



**LABORATORY MANUAL**

**AMITY UNIVERSITY CHHATTISGARH**

# Amity University Chhattisgarh

**Amity School of Engineering & Technology Batch : 2020 – 2024**

**Enrolment Number : A80105220036**

This is to certify that this is a bonafide record of the work done by **Ritik Gupta**

bearing enrollment number **A80105220036** of B. Tech **Computer Science & Engineering** semester **V** , from Amity School of Engineering & Technology, Amity University Chhattisgarh in the **Microprocessors and Microcontrollers** Laboratory with Course Code **CSE3506**.

University Examination held on --------------------------------- --.

Faculty in-charge Director-ASET

Examiner- 1

Examiner-2

## INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **S. NO.** | **PARTICULARS** | **PAGE NO.** | **SIGN.** |
| 1. | Introduction to Microsoft s Macro Assembler (MASM). | 4-6 |  |
| 2. | Programs for 16 bit arithmetic operations of 8086(using various Addressing Modes). | 7-9 |  |
| 3. | PROGRAM FOR SORTING AN ARRAY FOR 8086 | 10-12 |  |
| 4. | Program for searching for a number or character in a string for 8086. | 13-15 |  |
| 5. | Program for String Manipulation for 8086. | 16-17 |  |
| 6. | Interfacing ADC&DAC to 8086. | 18-19 |  |
| 7. | Interfacing stepper to 8086. | 20-21 |  |
| 8. | Program Using Arithmetic, Logical, and Bit Manipulation Instructions Of 8051. | 22-23 |  |
| 9. | Timer/Counters in 8051. | 24-25 |  |
| 10. | PROGRAM AND VERIFY INTERRUPT HANDLING IN 8051 | 26-27 |  |
| 11. | PROGRAM FOR UART OPERATION IN 8051. | 28-29 |  |
| 12. | Interfacing LCD to 8051. | 29-31 |  |

**Experiments No.1:**

**Aim: -Introduction to Microsoft s Macro Assembler (MASM).**

**Introduction: -**

* The Microsoft macro assembler is an x86 high level assembler for DOS and Microsoft windows. It supports wide varieties of macro facilities and structured programming idioms including high level functions for looping and procedures.
* A program called **assembler** used to convert the mnemonics of instructions along withthe data into the equivalent object code modules, these object code may further converted into executable code using linked and loader programs. This type of program is called as ASSEMBLY LANGUAGE PROGRAMMING.
* The assembler converts and Assembly language source file to machine code the binary equivalent of the assembly language program. In this respect, the assembler reads an ASCII source file from thedisk and program as output.
* The major different between compilers for a high level language like PASCAL and an Assembler is that the compiler usually emits severalmachine instructions for each PASCAL statement. The assembler generally emits a single machine instruction for each assembler language statement.
* Microsoft MASM version 6.11 contains updated software capable of processing printing instructions. Machine codes and instruction cycle counts are generated by MASM for all instructions on each processor beginning with 8086. To assemble the file PROG.ASM use this command: (better to use DOS command line)

**MASM PROG.ASM**

The MASM program will assemble the PROG.ASM file. (To create PROG.OBJ from PROG.ASM)

To create PROG.EXE from PROG.OBJ, use this LINK command:

LINK PROG.OBJ

It converts the contents of PROG.OBJ into PROG.EXE.

ASSEMBLER DIRECTIVES: The limits are given to the assembler using some predefined alphabetical strings called Assembler Directives which help assembler to correctly understand. The assembly Language programs to prepare the codes.

|  |  |  |  |
| --- | --- | --- | --- |
| DB | GROUP |  | EXTRN |
| DW | LABEL |  | TYPE |
| DQ | LENGTH |  | EVEN |
| DT | LOCAL |  | SEGMENT |
| ASSUME | NAME |  |  |
| END | OFFSET |  |  |
| ENDP | ORG |  |  |
| ENDS | PROC |  |  |
| EQU | PTR |  |  |

**EXECUTION OF ASSEMBLY LANGUAGE PROGRAMMING IN MASM SOFTWARE:**

Assembly language programming has 4 steps.

1. Entering Program
2. Compile Program
3. Linking a Program
4. Debugging a Program

Cmd

**PROCEDURE:**

1.

Entering

Program:

-

Start

Menu

Run

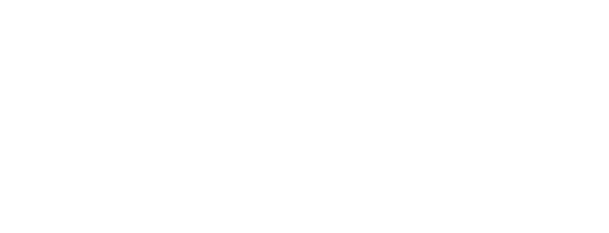
C:

\

MASM>

edit

filename.asm



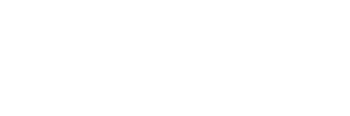
C:

\

MASM

\

filename.asm



This

is

editor

Enter

program

here

C:\cd MASM

After entering program save & exit (ALT F & Press S or ALT F &Press X)

