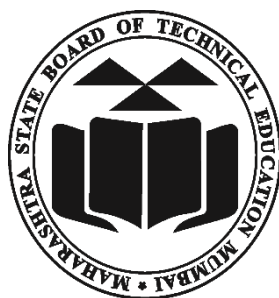


**A Laboratory Manual
for
Microprocessor
(22415)**

Semester – IV

(CO, CM, CW)



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

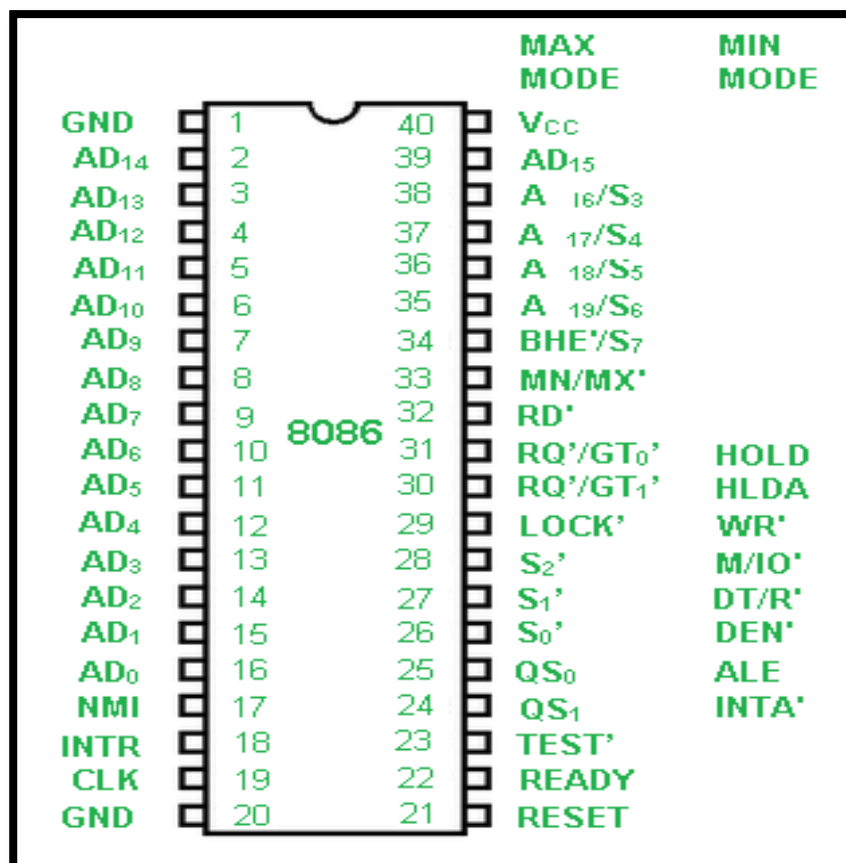
Sr.no	Practical Outcome	Date of performance	Date of submission	Marks	Remark
1	Identify various pins of the given microprocessor.	06/01/2024	13/01/2024		
2	Use Assembly Language Programming Tools and functions	13/01/2024	20/01/2024		
3	Use different addressing mode instruction in program Write an Assembly 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.	20/01/2024	27/01/2024		
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.	27/01/2024	03/02/2024		
5	(a) Write an ALP to divide two unsigned numbers 02 (b) Write an ALP to divide two signed numbers	03/02/2024	10/02/2024		
6	Write an ALP to add, subtract, multiply, divide two BCD numbers.	10/02/2024	17/02/2024		
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.	17/02/2024	24/02/2024		
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.	24/02/2024	02/03/2024		

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.	02/03/2024	09/03/2024		
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.	09/03/2024	16/03/2024		
11	Write an ALP for $Z = (A + B) * (C + D)$ using Procedure	16/03/2024	16/03/2024		
12	Write an ALP for $Z = (A + B) * (C + D)$ using MACRO.	16/03/2024	16/03/2024		

Subject: MICROPROCESSOR	Subject Code: 22415
Semester: 4 th Semester	Course: Computer Engineering
Laboratory No: L004B	Name of Subject Teacher: MISS.PRAGATI MALI
Name of Student: Aditya G. Makwana	Roll Id: 22203A0042

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

- **AIM:** Identify the various pins of the 8086 Microprocessor.
- **DIAGRAM:**



- **PIN DESCRIPTION:**

<i>Pin number</i>	<i>Pin name</i>	<i>Function</i>
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 trans receiver. 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 Trans receiver 8286. The trans receiver 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: MICROPROCESSORS	Subject Code:22415
Semester: 4th Semester	Course: Computer Engineering
Laboratory No: L004C	Name of Subject Teacher: PRAGATI MALI
Name of Student: ADITYA G. MAKWANA	Roll Id: 22203A0042

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

• ASSEMBLY LANGUAGE TOOLS

<u>Tools</u>	<u>Function</u>	<u>Software Used</u>
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.

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)

1. 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

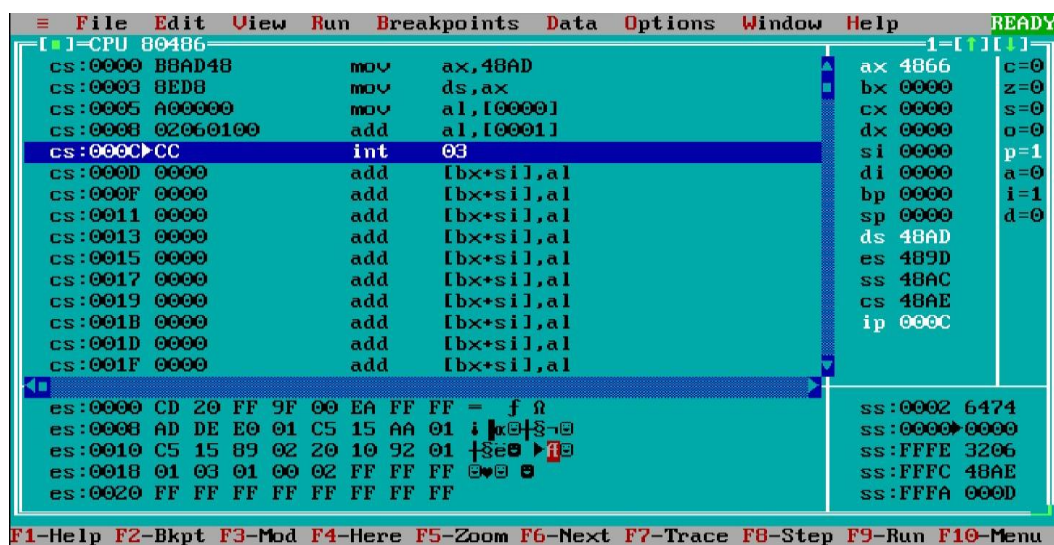
    ADD AL, NUM2

    INT 3

CODE ENDS

END START
```

- Output



The screenshot displays a DOS-based assembly debugger interface. The CPU is identified as 80486. The instruction pointer (IP) is 0000, and the instruction being executed is 'int 03'. The register window shows the following values: 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=0000. The memory window shows the program code starting at address 0000, with instructions like 'mov ax, 48AD', 'mov ds, ax', 'mov al, [0000]', 'add al, [0001]', 'int 03', and a series of 'add [bx+si], al' instructions.

2. 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

