Laboratory Report #7

Name: German E Felisarta III **Group Number:** 3

Laboratory Exercise Title: Hardware Interrupt Interfacing **Date Completed:** 11/28/2020

Exercise 7A:

Fig 1. Circuit Schematic

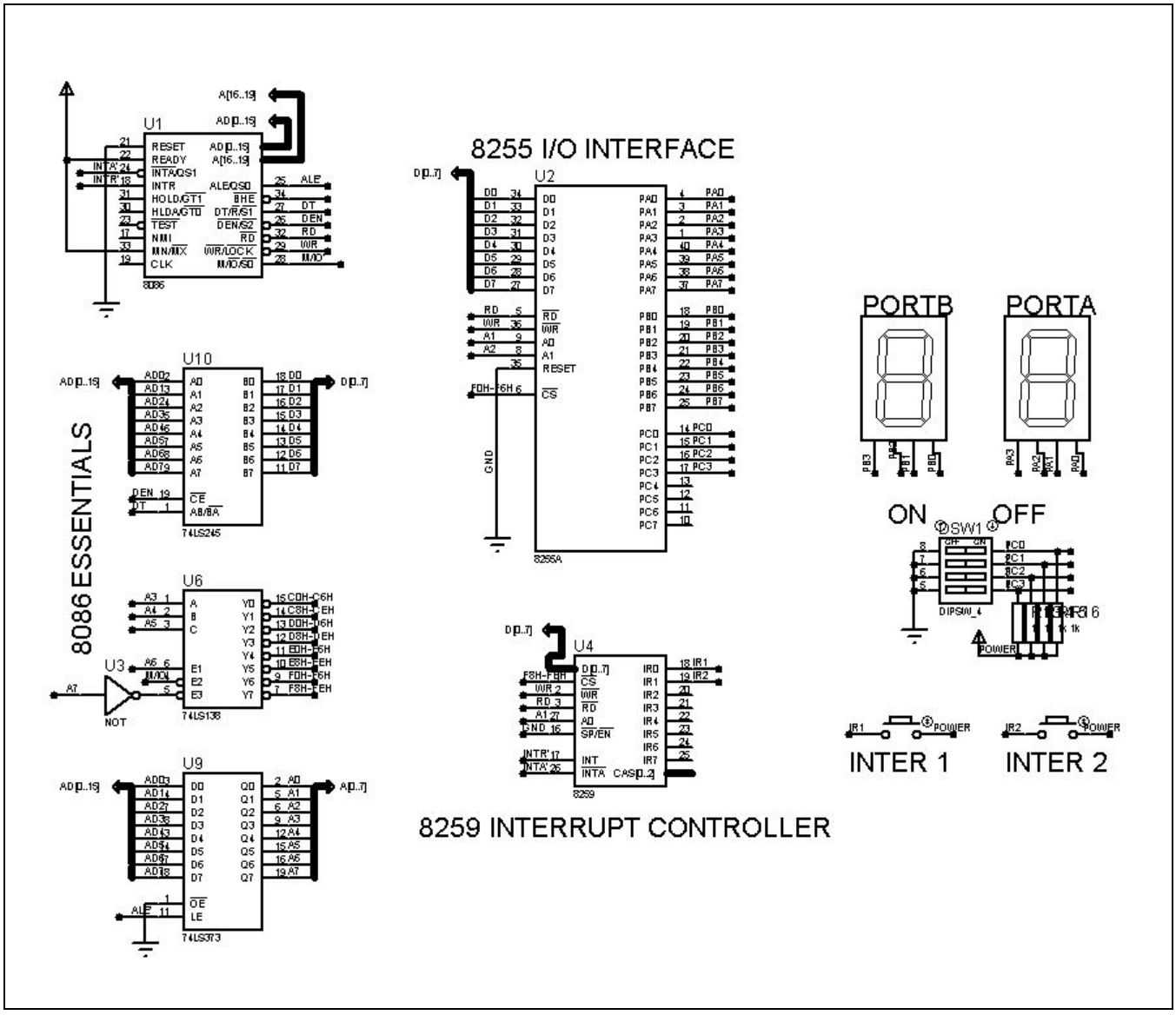


Fig 2. Assembly code for 7-1

```
 ;===================================================================
; Main.asm file generated by New Project wizard
;
; Created:   Thu Nov 5 2020
; Processor: 8086
; Compiler:  MASM32
;
; Before starting simulation set Internal Memory Size
; in the 8086 model properties to 0x10000
;
;          GERMAN E FELISARTA III 16101002    CpE3104 Grp 1
;
;===================================================================

PROCED1 SEGMENT
ISR1 PROC FAR
ASSUME CS:PROCED1, DS:DATA
ORG 08000H              ; write code within below starting at address 08000H

      PUSHF             ; push 16-bit operands
      PUSH AX           ; save program context
      PUSH DX

      MOV DX, PORTA     ; display '9' on the 7-segment in PORTA
      MOV AL, 09H
      OUT DX, AL

      POP DX            ; retrieve program context
      POP AX
      POPF              ; pop 16-bit operands
      IRET            ; return from interrupt


ISR1 ENDP               ; end of procedure
PROCED1 ENDS

PROCED2 SEGMENT
ISR2 PROC FAR
ASSUME CS:PROCED2, DS:DATA
ORG 09000H              ; write code within below starting at address 09000H

      PUSHF             ; push 16-bit operands
      PUSH AX           ; save program context
      PUSH DX

      MOV DX, PORTA     ; display '0' on the 7-segment in PORTA
      MOV AL, 00H
      OUT DX, AL

      POP DX            ; retrieve program context
      POP AX
      POPF              ; pop 16-bit operands
```

```
    IRET            ; return from interrupt

ISR2 ENDP               ; end of procedure
PROCED2 ENDS

DATA SEGMENT
ORG 0F000H

  PORTA   EQU 0F0H   ; PORTA address
  PORTB   EQU 0F2H   ; PORTB address
  PORTC   EQU 0F4H   ; PORTC address
  COM_REG EQU 0F6H   ; Command Register Address
  PIC1    EQU 0F8H   ; A1 = 0
  PIC2    EQU 0FAH   ; A1 = 1
  ICW1    EQU 013H   ; 8259 command word ICW1
  ICW2    EQU 080H   ; 8259 command word ICW2
  ICW4    EQU 03H    ; 8259 command word ICW4
