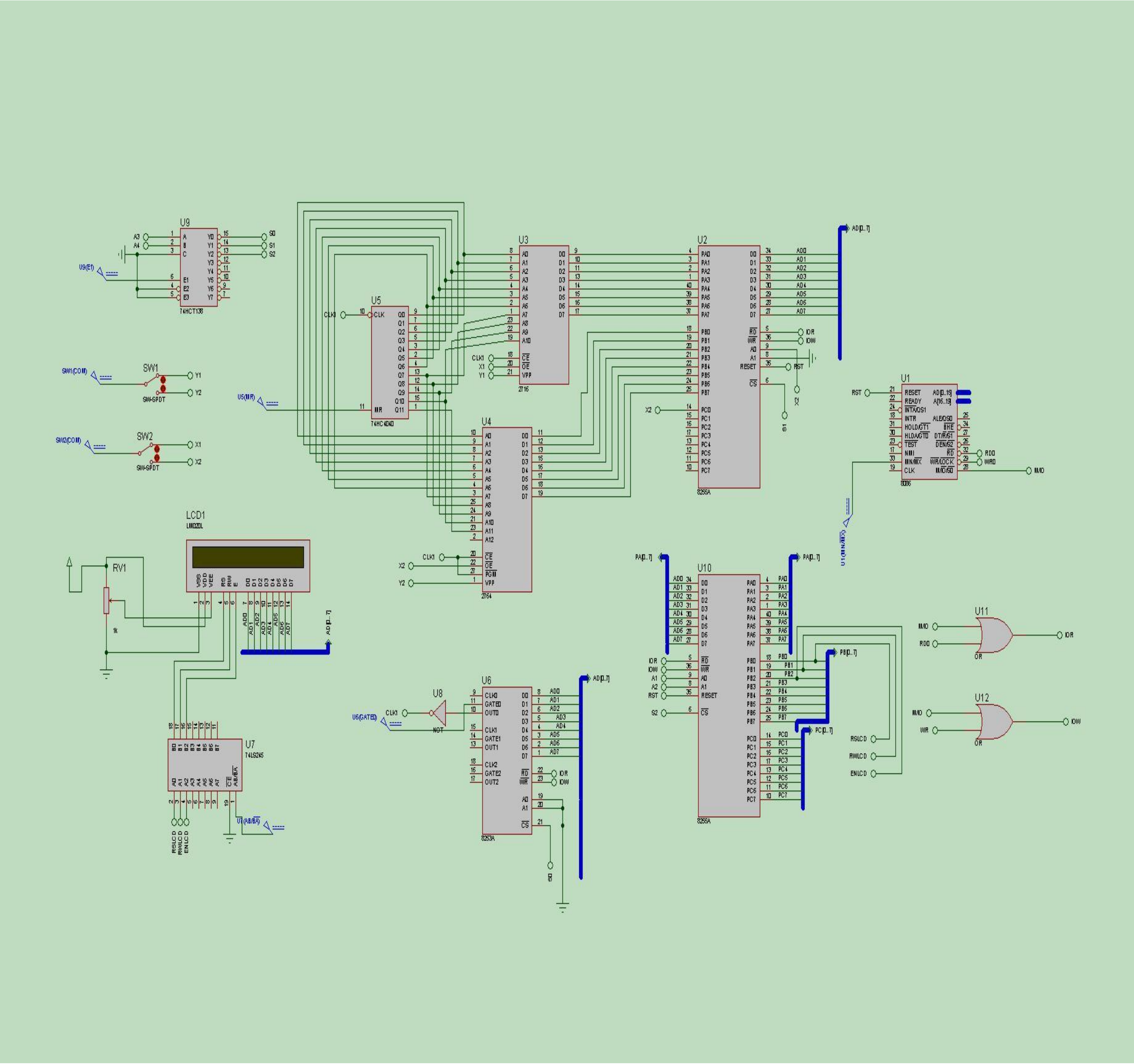
```
.model tiny

;8255 for data transfer
cr EQU 0eh ;control register
pa EQU 08h
pb EQU 0ah
pc EQU 0ch

;8255 for LCD
cr1 EQU 16h ;control register
prta EQU 10h
prtb EQU 12h
prtc EQU 14h

;8253
CR2 EQU 06h ;control register
cnt0 EQU 00h
cnt1 EQU 02h
cnt2 EQU 04h


.code
.startup


;initialising 8253
;Set to mode 3
;Counter 0 set to 17 to get 9 low pulses
;and 8 low pulses of 40 millisecs each

MOV AL, 00110110b
OUT CR2, AL
MOV AL, 11h
OUT cnt0, AL
MOV AL, 00h
OUT cnt0, AL

MOV CX,0


; for 8255 1st which we use for data transaction between processor and ROM
MOV AL, 10001001b
OUT cr, AL

IN AL, pc
AND AL, 00000001B;Here we check whether C0 is set to 1 which indicates
                                        ;which ROM is being programmed
CMP AL, 00h                             ;If C0 is zero,ROM1 is being programmed
JZ rom1

rom2:

MOV AL, 10000010b
OUT cr, AL                  ;control register programmed
loop1: IN AL, pb
                CMP AL,0
                JE loop1    ; to ensure program stops
                            ;till address becomes stable

CMP AL, 0ffh                ;compare to see whether the location is empty i.e. all 1's
JZ x1

;  Nothing done if location doesn't have FFh



x1: MOV AL, 80h
OUT cr, AL
MOV AL, 00h
OUT pb, AL
INC CX

;compare count with maxcount so that the loop can be exited if all the locations have been accessed
CMP CX,07ffh
JNZ rom2
JZ lr2
```

```
rom1:

MOV AL, 10010000b
OUT cr, AL
loop2 : IN AL,pa
                                CMP AL,0
                                JE loop2                        ; to ensure program stops
                                                                ;till address becomes stable

CMP AL, 0ffh                                                    ;compare to see whether the location is empty i.e. all 1's
JZ x2

;  Nothing done if location doesn't have FFh



x2: MOV AL, 80h
OUT cr, AL
MOV AL, 00h
OUT pa, AL
INC CX

;compare count with maxcount so that the loop can be exited if all the locations have been accessed
CMP CX,1FFFh

JNZ rom1
JZ lr1



lr1:



                ;writing on the command register for initialization

                CALL BEG_LCD ;calling lcd initialization
                CALL RITE_2716
                JMP lastcode

RITE_2716 PROC NEAR
                CALL CLS
                MOV AL, '2' ;display 2
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait before giving next character
                MOV AL, '7' ;display 7
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait before giving next character
                MOV AL, '1' ;display 1
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait
                MOV AL, '6' ;display 6
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait
                MOV AL, ' ' ;display space
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait
                MOV AL, 'P' ;display P
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait
                MOV AL, 'R' ;display R
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait
                MOV AL, 'O' ;display O
                CALL RITEDATA ;give to LCD
                CALL DLAY ;wait before giving next character
