Terna Engineering College
Computer Engineering Department

Program: Sem V

Course: Microprocessor Lab

Faculty: ARATHI BOYANAPALLI

LAB Manual

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No. 1

A.1 Aim:
Write assembly language program implement basic arithmetic operations on two 8 bit and 16-bit numbers

A.2 Prerequisite:
Basic knowledge of Digital Electronics, Computer Organization

A.3 Outcome:
After successful completion of this experiment, students will be able to

1. Use appropriate instructions to program microprocessors to perform various tasks.
2. Develop the program in assembly/ mixed language for Intel 8086 processor
3. Demonstrate the execution and debugging of assembly/mixed language program

A.4 Theory:
The following instruction may be used for implementation & its formats are as follows:

ADD

Syntax: add dest, src
dest: register or memory
src: register, memory, or immediate
Action: dest = dest + src
Flags Affected: OF, SF, ZF, AF, PF, CF
Notes: Works for both signed and unsigned numbers.

For Example: AX=1234H, BX=0100H

```
  1234H +
  0100H
 ------
  1334H
```

SUB Subtract two numbers
Syntax: sub dest, src
dest: register or memory
src: register, memory, or immediate
Action: dest = dest - src
Flags Affected: OF, SF, ZF, AF, PF, CF
Notes: Works for both signed and unsigned numbers.

MUL Unsigned multiply
Syntax: mulop8
mulop16
op8: 8-bit register or memory
op16: 16-bit register or memory
Action: If operand is op8, unsigned AX = AL * op8
If operand is op16, unsigned DX::AX = AX * op16
Flags Affected: OF, SF=?, ZF=?, AF=?, PF=?, CF

DIV Unsigned divide
Syntax: divop8
divop16
op8: 8-bit register or memory
op16: 16-bit register or memory
Action: If operand is op8, unsigned AL = AX / op8 and AH = AX % op8
If operand is op16, unsigned AX = DX::AX / op16 and DX = DX::AX % op16
Flags Affected: OF=?, SF=?, ZF=?, AF=?, PF=?, CF=?
Notes: Performs both division and modulus operations in one instruction.

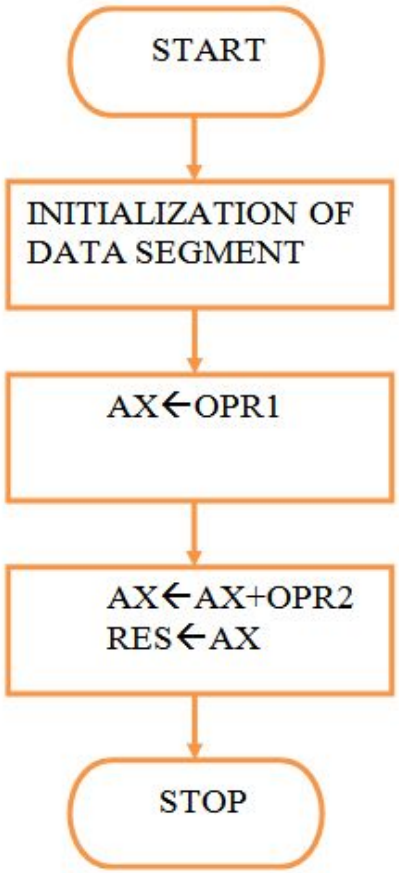
IDIV Signed divide
Syntax: idivop8
idivop16
op8: 8-bit register or memory
op16: 16-bit register or memory
Action: If operand is op8, signed AL = AX / op8 and AH = AX % op8
If operand is op16, signed AX = DX::AX / op16 and DX = DX::AX % op16
Flags Affected: OF=?, SF=?, ZF=?, AF=?, PF=?, CF=?
Notes: Performs both division and modulus operations in one instruction.

