**Department of Computer Engineering** 

Digital Hardware Systems

*C CpE 3104 - Microprocessors*

**Laboratory Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Laboratory Exercise No.:** | 6 | **Date Performed:** | 11/09/2022 |
| **Laboratory Exercise Title:** | Parallel I/O Interfacing | | |
| **Name of Student:** | Paul John Toral | **Document Version:** | 1.0 |

**Activity #1**

A picture containing text, indoor

Description automatically generated

**Figure 1: Schematic Diagram for Activity 1**

**Activity #2**

**Diagram, schematic

Description automatically generated**

**Figure 2: Schematic Diagram for Activity 2**

**Activity #3**

**Box and whisker chart

Description automatically generated with medium confidence**

**Figure 3: Schematic Diagram for Activity 3**

**Source Code #1**

;====================================================================

; Main.asm file generated by New Project wizard

;

; Created: Mon Oct 17 2022

; Processor: 8086

; Compiler: MASM32

;

; Before starting simulation set Internal Memory Size

; in the 8086 model properties to 0x10000

;====================================================================

DATA SEGMENT

PORTA EQU 0F0H;

PORTB EQU 0F2H;

PORTC EQU 0F4H;

COM\_REG EQU 0F6H;

LCD\_STR DB "HELLO!","$"

DATA ENDS

CODE SEGMENT PUBLIC 'CODE'

ASSUME CS:CODE

MOV AX, DATA

MOV DS, AX

ORG 0000H

START:

MOV DX, COM\_REG ; set the address

MOV AL, 10001001B

OUT DX, AL ; send the command byte

CALL INIT\_LCD;

LEA SI, LCD\_STR

MOV AL, 0C7H ; move cursor to 8th column of 2nd line

CALL INST\_CTRL ; send instruction to LCD

DISPLAY\_STR:

MOV AL, [SI]

CMP AL, '$'

JE EXIT

CALL DATA\_CTRL

INC SI

JMP DISPLAY\_STR

INST\_CTRL:

PUSH AX ; preserve value of AL

MOV DX, PORTA ; set port of LCD data bus (PORTA)

OUT DX, AL ; write data in AL to PORTA

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 02H ; E=1, RS=0 (access instruction reg)

OUT DX, AL ; write data in AL to PORTB

CALL DELAY\_1MS ; delay for 1 ms

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 00H ; E=0, RS=0

OUT DX, AL ; write data in AL to PORTB

POP AX ; restore value of AL

RET

DATA\_CTRL:

PUSH AX ; preserve value of AL

MOV DX, PORTA ; set port of LCD data bus (PORTA)

OUT DX, AL ; write data in AL to PORTA

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 03H ; E=1, RS=1 (access data register)

OUT DX, AL ; write data in AL to PORTB

CALL DELAY\_1MS ; delay for 1 ms

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 01H ; E=0, RS=1

OUT DX, AL ; write data in AL to PORTB

POP AX ; restore value of AL

RET

INIT\_LCD:

MOV AL, 38H ; 8-bit interface, dual-line display

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 08H ; display off, cursor off, blink off

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 01H ; clear display

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 06H ; increment cursor, display shift off

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 0CH ; display on, cursor off, blink off

CALL INST\_CTRL ; write instruction to LCD

RET

DELAY\_1MS:

MOV BX, 02CAH

L1:

DEC BX

NOP

JNZ L1

RET

RET

EXIT:

CODE ENDS

END START

**Source Code #2**

;====================================================================

; Main.asm file generated by New Project wizard

;

; Created: Mon Oct 17 2022

; Processor: 8086

; Compiler: MASM32

;

; Before starting simulation set Internal Memory Size

; in the 8086 model properties to 0x10000

;====================================================================

DATA SEGMENT

PORTA EQU 0F0H;

PORTB EQU 0F2H;

PORTC EQU 0F4H;

COM\_REG EQU 0F6H;

DATA ENDS

CODE SEGMENT PUBLIC 'CODE'

ASSUME CS:CODE

MOV AX, DATA

MOV DS, AX

ORG 0000H

START:

