

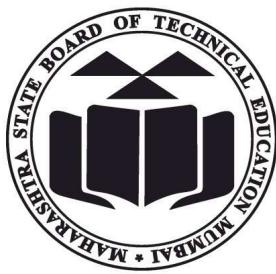
**A Laboratory Manual
for**

Microprocessor

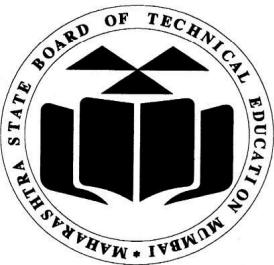
(22415)

Semester – IV

(CO, CM, CW)



**Maharashtra State
Board of Technical Education, Mumbai
(Autonomous) (ISO 9001:2015) (ISO/IEC 27001:2013)**



Maharashtra State Board of Technical Education

Certificate

This is to certify that Mr. / Ms.

Roll No.....of Fourth Semester of Diploma in

..... of Institute

.....

(Code.....) has attained predefined practical outcomes (PROs) satisfactorily in course **Microprocessor (22415)** for the academic year 20.....to 20..... as prescribed in the curriculum.

Place

Enrollment No.....

Date:

Exam Seat No.

Course Teacher

Head of the Department

Principal



Sr.no	Practical Outcome	Date of performance	Date of submission	Marks	Remark
1	Identify various pins of the given microprocessor.	6/1/24	13/1/24	25 25	Ques. ✓
2	Use Assembly Language Programming Tools and functions	13/1/24	20/1/24	24 25	Ques. ✓
3	Use different addressing mode instruction in program Write an Assembly Language Program (ALP) to add two given 8 and 16 bit numbers. (b) Write an Assembly Language Program (ALP) to subtract two given 8 and 16 bit numbers.	20/1/24	27/1/24	23 25	Ques. ✓
4	(a) Write an ALP to multiply two given 8 and 16 bit unsigned numbers. (b) Write an ALP to multiply two given 8 and 16 bit signed numbers.	27/1/24	3/2/24	23 25	Ques. ✓
5	(a) Write an ALP to divide two unsigned numbers 02 (b) Write an ALP to divide two signed numbers	3/2/24	10/2/24	23 25	Ques. ✓
6	Write an ALP to add, subtract, multiply, divide two BCD numbers.	10/2/24	17/2/24	23 25	Ques. ✓
7	(a) Write an ALP to perform block transfer data using string instructions 02 (b) Write an ALP to perform block transfer data without using string instructions.	17/2/24	24/2/24	23 25	Ques. ✓
8	Implement loop in assembly language program (a) Write an ALP to find sum of series of Hexadecimal Numbers. (b) Write an ALP to find sum of series of BCD numbers.	24/2/24	2/3/24	23 25	Ques. ✓

9	(a) Write an ALP to find smallest number from array of n numbers. (a) Write an ALP to find largest number from array of n numbers.	2 3 24	9 3 24	$\frac{23}{25}$	Ques. No. 9
10	(a) Write an ALP to arrange numbers in array in ascending order. (b) Write an ALP to arrange numbers in array in descending order.	9 3 24	16 3 24	$\frac{22}{25}$	Ques. No. 10
11	Write an ALP for $Z = (A + B) * (C + D)$ using Procedure	16 3 24	16 3 24	$\frac{22}{25}$	Ques. No. 11
12	Write an ALP for $Z = (A + B) * (C + D)$ using MACRO.	16 3 24	16 3 24	$\frac{22}{25}$	Ques. No. 12



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No:L004B	Name of Subject Teacher: Pragati Mali
Name of Student: Siddharth Shah	Roll Id: 22203A0041

Experiment No:	1
Title of Experiment	Identify the various pins of the 8086 Microprocessor

AIM: Identify the various pins of the 8086 Microprocessor.

DIAGRAM:

		MAX MODE	MIN MODE
GND	1	40	V _{CC}
AD ₁₄	2	39	AD ₁₅
AD ₁₃	3	38	A _{16/S₃}
AD ₁₂	4	37	A _{17/S₄}
AD ₁₁	5	36	A _{18/S₅}
AD ₁₀	6	35	A _{19/S₆}
AD ₉	7	34	BHE'/S ₇
AD ₈	8	33	MN/MX'
AD ₇	9	32	RD'
AD ₆	10	31	RQ'/GT ₀ '
AD ₅	11	30	RQ'/GT ₁ '
AD ₄	12	29	LOCK'
AD ₃	13	28	S ₂ '
AD ₂	14	27	S ₁ '
AD ₁	15	26	S ₀ '
AD ₀	16	25	QS ₀
NMI	17	24	QS ₁
INTR	18	23	TEST'
CLK	19	22	READY
GND	20	21	RESET

PIN DESCRIPTION:

Pin number	Pin name	Function
16-2, 39	AD0 – AD15	Address/Data bus. These are low order address bus. They are multiplexed with data.
38-35	A16-A19	High order address bus. These are multiplexed with status signals.
28	M/IO'	M/IO' signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicates the memory operation
27	DT/R'	DT/R' stands for Data Transmit/Receive signal. It decides the direction of data flow through the transreceiver. When it is high, data is transmitted out and vice-versa.
26	DEN'	DEN' stands for Data Enable and is available at pin 26. It is used to enable Transceiver 8286. The transceiver is a device used to separate data from the address/data bus.

25	ALE	ALE stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.
29	WR' OR LOCK	WR' stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.
18	INTR	It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not
21	RESET	It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.
22	READY	It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.
23	TEST'	When this signal is high, then the processor has to wait for IDLE state, else the execution continues

24	INTA'	When the microprocessor receives this signal, it acknowledges the interrupt.
30	HLDA	This signal acknowledges the HOLD signal.
31	HOLD	HOLD signal indicates to the processor that external devices are requesting to access the address/data buses.
32	RD'	RD' is available at pin 32 and is used to read signal for Read operation
33	MN/MX'	MN/MX' stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-versa.
34	BHE'	BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus AD8-AD15. This signal is low during the first clock cycle, thereafter it is active.
17	NMI	NMI stands for non-maskable interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.
1,20	GND	It provides ground for the microprocessor
40	VCC	It has 5V DC supply

CONCLUSION:

8086 was the first 16-bit microprocessor available in 40-pin DIP chip .By this practical we can understand the pin Number with their respective functions.

Subject: Microprocessor	Subject Code: 22415
Semester: 4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: Pragati Mali
Name of Student: Siddharth.P.Shah	Roll Id: 22203A0041

Experiment No:	2
Title of Experiment	Use Assembly Language Programming Tools and Functions

Aim: To Use Assembly Language Programming Tools and Functions

Table:

Tools	Function	Software Used
Assembler	An assembler is a program that converts source code program written in assembly language into object files in machine language.	TASM
Linker	A linker is a program that combines object file created by the assembler with other object files and link libraries and produces a single executable program.	TLINK for TASM and LINK.EXE

Debugger	A debugger is a program that allows you to trace the execution of a program and examine the content of registers and memory.	Turbo Debugger for TASM
Editor	An editor is used to create assembly language source files.	Notepad

Conclusion:

Learned about essential programming tools, including text editors for code creation assemblers for converting code to machine code , linker for combining multiple code modules, and debuggers for identifying and resolving programming errors



DEPARTMENT OF COMPUTER ENGINEERING

Subject:- Microprocessors	Subject Code: 22415
Semester: 4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: Pragati Mali
Name of Student: Aditya Makwana	Roll Id: 22203A0042

Experiment No:	3
Title of Experiment	Write an Assembly Language program to perform Addition and subtraction of two 8-bit and 16-bit numbers

- Software Used:-

TASM 1.4

Text Editor (Notepad) Addition of 8-bit numbers:-

- Program Code

DATA SEGMENT

NUM1 DB 55H

NUM2 DB 11H

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

START:

MOV AX, DATA

MOV DS, AX

MOV AL, NUM1 ADD AL, NUM2

INT 3

CODE ENDS

END START

- Output

The screenshot shows a debugger interface with the following details:

- Registers:** ax=4866, bx=0000, cx=0000, dx=0000, si=0000, di=0000, bp=0000, sp=0000, ds=48AD, es=489D, ss=48AC, cs=48AE, ip=000C.
- Stack:** ss:0002 6474, ss:0000>0000, ss:FFFE 3206, ss:FFFC 48AE, ss:FFFA 000D.
- Memory Dump:** A dump of memory starting at address 0000 CD 20 FF 9F 00 EA FF FF = f 8, followed by several bytes of hex data.
- Assembly View:** Shows assembly code from address 0000 to 001F. The instruction at 000C is highlighted as INT 3.
- Status Bar:** F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Addition of 16-bit numbers:-

- Program Code

```
DATA SEGMENT
```

```
    NUM1 DB 5555H
```

```
    NUM2 DB 1001H
```

```
DATA ENDS
```

```
CODE SEGMENT
```

```
    ASSUME CS: CODE, DS: DATA
```

```
START:
```

```
    MOV AX, DATA
```

```
    MOV DS, AX
```

```
    MOV AX, NUM1
```

```
    ADD AX, NUM2
```

```
    INT 3
```

```
CODE ENDS
```

```
END START
```

- Output

The screenshot shows a debugger window with the following details:

Registers:

ax	6556	c=0
bx	0000	z=0
cx	0000	s=0
dx	0000	o=0
si	0000	p=1
di	0000	a=0
bp	0000	i=1
sp	0000	d=0
ds	48AD	
es	489D	
ss	48AC	
cs	48AE	
ip	000C	

Stack Dump:

```

ss:0000 CD 20 FF 9F 00 EA FF FF = f 
ss:0008 AD DE E0 01 C5 15 AA 01 i [x] S 
ss:0010 C5 15 89 02 20 10 92 01 +Se [f] 
ss:0018 01 03 01 00 02 FF FF FF D [ ] 
ss:0020 FF FF FF FF FF FF FF FF FF 

```

Registers (continued):

ss	0002 6474
ss	0000 0000
ss	FFFE 3206
ss	FFFC 48AE
ss	FFFA 000D

Bottom Bar:

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Subtraction of 8-bit numbers:-

- Program Code

```
DATA SEGMENT

    NUM1 DB 55H

    NUM2 DB 11H

DATA ENDS

CODE SEGMENT

    ASSUME CS: CODE, DS: DATA

START:

    MOV AX, DATA

    MOV DS, AX

    MOV AL, NUM1

    SUB AL, NUM2

    INT 3

CODE ENDS END
START
```

- Output

The screenshot shows a debugger interface with the following details:

- Assembly View:** Displays the assembly code for the subtraction program.
- Registers View:** Shows the state of various CPU registers:
 - ax: 0192 (c=1)
 - bx: 2110 (z=0)
 - dx: 3282 (o=0)
 - si: F6D6 (p=0)
 - di: 03AC (a=0)
 - bp: 0100 (i=1)
 - sp: 0106 (d=1)
 - ds: 2110
 - es: 489D
 - ss: 0192
 - cs: 0000
 - ip: 0000
- Stack View:** Shows memory dump and stack contents.
- Status Bar:** Contains keyboard shortcuts for various debugger functions.

Subtraction of 16-bit numbers:-

- Program Code

```
DATA SEGMENT
```

```
    NUM1 DW 5555H
```

```
    NUM2 DW 1001H
```

```
DATA ENDS      CODE SEGMENT
```

```
ASSUME CS: CODE, DS: DATA
```

```
START:
```

```
    MOV AX, DATA
```

```
    MOV DS, AX
```

```
    MOV AX, NUM1
```

```
    SUB AX, NUM2
```

```
    INT 3
```

```
CODE ENDS      END START
```

- Output

READY

```

File Edit View Run Breakpoints Data Options Window Help
-[■]-CPU 80486-
cs:0000 B8AD48    mov    ax,48AD
cs:0003 8ED8    mov    ds,ax
cs:0005 A10000    mov    ax,[0000]
cs:0008 2B060200  sub    ax,[0002]
cs:000C>CC int   03
cs:000D 0000 add   [bx+si],al
cs:000F 0000 add   [bx+si],al
cs:0011 0000 add   [bx+si],al
cs:0013 0000 add   [bx+si],al
cs:0015 0000 add   [bx+si],al
cs:0017 0000 add   [bx+si],al
cs:0019 0000 add   [bx+si],al
cs:001B 0000 add   [bx+si],al
cs:001D 0000 add   [bx+si],al
cs:001F 0000 add   [bx+si],al
ax 4554 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ds 48AD
es 489D
ss 48AC
cs 48AE
ip 000C

ss:0002 6474
ss:0000>0000
ss:FFFFE 3202
ss:FFFC 48AE
ss:FFFA 000D

es:0000 CD 20 FF 9F 00 EA FF FF = f n
es:0008 AD DE E0 01 C5 15 AA 01 i [x0+S>0
es:0010 C5 15 89 02 20 10 92 01 +Se@ P#0
es:0018 01 03 01 00 02 FF FF FF @v@ 0
es:0020 FF FF

```

1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

- **CONCLUSION :-**

In this practical we learned how to perform addition and subtraction operations on 8-bit and 16-bit numbers in assembly language programming.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: MIC	Subject Code:22415
Semester: 4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: Pragati mali
Name of Student: Siddharth.P.Shah	Roll Id: 22203A0041

Experiment No:	4
Title of Experiment	Write an assembly language program to perform Multiplication and signed Multiplication of two 8- and 16-bit numbers

Aim: Write an assembly language program to perform Multiplication and signed Multiplication of two 8- and 16-bit numbers

Software Used : TASM & EMU8086

Write an assembly language program to calculate 8-bit multiplication.

DATA SEGMENT

 NUM1 DB 10H

 NUM2 DB 05H

DATA ENDS

CODE SEGMENT ASSUME CS: CODE , DS:DATA

START:

 MOV AX,DATA

 MOV DS,AX

 MOV AL,NUM1

 MUL AL,NUM2

 INT 3

CODE ENDS

END START

Output:

The screenshot shows a debugger interface with the CPU window open. The assembly code is as follows:

```
CPU 80486
cs:0000 BBAD48    mov    ax,48AD
cs:0003 8ED8      mov    ds,ax
cs:0006 A00000    mov    al,0000
cs:0008 BA1E0100  mov    bl,0001
cs:000C F6E3      mul    bl
cs:000E CC         int    03
```

The registers show:

ax 00A0	c=0
bx 0002	z=0
cx 0000	s=0
dx 0000	v=0
si 0000	p=0
di 0000	a=0
bp 0000	i=1
sp 0000	d=0

The stack dump shows memory starting at address 0000:

ds:0000 50 82 00 00 00 00 00 00	ps
ds:0008 00 00 00 00 00 00 00 00	
ds:0010 BB AD 48 8E D8 A9 00 00	11HAF6
ds:0018 BA 1E 01 00 P6 E3 CC 00 00 00 00	00 00 00 00

At the bottom, function keys are listed:

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Write an assembly language program to calculate 16-bit multiplication.

Code :

DATA SEGMENT

NUM1 DW 2000H

NUM2 DW 0001H

DATA ENDS

CODE SEGMENT ASSUME CS:CODE,DS:DATA START:

MOV AX,DATA

MOV DS,AX

MOV AX,NUM1

MOV BX,NUM2

MUL BX

INT 3

CODE ENDS

END START

Output:



The screenshot shows a debugger interface with two windows. The top window is titled '[1]-CPU 80486' and displays assembly code. The assembly code includes instructions like MOV, ADD, and INT 03. Registers shown in the status bar include AX, BX, CX, DX, SI, DI, BP, SP, DS, ES, SS, CS, and IP. The bottom window is titled '[2]-Dump' and shows memory dump data for segments DS, ES, and SS.

