CS F241 - MICROPROCESSOR PROGRAMMING AND INTERFACING

DESIGN ASSIGNMENT

2016B2A30927P MEGHA PALIWAL
2016B2A70721P POOJA TULSYAN
2016B2A70770P HONNESH ROHMETRA
2016B2A70773P ISHAN SHARMA

Group No: 11

Question No: 3

Problem Statement no 3

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".

Problem Description

The problem is to program 2716 and 2764 EPROM chips.
The microprocessor should sequentially access all the memory locations of 2716 and 2764, and write data in all memory locations. If the memory location is not erased then it needs to be erased first. ROM and RAM should be interfaced with 8086. The system bus of the microprocessor should not be directly interfaced to 2716 and 2764. Therefore, PPI 8255 IC must be used. The address to be accessed should be passed via some port of 8255 to 2716 and 2764, and the data to be written should be given as an input to both chips via some other port of 8255.
LCD is further connected to 74LS245 to display the result “27xy PROG”

Assumptions

- Initial data on data lines = FFh.
- Using only a 12-stage binary counter for convenience of design space on Proteus. In the case of programming 2764, after 2^12, counter will start again from zero and the circuit will work the same. Proper counters are shown in the chart design.
- Internal CLK of 8086 is used to provide basic timing signal to 8086.
- The data to be written in 2716 and 2764 is provided by the programmer in the ASM file itself.
- Clock frequency = 200Hz.