C:\MASM>

1. Compile the Program:-

C:\MASM> MASM filename.asm

Microsoft @macro assembler version

5.10Copy rights reserved© Microsoft Corp 1981 All rights reservedObject filename

[OBJ];

List filename [NUL, LIST];

Cross Reference [NUL, CRF]; Press enter the screen shows c>

1. **Linking aProgram:-**

c> link filename.obj

Microsoft @ overlay linker version 3.64 Copy rights reserved© Microsoft corp.1983-88. All rights reserved

Object module [.OBJ]; Run file [.EXE];

List [NUL MAP];

Libraries [LIB];

Press enter till screen chows c>

1. Debug a Program:-

C> debug filename.exe

* + - (Screen shows only dash)
    - t

‘t’ for trace the program execution by single stepping starting from the address SEG.OFFSET. ‘q’ for Quit from Debug & return to DOS.

**Experiments No.2**

**AIM :- Programs for 16 bit arithmetic operations for 8086 (using Various Addressing Modes).**

**EQUIPMENT REQUIRED:**

* 1. TASM Software
  2. PC with DOS and Debug program

**ALGORITHM:**

* 1. Define the values in data segment as per the addressing mode.
  2. Initialize the data segment register with data segment address
  3. Load the words as per the addressing mode and perform addition/ subtraction/ multiplication/ division and store the sum/

difference/product/quotient-remainder to the result address 4. Terminate the program

**PROGRAMS (using register addressing mode):**

### A. 16 BIT ADDITION

|  |  |
| --- | --- |
| 1  2 | assume cs:code,ds:data |
| 3 0000 | data segment |
| 4 0000 1243 | n1 dw 1243h |
| 5 0002 4567 | n2 dw 4567h |
| 6 0004 ???? | n3 dw ? |
| 7 0006  8 | data ends |
| 9 0000  10 | code segment |
| 11 0000 | start: |
| 12 0000 B8 0000s | mov ax,data |
| 13 0003 8E D8  14 | mov ds,ax |
| 15 0005 A1 0000r | mov ax,n1 |
| 16 0008 8B 1E 0002r | mov bx,n2 |
| 17 000C 03 C3 | add ax,bx |
| 18 000E A3 0004r | mov n3,ax |
| 19 0011 BE 0004r | lea si,n3 |
| 20 0014 CC  21 | int 3 |
| 22 0015 | code ends |
| 23 | end start |

### B. 16 BIT SUBTRACTION

|  |  |
| --- | --- |
| 1  2 | assume cs:code,ds:data |
| 3 0000 | data segment |
| 4 0000 FFFF | n1 dw 0ffffh |
| 5 0002 4567 | n2 dw 4567h |
| 6 0004 ???? | n3 dw ? |
| 7 0006  8 | data ends |
| 9 0000  10 | code segment |
| 11 0000 | start: |
| 12 0000 B8 0000s | mov ax,data |
| 13 0003 8E D8  14 | mov ds,ax |
| 15 0005 A1 0000r | mov ax,n1 |
| 16 0008 8B 1E 0002r | mov bx,n2 |
| 17 000C 2B C3 | sub ax,bx |
| 18 000E A3 0004r | mov n3,ax |
| 19 0011 BE 0004r | lea si,n3 |
| 20 0014 CC  21 | int 3 |
| 22 0015 | code ends |
| 23 | end start |

#### C. 16 BIT MULTIPLICATION

|  |  |
| --- | --- |
| 1  2 | assume cs:code,ds:data |
| 3 0000 | data segment |
| 4 0000 4444 | n1 dw 4444h |
| 5 0002 4567 | n2 dw 4567h |
| 6 0004 ???????? | n3 dd ? |
| 7 0008  8 | data ends |
| 9 0000  10 | code segment |
| 11 0000 | start: |
| 12 0000 B8 0000s | mov ax,data |
| 13 0003 8E D8  14 | mov ds,ax |
| 15 0005 A1 0000r | mov ax,n1 |
| 16 0008 8B 1E 0002r | mov bx,n2 |
| 17 000C F7 E3 | mul bx |
| 18 000E BE 0004r | lea si,n3 |
| 19 0011 89 04 | mov [si],ax |
| 20 0013 89 54 02 | mov [si+2],dx |

21

22 0016 CC int 3

23

1. 0017 code ends
2. end start

#### D. WORD BY BYTE DIVISION

|  |  |
| --- | --- |
| 1  2 | assume cs:code,ds:data |
| 3 0000 | data segment |
| 4 0000 0444 | n1 dw 0444h |
| 5 0002 45 | n2 db 45h |
| 6 0003 ???? | n3 dw ? |
| 7 0005  8 | data ends |
| 9 0000 | code segment 10 |
| 11 0000 | start: |
| 12 0000 B8 0000s | mov ax,data |
| 13 0003 8E D8  14 | mov ds,ax |
| 15 0005 A1 0000r | mov ax,n1 |
| 16 0008 8A 1E 0002r | mov bl,n2 |
| 17 000C F6 F3  18 | div bl |
| 19 000E A3 0003r | mov n3,ax |
| 20 0011 BE 0003r  21 | lea si,n3 |
| 22 0014 CC  23 | int 3 |
| 24 0015 | code ends |
| 25 | end start |