IMUL Signed multiply
Syntax: imulop8
imulop16
op8: 8-bit register or memory
op16: 16-bit register or memory
Action: If operand is op8, signed AX = AL * op8
If operand is op16, signed DX::AX = AX * op16
Flags Affected: OF, SF=?, ZF=?, AF=?, PF=?, CF

Algorithm :

Step I : Initialize the data segment.
Step II : Get the first number in the AX register.
Step III : Get the second number in the BX register.
Step IV : perform arithmetic operations on two numbers.
Step V : Display the AX/DXresult.
Step VI : Stop

Flowchart :



PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the ERP or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no ERP access available)

Roll No. : 50	Name: Amey Thakur
Class: TE-Comps B	Batch: B3
Date of Experiment: 03/08/2020	Date of Submission: 03/08/2020
Grade:	

B.1 Software Code written by a student:

(Paste your code completed during the 2 hours of practical in the lab here)

Refer B.2

B.2 Input and Output:

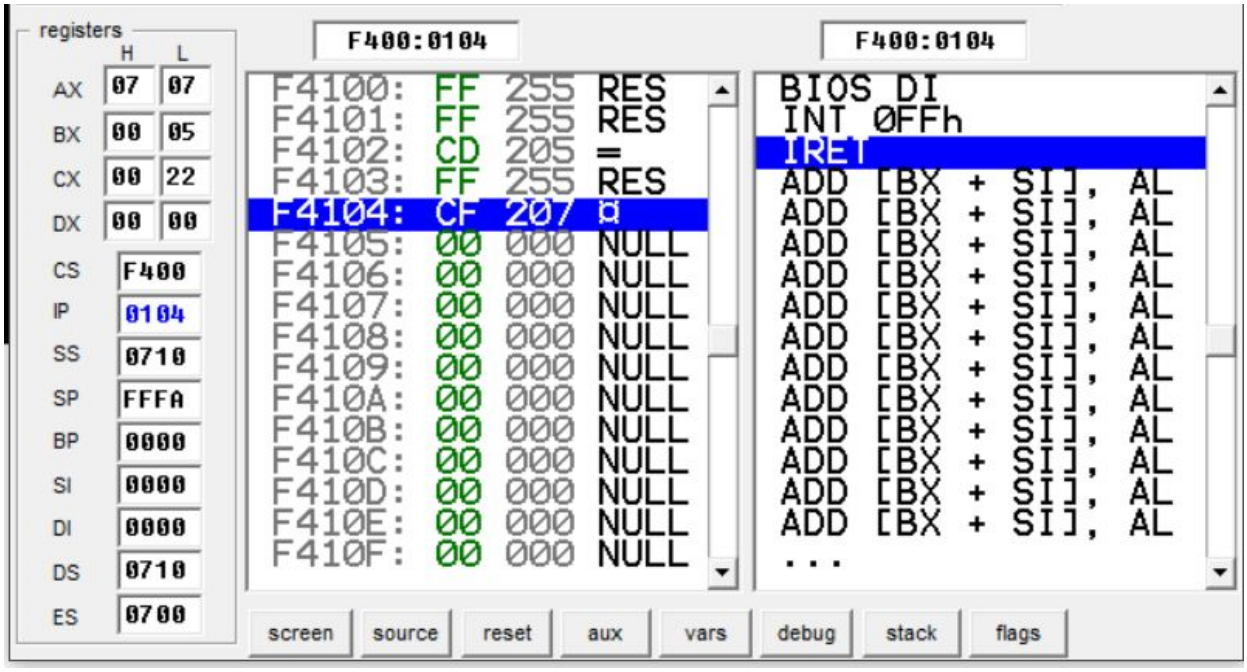
➔ **#Addition of two 8-bit numbers**

Input -

```
data segment
  a db 02h
  b db 05h
  c dw ?
data ends

code segment
assume cs:code,ds:data
  start:
  mov ax,data
  mov ds,ax
  mov al,a
  mov bl,b
  add al,bl
  mov c,ax
  int 3
  code ends
end start
```

