SCHOOL OF COMPUTER SCIENCE & ENGINEERING

SRM UNIVERSITY,

SRM NAGAR, KATTANKULATHUR-603203.

SUBJECT : **MICROPROCESSORS AND MICROCONTROLLERS**

SUBJECT CODE : 15CS205J **SEMESTER** : VI

CLASS : III CSE **HOURS/WEEK** : 2 HOURS

EX. NO	**NAME OF THE EXPERIMENTS**
Sl. No.	**Description of experiments**
	Assembly Language Programs Using TASM/MASM
1	Program involving Arithmetic Instructions on 16 bit data i. Addition & Subtraction ii. Multiplication & Division iii. Factorial of a given number
2	Program involving Data Transfer Instructions on 16 bit data i. Byte and Word data transfer in different addressing modes ii. Block Data Transfer
3	Program involving Bit Manipulation Instructions on 16 bit data -Given data is positive or negative
4	Implementation of Bubble Sort Algorithm

5	Program involving String Instructions on 16 bit data i. Reverse a given string and check whether it is a palindrome ii. String Display using Display Interrupt (Read your name from the keyboard and displays it at a specified location on the screen after the message "What is your name?" You must clear the entire screen before display)
6.	Time display using Interrupt (Read the current time from the system and display it in the standard format on the screen)
.	**Basic 8051 programming using C**
7	Port Programming
8	Timer-Counter Programming
9	Serial Programming
10	Interrupt Programming

PREPARED BY: **APPROVED BY,**

M.Karthikeyan **(HOD / CSE)**

<u>Steps to use MASM</u>

MASM:

1. TYPE IN NOTEPAD AND SAVE IT AS FILENAME.ASM(CLICK ALL FILES).
2. EDIT : EDIT FN.ASM
3. MASM FN.ASM
4. LINK FN.OBJ
5. DEBUG FN.EXE

```
-R
-T
-T
```

8086 PROGRAM:

ExNo:1 i) Program involving Arithmetic Instructions on 16 bit data

Addition

AIM: To implement assembly language program for addition of two 16-bit numbers.

APPARTUS: MASM Software, P.C.

PROGRAM:

```
DATA SEGMENT
N1 DW 1234H
N2 DW 2134H
RES DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
MOV DS, AX
MOV AX, N1
MOV BX, N2
ADD AX, BX
MOV RES, AX
INT 21H
CODE ENDS
END START
```

RESULT:
AX = 3368h

ExNo:1 i) Program involving Arithmetic Instructions on 16 bit data

Subtraction

AIM: To implement assembly language program for subtraction of two 16-bit numbers.

APPARTUS: MASM Software, P.C.

PROGRAM:
```
DATA SEGMENT
N1 DW 4444H
N2 DW 2121H
RES DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AX,N1
MOV BX,N2
SUB AX,BX
MOV RES,AX
INT 21H
CODE ENDS
END START
```

RESULT:
AX = 2323h

ExNo:1 ii) Program involving Arithmetic Instructions on 16 bit data

Multiplication

AIM: To implement assembly language program for Multiplication of two 16-bit numbers.

APPARTUS: MASM Software, P.C.

PROGRAM:

```
ASSUME CS : CODE, DS : DATA
CODE SEGMENT
MOV AX, DATA
MOV DS, AX
MOV AX, OPR1
MUL OPR2
MOV RESLW, AX
MOV RESHW, DX
HLT
CODE ENDS
DATA SEGMENT
OPR1 DW 2000H
OPR2  DW 4000H
RESLW  DW ?
RESHW  DW ?
DATA ENDS
END
 (DX)
```

Input:

```
OPR1 = 2000H
OPR2 = 4000H
```

Output:

```
RESLW = 0000H (AX)
RESHW = 0800H (DX)
```

ExNo:1 ii) Program involving Arithmetic Instructions on 16 bit data

Division

AIM: To implement assembly language program for Division of two 16-bit numbers.

APPARTUS: MASM Software, P.C.

Program:

```
ASSUME CS : CODE, DS : DATA

CODE SEGMENT

MOV AX, DATA

MOV DS, AX

MOV AX, OPR1

DIV OPR2

MOV RESQ, AL

MOV RESR, AH

HLT

CODE ENDS

DATA SEGMENT

OPR1 DW 2C58H

OPR2 DB 56H

RESQ DB ?

RESR DB ?

DATA ENDS

END
```

Input:

OPR1 = 2C58H (DIVIDEND)

OPR2 = 56H (DIVISOR) **Output:** RESQ = 84H (AL) RESR = 00H (AH)

ExNo:1 iii) Factorial of a given number

AIM: To implement assembly language program to find factorial of a given number

APPARTUS: MASM Software, P.C.