[1]-CPU 80486

```
cs:0000 BBAD48    mov    ax,48AD
cs:0003 0ED0    mov    ds,ax
cs:0006 A10000    mov    ax,10000
cs:0009 0B1E0200    mov    bx,10002
cs:000C F7E3    mul    bx
cs:000E CC int    03
cs:000F 0000    add    [bx+si],al
cs:0011 0000    add    [bx+si],al
cs:0013 0000    add    [bx+si],al
cs:0015 0000    add    [bx+si],al
cs:0017 0000    add    [bx+si],al
cs:0019 0000    add    [bx+si],al
cs:001B 0000    add    [bx+si],al
es:0000 CD 20 FF 9F 00 EA FF FF - j r
es:0003 AD DE E0 01 C5 15 AA 01 i se0 >A9
es:0010 C5 15 B9 02 20 10 92 01 i se0 >A9
es:0018 01 03 01 00 02 FF FF FF 3e9 0
ss:0002 6474
ss:0000>0000
```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

[2]-Dump

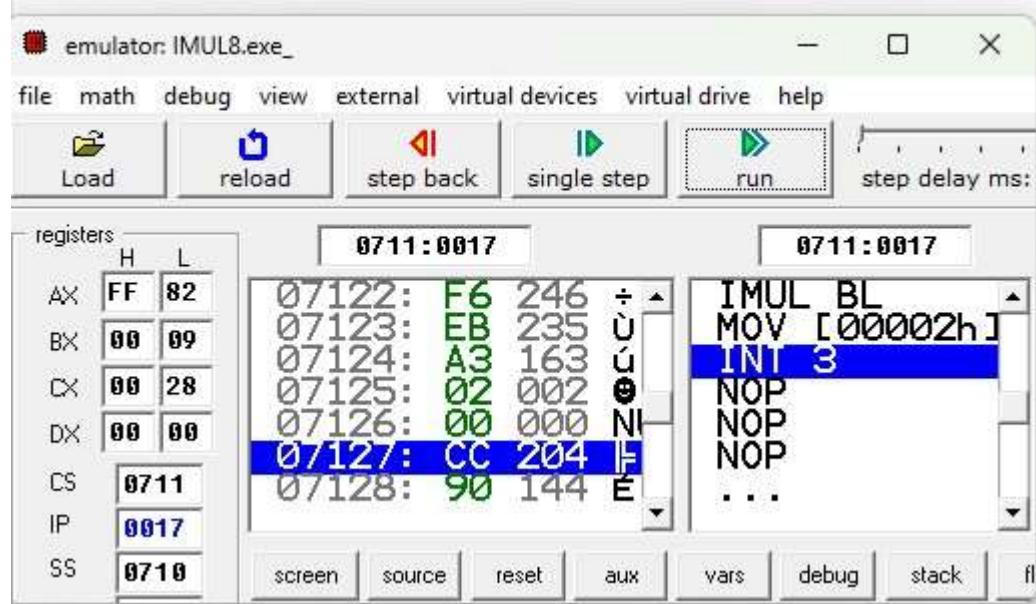
```
ds:0000 00 20 01 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0008 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0010 B8 AD 48 8E D8 A1 00 00 1 i HA#1
ds:0018 8B 1E 02 00 F7 E3 CC 00 i A# ≈ n ||
```

Write an Assembly Language Program (ALP) to perform Signed Multiplication for two given 8-bit numbers.

```
DATA SEGMENT
A DB 0F2H
B DB 09H C
DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:
MOV AX,DATA
MOV DS,AX
MOV AX,0000H
MOV BX,0000H
MOV AL,A
MOV BL,B
IMUL BL
MOV C,AX
INT 03H
CODE ENDS
END START
```

Output:

```
DATA SEGMENT
A DB 0F2H
B DB 09H
C DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOV AX, DATA
MOV DS, AX
MOV AX, 0000H
MOV BX, 0000H
MOV AL, A
MOV BL, B
IMUL BL
MOV C, AX
INT 03H
CODE ENDS
END START
```



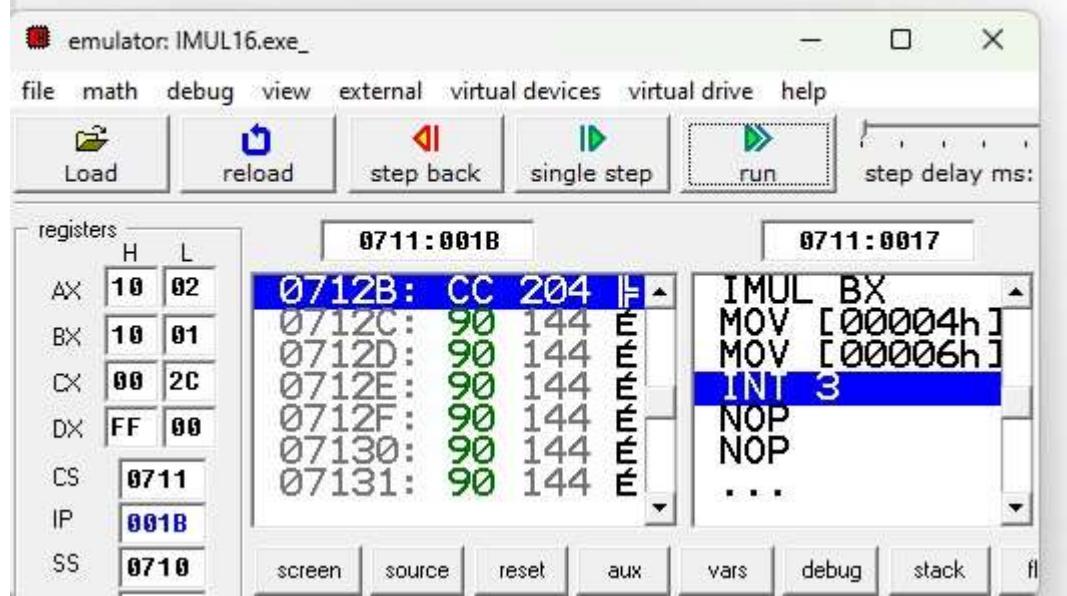
Write an Assembly Language Program (ALP) to perform Signed Multiplication for two given 16-bit numbers.

DATA SEGMENT

```
A DW 0F002H
B DW 1001H C
DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE,DS:DATA
START:
MOV AX,DATA
MOV DS,AX
MOV AX,0000H
MOV BX,0000H
MOV AX,A
MOV BX,B
IMUL BX
MOV WORD PTR C,AX
MOV WORD PTR C+2,DX
INT 03H
CODE ENDS
END START
```

Output:

```
DATA SEGMENT
A DW 0F002H
B DW 1001H
C DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOV AX, DATA
MOV DS, AX
MOV AX, 0000H
MOV BX, 0000H
MOV AX, A
MOV BX, B
IMUL BX
MOV WORD PTR C, AX
MOV WORD PTR C+2, DX
INT 03H
CODE ENDS
END START
```



Conclusion:

From This practical we get to know how to perform 8-16 bit Multiplication and Signed Multiplication.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: Pragati Mali
Name of Student: Siddharth Shah	Roll Id: 22203A0041

Experiment No:	5
Title of Experiment	Write an assembly language program to perform Divison and signed Division of two 8- and 16-bit numbers

AIM: -Write an assembly language program to perform Divison and signed Division of two 8- and 16-bit numbers

Used Software:

Emu8086

Write an Assembly Language Program (ALP) to Division two given 8-bit numbers.