Output -



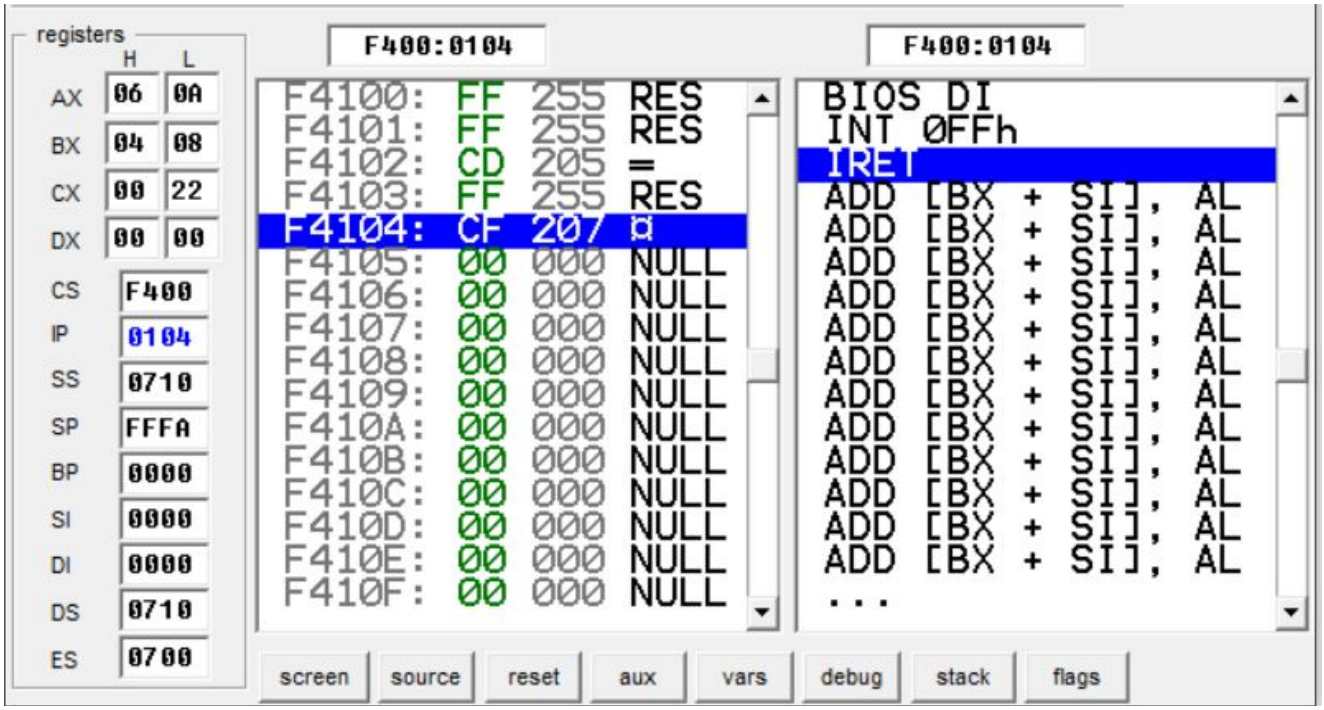
➔ **#Addition of 16-bit numbers**

Input -

```
data segment
  a dw 0202h
  b dw 0408h
  c dw ?
data ends

code segment
assume cs:code,ds:data
  start:
  mov ax,data
  mov ds,ax
  mov ax,a
  mov bx,b
  add ax,bx
  mov c,ax
  int 3
  code ends
end start
```