```
File Edit View Run Breakpoints Data Options Window Help READY
[CPU 80486] 1=[F1][F2][F3][F4][F5][F6][F7][F8][F9][F10]
cs:0000 B8AD48 mov ax,48AD ax 6556 c=0
cs:0003 BED8 mov ds,ax bx 0000 z=0
cs:0005 A10000 mov ax,[0000] cx 0000 s=0
cs:0008 03060200 add ax,[0002] dx 0000 o=0
cs:000C CC int 03 si 0000 p=1
cs:000D 0000 add [bx+si],al di 0000 a=0
cs:000F 0000 add [bx+si],al bp 0000 i=1
cs:0011 0000 add [bx+si],al sp 0000 d=0
cs:0013 0000 add [bx+si],al ds 48AD
cs:0015 0000 add [bx+si],al es 489D
cs:0017 0000 add [bx+si],al ss 48AC
cs:0019 0000 add [bx+si],al cs 48AE
cs:001B 0000 add [bx+si],al ip 000C
cs:001D 0000 add [bx+si],al
cs:001F 0000 add [bx+si],al
es:0000 CD 20 FF 9F 00 EA FF FF = f 0
es:0008 AD DE E0 01 C5 15 AA 01 i 0 0 0 0 0 0 0 0
es:0010 C5 15 89 02 20 10 92 01 s 0 0 0 0 0 0 0 0
es:0018 01 03 01 00 02 FF FF FF 0 0 0 0 0 0 0 0
es:0020 FF FF FF FF FF FF FF FF
ss:0002 6474
ss:0000 0000
ss:FFFE 3206
ss:FFFC 48AE
ss:FFFA 000D
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

3. 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 displays the DEBUG utility interface. The top menu bar includes File, Edit, View, Run, Breakpoints, Data, Options, Window, and Help. The CPU register window on the right shows the following values: ax=0192, bx=2110, cx=01A2, dx=3282, si=F6D6, di=03AC, bp=0100, sp=0106, ds=2110, es=489D, ss=0192, cs=0000, ip=0000. The memory window at the bottom shows the stack area with values: 742E, 706C, 6568, 6474, 4400. The command window shows the program code being executed, with the current instruction being 'int 03' at address 48AE:000C.

4. 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

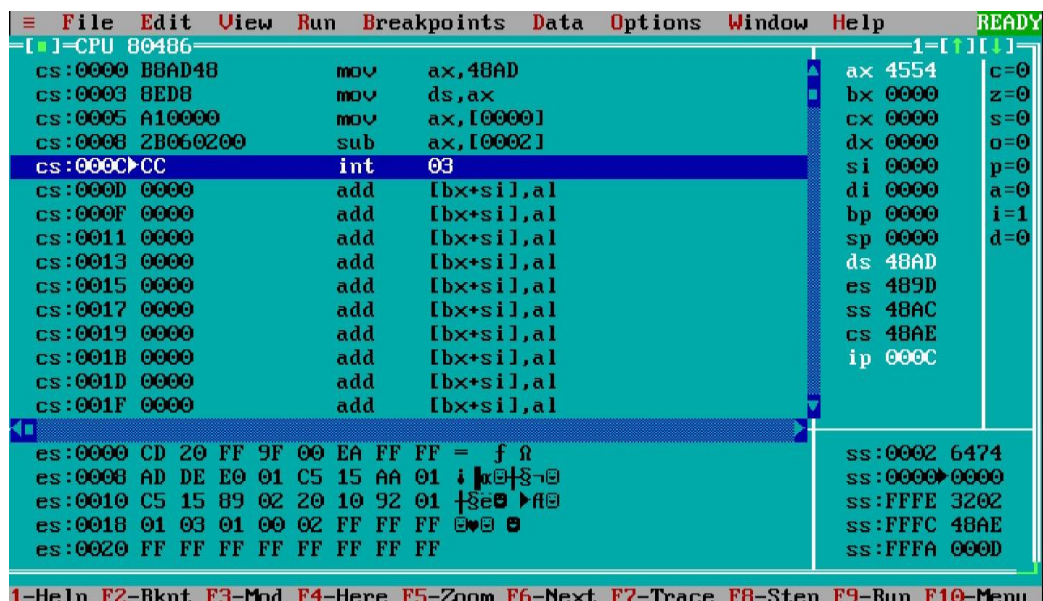
    MOV AX, NUM1

    SUB AX, NUM2

    INT 3

CODE ENDS
END START
```

- Output



```
File Edit View Run Breakpoints Data Options Window Help READY
[CPU 80486] 1-[[[[]]]
cs:0000 B8AD4B mov ax,4B9D
cs:0003 B8EDB 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
es:0000 CD 20 FF 9F 00 EA FF FF = f 0
es:0003 AD DE E0 01 C5 15 AA 01 i k 0 S 0
es:0010 C5 15 B9 02 20 10 92 01 +Se 0 A 0
es:0018 01 03 01 00 02 FF FF FF 0 0 0
es:0020 FF FF FF FF FF FF FF FF
ss:0002 6474
ss:0003 0000
ss:FFFE 3202
ss:FFFC 4BAE
ss:FFFA 000D
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:22414
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:	4
Title of Experiment	Write an assembly language program for multiplication of 8 bit and 16 bit signed and unsigned numbers

- **Software used :**

Emu8086

- **PROGRAM CODE :**

1. Write an ALP to multiply two given 8-bits numbers.

ANS: DATA SEGMENT

NUM1 DB 10H

NUM2 DB 05H

DATA ENDS

CODE SEGMENT ASSUME

CS:CODE,DS:DATA

START:

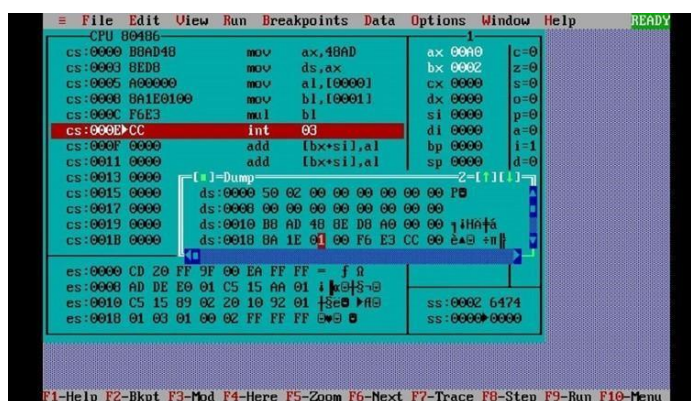
MOV AX,DATA

```

MOV DS,AX
MOV AL,NUM1
MUL AL,NUM2
INT 3
CODE ENDS
END START

```

Output:



2. Write an ALP to multiply two given 16-bits numbers.

ANS: 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:

```
File Edit View Run Breakpoints Data Options Window Help
[CPU] 80486 1-[F1][F2]
cs:0000 B8AD48 mov ax,4BAD ax 2000 c=0
cs:0003 8ED8 mov ds,ax bx 0001 z=0
cs:0005 A10000 mov ax,[0000] cx 0000 s=0
cs:0008 8B1E0200 mov bx,[0002] dx 0000 o=0
cs:000C F7E3 mul bx si 0000 p=0
cs:000E CC int 03 di 0000 a=0
cs:000F 0000 add [bx*si],al bp 0000 i=1
cs:0011 0000 add [bx*si],al sp 0000 d=0
cs:0013 0000 add [bx*si],al ds 4BAD
cs:0015 0000 add [bx*si],al es 4B9D
cs:0017 0000 add [bx*si],al ss 4BAC
cs:0019 0000 add [bx*si],al cs 4BAE
cs:001B 0000 add [bx*si],al ip 000E
es:0000 CD 20 FF 9F 00 EA FF FF = f 8
es:0000 AD DE E0 01 C5 15 AA 01 1 0 0 0 0 0 0 0
es:0010 C5 15 09 02 20 10 32 01 8 0 0 0 0 0 0
es:0018 01 03 01 00 02 FF FF FF 0 0 0 0 0 0
ss:0000 6474
ss:0000 0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