  OCW1    EQU 0FCH   ; 8259 command word OCW1

DATA ENDS

STK SEGMENT STACK
  BOS DW 64D DUP(?)  ; stack depth (bottom of stack)
  TOS LABEL WORD     ; top of stack
STK ENDS

CODE SEGMENT PUBLIC 'CODE'
ASSUME CS:CODE, DS:DATA, SS:STK

  ORG 0E000H        ; write code within below starting at address 0E000H

  START:
    MOV AX, DATA
    MOV DS, AX      ; set the data segment address
    MOV AX, STK
    MOV SS, AX      ; set the stack segment address
    LEA SP, TOS     ; set the address of SP as top of stack

    CLI             ; clears IF flag

    ; program the 8255
    MOV DX, COM_REG
    MOV AL, 10001001B
    OUT DX, AL
    MOV DX, PORTA
    MOV AL, 00111111B
    OUT DX, AL

    ; program the 8259
    MOV DX, PIC1    ; set the I/O address to access ICW1
    MOV AL, ICW1
    OUT DX, AL      ; send command word
    MOV DX, PIC2    ; set the I/O address to access ICW2, ICW4 and OCW1
```

```
    MOV AL, ICW2
    OUT DX, AL      ; send command word
    MOV AL, ICW4
    OUT DX, AL      ; send command word
    MOV AL, OCW1
    OUT DX, AL      ; send command word
    STI             ; enable INTR pin of 8086

    ; storing interrupt vector to interrup vector table in memory
    MOV AX, OFFSET ISR1 ; get offset address of ISR1(IP)
    MOV [ES:200H], AX   ; store offset address  to memory
    MOV AX, SEG ISR1    ; get segment address of ISR1 (CS)
    MOV [ES:202H], AX   ; store segment address to memory


    MOV AX, OFFSET ISR2 ; get offset address of ISR2 (IP)
    MOV [ES:204H], AX   ; store offset address to memory
    MOV AX, SEG ISR2    ; get segment address of ISR2 (CS)
    MOV [ES:206H], AX   ; store segment address to memory


  ; foreground routine
  HERE:
        MOV DX, PORTC               ; select portc
        IN AL, DX               ; import input
        AND AL, 0FH             ; convert the high upper nibble into low so that it can be properly compared

        CMP AL, 09H             ;checks if input is less than or equal to 9
        JLE DISPLAY
        MOV AL, 00H             ; if greater than then display 0

        DISPLAY:
          MOV DX, PORTB     ; select portb
          OUT DX, AL        ; output the input or 0

        JMP HERE

CODE ENDS
END START
```