**RESULT:**

**A. 16 BIT ADDITION**

AX= 57AA & SI=0004 ; D 0004 0005 AA 57

### B. 16 BIT SUBTRACTION

AX= BA98 & SI=0004 ; D 0004 0005 98 BA

1. **16 BIT MULTIPLICATION**

AX= CB5C & SI=0004 ; D 0000 0005 44 44 67 45 5C CB

1. **WORD BY BYTE DIVISION**

AX= 390F & SI=0003 ; D 0000 0004 44 04 45 0F 39

## Experiments No.3

### AIM: -PROGRAM FOR SORTING AN ARRAY FOR 8086

**EQUIPMENT REQUIRED:**

1. TASM Software
2. PC with DOS and Debug program

**ALGORITHM:**

1. Define the values in data segment
2. Initialize the data segment register with data segment address
3. Clear the various registers
4. Initialize outer counter for arranging the given numbers
5. Initialize inner counter for performing comparisons
6. Compare the first two values, if carry is generated then continue for next values 7. Otherwise, exchange both values and continue for next values
7. Continue from step 5 till the count is zero.
8. Terminate the program

**PROGRAM:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 3  4 | 1  0000  0000 0198 0135 |  | 0234 0098 | assume cs:code,ds:data data segment n1 dw 198h,135h,234h,098h |

1. 0008 0A\*(0000)
2. =0003 7
3. 001C
4. 0000
5. 0000
6. 0000 B8 0000s
7. 0003 8E D8

13

14 0005 33 C0 15 0007 33 D2

16 0009 33 C9

17

1. 000B BA 0003
2. 000E B9 0003
3. 0011 BE 0000r

res dw 10 dup(0) count equ 3

data ends

code segment start: mov ax,data mov ds,ax

xor ax,ax xor dx,dx xor cx,cx

mov dx,count

x1:mov cx,count lea si,n1

1. 0014 8B 04 mov ax,[si]
2. 0016 3B 44 02 x:cmp ax,[si+2]
3. 0019 72 05 jc l1
4. 001B 87 44 02 xchg ax,[si+2]
5. 001E 89 04 mov [si],ax
6. 0020 46 l1:inc si
7. 0021 46 inc si
8. 0022 8B 04 mov ax,[si]
9. 0024 E2 F0 loop x 30 0026 4A dec dx

31 0027 75 E5 jnz x1

32

33 0029 BE 0000r lea si,n1

34

1. 002C CC int 3h
2. 002D code ends
3. end start

**RESULT:**

AX=0234, SI=0000

D 0000 0007 98 00 35 01 98 01 34 02

**Experiments No.4**

**AIM:- Program for searching for a number or character in a string for 8086.**

**EQUIPMENT REQUIRED:**

1. TASM Software
2. PC with DOS and Debug program

**ALGORITHM:**

1. Define the values in data segment
2. Initialize the data segment register with data segment address
3. Clear all the various registers
4. Initialize the counter for number of comparisons
5. Compare the input with the numbers in an array one at a time
6. If zero flag is set, display the message ‘number found’
7. If zero flag is not set even after all the comparisons i.e., till the count is zero then display the message ‘number not found’
8. Terminate the program

**PROGRAM:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  |  | assume cs:code,ds:data |
| 2  3 | 0000 |  | data segment |
| 4 | 0000 12CD 3BCD | 34CD | n1 dw 12cdh,3bcdh,34cdh |
| 5 | 0006 04CD |  | n2 dw 04cdh |

|  |  |
| --- | --- |
| 1. 0008 70 61 73 73 77 6F 72+ msg1 db "number found   $"   1. 64 20 66 6F 75 6E 64+ 2. 20 24 3. 0018 70 61 73 73 77 6F 72+ msg2 db "number not found $" 4. 64 20 6E 6F 74 2066+ | |
| 11 6F 75 6E 64 20 24 |  |
| 12 =0003 | count equ 3 |
| 13 0 2C  14 | data ends |
| 15 0000 | code segment |
| 16 0000 | start: |
| 17 0000 B8 0000s | mov ax,data |
| 18 0003 8E D8 | mov ds,ax |
| 19 0005 33 C0 | xor ax,ax |
| 1. 0007 33 D2 2. 0009 33 C9 3. 000B B9 0003 4. 000E BE 0000r      1. 0011 A1 0006r 2. 0014 3B 04 3. 0016 74 0E 4. 0018 46 5. 0019 46 6. 001A E2 F8 7. 001C B4 09 8. 001E BA 0018r 9. 0021 CD 21 10. 0023 EB 08 90 11. 0026 B4 09 12. 0028 BA 0008r 13. 002B CD 21 14. 002D CC 15. 002E   39 | xor dx,dx xor cx,cx  mov cx,count lea si,n1 mov ax,n2 l1:cmp ax,[si] jz l2  inc si inc si  loop l1 mov ah,09h lea dx,msg2 int 21h jmp l3 l2:mov ah,09h lea dx,msg1 int 21h l3:int 3h code ends end start |

**RESULT:**

Input of 04cdh, the message password found is displayed.

