

Name: Kulvir Singh
Register Number: 19BCE2074

Microprocessor Lab Digital Assignment 1

1) 16 Bit Addition

Aim :

Write a program in 8086 Assembly Language to add 2 16 bit numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

19BCE2074. KULVIR SINGH.

classmate

Date _____

Page _____

* Program : 16 Bit addition

DATA SEGMENT

NUM1 DW 2074H ; variable for 1st number
NUM2 DW 2076H ; variable for 2nd number
SUM DW 2 DUP<0> ; Variable to store sum

DATA ENDS

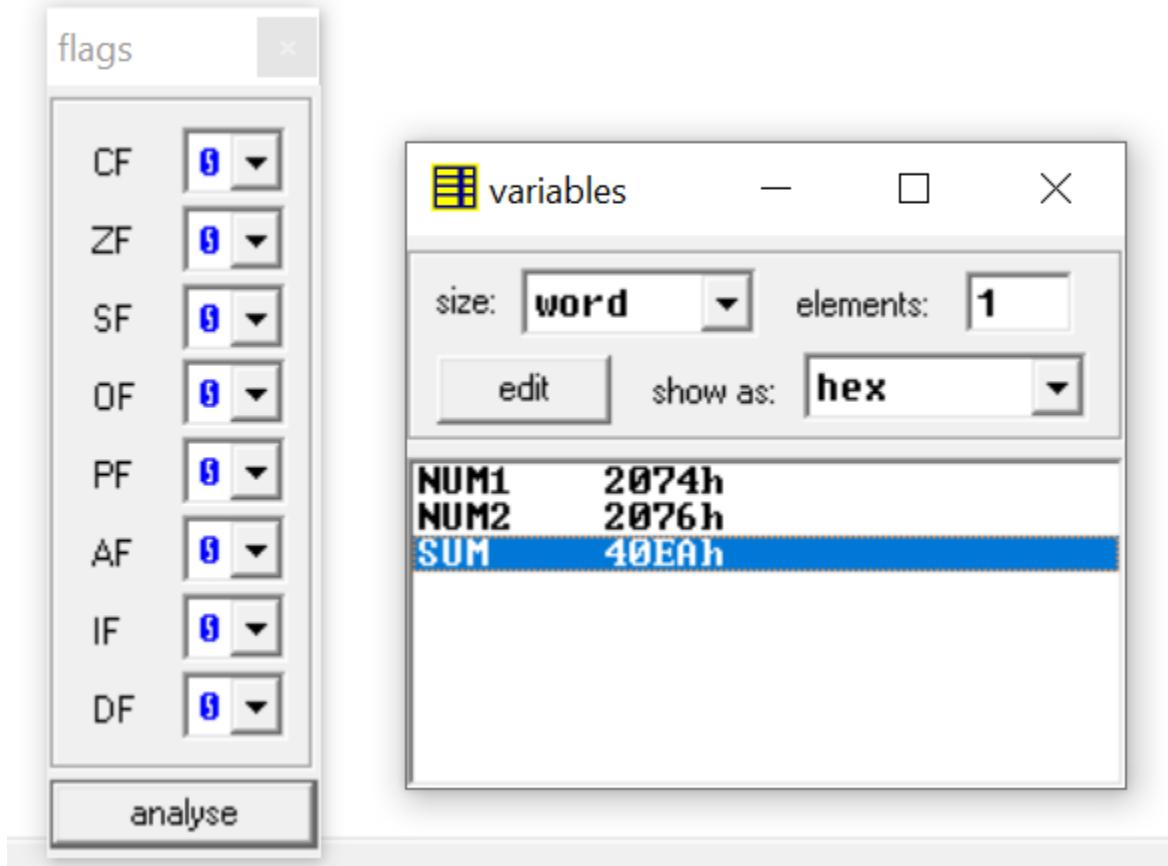
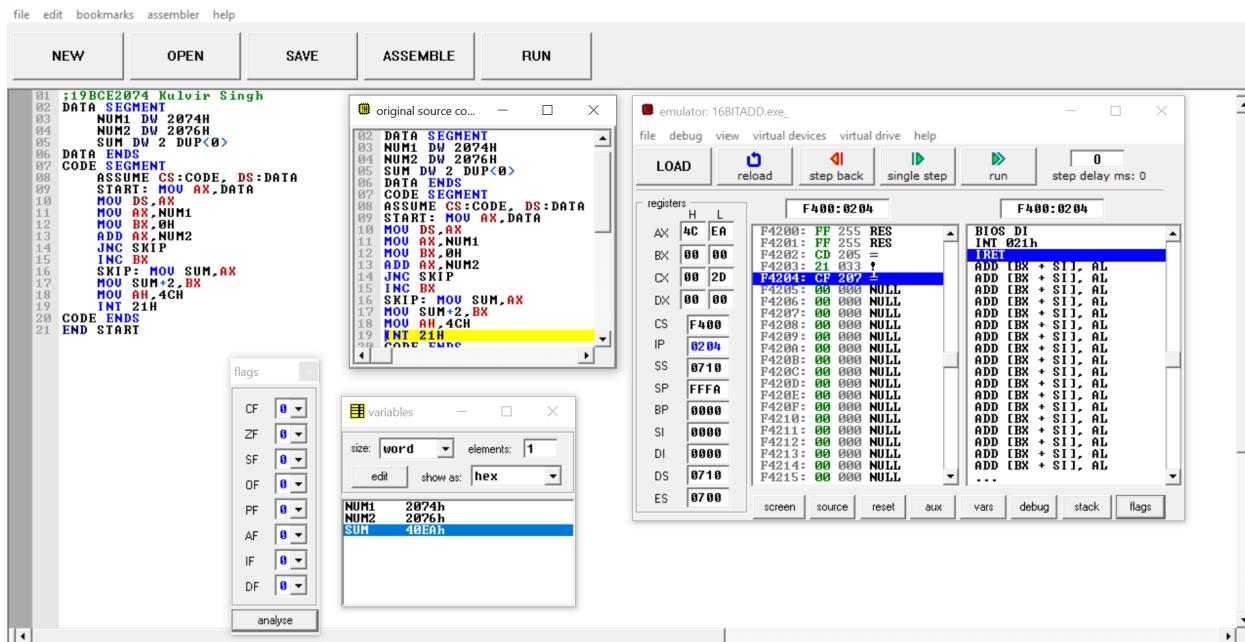
CODE SEGMENT