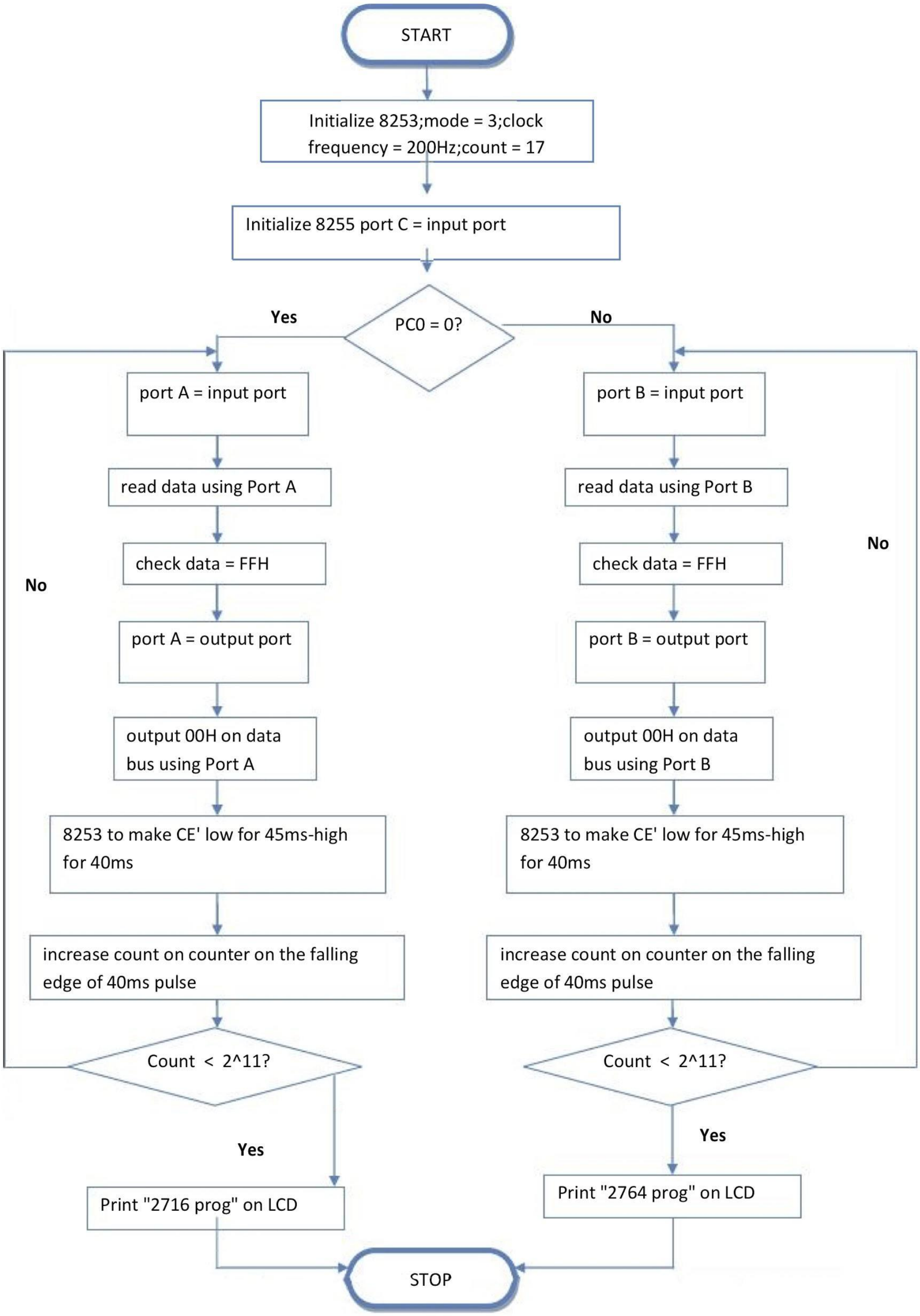
Components used

- IC 2716 - 2k EPROM
- IC 2732 - 4k EPROM
- IC 2764 - 8k EPROM
- IC 6116 – 2k RAM Chip
- 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
- 74LS373- Octal Latch

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

Flowchart


START
Initialize 8253;mode = 3;clock frequency = 200Hz;count = 17
Initialize 8255 port C = input port
PC0 = 0?
Yes
No
port A = input port
read data using Port A
check data = FFH
port A = output port
output 00H on data bus using Port A
8253 to make CE' low for 45ms-high for 40ms
increase count on counter on the falling edge of 40ms pulse
Count < 2^11?
No
Yes
Print "2716 prog" on LCD
port B = input port
read data using Port B
check data = FFH
port B = output port
output 00H on data bus using Port B
8253 to make CE' low for 45ms-high for 40ms
increase count on counter on the falling edge of 40ms pulse
Count < 2^11?
No
Yes
Print "2764 prog" on LCD
STOP

Assembly Language Code for the project:

```
.model tiny

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

;8255 for LCD
creg1 equ 16h ;control register
porta equ 10h
portb equ 12h
portc equ 14h

;8253
creg2 equ 06h ;control register
count0 equ 00h
count1 equ 02h
count2 equ 04h


.code
.startup



;initialising 8253
;Here we set the mode equal to 3
;The count of counter 0  is set to 17 to get 9 low pulses
```

```
;and 8 low pulses of 40 millisecs each


start1: mov al, 00110110b
out creg2, 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 lcd

mov al,10000000b
out creg1,al




; for 8255 1st which we use for data transaction between processor
and ROM
mov al, 10001001b
out creg, 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 creg, aL            ;control register programmed
loop1: in al, pb
            cmp al,0