MOV DX, COM\_REG ; set the address

MOV AL, 10001001B

OUT DX, AL ; send the command byte

CALL INIT\_LCD;

CHECK\_DAVBL:

MOV DX, PORTC

IN AL, DX

TEST AL, 10H

JZ CHECK\_DAVBL

IN AL, DX

AND AL, 0FH

PUSH AX

CHECK\_INPUT:

CMP AL, 0CH ; check if key pressed is \*

JE PRNT\_AST

CMP AL, 0EH ; check if key pressed is #

JE PRNT\_OCTO

CMP AL, 0DH ; check if key pressed is 0

JE PRNT\_0

CMP AL, 00H ; check if key pressed is 1

JE PRNT\_1

CMP AL, 01H ; check if key pressed is 2

JE PRNT\_2

CMP AL, 02H ; check if key pressed is 3

JE PRNT\_3

CMP AL, 04H ; check if key pressed is 4

JE PRNT\_4

CMP AL, 05H ; check if key pressed is 5

JE PRNT\_5

CMP AL, 06H ; check if key pressed is 6

JE PRNT\_6

CMP AL, 08H ; check if key pressed is 7

JE PRNT\_7

CMP AL, 09H ; check if key pressed is 8

JE PRNT\_8

CMP AL, 0AH ; check if key pressed is 9

JE PRNT\_9

JMP CHECK\_DAVBL

PRNT\_AST:

CALL CENTER

MOV AL, '\*'

JMP PRINT\_CHAR

PRNT\_OCTO:

CALL CENTER

MOV AL, '#'

JMP PRINT\_CHAR

PRNT\_0:

CALL CENTER

MOV AL, '0'

JMP PRINT\_CHAR

PRNT\_1:

CALL CENTER

MOV AL, '1'

JMP PRINT\_CHAR

PRNT\_2:

CALL CENTER

MOV AL, '2'

JMP PRINT\_CHAR

PRNT\_3:

CALL CENTER

MOV AL, '3'

JMP PRINT\_CHAR

PRNT\_4:

CALL CENTER

MOV AL, '4'

JMP PRINT\_CHAR

PRNT\_5:

CALL CENTER

MOV AL, '5'

JMP PRINT\_CHAR

PRNT\_6:

CALL CENTER

MOV AL, '6'

JMP PRINT\_CHAR

PRNT\_7:

CALL CENTER

MOV AL, '7'

JMP PRINT\_CHAR

PRNT\_8:

CALL CENTER

MOV AL, '8'

JMP PRINT\_CHAR

PRNT\_9:

CALL CENTER

MOV AL, '9'

JMP PRINT\_CHAR

PRINT\_CHAR:

CALL DATA\_CTRL

POP AX

JMP CHECK\_DAVBL

CENTER:

MOV AL, 0CAH

CALL INST\_CTRL

RET

INST\_CTRL:

PUSH AX ; preserve value of AL

MOV DX, PORTA ; set port of LCD data bus (PORTA)

OUT DX, AL ; write data in AL to PORTA

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 02H ; E=1, RS=0 (access instruction reg)

OUT DX, AL ; write data in AL to PORTB

CALL DELAY\_1MS ; delay for 1 ms

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 00H ; E=0, RS=0

OUT DX, AL ; write data in AL to PORTB

POP AX ; restore value of AL

RET

DATA\_CTRL:

PUSH AX ; preserve value of AL

MOV DX, PORTA ; set port of LCD data bus (PORTA)

OUT DX, AL ; write data in AL to PORTA

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 03H ; E=1, RS=1 (access data register)

OUT DX, AL ; write data in AL to PORTB

CALL DELAY\_1MS ; delay for 1 ms

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 01H ; E=0, RS=1

OUT DX, AL ; write data in AL to PORTB

POP AX ; restore value of AL

RET

INIT\_LCD:

MOV AL, 38H ; 8-bit interface, dual-line display

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 08H ; display off, cursor off, blink off

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 01H ; clear display

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 06H ; increment cursor, display shift off

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 0CH ; display on, cursor off, blink off