Fig 3. Interrupt 1 is Pressed

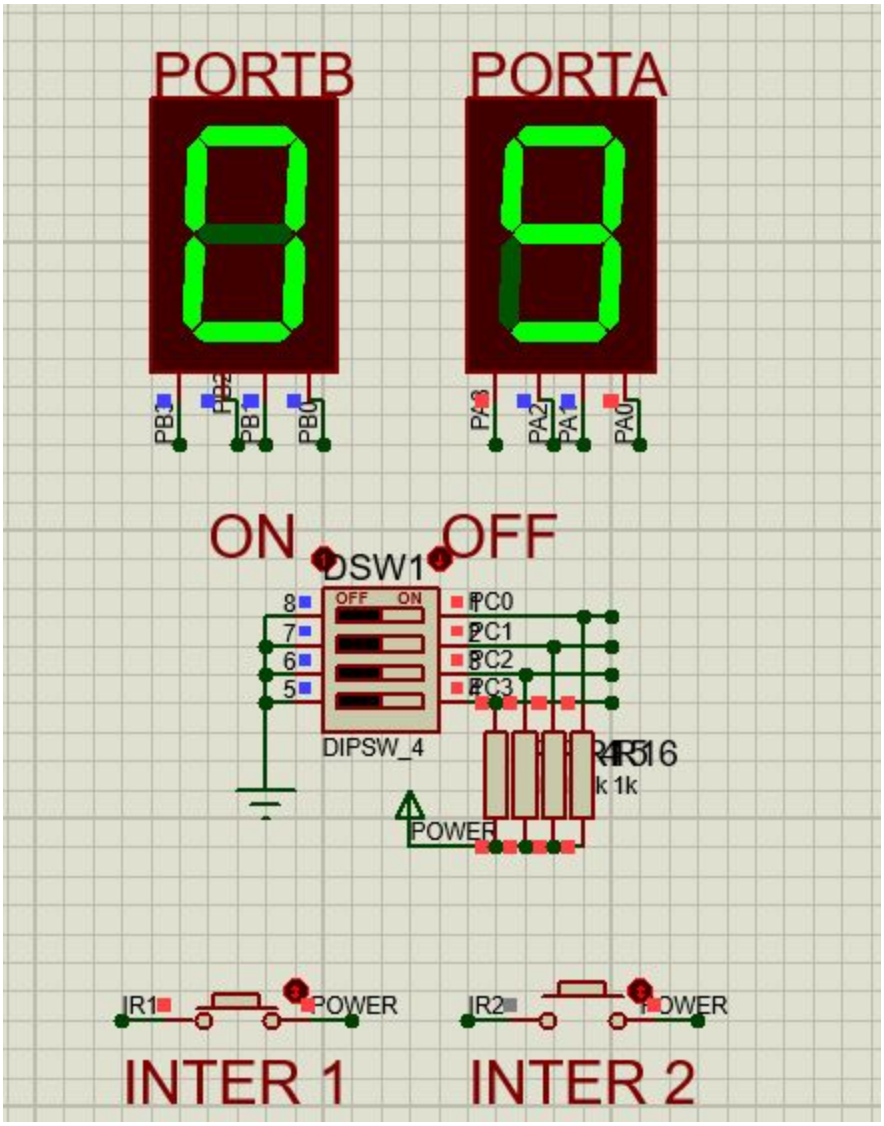


Fig 4. Interrupt 2 is pressed

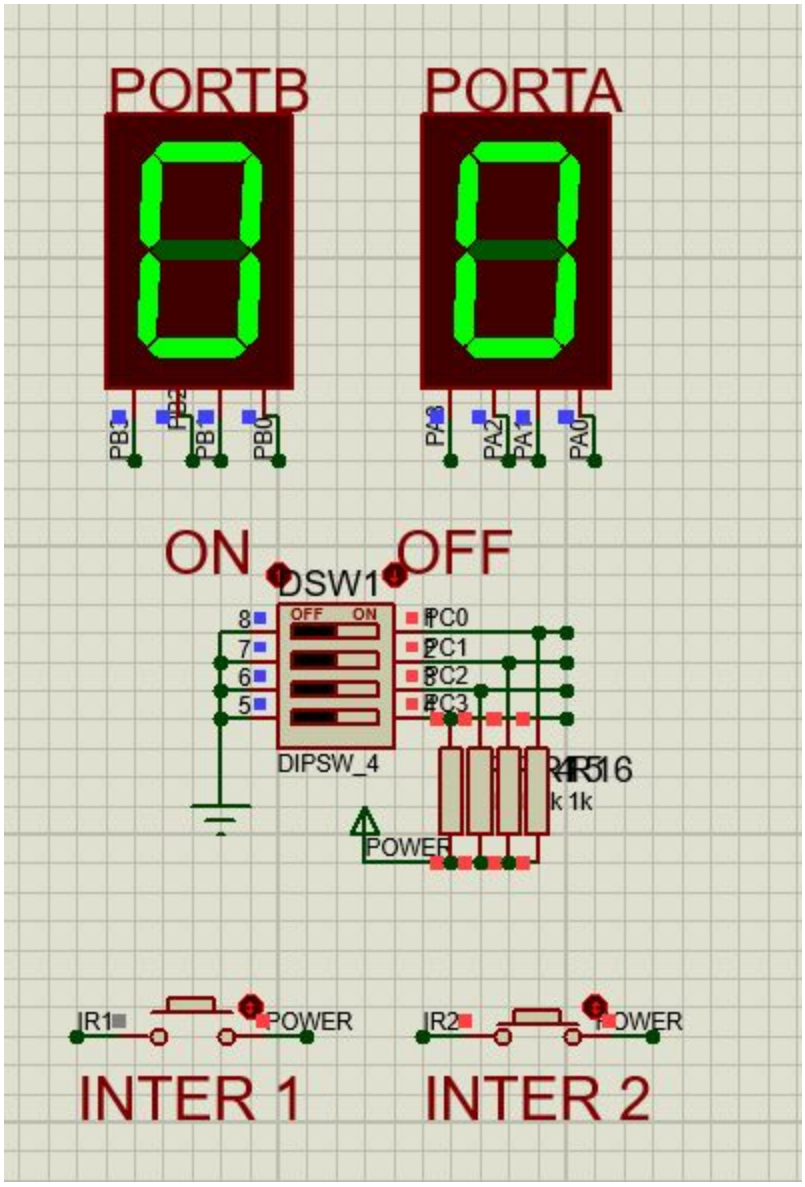
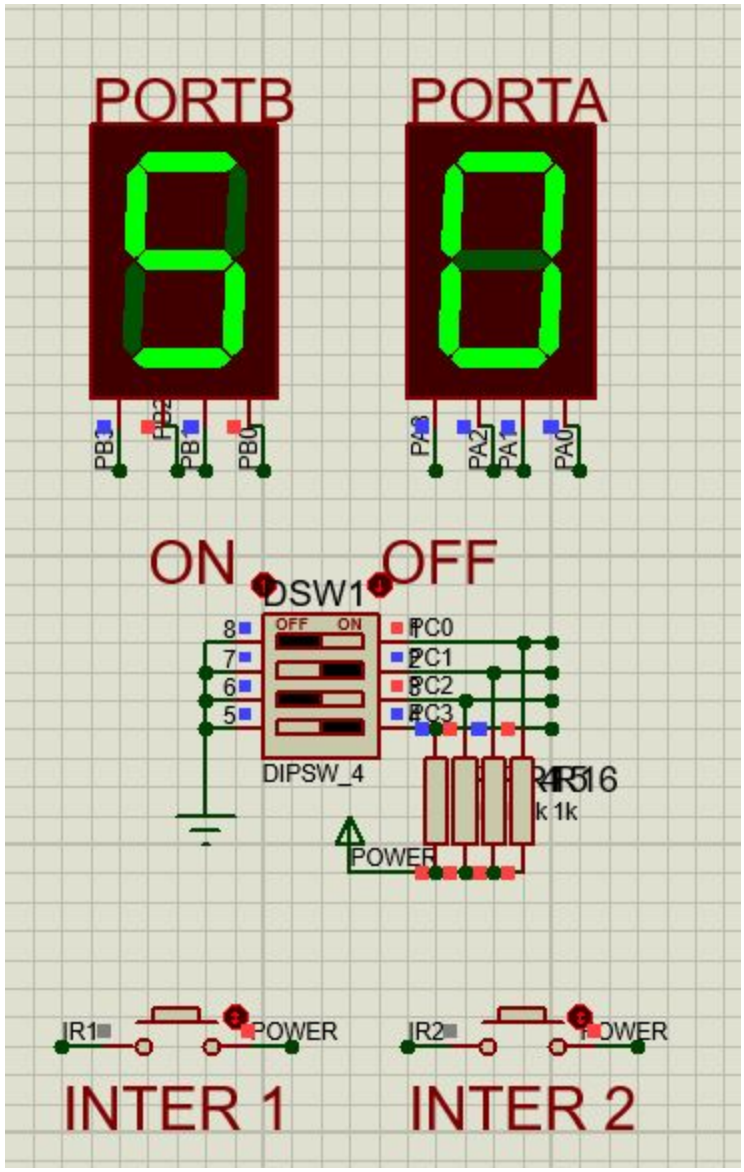


Fig 5. PORT B output based on DIP SWITCH



Exercise 7B:

6. Why do you think the LED is blinking steadily while other activities are going on?

Because the blinking instruction is in the foreground part of the code, meaning it would be the one that is being run always.

7. What do you think is the ultimate advantage of using interrupts especially involving I/O devices?

It can help in simulating instantaneous events. Also it could isolate events to only happen when interrupts are triggered.

NOTES:

I have noticed when programming the interrupt with a lot of CMP instructions and interaction with the AL register, the POP instruction will not work properly. To curb this problem, I had to make a Label before writing the POP function like the code below:

Fig 0. POP Instruction Problem

```
PRINT:
 MOV DX, PORTA      ; selects portA
 OUT DX, AL         ; outputs to PortB
 JMP GO             ; initiates pop instructions

GO:                 ; getout of interrupt function
 POP DX
 POP AX
 POPF
 IRET
```