Assume CS: Code, DS: Data
START: MOV AX, DATA ; general purpose register to store segments
MOV DS, AX ; AX ← DS → Store segments
MOV AX, NUM1 ; AX ← NUM1 → 2074H
MOV BX, 0H ; BX ← 0H, clear carry
ADD AX, NUM2 ; AX ← AX + NUM2
JNC SKIP ; Check for carry.
INC BX ; If carry generated increase BX by 1.
SKIP: MOV SUM, AX ; store value in sum.
MOV SUM+2, BX ; store carry.
MOV AH, 4CH ; interrupt terminate
INT 21H

CODE ENDS

END START

Screenshots :



Inference :

The program can successfully add two 16 bit numbers and stores the correct result in the variable.

2) 16 Bit Subtraction

Aim :

Write a program in 8086 Assembly Language to subtract 2 16 bit numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

19BCE2074 KULVIR SINGH

* Program : 16 Bit Subtraction. II : ~~Completed~~ *

DATA SEGMENT

NUM1 DW 19BC H ; NUM1 to store 1st no. for sub.

NUM2 DW 2074 H ; NUM2 to store 2nd no. for sub.

RES DW 2 DP> ; result variable for storing difference

DATA ENDS

CODE SEGMENT

Assume CS: Code, DS: Data

START: MOV AX, Data ; general purpose register

MOV DS, AX ; DS to store segments

MOV AX, NUM1 ; AX \leftarrow (NUM1) \Rightarrow 19BC H

MOV CX, 0000H ; clear carry flag for borrow.

SUB AX, NUM2 ; AX \leftarrow AX - NUM2

JNC SKIP ; Skip if there is no carry.

INC CX ; If borrow present (ie) carry present increment CX by 1.

