St. JOSEPH'S INSTITUTE OF TECHNOLOGY

St. Joseph's Group of Institutions

OMR, CHENNAI – 600 119

We Make You Shine

DEPARTMENT OF INFORMATION TECHNOLOGY

SUB CODE / LAB NAME: **EC8681 – MICROPROCESSOR AND MICROCONTORLLER LABORATORY**

Name: **A.R.Sharmila**

Year: **III** Semester: **V**

Branch: **B. Tech – INFORMATION TECHNOLOGY**

Roll No: **18IT1231** Reg. No: **312418205083**

INDEX

NAME OF THE STUDENT: **SHARMILA A.R.** DEPARTMENT: **IT**

ROLL NO: **18IT1231** REGISTER NO: **312418205083**

LAB INCHARGE: **Dr. KAVITHA KANNAN**

S.NO	DATE	NAME OF THE EXPERIMENTS	SUB. DATE	PAGE NO	MARK	SIGN
		CYCLE I				
1 a 1 b 1 c	1.10.2020	16-bit Addition / Subtraction using 8086 16-bit Multiplication / Division using 8086 Logical operations (and, or, xor, not) using 8086	08.10.2020			
2	1.10.2020	Move a data block without overlap	08.10.2020			
3 a 3 b 3 c	08.10.2020	Code Conversion using 8086 BCD Addition / Subtraction using 8086 Matrix operations using 8086	15.10.2020			
4 a 4 b 4 c	08.10.2020	String manipulations (copy, compare, find & replace) 8086 Sorting using 8086 Searching using 8086	15.10.2020			
5	15.10.2020	Stepper motor control	22.10.2020			
6	15.10.2020	Interfacing Programmable Keyboard of Display Controller	22.10.2020			
7	22.10.2020	Serial Interface and Parallel Interface	29.10.2020			
8	22.10.2020	A/D and D/A Interface and Waveform Generation	29.10.2020			
9	29.10.2020	EMULATOR PROGRAMS	04.11.2020			
		CYCLE II EXPERIMENTS				
10 a 10 b	04.11.2020	8-bit Addition / Subtraction using 8051 8-bit Multiplication / Division using 8051	11.11.2020			
11	04.11.2020	Logical Operation using 8051	11.11.2020			
12 a 12 b	04.11.2020	Finding square of an 8-bit number using 8051 Finding cube of an 8-bit number using 8051	11.11.2020			
13	11.11.2020	Finding 2's of an 8-bit number using 8051	17.11.2020			
14	11.11.2020	Unpacked BCD number to ASCII number using 8051	17.11.2020			

A.K. Sharmila
312418205083

Ex.No: 1(a) Programs for Basic arithmetic and logical operations.
Date: 18.10.20

Aim: To write an Assembly Language Program (ALP) to perform basic Arithmetic and Logical Operations.

(a) Addition of two numbers
(b) Subtraction of two numbers
(c) Multiplication of two numbers.
(d) Division of two numbers.
(e) Logical operation.

(i) Addition with carry.

Input:

1200 : 46
1201 : B6 [Addend]
1202 : D3
1203 : 98 [Augend]

Output:

1300 : 19
1301 : 4F [sum]
1302 : 01
1303 : 00 [carry]

Without carry

Input:

1200 : 34
1201 : 44 [Addend]
1202 : 24
1203 : 24 [Augend]

Output:

1300 : 58
1301 : 68 [Sum]
1302 : 00
1303 : 00 [carry]

(ii) Subtraction:

With Borrow

Input:

1200 : 03
1201 : 00 (minuend)
1202 : 05
1203 : 00 (Subtrahend)

Output:

1300 : 02 (Difference)
1301 : 00
1302 : 01 (Borrow)
1303 : 00

Without Borrow

Input:

1200 : 31
1201 : 82 (minuend)
1202 : 06
1203 : 34 (subtrahend)

Output

1300 : 2B (Diff)
1301 : 2E4E
1302 : 00
1303 : 00 (Borrow)

1 (B)

(iii) Multiplication

Input:

1200 : 02
1201 : 06 (Multiplicand)
1202 : 02
1203 : 06 (Multiplier)

Output:

1300 : 04 (Lower word of the pro)
1301 : 18
1302 : 24
1303 : 00 (Higher word of the pro.)

A.R. Sharmila
312418205083

(iv) Division

Input:

1200 : 06
1201 : 06 (Dividend)
1202 : 03
1203 : 03 (Divisor)

Output:

1300 : 02 (Quotient)
1301 : 00
1302 : 00 (Remainder)
1303 : 00

A.R. Sharmila
312418205083

1. Logical Operation
(c)

Aim: To perform logical operations on 16 bit data using 8086 microprocessor

Observation:

OR

Input:	Output
1000 - 01	1004 - 11
1001 - 11	1005 - 11
1002 - 10	
1003 - 00	

X-OR

Input	Output
1000 - 00	1004 - 01
1001 - 11	1005 - 01
1002 - 01	
1003 - 10	

AND

Input	Output
1000 - 00	1004 - 10
1001 - 11	1005 - 00
1002 - 01	
1003 - 10	

NAND

Input	Output
1000 - 01	1004 - FF
1001 - 10	1005 - EF
1002 - 00	
1003 - 11	

NOR

Input	Output
1000 - 01	1004 - EE
1001 - 10	1005 - EE
1002 - 11	
1003 - 01	

X-OR

Input	Output
1000 - 11	1004 - EF
1001 - 00	1005 - FF
1002 - 01	
1003 - 10	

NOT

Input	Output
1000 - 11	1004 - EF
1001 - 10	1005 - EF

Result: Thus the logical & arithmetic operations on 16 bit data has been performed.

Ex.NO: 2. Move a data block without overlap.

Aim: To write an 8086 ALP to move a block of data from source to destination without overlap.

Input:

1200 : 05
1201 : 03
1202 : 02
1203 : 01
1204 : 00

Output:

1300 : 05
1301 : 03
1302 : 02
1303 : 01
1304 : 00

Result: Thus the program for moving a block of data without overlap was written and executed.

Ex.NO: 3 Code conversion, Decimal Arithmetic & Matrix operations.

Aim: To write an Assembly Language to perform the following operations.

(a) Code conversion : BCD to Binary & Binary to BCD.
(b) Decimal Arithmetic : BCD addition & BCD subtraction.
(c) Matrix operations: Matrix Addition & Matrix multiplication.

A.R. Sharmila
312418205083

(A) Code conversion - BCD to Binary:

Input

1200: 85 (BCD data)

Output:

1201: 55

Binary to BCD

Input: 1200 : 55 [Binary data]

Output: 1201 : 85

decimal arithmetic - BCD addition

Input:

1200: 01 [1st data - BCD]
1201: 04
1202: 08 [2nd data - BCD]
1203: 02

Output:

1204: 09
1205: 06

BCD subtraction

Input:

1200: 18 [1st BCD]
1201: 04
1202: 09 [2nd BCD]
1203: 02

Output:

1204: 09
1205: 02.

Matrix Addition:

Input:

Matrix A:

2000: 00
2001: 01
2002: 02
2003: 03
2004: 04
2005: 05
2006: 06
2007: 07
2008: 08

Matrix B.

3000 : 09
3001 : 08
3002 : 07
3003 : 06
3004 : 05
3005 : 04
3006 : 03
3007 : 02
3008 : 01

Output:

3000 : 09
3001 : 09
3002 : 09
3003 : 09
3004 : 09
3005 : 09
3006 : 09
3007 : 09
3008 : 09.

Matrix multiplication.

Input:

Matrix A

1200 : 02
1201 : 02
1202 : 02
1203 : 02
1204 : 02
1205 : 02
1206 : 02
1207 : 02
1208 : 02

Matrix B

1300 : 02
1301 : 02
1302 : 02
1303 : 02
1304 : 02
1305 : 02
1306 : 02
1307 : 02
1308 : 02

Output:

1400 : 0C
1401 : 00
1402 : 0C
1403 : 00
1404 : 0C
1405 : 00
1406 : 0C
1407 : 00
1408 : 0C

Result: Thus code conversion, decimal arithmetic & matrix operation is successfully executed.

A.R. Sharmila
312418205083

Ex.NO: 4 String manipulation, sorting and searching.

Aim: To write an 8086 ALP to perform the following functions:

a) String Manipulation

copying a string
comparison of two strings
Scan a character in string.

Copying a string:

Input:

1200 : AA	
1201 : AB	
1202 : AC	
1203 : DA	

Output

1200 : AA	
1201 : AB	
1202 : AC	
1203 : DA	

Comparison of two strings

Input

1200 : 02	1300 : 02
1201 : 03	1301 : 03
1202 : 04	1302 : 04
1203 : 05	1303 : 05

Output:

3000: FFH

Scan a character:

Input:

1200 : AD
1201 : AB
1202 : AA
1203 : AD

Output:

3000 : FF

Result: Thus the string manipulation has been performed.

(b) Sorting : Ascending Order
Descending order.

Aim: To perform sorting operations using 8086.

Ascending:

Input:

1200: 04
1201: 39
1202: 40
1203: 30
1204: 78
1205: 462
1206: 42
1207: 32
1208: 38

Output:

1200: 04
1201: 30
1202: 4032
1203: 38
1204: 39
1205: 40
1206: 42
1207: 62
1208: 78

Descending:

Input:

1200: 04
1201: 39
1202: 40
1203: 30
1204: 78
1205: 962
1206: 42
1207: 32
1208: 38

Output:

1200: 04
1201: 78
1202: 62
1203: 42
1204: 40
1205: 39
1206: 38
1207: 32
1208: 30.

Result: Thus the sorting operation using 8086 has been performed successfully.

(c) Searching : Even & Odd numbers

Aim: To perform searching operations using 8086.

Input:

1200: 05
1201: 00
1202: 01
1203: 2
1204: 04
1205: 06

Output:

1300 = 01 odd
1301: 04 even.

Result: Thus the searching operation using 8086 has been performed successfully.

A.R. Sharmila
312418205083

Interfacing Experiments

Ex.No:1 Stepper Motor Control

Aim: To write an assembly language program in 8086 to rotate the motor at different speeds.

Program:

```
MOV CX,8000
MOV AL,80
MOV DX,8006
OUT DX, AL
MOV AL,00
MOV DX,8000
OUT DX, AL
AGAIN: MOV AL,06H/03H
OUT DX,AL
INT AAH
MOV AL,0CH
OUT DX,AL
INT AAH
MOV AL,09H/0CH
OUT DX,AL
INT AAH
MOV AL,03H
OUT DX,AL
INT AAH
JMP AGAIN
INT A5
```

Clockwise direction

06 0C 09 03

Anticlockwise direction

03 09 0C 06

Result: Thus the stepper motor was interfaced with 8086 and run in forward & reverse direction.

Ex.NO:8 Interfacing programmable keyboard and display controller 8279.

Aim: To display rolling message "ST. JOSEPH'S" in the display (or) to accept a key & display it.

Observation

Memory Location	7-segment LED format								Hex Data	Display
	d	c	b	a	dp	g	F	e		
1200 H	0	0	1	0	1	0	0	1	29	S
1201H	1	0	0	0	1	1	1	1	8F	T
1202H	1	1	1	1	1	1	1	1	FF	
1203H	0	0	0	1	1	1	1	1	1F	J
1204H	0	0	0	0	1	1	0	0	0C	O
1205H	0	0	1	0	1	0	0	1	29	S
1206H	0	1	1	0	1	0	0	0	68	E
1207H	1	1	0	0	1	0	0	0	C8	P
1208H	1	0	0	1	1	0	0	0	98	H
1209H	0	0	1	0	1	0	0	1	29	S

Result: Thus the rolling message for ~~"ST JOSEPHS"~~ "ST JOSEPHS" has been successfully executed.

A.R. Sharmila
312418205083.

Ex.No: 3 Serial Interface and parallel interface

Aim: To perform serial & parallel Interfacing.

Observation:

Output: 1250
↳ Serial Interface

Input:

SPDT switch position : 10110011

Output:

LED status: 10110011
↳ Parallel Interface

Result: Thus the program for serial & parallel interface are executed successfully.

Ex. No: 4 A/D and D/A Interface

Aim: to perform ADC & DAC Interfacing.

Observations:

Analog to digital:

Analog Voltage	Digital Data on LED display	Hex Code in Memory Location 1100
-5	00	8C
0	80	8C
3.75	E0	8C

Digital to Analog:

Waveform	Amplitude	Time Period (ms)
Square	2	56
Saw-tooth	2	3
Triangular	2	2.4

Result: thus the program to demonstrate DAC and ADC is successfully executed and verified.

EMULATOR PROGRAMS

1) 16 bit addition

edit: C:\emu8086\MySource\5083.16bitadd.asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

MOV AX, 0F101H ;Load the value to 0F101H
MOV BX, 0A202H ;Load the value to 0A202H
ADD AX,BX ; Addition of above two numbers generates carry and carry flag is set

ret
```

emulator: 5083.16bitadd.com

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	93	03
BX	A2	02
CX	00	09
DX	00	00
CS	F400	
IP	0154	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

F400:0154 F400:0154

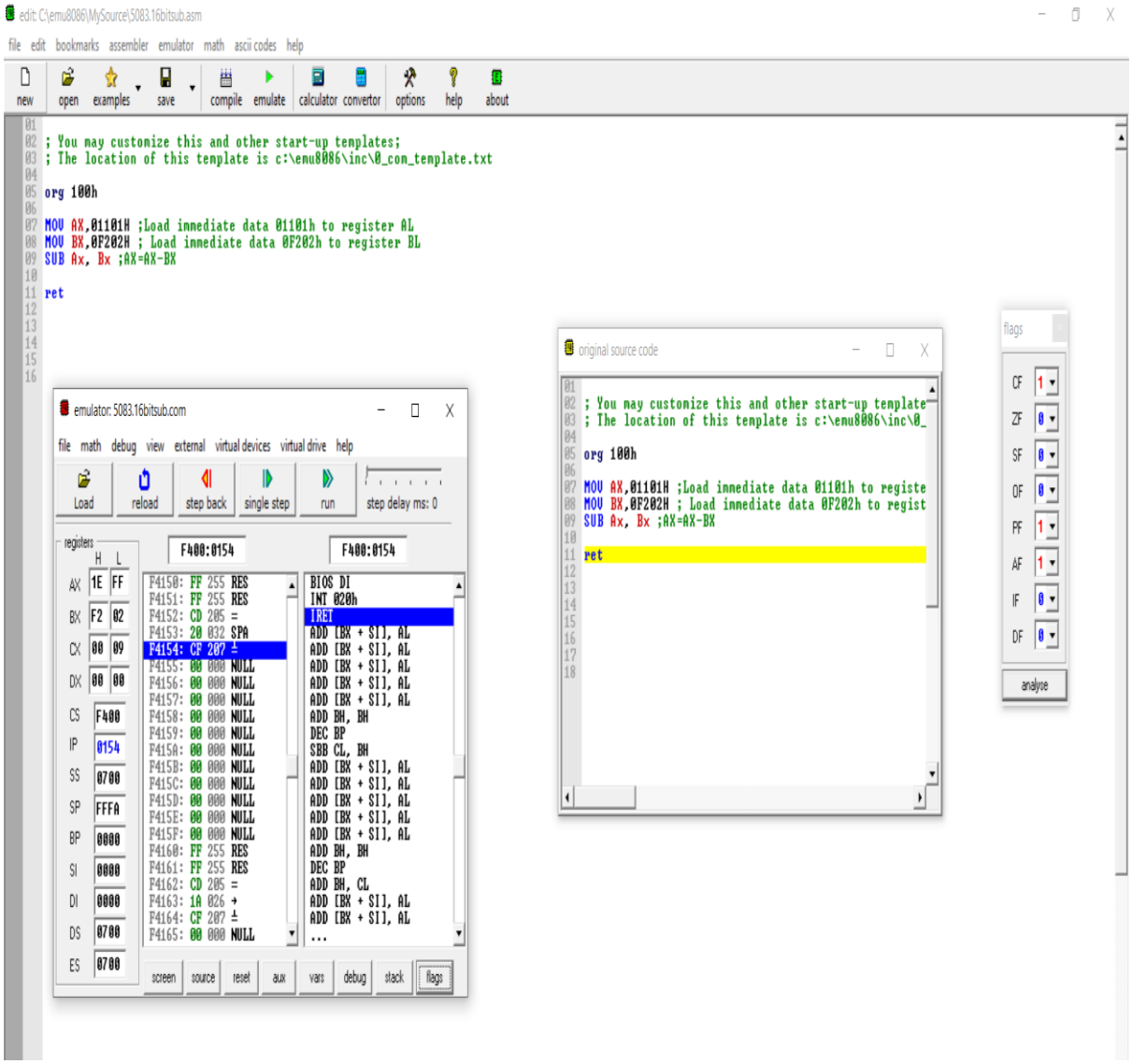
```
F4150: FF 255 RES      BIOS DI
F4151: FF 255 RES      INT 020h
F4152: CD 205 =        IRET
F4153: 20 032 SPA      ADD [BX + SI], AL
F4154: CF 207 ±        ADD [BX + SI], AL
F4155: 00 000 NULL     ADD [BX + SI], AL
F4156: 00 000 NULL     ADD [BX + SI], AL
F4157: 00 000 NULL     ADD [BX + SI], AL
F4158: 00 000 NULL     ADD BH, BH
F4159: 00 000 NULL     DEC BP
F415A: 00 000 NULL     SBB CL, BH
F415B: 00 000 NULL     ADD [BX + SI], AL
F415C: 00 000 NULL     ADD [BX + SI], AL
F415D: 00 000 NULL     ADD [BX + SI], AL
F415E: 00 000 NULL     ADD [BX + SI], AL
F415F: 00 000 NULL     ADD [BX + SI], AL
F4160: FF 255 RES      ADD BH, BH
F4161: FF 255 RES      DEC BP
F4162: CD 205 =        ADD BH, CL
F4163: 1A 026 →        ADD [BX + SI], AL
F4164: CF 207 ±        ADD [BX + SI], AL
F4165: 00 000 NULL     ...
```

screen source reset aux vars debug stack flags

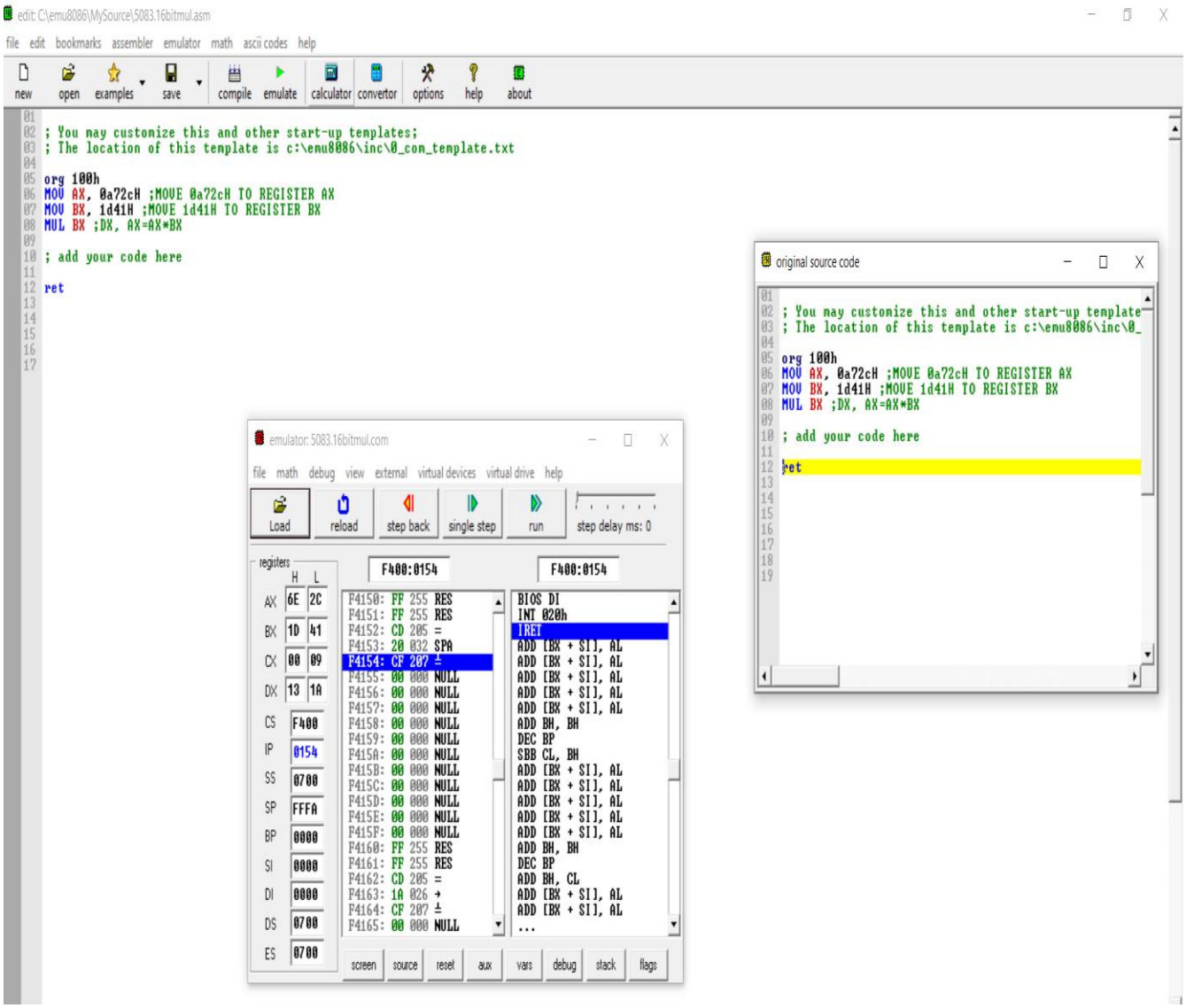
original source code

```
; You may customize this and other start-up template
; The location of this template is c:\emu8086\inc\0_

org 100h

MOV AX, 0F101H ;Load the value to 0F101H
MOV BX, 0A202H ;Load the value to 0A202H
ADD AX,BX ; Addition of above two numbers generates