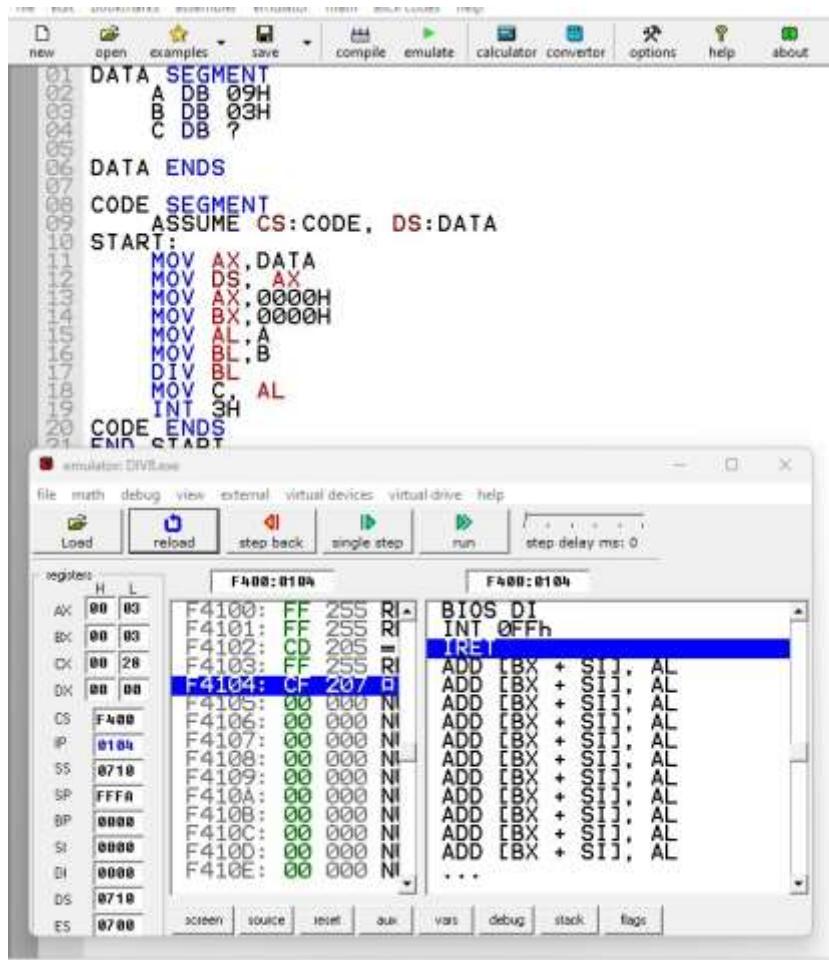
DATA SEGMENT

A DB 09H B
DB 03H C
DB ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA START:
MOV AX,DATA
MOV DS, AX
MOV AX,0000H
MOV BX,0000H
MOV AL,A
MOV BL,B
DIV BL
MOV C, AL
INT 3H
CODE
ENDS END
START



Write an Assembly Language Program (ALP) to Division two given16-bit numbers.

DATA SEGMENT

A DW 4444H

B DW 2002H

C DW ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA START:

MOV AX,DATA

MOV DS, AX

MOV AX,0000H

MOV BX,0000H

MOV AX,A

MOV BX,B

DIV BX

MOV C, AX

INT 3H

CODE ENDS

END START

```

01 DATA SEGMENT
02     A DW 4444H
03     B DW 2002H
04     C DW ?
05 DATA ENDS
06 CODE SEGMENT
07     ASSUME CS:CODE, DS:DATA
08 START:
09         MOV AX, DATA
10         MOV DS, AX
11         MOV AX, 0000H
12         MOV BX, 0000H
13         MOV AX, A
14         MOV BX, B
15         DIV BX
16         MOV C, AX
17         INT 3H
18 CODE ENDS
19 END START

```

Write an Assembly Language Program (ALP) to Signed Division two given 8-bit numbers.

```

DATA SEGMENT
    A DB 0F2H B
    DB 09H C DB ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, 0000H
    MOV BX, 0000H
    MOV AL, A
    MOV BL, B
    IDIV BL
    MOV C, AL
    INT 3H
CODE ENDS
END START

```

```

new open examples save compile emulate calculator converter options help about
01 DATA SEGMENT
02 A DB 0F2H
03 B DB 09H
04 C DB ?
05 DATA ENDS
06 CODE SEGMENT
07 ASSUME CS:CODE, DS:DATA
08 START:
09 MOV AX,DATA
10 MOV DS, AX
11 MOV AX,0000H
12 MOV BX,0000H
13 MOV AL,A
14 MOV BL,B
15 IDIV BL
16 MOV C,AL
17 INT 3H
18 CODE ENDS
19 END START
20 has context menu

```



Write an Assembly Language Program (ALP) to Signed Division two given 16-bit numbers.

```

DATA SEGMENT
A DW 0F444H
B DW 0002H C
DW ?

DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOV AX,DATA MOV
DS, AX
MOV AX,0000H
MOV BX,0000H
MOV AX,A
MOV BX,B
IDIV BX
MOV C, AX
INT 3H
CODE ENDS
END START

```

The screenshot shows a debugger interface with two main windows. The top window displays assembly code for a 16-bit division program. The bottom window shows the CPU register state and memory dump.

Assembly Code:

```

01 DATA SEGMENT
02 A DW 0F444H
03 B DW 0002H
04 C DW ?
05 DATA ENDS
06 CODE SEGMENT
07 ASSUME CS:CODE, DS:DATA
08 START:
09     MOV AX, DATA
10    MOV DS, AX
11    MOV AX, 0000H
12    MOV BX, 0000H
13    MOV AX, A
14    MOV BX, B
15    IDIV BX
16    MOV C, AX
17    INT 3H
18 CODE ENDS
19 END START

```

Registers and Memory Dump:

	H	L
AX	7A 22	
BX	00 02	
CX	00 28	
DX	00 00	
CS	F400	
IP	0104	
SS	0710	
SP	FFFF	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

Memory dump (F400:0104 to F400:010E):

```

F4100: FF 255 RI-
F4101: FF 255 RI-
F4102: CD 205 RI-
F4103: FF 255 RI-
F4104: CF 207 RI-
F4105: 00 000 NI
F4106: 00 000 NI
F4107: 00 000 NI
F4108: 00 000 NI
F4109: 00 000 NI
F410A: 00 000 NI
F410B: 00 000 NI
F410C: 00 000 NI
F410D: 00 000 NI
F410E: 00 000 NI

```

Instruction pointer (IP): 0104

Stack pointer (SP): FFFF

Registers (Registers tab):

- AX: 7A 22
- BX: 00 02
- CX: 00 28
- DX: 00 00
- CS: F400
- IP: 0104
- SS: 0710
- SP: FFFF
- BP: 0000
- SI: 0000
- DI: 0000
- DS: 0710
- ES: 0700

Buttons at the bottom of the registers window: screen, source, reset, aux, view, debug, slack, flags.

Conclusion: From This practical we get to know how to perform 816 bit Division and Signed Divison.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: MIC	Subject Code:22415
Semester: 4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: Pragati Mali
Name of Student: Siddharth Shah	Roll Id: 22203A0041

Experiment No:	6
Title of Experiment	Write an ALP to Add, Sub, Mul, Div two BCD numbers

Aim: Write an assembly language program to perform BCD Addition, Subtraction, Multiplication & Division of 8-bit numbers.