Fig 1. Circuit Schematic

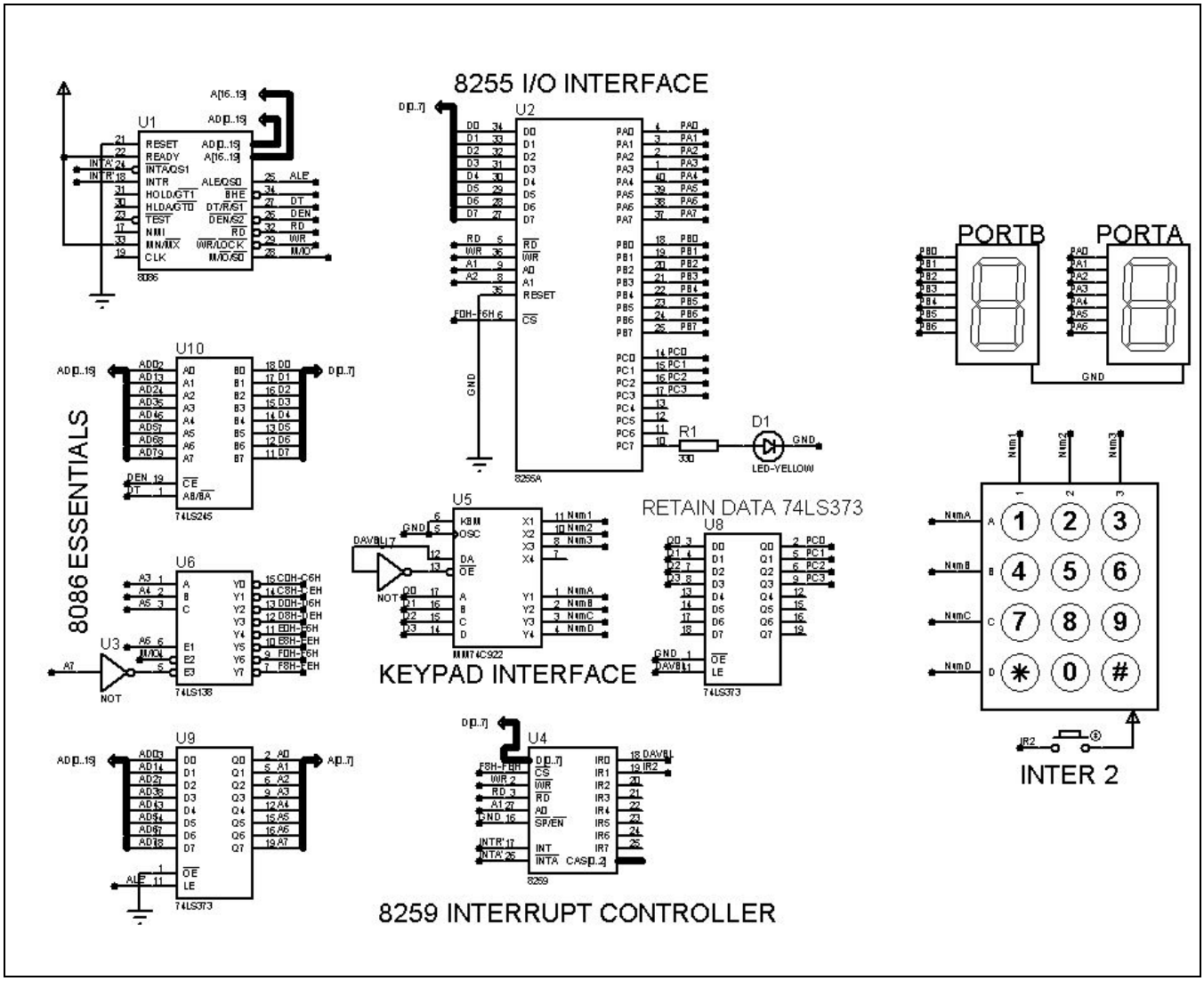


Fig 2. Assembly code for 7-1

```
;===================================================================
; Main.asm file generated by New Project wizard
;
; Created:   Thu Nov 5 2020
; Processor: 8086
; Compiler:  MASM32
;
; Before starting simulation set Internal Memory Size
; in the 8086 model properties to 0x10000
;
;          GERMAN E FELISARTA III 16101002    CpE3104 Grp 1
;
```

```
;===================================================================

PROCED1 SEGMENT
ISR1 PROC FAR
ASSUME CS:PROCED1, DS:DATA
ORG 08000H              ; write code within below starting at address 08000H

      PUSHF             ; push 16-bit operands
      PUSH AX           ; save program context
      PUSH DX

      MOV DX, PORTC     ; select portc
         IN AL, DX      ; import input
         AND AL, 0FH    ; Isolate the lower nibble only

         CONVERT:
            CMP AL, 00H             ; adjust the '1' input from keypad
            JE ONE
            CMP AL, 01H             ; adjust the '2' input from keypad
            JE TWO
            CMP AL, 02H             ; adjust the '3' input from keypad
            JE THREE
            CMP AL, 04H    ; adjust the '4' input from keypad
            JE FOUR
            CMP AL, 05H    ; adjust the '5' input from keypad
            JE FIVE
            CMP AL, 06H    ; adjust the '6' input from keypad
            JE SIX
            CMP AL, 08H             ; adjust the '7' input from keypad
            JE SEVEN
            CMP AL, 09H             ; adjust the '8' input from keypad
            JE EIGHT
            CMP AL, 0AH             ; adjust the '9' input from keypad
            JE NINE
            CMP AL, 0CH             ; adjust the '*' input from keypad
            JE DASH
            CMP AL, 0DH             ; adjust the '0' input from keypad
            JE ZERO
            CMP AL, 0EH             ; adjust the '#' input from keypad
            JE DASH
            JMP PRINT

         ONE:
          MOV AL, 00000110B
          JMP PRINT

         TWO:
          MOV AL, 01011011B
          JMP PRINT

         THREE:
          MOV AL, 01001111B
          JMP PRINT
```

```
        FOUR:
         MOV AL, 01100110B
         JMP PRINT

        FIVE:
         MOV AL, 01101101B
         JMP PRINT

        SIX:
         MOV AL, 01111101B
         JMP PRINT

        SEVEN:
         MOV AL, 00000111B
         JMP PRINT

        EIGHT:
         MOV AL, 01111111B
         JMP PRINT

        NINE:
         MOV AL, 01100111B
         JMP PRINT

        ZERO:
         MOV AL, 00111111B
         JMP PRINT

        DASH:
         MOV AL, 01000000B
         JMP PRINT

        PRINT:
         MOV DX, PORTA     ; selects portA
         OUT DX, AL        ; outputs to PortB
         JMP GO            ; initiates pop instructions

        GO:               ; getout of interrupt function
         POP DX
         POP AX
         POPF
         IRET

ISR1 ENDP                 ; end of procedure
PROCED1 ENDS

PROCED2 SEGMENT
ISR2 PROC FAR
ASSUME CS:PROCED2, DS:DATA
ORG 09000H                ; write code within below starting at address 09000H

      PUSHF               ; push 16-bit operands
```

```
PUSH AX          ; save program context
PUSH DX

MOV DX, PORTC     ; select portc
   IN AL, DX         ; import input
   AND AL, 0FH       ; Isolate the lower nibble only

   CONVERT:
      CMP AL, 00H             ; adjust the '1' input from keypad
      JE ONE
      CMP AL, 01H             ; adjust the '2' input from keypad
      JE TWO
      CMP AL, 02H             ; adjust the '3' input from keypad
      JE THREE
      CMP AL, 04H    ; adjust the '4' input from keypad
      JE FOUR
      CMP AL, 05H    ; adjust the '5' input from keypad
      JE FIVE
      CMP AL, 06H    ; adjust the '6' input from keypad
      JE SIX
      CMP AL, 08H             ; adjust the '7' input from keypad
      JE SEVEN
      CMP AL, 09H             ; adjust the '8' input from keypad