ret
```

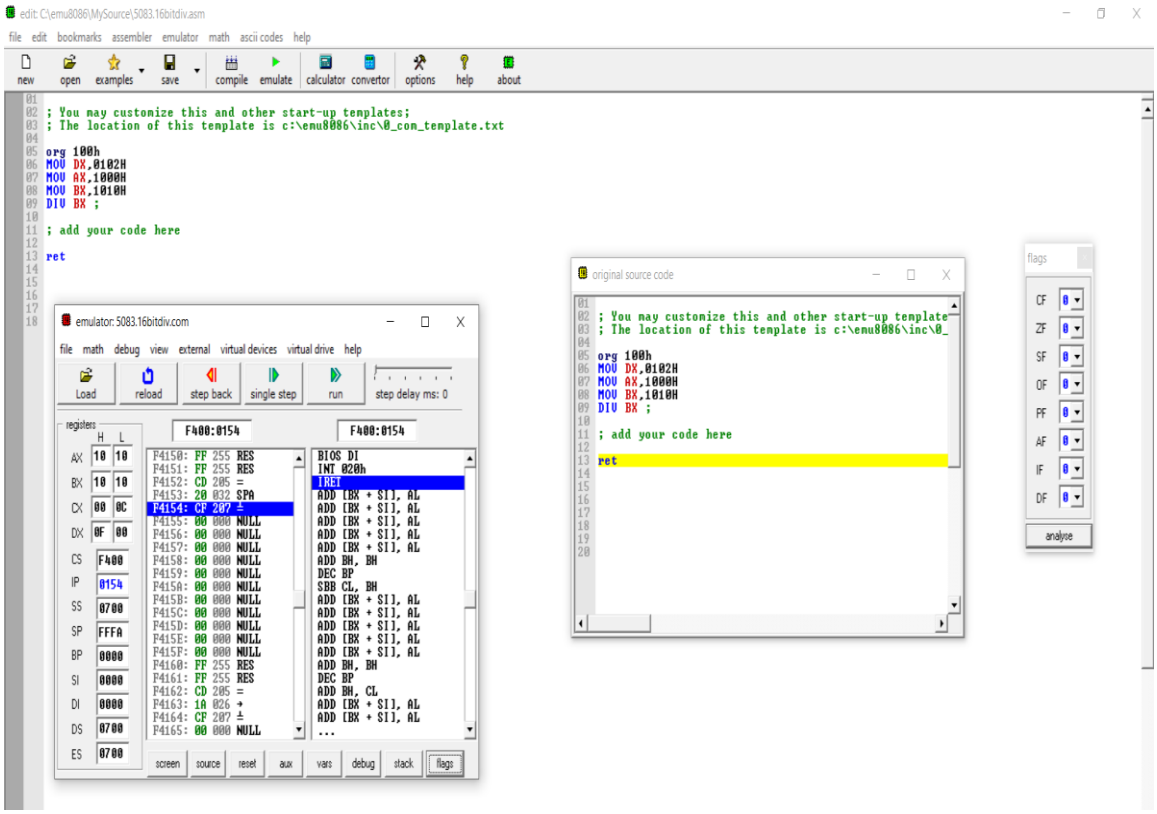
2) **16 bit subtraction**



3) 16 bit multiplication

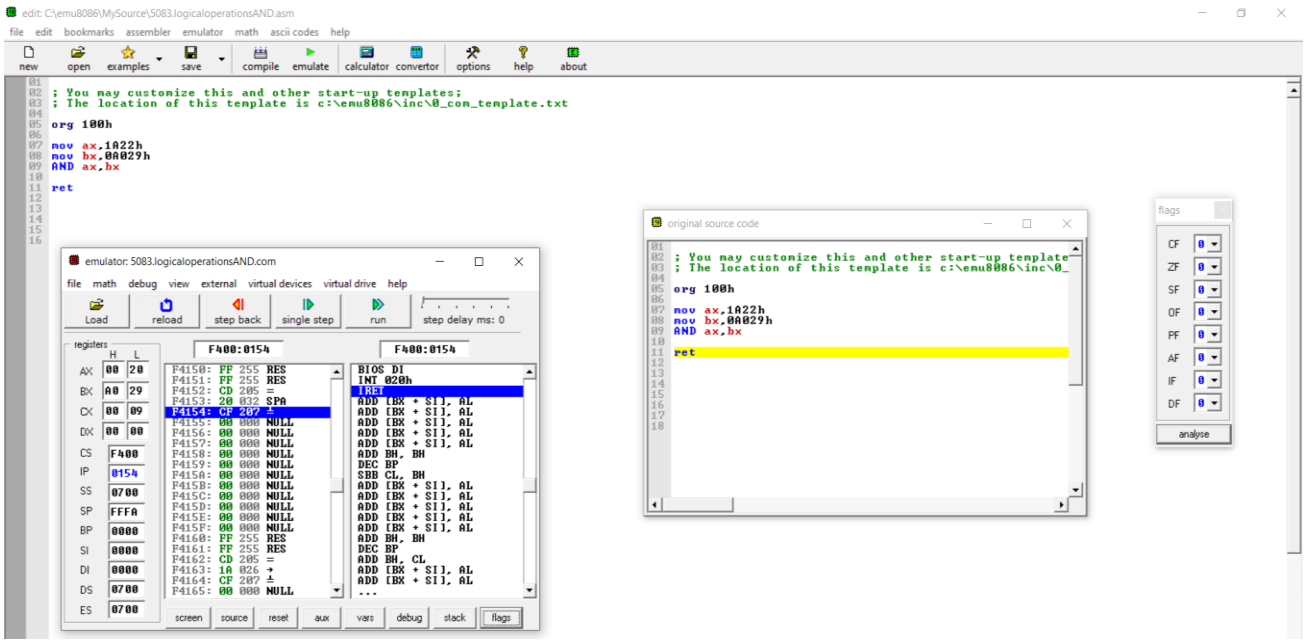


4)16bit division

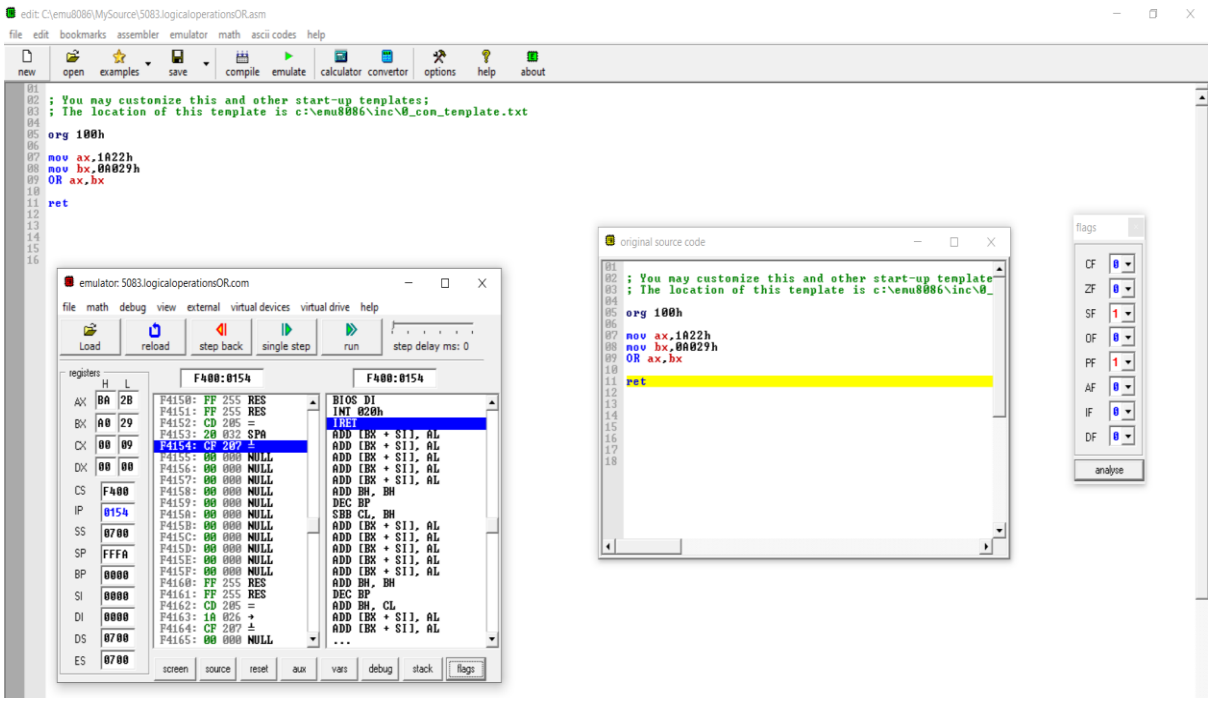


5)Logical Operations

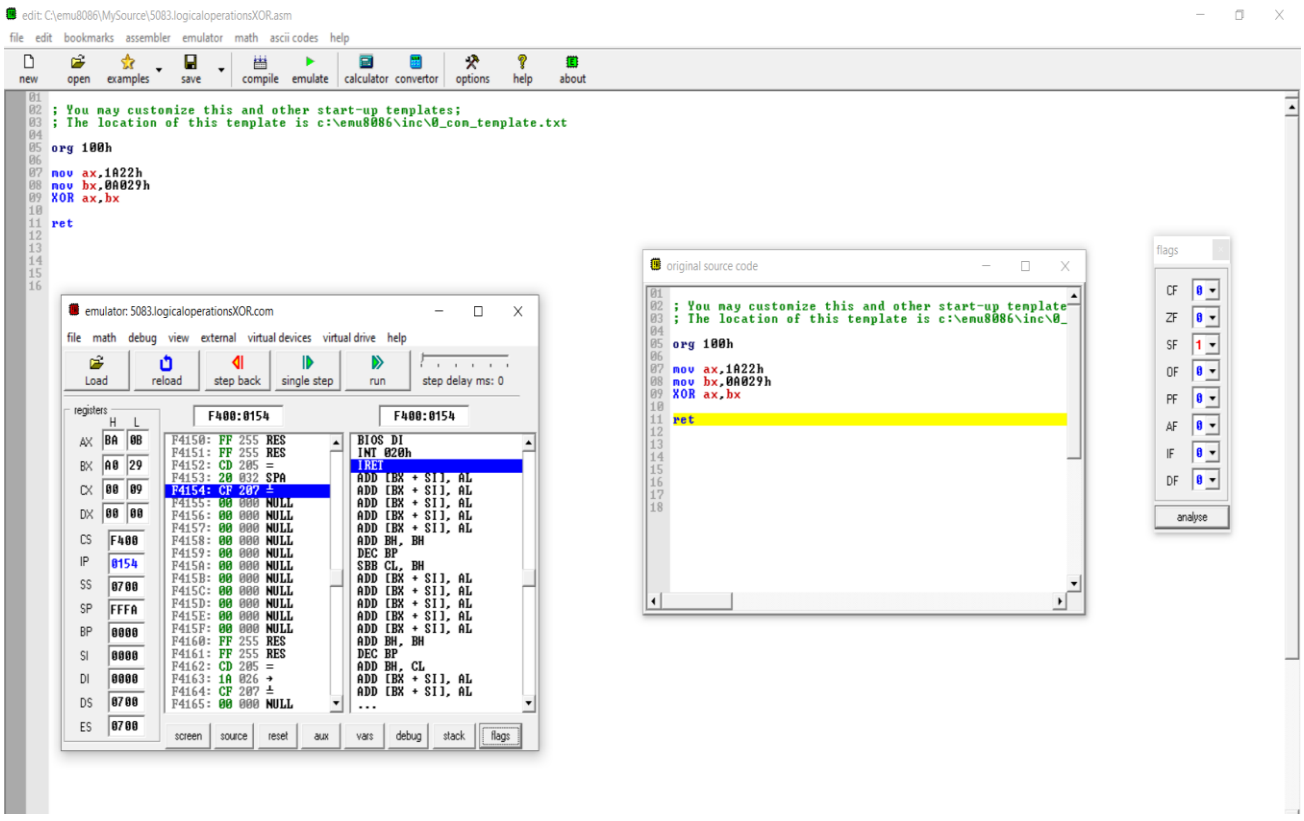
1.AND



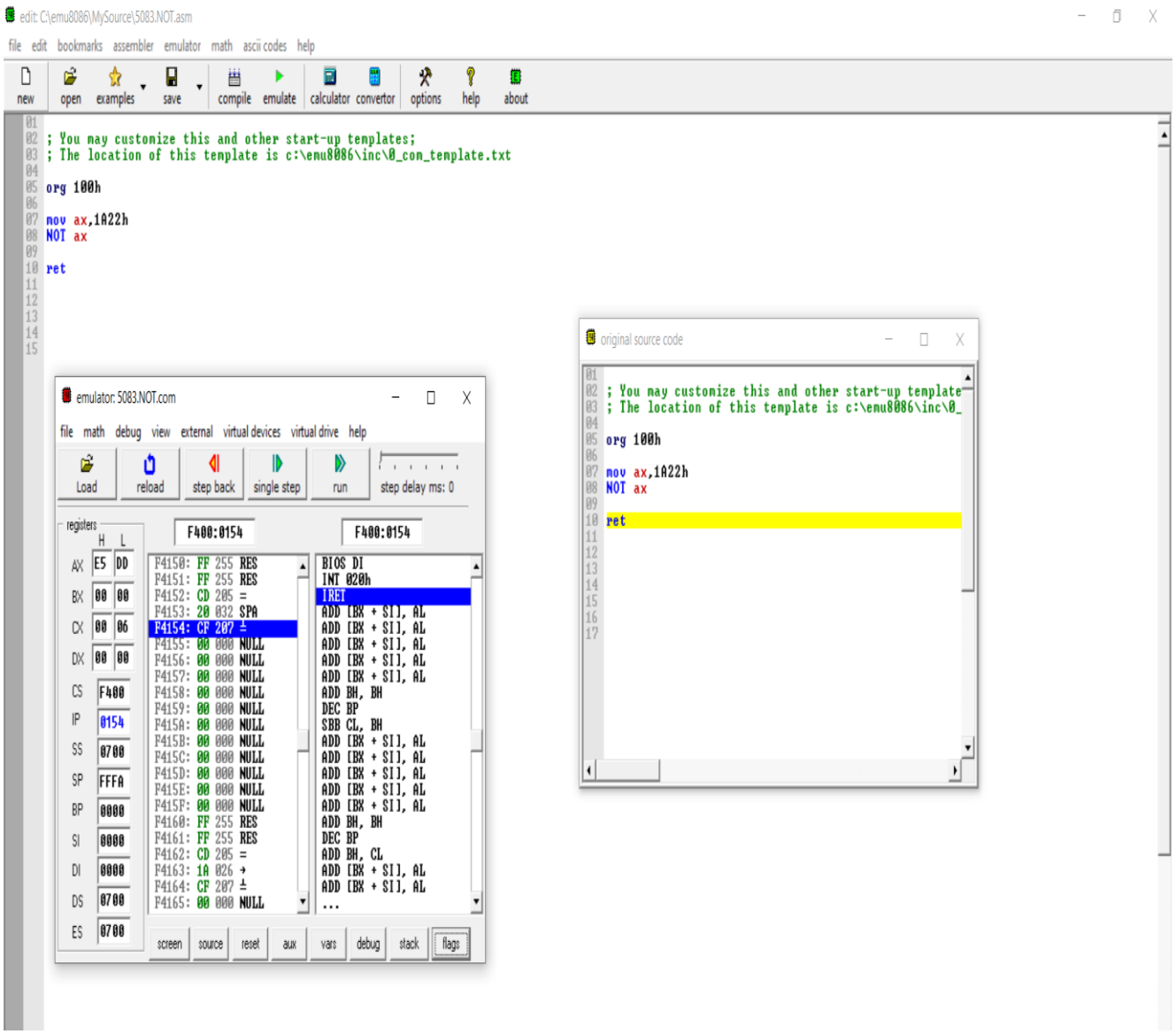
2.OR


edit: C:\emu8086\MySource\5083.logicaloperationsOR.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
mov ax,1A22h
mov bx,0A029h
OR ax,bx
ret
emulator: 5083.logicaloperationsOR.com
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX BA 2B
BX A0 29
CX 00 09
DX 00 00
CS F400
IP 0154
SS 0700
SP FFFA
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
F400:0154
F400:0154
screen source reset aux vars debug stack flags
original source code
flags
CF 0
ZF 0
SF 1
OF 0
PF 1
AF 0
IF 0
DF 0
analyse

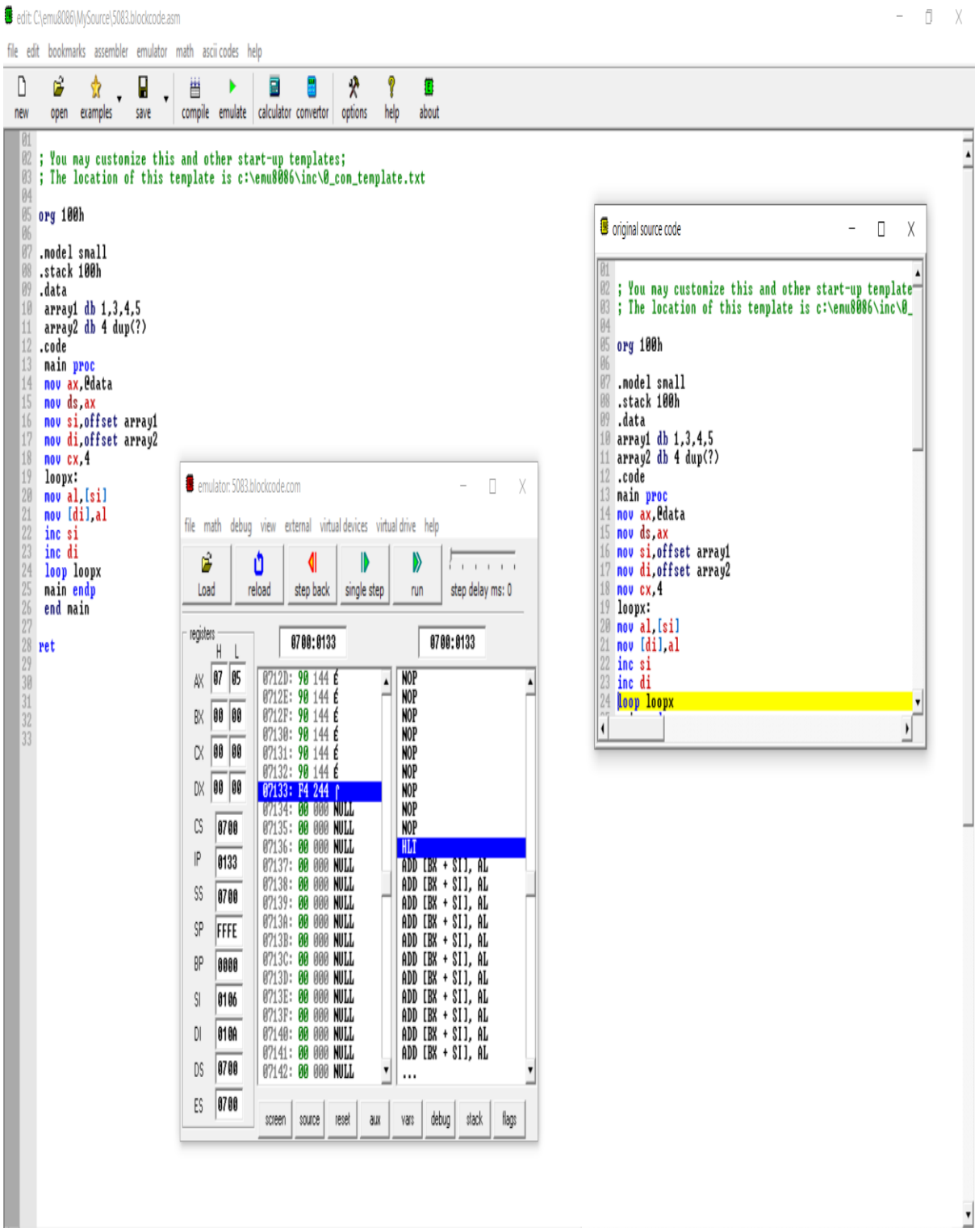
3.XOR


edit: C:\emu8086\MySource\5083.logicaloperationsXOR.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
mov ax,1A22h
mov bx,0A029h
XOR ax,bx
ret
emulator: 5083.logicaloperationsXOR.com
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX BA 0B
BX A0 29
CX 00 09
DX 00 00
CS F400
IP 0154
SS 0700
SP FFFA
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
F400:0154
F400:0154
screen source reset aux vars debug stack flags
original source code
flags
CF 0
ZF 0
SF 1
OF 0
PF 0
AF 0
IF 0
DF 0
analyse

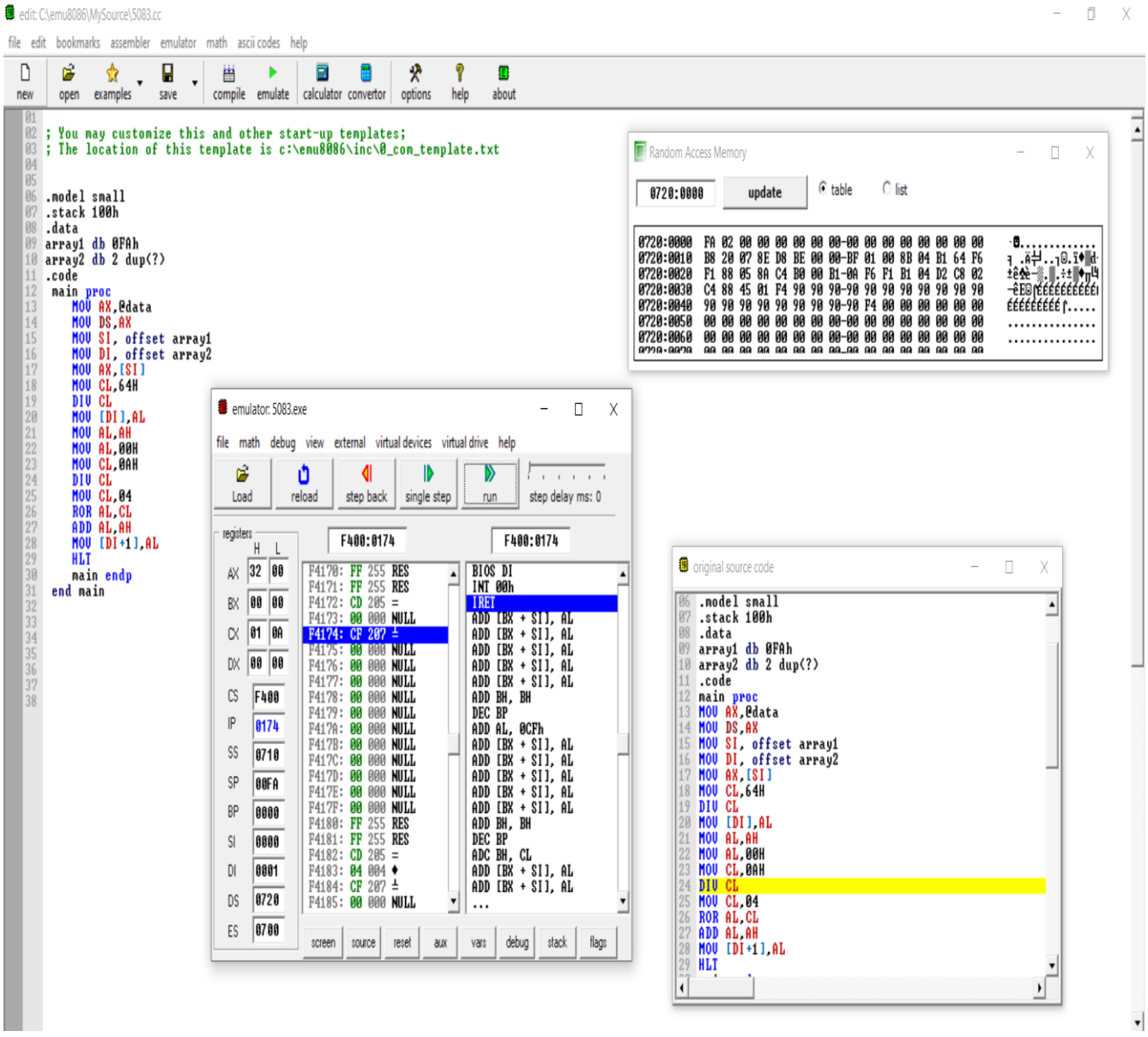
4.NOT



6)Blockcode

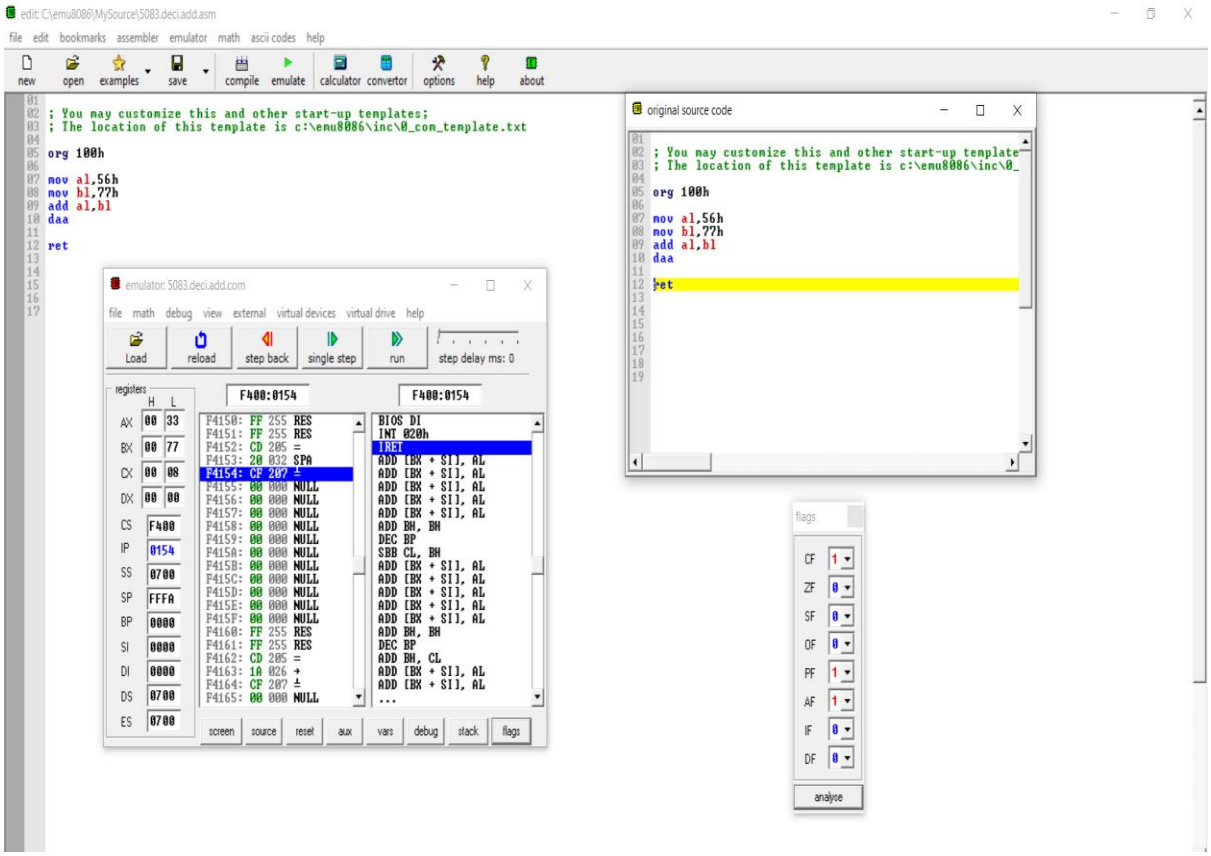

edit: C:\emu8086\MySource\5083.blockcode.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
.model small
.stack 100h
.data
array1 db 1,3,4,5
array2 db 4 dup(?)
.code
main proc
mov ax,@data
mov ds,ax
mov si,offset array1
mov di,offset array2
mov cx,4
loopx:
mov al,[si]
mov [di],al
inc si
inc di
loop loopx
main endp
end main
ret
emulator: 5083.blockcode.com
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
0700:0133
0700:0133
AX 07 05
BX 00 00
CX 00 00
DX 00 00
CS 0700
IP 0133
SS 0700
SP FFFE
BP 0000
SI 0106
DI 010A
DS 0700
ES 0700
07133: F4 244 ⌠
HLT
screen source reset aux vars debug stack flags
original source code
; You may customize this and other start-up template
; The location of this template is c:\emu8086\inc\0_
org 100h
.model small
.stack 100h
.data
array1 db 1,3,4,5
array2 db 4 dup(?)
.code
main proc
mov ax,@data
mov ds,ax
mov si,offset array1
mov di,offset array2
mov cx,4
loopx:
mov al,[si]
mov [di],al
inc si
inc di
loop loopx

7) Code conversion

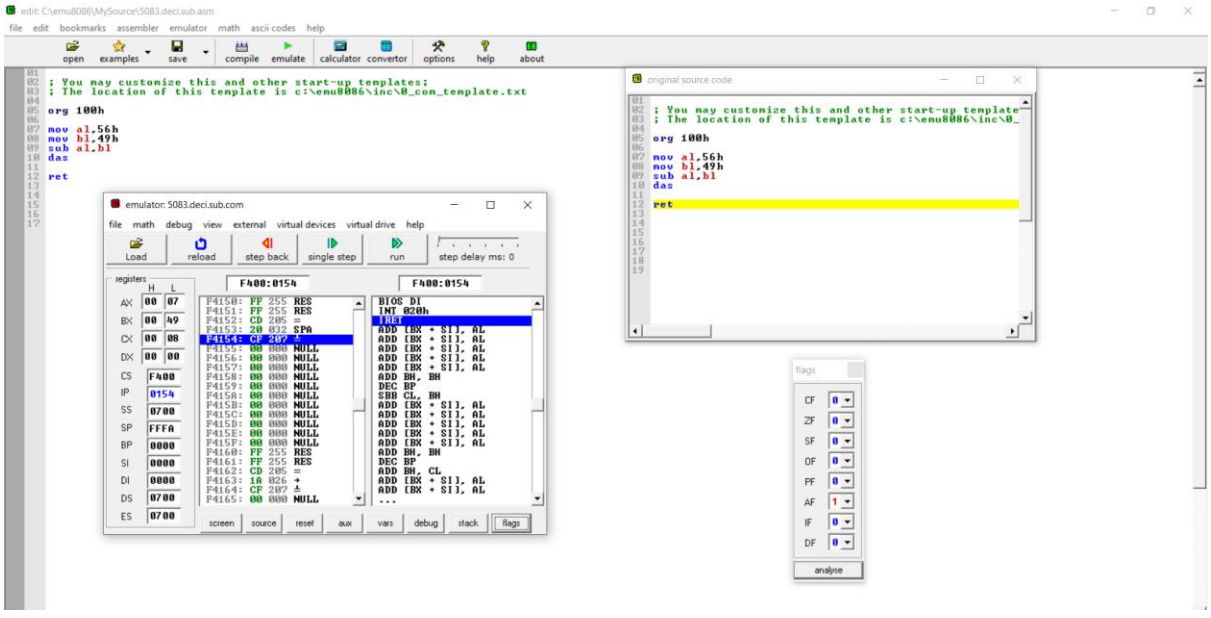


8)Decimal Arithmetic

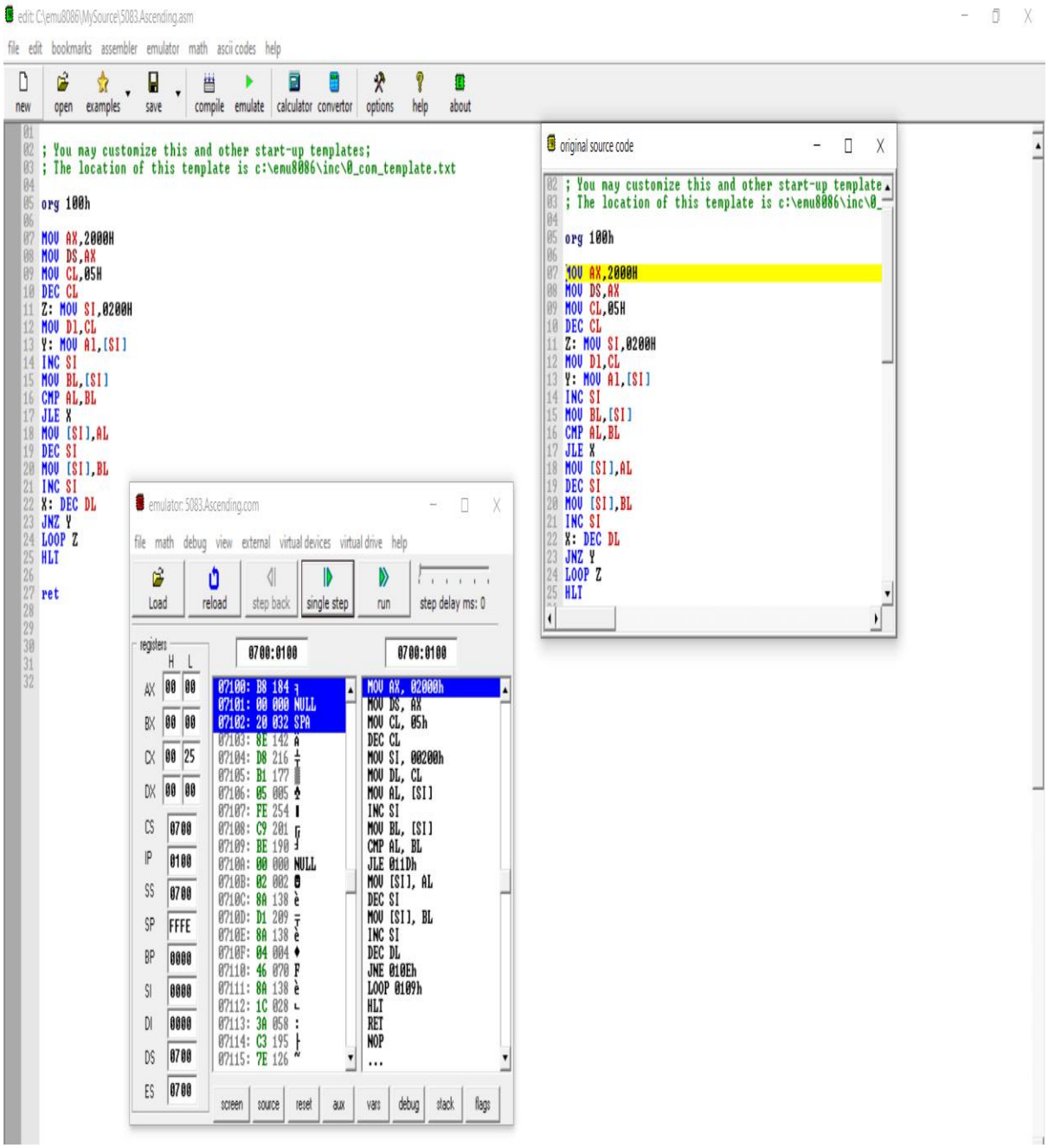
1.Add

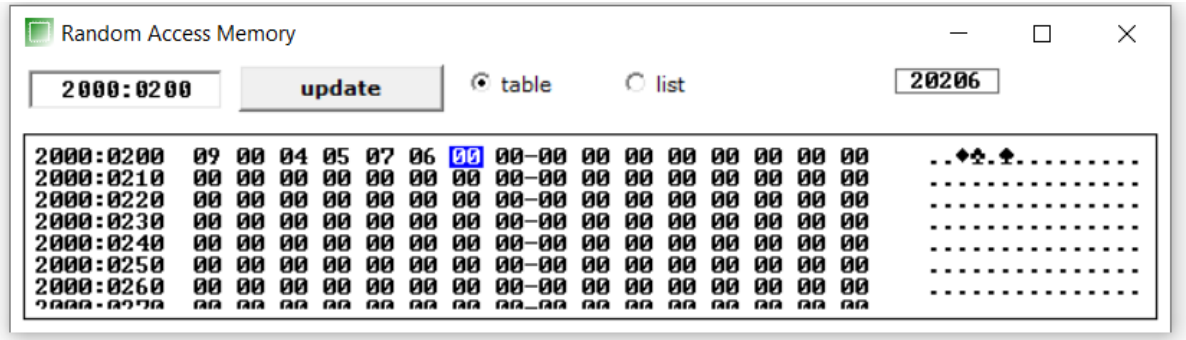


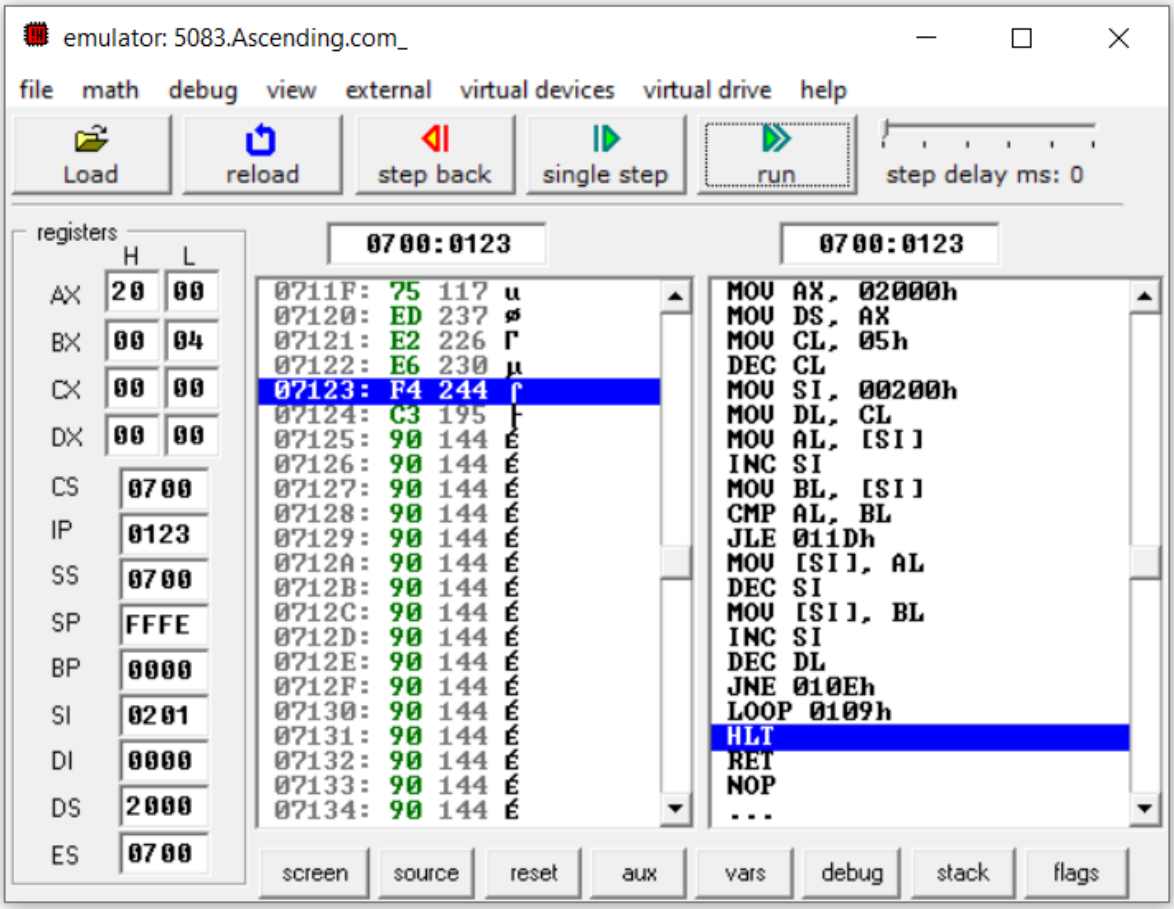
2.Subtract

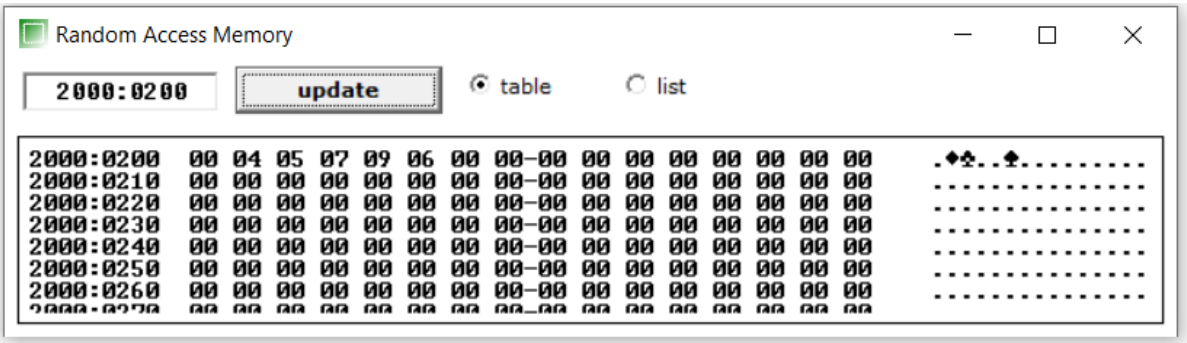


9)Ascending order

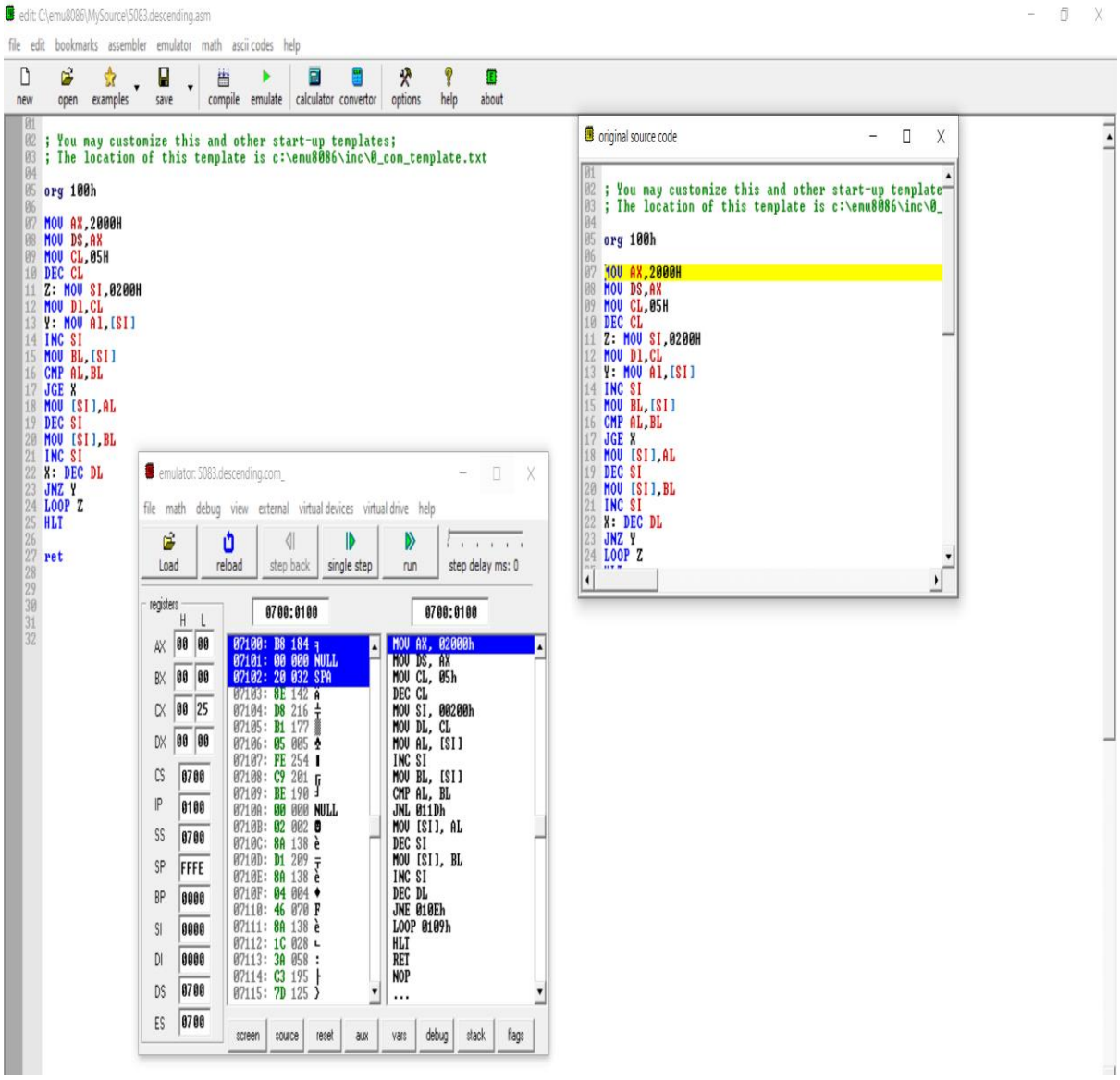


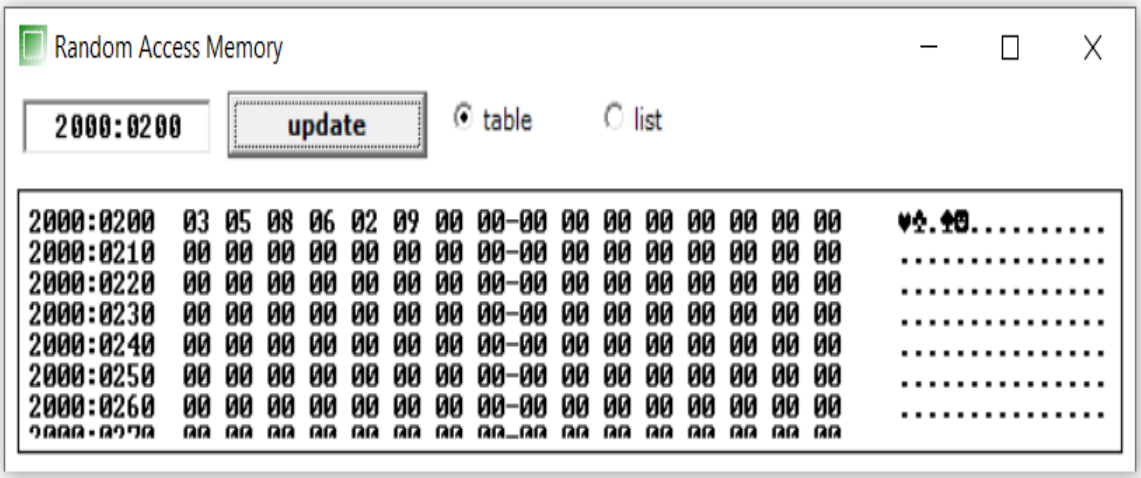

Random Access Memory
2000:0200
update
table
list
20206
2000:0200 09 00 04 05 07 06 00 00-00 00 00 00 00 00 00 00
2000:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

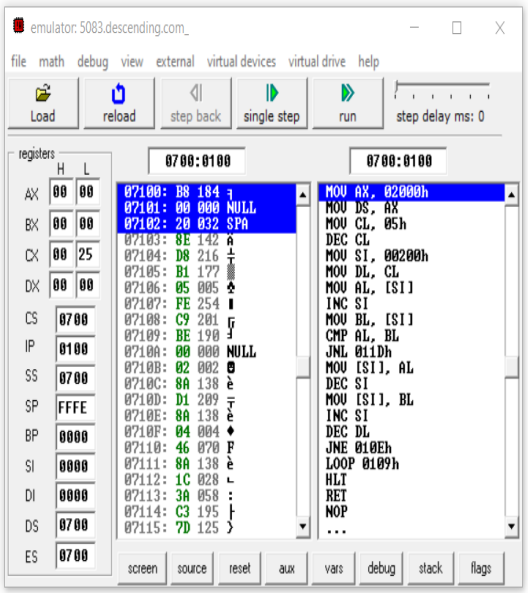

emulator: 5083.Ascending.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 20 00
BX 00 04
CX 00 00
DX 00 00
CS 0700
IP 0123
SS 0700
SP FFFE
BP 0000
SI 0201
DI 0000
DS 2000
ES 0700
0700:0123
0700:0123
07123: F4 244 ⌠
MOV AX, 02000h
MOV DS, AX
MOV CL, 05h
DEC CL
MOV SI, 00200h
MOV DL, CL
MOV AL, [SI]
INC SI
MOV BL, [SI]
CMP AL, BL
JLE 011Dh
MOV [SI], AL
DEC SI
MOV [SI], BL
INC SI
DEC DL
JNE 010Eh
LOOP 0109h
HLT
RET
NOP
screen source reset aux vars debug stack flags

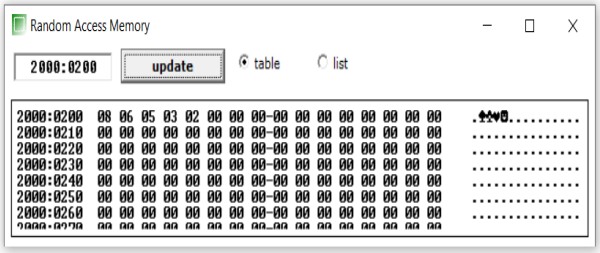

Random Access Memory
2000:0200
update
table
list
2000:0200 00 04 05 07 09 06 00 00-00 00 00 00 00 00 00 00
2000:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

10)Descending order

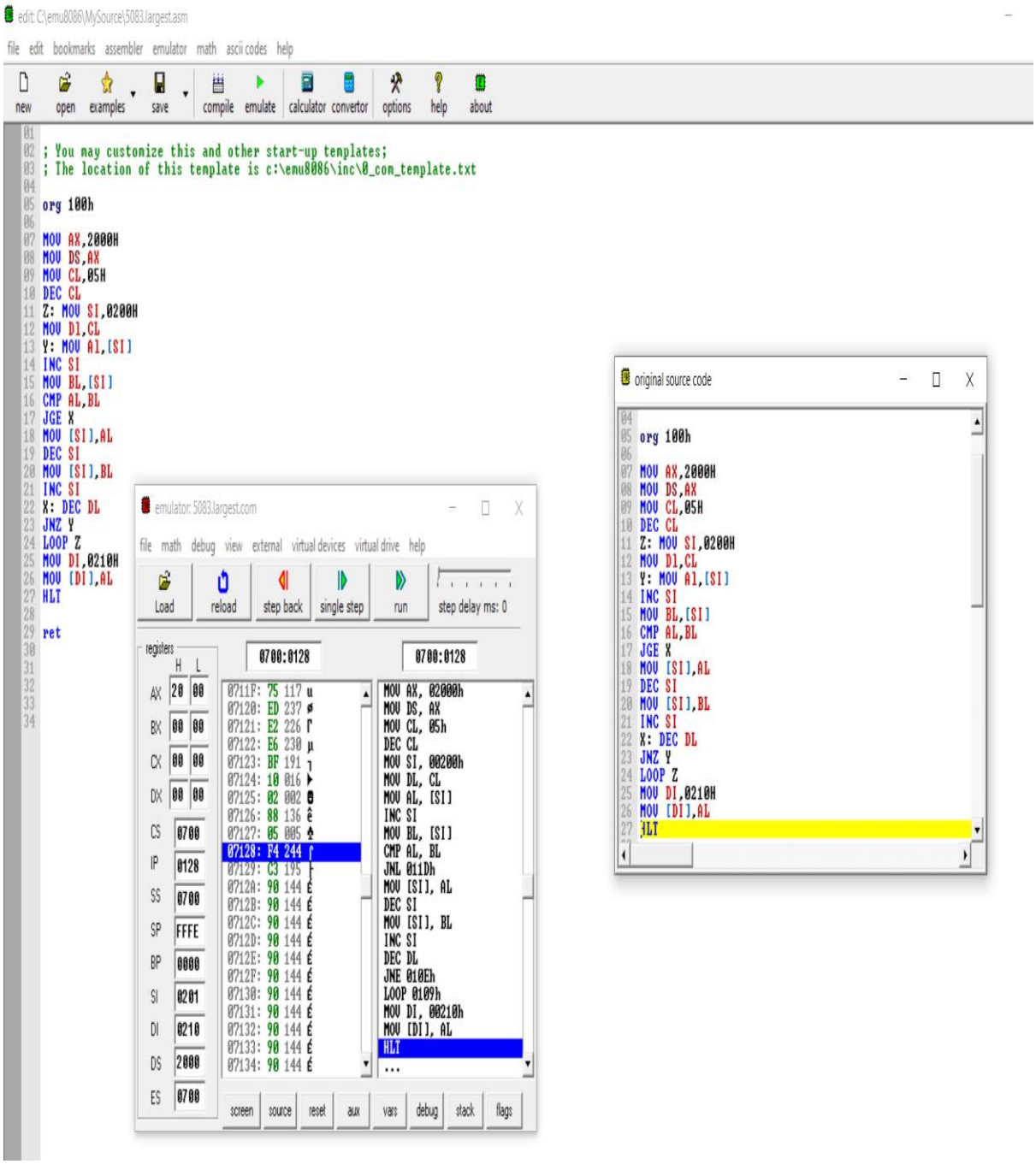


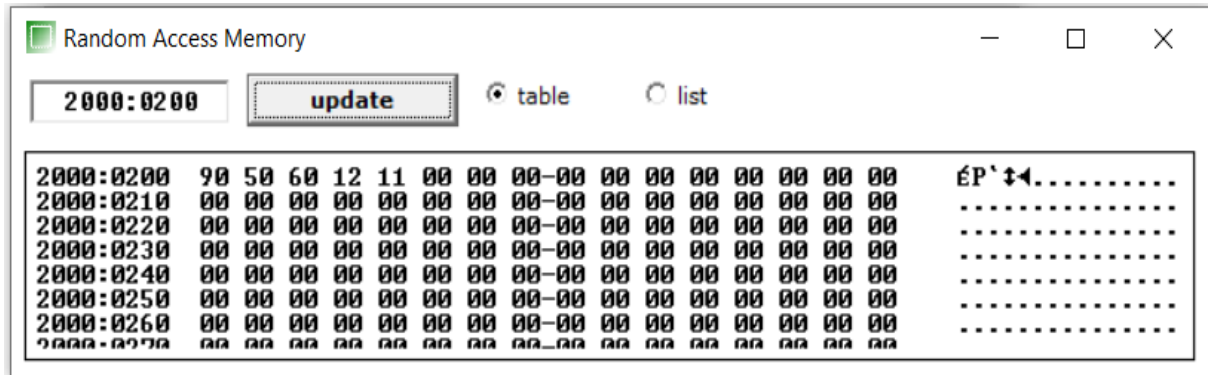

Random Access Memory
2000:0200
update
table
list
2000:0200 03 05 08 06 02 09 00 00-00 00 00 00 00 00 00 00
2000:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

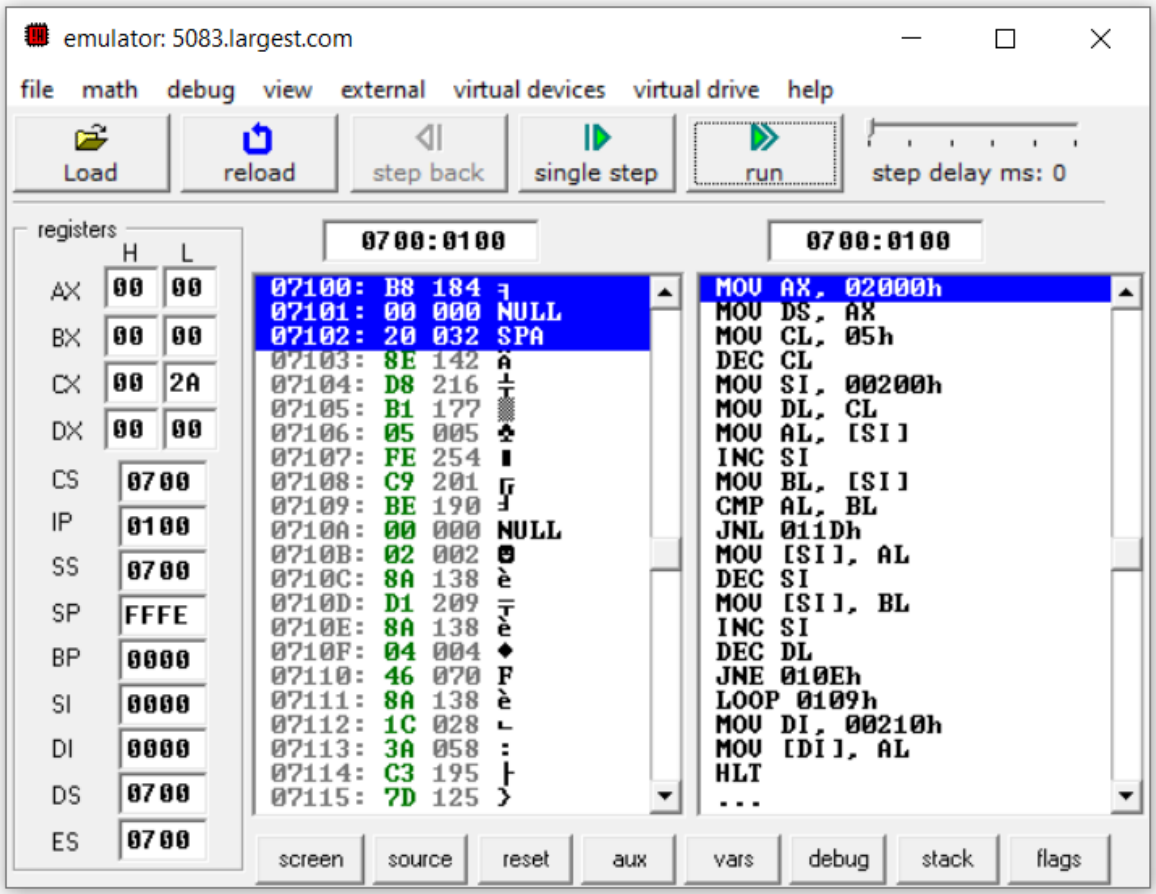

emulator: 5083.descending.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 00 25
DX 00 00
CS 0700
IP 0100
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0100
0700:0100
07100: B8 184
07101: 00 000 NULL
07102: 20 032 SPA
07103: 8E 142
07104: D8 216
07105: B1 177
07106: 05 005
07107: FE 254
07108: C9 201
07109: BE 190
0710A: 00 000 NULL
0710B: 02 002
0710C: 8A 138
0710D: D1 209
0710E: 8A 138
0710F: 04 004
07110: 46 070 F
07111: 8A 138
07112: 1C 028
07113: 3A 058 :
07114: C3 195
07115: 7D 125
MOV AX, 02000h
MOV DS, AX
MOV CL, 05h
DEC CL
MOV SI, 00200h
MOV DL, CL
MOV AL, [SI]
INC SI
MOV BL, [SI]
CMP AL, BL
JNL 011Dh
MOV [SI], AL
DEC SI
MOV [SI], BL
INC SI
DEC DL
JNE 010Eh
LOOP 0109h
HLT
RET
NOP
...
screen
source
reset
aux
vars
debug
stack
flags

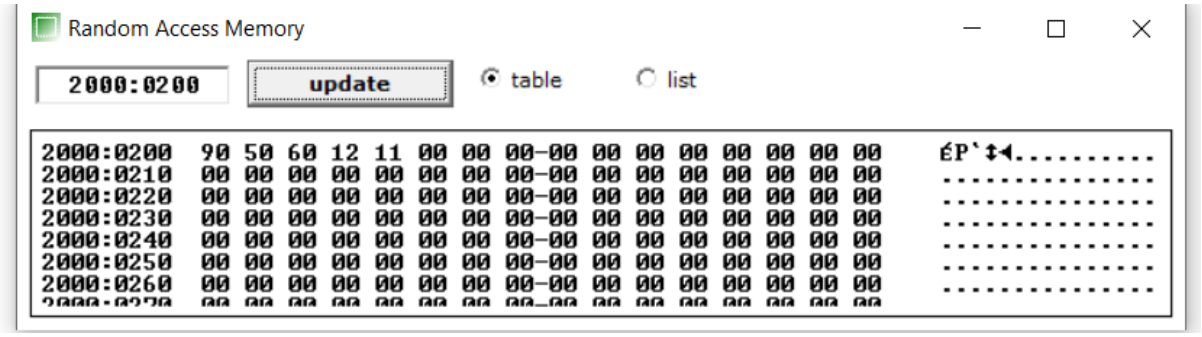

Random Access Memory
2000:0200
update
table
list
2000:0200 08 06 05 03 02 00 00 00-00 00 00 00 00 00 00 00
2000:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

11)Largest

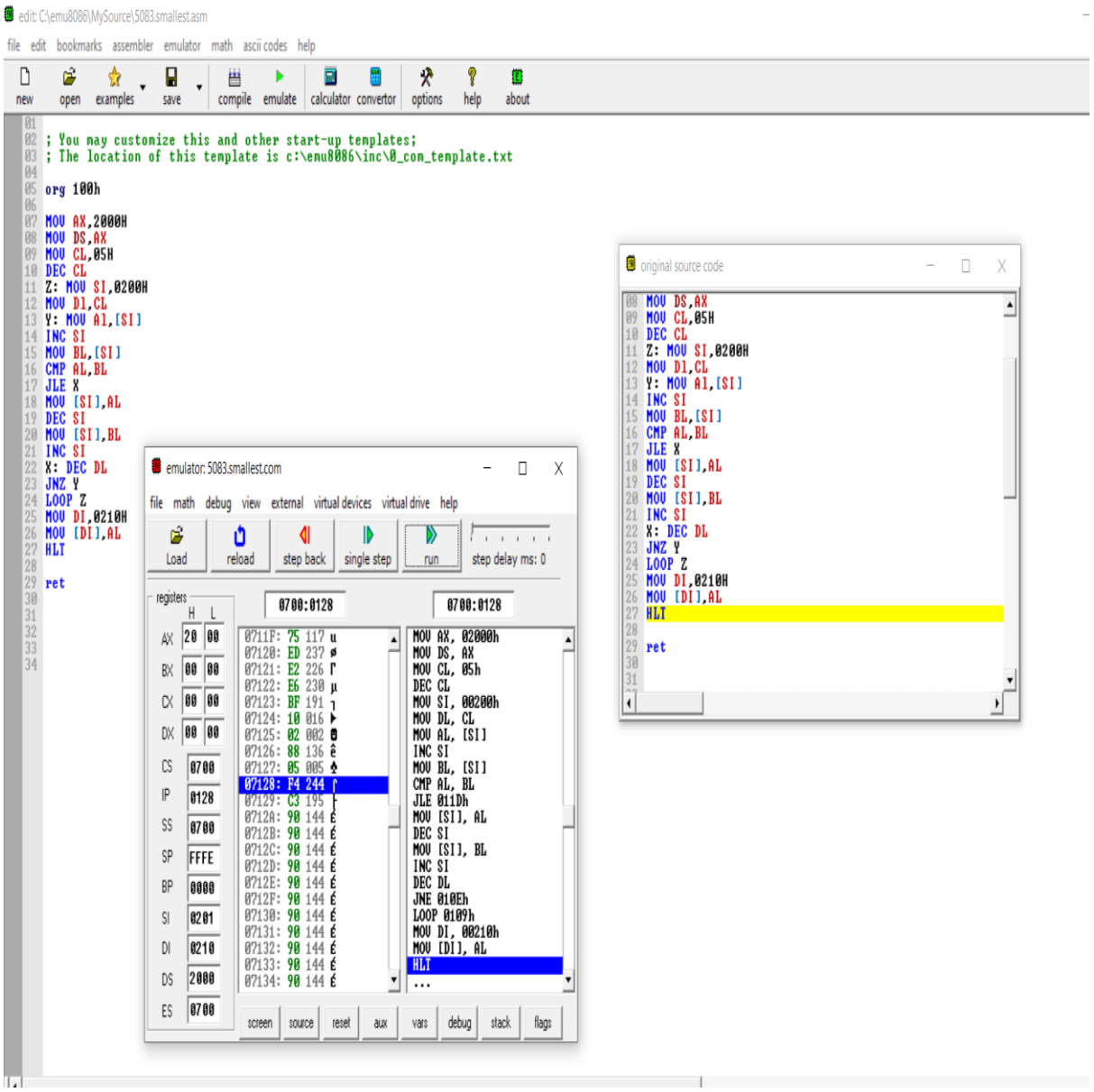

edit: C:\emu8086\MySource\5083.largest.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
MOV AX,2000H
MOV DS,AX
MOV CL,05H
DEC CL
Z: MOV SI,0200H
MOV DI,CL
Y: MOV AL,[SI]
INC SI
MOV BL,[SI]
CMP AL,BL
JGE X
MOV [SI],AL
DEC SI
MOV [SI],BL
INC SI
X: DEC DL
JNZ Y
LOOP Z
MOV DI,0210H
MOV [DI],AL
HLT
ret
emulator: 5083.largest.com
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 20 00
BX 00 00
CX 00 00
DX 00 00
CS 0700
IP 0128
SS 0700
SP FFFE
BP 0000
SI 0201
DI 0210
DS 2000
ES 0700
0700:0128
0700:0128
07128: F4 244
MOV AX, 02000h
MOV DS, AX
MOV CL, 05h
DEC CL
MOV SI, 00200h
MOV DL, CL
MOV AL, [SI]
INC SI
MOV BL, [SI]
CMP AL, BL
JNL 011Dh
MOV [SI], AL
DEC SI
MOV [SI], BL
INC SI
DEC DL
JNE 010Eh
LOOP 0109h
MOV DI, 00210h
MOV [DI], AL
HLT
screen source reset aux vars debug stack flags
original source code
org 100h
MOV AX,2000H
MOV DS,AX
MOV CL,05H
DEC CL
Z: MOV SI,0200H
MOV DI,CL
Y: MOV AL,[SI]
INC SI
MOV BL,[SI]
CMP AL,BL
JGE X
MOV [SI],AL
DEC SI
MOV [SI],BL
INC SI
X: DEC DL