Software Used:

EMU8086

1. Write an assembly language program to Perform 8-bit BCD Addition.

CODE:

```
DATA SEGMENT
A DB 80H
B DB 26H
RES_LSB DB 0H
RES_MSB DB 0H
DATA ENDS
```

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AL, A
    MOV BL, B
    ADD AL, BL
    DAA
    JNC NEXT
    INC RES_MSB
NEXT:
    MOV RES_LSB , AL
    INT 03H CODE
ENDS
END START
```

Output:

The screenshot displays a debugger environment with several windows open:

- Assembly Window:** Shows the assembly code for a program named BCDADD.exe. The code includes segment declarations (DATA, CODE), variable definitions (A, B), and instructions (MOV, ADD, DAA, JNC, INT). Some instructions like MOV AX, DATA and MOV DS, AX are highlighted in red.
- Flags Window:** A dialog titled "lexical flag analyser" showing various CPU flag conditions. It lists flags like CF, ZF, SF, OF, PF, AF, IF, and DF, each with a dropdown menu and a corresponding condition.
- Registers Window:** Shows the current state of CPU registers:

	H	L
AX	07	06
BX	00	26
CX	00	22
DX	00	00
CS	0711	
IP	0011	
SS	0710	
- Stack Window:** Displays the stack contents at address 0711:0011, showing multiple NOP (00 44 EE) instructions.
- Memory Dump Window:** Shows the memory dump from address 07121 to 07127, all containing the value CC 204 (opcode for INT 3).

2. Write an assembly language program to Perfrom 8-bit BCD Subtraction.

CODE:

```
DATA SEGMENT
```

```
    A DB 80H
```

```
    B DB 26H
```

```
    RES_LSB DB 0H
```

```
DATA ENDS
```

```
CODE SEGMENT
```

```
ASSUME CS:CODE, DS:DATA
```

```
START:
```

```
    MOV AX, DATA
```

```
    MOV DS, AX
```

```
    MOV AL, A
```

```
    MOV BL, B
```

```
    SUB AL, BL
```

```
    DAS
```

```
    MOV RES_LSB , AL
```

```
    INT 03H CODE
```

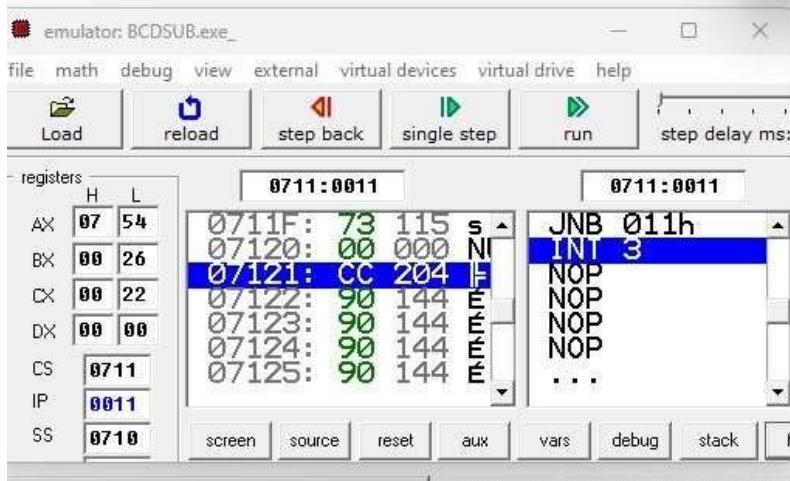
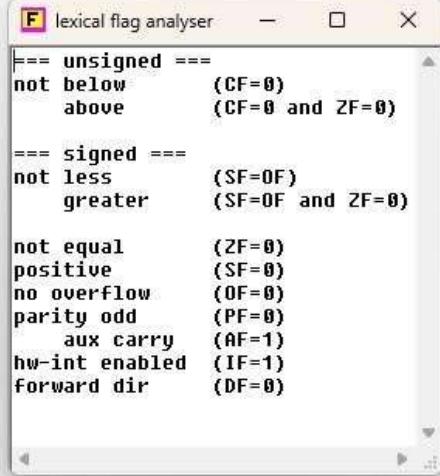
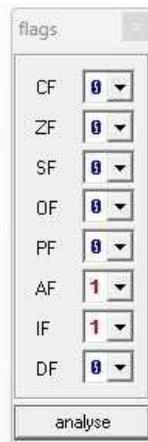
```
ENDS
```

```
END START
```

Output:

```
DATA SEGMENT
    A DB 80H
    B DB 26H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AL, A
    MOV BL, B
    SUB AL, BL
    DAS
    JNC NEXT
NEXT:
    INT 03H
CODE ENDS
END START
```



3. Write an assembly language program to Perform 8-bit BCD Multiplication.

CODE:

DATA SEGMENT

A DB 12H

B DB 03H

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START:

MOV AX, DATA

MOV DS, AX

MOV AX, 0000H

MOV CL, B

UP:

ADD AL, A

DAA

DEC CL

JNZ UP

INT 03H

CODE ENDS

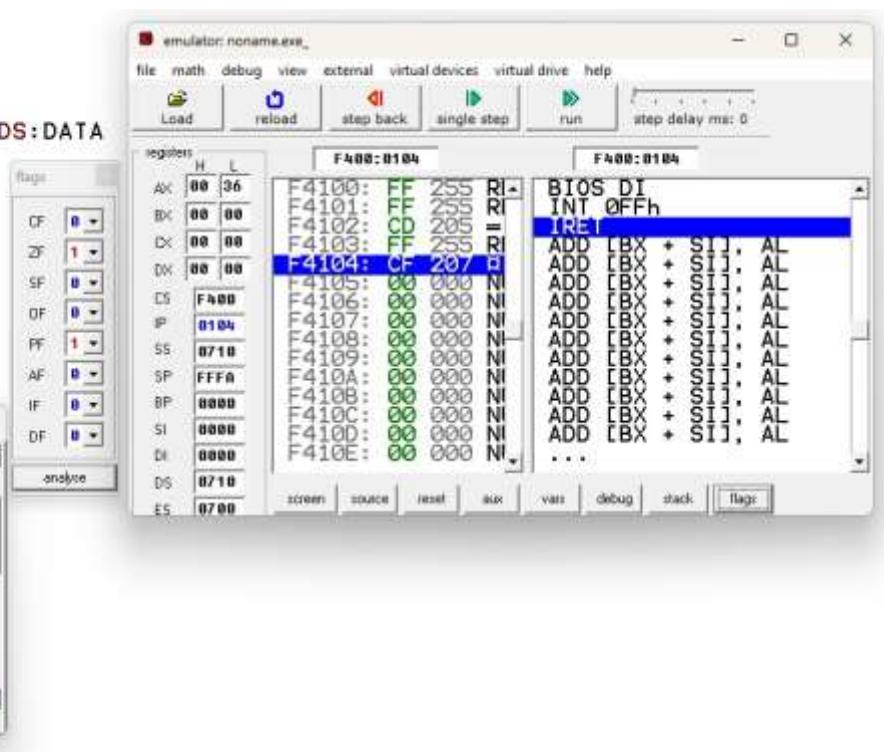
END START

Output:

```
DATA SEGMENT
    A DB 12H
    B DB 03H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, 0000H
    MOV CL, B
    UP:
        ADD AL, A
        DAA
        DEC CL
        JNZ UP
        INT 03H
CODE ENDS
END START
```

```
06  CODE SEGMENT
07  ASSUME CS:CODE,
08  START:
09  MOV AX, DATA
10  MOV DS, AX
11  MOV AX, 0000H
12  MOV CL, B
13  UP:
14  ADD AL, A
15  DAA
16  DEC CL
17  JNZ UP
18  INT 03H
CODE ENDS
```



4. Write an assembly language program to Perform 8-bit BCD Division.

CODE:

```
DATA SEGMENT  
A DB 16H  
B DB 03H    DATA ENDS
```

CODE SEGMENT

```
ASSUME CS:CODE, DS:DATA START:
```

```
MOV AX, DATA  
MOV DS, AX
```

```
MOV AH, 00H  
MOV AL, A
```

```
MOV CL, B  
MOV CH, 00H
```

```
DIV_LOOP:
```

```
CMP AL, CL  
JB DIV_END  
SUB AL, CL  
DAS  
INC AH  
JMP DIV_LOOP
```

DIV_END:

```

INT 03H
CODE ENDS
END START

```

Output:

The screenshot shows a 16-bit assembly language debugger interface. On the left, the assembly code is displayed:

```

DATA SEGMENT
A DB 16H
B DB 03H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOV AX, DATA
MOV DS, AX
MOV AH, 00H
MOV AL, A
MOV CL, B
MOV CH, 00H
DIV_LOOP:
CMP AL, CL
JB DIV_END
SUB AL, CL
DAS
INC AH
JMP DIV_LOOP
DIV_END:
INT 03H
CODE ENDS
END START

```

Below the code is a register window showing the state of various CPU registers:

Registers	H	L
AX	05	01
BX	00	00
CX	00	03
DX	00	00
SI	F400	
IP	0104	
SS	0710	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

The memory dump window shows the memory starting at address F400:

Address	Value	Content
F400:0104	FF 255 RI	BIOS DI
F401:0104	FF 255 RI	INT OFFh
F402:0104	CD 205 =	IRET
F403:0104	FF 255 RI	ADD [BX + SI], AL
F404:0104	CF 207 01	ADD [BX + SI], AL
F405:0104	00 000 NI	ADD [BX + SI], AL
F406:0104	00 000 NI	ADD [BX + SI], AL
F407:0104	00 000 NI	ADD [BX + SI], AL
F408:0104	00 000 NI	ADD [BX + SI], AL
F409:0104	00 000 NI	ADD [BX + SI], AL
F40A:0104	00 000 NI	ADD [BX + SI], AL
F40B:0104	00 000 NI	ADD [BX + SI], AL
F40C:0104	00 000 NI	ADD [BX + SI], AL
F40D:0104	00 000 NI	ADD [BX + SI], AL
F40E:0104	00 000 NI	...

At the bottom, there is a source code editor window titled "original source.asm" showing the assembly code with the INT 03H instruction highlighted.

Conclusion: From this practical we get to know how to perform 8 bit BCD Addition, Subtraction ,Multiplication & Division.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code: 22415
Semester: 04	Course: CO4I-A
Laboratory No: L004B	Name of Subject Teacher: Pragati Mali
Name of Student: Siddharth Shah	Roll ID: 22203A0041

Experiment No:	7
Aim	a. Write an ALP to perform block transfer data using string instruction. b. Write an ALP to perform block transfer data without using string instruction.

1. Block Transfer data without using string instruction.

CODE:

DATA SEGMENT

SRC_BLK DB 11H, 22H, 33H, 44H, 55H

DST_BLK DB 5 DUP(0)

DATA ENDS

CODE SEGMENT ASSUME

CS:CODE, DS:DATA START:

MOV AX, DATA

```
MOV DS, AX  
MOV CX, 05H  
LEA SI, SRC_BLK  
LEA BX, DST_BLK
```

UP:

```
MOV AX, [SI]
```

```
MOV [BX], AX
```

```
INC SI
```

```
INC BX
```

```
DEC CX
```

```
JNZ UP
```

```
INT 03H
```

```
CODE ENDS
```

```
END START
```

Output:

The screenshot shows a debugger interface with the following components:

- Assembly View:** Displays the assembly code with labels and instructions. The instruction at address 0710E is highlighted in yellow.
- Registers View:** Shows the CPU register values. The AX register has a value of 11 55.
- Memory Dump View:** Shows memory starting at address 07100. The first few bytes are 11 017, followed by several ADD instructions.

2. Block Transfer data with string instruction.

```
SRC_BLK DB 11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H,  
99H, 0AAH
```

```
DST_BLK DB 0AH DUP(0)  
DATA ENDS
```

```
CODE SEGMENT
```

```
ASSUME CS:CODE, DS:DATA    START:
```

```
MOV AX, DATA
```

```
MOV DS, AX
```

```
MOV ES, AX
```

```
MOV CX, 0AH
```

```
LEA SI, SRC_BLK
```

```
LEA DI, DST_BLK
```

```
UP:
```

```
MOVSB
```

```
LOOP UP
```

```
INT 03H
```

```
CODE ENDS END START
```

Output:

The screenshot displays a debugger environment with several windows:

- Assembly Window:** Shows the assembly code:

```
DATA SEGMENT
SRC_BLK DB 11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H, 99H, 0AAH
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOV AX, DATA
MOV DS, AX
MOV ES, AX
MOV CX, 0AH
LEA SI, SRC_BLK
LEA DI, DST_BLK

UP:
MOVSB
LOOP UP
INT 03H
CODE ENDS
END START
```
- Registers Window:** Shows register values:

register	H	L
AX	07	18
BX	00	00
CX	00	00
DX	00	00
ES	F400	
IP	0100	
SS	0710	
SP	FFFF	
BP	0000	
SI	0000	
DI	0014	
DS	0710	
ES	0710	
- Memory Dump Window:** Shows memory dump from 07100 to F400:

Address	Value	Hex	Char
07100	11	01	1
07101	22	03	2
07102	33	05	3
07103	44	06	4
07104	55	08	5
07105	66	09	6
07106	77	0A	7
07107	88	0B	8
07108	99	0C	9
07109	AA	0D	A
0710A	11	0E	B
0710B	22	0F	C
0710C	33	10	D
0710D	44	11	E
0710E	55	12	F
- Flags Window:** Shows flag status:

Flag	Value
CF	0
ZF	0
SF	0
DF	0
PF	0
AF	0
IF	0
DF	0

Conclusion:

With this practical we learned to perform block data transfer with and without using String Instruction



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code: 22415
Semester: 04	Course: CO4IA
Laboratory No: L004A	Name of Subject Teacher: Prof. Pragati Mali
Name of Student: Siddharth Shah	Roll ID: 22203A0041

Experiment No:	8
Aim	Implement Loop in Assembly language Program. (A)Write an ALP to find sum of series of Hexadecimal Numbers. (B)Write an ALP to find sum of series of BCD Numbers.

1. Write an ALP to find sum of series of Hexadecimal Numbers.

DATA SEGMENT

 SERIES DB 11H, 02H, 03H, 01H,
 00H

 SUM DB 00H

DATA ENDS

CODE SEGMENT

 ASSUME CS:CODE, DS:DATA

START:

 MOV AX, DATA

 MOV DS, AX

```
MOV AX, 0000H  
MOV CX, 04H  
LEA BX, SERIES
```

```
REPT:  
ADD AL, [BX]  
INC BX  
DEC CX  
JNZ REPT  
MOV SUM, AL  
MOV DL, SUM  
INT 03H  
CODE ENDS END  
START
```

Output:

```
DATA SEGMENT
    SERIES DB 11H, 02H, 03H, 01H, 00H
    SUM DB 00H
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, 0000H
    MOV CX, 04H
    LEA BX, SERIES