```
[Dump] 2-[F1][F2]
ds:0000 00 20 01 00 00 00 00 00
ds:0003 00 00 00 00 00 00 00 00
ds:0010 B8 AD 48 8E DB A1 00 00 1 0 0 0 0 0
ds:0018 8B 1E 02 00 F7 E3 CC 00 1 0 0 0 0 0
```

3. Write an ALP to multiply two given 8-bits signed numbers.

ANS: DATA SEGMENT

A DB 0F2H

B DB 09H

C DW ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,CS: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:

The screenshot displays an 8086 emulator window titled "emulator: IMUL 8.exe_". The assembly code is shown in the background, and the foreground shows the emulator's interface with the following components:

- Registers:** A table on the left showing the state of various registers.

Register	H	L
AX	FF	82
BX	00	09
CX	00	28
DX	00	00
CS	0711	
IP	0017	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	
- Memory Window:** A central pane showing memory addresses and their contents. The address 07127: is highlighted in blue, showing the value CC 204.

Address	Value
07122:	F6 246
07123:	EB 235
07124:	A3 163
07125:	02 002
07126:	00 000
07127:	CC 204
07128:	90 144
07129:	90 144
0712A:	90 144
0712B:	90 144
0712C:	90 144
0712D:	90 144
0712E:	90 144
0712F:	90 144
07130:	90 144
07131:	90 144
- Code Window:** A pane on the right showing the assembly code being executed. The instruction "INT 3" is highlighted in blue.

```

MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
IMUL BL
MOV [00002h], AX
INT 3
NOP
NOP
NOP
NOP
NOP
NOP
...

```

4. Write an ALP to multiply two given 16-bits signed numbers.

ANS: DATA SEGMENT

A DW 0F002H

B DW 1001H

C DD ?

DATA ENDS

CODE SEGMENT

ASSUME DS:CODE,CS: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

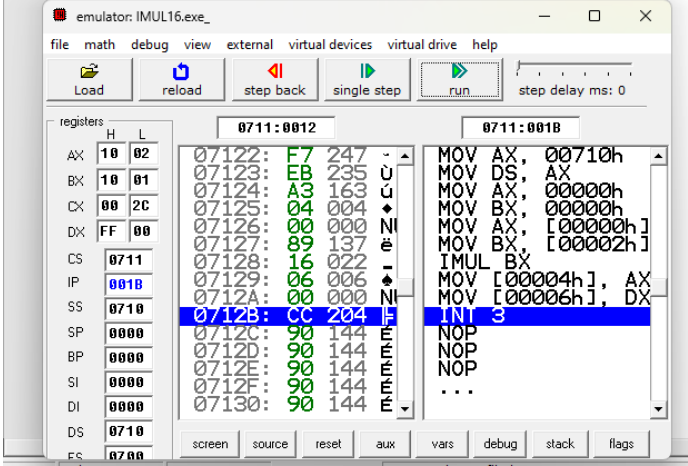
INT 3

CODE ENDS

END START

Output:

```
01 DATA SEGMENT
02 A DW 0F002H
03 B DW 1001H
04 C DD ?
05 DATA ENDS
06
07 CODE SEGMENT
08 ASSUME DS:CODE,CS:DATA
09 START:
10 MOV AX,DATA
11 MOV DS,AX
12 MOV AX,0000H
13 MOV BX,0000H
14 MOV AX,A
15 MOV BX,B
16 IMUL BX
17 MOV WORD PTR C,AX
18 MOV WORD PTR C+2,DX
19 INT 3
20 CODE ENDS
21 END START
```



- Conclusion :

In above practical we learn how to do the multiplication of signed and Unsigned 8 bit and 16-bit numbers



DEPARTMENT OF COMPUTER ENGINEERING

Subject: MIC	Subject Code:22414
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:	5
Title of Experiment	Write an assembly language program for division of 8 bit and 16 bit signed and unsigned numbers

- **Software used :**

Emu8086

- **PROGRAM CODE :**

1. Write an ALP to divide two given 8-bits numbers.

ANS: DATA SEGMENT

A DB 34H

B DB 05H

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

```

Output:

```

01 DATA SEGMENT
02   A DB 34H
03   B DB 05H
04   C DB ?
05 DATA ENDS
06
07 CODE SEGMENT
08   ASSUME CS:CODE,DS:DATA
09   START:
10     MOV AX,DATA
11     MOV DS,AX
12     MOV AX,0000H
13     MOV BX,0000H
14     MOV AL,A
15     MOV BL,B
16     DIV BL
17     MOV C,AL
18     INT 3H
19   CODE ENDS
20 END START

```

emulator: DIV8.exe_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers	H	L
AX	02	0A
BX	00	05
CX	00	28
DX	00	00
CS	0711	
IP	0017	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

0711:0017				0711:0017			
07122:	F6	246	÷	MOV	AX,	00710h	
07123:	F3	243	¾	MOV	DS,	AX	
07124:	A2	162	6	MOV	AX,	00000h	
07125:	02	002	0	MOV	BX,	00000h	
07126:	00	000	NI	MOV	AL,	[00000h]	
07127:	CC	204	IF	MOV	BL,	[00001h]	
07128:	90	144	EE	DIV	BL		
07129:	90	144	EE	MOV	[00002h],	AL	
0712A:	90	144	EE	INT	3		
0712B:	90	144	EE	NOP			
0712C:	90	144	EE	NOP			
0712D:	90	144	EE	NOP			
0712E:	90	144	EE	NOP			
0712F:	90	144	EE	NOP			
07130:	90	144	EE	NOP			
07131:	90	144	EE	...			

screen source reset aux vars debug stack flags

2. Write an ALP to divide two given 16-bits numbers.

ANS: 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

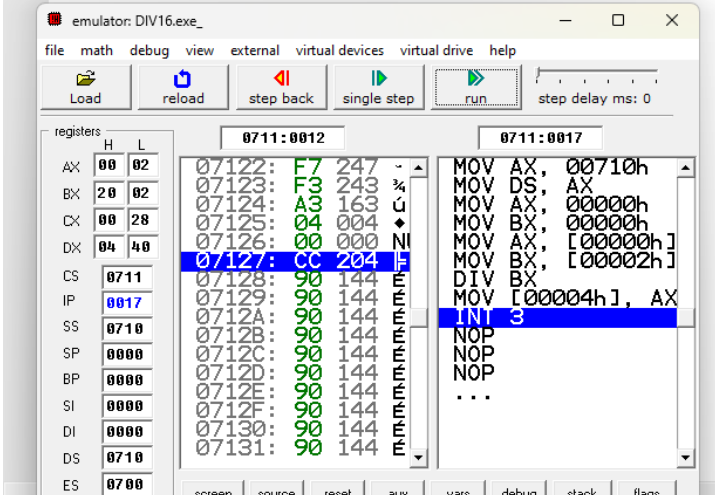
INT 3H

CODE ENDS

END START

Output:

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



The screenshot shows the DIV16.exe emulator window. The registers window on the left displays the following values: AX: 00, BX: 20, CX: 00, DX: 04, CS: 0711, IP: 0017, SS: 0710, SP: 0000, BP: 0000, SI: 0000, DI: 0000, DS: 0710, ES: 0700. The main window shows the assembly code being executed, with the instruction 'INT 3' highlighted in blue. The status bar at the bottom shows 'screen', 'source', 'reset', 'aux', 'vars', 'debug', 'stack', and 'flags'.