JNZ Y
LOOP Z
MOV DI,0210H
MOV [DI],AL
HLT

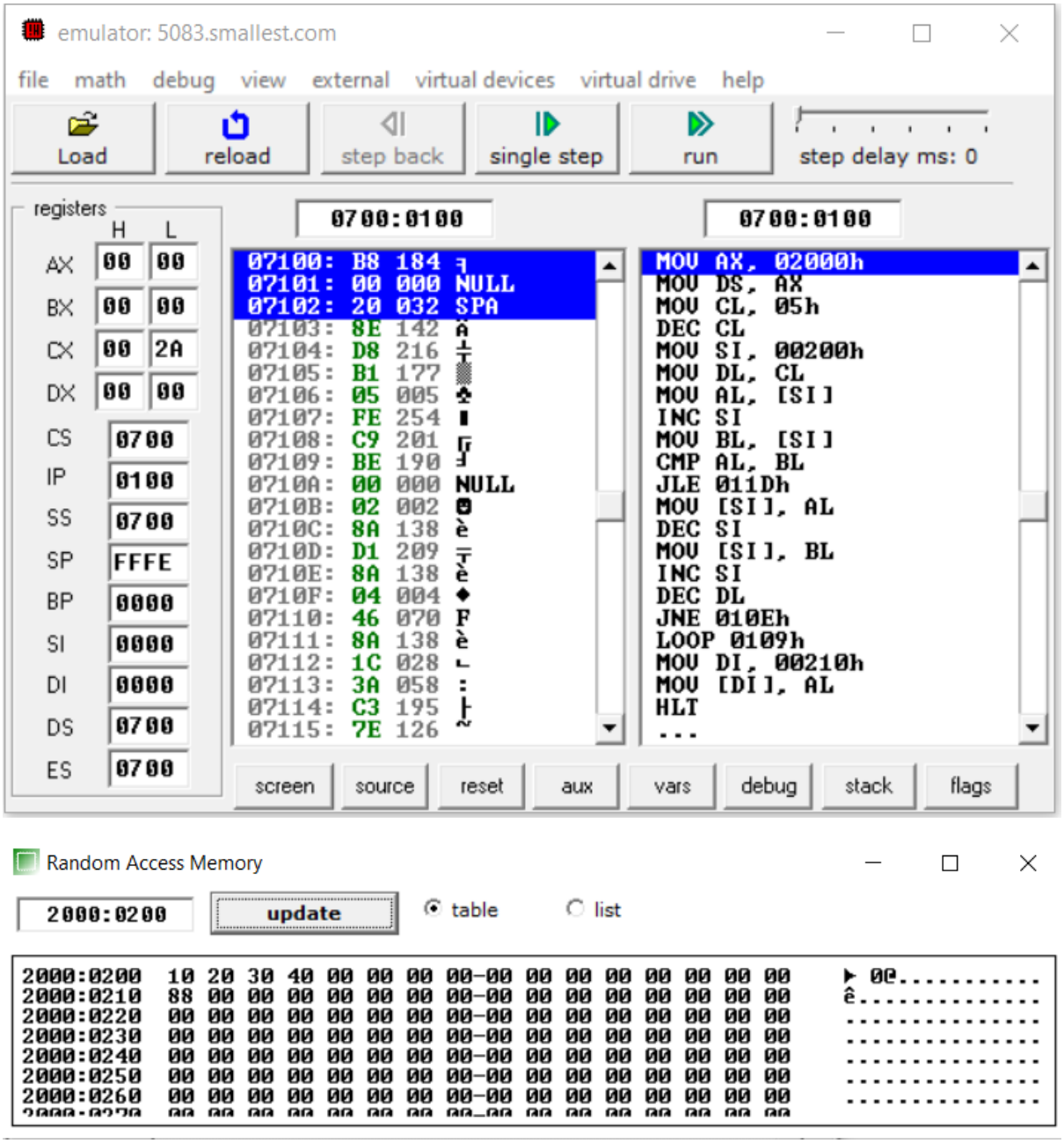

Random Access Memory
2000:0200
update
table
list
2000:0200 90 50 60 12 11 00 00 00-00 00 00 00 00 00 00 00 ÉP`↕◄..........
2000:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....


emulator: 5083.largest.com
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 2A
DX 00 00
CS 0700
IP 0100
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0100
0700:0100
07100: B8 184
07101: 00 000 NULL
07102: 20 032 SPA
07103: 8E 142
07104: D8 216
07105: B1 177
07106: 05 005
07107: FE 254
07108: C9 201
07109: BE 190
0710A: 00 000 NULL
0710B: 02 002
0710C: 8A 138
0710D: D1 209
0710E: 8A 138
0710F: 04 004
07110: 46 070 F
07111: 8A 138
07112: 1C 028
07113: 3A 058 :
07114: C3 195
07115: 7D 125 }
MOV AX, 02000h
MOV DS, AX
MOV CL, 05h
DEC CL
MOV SI, 00200h
MOV DL, CL
MOV AL, [SI]
INC SI
MOV BL, [SI]
CMP AL, BL
JNL 011Dh
MOV [SI], AL
DEC SI
MOV [SI], BL
INC SI
DEC DL
JNE 010Eh
LOOP 0109h
MOV DI, 00210h
MOV [DI], AL
HLT
...
screen
source
reset
aux
vars
debug
stack
flags


Random Access Memory
2000:0200
update
table
list
2000:0200 90 50 60 12 11 00 00 00-00 00 00 00 00 00 00 00 ÉP`↕◄.....
2000:0210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....
2000:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ....

12)Smallest


edit: C:\emu8086\MySource\5083.smallest.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
MOV AX,2000H
MOV DS,AX
MOV CL,05H
DEC CL
Z: MOV SI,0200H
MOV DI,CL
Y: MOV AL,[SI]
INC SI
MOV BL,[SI]
CMP AL,BL
JLE X
MOV [SI],AL
DEC SI
MOV [SI],BL
INC SI
X: DEC DL
JNZ Y
LOOP Z
MOV DI,0210H
MOV [DI],AL
HLT
ret
emulator: 5083.smallest.com
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 20 00
BX 00 00
CX 00 00
DX 00 00
CS 0700
IP 0128
SS 0700
SP FFFE
BP 0000
SI 0201
DI 0210
DS 2000
ES 0700
0700:0128
0700:0128
07128: F4 244
MOV AX, 02000h
MOV DS, AX
MOV CL, 05h
DEC CL
MOV SI, 00200h
MOV DL, CL
MOV AL, [SI]
INC SI
MOV BL, [SI]
CMP AL, BL
JLE 011Dh
MOV [SI], AL
DEC SI
MOV [SI], BL
INC SI
DEC DL
JNE 010Eh
LOOP 0109h
MOV DI, 00210h
MOV [DI], AL
HLT
screen source reset aux vars debug stack flags
original source code


emulator: 5083.smallest.com
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 2A
DX 00 00
CS 0700
IP 0100
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0100
07100: B8 184
07101: 00 000 NULL
07102: 20 032 SPA
07103: 8E 142
07104: D8 216
07105: B1 177
07106: 05 005
07107: FE 254
07108: C9 201
07109: BE 190
0710A: 00 000 NULL
0710B: 02 002
0710C: 8A 138
0710D: D1 209
0710E: 8A 138
0710F: 04 004
07110: 46 070 F
07111: 8A 138
07112: 1C 028
07113: 3A 058 :
07114: C3 195
07115: 7E 126
0700:0100
MOV AX, 02000h
MOV DS, AX
MOV CL, 05h
DEC CL
MOV SI, 00200h
MOV DL, CL
MOV AL, [SI]
INC SI
MOV BL, [SI]
CMP AL, BL
JLE 011Dh
MOV [SI], AL
DEC SI
MOV [SI], BL
INC SI
DEC DL
JNE 010Eh
LOOP 0109h
MOV DI, 00210h
MOV [DI], AL
HLT
...
screen source reset aux vars debug stack flags

Random Access Memory
2000:0200
update
table
list
2000:0200 10 20 30 40 00 00 00 00-00 00 00 00 00 00 00 00
2000:0210 88 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
2000:0260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

EC8681 - Microprocessors and microcontrollers Laboratory

Cycle - II

Reg.No: 312418205083

Roll.No: 18IT1231

Year & section: III. IT-B

Department: IT

Name: A.R.Sharmila

Date: 29.10.20

Ex.No: 1(A) 8 Bit addition / subtraction using 8051

Aim: To perform 8 bit addition / subtraction using 8051 microcontroller.

ADDITION

Input

4500 : 05 (Addeuend)
4501 : 06 (Angend)

Output:

4502 : 0B (Sum)
4503 : 00 (Carry)

Manual Calulation:

0000 0101 ⇒ 05
0000 0110 ⇒ 06

0000 1011 ⇒ 0B

SUBTRACTION

Input:

4500 : 0A (Minued)
4501 : 05 (Subtrahend)

Output:

4502 : 05 (Difference)
4503 : 00 (Sign Bit)

Manual Calculation

0000 1010 ⇒ 0A
0000 0101 ⇒ 05

0000 0101 ⇒ 05

Result:

Thus the 8-bit addition / subtraction using 8051 has been performed successfully.

Ex. No: 1(B) 8-Bit Multiplication / Division Using 8051.

Aim: To perform 8-bit multiplication / division using 8051.

Input for multiplication

4500 : 02

4501 : 03

Output

4502 : 06 (Lower byte of product)

4503 : 00 (Higher byte of product).

Calculations:

0000 02 ⇒ 0000 0010

03 ⇒ 0000 0011

0000 0010

00000 010*

0000000 00**

0000000 0***

000000 0****

00000 00*****

0000 000******

000 0000*******

00 00000********

06 ⇐ 00000000 | 00000110

(Higher Byte) (Lower Byte)

Input for division

4500 : 04

4501 : 02

Output

4502 : 02 (Quotient)

4503 : 00 (Remainder)

Calculations:

0000 0010 ⇒ 02 (Q)

10 | 0000 0100 ⇒ 04 (Divi)

⇓ 02 (Divisor) | 0000 010↓

0000 0000 ⇒ 00 (R)

10 > 0 = 0

10 > 00 = 0

10 > 000 = 0

10 > 0000 = 0

10 > 00000 = 0

10 > 000001 = 0

10 > 0000010 = 1

10 > 00 = 0

Result: Thus the 8 bit multiplication / division using 8051 has been performed successfully.

Ex.NO:2 Logical Operations of 8-bit Using 8051.

Aim: To perform logical operations for clear, set, rotate, swap & logical clear AND using 8051.

CLEAR (AND)

Input

4500 : 00
4501 : 01

Output:

4502 : 00

Calculations:

0000 0000 ⇒ 00
0000 0001 ⇒ 01
0000 0000 ⇒ 00

SET (OR)

Input

4500 : 00
4501 : 01

Output:

4502 : 01

Calculations:

0000 0000 ⇒ 00
0000 0001 ⇒ 01
0000 0001 ⇒ 01

Rotate

Input: 4500 : 0D

Output:

4501 : 86 (Right)
4502 : 1A (Left)

Calculations:

0000 1101
1000 0110
↳ 06

0000 1101
0001 1010
↳ 1A

Swap

Input:

4500 : 0D

Output:

4501 : D0

Calculations:

0000 1101 ⇒ D0
1101 0000

Result: Thus the logical operations for clear, set, rotate and swap has been performed successfully

Ex. No: 3(A) Square of 8-bit Number using 8051

Aim: To perform a squaring on a 8 bit number using 8051.

Input:

4500:03

Output: 4501:09

Calculation:

0000 0011 x 0000 0011

0000 0011 ⇒ 03
0000 0011 ⇒ 03

0000 0011
00000011*
000000**
0000***
.....
.....

00001001 ⇒ 09

Result: Thus the squaring of an 8-bit number using 8051 has been performed successfully.

EX.NO: 3(B) Cube of an 8-bit Number Using 8051.

Aim: To perform an 8-bit cube using 8051.

Input:

4500 : 02

Output:

4501 : 08

Manual calculation:

```
   0000 0010  => 02
   0000 0010  => 02
  ------------
   0000 0000
  0 0000 010*
  - - - - - - ...
  ------------
  0 0000 0100  => 04
    0000 0010  x
  ------------
  0 0000 0000
0 0 0 000 100*
  ------------
... .. 0000 1000  => 08
```

Result: Thus to perform an 8-bit cube using 8051 has been performed successfully.

Ex.NO:04 2's Complement of an 8-bit using 8051.

Aim: to find 2's complement of an 8-bit number using 8051.

Input:

4500 : 25

Output:

4501 : DB

Calculation:

0010 0101 ⇒ 25

1101 1010 → 1's complement

+1

1101 1011 ⇒ DB.

Result: Thus the 2's complement of an 8-bit number using 8051 has been performed successfully.

Ex.NO: 5 Unpacked BCD Number to ASCII Number - 8051

Aim: to convert unpacked BCD number to ASCII number using 8051.

Input:

4500: 02

4501: 09

Output:

4502: 32

4503: 39

Manual Calculation:

```
0000  0010 ⇒ 02
0011  0000 ⇒ 30
-----------------
0011  0010 ⇒ 32
```

```
0000  1001 ⇒ 02
0011  0000 ⇒ 03
-----------------
0011  1001 ⇒ 39
```

Result: Thus the unpacked BCD number to ASCII number using 8051 has been converted successfully

Cycle - I

Manual calculations.

1(A) 16-Bit Addition / Subtraction.

Addition with carry

```
  7C87
  A5BA
------
01 2142
```

Addition without carry.

```
  CA05
  23A1
------
  EDA6
```

Subtraction with Borrow

```
  A163
  A335
------
  FE2D
```

Subtraction without Borrow

```
  BC57
  CB65
------
01 F1E1
```

1(B) 16 Bit Multiplication / Division.