SKIP: MOV RES, AX ; store result of subtraction

MOV RES+2, CX ; store the borrow value

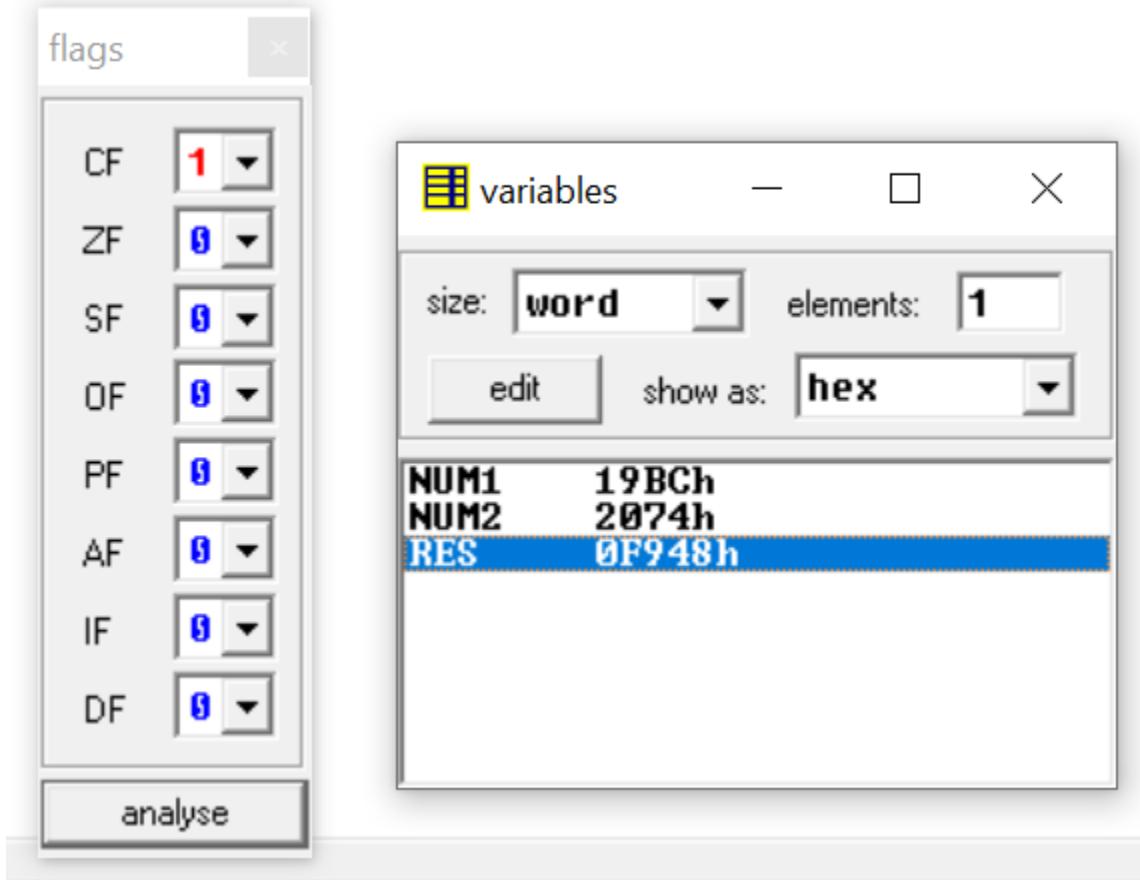
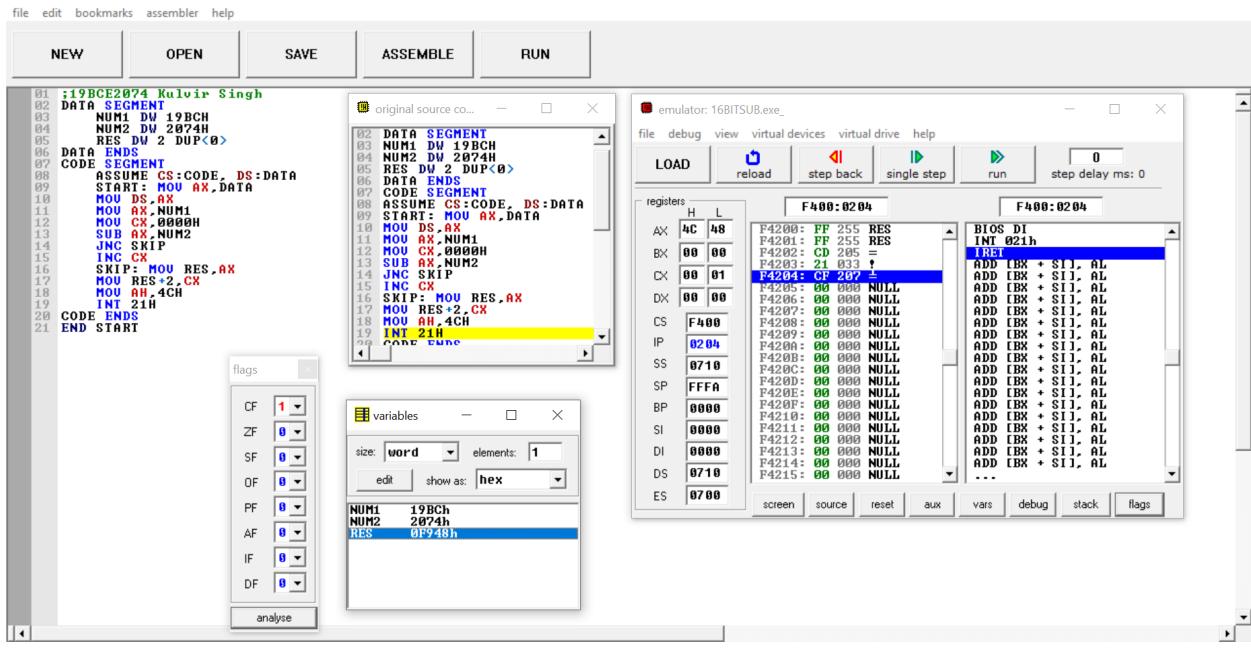
MOV AH, 4CH ; terminate

INT 21H

CODE ENDS

END START

Screenshots :



Inference :

The program can successfully subtract two 16 bit numbers and store the correct result in the variable.