            je loop1    ;this loop ensures that program doesnt proceed
forward
                                    ;when address stablisation in being done

cmp al, 0ffh            ;comparision to see whether the location is
empty i.e. all 1's
jz x1

;There is nothing specified in the problem on what to do if the
location content
;is not found to be FFh. So we have left the space as it is.
;Although some minor operation like glowing a LED can be done.

x1: mov al, 80h
out creg, 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,1fffh
jnz rom2
```

```
jz lcdrom2




rom1:

mov al, 10010000b
out creg, al
loop2 : in al,pa
              cmp al,0
              je loop2          ;this loop ensures that program doesnt
proceed forward
                                        ;when address stablisation in being
done

cmp al, 0ffh                    ;comparision to see whether the location is
empty i.e. all 1's
jz x2

;There is nothing specified in the problem on what to do if the
location content
;is not found to be FFh. So we have left the space as it is.
;Although some minor operation like glowing a LED can be done.

x2: mov al, 80h
out creg, 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,07FFh

jnz rom1
jz lcdrom1



lcdrom1:

; initialise hardware
        ; initialise the lcd
        ; check for busy status
        ; clear the screen
        ; display 'empty'
        ;call init_motor


        ;writing on the command register for initialization

        CALL LCD_INIT ;calling lcd initialization
        CALL WRITE_2716
        JMP lastcode

WRITE_2716 PROC NEAR
        CALL CLS
        MOV AL, '2' ;display ‘2’ letter
        CALL DATWRIT ;issue it to LCD
        CALL DELAY ;wait before issuing the next character
        CALL DELAY ;wait before issuing the next character
```