```
   3A58
   6315 x
----------
5E076EED
```

```
2448
8778 ÷

      0000
⇒8778 | 2448
      | 2448
```

INDEX

NAME OF THE STUDENT: **DEPARTMENT:**
ROLL NO. : **REGISTER NO. :**
LAB INCHARGE :

S.NO	DATE	NAME OF THE EXPERIMENT	DATE OF SUBMISSION	PAGE NO	MARK	SIGNATURE OF THE STAFF

SYLLABUS

EC8681 MICROPROCESSOR AND MICROCONTROLLER LAB L T P C 0 0 3 2

8086 Programs using kits and MASM

1. Basic arithmetic and Logical operations
2. Move a data block without overlap
3. Code conversion, decimal arithmetic and Matrix operations.
4. Floating point operations, string manipulations, sorting and searching
5. Password checking, Print RAM size and system date
6. Counters and Time Delay

Peripherals and Interfacing Experiments

7. Traffic light control
8. Stepper motor control
9. Digital clock
10. Key board and Display
11. Printer status
12. Serial interface and Parallel interface
13. A/D and D/A interface and Waveform Generation

8051 Experiments using kits and MASM

14. Basic arithmetic and Logical operations
15. Square and Cube program, Find 2"s complement of a number
16. Unpacked BCD to ASCII

TOTAL - 60 PERIODS

LIST OF EXPERIMENTS:

CYCLE - I

1. (a)16 Bit Addition and Subtraction using 8086.
   (b)16 Bit Multiplication and Division using 8086.
2. String Manipulation using 8086.
3. (a) Sorting in Ascending Order using 8086.
   (b) Searching of Largest number using 8086.
   (c) Decimal arithmetic operations on 8 bit numbers using 8086.
4. Movement of data block using 8086.
5. Code-Conversion using 8086.
6. (a) Password Checking using MASM.
   (b) Finding RAM size using MASM.
   (c) Displaying System Date using MASM.
7. Matrix Operation using 8086.
8. Traffic light controller using 8086.
9. Interfacing Stepper Motor using 8086.

CYCLE - II

10. Interfacing Programmable Peripheral Interface 8255 using 8086.
11. Interfacing Programmable Keyboard and Display Controller 8279 using 8086.
12. (a)Printing a single character using VBMB-005 and 8086.
    (b) Setting and Displaying the time in RTC interface board VBMB-015 using 8086.
13. (a)Interfacing Analog to Digital Converter using 8086.
    (b) Interfacing Digital to Analog Converter using 8086.
    (c) Interfacing Timer-8253 using 8086.
14. Interfacing USART-8251 using 8086.
15. (a) 8 bit addition/subtraction using 8051.
    (b) 8 bit multiplication & 8 bit division using 8051.
16. Logical operation using 8051 microcontroller.
17. (a) Finding square of an 8 bit number using 8051.
    (b) Finding cube of an 8 bit number using 8051.
18. Finding 2's complement of an 8 bit number using 8051.
19. Unpacked BCD number to ASCII number using 8051.
20. Study of ARM Processor.

Ex. No.	16 Bit Addition and Subtraction using 8086	Date

AIM: To add/ subtract two 16 bit numbers residing in memory and to store the result in memory.

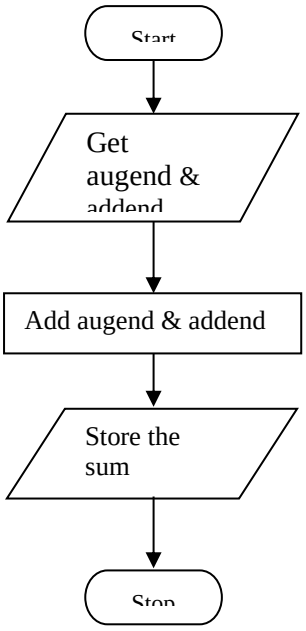
APPARATUS REQUIRED: 8086 microprocessor kit, Power supply.

ALGORITHM:

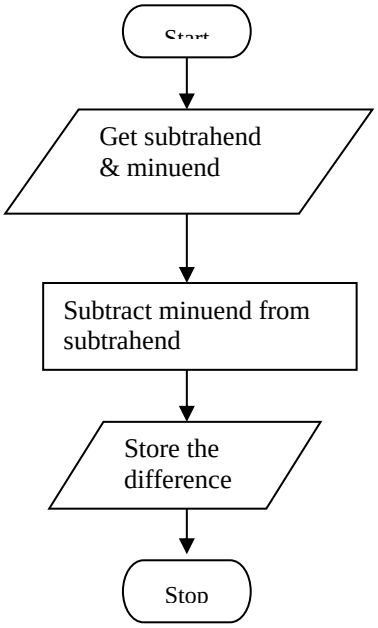
1. Move the content in the memory to the AX register.
2. Increment the memory location.
3. Add the content in the memory to the AX register.
4. Move the result to a memory location.
5. Halt.

FLOWCHART:

ADDITION



SUBTRACTION



ADDITION

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV CX,0000H		Initialize counter CX
		MOV AX,[1300]		Get the first data in Ax reg
		MOV BX,[1302]		Get the second data in BX reg
		ADD AX,BX		Add the contents of both the regs AX,BX
		JNC L1		Check for carry
		INC CX		If carry exists, increment the CX
	L1:	MOV[1500],CX		Store the carry
		MOV [1502],AX		Store the sum
		HLT		Stop the program

SUBTRACTION

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV CX,0000H		Initialize counter CX
		MOV AX,[1300]		Get the first data in Ax reg
		MOV BX,[1302]		Get the second data in BX reg
		SUB AX,BX		Subtract the contents of BX from AX
		JNC L1		Check for borrow
		INC CX		If borrow exists, increment the CX
	L1:	MOV[1500],CX		Store the borrow
		MOV [1502],AX		Store the difference
		HLT		Stop the program

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. Differentiate between 8085 and 8086 processor?
2. Give some examples for 16-bit processors.
3. What are the functional blocks of 8086 processor?
4. What is an assembler and what type of assembler is used in 8086 based systems?
5. Compare the bus status of 8085 & 8086 during instruction execution?

Ex. No.	16 Bit Multiplication and Division using 8086	Date

AIM: To multiply two 16 bit numbers in memory and store the result in memory and to perform division of a 16 bit number by a 16 bit number and to store quotient and remainder in memory.

APPARATUS REQUIRED: 8086 Microprocessor kit, Power supply.

ALGORITHM:

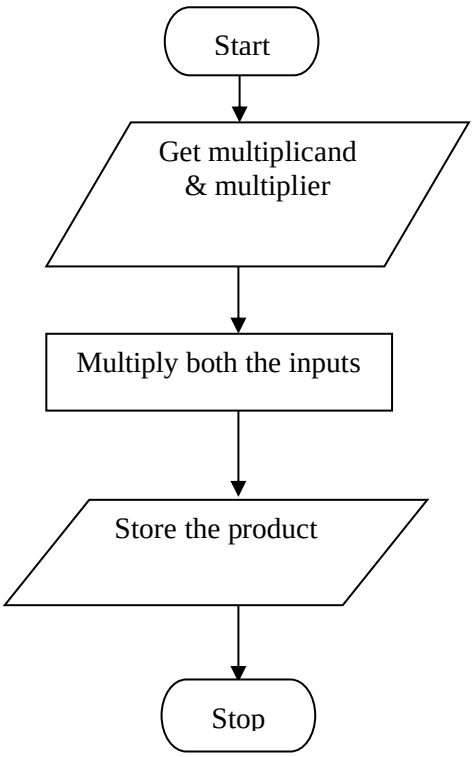
(i) 16 bit Multiplication:

1. Start the program.
2. Get the multiplicand and multiplier.
3. Find the product.
4. Store the result and terminate the program.

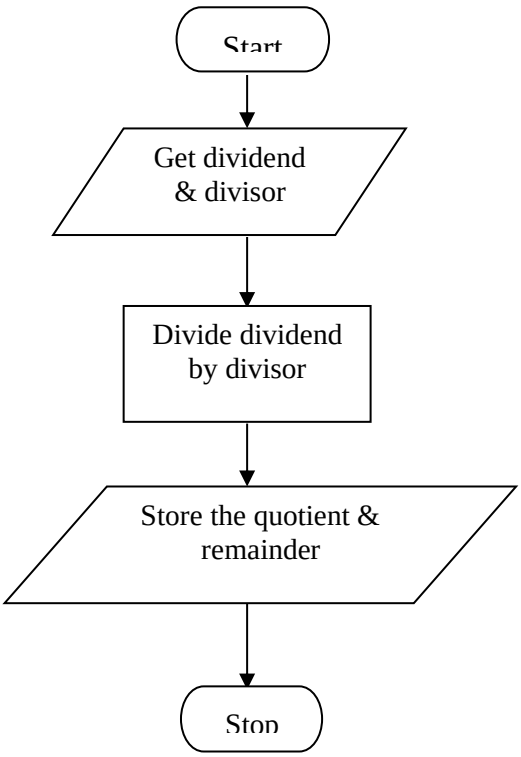
(ii) 16 bit Division:

1. Start the program.
2. Get the Dividend and Devisor.
3. Find the Quotient and Reminder.
4. Store the result and terminate the program.

MULTIPLICATION



DIVISION



MULTIPLICATION

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV AX,[1300]		Get the first data in Ax reg
		MOV BX,[1302]		Get the second data in BX reg
		MUL BX		Multiply both
		MOV[1500],AX		Store the lower order product
		MOV AX, DX		Copy the higher order product to AX
		MOV[1502],AX		Store the higher order product
		HLT		Stop the program

DIVISION

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV AX,[1300]		Get the first data (Lower order byte)
		MOV DX,[1302]		Get the first data (Higher order byte)
		MOV BX,[1304]		Get the second data (Divisor)
		DIV BX		Divide the dividend by divisor
		MOV[1500],AX		Store the quotient
		MOV AX, DX		Copy the remainder to AX
		MOV[1502],AX		Store the remainder
		HLT		Stop the program

OUTPUT:

RESULT:

REVIEWQUESTIONS:

1. What is the register pair generally used while doing 16-bit multiplication?
2. How the multiplication process differs in 8086 compared to 8085?
3. What are the flags affected by the multiplication process?
4. Where is the Quotient stored in 16-bit Division?
5. Where is the Reminder stored in 32-bit Division?

Ex. No.	String Manipulation Using 8086	Date

AIM: To write a 8086 program

a) To copy a string of data words from one location to the other.
b) To search a word from a string.
c) To find and replace a word from a string.

APPARATUS REQUIRED: 8086 Microprocessor kit, power supply.

ALGORITHM:

a) Copying a String

1. Initialize DS, SI, DI, ES.
2. Store the length of the string in CX register.
3. Move the byte from DS to ES till CX =0.

b) Search a character in the string

1.Initialise ES and DI.
2. Store the no of characters in the string in CX.
3. Move the byte to be searched to AL.
4. Store the ASCII code of character in BL.
5. Scan for the byte in ES. If the byte is not found ZF ≠ 1 and repeat scanning.
6. If the byte is found ZF=1, display 01 in destination address. Otherwise, display 00 in destination address.

c) Find & Replace

1.Initialise ES and DI.
2. Store the no of characters in the string in CX.
3. Move the byte to be searched to AL.
4. Store the ASCII code of character in BL.
5. Scan for the byte in ES. If the byte is not found ZF ≠ 1 and repeat scanning.
6. If the byte is found ZF=1, move the content of BC register ES, DI.

COPYING A STRING

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV SI,1400H		Initialize starting address
		MOV DI,1500H		Initialize destination address
		MOV CX,0006H		Initialize array size
		CLD		Clear direction flag
		REP MOVSB		Copy the contents of source into destination until count
		HLT		Stop

SEARCHING FOR A CHARACTER IN THE STRING

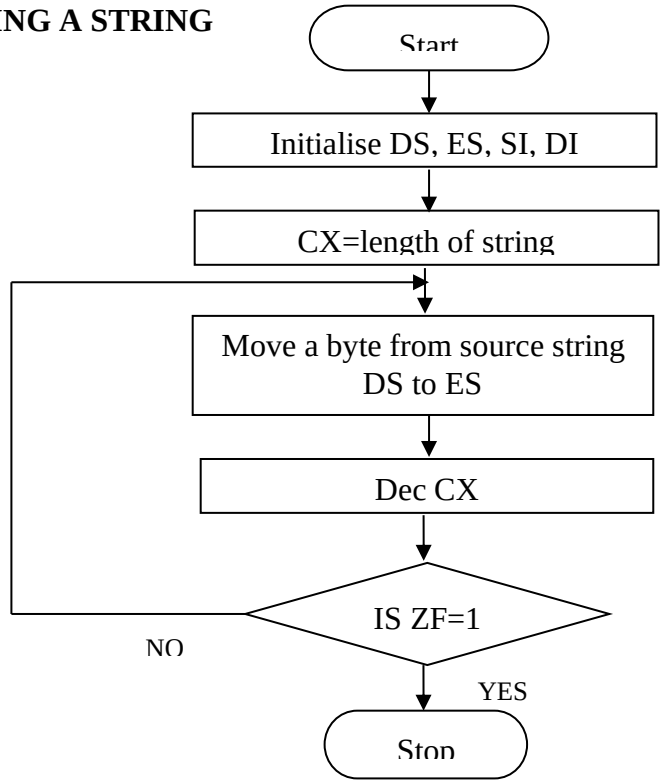
ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV DI,1400H		Initialize Starting address
		MOV SI,1500H		Initialize destination address
		MOV CX,0006H		Initialize array size
		MOV BL,00H		Initialize the relative address
		CLD		Clear direction flag
		MOV AL,08H		Store the character to be searched
	LOOP 2	NOP		Delay
		SCASB		Scan until the character is found
		JNZ LOOP 1		Jump if the character is found
		MOV[SI],BL		Move the relative address to SI
		INC SI		Increment the memory pointer
	LOOP 1	INC BL		Increment the relative address
		LOOP LOOP2		Repeat until the count reaches zero
		HLT		Stop

FIND AND REPLACE A CHARACTER IN THE STRING

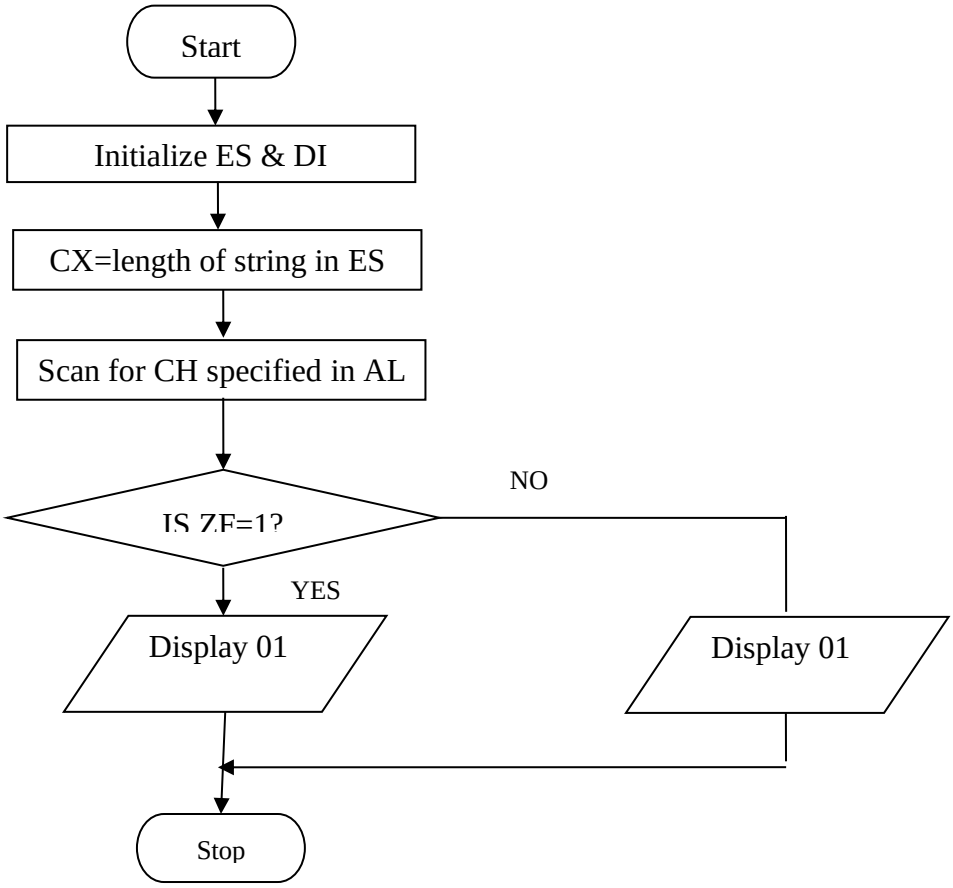
ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV DI,1400H		Initialize destination address
		MOV CX,0006H		Initialize array size
		CLD		Clear direction flag
		MOV AL,08H		Store the character to be searched
		MOV BH,30H		Store the character to be replaced
	LOOP2	SCASB		Scan until the character is found
		JNZ LOOP1		Is the string found
		DEC DI		Decrement the destination address
		MOV [DI],BH		Replace the string
	LOOP1	LOOP LOOP2		Continue until the count is zero
		HLT		Stop

FLOWCHART:

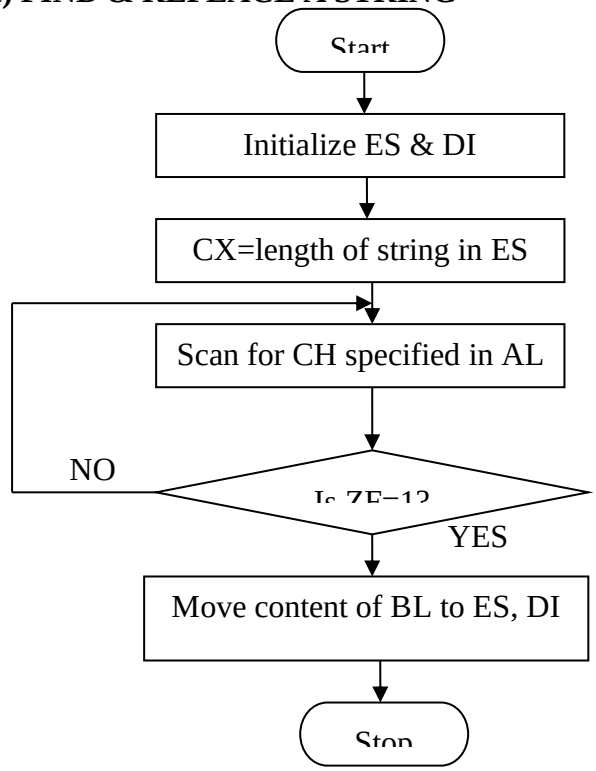
a) COPYING A STRING



b) SEARCHING A STRING



c) FIND & REPLACE A STRING



OUTPUT:

a)

b)

c)

RESULT:

REVIEW QUESTIONS:

1. What is use of stack segment and extra segment?
2. What is the use of flags register in 8086?
3. What is the use of index register in 8086?
4. What is the use of SCASB instruction?
5. Difference between REP and REPNE instruction.

Ex. No.	Sorting in Ascending Order using 8086	Date

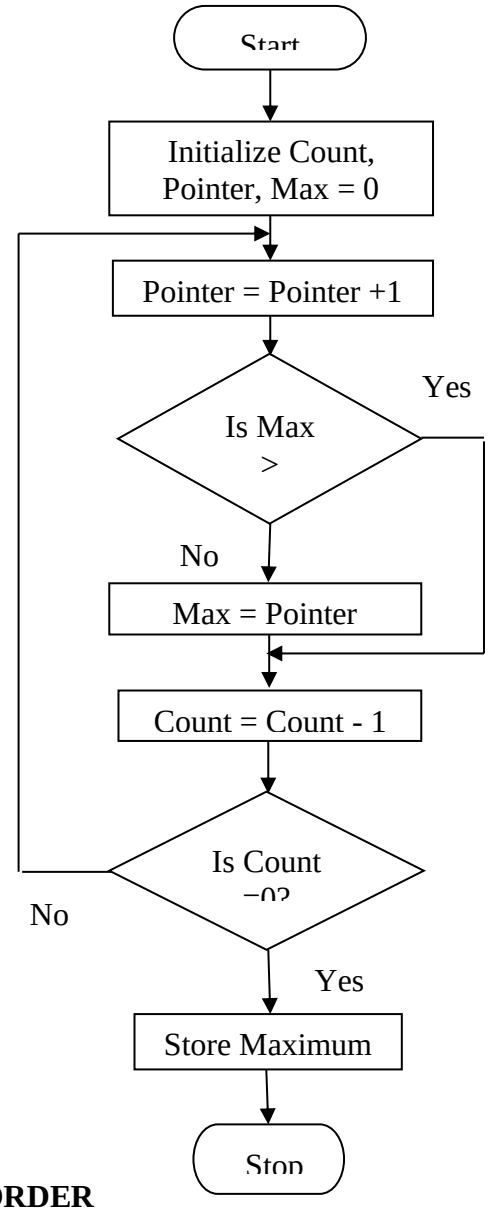
AIM: To write 8086 program to sort a given array of data in ascending order.

APPARATUS REQUIRED: 8086 Microprocessor kit, power supply.

ALGORITHM:

1. Load the array count in two registers C_1 and C_2.
2. Get the first two numbers.
3. Compare the numbers and exchange if necessary so that the two numbers are in ascending order.
4. Decrement C_2.
5. Get the third number from the array and repeat the process until C_2 is 0.
6. Decrement C_1 and repeat the process until C_1 is 0.

FLOWCHART:



ASCENDING ORDER

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV SI,1200H		Initialize memory location for array size
		MOV CL, [SI]		Number of comparisons in CL
	L4 :	MOV SI,1200H		Initialize memory location for array size
		MOV DL, [SI]		Get the count in DL
		INC SI		Go to next memory location
		MOV AL, [SI]		Get the first data in AL

	L3 :	INC SI		Go to next memory location
		MOV BL, [SI]		Get the second data in BL
		CMP AL, BL		Compare two data's
		JNB L1		If AL < BL go to L1
		DEC SI		Else, Decrement the memory location
		MOV [SI], AL		Store the smallest data
		MOV AL, BL		Get the next data AL
		JMP L2		Jump to L2
	L1 :	DEC SI		Decrement the memory location
		MOV [SI], BL		Store the greatest data in memory location
	L2 :	INC SI		Go to next memory location
		DEC DL		Decrement the count
		JNZ L3		Jump to L3, if the count is not reached zero
		MOV [SI], AL		Store data in memory location
		DEC CL		Decrement the count
		JNZ L4		Jump to L4, if the count is not reached zero
		HLT		Stop

OUTPUT:

RESULT:

Ex. No.	Searching of Largest number using 8086	Date

AIM: To write 8086 program to find largest number in a given array.

APPARATUS REQUIRED: 8086 Microprocessor kit, power supply.

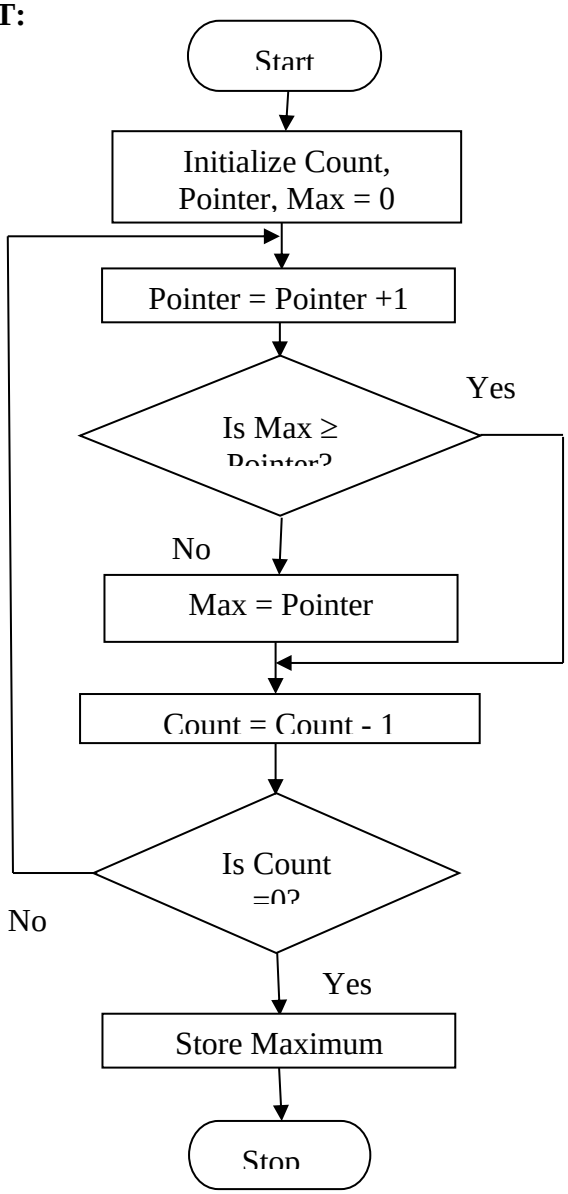
ALGORITHM:

1. Load the array count in a register CL.
2. Get the first two numbers.
3. Compare the numbers and exchange if the number is small.
4. Get the third number from the array and repeat the process until CL is 0.

LARGEST NUMBER IN AN ARRAY

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV SI, 1200H		Initialize array size
		MOV CL, [SI]		Initialize the count
		INC SI		Go to next memory location
		MOV AL, [SI]		Move the first data in AL
		DEC CL		Decrement the count
	L2 :	INC SI		Move the SI pointer to next data
		CMP AL, [SI]		Compare two data's
		JNB L1		If AL > [SI] then go to L1 (no swap)
		MOV AL, [SI]		Else move the large number to AL
	L1 :	DEC CL		Decrement the count
		JNZ L2		If count is not zero go to L2
		MOV DI, 1300H		Initialize DI with 1300H
		MOV [DI], AL		Else store the biggest number in 1300 location
		HLT		Stop

FLOWCHART:



OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. Explain CMP instruction.
2. Compare CMP and SUB instruction.
3. What are the flags get affected when CMP instruction used?
4. State the function of SI and DI register.
5. Distinguish the purpose of CS and DS registers in 8086

Ex. No.	Decimal arithmetic operations on 8 bit numbers using 8086	Date

AIM: To perform decimal arithmetic operations on 8 bit numbers and store the result in memory.

APPARATUS REQUIRED: 8086 microprocessor kit and power supply.

ALGORITHM:

DECIMAL ADDITION

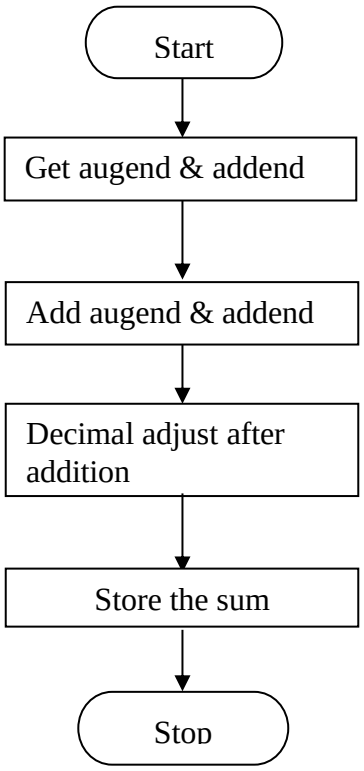
1. Load two 8 bit numbers from memory and store them in two registers AL and BL.
2. Add them and decimal adjust them using instruction DAA.
3. Move the result to a memory location.
4. Halt.

DECIMAL SUBTRACTION

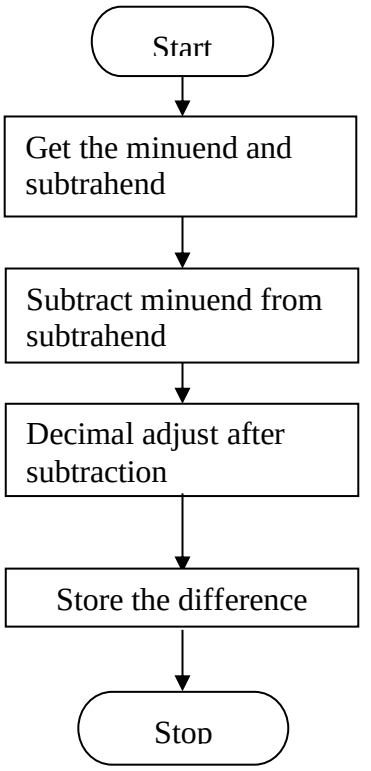
1. Load two 8 bit numbers from memory and store them in two registers AL and BL.
2. Subtract them and decimal adjust them using instruction DAS.
3. Move the result to a memory location.
4. Halt.

FLOWCHART:

DECIMAL ADDITION



DECIMAL SUBTRACTION



ADDITION

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV AL,[1300]		Get the first data in Ax reg
		MOV BL,[1301]		Get the second data in BX reg
		ADD AL,BL		Add the contents of both the regs AX,BX
		DAA		Decimal adjust after addition
		MOV [1502],AL		Store the sum
		HLT		Stop the program

SUBTRACTION

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV AL,[1300]		Get the first data in Ax reg
		MOV BL,[1301]		Get the second data in BX reg
		SUB AL,BL		Add the contents of both the regs AX,BX
		DAS		Decimal adjust after subtraction
		MOV [1502],AL		Store the difference
		HLT		Stop the program

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. Explain DAA instruction.
2. Explain DAS instruction.
3. What are the ASCII adjust instructions available in 8086?
4. State the function of AAA, DAA, CBW, DAS 8086 instructions.
5. Distinguish the purpose of CS and DS registers in 8086

Ex. No.	Movement of data block using MASM	Date

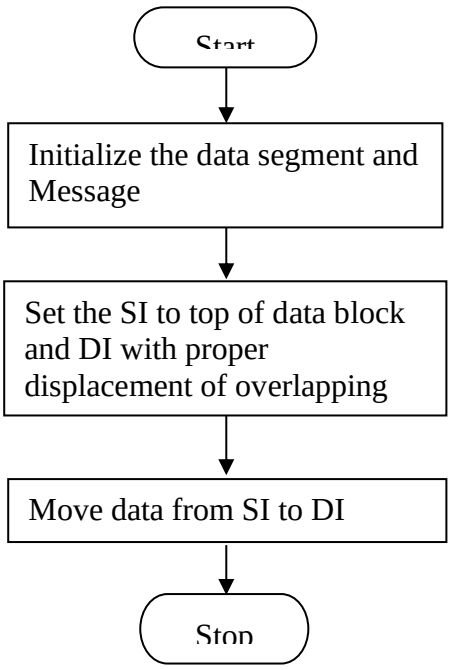
AIM: To move a block of data without overlap.

APPARATUS REQUIRED: Computer with MASM software

ALGORITHM:

1. Initialize the data segment memory location.
2. Set SI as the top of initial data block DI with displacement for overlapping.
3. Move the data from memory pointed by SI to DI.

FLOWCHART



PROGRAM	COMMENTS