Input of 2340h, the message password not found is displayed.

## Experiments No.5

**AIM:- Program to move the string from source location to destination location.( MOVE THE STRING)**

**EQUIPMENT REQUIRED:**

1. TASM Software
2. PC with DOS and Debug program

**ALGORITHM:**

1. Define the string to be displayed in data segment
2. Initialize the data segment register with data segment address
3. Initialize the source pointer to the starting address of defined string
4. Initialize the destination pointer to the location where the string is to be stored
5. Move the contents from the source to the destination till the ‘$’ (termination character) is found
6. Display the string from the destination location
7. Terminate the program

**PROGRAM:**

|  |  |  |
| --- | --- | --- |
| 1  2 |  | assume cs:code,ds:data |
| 3 0000  4 |  | data segment |
| 1. 0000 20 68 65 6C 6C 6F 2. 65 76 65 72 79 6F 6E+ 3. 65 24   8 | 20+ | msg db " hello everyone","$" |
| 9 0010 0A 0D 24 10 |  | nline db 10,13,'$' |
| 11 0013 28\*(24)  12 |  | msg1 db 40 dup('$') |
| 13 003B  14  15  16 |  | data ends |
| 17 0000  18 |  | code segment |
| 19 0000 B8 0000s |  | start:mov ax,data |
| 20 0003 8E D8 |  | mov ds,ax |

|  |  |
| --- | --- |
| 21  22 0005 B4 09 | mov ah,09h |
| 23 0007 BA 0000r | lea dx,msg |
| 24 000A CD 21 | int 21h |
| 25  26 000C BE 0000r | lea si,msg |
| 27 000F BF 0013r | lea di,msg1 |
| 28  29 0012 8A 04 | l1: mov al,[si] |
| 30 0014 3C 24 | cmp al,'$' |
| 31 0016 74 06 | je l2 |
| 32 0018 88 05 | mov [di],al |
| 33 001A 46 | inc si |
| 34 001B 47 | inc di |
| 35 001C EB F4 | jmp l1 |
| 36  37 001E B4 09 | l2:mov ah,09h |
| 38 0020 BA 0010r | lea dx,nline |
| 39 0023 CD 21 | int 21h |
| 40  41 0025 B4 09 | mov ah,09h |
| 42 0027 BA 0013r | lea dx,msg1 |
| 43 002A CD 21 | int 21h |
| 44  45 002C B4 4C | mov ah,4ch |
| 46 002E CD 21 | int 21h |
| 47 48 0030 | code ends |
| 49    **RESULT:** hello everyone | end start |

**Experiments No.6: Interfacing ADC&DAC to 8086.**

**AIM:** Write an ALP to convert the analog signal into its equivalent digital form.

**EQUIPMENT REQUIRED:**

* + 1. 8086 kit
    2. A to D converter interfacing card
    3. Flat ribbon cable bus 4. Power supply to 8086 kit

5. Jumper.

**HARDWARE CONNECTIONS REQUIRED:**

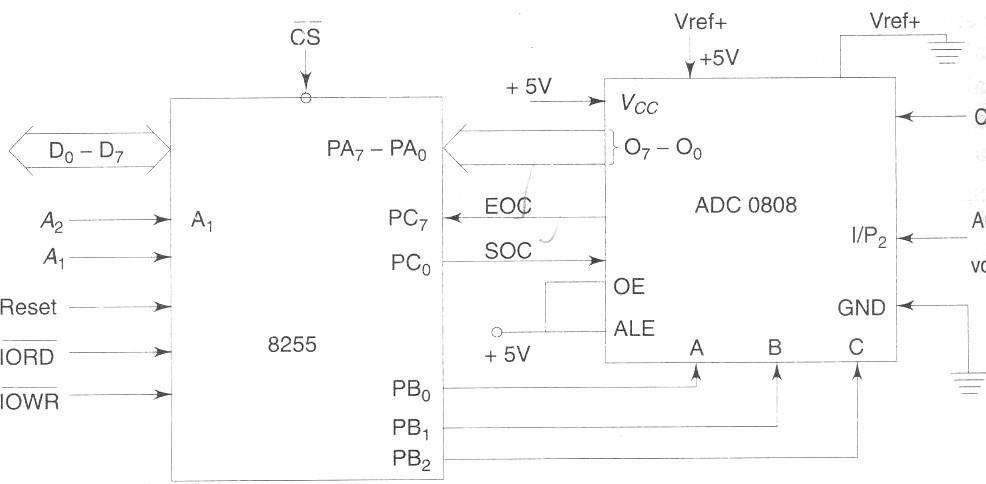
* 1. connect J2 to provide 8 channels of ADC which are selected by address supplied by port B (J3) and latched by Pc4 bit.

* 1. This port B is read port of ADC while Pc1 (lower port C) is input while Pc4,5,6 (upper port C) is output commands.

* 1. To experiment use on board potentiometer as voltage source by shorting 7J1 & 8J1.