Program:

```
DATA SEGMENT
X DW 06H
FACT DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AX,01H
MOV CX,X
UP: MUL CX
LOOP UP
MOV FACT,AX
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

Input: 06

Output: 2D0H

ExNo:2 i) Byte and Word data transfer in different addressing modes

AIM: To implement assembly language program for Byte and Word data transfer in different addressing modes

APPARTUS: MASM Software, P.C.

Program:

```
 DATA SEGMENT
DATA1 DB 23H
DATA2 DW 1234H
DATA3 DB 0H
DATA4 DW 0H
DATA5 DW 2345H,6789H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA ;  //Initialize DS to point to start of the memory
MOV DS,AX ; //set aside for storing of data
MOV AL,25X ; //copy 25H into 8 bit AL register
MOV AX,2345H ;// copy 2345H into 16 bit AX register
MOV BX,AX ;// copy the content of AX into BX register(16 bit)
MOV CL,AL ;// copy the content of AL into CL register
MOV AL,DATA1 ;// copies the byte contents of data segment memory
                            location DATA1 into 8 bit AL
MOV AX,DATA2 ;// copies the word contents of data segment memory
                            ;location DATA2 into 16 bit AX
MOV DATA3,AL ;// copies the AL content into the byte contents of data
                           ;segment memory location DATA3
MOV DATA4,AX ;//copies the AX content into the word contents of
                            ;data segment memory location DATA4
MOV BX,OFFSET DATA5 ;// The 16 bit offset address of DS memeory
location
                                       ; DATA5 is copied into BX
MOV AX,[BX] ; //copies the word content of data segment
                        ;memory location addressed by BX into
                           ;AX(register indirect addressing)
MOV DI,02H ;         address element
MOV AX,[BX+DI} ;       copies the word content of data segment
                                  ;memory location addressed by BX+DI into
                                  ;AX(base plus indirect addressing)
```

```
MOV AX,[BX+0002H] ;   copies the word content of data segment
                      ;(16 bit)
MOV AL,[DI+2] ;       register relative addressing
MOV AX,[BX+DI+0002H] ;          copies the word content of data
segment
                      ;memory location addressed by BX+DI+0002H
                           ;into AX(16 bit)
MOV AH,4CH ;               Exit to DOS with function call 4CH
    INT 21H
CODE ENDS ;                Assembler stop reading
END START
```

ExNo:2 ii) Block Data Transfer

AIM: To implement assembly language program for Block Data Transfer

APPARTUS: MASM Software, P.C.

Block move (with and with out overlapping)

Without overlapping

Program:

```
DATA SEGMENT
X DB 01H,02H,03H,04H,05H ;          Initialize Data Segments Memory
Locations
Y DB 05 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA ;                 Initialize DS to point to start of the
memory
MOV DS,AX ;                         set aside for storing of data
MOV CX,05H ;                        Load counter
LEA SI,X+04 ;                   SI pointer pointed to top of the memory block
LEA DI,X+04+03 ; 03              is displacement of over lapping, DI pointed to
                                ;the top of the destination block
```

Before execution

00
00
00
00
00
05
04
03
02
01

After execution

05
04
03
02
01
05
04
03
02
01
Y,DI
X, SI

With Overlapping
Program:

```
DATA SEGMENT
X DB 01H,02H,03H,04H,05H ;          Initialize Data Segments Memory
Locations
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA ;                 Initialize DS to point to start of the
memory
MOV DS,AX ;                         set aside for storing of data
MOV CX,05H ;                        Load counter
LEA SI,X+04 ;                   SI pointer pointed to top of the memory block
LEA DI,X+04+03 ;                03 is displacement of over lapping, DI pointed
to
                                ;the top of the destination block
UP: MOV BL,[SI] ;           Move the SI content to BL register
MOV [DI],BL ;                    Move the BL register to content of DI
DEC SI ;                        Update SI and DI
DEC DI
DEC CX ;                         Decrement the counter till it becomes zero
JNZ UP
MOV AH,4CH
INT 21H
CODE ENDS
```

END START
DS Before execution
xx
xx
xx
xx
xx
05
04
03
02
01
DI
SI
X
DS After execution
xx
xx
05
04
03
- 02
01
03
02
01
-

ExNo:3 Program involving bit manipulation instruction

AIM: To implement assembly language program involving bit manipulation instruction If given data is positive or negative

APPARTUS: MASM Software, P.C.

Program:

```
DATA SEGMENT
NUM DB 12H
MES1 DB 10,13,'DATA IS POSITIVE $'
MES2 DB 10,13,'DATA IS NEGATIVE $'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AL,NUM
ROL AL,1
JC NEGA
;Move the Number to AL.
;Perform the rotate left side for 1 bit position.
;Check for the negative number.
MOV DX,OFFSET MES1 ;Declare it positive.
JMP EXIT ;Exit program.
NEGA: MOV DX,OFFSET MES2;Declare it negative.
EXIT: MOV AH,09H
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
```