      JE EIGHT
      CMP AL, 0AH             ; adjust the '9' input from keypad
      JE NINE
      CMP AL, 0CH             ; adjust the '*' input from keypad
      JE DASH
      CMP AL, 0DH             ; adjust the '0' input from keypad
      JE ZERO
      CMP AL, 0EH             ; adjust the '#' input from keypad
      JE DASH
      JMP PRINT

   ONE:
    MOV AL, 00000110B
    JMP PRINT

   TWO:
    MOV AL, 01011011B
    JMP PRINT

   THREE:
    MOV AL, 01001111B
    JMP PRINT

   FOUR:
    MOV AL, 01100110B
    JMP PRINT

   FIVE:
    MOV AL, 01101101B
    JMP PRINT
```

```
        SIX:
         MOV AL, 01111101B
         JMP PRINT

        SEVEN:
         MOV AL, 00000111B
         JMP PRINT

        EIGHT:
         MOV AL, 01111111B
         JMP PRINT

        NINE:
         MOV AL, 01100111B
         JMP PRINT

        ZERO:
         MOV AL, 00111111B
         JMP PRINT

        DASH:
         MOV AL, 01000000B
         JMP PRINT

        PRINT:
         MOV DX, PORTB      ; selects portB
         OUT DX, AL         ; outputs to PortB
         JMP GO                     ; initiates pop instructions

        GO:                 ; getout of interrupt function
         POP DX
         POP AX
         POPF
         IRET
ISR2 ENDP               ; end of procedure
PROCED2 ENDS

DATA SEGMENT
ORG 0F000H

   PORTA   EQU 0F0H    ; PORTA address
   PORTB   EQU 0F2H    ; PORTB address
   PORTC   EQU 0F4H    ; PORTC address
   COM_REG EQU 0F6H    ; Command Register Address
   PIC1    EQU 0F8H    ; A1 = 0
   PIC2    EQU 0FAH    ; A1 = 1
   ICW1    EQU 013H    ; 8259 command word ICW1
   ICW2    EQU 080H    ; 8259 command word ICW2
   ICW4    EQU 03H     ; 8259 command word ICW4
   OCW1    EQU 0FCH    ; 8259 command word OCW1

DATA ENDS
```

```
STK SEGMENT STACK
   BOS DW 64D DUP(?)   ; stack depth (bottom of stack)
   TOS LABEL WORD      ; top of stack
STK ENDS

CODE SEGMENT PUBLIC 'CODE'
ASSUME CS:CODE, DS:DATA, SS:STK
ORG 0E000H          ; write code within below starting at address 0E000H

   START:
      MOV AX, DATA
      MOV DS, AX      ; set the data segment address
      MOV AX, STK
      MOV SS, AX      ; set the stack segment address
      LEA SP, TOS     ; set the address of SP as top of stack

      CLI             ; clears IF flag

      ; program the 8255
      MOV DX, COM_REG
      MOV AL, 81H
      OUT DX, AL

      MOV DX, PORTA
      MOV AL, 00H
      OUT DX, AL

         MOV DX, PORTB
      MOV AL, 00H
         OUT DX, AL

      ; program the 8259
      MOV DX, PIC1    ; set the I/O address to access ICW1
      MOV AL, ICW1
      OUT DX, AL      ; send command word
      MOV DX, PIC2    ; set the I/O address to access ICW2, ICW4 and OCW1
      MOV AL, ICW2
      OUT DX, AL      ; send command word
      MOV AL, ICW4
      OUT DX, AL      ; send command word
      MOV AL, OCW1
      OUT DX, AL      ; send command word
      STI             ; enable INTR pin of 8086

      ; storing interrupt vector to interrup vector table in memory
      MOV AX, OFFSET ISR1 ; get offset address of ISR1(IP)
      MOV [ES:200H], AX   ; store offset address  to memory
      MOV AX, SEG ISR1    ; get segment address of ISR1 (CS)
      MOV [ES:202H], AX   ; store segment address to memory


      MOV AX, OFFSET ISR2 ; get offset address of ISR2 (IP)
```

```
    MOV [ES:204H], AX   ; store offset address to memory
    MOV AX, SEG ISR2    ; get segment address of ISR2 (CS)
    MOV [ES:206H], AX   ; store segment address to memory

  ; foreground routine
  HERE:
    MOV DX, PORTC           ; select portC
      MOV AL, 80H           ; activate led at PC7
      OUT DX, AL

      CALL DELAY

      MOV AL, 00H           ; deactivate led at PC7
      OUT DX, AL

      CALL DELAY

      JMP HERE              ; loop system

DELAY PROC NEAR                     ; TIME DELAY (optional)
  MOV CX, 0FFFFh
  DELAY_LOOP:
    DEC CX
    CMP CX, 00H
    JNZ DELAY_LOOP
  RET
DELAY ENDP


CODE ENDS
END START
```