3) 32 Bit Addition

Aim :

Write a program in 8086 Assembly Language to add 2 32 bit numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

19BCE2074

KULVIR SINGH

classmate

Date _____

Page _____

* Program : 32 Bit Addition

DATA SEGMENT

NUM1 DD 19BCE207H ; 32 Bit no.

NUM2 DD 19BCE004H ; 32 Bit no.

SUM DW 3 DUP(0) ; 3 word space reserved for sum

DATA ENDS

CODE SEGMENT

ASSUME DS: DATA, CS: CODE

START: MOV AX, DATA ; Segment address can only

MOV DS, AX ; be loaded using AX register

MOV AX, NUM1 ; LSB of Num1 stored

MOV AX, NUM2 ; LSB of Num2 + Num1 stored

MOV SUM, AX ; sum ← LSB of Num1 + Num2

MOV AX, NUM1+2 ; AX ← MSB of Num1

ADC AX, NUM2+2 ; AX ← LSB tally + MSB add.

MOV SUM+2, AX ; MSB add res. stored

JNC SKIP ; check for carry

MOV SUM+4, 0IH ; stores of array

SKIP: MOV AH, 4CH ; terminates

INT 21H

CODE ENDS

END START

Screenshots :

file edit bookmarks assembler help

NEW OPEN SAVE ASSEMBLE RUN

```

01 :19BCE2024 Kulvir Singh
02 DATA SEGMENT
03 NUM1 DD 19BCE207H
04 NUM2 DD 19BCE004H
05 SUM DW 3 DUP(0)
06 DATA ENDS
07 CODE SEGMENT
08 ASSUME DS:DATA, CS:CODE
09 START: MOV AX,DATA
10 MOU DS,AX
11 MOU AX,NUM1
12 ADD AX,NUM2
13 MOU SUM,AX
14 MOU AX,NUM1+2
15 ADC AX,NUM2+2
16 MOU SUM+2,AX
17 JNC SKIP
18 MOU SUM+4,01H
19 SKIP: MOU AH,4CH
20 INT 21H
21 CODE ENDS
22 END START

```

original source code

emulator: 32BITADD.exe

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

registers

	H	L
AX	4C	79
BX	00	00
CX	00	35
DX	00	00
CS	F400	
IP	0204	
SS	0710	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

F400:0204 F400:0204

```

F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 285 =
F4203: 48 00000000
F4204: CF 202 1
F4205: 00 0000 NULL
F4206: 00 0000 NULL
F4207: 00 0000 NULL
F4208: 00 0000 NULL
F4209: 00 0000 NULL
F420A: 00 0000 NULL
F420B: 00 0000 NULL
F420C: 00 0000 NULL
F420D: 00 0000 NULL
F420E: 00 0000 NULL
F420F: 00 0000 NULL
F4210: 00 0000 NULL
F4211: 00 0000 NULL
F4212: 00 0000 NULL
F4213: 00 0000 NULL
F4214: 00 0000 NULL
F4215: 00 0000 NULL

```

BIOS DI INT 021h RET

screen source reset aux vars debug stack flags

flags

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

variables

size: dword elements: 1

edit show as: hex

NUM1	19BCE207h
NUM2	19BCE004h
SUM	3379C20Bh

Inference :

The program can successfully add two 32 bit numbers and store the correct result in the variable.

4) 32 Bit Subtraction

Aim :

Write a program in 8086 Assembly Language to subtract 2 32 bit numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

19BCE2074 KULVIR SINGH

classmate
Date _____
Page _____

* Program : 32 Bit Subtraction

DATA SEGMENT

NUM1 DD 198CE207H ; 32 Bit Number

NUM2 DD 12345678H ; 32 Bit Number

RES DW ? ; 16 Bit Result variable

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV AX, DATA

MOV DS, AX

MOV DL, 00H

MOV AX, NUM1 ; AX ← (NUM1)

MOV BX, NUM2 ; BX ← (NUM2)

SUB AX, BX ; Subtract LSB of 2 nos.