Output -



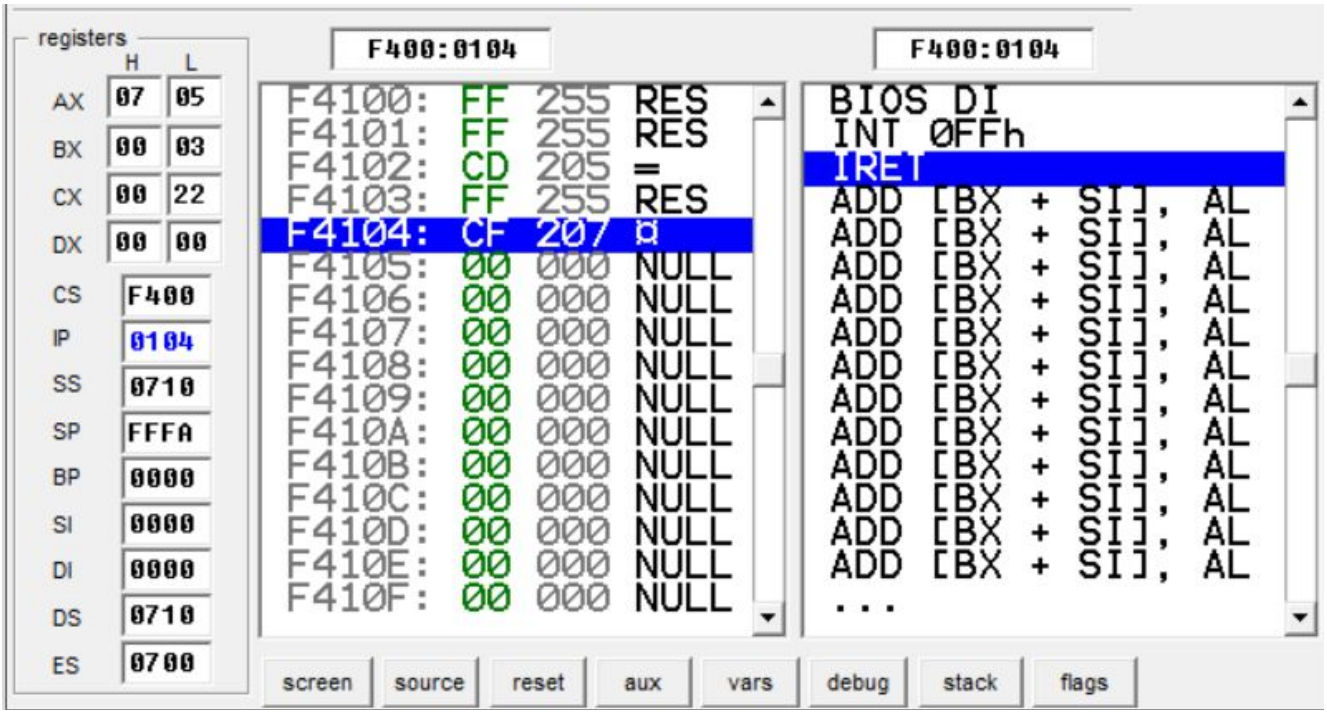
➔ **#Subtraction of 8-bit numbers**

Input -

```
data segment
  a db 08h
  b db 03h
  c dw ?
data ends

code segment
assume cs:code,ds:data
start:
  mov ax,data
  mov ds,ax
  mov al,a
  mov bl,b
  sub al,bl
  mov c,ax
  int 3
  code ends
end start
```