3. Write an ALP to divide two given 8-bits signed numbers.

ANS: 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

Output:

The screenshot displays an 8086 emulator interface. The top pane shows the assembly code being executed, with line numbers 01 through 22. The code defines a data segment with variables A, B, and C, and a code segment starting at START. The instructions include MOV DS, AX, MOV AX, 0000H, MOV BX, 0000H, MOV AL, A, MOV BL, B, IDIV BL, MOV C, AL, INT 3H, and CODE ENDS. The bottom pane shows the execution progress, with the instruction pointer (IP) at 0017. The registers window on the left shows the current state of the registers, and the right pane shows the disassembled instructions being executed, including MOV AX, 00710h, MOV DS, AX, MOV AX, 00000h, MOV BX, 00000h, MOV AL, [00000h], MOV BL, [00001h], IDIV BL, MOV [00002h], AL, and INT 3.

```

01 DATA SEGMENT
02     A DB 0F2H
03     B DB 09H
04     C DB ?
05 DATA ENDS
06
07 CODE SEGMENT
08     ASSUME CS:CODE, DS:DATA
09     START:
10         MOV AX, DATA
11         MOV DS, AX
12         MOV AX, 0000H
13         MOV BX, 0000H
14         MOV AL, A
15         MOV BL, B
16         IDIV BL
17         MOV C, AL
18         INT 3H
19     CODE ENDS
20 END START
21
22

```

emulator: IDIV8.exe_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	08	1A
BX	00	09
CX	00	28
DX	00	00
CS	0711	
IP	0017	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

0711:0017

Address	Hex	Dec	Op
07122:	F6	246	÷
07123:	FB	251	
07124:	A2	162	
07125:	02	002	
07126:	00	000	
07127:	CC	204	
07128:	90	144	
07129:	90	144	
0712A:	90	144	
0712B:	90	144	
0712C:	90	144	
0712D:	90	144	
0712E:	90	144	
0712F:	90	144	
07130:	90	144	
07131:	90	144	

MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
IDIV BL
MOV [00002h], AL
INT 3
NOP
NOP
NOP
NOP
NOP
NOP
...

screen source reset aux vars debug stack flags

4. Write an ALP to divide two given 16-bits signed numbers.

ANS:

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

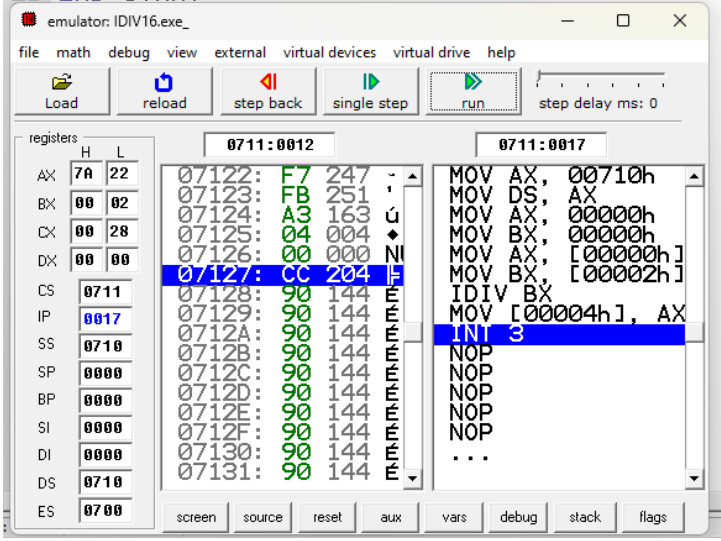
INT 3H

CODE ENDS

END START

Output:

```
01 DATA SEGMENT
02 A DW 0F444H
03 B DW 0002H
04 C DW ?
05 DATA ENDS
06
07 CODE SEGMENT
08 ASSUME CS:CODE,DS:DATA
09 START:
10 MOV AX,DATA
11 MOV DS,AX
12 MOV AX,0000H
13 MOV BX,0000H
14 MOV AX,A
15 MOV BX,B
16 IDIV BX
17 MOV C,AX
18 INT 3H
19 CODE ENDS
20 END START
21
```



- Conclusion :

In above practical we learn how to do the division of signed and Unsigned 8 bit and 16-bit numbers



DEPARTMENT OF COMPUTER ENGINEERING

Subject: MIC	Subject Code:22414
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:	6
Title of Experiment	Write an assembly language program for BCD addition, subtraction, multiplication and division of 8 bit numbers

- **Software used :**

Emu8086

- **PROGRAM CODE :**

1. Write an ALP to BCD add two given 8-bits numbers.

ANS: DATA SEGMENT

A DB 80H

B DB 26H

C DW ?

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

MOV C,AX

NEXT:

INT 3H

CODE ENDS

END START

Output:

The screenshot displays an 8086 emulator interface with the following components:

- Assembly Code Window (Left):** Shows the assembly code with line numbers 01 to 21. The code includes data segment definitions, a start routine, and an interrupt call.
- Registers Window (Top Right):** Displays the state of 16-bit registers. The AX register is highlighted with a value of 0014h. Other registers like BX, CX, DX, SI, DI, and DS are also visible.
- Source Code Window (Bottom Left):** A smaller view of the assembly code, with the 'INT 3H' instruction highlighted in yellow.
- Flags Window (Bottom Center):** Shows the status of various CPU flags. The Carry Flag (CF) is set to 1, while others like Zero Flag (ZF), Sign Flag (SF), and Overflow Flag (OF) are 0.
- Lexical Flag Analyser (Bottom Right):** A tool that interprets the flag values. It shows 'unsigned below' (CF=1) and 'not less' (SF=0F), indicating the result of the previous operation.