**INTERFACING CIRCUIT:**

**ADC 0808 WITH 8086 THROUGH 8255:**

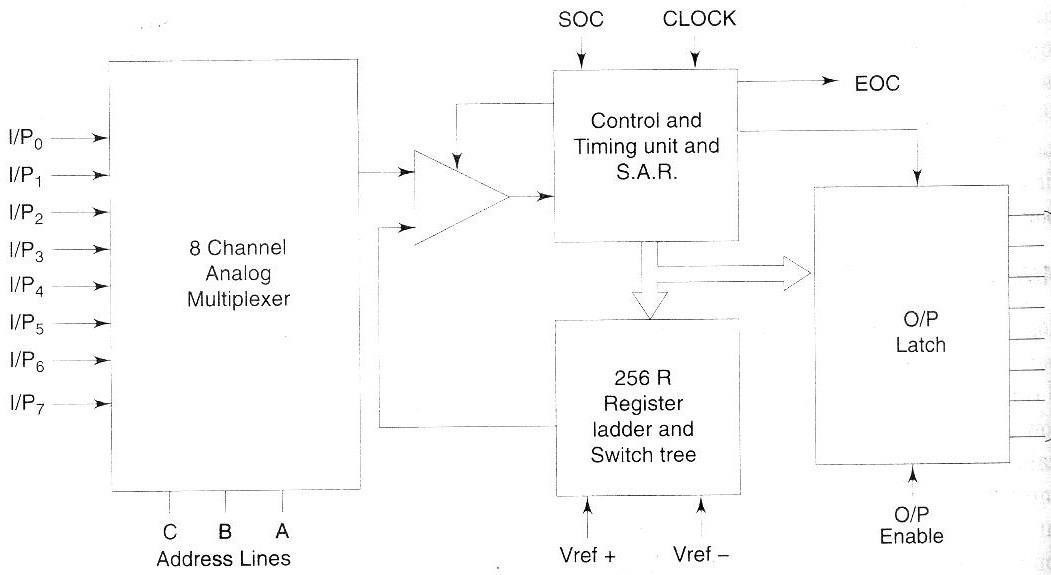


**DESCRIPTION ABOUT IC’s:**

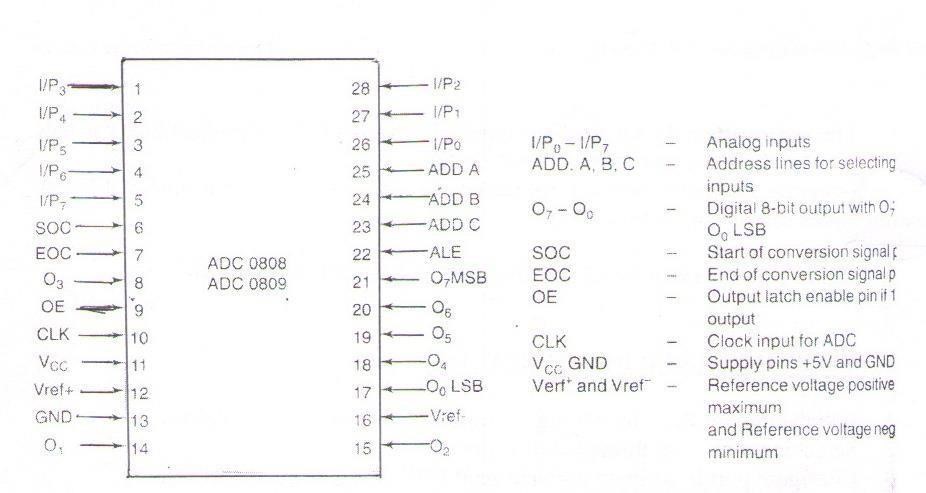
**ADC 0808:**

\* The ADC chips 0808 are 8 bit CMOS, successive approximation converters.

Block diagram of ADC



Pin diagram of ADC 0808/0809



**PROGRAM:**

MOV AL, 81

MOV DX, 8807 ; Configuring ports as output ports except port C OUT DX, AL

MOV AL, 00

MOV DX, 8803 ; Sending channel addr on port B OUT DX, AL

MOV AL, 08

MOV DX, 8807 ; Generate ALE signal on PC3 OUT DX, AL

MOV AL, 09

MOV DX, 8803

OUT DX, AL ; Configure port B as input port

MOV AL, 0C ; Generate start of conversion pulse on PC6

OUT DX, AL

MOV AL, 0D

OUT DX, AL

MOV AL, 0C

OUT DX, AL

MOV DX, 8805

Above: IN AL,DX ; Read End of conversion on PC1

AND AL,02

JZ Above

MOV AL, 0B

MOV DX, 8807; Set O/P enable signal high

OUT DX, AL

MOV AL, 8803; Read the status from AL register IN AL, DX

INT 3 ; TERMINATE

**RESULT:** When potentiometer was in minimum position the digital output is 00 and when maximum output at AL is FF.

**Experiments No.7: Interfacing stepper to 8086.**

**AIM:** Program to interface the stepper motor to 8086 microprocessor

**EQUIPMENT REQUIRED:**

1. 8086 microprocessor trainer kit
2. Stepper motor interfacing module
3. Flat ribbon cable bus
4. Keyboard
5. Power chord