Fig 3. Numpad keys are pressed

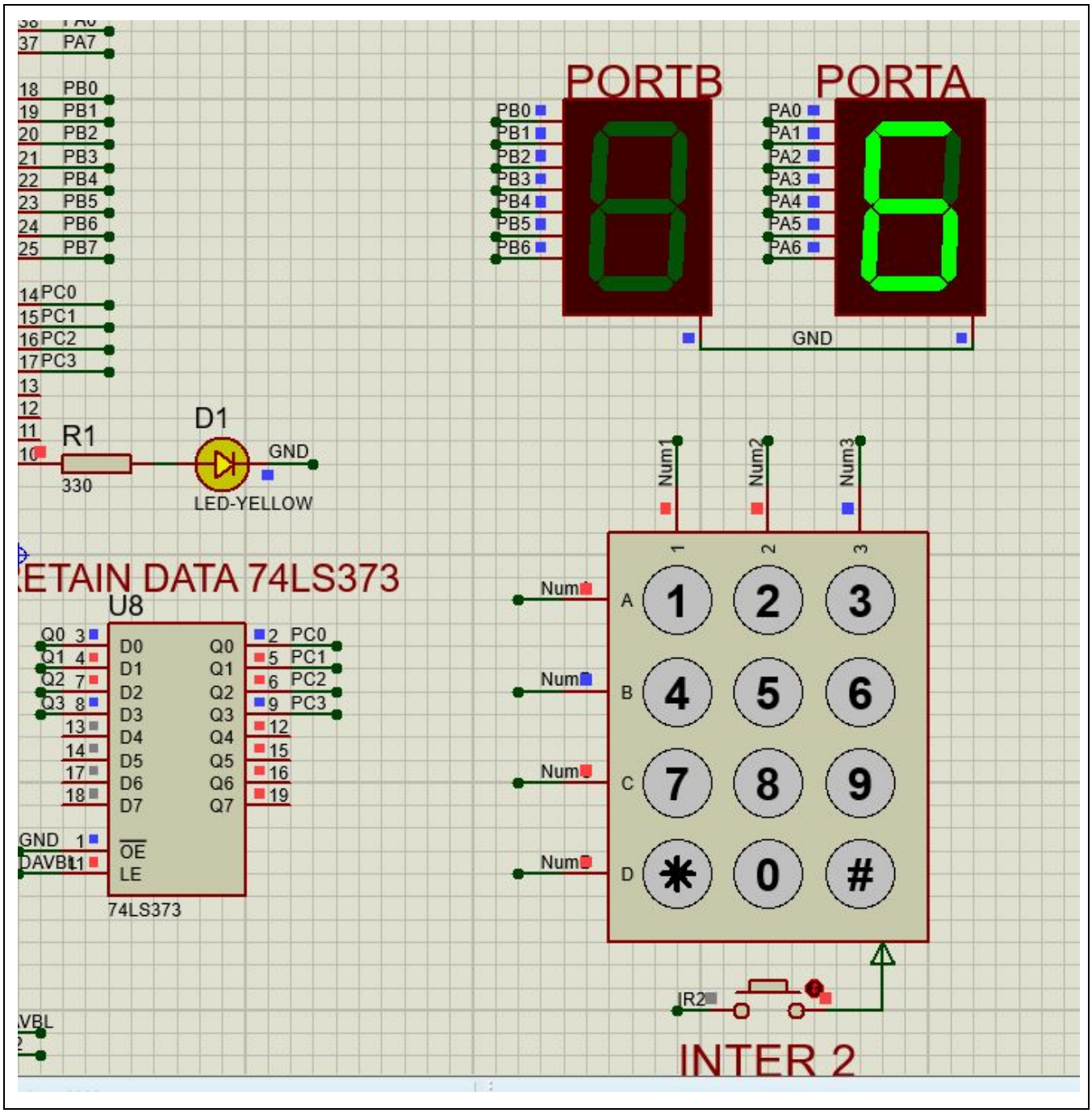