2. Write an ALP to BCD subtract two given 16-bits numbers.

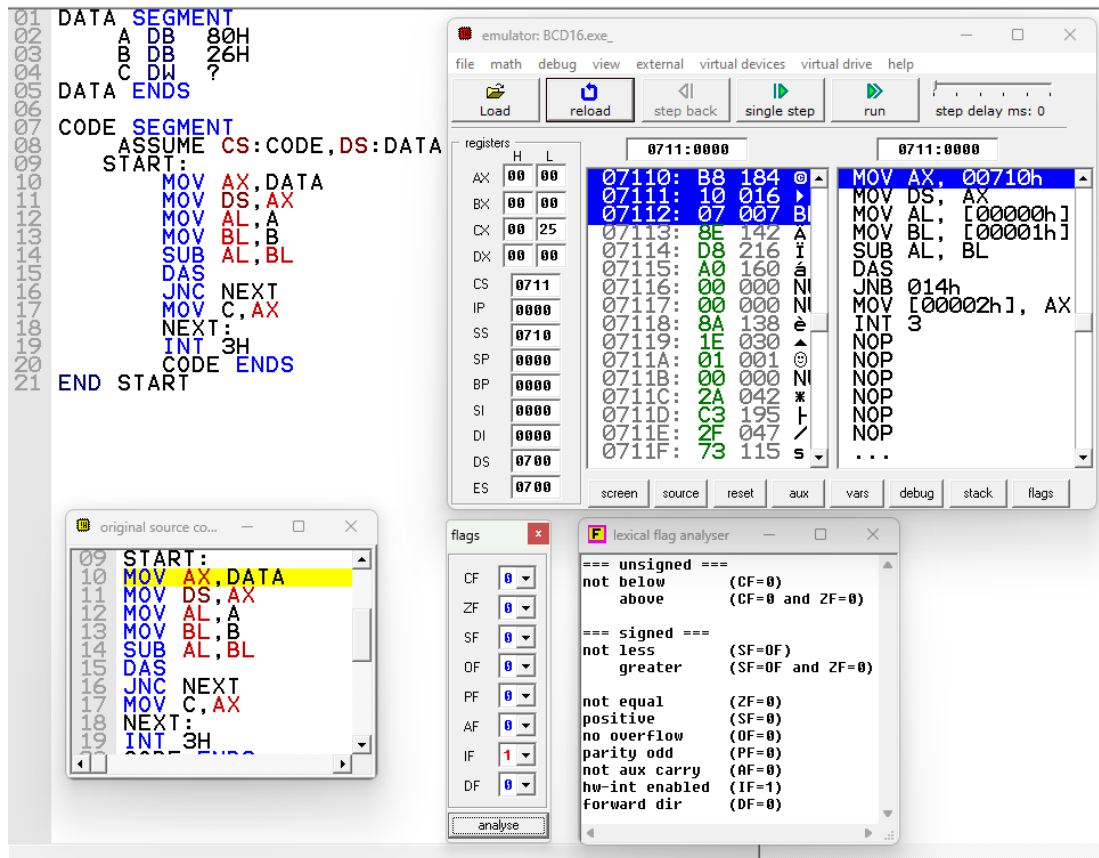
ANS: DATA SEGMENT

```
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
```

Output:



3. Write an ALP to BCD multiply two given 8-bits numbers.

ANS:

DATA SEGMENT

A DB 12H

B DB 09H

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START:

MOV AX, DATA

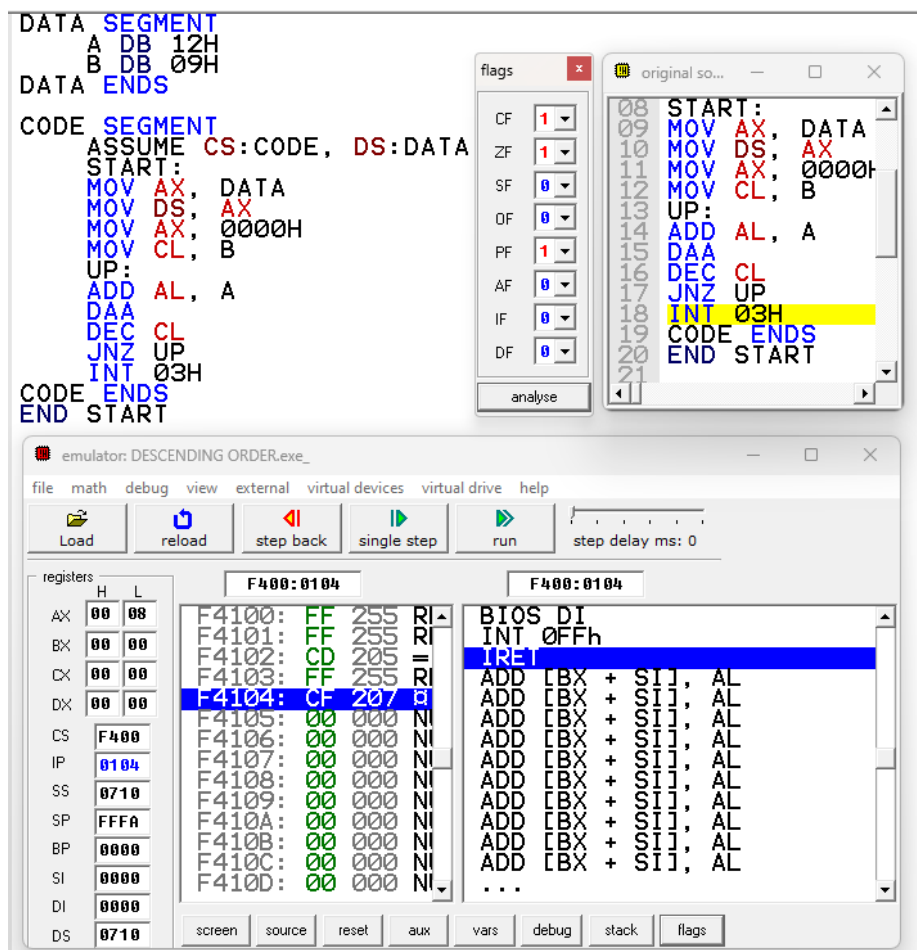
MOV DS, AX

```

MOV AX, 0000H
MOV CL, B
UP:
ADD AL, A
DAA
DEC CL
JNZ UP
INT 03H
CODE ENDS
END START

```

Output:



4. Write an ALP to BCD divide two given 8-bits numbers.

ANS:

DATA SEGMENT

A DB 12H

B DB 09H

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 50H ; Dividend (in BCD)
B DB 05H ; Divisor (in BCD)
DATA ENDS

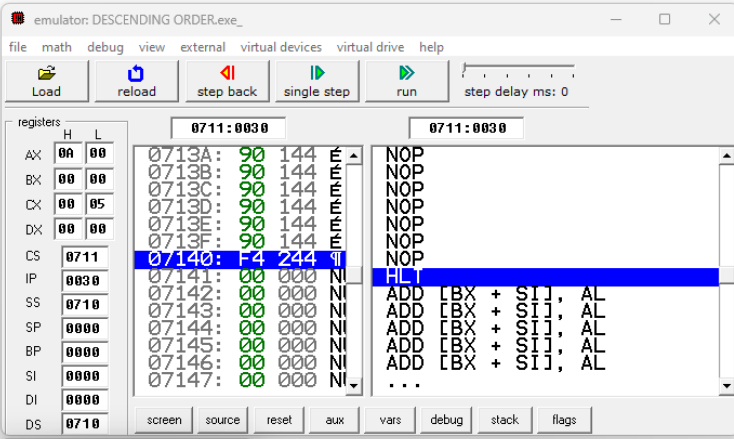
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
MOV AX, DATA
MOV DS, AX

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
```



emulator: DESCENDING ORDER.exe

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	00	00
BX	00	00
CX	00	05
DX	00	00
CS	0711	
IP	0030	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	

0711:0030 0711:0030

0713A: 90 144 NOP
0713B: 90 144 NOP
0713C: 90 144 NOP
0713D: 90 144 NOP
0713E: 90 144 NOP
0713F: 90 144 NOP
07140: F4 244 J
07141: 00 000 NI
07142: 00 000 NI
07143: 00 000 NI
07144: 00 000 NI
07145: 00 000 NI
07146: 00 000 NI
07147: 00 000 NI
...
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL

screen source reset aux vars debug stack flags

original so... 15 MOV CL, B
16 MOV CH, 00H
17
18
19
20 DIV_LOOP:
21 CMP AL, CL
22 JB DIV_END
23 SUB AL, CL
24 DAS
25 INC AH
26 JMP DIV_LOOP

- Conclusion :

In above practical we learn how to do the BCD addition ,subtraction ,division , multiplicatio
of given numbers



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No: L004	Name of Subject Teacher: Pragati Mali
Name of Student: Aditya Makwana	Roll Id: 22203A0042

Experiment No:	7
Title of Experiment	a) Write an ALP to perform block data transfer using string instructions. b) Write an ALP to perform block data transfer without using string instructions

1. Write an ALP to perform block data transfer without using string instructions?

ANS:

PROGRAM:

```
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 the emu8086 debugger interface. The main window displays the assembly code with the current instruction highlighted. The registers window shows the state of the CPU registers. The flags window shows the status of the CPU flags. The variables window shows the memory contents of the variables.