Output -



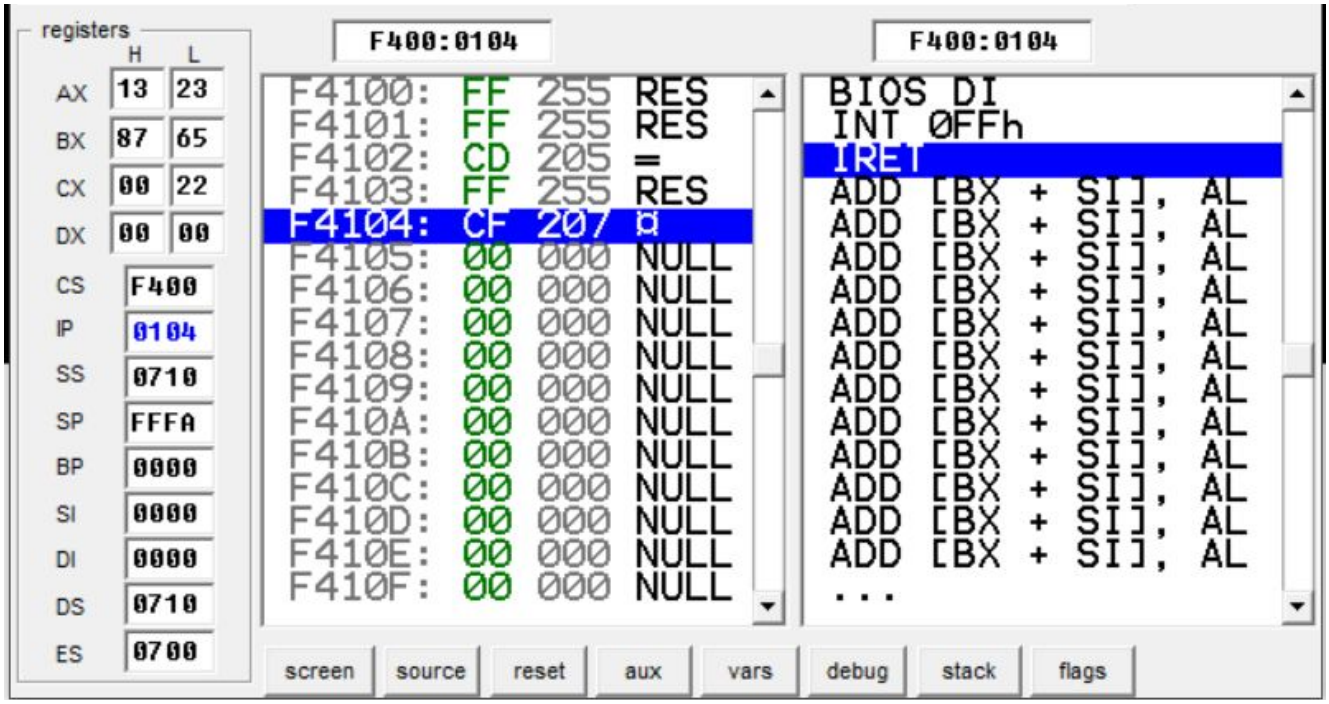
➔ **#Subtraction of 16-bit numbers**

Input -

```
data segment
  a dw 9A88h
  b dw 8765h
  c dw ?
data ends

code segment
assume cs:code,ds:data
  start:
  mov ax,data
  mov ds,ax
  mov ax,a
  mov bx,b
  sub ax,bx
  mov c,ax
  int 3
  code ends
end start
```

Output -



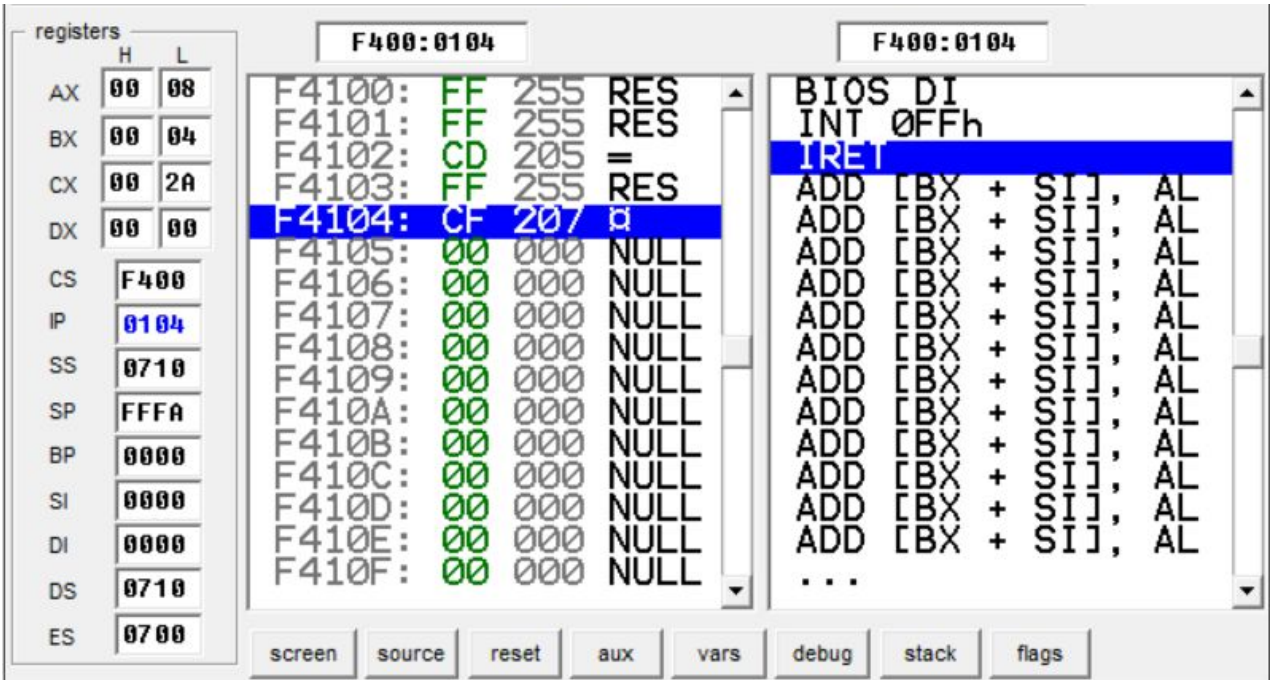
➔ **#Multiplication of 8-bit numbers**

Input -

```
data segment
  a db 02h
  b db 04h
  c dw ?
data ends

code segment
assume cs:code, ds:data
start:
  mov ax,data
  mov ds,ax
  mov ax,0000h
  mov bx,0000h
  mov al,a
  mov bl,b
  mul b
  mov c,ax
  int 3
  code ends
end start
```

