CS F241 - MICROPROCESSOR PROGRAMMING AND INTERFACING (DESIGN PROJECT)
GROUP NUMBER 5
PROJECT NUMBER 3
(EPROM PROGRAMMER)

Team Members: Indraneel Ghosh(2016B1A70938P)

Kumar Deovrata(2016B1A70939P)

Avish Khosla(2016B1A80324P)

Sai Harsha Kantamaneni(2016B1A80238P)

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 using a 12-stage binary counter.
- While programming 2764, after 2^12, the counter will start again from zero and the circuit will work the same.
- Frequency of clock input is set to 200Hz and time for clock is 5 millseconds.
- Initial data on data lines: FFh.

Details of the components used:

- IC 2764 - 8k EPROM
- IC 2716 - 2k EPROM
- IC 8253 - Programmable Interval Timer
- IC 8255 - Programmable Peripheral Interface
- 74HCT138 - 3:8 decoder
- 74HC4040 - 12 stage binary counter
- 74LS245 - Bidirectional Buffer -
- LM020L - LCD
- 8086 - Intel x86 microprocessor
- SW-SPDT

Memory Mapping:

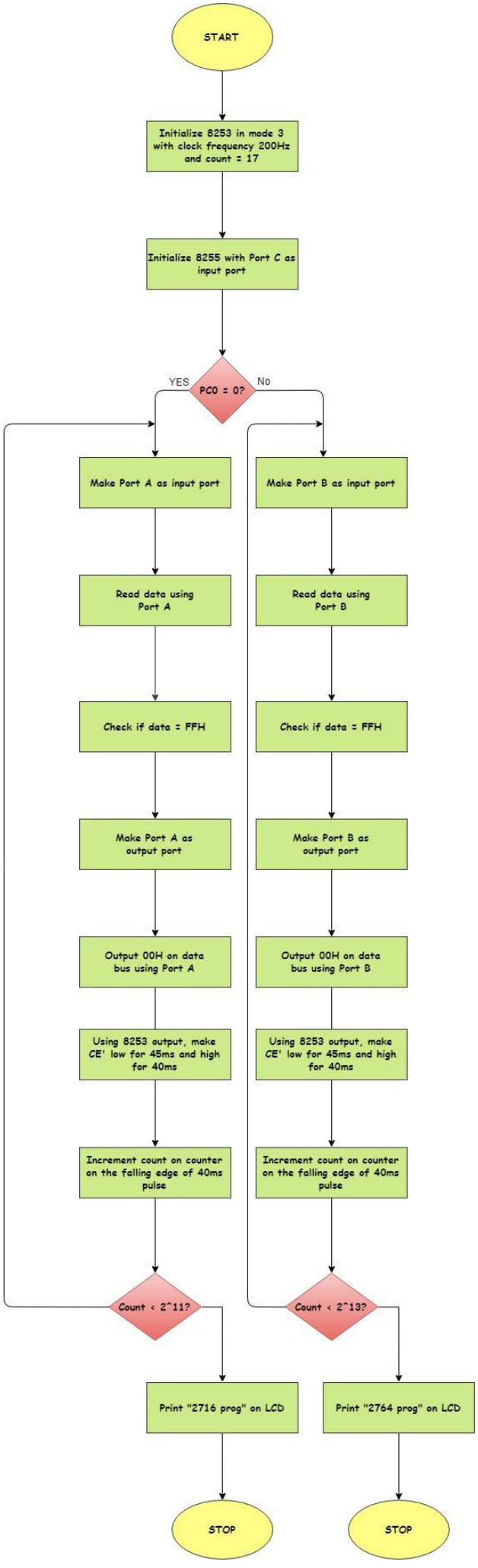
- 2716: 2000H-27FFH
- 2764: 3000H-4FFFH
- 8255: 0010H-0016H(used for interfacing LCD)
- 8255: 0008H-000EH(used for interfacing ROM)
- 8253: 0000H-0006H

Simulation Details

1. The code for the problem statement is written into .asm file.
2. Then, the asm file is compiled using emu Compiler.
3. The generated .com file is simulated in PROTEUS Version 8.1.

NOTE: As we were using Proteus 8.1 we were unable to create the .dsn file. We used the .pdsprj format from Proteus. Hence, to run the code please view the submission in Proteus 8.1.

Flowchart for Algorithm used in the Design(Next Page)


START
Initialize 8253 in mode 3 with clock frequency 200Hz and count = 17
Initialize 8255 with Port C as input port
YES
PC0 = 0?
No
Make Port A as input port
Make Port B as input port
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
porta equ 08h
portb equ 0ah
portc equ 0ch
cr equ 0eh ;control register

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

;8253
cr2 equ 06h ;control register
count0 equ 00h
count1 equ 02h
count2 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 count0, al
mov al, 00h
out count0, 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, portc
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, portb
        cmp al,0
        je loop1        ; to ensure program stops
                                        ;till address becomes stable

cmp al, 0ffh            ;comportare 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 portb, al
        inc cx
        cmp cx,07ffh
jnz rom2
jz lr2

rom1:
mov al, 10010000b
out cr, al

loop2 :
                in al,porta
                cmp al,0
                je loop2                ; to ensure program stops
                                        ;till address becomes stable

                cmp al, 0ffh            ;comportare 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 porta, al
        inc cx
        ;comportare 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 WRITE_2716
        JMP lastcode

lr2:
        ;writing on the command register for initialization
        CALL BEG_LCD ;calling lcd initialization
        CALL WRITE_2764
        JMP  lastcode