    REPT:
        ADD AL, [BX]
        INC BX
        DEC CX
    JNZ REPT
    MOV SUM, AL
    MOV DL, SUM
    INT 03H
CODE ENDS
END START
```

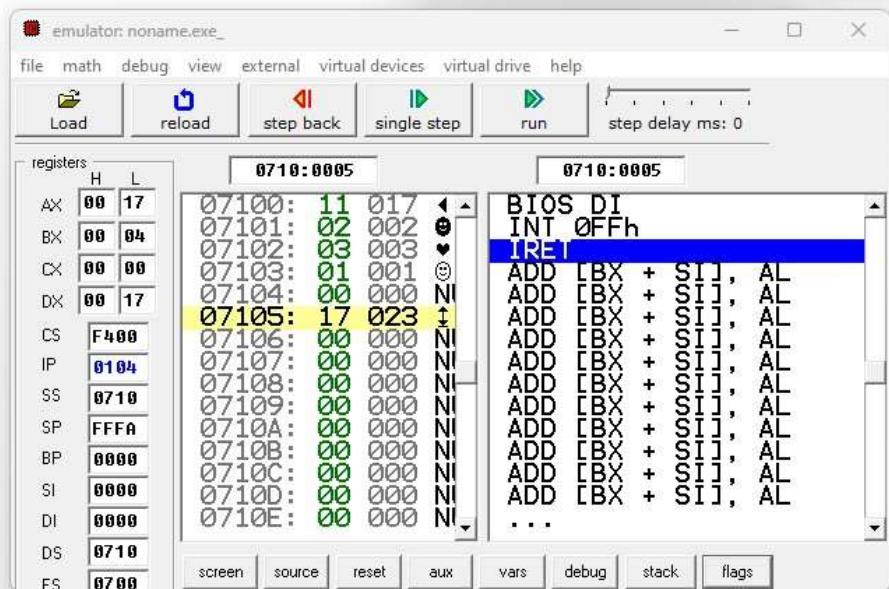
original source co... X

10	MOV DS, AX
11	MOV AX, 0000H
12	MOV CX, 04H
13	LEA BX, SERIES
14	
15	REPT:
16	ADD AL, [BX]
17	INC BX
18	DEC CX
19	JNZ REPT
20	MOV SUM, AL
21	MOV DL, SUM
22	INT 03H
23	CODE ENDS

flags

CF	0
ZF	1
SF	0
OF	0
PF	1
AF	0
IF	0
DF	0

analyse



2. Write an ALP to find sum of series of BCD Numbers

DATA SEGMENT

SERIES DB 11H, 22H, 22H, 11H, 55H

SUM DB 00H

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX, DATA

MOV DS, AX

MOV AX, 0000H

MOV CX, 05H LEA BX, SERIES REPT:

ADD AL, [BX]

DAA

MOV SUM, AL

INC BX

DEC CX

JNZ REPT

MOV DL, SUM

INT 03H

CODE ENDS

END START

Output:

The screenshot shows a debugger interface with several windows:

- Assembly Window:** Displays the assembly code for the program. It includes segments for DATA and CODE, and labels for START, REPT, and IRET.
- Registers Window:** Shows the current values of CPU registers. AX is 00 21, BX is 00 05, CX is 00 00, DX is 00 21, CS is F400, IP is 0104, SS is 0710, SP is FFFA, BP is 0000, SI is 0000, DI is 0000, DS is 0710, and EC is 0700.
- Stack Window:** Shows the stack memory starting at address F4100. The stack contains several FF (opcode) bytes followed by the INT 0FFh instruction (opcode C0 207 E).
- Variables Window:** Shows the variable definitions for SERIES and SUM. SERIES is defined as DB 11H, 22H, 22H, 11H, 55H, and SUM is defined as DB 00H.
- Flags Window:** Shows the current state of various CPU flags: CF (1), ZF (1), SF (0), OF (0), PF (1), AF (0), IF (0), and DF (0).
- Emulator Window:** Shows the emulator window with the title "emulator: noname.exe". It has tabs for file, math, debug, view, external, virtual devices, virtual drive, and help. Buttons for Load, Reload, Step Back, Single Step, Run, and Step Delay ms: 0 are visible.

Conclusion:

With this practical we learned to perform sum of series of hexadecimal and BCD numbers.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code: 22415
Semester: 4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: Prof. Pragati Mali
Name of Student: Siddharth Shah	Roll Id: 22203A0041

Experiment No:	9
Title of Experiment:	(A) Write an ALP to find the smallest number from Array of n numbers. (B)Write an ALP to find largest number from Array of n numbers

1. Write an ALP to find the smallest number from Array of n numbers.

DATA SEGMENT

ARRAY DB 12H, 07H, 25H, 4BH, 02H

SMALL DB 00H

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

START:

MOV AX, DATA

MOV DS, AX

MOV AX, 0000H

MOV CX, 05H

LEA SI, ARRAY

MOV AL, [SI]

DEC CX

UP:

INC SI

CMP AL, [SI]

JLE NEXT

MOV AL, [SI]

NEXT:

LOOP UP

MOV SMALL, AL

INT 03H

CODE ENDS

END START

OUTPUT:

The screenshot displays a debugger interface with several windows:

- Assembly Window:** Shows the assembly code with line numbers from 14 to 27. Lines 14-16 and 26-27 are highlighted in yellow.
- Registers Window:** Shows CPU register values at address 0710:0005. Registers include AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, and ES. The IP register shows the instruction address 0710:0005.
- Stack Window:** Shows the stack contents starting at address 0710:0000, with the value 00000000 at the top.
- Variables Window:** Shows memory variables at address 0710:0000. It lists two variables: "ARRAY" with value 12h, 07h, 25h, 4Bh, 02h and "SMALL" with value 02h.
- Emulator Window:** Shows the emulator window title "emulator: noname.exe.". It has buttons for Load, reload, step back, single step, run, and step delay ms: 0. The status bar shows addresses 0710:0005 and 0710:0005.

```
DATA SEGMENT
    ARRAY DB 12H, 07H, 25H, 4BH, 02H
    SMALL DB 00H
DATA ENDS

CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, 0000H
    MOV CX, 05H
    LEA SI, ARRAY
    MOV AL, [SI]
    DEC CX

    UP:
        INC SI
        CMP AL, [SI]
        JLE NEXT
        MOV AL, [SI]
    NEXT:
        LOOP UP
        MOV SMALL, AL
        INT 03H
CODE ENDS
END START
```

14 LEA SI, ARRAY
15 MOV AL, [SI]
16 DEC CX
17
18 UP:
19 INC SI
20 CMP AL, [SI]
21 JLE NEXT
22 MOV AL, [SI]
23 NEXT:
24 LOOP UP
25 MOV SMALL, AL
26 INT 03H
27 CODE ENDS

2. Write an ALP to find the largest number from Array of n numbers.

DATA SEGMENT

ARRAY DB 12H, 07H, 25H, 70H, 02H

LARGE DB 00H

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

START:

MOV AX, DATA

MOV DS, AX

MOV AX, 0000H

MOV CX, 05H

LEA SI, ARRAY

MOV AL, [SI]

DEC CX

UP:

INC SI

CMP AL, [SI]

JNL NEXT

MOV AL, [SI]

NEXT:

LOOP UP

MOV LARGE, AL

INT 03H

CODE ENDS

END START

OUTPUT

The screenshot shows a debugger interface with several windows:

- Assembly Window:** Displays the assembly code for the program. Lines 15 through 28 are highlighted in yellow. The code includes segments for DATA and CODE, loops, and various instructions like MOV, CMP, JNL, and INT.
- Registers Window:** Shows the state of CPU registers. The IP register is at address F400:0104, which corresponds to the instruction INT 03H.
- Stack Window:** Shows the stack contents starting at address F4100. The stack grows downwards, with the current top of the stack at F4104 containing the value 00 00.
- Memory Dump Window:** Shows the memory dump for the variable ARRAY. It contains two bytes: 12h at address 0000H and 70h at address 0001H.

CONCLUSION:-

Different addressing mode of instructions are implemented in above program:

- To find the smallest number from Array of n numbers.
- To find the largest number from Array of n numbers.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code: 22415
Semester: 04	Course: CO4I-A
Laboratory No: L004B	Name of Subject Teacher: Prof. Pragati Mali
Name of Student: Siddharth Shah	Roll ID: 22203A0041

Experiment No:	10
Aim	(A) Write an ALP to arrange numbers in array in ascending order (B) Write an ALP to arrange numbers in array in descending order

1. Write an ALP to arrange numbers in array in ascending order.

Code:

DATA SEGMENT

 ARRAY DB 12H, 07H, 15H, 23H, 02H

DATA ENDS

CODE SEGMENT

 ASSUME CS: CODE, DS: DATA START:

 MOV AX, DATA

 MOV DS, AX

 MOV BX, 05H

TOP:

 LEA SI, ARRAY

 MOV CX, 04H

UP:

 MOV AL, [SI]

 CMP AL, [SI+1]

 JLE DN

 MOV DL, [SI+1]

 MOV [SI+1], AL

 MOV [SI], DL

DN:

 ADD SI, 01H