ASSUME CS:CODE,DS:DATA	
DATA SEGMENT X DB 01H,02H,03H,04H,05H ;	Initialize Data Segments Memory Locations
DATA ENDS	
CODE SEGMENT	
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+05	05 is displacement for non 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	

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is use of JNC?
2. What is the use of Data Segment?
3. What is Assembler Directives?
4. Difference between ENDS and ENDP.
5. What are the salient features of MASM?

Ex. No.	Code-Conversion using 8086	Date

AIM: To convert a binary data to BCD data using 8086 microprocessor.

APPARATUS REQUIRED: 8086 microprocessor kit, Power supply.

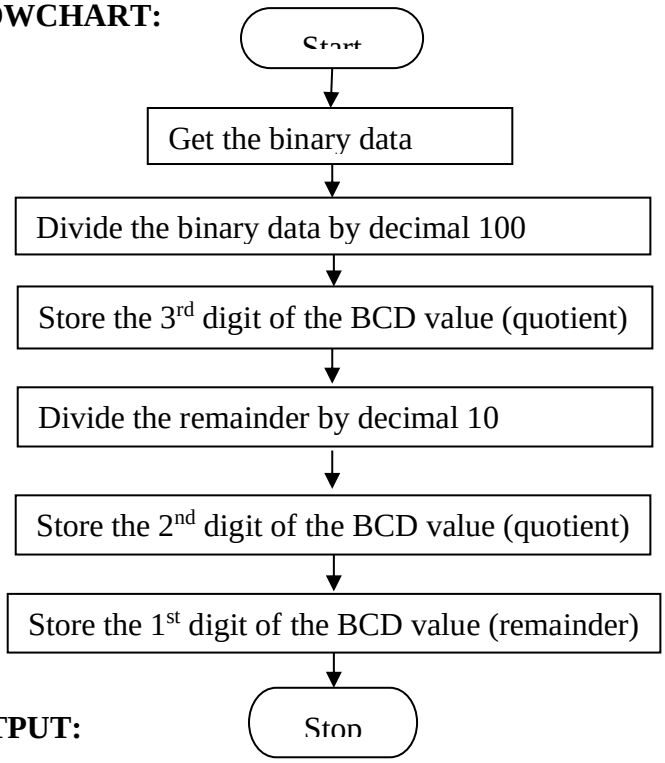
ALGORITHM:

1. Get the binary data.
2. Divide the data by decimal 100.
3. The quotient of the above result is the 3rd digit of BCD data.
4. Divide the remainder of the above result by decimal 10.
5. The quotient of the above result is the 2nd digit of the BCD data.
6. The remainder of the above result is the 1stdigit of the BCD data.

PROGRAM

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV AX,[1200]		Get the data in AX reg
		MOV CL,64H		Move 64H(100 in decimal) to CL
		DIV CL		Divide contents of AX by CL
		MOV [1501],AL		Storing the quotient as 3rd digit of BCD data
		MOV AL,AH		Move the remainder to AL
		MOV AH,00H		Clearing AH
		MOV CL,0AH		Move 0AH(10 in decimal) to CL
		DIV CL		Divide contents of AX by CL
		MOV CL,04		Move 04H to CL
		ROR AL,CL		Rotate right contents of AL by CL positions
		ADD AL,AH		Add the contents of AL and AH
		MOV [1500],AL		Storing the 2nd and 1st digit of BCD data
		HLT		Stop the program

FLOWCHART:



OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is meant by BCD?
2. How can you convert ASCII to HEX code?
3. What is meant by ROR instruction?
4. What is the importance of Code conversion?
5. Difference between BCD code and Excess 3 code.

Ex. No.	Password Checking using MASM	Date

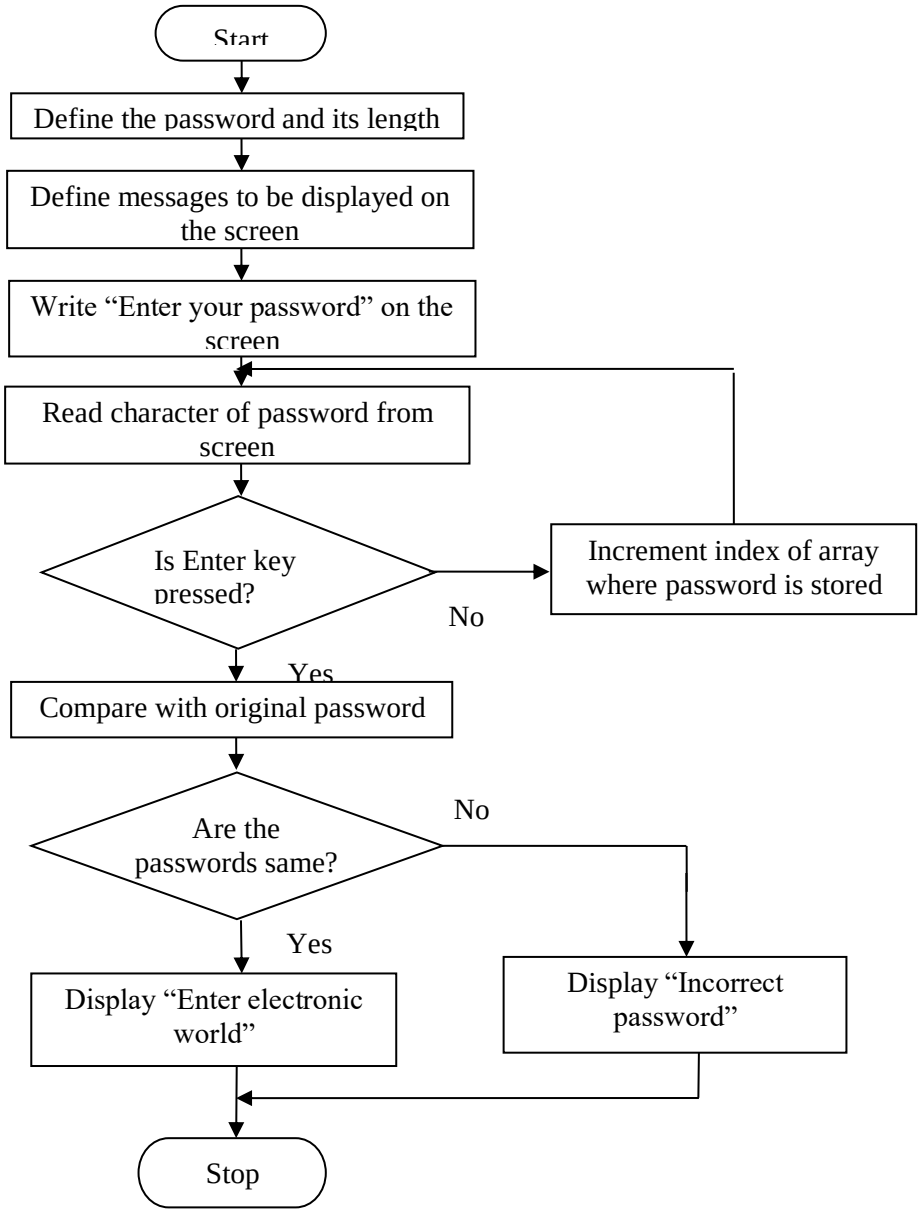
AIM: To write an ALP to check the password and validate user using MASM.

APPARATUS REQUIRED: PC with MASM

ALGORITHM:

1. Start
2. Define the password and its length
3. Define the messages to be displayed on the screen
4. Write "Enter password" on the screen by storing 09H in AH and calling INT 21H
5. Read characters into an array till enter key is pressed by storing 08H in AH and calling INT 21H
6. Compare the password entered with the password stored.
7. Display "Welcome to Electronics World" if the password is correct
8. Display "Incorrect password" if the password is incorrect

FLOWCHART:



PROGRAM	COMMENTS
ASSUME CS:CODE,DS:DATA	
DATA SEGMENT	Define password and its length
PASSWORD DB 'MASM1234'	
LEN EQU ($-PASSWORD)	
MSG1 DB 10,13,'ENTER YOUR PASSWORD: $'	Define messages to be displayed on screen
MSG2 DB 10,13,'WELCOME TO ELECTRONICS WORLD!!$'	
MSG3 DB 10,13,'INCORRECT PASSWORD!$'	
NEW DB 10,13,'$'	
INST DB 10 DUP(0)	Create an array of size 10
DATA ENDS	End of Data Segment
CODE SEGMENT	Start of code segment
START: MOV AX,DATA	Move data to AX
MOV DS,AX	Copy content of AX to DS
LEA DX,MSG1	
MOV AH,09H	Write MSG1 on screen
INT 21H	
MOV SI,00	Initialize source index to 00
UP1: MOV AH,08H	Read character into array by comparing each character entered with character return (0DH)
INT 21H	
CMP AL,0DH	
JE DOWN	
MOV [INST+SI],AL	
MOV DL,'*'	
MOV AH,02H	
INT 21H	
INC SI	
JMP UP1	
DOWN: MOV BX,00	If enter key is pressed move to loop for comparing passwords
MOV CX,LEN	Move content of Len to CX
CHECK: MOV AL,[INST+BX]	Move first character of entered password to AL
MOV DL,[PASSWORD+BX]	Move first character of original password to DL
CMP AL,DL	Compare characters
JNE FAIL	If incorrect, move to FAIL
INC BX	Increment BX
LOOP CHECK	Move to CHECK to compare the next character
LEA DX,MSG2	If passwords match, display MSG2 on screen
MOV AH,09H	
INT 21H	
JMP FINISH	Jump to FINISH
FAIL: LEA DX,MSG3	If passwords do not match, display MSG3 on screen
MOV AH,009H	
INT 21H	
FINISH: NOP	Break from code and stop
CODE ENDS	
END START	
END	

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. How do you read and write characters on to screen using interrupts?
2. What is the significance of LEA instruction?
3. What is an assembler directive?
4. Give some examples for assembler directives?
5. How a procedure is represented in assembler directive?

Ex. No.	Finding RAM size using MASM	Date

AIM:

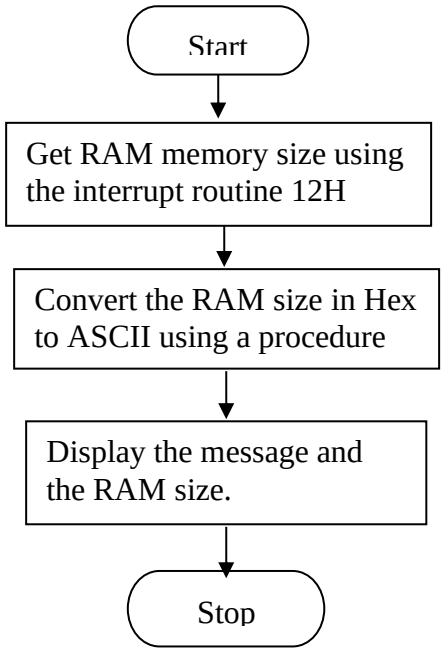
To write an 8086 ALP to find memory size of the PC you are using. Using appropriate message, the display should indicate memory size in Kilo bytes using 4 hex digits.

APPARATUS REQUIRED: PC with MASM

ALGORITHM:

1. Initialize the data segment with required constants.
2. Get the RAM sizes in KB using the INT 12H.
3. Convert the obtained RAM size in HEX to ASCII.
4. Display the message and the RAM size.

FLOWCHART



PROGRAM	COMMENTS
ASSUME CS:CODE,DS:DATA	
DATA SEGMENT	
MSG DB 'MEMORY SIZE IN KILO BYTES ='	Initializing required constants
ASCRES DB 4 DUP(?),'HEX',0CH,0AH,'$'	
RES DW ?	
HEXCODE DB '0123456789ABCDEF'	
DATA ENDS	End of Data Segment
CODE SEGMENT	

HEX_ASC PROC	Start of Procedure
MOV DL,10H	Move 10H to DL
MOV AH,0	Make higher order byte of AX as 0
MOV BX,0	Make BX as 0
DIV DL	Divide lower order byte of memory size by 10H
MOV BL,AL	Move quotient to BL
MOV DH,HEXCODE[BX]	Move hex equivalent of quotient to DH
MOV BL,AH	Move remainder to BL
MOV DL,HEXCODE[BX]	Move hex equivalent of remainder to DH
RET	End of Procedure
HEX_ASC ENDP	End of Procedure
MAIN: MOV AX,DATA	Move the message start data to Ax
MOV DS,AX	Move the contents to DS
INT 12H	Stores RAM Memory size in Hex in KB to AX
MOV RES,AX	Stores RAM size in data segment location RES
MOV AL, BYTE PTR RES	A byte of data pointed by RES is stored in AL
CALL HEX_ASC	Calling the procedure
MOV [ASCRES+2],DH	Storing the lowest order byte of ram size in ASCRES+2 mem location
MOV [ASCRES+3],DL	Storing the next lower order byte of ram size in ASCRES+3 mem location
MOV AL,BYTE PTR RES+1	A byte of data pointed by RES+1 is stored in AL
CALL HEX_ASC	Calling the procedure
MOV [ASCRES],DH	Storing the next lower order byte of ram size in ASCRES mem location
MOV [ASCRES+1],DL	Storing the higher order byte of ram size in ASCRES+1 mem location
MOV DX,OFFSET MSG	Get start address of MSG in DX
MOV AH,09H	To display data
INT 21H	Interrupt for BIOS
MOV AH,4CH	To return to ms-dos
INT 21H	Interrupt for BIOS
CODE ENDS	Forcing the assembler to start next address which is divisible by 16 i.e. creating the 4 word boundry
END MAIN	Ending the main program

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is a RAM?
2. What are the types of RAM?
3. How many 32kB RAMs can be interfaced with 8086?
4. What is the necessity of RAM in processor?
5. Differentiate EPROM and EEPROM.

Ex. No.	Displaying System Date using MASM	Date

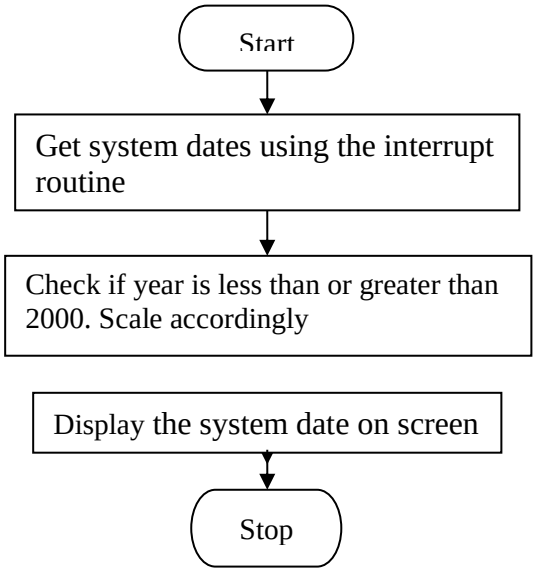
AIM: To write an 8086 ALP to display System date using MASM.

APPARATUS REQUIRED: PC with MASM

ALGORITHM:

1. Start
2. Define the days of a week and the months of a year
3. Move 2AH to AH and call INT 21H. Now the system year will be saved in CX, month in DH, day in DL.
4. Display day of the week.
5. Display month.
6. Check if year is less or greater than 2000. Scale accordingly.
7. Display year.
8. Stop.

FLOWCHART



PROGRAM	COMMENTS
ASSUME CS:CODE, DS:DATA	Code and Data segment initialization
DATA SEGMENT	Start of Data Segment
YY DW ?	
MM DB ?	
D DB ?	
TDAY DW SUN,MON,TUE,WED,THU,FRI,SAT	Define the days of a week
SUN DB'SUNDAY,$'	
MON DB'MONDAY,$'	
TUE DB'TUESDAY,$'	
WED DB'WEDNESDAY,$'	
THU DB'THURSDAY,$'	
FRI DB'FRIDAY,$'	
SAT DB'SATURDAY,$'	
TMON DW JAN,FEB,MAR,APR,MAY,JUN,JUL, AUG, SEP, OCT,NOV,DEC	Define the months of a year
JAN DB'JANUARY,$'	

FEB DB'FEBRUARY,$'	
MAR DB'MARCH,$'	
APR DB'APRIL,$'	
MAY DB 'MAY,$'	
JUN DB 'JUNE,$'	
JUL DB 'JULY,$'	
AUG DB 'AUGUST,$'	
SEP DB 'SEPTEMBER,$'	
OCT DB 'OCTOBER,$'	
NOV DB 'NOVEMBER,$'	
DEC DB 'DECEMBER,$'	
DATA ENDS	End of Data Segment
CODE SEGMENT	
DISCHAR MACRO CHAR	
PUSH AX	Save AX
PUSH DX	Save DX
MOV DL,CHAR	Display character
MOV AH,02H	Move 02 to AH
INT 21H	
POP DX	Restore DX
POP AX	Restore AX
ENDM	End of Main
START: MOV AX,DATA	Initialize data segment
MOV DS,AX	
CALL FAR PTR PDATE	Display code
MOV AH,4CH	Exit to DOS
INT 21H	
PDATE PROC FAR	Move 2A to AH
MOV AH,2AH	
INT 21H	
MOV [YY],CX	Save year
MOV [MM],DH	Save month
MOV [D],DL	Save day
MOV AH,0	Get a day of the week
ROL AX,1	Rotate right Accumulator
MOV SI,OFFSET TDAY	Address day table
ADD SI,AX	Add SI and AX
MOV DX, [SI]	Address day of week
MOV AH,09H	Display day of week
INT 21H	
MOV AL,[D]	Get day of month
MOV AH,00H	Clear the contents of AH
AAM	
OR AH,AH	Convert to BCD
JZ DIGIT0	If tens is 0
ADD AH,30H	Convert tens
DISCHAR AH	Display tens

DIGIT0: ADD AL,30H	Convert units
DISCHAR AL	Display unit
DISCHAR ' '	Leave space
MOV AL,[MM]	Get month
SUB AL,1	Sub the contents of AL with 1
MOV AH,0	Move 0 to AH
ROL AX,1	Rotate accumulator
MOV SI,OFFSET TMON	Address month table
ADD SI,AX	Add Si and AX contents
MOV DX, [SI]	Address month
MOV AH,09H	Display month
INT 21H	
MOV AX,[YY]	Read year
CMP AX,07D0H	Check for year 2000
JB DIS19	If below year 2000
SUB AX,07D0H	Scale for 00-99
DISCHAR '2'	Display 2
DISCHAR '0'	Display 0
JMP SKIP	
DIS19: SUB AX,076CH	Scale 1900-1999
DISCHAR '1'	Display 1
DISCHAR '9'	Display 9
SKIP: AAM	Convert to BCD
ADD AX,3030H	Convert to ASCII
DISCHAR AH	Display tens
DISCHAR AL	Display units
RET	Return
PDATE ENDP	
CODE ENDS	
END START	End of the program

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is the functionality of OFFSET?
2. What does AAM perform?
3. What is the functionality of MACRO?
4. Distinguish PUSH and POP instructions.
5. Distinguish MACRO and PROC.

Ex. No.	Matrix Operation using 8086	Date

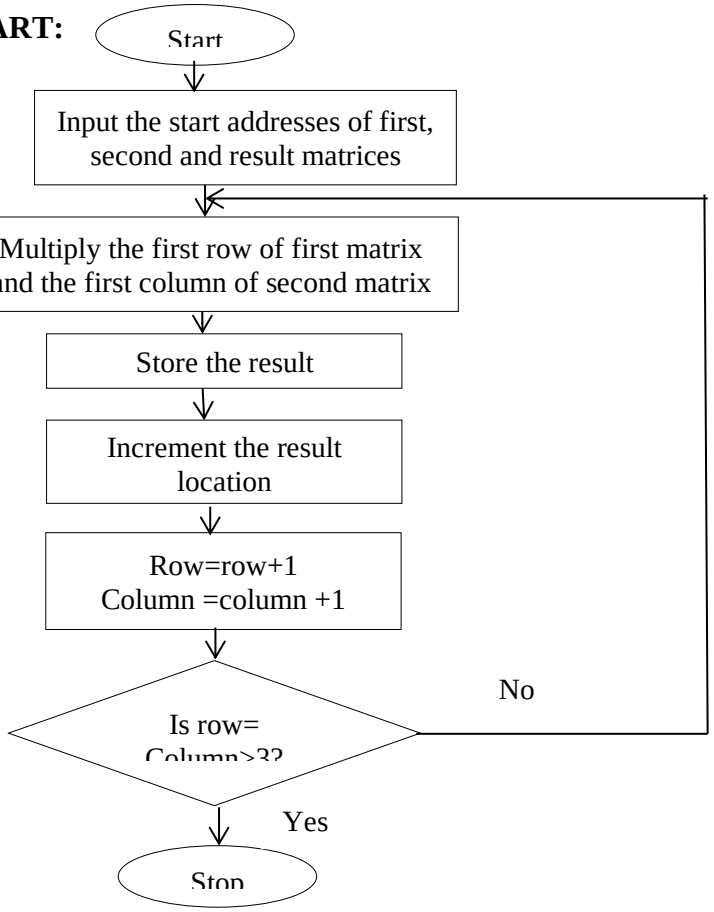
AIM: To perform multiplication of two 3×3 matrices using 8086 microprocessor.

APPARATUS REQUIRED: 8086 microprocessor kit, Power supply.

ALGORITHM:

1. Get the start addresses of first matrix, second matrix and the result matrix.
2. Move the first and second matrices into AL and BL respectively.
3. Multiply the contents of first row and first column and store the result in the first location in the result matrix.
4. Increment the result matrix location and store the product of first row and second column in the new location.
5. Repeat the above steps till the end of the result location is reached.
6. Stop.

FLOWCHART:



PROGRAM

ADDRESS	LABEL	PROGRAM	OPCODE	COMMENTS
		MOV SI,1200H		Store start address of 1st matrix in SI
		MOV BP, 1300H		Store start address of 2nd matrix in BP
		MOV DI,1500H		Store start address of result matrix in DI
	L2:	MOV CX,0000H		Initialize the count
	L1:	MOV AL,[SI]		Move 1st value of 1st row in 1st matrix to Al
		MOV BL,[BP]		Move 1st value of 1st column in 2nd matrix to BL
		MUL BL		Multiply both the values
		ADD CX,AX		Add it with previous multiplied value
		ADD BP,03		Move to next column in 2nd matrix
		INC SI		Move to next element in same row in 1st matrix

		CMP BP,1309		Check if end of column reached
		JB L1		Jump to L1
		MOV [DI],CL		Move elements to result matrix
		INC DI		Points to next element of result matrix
		SUB SI,03		Move to previous row
		SUB BP,08		Move to next column
		CMP BP,1303		Check if end of column reached
		JB L2		Jump to L2
		ADD SI,03		Move to next row
		SUB BP,03		Start with 1st column
		CMP DI,1509		Check if end of result matrix reached
		JB L2		Jump to L2
		HLT		Stop

OUTPUT:

RESULT:

REVIEW QUESTIONS

1. How can matrix addition be performed using 8086?
2. Explain the importance of CMP instruction.
3. Difference between Matrix multiplication and Divison
4. What is meant by Index register?
5. Explain the importance of matrix in Calculations.

Ex. No.	Traffic light controller using 8086	Date

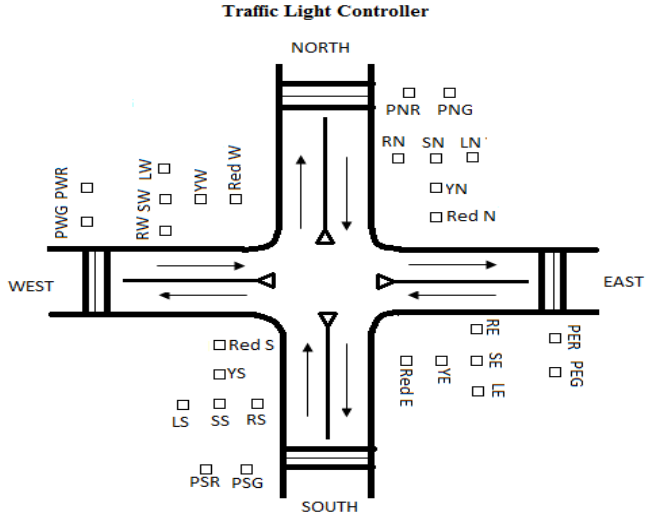
AIM: To write a program for traffic light controller by interfacing with 8086 microprocessor.

APPARATUS REQUIRED: 8086 microprocessor kit, Power supply, CRO, Traffic light controller interfacing board.

THEORY:

The 8086 micro processor announced by Intel in 1978 was Intel‘s first 16-bit micro processor and the first in the family it is sub-divided into two principle units. The execution unit EU, including the ALU eight 160bit general register, a 16-bit flag register and register unit. The bus interface unit, includes an address for adder calculation for 16-bit segment register a 16-bit instruction pointer {ip} a 6 bytes instruction queue and bus control logic, The 8086 chip has 40-pins including 16data pin and 20 address pins, the 8086 microprocessor is used in various applications. In our program use 8086 in traffic light controller using 8086 LCD display

TRAFFIC CONTROL DIAGRAM:



Abbreviations

PWG: Pedestrian West Green
PWR: Pedestrian West Red
PNR: Pedestrian North Red
PNG: Pedestrian North Green
PER: Pedestrian East Red
PEG: Pedestrian East Green
PSR: Pedestrian South Red
PSG: Pedestrian South Green
RW: West Right
SW: West Straight
LW: West Left
YW: West Yellow
Red W: West Red
RE: East Right
SE: East Straight
LE: East Left
YE: East Yellow
Red E: East Red
RS: South Right
SS: South Straight
LS: South Left
YS: South Yellow
Red S: South Red

PORT CONFIGURATION:

CYCLE	PORT A							
	PA_7	PA6	PA_5	PA_4	PA_3	PA_2	PA_1	PA_0
	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1
	RN	LN	R_tN	SE	OE	RE	LE	R_tE
	R	G	G	G	Y	R	G	G
I	0	1	0	0	0	1	0	0
	0	1	0	0	1	0	0	0
II	1	0	0	1	0	0	1	0
III	1	0	0	0	0	1	0	0
IV	1	0	0	0	0	1	0	0
	0	0	0	0	0	1	0	0

CYCLE	PORT B							
	PB_7	PB_6	PB_5	PB_4	PB_3	PB_2	PB_1	PB_0
	D_{20}	D_{19}	D_{18}	D_{17}	DL_7 DL_8	DL_5 DL_6	DL_3 DL_4	DL_1 DL_2
	SS	OS	RS	LS	PW	PN	PE	PS
	G	Y	R	G	Dual	Dual	Dual	Dual
I	0	0	1	0	0	1	1	1
II	0	0	1	0	1	0	1	1
	0	1	0	0	1	0	1	1
III	1	0	0	1	1	1	0	1
IV	0	0	1	0	1	1	1	0

CYCLE	PORT C							
	PC_7	PC_6	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0
	D_{16}	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9
	R_tS	SW	OW	RW	LW	R_tW	SN	ON
	G	G	Y	R	G	G	G	Y
I	0	0	0	1	0	0	1	0
II	0	0	0	1	0	0	0	0
	0	0	0	1	0	0	0	0
	0	0	1	0	0	0	0	0
	0	1	0	0	1	0	0	0
	0	1	0	0	1	0	0	1

SEQUENCE OF OPERATION:

CYCLE I: NORTH- GREEN AND EAST- ORANGE; Pedestrian can cross the road on West.
CYCLE II: EAST- GREEN AND SOUTH - ORANGE; Pedestrian can cross the road on North.
CYCLE III: SOUTH- GREEN AND WEST - ORANGE; Pedestrian can cross the road on East.
CYCLE IV: WEST - GREEN AND NORTH - ORANGE; Pedestrian can cross the road on South.

PROGRAM FOR TLC USING 8086 LCD MNEMONICS:

```
ORG  1000H
CNTRL  EQU 26H
PORTA  EQU 20H
PORTB  EQU 22H
PORTC  EQU 24H
```

ADDRESS	LABEL	MNEMONICS	OPCODE
1000	START	MOV AL,80H	
1003		OUT CNTRL,AL	
1005	REPEAT	MOV BX,LOOKUP	
1009		MOV SI,LABEL	
100D		CALL OUT	
1010		MOV AL,[SI]	
1012		OUT PORTA,AL	
1014		CALL CELAY1	
1017		INC SI	
1018		INC BX	
1019		CALL OUT	
101C		MOV AL,[SI]	
101E		OUT PORTB,AL	
1020		CALL DELAY1	
1023		INC SI	
1024		INCBX	
1025		CALL OUT	
1028		MOV AL,[SI]	
102A		OUT PORTC,AL	
102C		CALL DELAY1	
102F		INC SI	
1030		INC BX	
1031		CALL OUT	
1034		MOV AL,[SI]	
1036		OUT PORTC,AL	
1038		INC SI	
1039		MOV AL,[SI]	
103B		OUT PORTA,AL	
103D		CALL DELAY1	
1040		JMP REPEAT	
1043	OUT	MOV AL,[BX]	
1045		OUT PORTC,AL	
1047		INC BX	
1048		MOV AL,[BX]	
104A		OUT PORTB,AL	
104C		INC BX	
104D		MOV AL,[BX]	
104F		OUT PORTA,AL	
1051		CALL DELAY	
1054		RET	
1055	DELAY	MOV DI,00040H	
1059	A	MOV DX,0FFFFH	

105D	A1	DEC DX	
105E		JNZ A1	
1060		DEC D1	
1061		JNZ A	
1063		RET	
1064	DELAY1	MOV DI,00015H	
1068	B	MOV DX,0FFFFH	
106C	B1	DEC DX	
106D		JNZ B1	
106F		DEC DI	
1070		JNZ B	
1072		RET	
1073	LOOK UP	DB 12H,27H,44H,10H	
1077		2BH,92H,10H,9DH	
107B		84H,48H,2EH,84H	
107F		DB 48H 4BH,20H,49H, 04H	
1083		END	

RESULT:

REVIEW QUESTIONS:

1. Give the sequence of operation in traffic light controller.
2. What is the name of the peripheral device used to interface traffic light controller with microprocessor?
3. What is 8255?
4. How many input and output ports are in PPI?
5. What is BSR mode?

Ex. No.	Interfacing Stepper Motor using 8086	Date

AIM: To write an assembly language program in 8086 to rotate the motor at different speeds.

APPARATUS REQUIRED: 8086 Microprocessor kit, Power Supply & Stepper Motor.

ALGORITHM:

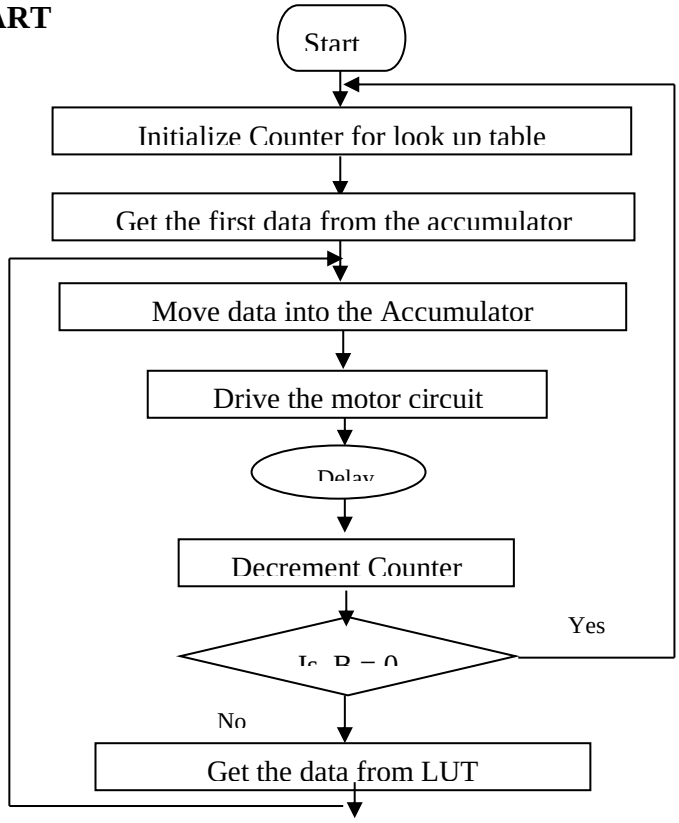
For running stepper motor clockwise and anticlockwise directions

(i) Get the first data from the lookup table.
(ii) Initialize the counter and move data into accumulator.
(iii) Drive the stepper motor circuitry and introduce delay
(iv) Decrement the counter is not zero repeat from step (iii)
(v) Repeat the above procedure both for backward and forward directions.

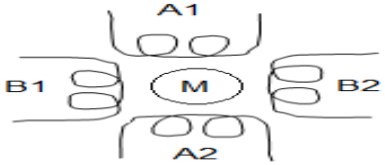
PROGRAM:

PROGRAM	COMMENTS
START : MOV DI, 1200	Initialize memory location to store the array of number
MOV CX, 0004H	Initialize array size
LOOP 1 : MOV AL, [DI]	Copy the first data in AL
OUT C0, AL	Send it through port address
MOV DX, 1010	Introduce delay
LOOP2: DEC DX	
JNZ LOOP2	
INC DI	Go to next memory location
LOOP LOOP1	Loop until all the data's have been sent
JMP START	Go to start location for continuous rotation

FLOW CHART



WINDING DIAGRAM OF STEPPER MOTOR



LOOK UP TABLE:

	Clockwise Direction					Anti-Clockwise Direction				
Memory Location	**A1**	**A2**	**B1**	**B2**	**HEX Code**	**A1**	**A2**	**B1**	**B2**	**HEX Code**
1200	1	0	0	1	09	1	0	1	0	0A
1201	0	1	0	1	05	0	1	1	0	06
1202	0	1	1	0	06	0	1	0	1	05
1203	1	0	1	0	0A	1	0	0	1	09

RESULT:

REVIEW QUESTIONS:

1. What are the applications of stepper motor
2. Discuss the salient features of stepper motor
3. What are the scheme used in stepper motor
4. If Ns=4 & Nr=3. Calculate the step size.
5. How can the speed of stepper motor can be controlled?

Ex. No.	**Interfacing Programmable Peripheral Interface 8255 using 8086**	Date

AIM: To initialize port A as input and to give the data by SPDT switches through port A and store the data for mode 0, 1, 2 of 8255.

APPARATUS REQUIRED: Microprocessor kit, power supply, 8255 interface board.

THEORY:

BSR mode

Bit set/reset, applicable to PC only. One bit is S/R at a time. Control word:

D7	D6	D5	D4	D3	D2	D1	D0
0 (0=BSR)	X	X	X	B2	B1	B0	S/R (1=S,0=R)

Bit select: (Taking Don't care's as 0)

B2	B1	B0	PC bit	Control word (Set)	Control word (reset)
0	0	0	0	0000 0001 = 01h	0000 0000 = 00h
0	0	1	1	0000 0011 = 03h	0000 0010 = 02h
0	1	0	2	0000 0101 = 05h	0000 0100 = 04h
0	1	1	3	0000 0111 = 07h	0000 0110 = 06h
1	0	0	4	0000 1001 = 09h	0000 1000 = 08h
1	0	1	5	0000 1011 = 0Bh	0000 1010 = 0Ah
1	1	0	6	0000 1101 = 0Dh	0000 1100 = 0Ch
1	1	1	7	0000 1111 = 0Fh	0000 1110 = 0Eh

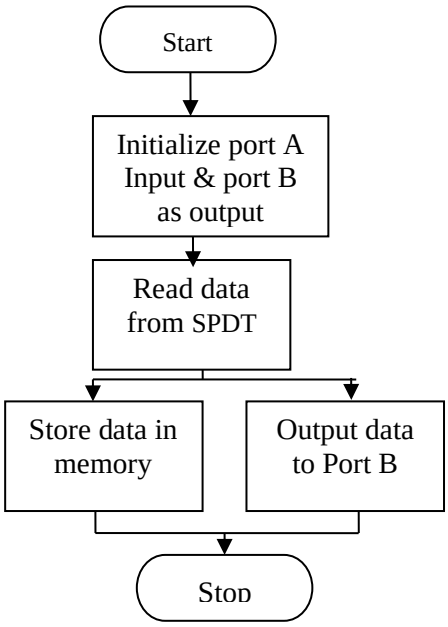
I/O mode:

D7	D6	D5	D4	D3	D2	D1	D0
1 (1=I/O)	GA mode select		PA	PCU	GB mode select	PB	PCL

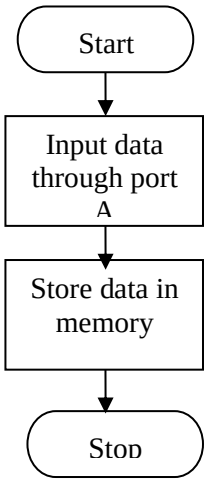
- D6, D5: GA mode select:
  - 00 = mode0
  - 01 = mode1
  - 1X = mode2
- D4(PA)0, D3(PCU): 1=input 0=output
- D2: GB mode select: 0=mode0, 1=mode1
- D1(PB), D0(PCL): 1=input 0=output

FLOWCHART:

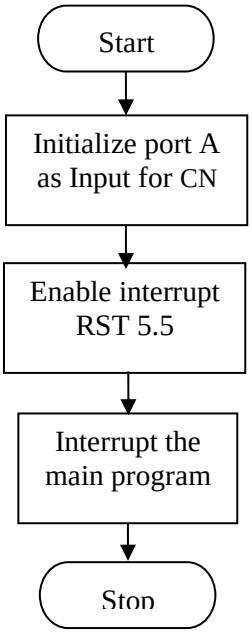
MODE 0



MODE 1



MODE 2



MODE 0:

PROGRAM	COMMENTS
MOV DX,00C6	Initialize DX reg with port address for control word
MOV AL,90H	Control word
OUT DX,AL	Send it to control port
MOV DX,00C0	Set DX reg with port A address
IN AL,DX	Get the contents of port A in AL
MOV DX,00C2	Set DX reg with port A address
OUT DX,AL	Send the contents of port B to port address
HLT	Stop

MODE 1:

PROGRAM	COMMENTS
MOV DX,00C6	Initialize DX reg with port address for control word
MOV AL,0B0H	Control word
OUT DX,AL	Send it to control port
MOV AL,09H	Control word for BSR mode
OUT DX,AL	Send it to control port
MOV DX,00C4	Set DX reg with port c address
L1:IN AL,DX	Get the contents of port C in AL
AND AL,20H	Mask RST 6.5
JZ L1	Check whether it is enabled
MOV DX,00C0	Set DX reg with port A address
IN AL,DX	Get the contents of port A in AL
MOV DX,00C2	Set DX reg with port B address
OUT DX,AL	Send the contents of AL to port B address
HLT	Stop

MODE 2:

PROGRAM	COMMENTS
MOV DX,00C6	Initialize DX reg with port address for control word
MOV AL,0C0H	Control word
OUT DX,AL	Send it to control port
MOV AL,09H	Control word for BSR mode
OUT DX,AL	Send it to control port
MOV DX,00C4	Set DX reg with port c address
L1:IN AL,DX	Get the contents of port C in AL
AND AL,20H	Mask RST 6.5
JZ L1	Check whether it is enabled
MOV DX,00C0	Set DX reg with port A address
IN AL,DX	Get the contents of port A in AL
MOV DX,00C2	Set DX reg with port B address
OUT DX,AL	Send the contents of AL to port B address
HLT	Stop

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. Show How 8255 can be operated in mode 1?
2. Show how 8255 can be operated in mode2?
3. Write the control word?
4. What is BSR mode?
5. Explain Mode 2of 8255.

Ex. No.	Interfacing Programmable Keyboard and Display Controller 8279 using 8086	Date

AIM: To display rolling message "ST JOSEPHS" in the display (or) to accept a key and display it.

APPARATUS REQUIRED: 8086 microprocessor key, power supply of interfacing board.

ALGORITHM:

Display:

1. Initialise the count.
2. Set 8279 for 8 digit character display, right entry.
3. Set 8279 for clearing to display.
4. Write the command to display.
5. Load the character into display and accumulator kit.
6. Introduce the delay.
7. Repeat from step 1.

Read a pressed key:

The code will be entered into the FIFO whenever a key is pressed.

i) Read the FIFO

ii) Check if the least significant 3 bits is less than 0111 B, because any key closure will increment the row indicates by the 3 AAA bits.

iii) Read the data from FIFO RAM, which is the key code.

THEORY:

1. Display Mode Setup: ***Control word-10H***

0	0	0	D	D	K	K	K
0	0	0	1	0	0	0	0

DD

00 - 8Bit character display left entry
01 - 16Bit character display left entry
10 - 8Bit character display right entry
11 - 16Bit character display right entry

KKK- Key Board Mode

000 - 2Key lockout.

2. Clear Display:*Control word-DCH*

1	1	0	CD	CD	CD	CF	CA
1	1	0	1	1	1	0	0

11 **A0-3: B0-3 =FF**

1-Enables Clear display
0-Contents of RAM will be displayed

1-FIFO Status is cleared

1-Clear all bits
(Combined effect of CD)

3. Write Display:*Control word-90H*

1	0	0	AI	A	A	A	A
1	0	0	1	0	0	0	0

Auto increment = 1, the row address selected will be incremented after each of read and write operation of the display RAM.

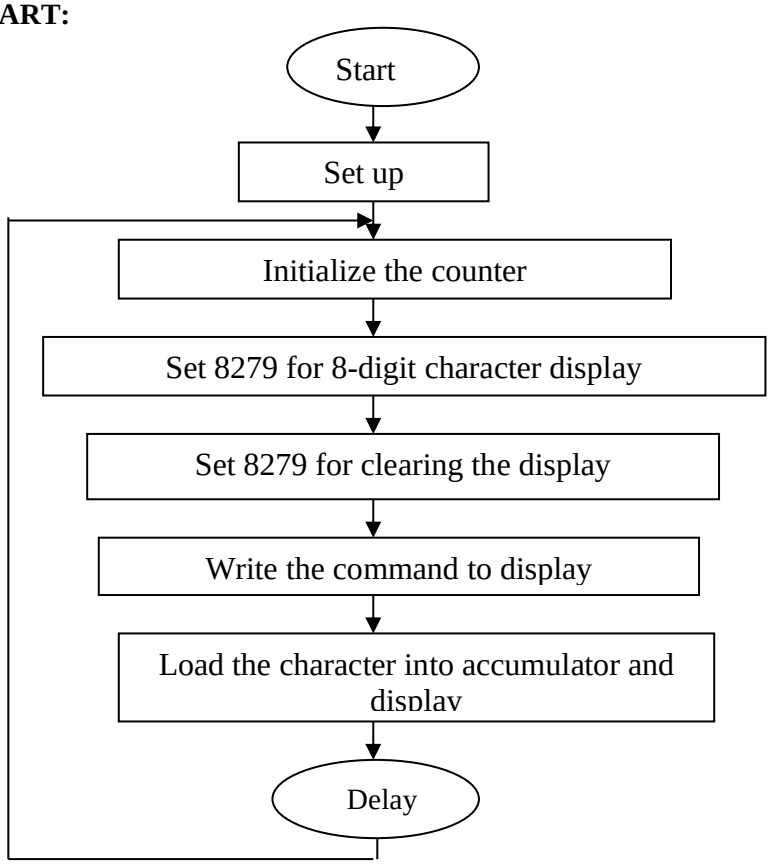
Selects one of the 16 rows of display.

4. Read a key Pressed:

In the scanned keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard and the status of CNTL and SHIFT lines. In the hardware, CNTL and SHIFT inputs of 8279 are grounded.

0	0	E	E	E	X	X	X
CNTL	SHIFT	Scan (indicates the row in which the key was found)			Return (indicates the column in which the key was found)		

FLOWCHART:



READ A PRESSED KEY:

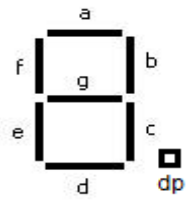
PROGRAM	COMMENTS
LOOP: IN AL C2	Get the pressed key data from C2
TEST AL, 07	Check if key is pressed
JZ LOOP	If key is not pressed jump to loop
MOV AL,40	Move command word to Accumulator
OUT C2, AL	Output the data in C2
IN AL,C0	Get the key information
MOV AH,00	Clear AH register
MOV BX,00C0H	Move 00C0 to BX
SUB AX,BX	Subtract BX from AX
MOV [1200], AL	Display the same in 1200 memory location
HLT	Finish the program

ROLLING DISPLAY:

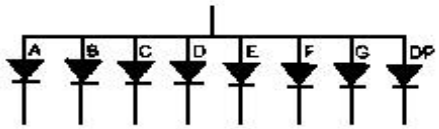
PROGRAM	COMMENTS
START:MOV SI,1200H	Initialize array
MOV CX,000FH	Initialize array size
MOV AL,10	Store the control word for display mode
OUT C2,AL	Send through output port
MOV AL,0CC	Store the control word to clear display
OUT C2,AL	Send through output port
MOV AL,090	Send the control word to write display
OUT C2,AL	Send through output port
L1:MOV AL,[SI]	Get the first data
OUT C0,AL	Send through output port
CALL DELAY	Give delay
INC SI	Go&get next data
DEC CX	Decrement the value of count array
JNZ LI	Jump on no zero condition
JMP START	Go to starting location
DELAY:MOV DX,0FFFFH	Store 16 bit count value
LOOP1:DEC DX	Decrement count value
JNZ LOOP1	Loop until count values becomes zero
RET	Return to main program

OUTPUT:

MEMORY LOCATION	7-SEGMENT LED FORMAT								HEX DATA	DISPLAY
	d	c	b	a	dp	g	F	e		
1200H										
1201H										
1202H										
1203H										
1204H										
1205H										
1206H										
1207H										



7-segment display with segment identification



Common anode Equivalent circuit

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is the control word for Display mode setup?
2. Write the control word for Clear Display.
3. Write the control word for Write Display.
4. What is meant by Control Register?
5. What is meant by Mode word?

Ex. No.	Printing Single Character using VBMB-005 and 8086	Date

AIM: To write an 8086 ALP to print a single character.

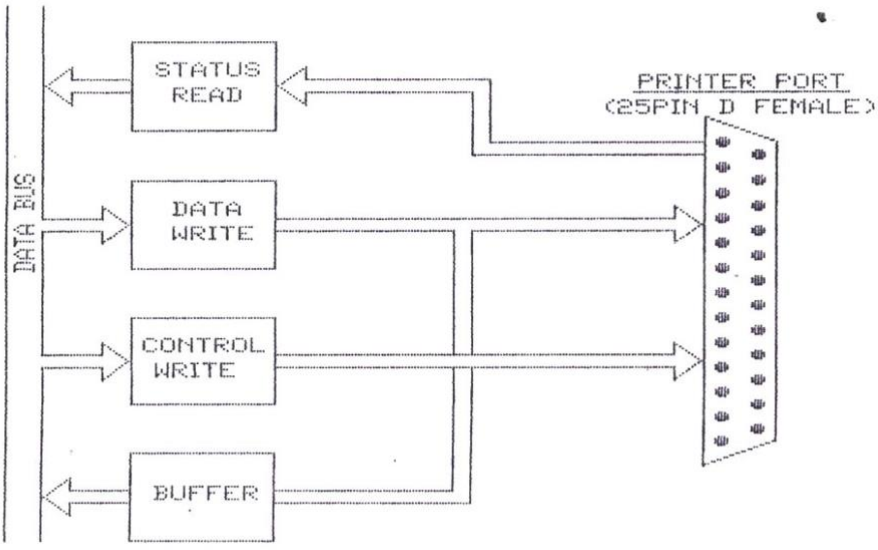
APPARATUS REQUIRED: 8086 microprocessor, power supply, vbmb-005 (printer interfacing board),printer, printer interfacing cable.

ALGORITHM:

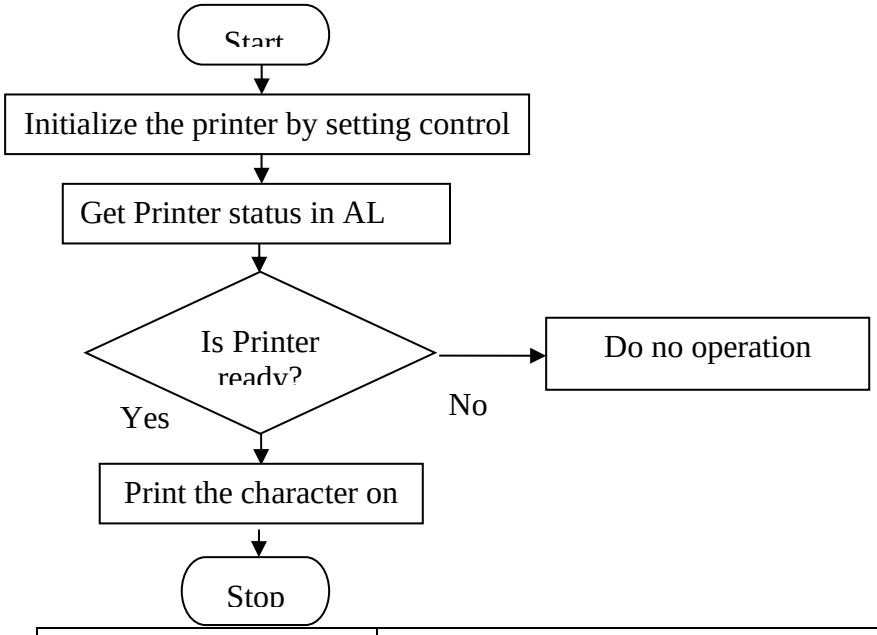
1. Start
2. Initialize the printer by providing appropriate control word
3. Read the printer status to make sure it is ready.
4. Load the character to be printed on to the data register.
5. Stop

BLOCK DIAGRAM:

Centronic Printer Interface Board



FLOWCHART:



PROGRAM	COMMENTS
MOV AL,05H	Strobe and Init pin are initialized for printer to indicate a character is ready for print
OUT 0D0H,AL	Control word sent to control register
IN AL,0C0H	Read the status of printer
AND AL,20H	To check for printer error
CMP AL,20H	Compare 20 and the contents of AL
JNZ ERR	Jump to error routine if error bit is set
MOV AL,41H	Routine to print a character
CALL PRINT	Call Print subroutine
MOV AL,0AH	To print character 'A'
CALL PRINT	Call Print subroutine
HLT	Halt
PRINT: MOV BL,AL	To save temporarily the character to be printed
CALL CHECK	Call Check subroutine
STAS: MOV AL,BL	Get back the character to be printed in AL
OUT 0C8H,AL	The character is sent to data register
MOV AL,01H	Initialize the printer
OUT 0D0H,AL	Control word sent to control register
NOP	No Operation
NOP	
MOV AL,05H	Set strobe and init pin of printer
OUT 0D0H,AL	Control word sent to control register
RET	Return
CHECK: IN AL,0C0H	Read the status of printer
AND AL,20H	To check for printer error
JZ CHECK	
IN AL,0C0H	Read the status of printer
AND AL,80H	To check if printer is busy
CMP AL,80H	Compare 80 with the contents of AL
JNZ STAS	Check and jump
JMP CHECK	
ERR: NOP	
RET	Halt

OUTPUT:

RESULT:

REVIEW QUESTIONS

1. Which interrupt subroutine is used to return printer status?
2. Explain ROL instruction
3. Explain Printer Port.
4. What is meant by Return Instruction?
5. Differentiate CMP and SUB Instructions.

Ex. No.	Setting and Displaying the time in RTC interface board using 8086	Date

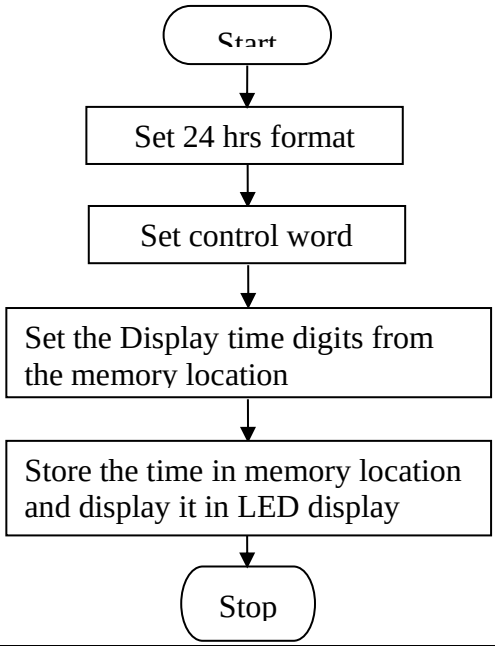
AIM: To write an 8086 ALP to set and display the time in RTC interface board.

APPARATUS REQUIRED: 8086 microprocessor, power supply, RTC interfacing board, interfacing cable.

ALGORITHM:

1. Set 24hours format.
2. Set the control word.
3. Set the display time from the memory location
4. Display the time in LED and store the time in the memory location.

FLOWCHART:



PROGRAM	COMMENTS
MOV AL,05H	Set 24hrs format and RST=1
OUT 0DEH,AL	Control word sent to control register
MOV AL,04H	Set 24hrs format and RST=0
OUT 0DEH,AL	Control word sent to control register
MOV SI,1300H	Read Display time from memory location
MOV AL,[SI]	Move the contents of SI to AL
OUT 0C0H,AL	Set the LSB digit of seconds
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0D0H,AL	Set the MSB digit of seconds

INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0C2H,AL	Set the LSB digit of minutes
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0D2H,AL	Set the MSB digit of minutes
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0C4H,AL	Set the LSB digit of hours
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0D4H,AL	Set the MSB digit of hours
L1: MOV SI,1320H	Memory location to store the time
IN AL,0D4H	Read the MSB digit of hours and store it in memory
AND AL,0FH	Perform logical AND for OF and contents of AL
MOV [SI],AL	Move the contents of AL to SI
IN AL,0C4H	Read the LSB digit of hours and store it in memory
AND AL,0FH	Perform logical AND for OF and contents of AL
INC SI	Increment the content of SI register
MOV [SI],AL	Move the contents of AL to SI
IN AL,0D2H	Read the MSB digit of minutes and store it in memory
AND AL,0FH	Perform logical AND for OF and contents of AL
INC SI	Increment the content of SI register
MOV [SI],AL	Move the contents of AL to SI
IN AL,0C2H	Read the LSB digit of minutes and store it in memory
AND AL,0FH	Perform logical AND for OF and contents of AL
INC SI	Increment the content of SI register
MOV [SI],AL	Move the contents of AL to SI
IN AL,0D0H	Read the MSB digit of seconds and store it in memory
AND AL,0FH	Perform logical AND for OF and contents of AL
INC SI	Increment the content of SI register
MOV [SI],AL	Move the contents of AL to SI
IN AL,0C0H	Read the LSB digit of seconds and store it in memory
AND AL,0FH	Perform logical AND for OF and contents of AL
INC SI	Increment the content of SI register
MOV [SI],AL	Move the contents of AL to SI
OUT_CHECK: MOV SI,1320H	Move the value 1320 to SI
MOV AL,[SI]	Move the contents of SI to AL
OUT 0E0H,AL	Display in first LED display
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0F0H,AL	Display in second LED display
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0E2H,AL	Display in third LED display
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL

OUT 0F2H,AL	Display in fourth LED display
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0E4H,AL	Display in fifth LED display
INC SI	Increment the content of SI register
MOV AL,[SI]	Move the contents of SI to AL
OUT 0F4H,AL	Display in sixth LED display
HLT	

OUTPUT:

RESULT:

REVIEW QUESTIONS

1. What type of RTC kit is used?
2. What is the format of time being displayed?
3. What are the different functionalities of RTC kit?
4. Whether 7 segment display used here is common anode or common cathode type.
5. What are the addresses of hour, minute and second register?

Ex. No.	Interfacing Analog To Digital Converter using 8086	Date

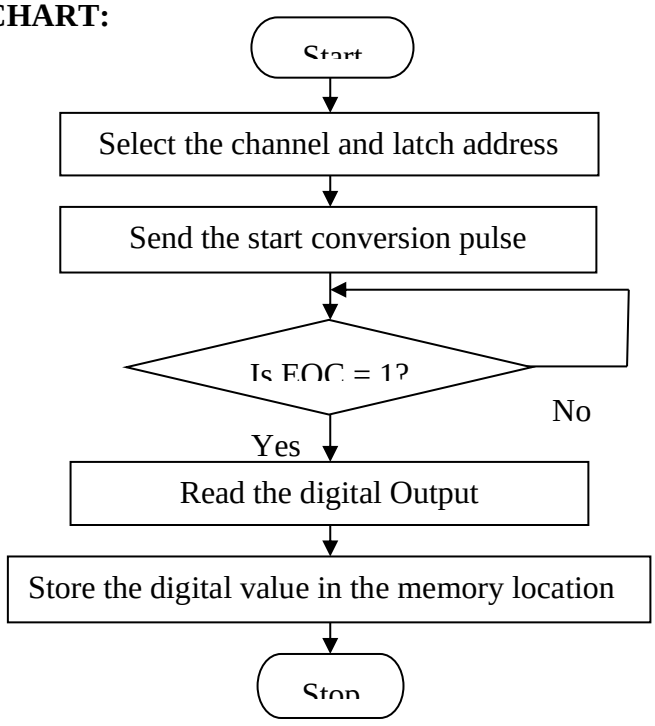
AIM: To write an assembly language program to convert an analog signal into a digital signal using an Analog to Digital Converter interfacing.

APPARATUS REQUIRED:8086 Microprocessor Kit, ADC0809 Interface Board, Digital Multimeter & Power Supply.

ALGORITHM:

(i) Select the channel and latch the address.
(ii) Send the start conversion pulse.
(iii) Read EOC signal.
(iv) If EOC = 1 continue else go to step (iii)
(v) Read the digital output.
(vi) Store it in a memory location.

FLOW CHART:



PROGRAM	COMMENTS
MOV AL, 00	Load accumulator with value for ALE high
OUT 0C8, AL	Send through output port
MOV AL, 08	Load accumulator with value for ALE low
OUT 0C8, AL	Send through output port
MOV AL, 01	Store the value to make SOC high in the accumulator
OUT 0D0, AL	Send through output port
MOV AL, 00	Introduce delay
MOV AL, 00	
MOV AL, 00	
MOV AL, 00	Store the value to make SOC low the accumulator
OUT 0D0, AL	Send through output port
L1 : IN AL, 0D8	Read the EOC signal from port & check for end of conversion
AND AL, 01	
CMP AL, 01	
JNZ L1	If the conversion is not yet completed, read EOC signal from port again
IN AL, 0C0	Read data from port
MOV BX, 1100	Initialize the memory location to store data
MOV [BX], AL	Store the data
HLT	Stop

OUTPUT:

Analog voltage	Digital Data on LED Display	Hex Code in Memory Location 1100

RESULT:

REVIEW QUESTIONS:

1. Classify ADC.
2. What are the control lines of ADC?
3. What do you mean by ALE?
4. What is the function of SOC?
5. What is the function of EOC?

Ex. No	Interfacing Digital To Analog Converter using 8086.	Date

AIM: To write an assembly language program for digital to analog conversion, to convert digital inputs into analog outputs & to generate different waveforms.

APPARATUS REQUIRED: 8086 Microprocessor Kit, DAC0800 Interface Board, CRO.

ALGORITHM:

MEASUREMENT OF ANALOG VOLTAGE:

(i) Send the digital value of DAC.
(ii) Read the corresponding analog value of its output.

WAVEFORM GENERATION:

Square Waveform:

(i) Send low value (00) to the DAC.
(ii) Introduce suitable delay.
(iii) Send high value to DAC.
(iv) Introduce delay.
(v) Repeat the above procedure.

Saw-tooth waveform:

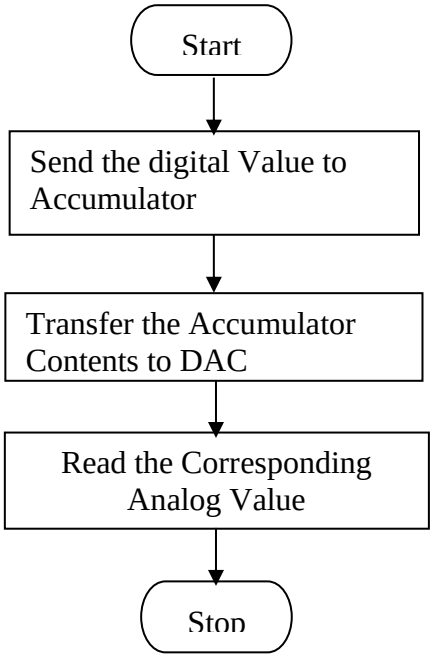
(i) Load low value (00) to accumulator.
(ii) Send this value to DAC.
(iii) Increment the accumulator.
(iv) Repeat step (ii) and (iii) until accumulator value reaches FF.
(v) Repeat the above procedure from step 1.

Triangular waveform:

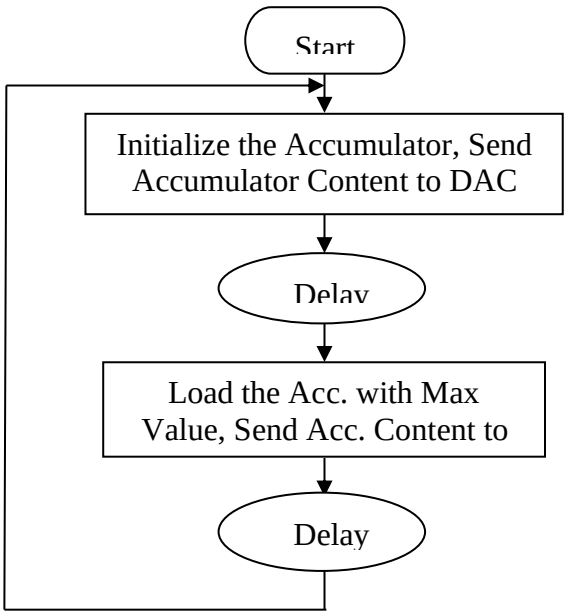
(i) Load the low value (00) in accumulator.
(ii) Send this accumulator content to DAC.
(iii) Increment the accumulator.
(iv) Repeat step 2 and 3 until the accumulator reaches FF, decrement the accumulator and send the accumulator contents to DAC.
(v) Decrementing and sending the accumulator contents to DAC.
(vi) The above procedure is repeated from step (i)

FLOW CHART

MEASUREMENT OF ANALOG VOLTAGE



SQUARE WAVE FORM



PROGRAM: ***Square Wave***

PROGRAM	COMMENTS
L2 : MOV AL, 00H	Load 00 in accumulator
OUT C0, AL	Send through output port
CALL L1	Give a delay
MOV AL, 0FF	Load FF in accumulator
OUT C0, AL	Send through output port
CALL L1	Give a delay
JMP L2	Go to starting location
L1 : MOV CX, 05FF	Load count value in CX register
L3 : LOOP L3	Decrement until it reaches zero
RET	Return to main program

PROGRAM: ***Saw tooth Wave***

PROGRAM	COMMENTS
L2 : MOV AL, 00H	Load 00 in accumulator
L1 : OUT C0, AL	Send through output port
INC AL	Increment contents of accumulator
JNZ L1	Send through output port until it reaches FF
JMP L2	Go to starting location

PROGRAM: ***Triangular Wave***

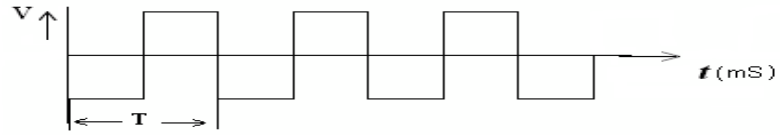
PROGRAM	COMMENTS
L3 : MOV AL, 00H	Load 00 in accumulator
L1 : OUT C0, AL	Send through output port
INC AL	Increment contents of accumulator
JNZ L1	Send through output port until it reaches FF
MOV AL, 0FF	Load FF in accumulator
L2 : OUT C0, AL	Send through output port
DEC AL	Decrement contents of accumulator
JNZ L2	Send through output port until it reaches 00
JMP L3	Go to starting location

MEASUREMENT

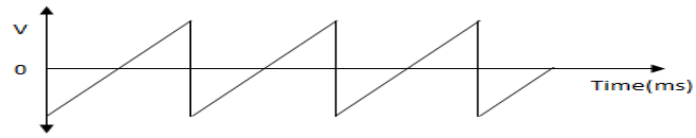
DIGITAL DATA	ANALOG VOLTAGE
00	-5V
7F	0V
FF	+5V

MODEL GRAPH:

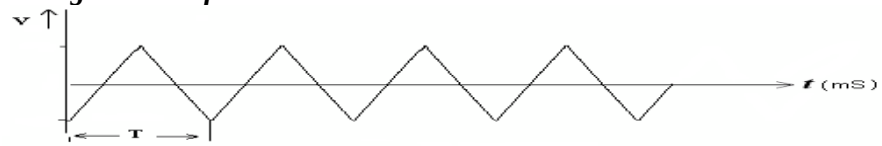
Square Waveform*:*



Saw-tooth waveform:



Triangular waveform:



OUTPUT:

WAVE FORMS	AMPLITUDE	TIME PERIOD
Square Waveform		
Saw-tooth waveform		
Triangular waveform		

RESULT:

REVIEW QUESTIONS:

1. What are the 4 commands used for interfacing?
2. Which controller monitors data transfer?
3. Which monitors address line?