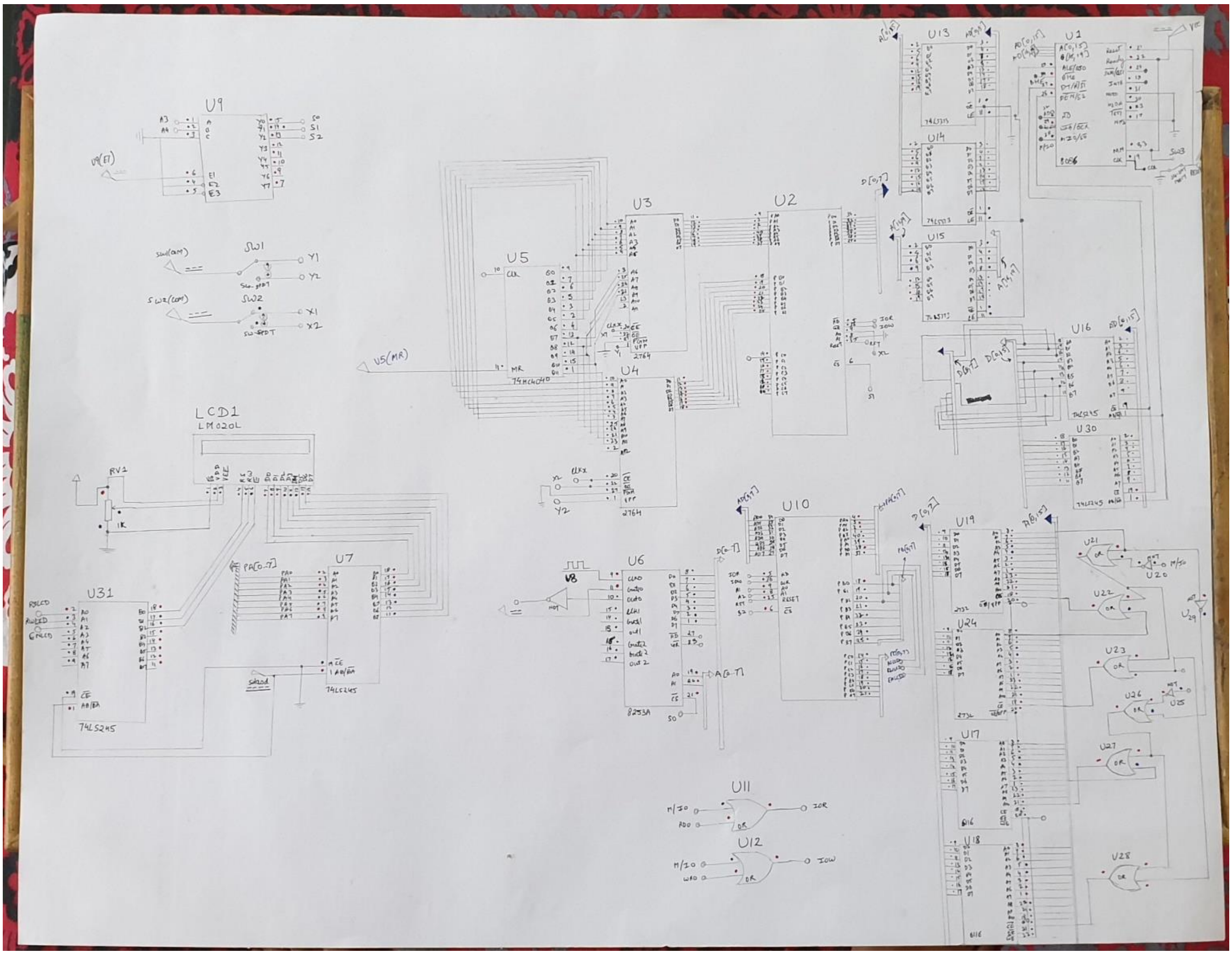
```
MOV AL, '7' ;display ‘7’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait before issuing the next character
MOV AL, '1' ;display ‘1’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, '6' ;display ‘6’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, ' ' ;display ‘ ’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, 'P' ;display ‘P’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, 'R' ;display ‘R’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, 'O' ;display ‘O’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, 'G' ;display ‘G’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
```

```
        CALL DELAY ;wait
        RET
WRITE_2716 ENDP


lcdrom2:
; initialise hardware
        ; initialise the lcd
        ; check for busy status
        ; clear the screen
        ; display 'empty'
        ;call init_motor


        ;writing on the command register for initialization

        CALL LCD_INIT ;calling lcd initialization
        CALL WRITE_2764
        JMP  lastcode


LCD_INIT PROC NEAR
        MOV AL, 38H ;initialize LCD for 2 lines & 5*7 matrix
        CALL COMNDWRT ;write the command to LCD
        CALL DELAY ;wait before issuing the next command
        CALL DELAY ;this command needs lots of delay
        CALL DELAY
        MOV AL, 0EH ;send command for LCD on, cursor on
        CALL COMNDWRT
        CALL DELAY
        MOV AL, 01  ;clear LCD
        CALL COMNDWRT
        CALL DELAY
        MOV AL, 06  ;command for shifting cursor right
        CALL COMNDWRT
        CALL DELAY
        RET
LCD_INIT ENDP

CLS PROC
        MOV AL, 01  ;clear LCD
        CALL COMNDWRT
        CALL DELAY
        CALL DELAY
        RET
CLS ENDP

COMNDWRT PROC ;this procedure writes commands to LCD
        MOV DX, PORTA
        OUT DX, AL  ;send the code to Port A
        MOV DX, PORTB
        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
COMNDWRT ENDP

WRITE_2764 PROC NEAR
        CALL CLS
```