Assembly Code (Main Window):

```

0000: SRC_BLK DB 11h, 22h, 33h, 44h, 55h
0001: DST_BLK DB 5 DUP(0)
0002: DATA ENDS
0003: CODE SEGMENT
0004: ASSUME CS:CODE, DS:DATA
0005: START:
0006: MOV AX, DATA
0007: MOV DS, AX
0008: MOV CX, 05H
0009: LEA SI, SRC_BLK
0010: LEA BX, DST_BLK
0011: UP:
0012: MOV AX, [SI]
0013: MOV [BX], AX
0014: INC SI
0015: INC BX
0016: DEC CX
0017: JNZ UP
0018: INT 03H
0019: CODE ENDS
0020: END START

```

Registers Window:

Register	H	L
AX	00	00
BX	00	00
CX	00	28
DX	00	00
CS	0711	
IP	0000	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

Flags Window:

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

Variables Window:

Variable	Value
SRC_BLK	11h, 22h, 33h, 44h, 55h
DST_BLK	00h, 00h, 00h, 00h, 00h

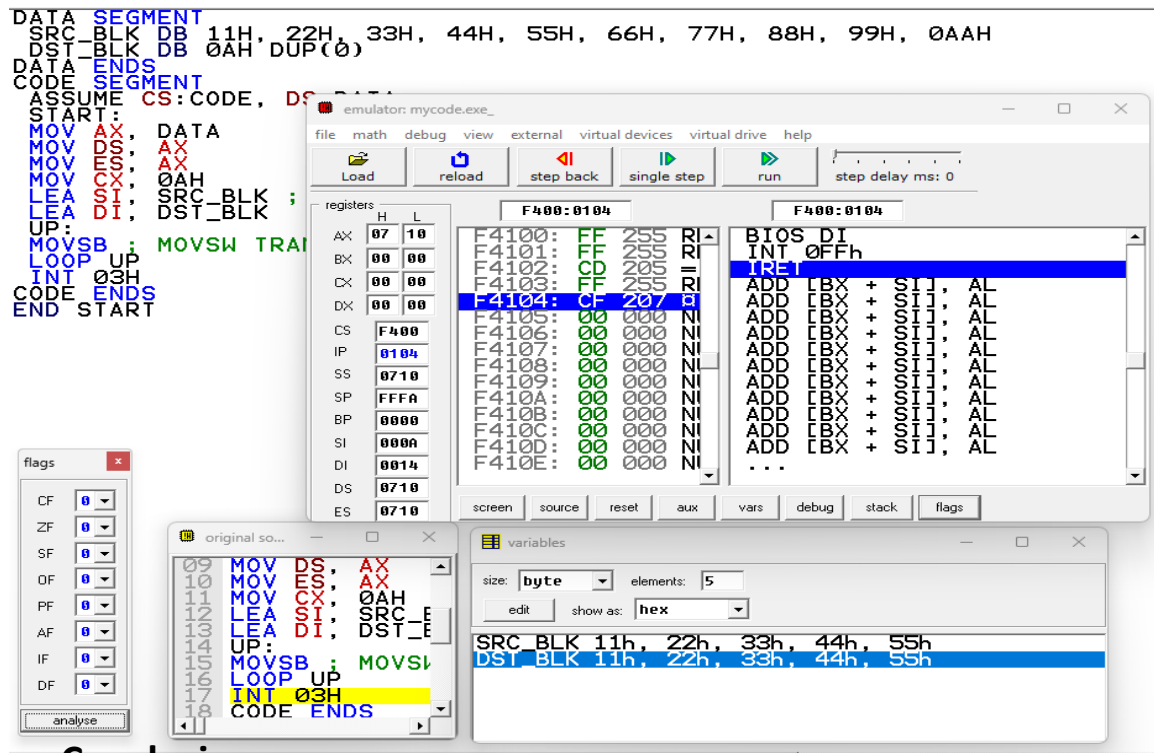
2. Write an ALP to perform block data transfer using string instructions ?

ANS:

PROGRAM:

```
DATA SEGMENT
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 ; MOV SI, OFFSET SRC_BLK
LEA DI, DST_BLK
UP:
MOVSX ; MOVSW TRANSFER TWO BYTES
LOOP UP
INT 03H
CODE ENDS
END START
```


OUTPUT:



• Conclusion:

In this practical we learnt how to operate on array to sort it in ascending as well as descending order.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: Microprocessor	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No: L004	Name of Subject Teacher: Pragati Mali
Name of Student: Aditya Makwana	Roll Id: 22203A0042

Experiment No:	8
Title of Experiment	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.

ANS:

PROGRAM:

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

```

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

ANS:

PROGRAM:

DATA SEGMENT

```

SERIES DB 05H, 02H, 02H, 11H, 05H SUM
DB ?
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 displays an x86 emulator window titled "emulator: DESCENDING ORDER.exe_". The interface includes a menu bar (file, math, debug, view, external, virtual devices, virtual drive, help), a toolbar with buttons for Load, reload, step back, single step, run, and a step delay slider, and a status bar showing "step delay ms: 0".

On the left side of the emulator, the assembly code is loaded and color-coded. The code is as follows:

```

DATA SEGMENT
SERIES DB 05H, 02H, 02H, 11H, 05H
SUM DB ?
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

```

The registers window shows the following values:

Register	H	L
AX	00	25
BX	00	05
CX	00	00
DX	00	25
CS	F400	
IP	0104	
SS	0710	
SP	FFFF	
BP	0000	
SI	0000	
DI	0000	
DS	0710	

The memory window shows the following data:

Address	Value	Comment
F4100:	FF 25 55 RI	BIOS DI
F4101:	FF 25 55 RI	INT 0FFh
F4102:	CD 20 55 RI	IRET
F4103:	FF 25 55 RI	
F4104:	CF 20 70 RI	ADD [BX], AL
F4105:	00 00 00 NI	ADD [BX], AL
F4106:	00 00 00 NI	ADD [BX], AL
F4107:	00 00 00 NI	ADD [BX], AL
F4108:	00 00 00 NI	ADD [BX], AL
F4109:	00 00 00 NI	ADD [BX], AL
F410A:	00 00 00 NI	ADD [BX], AL
F410B:	00 00 00 NI	ADD [BX], AL
F410C:	00 00 00 NI	ADD [BX], AL
F410D:	00 00 00 NI	ADD [BX], AL

The flags window shows the following values:

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

The variables window shows the following values:

Variable	Value
SERIES	05h, 02h, 02h, 11h, 05h
SUM	25h

- **Conclusion:**

In this practical we learned how to find sum of series of Hexadecimal and BCD numbers.

Subject: Microprocessor	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No: L004	Name of Subject Teacher: Pragati Mali
Name of Student: Aditya Makwana	Roll Id: 22203A0042

Experiment No:	9
Title of Experiment	a) Write an ALP to find smallest number in an array. b) Write an ALP to find largest number in an array.