Output -



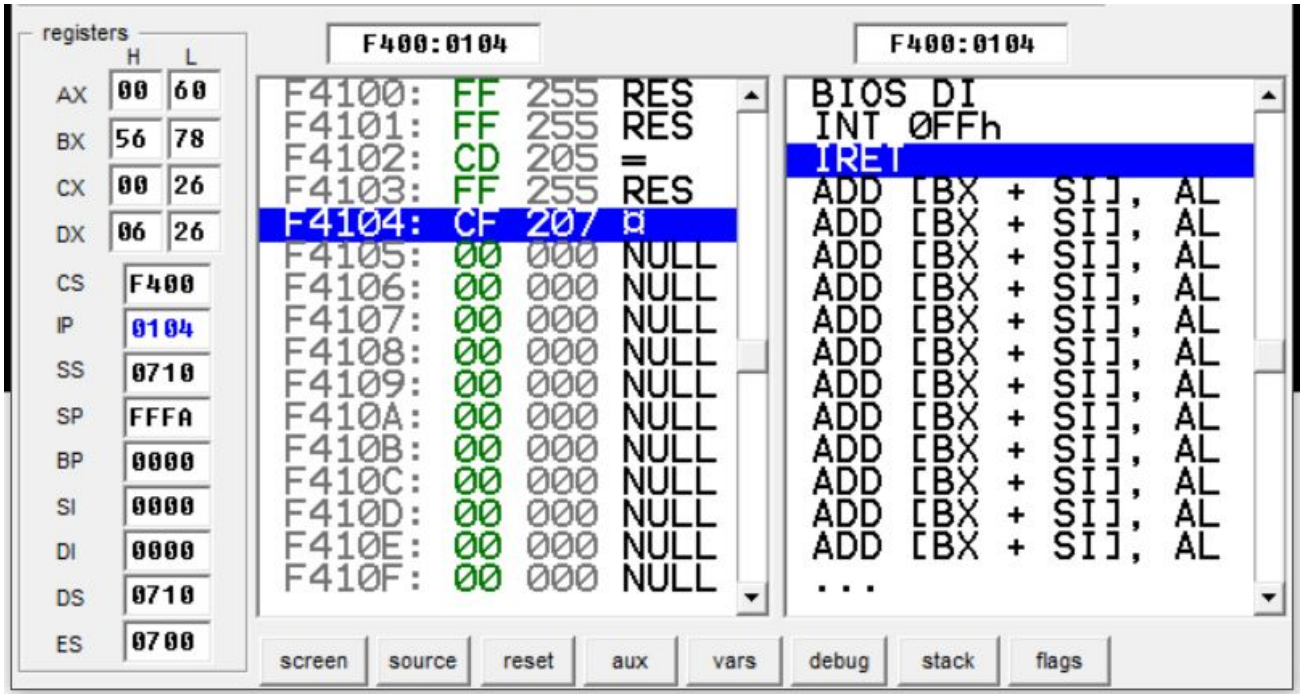
➔ **#Multiplication of 16-bit numbers**

Input -

```
data segment
  a dw 1234h
  b dw 5678h
  c dd ?
data ends

code segment
assume ds:data, cs:code
  start:
  mov ax,data
  mov ds,ax
  mov ax,a
  mov bx,b
  mul bx
  mov word ptr c,ax
  mov word ptr c+2,dx
  int 3
  code ends
end start
```