Fig 4. Interrupt 2 is pressed

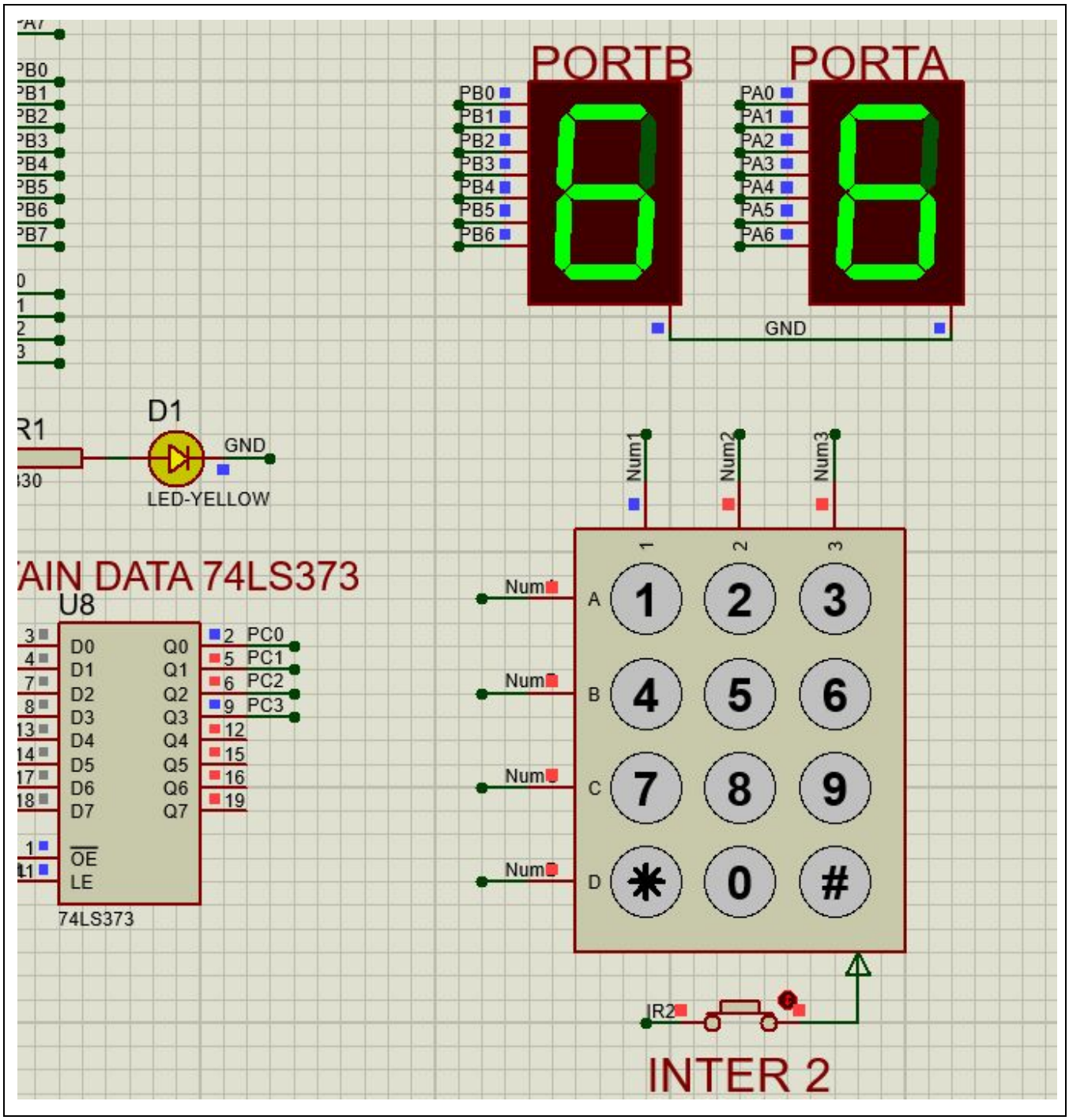


Fig 5. Asterisk is pressed