1. Write an ALP to find smallest number in an array?

ANS:

PROGRAM:

```

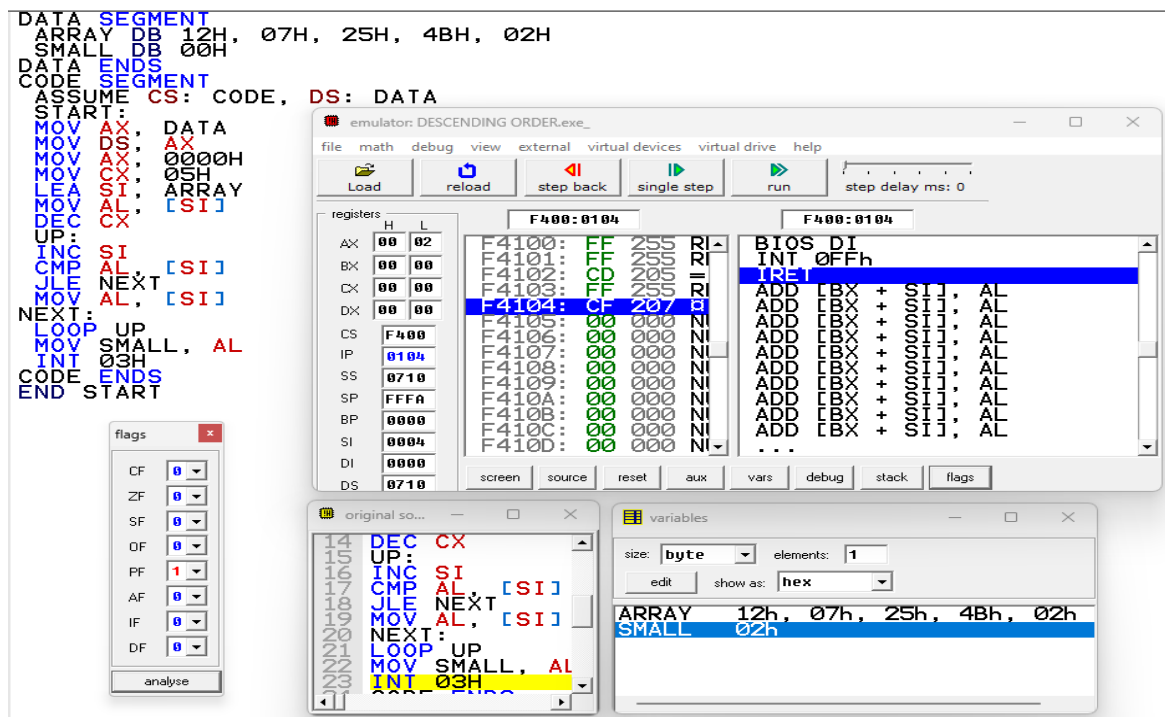
DATA SEGMENT
ARRAY DB 12H, 07H, 25H, 4BH, 02H
SMALL DB 00H
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START:
MOV AX, DATA
MOV DS, AX
MOV AX, 0000H
MOV CX, 05H
LEA SI, ARRAY
MOV AL, [SI]
DEC CX
  
```

```

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

```

OUTPUT:



2. Write an ALP to perform block data transfer using string instructions.

ANS:

PROGRAM:

```

DATA SEGMENT
SERIES DB 05H, 02H, 02H, 11H, 05H
SUM DB ?
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 displays an x86 emulator interface with several windows:

- Assembly Code Window:** Shows the assembly code being executed. The current instruction is `INT 03H` at address `0000:0000`. The code includes a data segment, a loop, and a call to `INT 03H`.
- Registers Window:** Shows the state of the registers. The `AX` register contains `00 70`, `SI` contains `0004`, and `DI` contains `0000`.
- Flags Window:** Shows the state of the flags. The `CF` (Carry Flag) is set to `0`, and the `DF` (Direction Flag) is set to `0`.
- Memory Window:** Shows the memory contents. The `BIOS DI` segment is visible, with `INT 0FFh` at address `0000:0000`.
- Variables Window:** Shows the state of variables. The `ARRAY` variable is set to `12h, 07h, 25h, 70h, 02h`, and the `LARGE` variable is set to `70h`.

- **Conclusion:**

With this practical we learned how to find largest and smallest number in an array.

Subject: Microprocessor	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No: L004	Name of Subject Teacher: Pragati Mali
Name of Student: Aditya Makwana	Roll Id: 22203A0042

Experiment No:	10
Title of Experiment	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 ?

ANS:

PROGRAM:

```

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] ; Load the current element into AL?