Output -



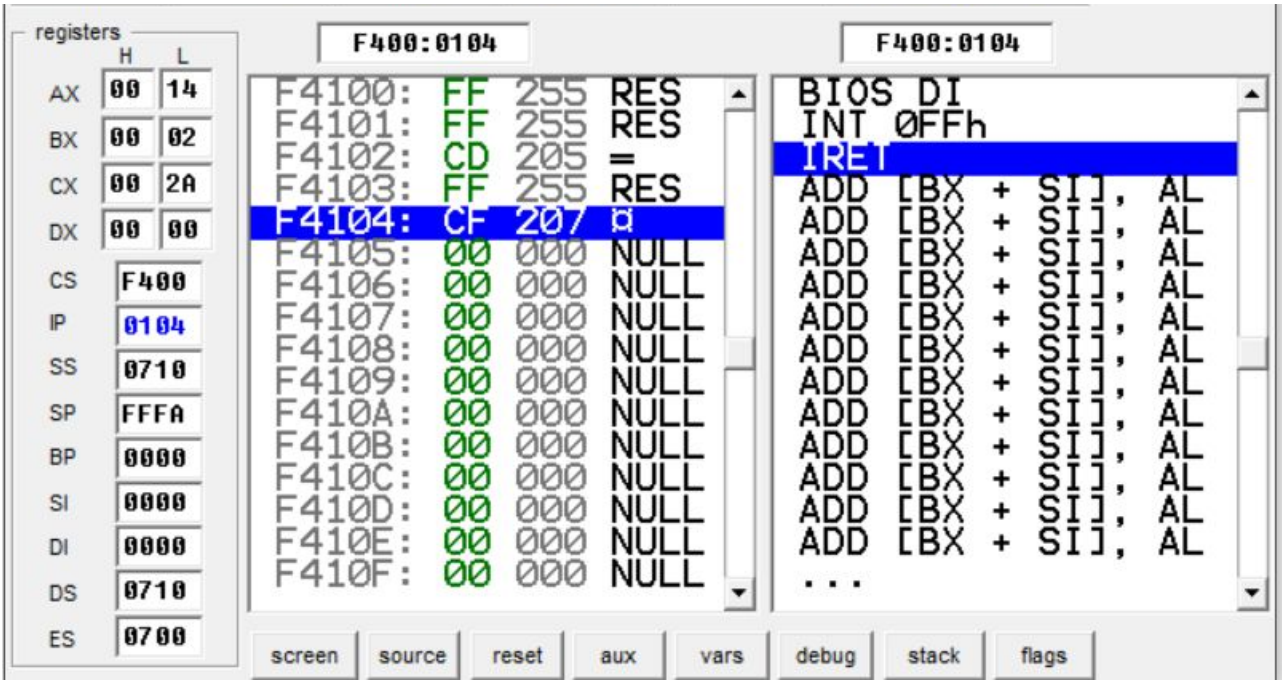
➔ **#Division of 8-bit numbers**

Input -

```
data segment
  a db 28h
  b db 02h
  c dw ?
data ends

code segment
assume cs:code, ds:data
  start:
  mov ax,data
  mov ds,ax
  mov ax,0000h
  mov bx,0000h
  mov al,a
  mov bl,b
  div b
  mov c,ax
  int 3
  code ends
end start
```