**HARDWARE CONNECTIONS REQUIRED:**

1. Connect P3 on 86M to the connector C1 on the interfacing using a 26 core flat cable.
2. Motor is a Z phase, 6 wire motor.
3. Power connections:

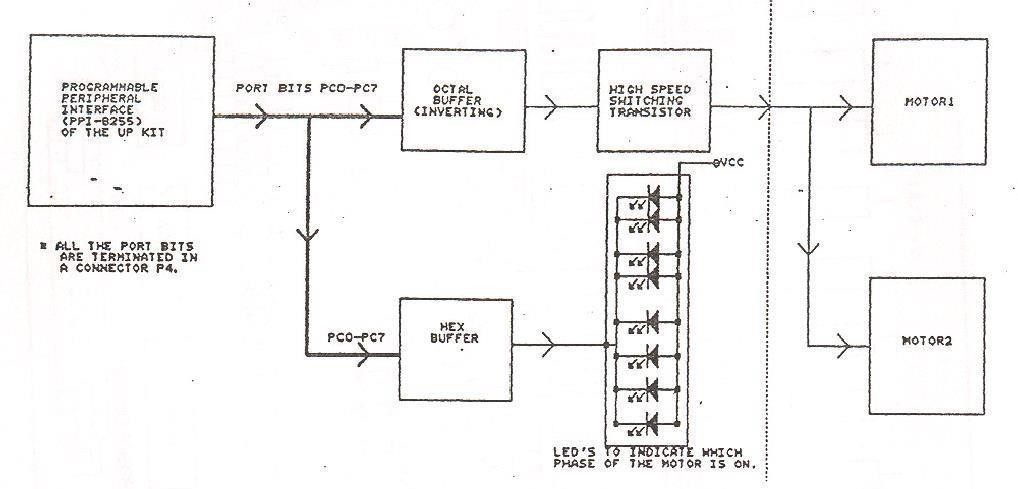
A 4-Way power mate female is provided with wires.

White/blue/orange - +5V

Black & Red - COM (GND)

Green - +12V or 6V

**INTERFACING CIRCUIT:**



**PROGRAM 2 :**

2000 B0 80 MOV AL,80 ; Initialize 8255

2002 BA C6 FF MOV DX,FFC6

1. EE OUT DX,AL

1. B0 EE START:MOV AL,EE ; Byte to switch on A phase

2008 BA C4 FF MOV DX,FFC4

200B EE OUT DX,AL

200C E8 F1 00 CALL DELAY ; Wait

200F D0 C0 ROL AL,1 ; Rotate phases

2011 EB F8 JMP L1 2013 CC INT 3

2100 B9 FF FF DELAY:MOV CX,0FFFF ; Initialize counter(Hex value)

1. 90 L1: NOP ; No operation
2. 90 NOP
3. 90 NOP
4. 49 DEC CX
5. 75 FA JNZ L1

2109 C3 RET

**RESULTS:**

Observed the stepper motor rotation

**Experiments No.8: Program Using Arithmetic, Logical, and Bit Manipulation Instructions Of 8051.**

**AIM:** Program using arithmetic, logical and bit manipulation instructions of 8051.

**EQUIPMENT REQUIRED:**

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC,
4. Keil µ Vision 3 software

**PROCEDURE:**

1. Write the assembly language program for this task in the keil compiler
2. Load the two values into two registers
3. Perform the required operation using corresponding operator
4. End the program

**PROGRAM:**

**A. ARITHMETIC OPERATIONS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADDITION**: C:0X0000 |  | 7405 |  | MOV A,#0X05 |
| C:0X0002 |  | 74F005 |  | MOV B(0XF0),#0X05 |
| C:0X0005 |  | 25F0 |  | ADD A,B(0XF0) END |
| **SUBSTRACTION:** C:0X0000 |  | 7405 |  | MOV A,#0X05 |
| C:0X0002 |  | 75F002 |  | MOV B,B(0XF0),#0X02 |
| C:0X0005 |  | 95F0 |  | ADD A,B(0XF0) |

END

**MULTIPLICATION:**

|  |  |  |  |
| --- | --- | --- | --- |
| C:0X0000 7402 | |  | MOV A,#0X05 |
| C:0X0002 | 74F003 |  | MOV B(0XF0),#0X03 |
| C:0X0005 | AA |  | MUL AB END |
| **DIVISION:** C:0X0000 | 7409 |  | MOV A,#0X09 |
| C:0X0002 | 75F003 |  | MOV B(0XF0),#0X03 |
| C:0X0005 | 84 |  | DIV AB END |