CALL INST\_CTRL ; write instruction to LCD

RET

DELAY\_1MS:

MOV BX, 02CAH

L1:

DEC BX

NOP

JNZ L1

RET

RET

EXIT:

CODE ENDS

END START

**Source Code #3**

;====================================================================

; Main.asm file generated by New Project wizard

;

; Created: Thu Nov 10 2022

; Processor: 8086

; Compiler: MASM32

;

; Before starting simulation set Internal Memory Size

; in the 8086 model properties to 0x10000

;====================================================================

DATA SEGMENT

PORTA EQU 0F0H;

PORTB EQU 0F2H;

PORTC EQU 0F4H;

COM\_REG EQU 0F6H;

PORTA2 EQU 0E0H;

PORTB2 EQU 0E2H;

PORTC2 EQU 0E4H;

COM\_REG2 EQU 0E6H;

LOAD\_CTR0 EQU 0F8H ; counter 0 address

TIMER\_REG EQU 0FEH ; 8253 Command register address

DISPENSE\_PROMPT DB "Dispensing...","$"

READY DB "Get Your Drink :D","$"

L\_CK DB "[1] Coke Large","$"

M\_CK DB "[2] Coke Medium","$"

L\_SPR DB "[3] Sprite Large","$"

M\_SPR DB "[4] Sprite Medium","$"

DATA ENDS

CODE SEGMENT PUBLIC 'CODE'

ASSUME CS:CODE

MOV AX, DATA

MOV DS, AX

ORG 0000H

START:

MOV DX, COM\_REG ; set the address for 1st 8255

MOV AL, 10001001B

OUT DX, AL ; send the command byte

MOV DX, COM\_REG2 ; set the address for 2nd 8255

MOV AL, 10001001B

OUT DX, AL ; send the command byte

CALL INIT\_LCD;

MOV DX, TIMER\_REG

MOV AL, 00111000B

OUT DX, AL

DISPLAY:

CALL CLEAR\_SCREEN

;printing first line;

LEA SI, L\_CK

MOV AL, 081H

CALL INST\_CTRL

CALL PRINT\_STR

;printing second line;

LEA SI, M\_CK

MOV AL, 0C1H

CALL INST\_CTRL

CALL PRINT\_STR

;printing third line;

LEA SI, L\_SPR

MOV AL, 095H

CALL INST\_CTRL

CALL PRINT\_STR

;printing fourth line;

LEA SI, M\_SPR

MOV AL, 0D5H

CALL INST\_CTRL

CALL PRINT\_STR

CHECK\_DAVBL:

MOV DX, PORTC

IN AL, DX

TEST AL, 10H

JZ CHECK\_DAVBL

IN AL, DX

AND AL, 0FH

PUSH AX

CHECK\_INPUT:

CMP AL, 00H ; check if key pressed is 0

JE GET\_L\_CK

CMP AL, 01H ; check if key pressed is 1

JE GET\_M\_CK

CMP AL, 02H ; check if key pressed is 2

JE GET\_L\_SPR

CMP AL, 04H ; check if key pressed is 3

JE GET\_M\_SPR

JMP CHECK\_DAVBL

GET\_L\_CK:

CALL CLEAR\_SCREEN

LEA SI, DISPENSE\_PROMPT

CALL LINE\_2

CALL PRINT\_STR

MOV CL, 07H

JMP LED\_1

GET\_M\_CK:

CALL CLEAR\_SCREEN

LEA SI, DISPENSE\_PROMPT

CALL LINE\_2

CALL PRINT\_STR

MOV CL, 04H

JMP LED\_2

GET\_L\_SPR :

CALL CLEAR\_SCREEN

LEA SI, DISPENSE\_PROMPT

CALL LINE\_2

CALL PRINT\_STR

MOV CL, 07H

JMP LED\_3

GET\_M\_SPR:

CALL CLEAR\_SCREEN

LEA SI, DISPENSE\_PROMPT

CALL LINE\_2

CALL PRINT\_STR

MOV CL, 04H

JMP LED\_4

LED\_1:

MOV DX, PORTA2

MOV AL, 00000001B

OUT DX,AL

CALL DISPLAY\_COUNT

CALL FINISHED

POP AX

JMP CHECK\_DAVBL

LED\_2:

MOV DX, PORTA2

MOV AL, 00000010B

OUT DX,AL

CALL DISPLAY\_COUNT

CALL FINISHED

POP AX

JMP CHECK\_DAVBL

LED\_3:

MOV DX, PORTA2

MOV AL, 00000100B

OUT DX,AL

CALL DISPLAY\_COUNT

CALL FINISHED

POP AX

JMP CHECK\_DAVBL

LED\_4:

MOV DX, PORTA2

MOV AL, 00001000B

OUT DX,AL

CALL DISPLAY\_COUNT

CALL FINISHED

POP AX

JMP CHECK\_DAVBL

PRINT\_STR:

MOV AX, [SI] ; store character

CMP AL, '$' ; check if end

JE DELAY\_1MS ; jump if end

CALL DATA\_CTRL ; display character

INC SI ; increment to next character in the array

JMP PRINT\_STR ; loop back procedure

DISPLAY\_COUNT:

CALL LINE\_3

MOV AL, 030H

ADD AL, CL

CALL DATA\_CTRL

MOV AL, 's'

CALL DATA\_CTRL

CALL DELAY\_1S

DEC CL

CMP CL, 00H

JNE DISPLAY\_COUNT

RET

FINISHED:

CALL CLEAR\_SCREEN

LEA SI, READY

MOV AL, 0C1H

CALL INST\_CTRL

CALL PRINT\_STR

MOV DX, PORTA2

MOV AL, 00000000B

OUT DX,AL

CALL DELAY\_1S

JMP DISPLAY

RET

CENTER:

MOV AL, 0CAH

CALL INST\_CTRL

RET

LINE\_2:

MOV AL, 0C4H

CALL INST\_CTRL

RET

LINE\_3:

MOV AL, 09EH

CALL INST\_CTRL

RET

INST\_CTRL:

PUSH AX ; preserve value of AL

MOV DX, PORTA ; set port of LCD data bus (PORTA)

OUT DX, AL ; write data in AL to PORTA

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 02H ; E=1, RS=0 (access instruction reg)

OUT DX, AL ; write data in AL to PORTB

CALL DELAY\_1MS ; delay for 1 ms

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 00H ; E=0, RS=0

OUT DX, AL ; write data in AL to PORTB

POP AX ; restore value of AL

RET

DATA\_CTRL:

PUSH AX ; preserve value of AL

MOV DX, PORTA ; set port of LCD data bus (PORTA)

OUT DX, AL ; write data in AL to PORTA

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 03H ; E=1, RS=1 (access data register)

OUT DX, AL ; write data in AL to PORTB

CALL DELAY\_1MS ; delay for 1 ms

MOV DX, PORTB ; set port of LCD control lines (PORTB)

MOV AL, 01H ; E=0, RS=1

OUT DX, AL ; write data in AL to PORTB

POP AX ; restore value of AL

RET

INIT\_LCD:

MOV AL, 38H ; 8-bit interface, dual-line display

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 08H ; display off, cursor off, blink off

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 01H ; clear display

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 06H ; increment cursor, display shift off

CALL INST\_CTRL ; write instruction to LCD

MOV AL, 0CH ; display on, cursor off, blink off

CALL INST\_CTRL ; write instruction to LCD

RET

;==========1 milisecond timer ============;

DELAY\_1MS:

MOV BX, 02CAH

L1:

DEC BX

NOP

JNZ L1

RET

RET

;==========1 second timer ============;

DELAY\_1S:

MOV DX, LOAD\_CTR0

MOV AL, 0A0H

OUT DX, AL

MOV AL, 0FH

OUT DX, AL

TIMER:

MOV DX, PORTC2

IN AX, DX

MOV AH, 00H

AND AL, 00000001B ; MASK OFF

CMP AL, 00H

JNE TIMER

RET

CLEAR\_SCREEN:

MOV AL, 01H

CALL INST\_CTRL

RET

EXIT:

CODE ENDS

END START