WRITE_2716 PROC NEAR
        PUSHA
        PUSHF
        CALL CLS
        MOV AL, '2' ;display 2
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait before giving next character
        MOV AL, '7' ;display 7
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait before giving next character
        MOV AL, '1' ;display 1
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
```

```
        MOV AL, '6' ;display 6
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, ' ' ;display sportace
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'P' ;display P
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'R' ;display R
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'O' ;display O
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'G' ;display G
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        POPF
        POPA
        RET
WRITE_2716 ENDP
```

```
BEG_LCD PROC NEAR
        PUSHA
        PUSHF
        MOV AL, 38H ;initialize LCD
        CALL COWRITE ;write the command to LCD
        CALL DELAY ;wait before giving next command
        CALL DELAY ;
        CALL DELAY
        MOV AL, 0EH ;send command for LCD on, cursor on
        CALL COWRITE
        CALL DELAY
        MOV AL, 01  ;clear LCD
        CALL COWRITE
        CALL DELAY
        MOV AL, 06  ;command for shifting cursor right
        CALL COWRITE
        CALL DELAY
        POPF
        POPA
        RET
BEG_LCD ENDP

CLS PROC
        PUSHA
        PUSHF
        MOV AL, 01  ;clear LCD
        CALL COWRITE
        CALL DELAY
        CALL DELAY
        POPF
        POPA
```

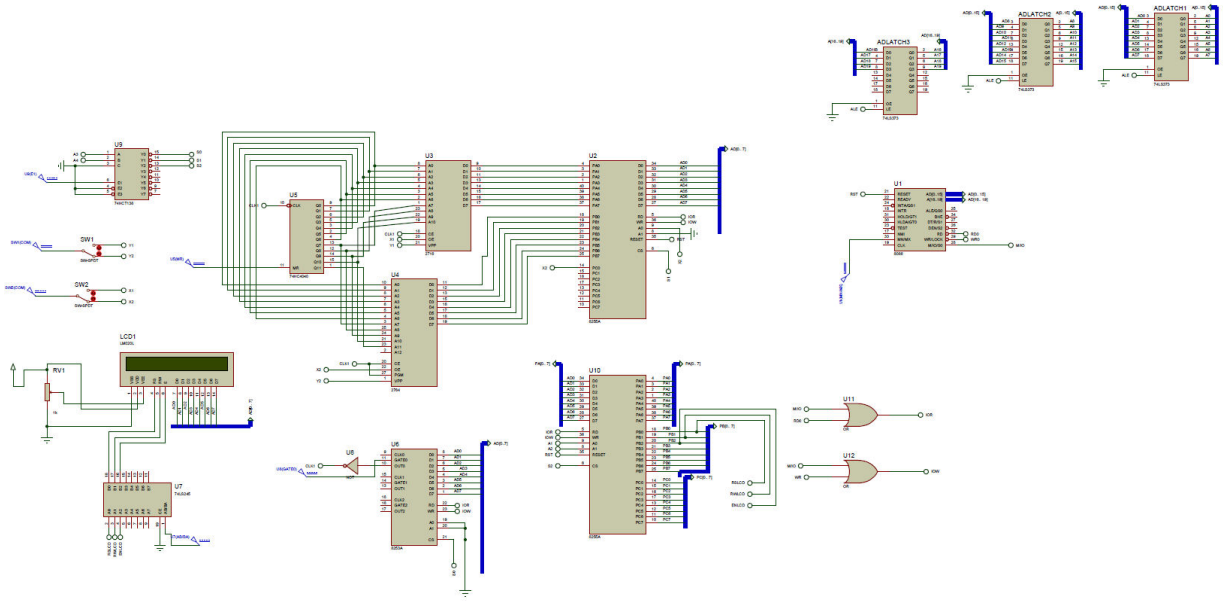
```
        RET
CLS ENDP

COWRITE PROC ;this procedure writes commands to LCD
        pusha
        pushf
        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
        MOV AL, 00000000B ;RS=0,R/W=0,E=0 for H-To-L pulse
        OUT DX, AL
        popf
        popa
        RET
COWRITE ENDP

WRITE_2764 PROC NEAR
        pusha
        pushf
        CALL CLS
        MOV AL, '2' ;display 2
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait before giving next character
        MOV AL, '7' ;display 7
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait before giving next character
        MOV AL, '6' ;display 6
```

```
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, '4' ;display 4
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, ' ' ;display sportace
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'P' ;display P
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'R' ;display R
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'O' ;display O
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        MOV AL, 'G' ;display G
        CALL WRITEDATA ;give output to LCD
        CALL DELAY ;wait before giving next character
        CALL DELAY ;wait
        popf
        popa
        RET
WRITE_2764 ENDP
```

```
WRITEDATA PROC
        PUSHA
        PUSHF
        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
        POPF
        POPA
        RET
WRITEDATA ENDP ;writing on the lcd ends


;delay in the circuit here the delay of 20 millisecond is produced
DELAY PROC
        pusha
        PUSHF
        MOV CX, 1325 ;1325*15.085 usec = 20 msec
        W1:
                NOP
                NOP
                NOP
                NOP
                NOP
        LOOP W1
```

```
        POPF
        popa
        RET
DELAY ENDP

;using BIOS delay function
DLAY PROC
pusha
PUSHF
MOV     CX, 00H
MOV     DX, 9C40H
MOV     AH, 86H
INT     15H
POPF
popa
DLAY ENDP

lastcode:
.exit
END
```

Circuit Design:



(A separate PDF file for the design diagram has been attached in the zip folder. Please refer to that image for more clarity.)

References:

- Datasheet of LCD(LM020L)-- www.datasheetspdf.com/datasheet/LM020L.html
- Barry Brey
- Lecture Slides