                CALL DLAY ;wait
                MOV AL, 'G' ;display G
```

```
		CALL RITEDATA ;give to LCD
		CALL DLAY ;wait before giving next character
		CALL DLAY ;wait
		RET
RITE_2716 ENDP


lr2:



		;writing on the command register for initialization

		CALL BEG_LCD ;calling lcd initialization
		CALL RITE_2764
		JMP  lastcode


BEG_LCD PROC NEAR
		MOV AL, 38H ;initialize LCD
		CALL CORITE ;write the command to LCD
		CALL DLAY ;wait before giving next command
		CALL DLAY ;
		CALL DLAY
		MOV AL, 0EH ;send command for LCD on, cursor on
		CALL CORITE
		CALL DLAY
		MOV AL, 01  ;clear LCD
		CALL CORITE
		CALL DLAY
		MOV AL, 06  ;command for shifting cursor right
		CALL CORITE
		CALL DLAY
		RET
BEG_LCD ENDP

CLS PROC
		MOV AL, 01  ;clear LCD
		CALL CORITE
		CALL DLAY
		CALL DLAY
		RET
CLS ENDP

CORITE PROC ;this procedure writes commands to LCD
		MOV DX, prtA
		OUT DX, AL  ;send the code to prt A
		MOV DX, prtB
		MOV AL, 00000100B ;RS=0,R/W=0,E=1 for H-To-L pulse
		OUT DX, AL
		NOP
		NOP
		MOV AL, 00000000B ;RS=0,R/W=0,E=0 for H-To-L pulse
		OUT DX, AL
		RET
CORITE ENDP

RITE_2764 PROC NEAR
		CALL CLS
		MOV AL, '2' ;display 2
		CALL RITEDATA ;give to LCD
		CALL DLAY ;wait before giving next character
		CALL DLAY ;wait before giving next character
		MOV AL, '7' ;display 7
		CALL RITEDATA ;give to LCD
		CALL DLAY ;wait before giving next character
		CALL DLAY ;wait before giving next character
		MOV AL, '6' ;display 6
		CALL RITEDATA ;give to LCD
		CALL DLAY ;wait before giving next character
		CALL DLAY ;wait
		MOV AL, '4' ;display 4
		CALL RITEDATA ;give to LCD
		CALL DLAY ;wait before giving next character
		CALL DLAY ;wait
		MOV AL, ' ' ;display space
		CALL RITEDATA ;give to LCD
		CALL DLAY ;wait before giving next character
		CALL DLAY ;wait
```

```
        MOV AL, 'P' ;display P
        CALL RITEDATA ;give to LCD
        CALL DLAY ;wait before giving next character
        CALL DLAY ;wait
        MOV AL, 'R' ;display R
        CALL RITEDATA ;give to LCD
        CALL DLAY ;wait before giving next character
        CALL DLAY ;wait
        MOV AL, 'O' ;display O
        CALL RITEDATA ;give to LCD
        CALL DLAY ;wait before giving next character
        CALL DLAY ;wait
        MOV AL, 'G' ;display G
        CALL RITEDATA ;give to LCD
        CALL DLAY ;wait before giving next character
        CALL DLAY ;wait
        RET
RITE_2764 ENDP

RITEDATA PROC
        PUSH DX  ;save DX
        MOV DX,prtA  ;DX=prt A address
        OUT DX, AL ;issue the char to LCD
        MOV AL, 00000101B ;RS=1, R/W=0, E=1 for H-to-L pulse
        MOV DX, prtB ;prt B address
        OUT DX, AL  ;make enable high
        MOV AL, 00000001B ;RS=1,R/W=0 and E=0 for H-to-L pulse
        OUT DX, AL
        POP DX
        RET
RITEDATA ENDP ;writing on the lcd ends

;delay in the circuit here the delay of 20 millisecond is produced
DLAY PROC
        MOV CX, 1325 ;1325*15.085 microsec = 20 msec
        W1:
                NOP
                NOP
                NOP
                NOP
                NOP
        LOOP W1
        RET
DLAY ENDP

lastcode: NOP

.exit
END
```

Circuit Diagram-



____END____

REFERENCES:

- **Datasheet of LCD(LM020L)--** **www.datasheetspdf.com/datasheet/LM020L.html**