4. What is the use of peripheral controller in I/O interface?
5. Explain the need for ADC.

Ex. No	Interfacing Timer-8253 using 8086	Date

AIM: To study different modes of operation of programmable timer 8253.

APPARATUS REQUIRED: 8086 Microprocessor kit, Power supply, 8253 Interfacing board & CRO.

ALGORITHM:

Mode 2-*Rate Generator*

1. Initialize channel 0 in mode 2
2. Initialize the LSB of the count.
3. Initialize the MSB of the count.
4. Trigger the count
5. Read the corresponding output in CRO.

Mode 3-*Square Wave Generator*

1. Initialize channel 0 in mode 3
2. Initialize the LSB of the count.
3. Initialize the MSB of the count.
4. Trigger the count
5. Read the corresponding output in CRO.

CONTROL WORD FORMAT:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	**SC0**	**RL1**	**RL0**	**M2**	**M1**	**M0**	**BCD**
0	0	1	1	0	1	0	0
0	0	1	1	0	1	1	0

Mode 2 = 34 H

Mode 3 = 36 H

SC1	SC0	CHANNEL SELECT	RL1	RL0	READ/LOAD
0	0	CHANNEL 0	0	0	LATCH
0	1	CHANNEL 1	0	1	LSB
1	0	CHANNEL 2	1	0	MSB
1	1	-----	1	1	LSB FIRST, MSB NEXT

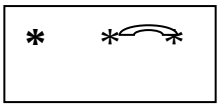
BCD --0 –BINARY COUNTER 1 --BCD COUNTER

M2	M1	M0	MODE
0	0	0	MODE 0
0	0	1	MODE 1
0	1	0	MODE 2
0	1	1	MODE 3
1	0	0	MODE 4
1	0	1	MODE 5

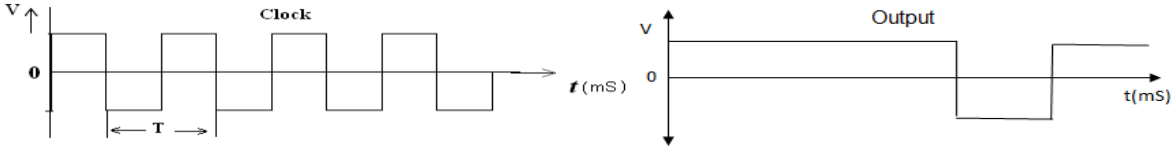
PORT PIN ARRANGEMENT

1 CLK 0
2 CLK 1
3 CLK 2
4 OUT 0
5 OUT 1
6 OUT 2
7 GATE 0
8 GATE 1
9 GATE 2
10 GND

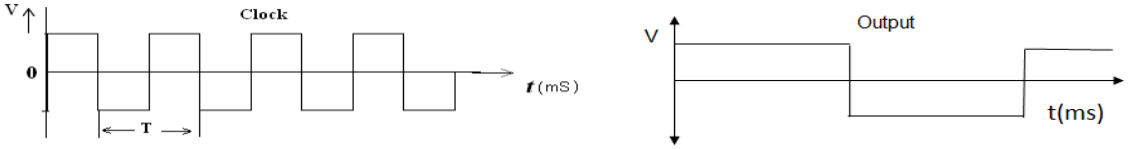
DEBOUNCE CIRCUIT CONNECTION



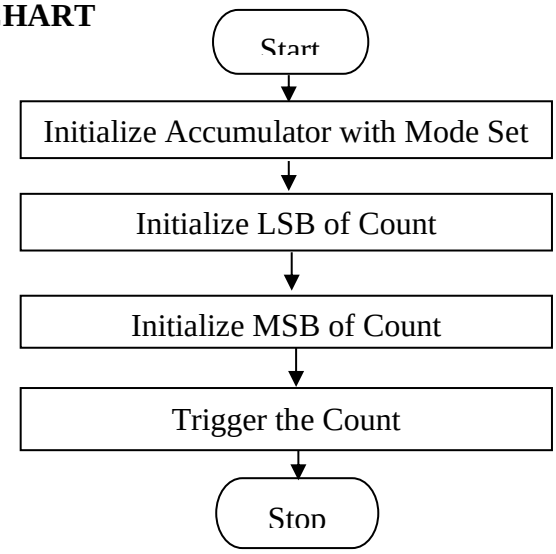
MODEL GRAPH: *RATE GENERATOR* (N-1)



SQUARE WAVE GENERATOR (N/2)



FLOW CHART



PROGRAM:

MODE 2 – RATE GENERATOR

PROGRAM	COMMENTS
MOV AL, 74H	Store the control word in accumulator
OUT 0CE,AL	Send through output port
MOV AL, 0AH	Copy lower order count value in accumulator
OUT 0CA,AL	Send through output port
MOV AL, 00H	Copy higher order count value in accumulator
OUT 0CA,AL	Send through output port
HLT	Stop

MODE 3 – SQUARE WAVE GENERATOR

PROGRAM	COMMENTS
MOV AL, 36H	Store the control word in accumulator
OUT 0CE,AL	Send through output port
MOV AL, 10	Copy lower order count value in accumulator
OUT 0C8,AL	Send through output port
MOV AL, 00H	Copy higher order count value in accumulator
OUT 0C8,AL	Send through output port
HLT	Stop

OUTPUT:

WAVE FORMS	AMPLITUDE	TIME PERIOD
Clock Signal		
Rate Generator		
Square Wave Generator		

RESULT:

REVIEW QUESTIONS:

1. What is use of stack segment and extra segment?
2. What is the use of flags register in 8086?
3. What is the use of index register in 8086?
4. What is the use of SCASB instruction?
5. Difference between REP and REPNE instruction.

Ex. No.	Interfacing USART-8251 using 8086	Date

AIM: To study interfacing technique of 8251 (USART) with microprocessor 8086 and write an ALP to Transmit and Receive data between two serial ports with RS232 cable.

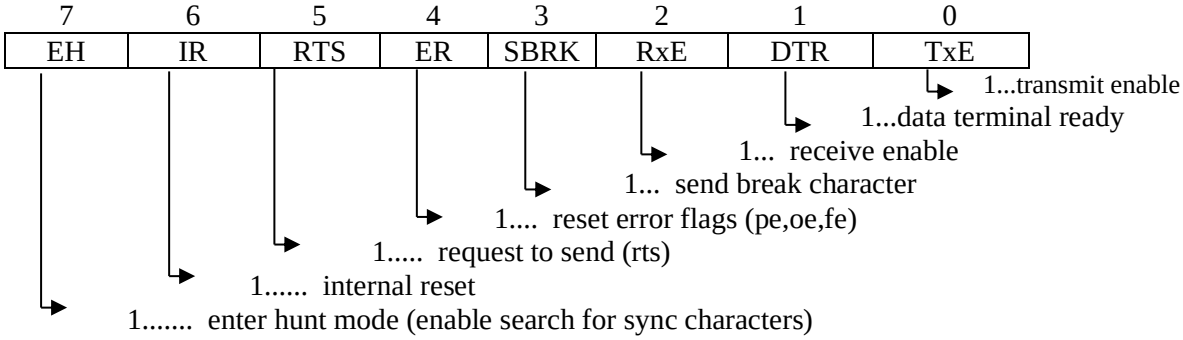
APPARATUS REQUIRED:8086 kit (2 Nos.), Power Supply & RS232 cable.

ALGORITHM:

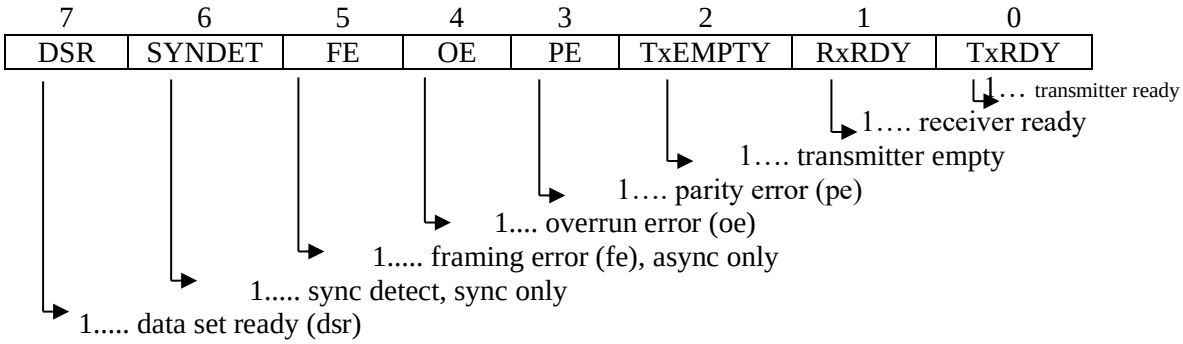
1. Initialize 8253 and 8251 to check the transmission and reception of a character
2. Initialize8253 to give an output of 150Khz at channel 0 which will give a 9600 baud rate of 8251.
3. The command word and mode word is written to the 8251 to set up for subsequent operations

The status word is read from the 8251 on completion of a serial I/O operation, or when the host CPU is checking the status of the device before starting the next I/O operation

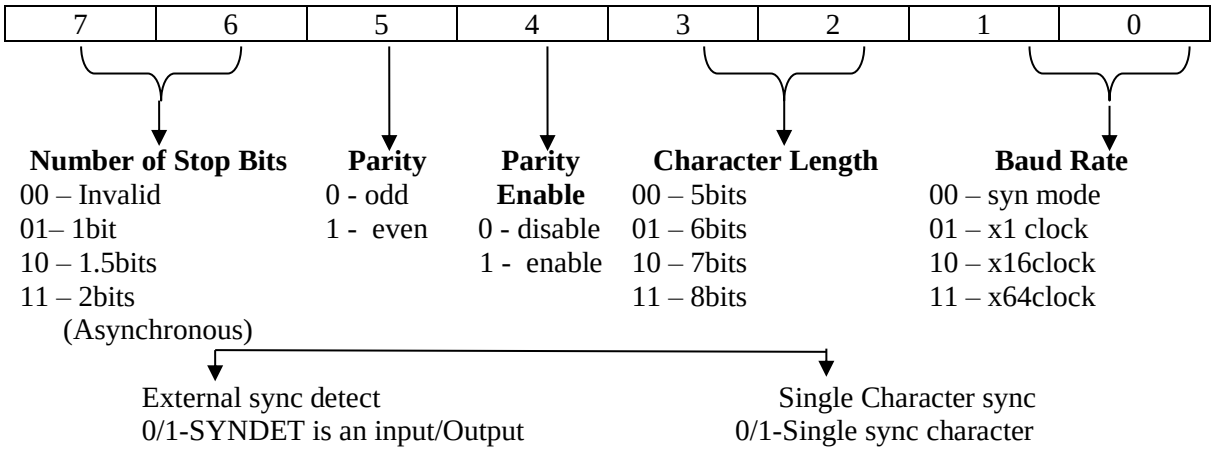
COMMAND INSTRUCTION



Status Byte



Mode Instruction



PROGRAM:*TRANSMITTER*

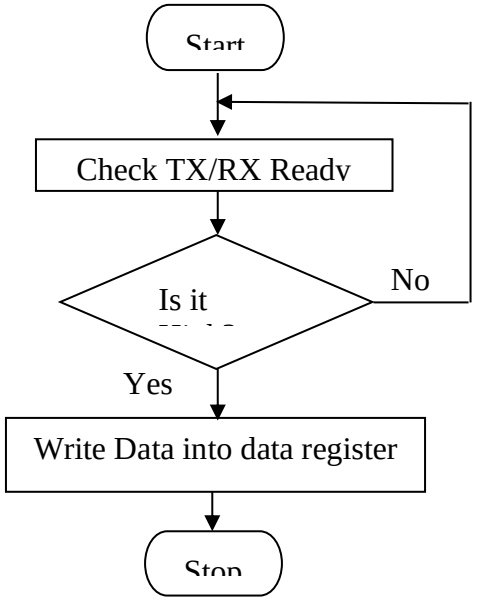
PROGRAM	COMMENTS
MOV SI, 1500	Initialize memory location for array size
MOV AL, 36H	Setting the command word to get required baud rate
OUT 16H, AL	
MOV AL, 40H	Setting the command word for synchronization

OUT 10H, AL	
MOV AL, 01H	Enabling transmitter & receiver
OUT 10H, AL	
RELOAD:MOV CL, 05H	Set count
CHECK: IN AL, OAH	Check for transmitter ready
AND AL, 04H	
JZ CHECK	
MOV AL, [SI]	Sending data
OUT 08H, AL	
INC SI	Checking for end of transmission
CMP AL, 3FH	
JNZ RELOAD	
DEC CL	
JNZ CHECK	
INT 02	Halt

RECEIVER

PROGRAM	**COMMENTS**
MOV SI,1500	Initialize memory location for array size
MOV AL, 36H	Setting the command word to get required baud rate
OUT 16H, AL	
MOV AL, 40H	Setting the command word for synchronization
OUT 10H, AL	
MOV AL, 01H	Enabling transmitter & receiver
OUT 10H, AL	
RELOAD:MOV CL, 05H	Set count
CHECK: IN AL, OAH	Check for receiver ready
AND AL, 02H	
JZ CHECK	
IN AL, 08H	Receiving data
MOV [SI], AL	
INC SI	Checking for end of reception
CMP AL, 3FH	
JNZ RELOAD	
DEC CL	Checking the count
JNZ CHECK	
INT 02	Halt

FLOW CHART



OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is use of stack segment and extra segment?
2. What is the use of flags register in 8086?
3. What is the use of index register in 8086?
4. What is the use of SCASB instruction?
5. Difference between REP and REPNE instruction.

Ex. No.	8 bit Addition & Subtraction using 8051	Date

AIM: To write an ALP to perform 8 bit addition/subtraction in 8051.

APPARATUS REQUIRED: 8051 microcontroller kit, power supply.

ALGORITHM:

8 BIT ADDITION:

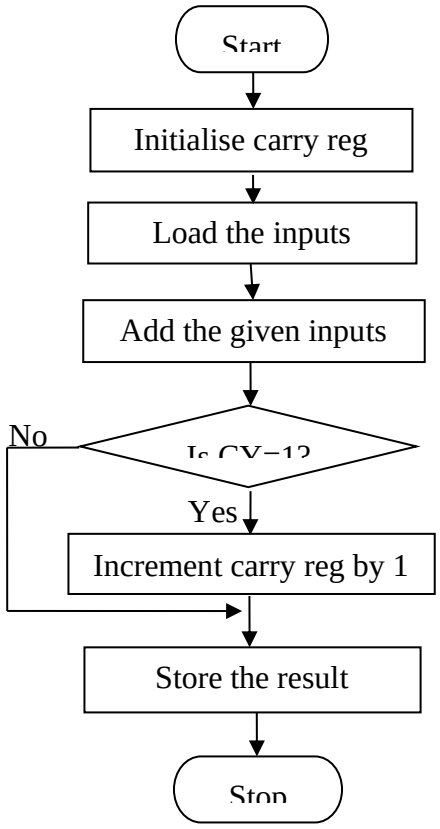
1. Clear the program status word.
2. Load the first number in the accumulator.
3. Load the second reg. in the register R_0.
4. Load the destination address in the DPTR.
5. Add the 2 numbers and Store the sum and carry in the destination address.
7. Terminate the program.

8 BIT SUBTRACTION:

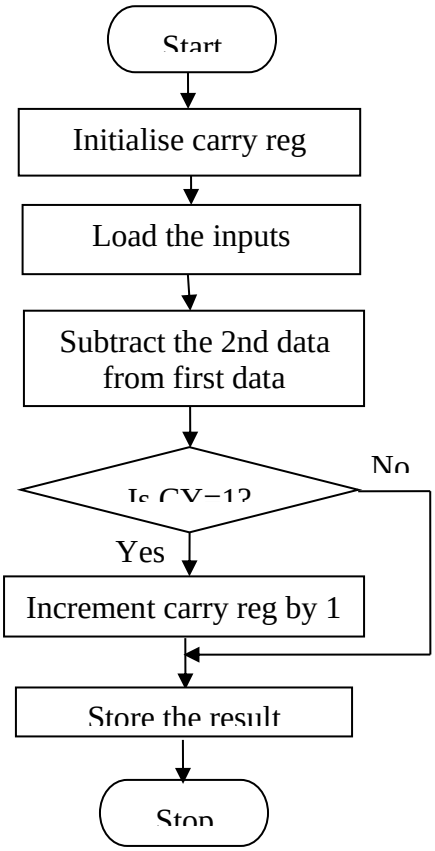
1. Clear PSW.
2. Select the register by giving proper values.
3. Load the accumulator with 1st data & reg with 2nd data.
4. Subtract 2nd data from 1st data.
5. Store the diff and borrow and terminate the program.

FLOWCHART

ADDITION



SUBTRACTION



8 BIT ADDITION (IMMEDIATE ADDRESSING)

ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		CLR	C		Clear the PSW
		MOV	A,# data1		Load 1st number in the accumulator
		ADDC	A,# data2		Add the two numbers
		MOV	DPTR,#4500		Load destination address in the DPTR
		MOVX	@ DPTR, A		Store sum in destination address
	L1	SJMP	L1		Terminate the program

8 BIT SUBTRACTION (IMMEDIATE ADDRESSING)

ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		CLR	C		Clear the PSW
		MOV	A,# data1		Load 1st number in accumulator
		SUBB	A,# data2		Subtract data1 from data2
		MOV	DPTR,#4500		Load destination address in DPTR
		MOVX	@ DPTR, A		Store difference
	L1	SJMP	L1		Terminate the program

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1.What is DPTR?
2.What is the difference between Microprocessor and Microcontroller?
3.What is the use of stack pointer?
4. What is the use of SJMP?
5. What is meant by Register Bank?

Ex. No.	8 bit multiplication & 8 bit division using 8051	Date

AIM: To write a program for 8 bit multiplication & 8 bit division using 8051 microcontroller.

APPARATUS REQUIRED: 8051 microcontroller kit, power supply.

ALGORITHM:

8 BIT MULTIPLICATION:

1. Clear PSW.
2. Select register bank by giving proper values.
3. Load accumulator A with any derived 8 bit data.
4. Load registers B with 2nd data.
5. Multiply there 2 nods.
6. Store the result and Terminate the program.

8 BIT DIVISION:

1. Clear PSW.
2. Select register bank by giving proper values.
3. Load A with 1st data dividend.
4. Load B wit 8 bit divisor.
5. Divide A/B.
6. Store the quotient & remainder and terminate the program.

8 BIT MULTIPLICATION

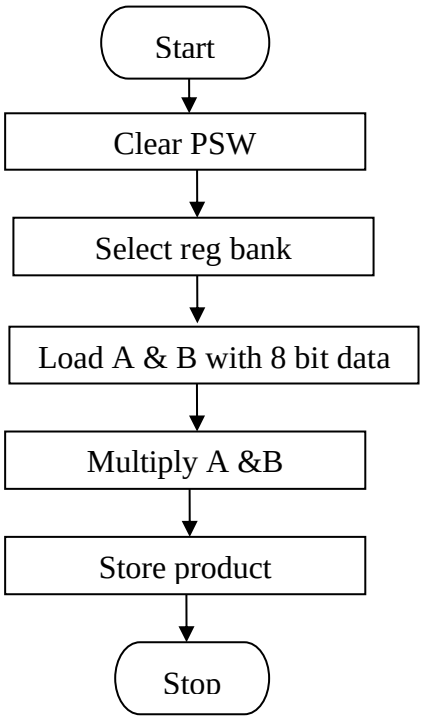
ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		MOV	A, #data1		Load A register with data1
		MOV	B, #data2		Load B register with data2
		MUL	AB		Multiply A &B
		MOV	DPTR, # 4500H		Initialize destination address
		MOVX	@ DPTR, A		Store lower order product
		INC	DPTR		Increment DPTR
		MOV	A,B		Move higher order product to A
		MOVX	@ DPTR, A		Store higher order product
	STOP	SJMP	STOP		Terminate the program

8 BIT DIVISION

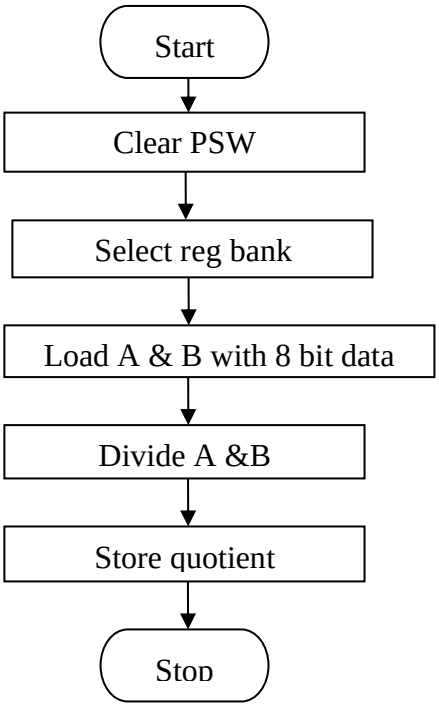
ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		MOV	A, #data1		Load A register with data1
		MOV	B, #data2		Load B register with data2
		DIV	AB		Divide A &B
		MOV	DPTR, # 4500H		Initialize destination address
		MOVX	@ DPTR, A		Store quotient
		INC	DPTR		Increment the data pointer
		MOV	A,B		Move remainder to reg A
		MOV	@ DPTR, A		Store remainder
	STOP	SJMP	STOP		Terminate the program

FLOWCHART:

Multiplication



Division



OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is Immediate addressing Mode?
2. What is the difference between Microprocessor and Microcontroller?
3. What is the use of Accumulator?
4. What is the use of Interrupt?
5. What is meant by Port?

Ex. No.	Logical operation using 8051 microcontroller	Date

AIM: To write a program for performing logical operation using 8051 microcontroller.

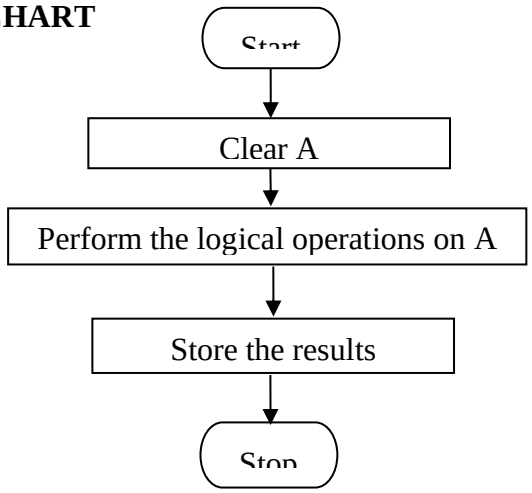
APPARATUS REQUIRED: 8051 microcontroller kit, power supply.

ALGORITHM:

1. Clear accumulator A.
2. Perform logical operations such as clear, set, rotate, swap and logical AND on contents of A.
3. Store the results and terminate the program.

ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		MOV	DPTR, # 4500H		Initialize input data address
		CLR	A		Clear accumulator
		MOVX	@ DPTR, A		Store Acc content
		MOV	A,PSW		Move PSW to Acc
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR, A		Store PSW
		CLR	PSW.6		Clear the AC flag
		SETB	PSW.7		Set bit C flag
		MOV	A,PSW		Move PSW to Acc
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR, A		Store PSW
		RLC	A		Rotate Accumulator with carry
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR, A		Store reg A
		ANL	C,ACC.7		Logical AND
		MOV	A,PSW		Move PSW to Acc
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR, A		Store PSW
		SWAP	A		Swap upper and lower nibble of A
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR, A		Store reg A
	STOP	SJMP	STOP		Terminate the program

FLOWCHART



OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is the use of MOV instruction?
2. How you will compare two strings in 8051?
3. What is the use of DPTR?
4. Explain ANL Instruction.
5. Explain Swap Instruction.

Ex. No.	Finding square of an 8 bit number using 8051	Date

AIM: To write a program for finding square of an 8 bit number using 8051 microcontroller.

APPARATUS REQUIRED: 8051 microcontroller kit, power supply.

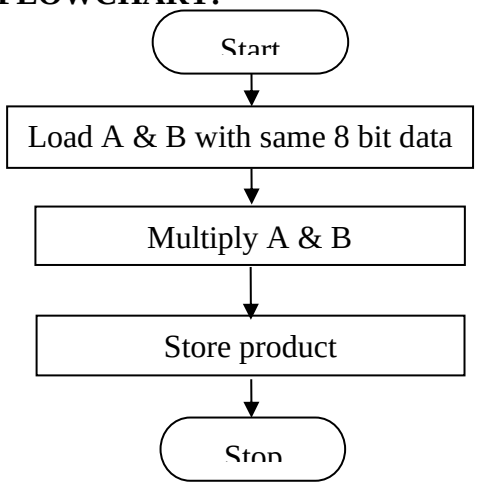
ALGORITHM:

1. Load accumulator A with any desired 8 bit data.
4. Load registers B with the same data.
5. Multiply the contents of A and B registers.
6. Store the result and terminate the program.

PROGRAM:

ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		MOV	DPTR, # 4500H		Initialize input data address
		MOVX	A, @ DPTR		Load accumulator A with any desired 8 bit data
		MOV	B,A		Load registers B with the same data
		MUL	AB		Multiply A & B register contents
		INC	DPTR		Increment DPTR
		MOV	R0,A		Move lower product to R0
		MOV	A,B		Move higher product to A
		MOVX	@ DPTR, A		Store higher product
		INC	DPTR		Increment DP
		MOV	A,R0		Move content of R0 to A
		MOVX	@ DPTR, A		Store lower product
	STOP	SJMP	STOP		Terminate the program

FLOWCHART:



OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. Which are the different addressing modes of 8051?
2. Where are the results of MUL AB stored?
3. What is the use of MOVX instruction?
4. What is meant by watch dog timer?
5. What is Accumulator?

Ex. No.	Finding cube of an 8 bit number using 8051	Date

AIM: To write a program for finding cube of an 8 bit number using 8051 microcontroller.

APPARATUS REQUIRED: 8051 microcontroller kit, power supply.

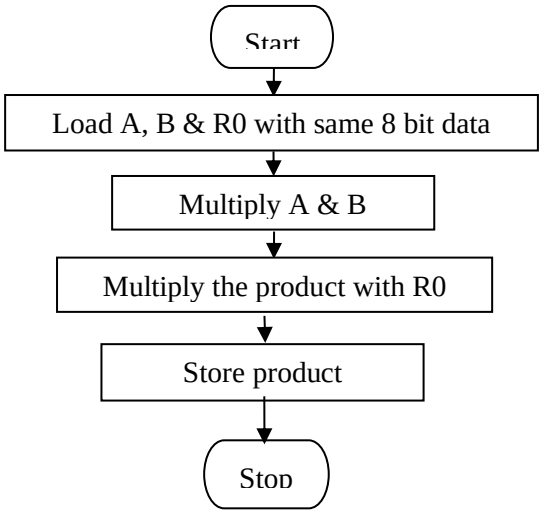
ALGORITHM:

1. Load accumulator A with any desired 8 bit data.
2. Load registers B and R0 with the same data.
3. Multiply the contents of A and B registers.
4. Multiply the lower and the higher byte of the above result separately with contents of R0.
5. Add the above results appropriately.
6. Store the result and terminate the program.

PROGRAM:

ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		MOV	DPTR, # 4500H		Initialize input data address
		MOVX	A, @ DPTR		Load accumulator A with any desired 8 bit data
		MOV	R0,A		Load register B with the same data
		MOV	B,A		Load register R0 with the same data
		MUL	AB		Multiply the contents of A and B registers
		PUSH	B		PUSH content of B
		MOV	B,A		Move content of A to B
		MOV	A,R0		Move R0 to A
		MUL	AB		Multiply the contents of A and B registers
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR, A		Store Acc
		MOV	R2,B		Move B to R2
		POP	B		POP content of B
		MOV	A,R0		Move R0 to A
		MUL	AB		Multiply the contents of A and B registers
		ADD	A,R2		Add A and R2 contents
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR, A		Store Acc
		INC	DPTR		Increment DPTR
		MOV	A,B		Move B to A
		MOVX	@ DPTR, A		Store the product
	STOP	SJMP	STOP		Terminate the program

FLOWCHART:



OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. Which are the different addressing modes of 8051?
2. Where are the results of MUL AB stored?
3. What is the use of MOVX instruction?
4. What is meant by watch dog timer?
5. What is Accumulator?

Ex. No.	Finding 2's complement of an 8 bit number using 8051	Date

AIM: To write a program for finding 2's complement of an 8 bit number using 8051 microcontroller.

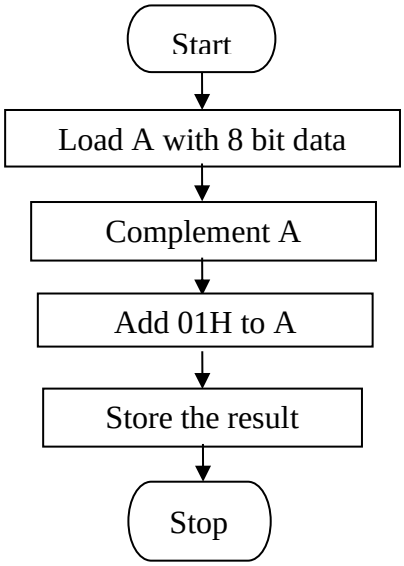
APPARATUS REQUIRED:

8051 microcontroller kit, power supply.

ALGORITHM:

1. Load accumulator A with any desired 8 bit data.
2. Complement the contents of A.
3. Add 01H to the contents of A.
4. Store the result and terminate the program

FLOWCHART:



PROGRAM:

ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		MOV	DPTR, # 4500H		Initialize input data address
		MOVX	A, @ DPTR		Load accumulator with data
		CPL	A		Complement the contents of A
		ADD	A,#01H		Add 01H to the contents of A
		INC	DPTR		Increment DPTR
		MOVX	@ DPTR,A		Store the result
	HERE:	SJMP	HERE		Terminate the program

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is meant by CPL instruction?
2. What is the difference between SJMP and LJMP?
3. What is the use of MOVX instruction?
4. What is meant by watch dog timer?
5. What is Accumulator?

Ex. No.	Unpacked BCD number to ASCII number using 8051	Date

AIM: To write a program for unpacked BCD number to ASCII number using 8051 microcontroller.

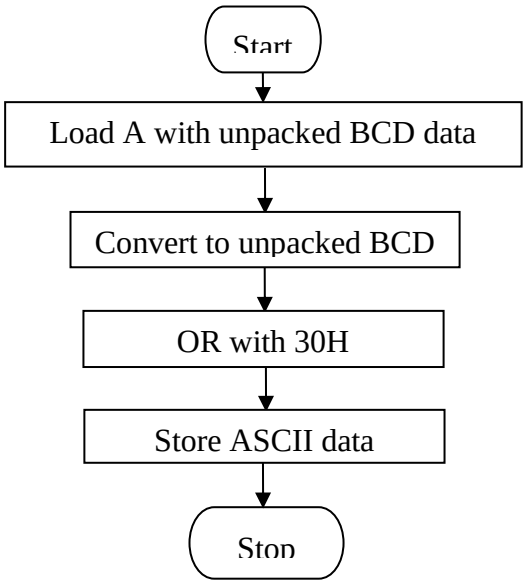
APPARATUS REQUIRED:

8051 microcontroller kit, power supply.

ALGORITHM:

1. Load accumulator A with any desired 8 bit data.
2. Load registers B and R0 with the same data.
3. Multiply the contents of A and B registers.
4. Multiply the lower and the higher byte of the above result separately with contents of R0.
5. Add the above results appropriately.
6. Store the result and terminate the program.

FLOWCHART:



PROGRAM:

ADDRESS	LABEL	MNEMONICS	OPERAND	HEXCODE	COMMENTS
		MOV	DPTR, # 4500H		Initialize input data address
		MOVX	A, @ DPTR		Load accumulator A with 1st unpacked BCD digit
		MOV	R0,A		Copy Acc to R0
		INC	DPTR		Increment DPTR
		MOVX	A, @ DPTR		Load accumulator A with 2nd[t] unpacked BCD digit
		ORL	A,#30H		Logical OR with 30H
		INC	DPTR		Increment DPTR
		MOVX	@DPTR,A		Store 2nd ASCII digit
		MOV	A,R0		Move R0 to A
		ORL	A,#30H		Logical OR with 30H
		INC	DPTR		Increment DPTR
		MOVX	@DPTR,A		Store 2nd ASCII data
	STOP	SJMP	STOP		Terminate the program

OUTPUT:

RESULT:

REVIEW QUESTIONS:

1. What is meant by 'packed BCD' number?
2. Differentiate between ANL and ORL instruction.
3. What does SWAP A instruction do?
4. Differentiate between Packed and Unpacked BCD numbers.
5. What is register addressing Mode?

Ex. No.	STUDY OF ARM PROCESSOR	Date

AIM: To Study about the architecture of ARM processor (ARM CORTEX LPC4088).

THEORY:

Features of LPC4088 ARM Processor:

Processor — NXP's Cortex-M4 LPC4088 microcontroller in BGA 144 pin package, running at up to 120 MHz.

Memory

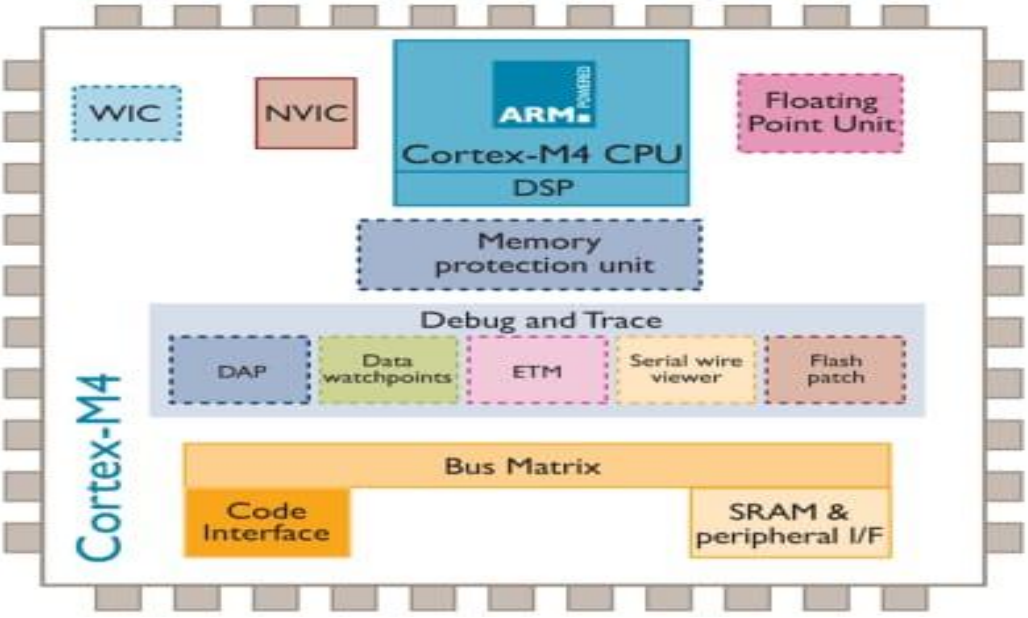
- 512 KByte on-chip FLASH
- 96 KByte on-chip SRAM
- 4 KByte on-chip E2PROM
- 8 MByte QSPI FLASH (can execute program code and/or contain a file system)
- 32 MByte SDRAM with 32-bit databus access
- On-board globally unique MAC address

Clock Crystals — 12.000 MHz main and 32.768 kHz RTC crystals

Interfaces / Connectors — • 2x22 pin edge pins
• Up to 165 General Purpose I/O (GPIO) pins depending on the packaging, with configurable pull-up/down resistors, open-drain mode,

	and repeater mode. • 10/100Mbps Ethernet (RJ45) • USB-A (USB Host interface) • USB-micro B (USB Device interface) • USB-micro B (mbed HDG debug interface) • 20 position SDW/Trace connector (ARM standard debug connector) • 61 pos 0.3 mm pitch FPC connector for display expansion • 20 pos XBee compatible connector for RF module add-on
Power	• 4.5 - 5.5V input on pin 2, or • via micro-B USB HDK connector, or • via trace connector (+5V).
Other	• Proper ESD protection on communication interfaces • Real-Time Clock (RTC) with a separate power domain. • 12-bit Analog-to-Digital Converter (ADC) with input multiplexing among eight pins, conversion rates up to 400 kHz, and multiple result registers. • 10-bit Digital-to-Analog Converter (DAC) with dedicated conversion timer and DMA support. • CMSIS-DAP Interface On-board (debug interface functions). • Industrial temperature specified (-40 to +85 degrees Celsius).
Target Applications	• Communications:Point-of-sale terminals, web servers, multi-protocol bridges • Industrial/Medical:Automation controllers, application control, robotics control, HVAC, PLC, inverters, circuit breakers, medical scanning, security monitoring, motor drive. • Consumer/Appliance:Audio, MP3 decoders, alarm systems, displays, printers, scanners, small appliances, fitness equipment .

ARCHITECTURE OF ARM CORTEX M4



WIC(Wakeup Interrupt Controller):

The WIC is an optional peripheral that can detect an interrupt and wake the processor from deep sleep mode. The WIC is not programmable, and does not have any registers or user interface. It operates entirely from hardware signals.

NVIC (Nested Vectored Interrupt Controller):

The NVIC supports up to 240 interrupts, each with up to 256 levels of priority that can be changed

dynamically. The processor and NVIC can be put into a very low-power sleep mode, leaving the Wake Up Controller (WIC) to identify and prioritize interrupts.

Floating Point Unit(FPU):
The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

Memory Protection Unit (MPU):
The MPU divides the memory map into a number of regions, and defines the location, size, access permissions, and memory attributes of each region. It supports:

- independent attribute settings for each region
- overlapping regions
- export of memory attributes to the system.

Debug and Trace:
Debug features:

- Run Control of the processor allowing you to start and stop programs
- Single Step one source or assembler line
- Set breakpoints while the processor is running
- Read/write memory contents and peripheral registers on-the-fly
- Program internal and external FLASH memory

Trace features:

- Serial Wire Viewer (SWV) provides program counter (PC) sampling, data trace, event trace, and instrumentation trace information
- Instruction (ETM) Trace streamed directly to your PC enabling debugging of historical sequences, software profiling, and code coverage analysis.

Bus Matrix:
The bus matrix connects the processor and debug interface to the external buses. The bus matrix interfaces to the following external buses:

- ICode bus. This is for instruction and vector fetches from code space. This is a 32-bit AHB-Lite bus.
- DCode bus. This is for data load/stores and debug accesses to code space. This is a 32-bit AHB-Lite bus.
- System bus. This is for instruction and vector fetches, data load/stores and debug accesses to system space. This is a 32-bit AHB-Lite bus.
- PPB. This is for data load/stores and debug accesses to PPB space. This is a 32-bit APB (v2.0) bus.

RESULT:

REVIEW QUESTIONS:
1. What is meant by ARM processor?
2. What does LPC stands for?
3. What are the features of LPC4088?
4. What are the applications of ARM processor?
5. List out the cortex-M series processors.