```
END START
```

Output: Data is positive
Positive Numbers: 00-7F
Negative numbers: 80-FF

ExNo:4   Bubble sort using MASM

AIM: To implement assembly language program for Bubble Sort

APPARTUS: MASM Software, P.C.

Program:

Ascending Order:

```
model small

.data
arr db 5h,7h,6h,4h,10h,09h
len db $-arr

.code
start: mov ax,@data
       mov ds,ax
       mov cl,len
lp1:   mov bx,cx
       lea si,arr
lp2:   mov al,[si]
       inc si
       cmp [si],al
       jb lp3
       xchg [si],al
       mov [si-1],al
lp3:   dec bx
       jnz lp2
       loop lp1
       mov ah,4ch
       int 21h
```

```
end start
```

Descending Order:

```
.model small

.data
arr db 5h,7h,6h,4h,10h,09h
len db $-arr

.code
start: mov ax,@data
       mov ds,ax
       mov cl,len
lp1:   mov bx,cx
       lea si,arr
lp2:   mov al,[si]
       inc si
       cmp [si],al
       jb lp3
       xchg [si],al
       mov [si-1],al
lp3:   dec bx
       jnz lp2
       loop lp1
       mov ah,4ch
       int 21h
end start
```

ExNo:5 i) Reverse a given string and check whether it is a palindrome

AIM: To implement assembly language program to check whether the string is a palindrome.

APPARTUS: MASM Software, P.C.

Program:

```
DATA SEGMENT
STR1 DB 'LIRIL'
LEN EQU $-STR1
STR2 DB 20 DUP(0)
;start of data segment
MES1 DB 10,13,'WORD IS PALINDROME$'
MES2 DB 10,13,'WORD IS NOT PALINDROME$'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA,ES:DATA
START: MOV AX,DATA
MOV DS,AX
MOV ES,AX
LEA SI,STR1
LEA DI,STR2+LEN-1
MOV CX,LEN
UP: CLD
LODSB
STD
STOSB
LOOP UP
LEA SI,STR1
LEA DI,STR2
CLD
MOV CX,LEN
REPE CMPSB
```

```
CMP CX,0H
JNZ NOTPALIN
LEA DX,MES1
MOV AH,09H
INT 21H
JMP EXIT
NOTPALIN: LEA DX,MES2
MOV AH,09H
INT 21H
EXIT: MOV AH,4CH
INT 21H
CODE ENDS
END START
```

```
OUTPUT: WORD IS PALINDROME
```

ExNo:5 ii) Program to use DOS interrupt INT 21H function called for reading a character from keyboard, buffered keyboard input, display of character and string on console.

AIM: To implement assembly language program to display a String

APPARTUS: MASM Software, P.C.

Program:

```
DATA SEGMENT
INKEY DB ?
BUF DB 20 DUP(0)
MES DB 10,13, ' SRM UNIVERSITY $'
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE , DS:DATA
START: MOV AX,DATA
MOV DS,AX
MOV AH,01H
INT 21H
```

```
;DOS function to read a character from keyboard ;with
echo. [AL = 8bit character]
MOV INKEY,AL ;Returns ASCII value of the pressed key.
MOV BUF,10
MOV AH,0AH
LEA DX,BUF
INT 21H
MOV AH,06H
MOV DL,'A'
INT 21H
MOV AH,09H
LEA DX,MES
INT 21H
MOV AH,4CH
INT 21H
CODE ENDS
END START
```

ExNo:6 Time display using Interrupt (Read the current time from the system and display it in the standard format on the screen)

AIM: To implement assembly language program to display a system time

APPARTUS: MASM Software, P.C.

Program:

```
DATA SEGMENT
HOUR DB ?
MIN DB ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
DISCHAR MACRO CHAR
PUSH AX
PUSH DX
MOV DL,CHAR
MOV AH,02
INT 21H
POP DX
POP AX
ENDM
START: MOV AX,DATA
MOV DS,AX
CALL TIME
MOV AH,4CH
INT21H
TIME PROC NEAR
MOV AH,2CH ;function to read system time
INT21H
MOV HOUR,CH
MOV MIN,CL
CMP CH,12
JB DOWN
SUB CH,12
DOWN: MOV AL,CH
MOV AH,00H
AAM
MOV AX,3030H
DISCHAR AH
DISCHAR AL
DISCHAR':'
MOV AL,CL
MOV AH,00H
AAM
- 56 -
ADD AX,3030H
DISCHAR AH
DISCHAR AL
DISCHAR' '
CMP HOUR,12
JB AM
DISCHAR 'P'
JMP DOWN1
AM: DISCHAR'A'
DOWN1: DISCHAR'M'
RET
TIME ENDP
CODE ENDS
END START
```

ExNo:7 Port Programming

i)Write an 8051 C program to send values of –4 to +4 to port P1.