|  |  |  |  |
| --- | --- | --- | --- |
| **RESULT:** ADDITION: A=05  SUBTRACTION:A=03  MULTIPLICATION:A=0F  DIVISION:A=03    **B.LOGICAL OPERATIONS**  **OR LOGIC:**  C:0X0000 7403 | |  | MOV A,#0X03 |
| C:0X0002 | 74F002 |  | MOV B(0XF0),#0X02 |
| C:0X0005 | 45F0 |  | ORL A,B(0XF0) END |
| **AND LOGIC:** C:0X0000 | 7404 |  | MOV A,#0X04 |
| C:0X0002 | 75F002 |  | MOV B(0XF0),#0X02 |
| C:0X0005 | 55F0 |  | ANL A,B(0XF0) END |
| **XOR LOGIC:** C:0X0000 | 7404 |  | MOV A,#0X04 |
| C:0X0002 | 75F002 |  | MOV B(0XF0),#0X02 |
| C:0X0005 | 65F0 |  | XRL A,B(0XF0) END |
| **RESULT:**  OR LOGIC: A=03  AND LOGIC: A=00  XOR LOGIC: A=06    **BIT MANIPULATION OPERATORS**  **ROTATE RIGHT:**  C:0X0000 740F  C:0X0002 03 | |  | MOV A,#0X0F  RR A  END |
| **ROTATE RIGHT WITH CARRY:**  C:0X0000 7405 | |  | MOV A,#0X05 |
| C:0X0002 13 | |  | RRC A END |
| **ROTATE LEFT:**  C:0X0000 7405 | |  | MOV A,#0X05 |
| C:0X0002 23 | |  | RL A END |
| **ROTATE LEFT WITH CARRY:**  C:0X0000 740C | |  | MOV A,#0X0C |
| C:0X0002 33 | |  | RLC A END |
| **SWAP:**  C:0X0000 7408 | |  | MOV A,#0X08 |

|  |  |
| --- | --- |
| C:0X0002 14    **RESULT:**  ROTATE RIGHT:A=07  ROTATE RIGHT WITH CARRY:A=02  ROTATE LEFT:A=0A  ROTATE LEFT WITH CARRY:A=18 SWAP:A=80 | SWAP A END |

**Experiments-9: Timer/Counters in 8051.**

**AIM:** To create the square wave of 50% duty cycle on the p1.5 bit. Timer 0 is used to generate the time delay.

**EQUIPMENT REQUIRED:**

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

**PROCEDURE:**

1. Write the assembly language program for this task in the keil compiler.
2. Load the delay values into timer 0
3. Complement the port line and start timer
4. If timer is complete, complement the port line and again start the timer by reloading values
5. This is repeated continuously

**PROGRAM:**

|  |  |  |
| --- | --- | --- |
| MOV TMOD,#01 |  | ;Timer 0, mode 1(16-bit mode) |
| HERE: MOV TL0,#0F2H |  | ;TL0 =F2H, the Low byte |
| MOV TH0,#0FFH |  | ;TH0 =FFH, the High byte |
| CPL P1.5  A CALL DELAY |  | ; Toggle P1.5 |
| SJMP HERE    ;… .............. delay using Timer 0  DELAY: |  | ; load TH, TL again |
| SET B TR0  AGAIN: JNB TF0, AGAIN |  | ; Start Timer 0  ; Monitor Timer 0 Flag Until  ; It rolls over |
| CLR TR0 |  | ; Stop Timer 0 |
| CLR TF0 |  | ; Clear Timer 0 Flag |
| RET |  |  |

**RESULT:** As the timer 0 rolls over ,the port line p1.5 status toggles continuously.

**Experiments-10: PROGRAM AND VERIFY INTERRUPT**

## HANDLING IN 8051

**AIM:** Write a program to generate a square wave of 50 Hz frequency on pin P1.2. Use an interrupt for timer 0. Assume the XTAL =11.0592 MHz.

**EQUIPMENT REQUIRED:**

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

**PROCEDURE:**

1. Write the assembly language program for this task in the keil compiler.
2. Load the delay values into timer 0
3. Complement the port line and start timer
4. If timer is complete, the corresponding timer interrupt is generated, complement the port line and again start the timer by reloading values
5. This is repeated continuously

**PROGRAM:**

ORG 0

LJMP MAIN

ORG 000BH ; ISR for timer0

CPL P1.2 ; complement p1.2

MOV TL0,#00 ;reload timer values

MOV TH0,#0DCH

RETI ; return from interrupt

ORG 30H ; main program for initialization

|  |  |
| --- | --- |
| MAIN: | MOV TMOD,#01H ;timer 0, mode 1  MOV TL0,#00H  MOV TH0,#0D2H  MOV IE,#82H ;enable timer 0 interrrupt  SETB TR0 ; start timer |
| HERE: | SJMP HERE ; stay here until interrupted END |
|  | **RESULT:**  The functionality of timer interrupts is verified. |

**Experiments-11: PROGRAM FOR UART OPERATION IN 8051.**

**AIM:** To send the data serially out from microcontroller to pc.

**EQUIPMENT REQUIRED:**

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC,
4. Keil µ Vision

3 software

20

|  |  |  |
| --- | --- | --- |
| **PROGRAM:**      RELOAD)  MODE 1 | ORG 00H  MOV TMOD,#20H  SCON,#50H | ;TO SELECT TIMER 1 IN MODE 2(AUTO MOV  ; TO SELECT SERIAL COMMUNICATION IN  ; AND RECEIVER ENABLE |
|  | MOV TH1,#0FDH DPTR,#MSG | ; TO SET THE BAUD RATE TO 9600 MOV |
| BACK: | CLR A |  |