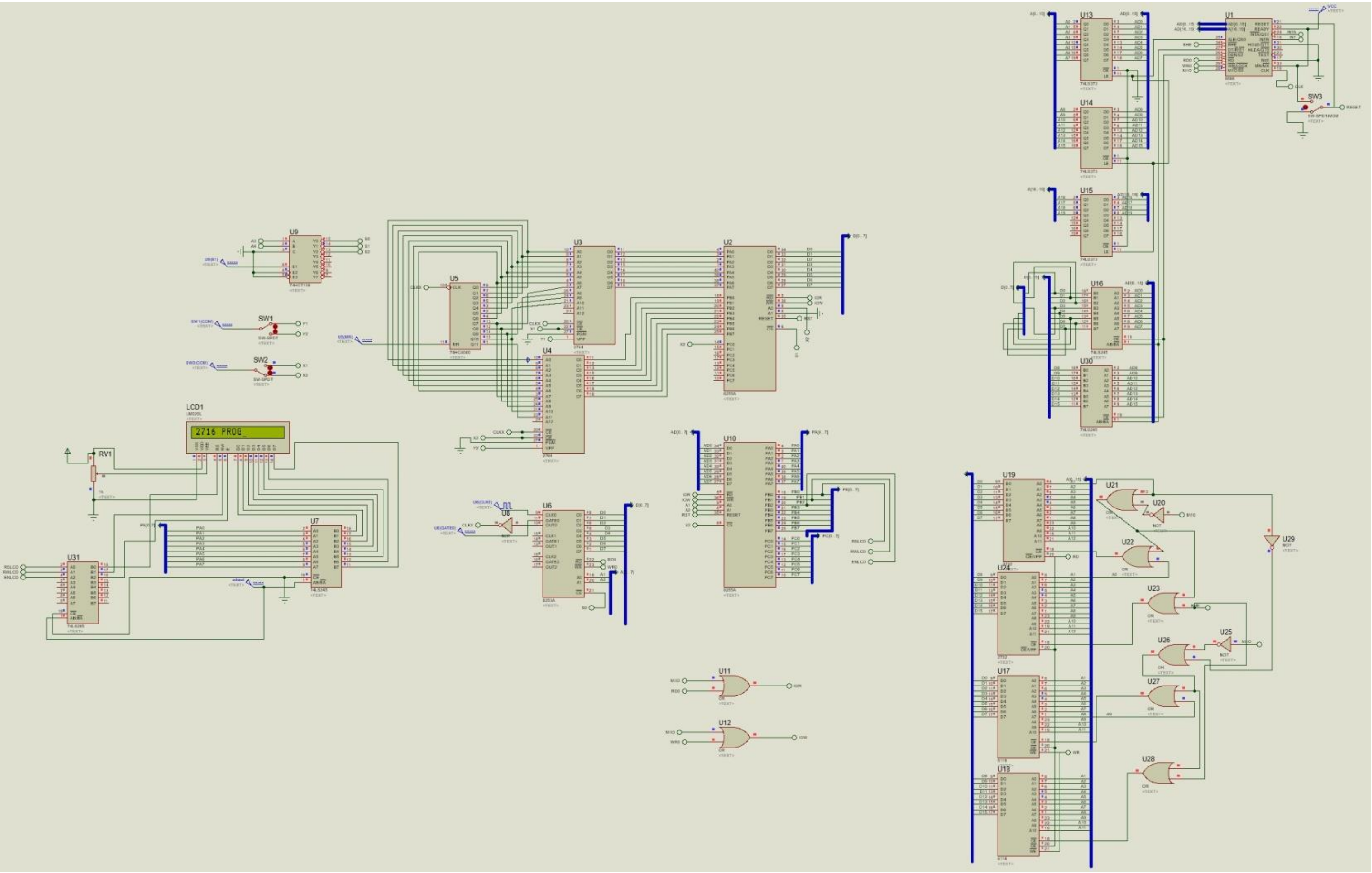
```
MOV AL, '2' ;display ‘2’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait before issuing the next character
MOV AL, '7' ;display ‘7’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait before issuing the next character
MOV AL, '1' ;display ‘6’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, '6' ;display ‘4’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, ' ' ;display ‘ ’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, 'P' ;display ‘P’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, 'R' ;display ‘R’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
CALL DELAY ;wait
MOV AL, 'O' ;display ‘O’ letter
CALL DATWRIT ;issue it to LCD
CALL DELAY ;wait before issuing the next character
```

```
        CALL DELAY ;wait
        MOV AL, 'G' ;display ‘G’ letter
        CALL DATWRIT ;issue it to LCD
        CALL DELAY ;wait before issuing the next character
        CALL DELAY ;wait
        RET
WRITE_2764 ENDP

DATWRIT PROC
        PUSH DX  ;save DX
        MOV DX,PORTA  ;DX=port 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, PORTB ;port 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
DATWRIT ENDP ;writing on the lcd ends

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

lastcode: NOP

.exit
END
```

Design Drawn on Chart



Circuit Diagram -



References -

8086 DataSheet- https://www.archive.ece.cmu.edu/~ece740/f11/lib/exe/fetch.php?media=wiki:8086-datasheet.pdf

LM_020L Data sheet - https://www.digchip.com/datasheets/parts/datasheet/000/LM020L-pdf.php

2716 Datasheet- http://pdf.datasheet.live/5610c515/intel.com/ID2716.pdf

2764 Datasheet - http://pdf.datasheet.live/72d0ebcf/intel.com/LD2764A-30.pdf

74LS373 Datasheet - https://ecee.colorado.edu/~mcclurel/sn74ls373rev5.pdf

The Intel Microprocessors-Barry and Brey- https://userpages.umbc.edu/~squire/intel_book.pdf