```
//Signed numbers
#include <reg51.h>
void main(void)
{
char mynum[]={+1,-1,+2,-2,+3,-3,+4,-4};
unsigned char z;
for (z=0;z<=8;z++)
P1=mynum[z];
}
```

ii)Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50,000 times.

Solution:

```
#include <reg51.h>
sbit MYBIT=P1^0;
void main(void)
{
unsigned int z;
for (z=0;z<=50000;z++)
{
MYBIT=0;
MYBIT=1;
}
}
```

iii)Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
{
while (1) //repeat forever
{
p1=0x55;
MSDelay(250);
p1=0xAA;
MSDelay(250);
}
}
void MSDelay(unsigned int itime)
{
unsigned int i,j;
for (i=0;i<itime;i++)
for (j=0;j<1275;j++);
}
```

ExNo:8 Timer-Counter Programming

i) Write an 8051 C program to toggle only pin P1.5 continuously every 250 ms. Use Timer 0, mode 2 (8-bit auto-reload) to create the delay.

Solution:

```
#include <reg51.h>
void T0M2Delay(void);
sbit mybit=P1^5;
void main(void){
unsigned char x,y;
while (1) {
mybit=~mybit;
for (x=0;x<250;x++)
for (y=0;y<36;y++) //we put 36, not 40
T0M2Delay();
}
}
void T0M2Delay(void){
TMOD=0x02;
TH0=-23;
TR0=1;
while (TF0==0);
TR0=0;
TF0=0;
}
```

ii) Write an 8051 C program to create a frequency of 2500 Hz on pin P2.7. Use Timer 1, mode 2 to create delay.

Solution:

```
#include <reg51.h>
void T1M2Delay(void);
sbit mybit=P2^7;
void main(void){
unsigned char x;
while (1) {
mybit=~mybit;
T1M2Delay();
}
}
void T1M2Delay(void){
TMOD=0x20;
TH1=-184;
TR1=1;
while (TF1==0);
TR1=0;
TF1=0;
}
```

ExNo:9 Serial Programming

Write a program for the 8051 to transfer letter “A” serially at 4800 baud, continuously.

Solution:

```
MOV TMOD,#20H ;timer 1,mode 2(auto reload)
MOV TH1,#-6 ;4800 baud rate
MOV SCON,#50H ;8-bit, 1 stop, REN enabled
SETB TR1 ;start timer 1
AGAIN: MOV SBUF,#”A” ;letter “A” to transfer
HERE: JNB TI,HERE ;wait for the last bit
CLR TI ;clear TI for next char
SJMP AGAIN ;keep sending A
```

Write a program for the 8051 to transfer “YES” serially at 9600 baud, 8-bit data, 1 stop bit, do this continuously

Solution:

```
MOV TMOD,#20H ;timer 1,mode 2(auto reload)
MOV TH1,#-3 ;9600 baud rate
MOV SCON,#50H ;8-bit, 1 stop, REN enabled
SETB TR1 ;start timer 1
AGAIN: MOV A,#”Y” ;transfer “Y”
ACALL TRANS
MOV A,#”E” ;transfer “E”
ACALL TRANS
MOV A,#”S” ;transfer “S”
ACALL TRANS
SJMP AGAIN ;keep doing it
;serial data transfer subroutine
TRANS: MOV SBUF,A ;load SBUF
HERE: JNB TI,HERE ;wait for the last bit
CLR TI ;get ready for next byte
RET
```

ExNo:10 Interrupt Programming

Write a program in which the 8051 reads data from P1 and writes it to P2 continuously while giving a copy of it to the serial COM port to be transferred serially. Assume that XTAL=11.0592. Set the baud rate at 9600.

Solution:

```
ORG 0000H

LJMP MAIN

ORG 23H

LJMP SERIAL ;jump to serial int ISR

ORG 30H

MAIN: MOV P1,#0FFH ;             make P1 an input port
MOV TMOD,#20H ;                  timer 1, auto reload
MOV TH1,#0FDH ;                  9600 baud rate
MOV SCON,#50H ;                  8-bit,1 stop, ren enabled
MOV IE,10010000B ;                       enable serial int.
SETB TR1 ;                       start timer 1
BACK: MOV A,P1 ;                 read data from port 1
MOV SBUF,A ;                     give a copy to SBUF
MOV P2,A ;                       send it to P2
SJMP BACK ;                      stay in loop indefinitely

;-----------------SERIAL PORT ISR
ORG 100H
SERIAL: JB TI,TRANS;             jump if TI is high
MOV A,SBUF ;                     otherwise due to receive
CLR RI ;                         clear RI since CPU doesn't
RETI ;                           return from ISR
TRANS: CLR TI ;                  clear TI since CPU doesn't
RETI ;                           return from ISR
END
```