MOVC A,@A+DPTR ; TO LOAD THE MESSAGE INTO

ACCUMULATOR

; CHARACTER BY CHARACTER

JZ NEXT

ACALL TRANSMIT ; CALL SUBROUTINE FOR TRANSMISSION INC

DPTR

SJMP BACK

NEXT: SJMP NEXT ; TERMINATION OF THE PROGRAM

/\*TRANSFERRING DATA SERIALLY\*/

|  |  |
| --- | --- |
| TRANSMIT: SETB TR1 | ; TO START THE TIMER |
| MOV SBUF,A | ; LOAD DATA INTO SBUF |
| JNB TI,$ | ; WAIT FOR THE TRANSMISSION TO BE COMPLETED |
| CLR TI | ; CLEAR TI FOR NEXT TRANSMISSION RET |

MSG: DB 13,10,"WELCOME TO ALL",13,10,0 END

**RESULT:**

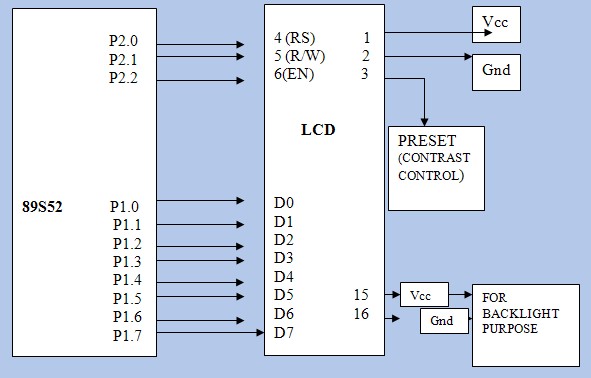
The data bytes is transmitted serially out and is displayed on hyper-terminal of PC.

**Experiments-12: Interfacing LCD to 8051. AIM:** To display message on the LCD.

**EQUIPMENT REQUIRED:**

1. 8051 Trainer kit
2. RS232 cable
3. Max 232 IC
4. Keil µ Vision 3 software

**HARDWARE CONNECTIONS REQUIRED:**



LCD CONTROL PINS- RS-P3.5

RW-P3.6

EN-P3.7

LCD DATA PINS- PORT 2(P2)

**PROGRAM:**

|  |  |  |  |
| --- | --- | --- | --- |
| PINS |  | RS11 EQU P2.0  RW1 EQU P2.1 | ;ASSIGNING NAMES TO THE PORT |

EN EQU P2.2

ORG 00H

MOV DPTR,#COMM

STAY

:

CLR A

M

O

V C

A

,

@

A

+

D

P T

R

J

Z

M

E

S

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SUBROUTINE | ACALL COMMAND  INC DPTR  SJMP STAY | | ; CALL | COMMAND |
| MESG: | MOV DPTR, #MSG  ACALL DISPLAY  ACALL DELAY  MOV A, #0C2H  ACALL COMMAND  MOV DPTR,#MSG1  ACALL DISPLAY  ACALL DELAY | |  |  |
| HERE1: | SJMP HERE1 | | ; STAY HERE |  |
| DISPLAY: | CLR A  MOVC A,@A+DPTR  JZ XX  ACALL DATA1 INC DPTR  SJMP DISPLAY | | ; CALL DATA SUBROUTINE | |
| XX: | RET | |  | |
| COMMAND: |  | | ; SEND COMMAND TO LCD | |
|  | MOV P1, A | | ; SEND COMMAND TO PORT | |
|  | CLR RS11 | | ; RS=0 FOR COMMAND | |
|  | CLR RW1 | | ; RW=O FRO WRITE OPERATION | |
|  | SETB EN | | ; EN=1 FOR HIGH PULSE | |
|  | ACALL DELAY | | ; GIVE LCD SOME TIME | |
|  | CLR EN | | ; EN=0 FOR HIGH TO LOW | |
| PULSE | RET | |  | |
| DATA 1 : |  | ; SEND DATA TO LCD | | |
|  | MOV P1, A ; SEND DATA TO PORT | | | |

|  |
| --- |
| SETB RS11 ; RS=1 FOR DATA  CLR RW1 ; RW=0 FOR WRITE  OPERATIONSETB EN ; EN=1 FOR HIGH  PULSE ACALL DELAY ; GIVE LCD SOME  TIME  CLR EN ; EN=0 FOR HIGH TO LOW  PULSERET  DELAY: MOV R5, #255 HERE: MOV R6,  #255  DJNZ R6,  $ DJNZ  R5, HERE  RET  COMM: DB 38H, 0CH, 01H, 06H, 84H, 0 ;  COMMANDSMSG: DB  "WINEYARD", 0 ; DATA MSG1: DB  "TECHNOLOGIES", 0 ; DATA  END    **RESULT:**  The message displayed on the LCD screen is observed.                                        24    25 |