Output -



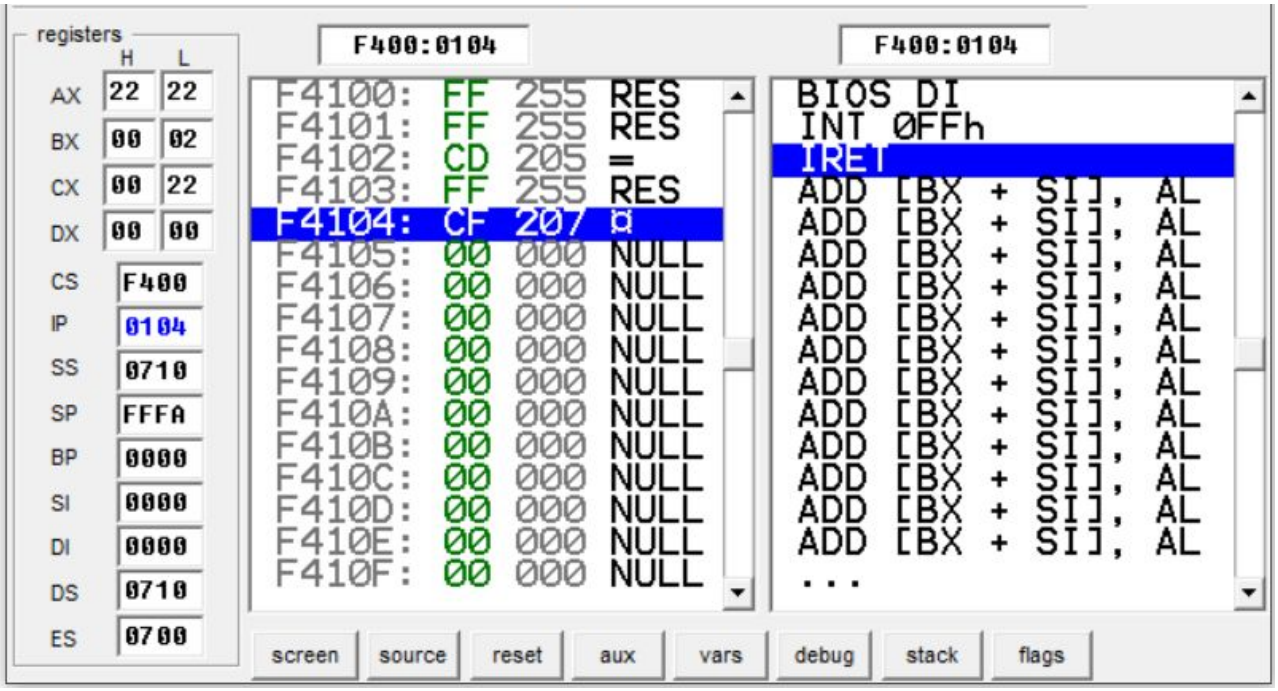
➔ **#Division of 16-bit numbers**

Input -

```
data segment
  a dw 4444h
  b dw 0002h
  c dw ?
data ends

code segment
assume ds:data, cs:code
  start:
  mov ax,data
  mov ds,ax
  mov ax,a
  mov bx,b
  div bx
  mov c,ax
  int 3
  code ends
end start
```

Output -



B.3 Observations and learning:

(Students are expected to comment on the output obtained with clear observations and learning for each task/ subpart assigned)

- 8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of a powerful instruction set, which provides operations like multiplication and division easily.
- It supports two modes of operation, i.e. Maximum mode and Minimum mode. The maximum mode is suitable for systems having multiple processors and Minimum mode is suitable for systems having a single processor.

B.4 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)

We successfully implemented basic arithmetic operations on two 8 bit and 16-bit numbers.

B.5 Question of Curiosity

Q1.Write Algorithm to add two sixteen-bit no's with example

Ans:

➔ Algorithm –

1. Load both the lower bit and higher bit of the first number at once.
2. Copy the content HL pair to the DE pair register.
3. Now load the lower and higher bit of the second number in the HL pair register.
4. ADD both the register pair content using DAD operation.
5. Now move the result at the memory location.

➔ Example -

We are taking two numbers BCAD + FE2D = 1BADA

Input -

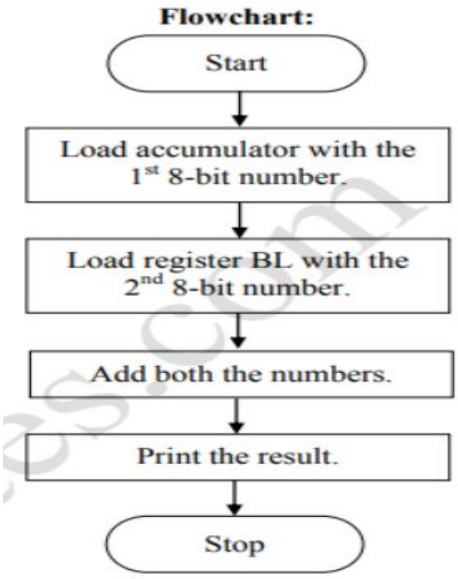
Address	Data
...	...
3000	AD
3001	BC
3002	2D
3003	FE
...	...

Output -

Address	Data
...	...
3004	DA
3005	BA
3006	01
...	...

Q2. Draw flowchart for adding two eight-bit no's

Ans:



Q3.List all arithmetic instructions with example

Ans:

- ➔ Here D stands for destination and S stands for source.
- ➔ D and S can either be register, data or memory address.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
ADD	D, S	D = D + S	ADD AX, [2050]
ADC	D, S	D = D + S + prev. carry	ADC AX, BX
SUB	D, S	D = D – S	SUB AX, [SI]
SBB	D, S	D = D – S – prev. carry	SBB [2050], 0050
MUL	8-bit register	AX = AL * 8-bit reg.	MUL BH
MUL	16-bit register	DX AX = AX * 16-bit reg.	MUL CX

IMUL	8 or 16-bit register	performs signed multiplication	IMUL CX
DIV	8-bit register	AX = AX / 8-bit reg. ; AL = quotient ; AH = remainder	DIV BL
DIV	16-bit register	DX AX / 16-bit reg. ; AX = quotient ; DX = remainder	DIV CX
IDIV	8 or 16-bit register	performs signed division	IDIV BL
INC	D	D = D + 1	INC AX
DEC	D	D = D – 1	DEC [2050]
CBW	none	converts signed byte to word	CBW
CWD	none	converts signed byte to double word	CWD
NEG	D	D = 2’s complement of D	NEG AL
DAA	none	decimal adjust accumulator	DAA
DAS	none	decimal adjust accumulator after subtraction	DAS
AAA	none	ASCII adjust accumulator after addition	AAA
AAS	none	ASCII adjust accumulator after subtraction	AAS
AAM	none	ASCII adjust accumulator after multiplication	AAM
AAD	none	ASCII adjust accumulator after division	AAD