```
LOOP UP  
DEC BX  
JNZ TOP
```

```
LEA SI, ARRAY  
MOV CX, 5
```

```
DISPLAY_LOOP:  
MOV DL, [SI]  
INC SI  
LOOP DISPLAY_LOOP
```

```
INT 03H
```

```
CODE ENDS  
END START
```

Output:

The screenshot displays a debugger environment with several windows:

- Assembly Window:** Shows the assembly code for a program. It includes segments for DATA and CODE, and labels for START, TOP, UP, DN, and DISPLAY_LOOP. The code uses registers AX, DS, BX, SI, CX, AL, DL, and flags like CF and OF.
- Registers Window:** Shows the state of CPU registers. The AX register is set to 0715. Other registers include BX (0000), CX (0000), DX (0023), CS (F400), IP (0104), SS (0710), SP (FFFF), BP (0000), SI (0005), DI (0000), DS (0710), and ES (0700). The IP register is highlighted in yellow, pointing to the instruction INT 03H.
- Stack Window:** Shows the stack contents starting at address F400:0104. The stack grows downwards, with the top of the stack at F400:0104 containing the value FF 255. Subsequent stack frames show values such as FF 255, CD 205, FF 255, and so on.
- Memory Dump Window:** Shows the memory dump for the variable ARRAY. The array is defined at address 02h, containing the bytes 07h, 12h, 15h, 23h, and 02h.

```
DATA SEGMENT
    ARRAY DB 12H, 07H, 15H, 23H, 02H
DATA ENDS

CODE SEGMENT
    ASSUME CS: CODE, DS: DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV BX, 05H
    TOP:
        LEA SI, ARRAY
        MOV CX, 04H
    UP:
        MOV AL, [SI]
        CMP AL, [SI+1]
        JLE DN
        DISPLAY_LOOP:
            MOV DL, [SI]
            INC SI
            LOOP DISPLAY_LOOP
            MOV DL, [SI+1]
            MOV [SI+1], AL
            MOV [SI], DL
    DN:
        ADD SI, 01H
        LOOP UP
        DEC BX
        JNZ TOP

        LEA SI, ARRAY
        MOV CX, 5
    DISPLAY_LOOP:
        MOV DL, [SI]
        INC SI
        LOOP DISPLAY_LOOP
        INT 03H

CODE ENDS
END START
```

2. Write an ALP arrange numbers in array in descending order.

DATA SEGMENT

ARRAY DB 12H, 07H, 15H, 23H, 02H

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE, DS: DATA START:

MOV AX, DATA

MOV DS, AX

MOV BX, 05H

TOP:

LEA SI, ARRAY

MOV CX, 04H

UP:

MOV AL, [SI]

CMP AL, [SI+1]

JGE DN

MOV DL, [SI+1]

MOV [SI+1], AL

MOV [SI], DL

DN:

ADD SI, 01H

LOOP UP

DEC BX

JNZ TOP

LEA SI, ARRAY

MOV CX, 5

DISPLAY_LOOP:

MOV DL, [SI]

INC SI

LOOP DISPLAY_LOOP

INT 03H

```
CODE ENDS  
END START
```

Output:

The screenshot shows a debugger interface with several windows:

- Assembly Window:** Displays the assembly code for the program. It includes sections for DATA and CODE segments, with labels like START, TOP, UP, DN, and DISPLAY_LOOP. The code uses registers AX, DS, BX, SI, CX, and DL.
- Registers Window:** Shows the state of various CPU registers at address F400:0104. Registers include AX, BX, CX, DX, SI, IP, SS, SP, BP, DI, DS, and ES. Values shown include FF, 255, CD, 00, 01, 02, F4B0, 0104, 0710, and 0700.
- Stack Window:** Shows the stack contents starting at address F400:0104. The stack grows downwards, with values like BIOS, DI, INT, 0FFh, and IREI.
- Memory Dump Window:** Shows the memory dump for the array, with values 23h, 15h, 12h, 07h, and 02h.

Conclusion:

With this practical we learned how to arrange arrays in Ascending and Descending Order.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: MIC	Subject Code:224145
Semester:4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: Pragati Mali
Name of Student: Siddharth Shah	Roll Id: 22203A0041

Experiment No:	11
Title of Experiment	Write an ALP using Procedure to solve equation such as $Z=(A+B)*(C+D)$.

**Q. Write an ALP using Procedure to solve equation such as
 $Z=(A+B)*(C+D)$.**

DATA SEGMENT

P DB 04H

Q DB 02H

R DB 01H

S DB 02H

Z DW 00H

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

START:

MOV AX, DATA

MOV DS, AX

MOV AL, P

MOV BL, Q

CALL ADD_BYTE

MOV CL, AL

MOV AL, R

MOV BL, S

CALL ADD_BYTE

MUL CL

MOV Z, AX

```

INT 3H
ADD_BYTE PROC
ADD AL, BL
RET
ENDP

```

```

CODE ENDS
END START

```

OUTPUT:

The screenshot shows a debugger interface with three main windows:

- Assembly Window:** Displays the assembly code with comments explaining the operations. The comments are as follows:
 - MOV AX, DATA
 - MOV DS, AX
 - MOV AL, P : Load P into AL
 - MOV BL, Q : Load Q into BL
 - CALL ADD_BYTE : Call the ADD_BYTE procedure to add AL and BL
 - MOV CL, AL : Move result of addition into CL
 - MOV AL, R : Load R into AL
 - MOV BL, S : Load S into BL
 - CALL ADD_BYTE : Call ADD_BYTE to add AL and BL
 - MUL CL : Multiply CL and AL
 - MOV Z, AX : Move result of multiplication into Z
 - INT 3H : Terminate program
 - ADD_BYTE PROC
 - ADD AL, BL : Add AL and BL
 - RET : Return from procedure
- Registers Window:** Shows the CPU register values. The AX register has a value of 12 (hex 000C). Other registers like BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, and ES are initialized to 0.
- Memory Dump Window:** Shows memory starting at address F400:0104. The first byte is FF (opcode), followed by 255 (immediate value). The instruction is identified as ADD EBX + SI, A.

Subject: MICROPROCESSOR	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No:L004	Name of Subject Teacher: Pragati Mali
Name of Student: Siddharth Shah	Roll Id: 22203A0041

Experiment No:	12
Title of Experiment	Write an ALP for $Z=(A+B)*(C+D)$ using macro.

• **PROGRAM:-**

DATA SEGMENT

A DB 04H

B DB 04H

C DB 01H

D DB 02H

R1 DB 00H

R2 DB 00H

Z DW 00H

DATA ENDS

SUM_BYTE MACRO A1, A2, RES

MOV AL, A1 ; Move A1 into AL

ADD AL, A2 ; Add A2 to AL

MOV RES, AL ; Move the result in AL to RES

ENDM

CODE SEGMENT

ASSUME CS: CODE, DS: DATA

START:

MOV AX, DATA

MOV DS, AX

SUM_BYT E A, B, R1 ; Call SUM_BYT E macro to add A and B, store result in R1

SUM_BYT E C, D, R2 ; Call SUM_BYT E macro to add C and D, store result in R2

MOV AL, R1 ; Move R1 into AL

MUL R2 ; Multiply R2 with AL, result in AX

MOV Z, AX ; Move result of multiplication into Z

INT 3H

CODE ENDS

END START

• OUTPUT:-