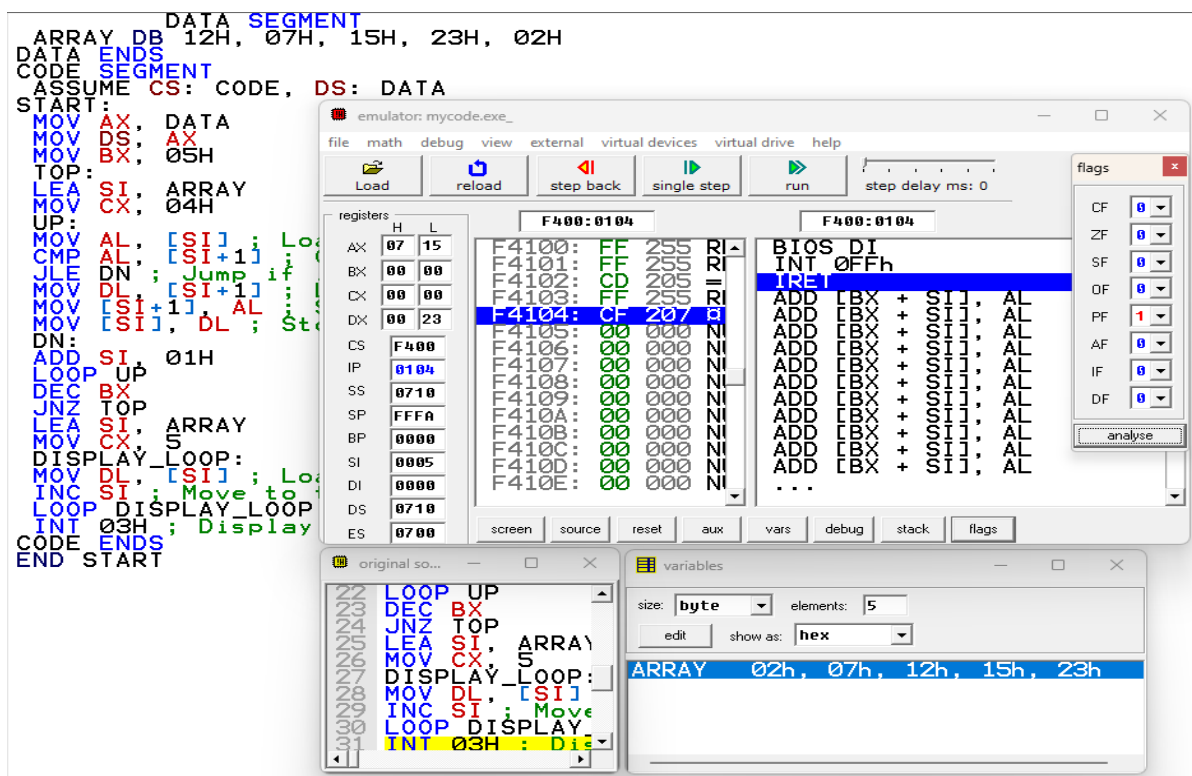
```

```

CMP AL, [SI+1] ; Compare with the next element?
JLE DN ; Jump if less than or equal, skip swap?
MOV DL, [SI+1] ; Load the next element into DL?
MOV [SI+1], AL ; Store AL into the next position?
MOV [SI], DL ; Store DL into the current position?
DN:
ADD SI, 01H
LOOP UP
DEC BX
JNZ TOP
LEA SI, ARRAY
MOV CX, 5
DISPLAY_LOOP:
MOV DL, [SI] ; Load the value to be displayed?
INC SI ; Move to the next position?
LOOP DISPLAY_LOOP ; Loop until all elements are displayed ?
INT 03H ; Display the character ?
CODE ENDS
END START

```

OUTPUT:



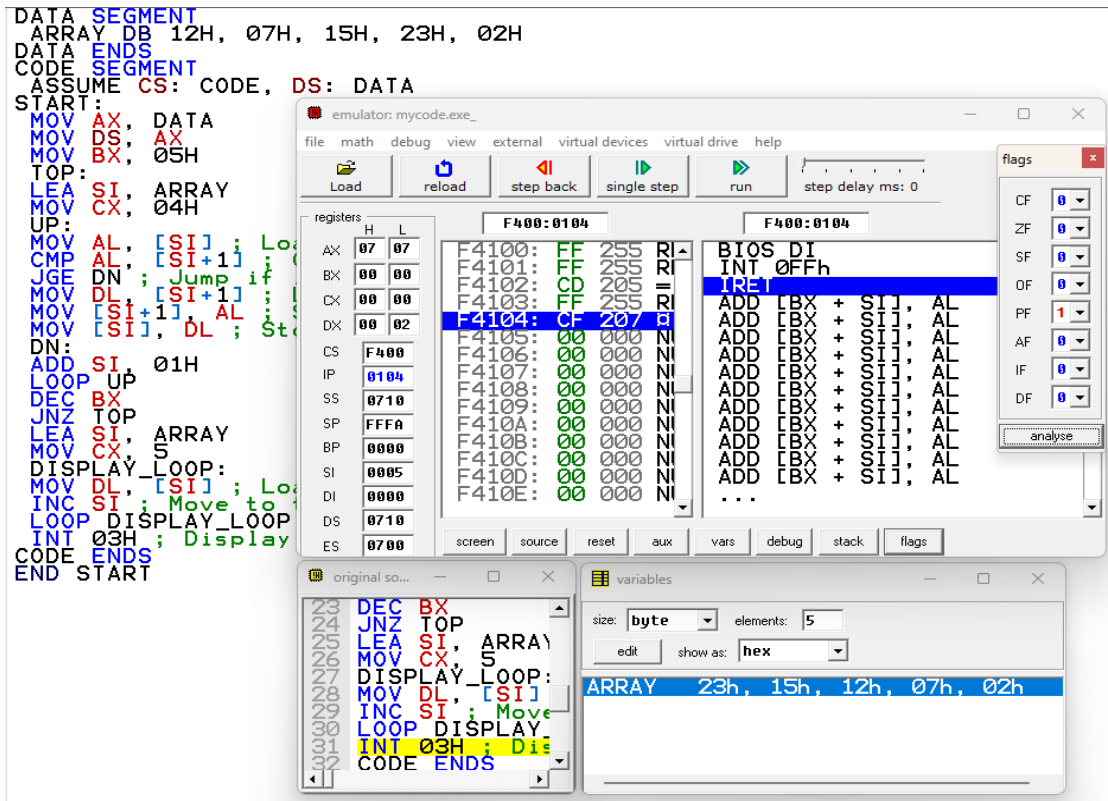
2. Write an ALP program to arrange array in Descending order ?

ANS:

PROGRAM:

```
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] ; Load the current element into AL?
CMP AL, [SI+1] ; Compare with the next element?
JGE DN ; Jump if less than or equal, skip swap?
MOV DL, [SI+1] ; Load the next element into DL?
MOV [SI+1], AL ; Store AL into the next position?
MOV [SI], DL ; Store DL into the current position?
DN:
ADD SI, 01H
LOOP UP
DEC BX
JNZ TOP
LEA SI, ARRAY
MOV CX, 5
DISPLAY_LOOP:
MOV DL, [SI] ; Load the value to be displayed?
INC SI ; Move to the next position?
LOOP DISPLAY_LOOP ; Loop until all elements are displayed ?
INT 03H ; Display the character ?
CODE ENDS
END START
```

OUTPUT:



- **Conclusion:**

In this practical we learnt how to operate on array to sort it in ascending as well as descending order.



DEPARTMENT OF COMPUTER ENGINEERING

Subject: MICROPROCESSOR	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No:L004	Name of Subject Teacher: Pragati Mali
Name of Student: Aditya Makwana	Roll Id: 22203A0042

Experiment No:	11
Title of Experiment	Write an ALP for $Z=(P+Q)*(R+S)$ using procedure.

- **PROGRAM:-**

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 ; 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

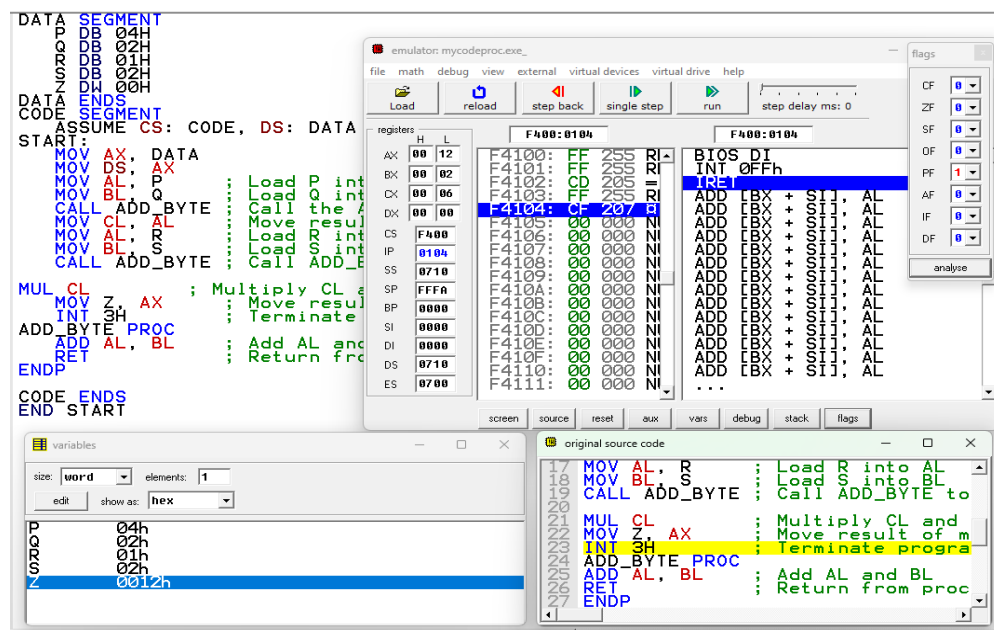
RET ; Return from procedure

ENDP

CODE ENDS

END START

- OUTPUT: -**





DEPARTMENT OF COMPUTER ENGINEERING

Subject: MICROPROCESSOR	Subject Code:22415
Semester:4 th Semester	Course: Computer Engineering
Laboratory No:L004	Name of Subject Teacher: Pragati Mali
Name of Student: Aditya Makwana	Roll Id: 22203A0042

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_BYTE A, B, R1 ; Call SUM_BYTE macro to add A and B, store result in R1

SUM_BYTE C, D, R2 ; Call SUM_BYTE 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: -**