MOV RES, AX ; Store result of sub in RES

MOV AX, NUM1+2 ; MSB of NUM1 to AX

MOV BX, NUM2+2 ; MSB of NUM2 to BX

SBB AX, BX ; Subtraction with borrow

MOV NUM2+2, AX ; MSB of SBD result stored

JNC SKIP ; If carry then move to skip

SKIP: MOV BYTE PTR RES+4, DL ; Store carry

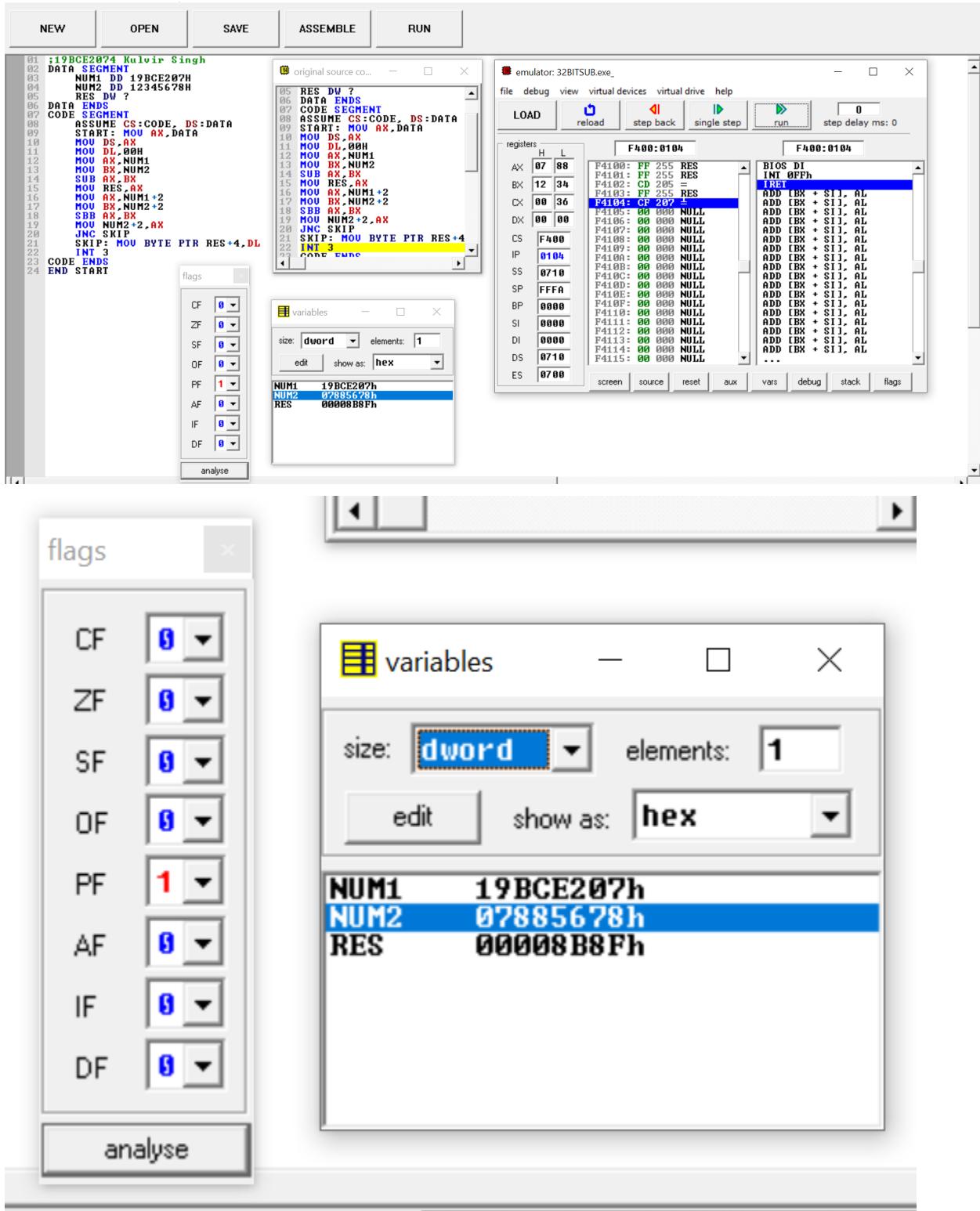
MOV AH, 04CH ; terminate

INT 21H

CODE ENDS

END START

Screenshots :



Inference :

The program can successfully subtract two 32 bit numbers and store the correct result in the variable.

5) 16 Bit Multiplication UNSIGNED

Aim :

Write a program in 8086 Assembly Language to multiply 2 16 bit UNSIGNED numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

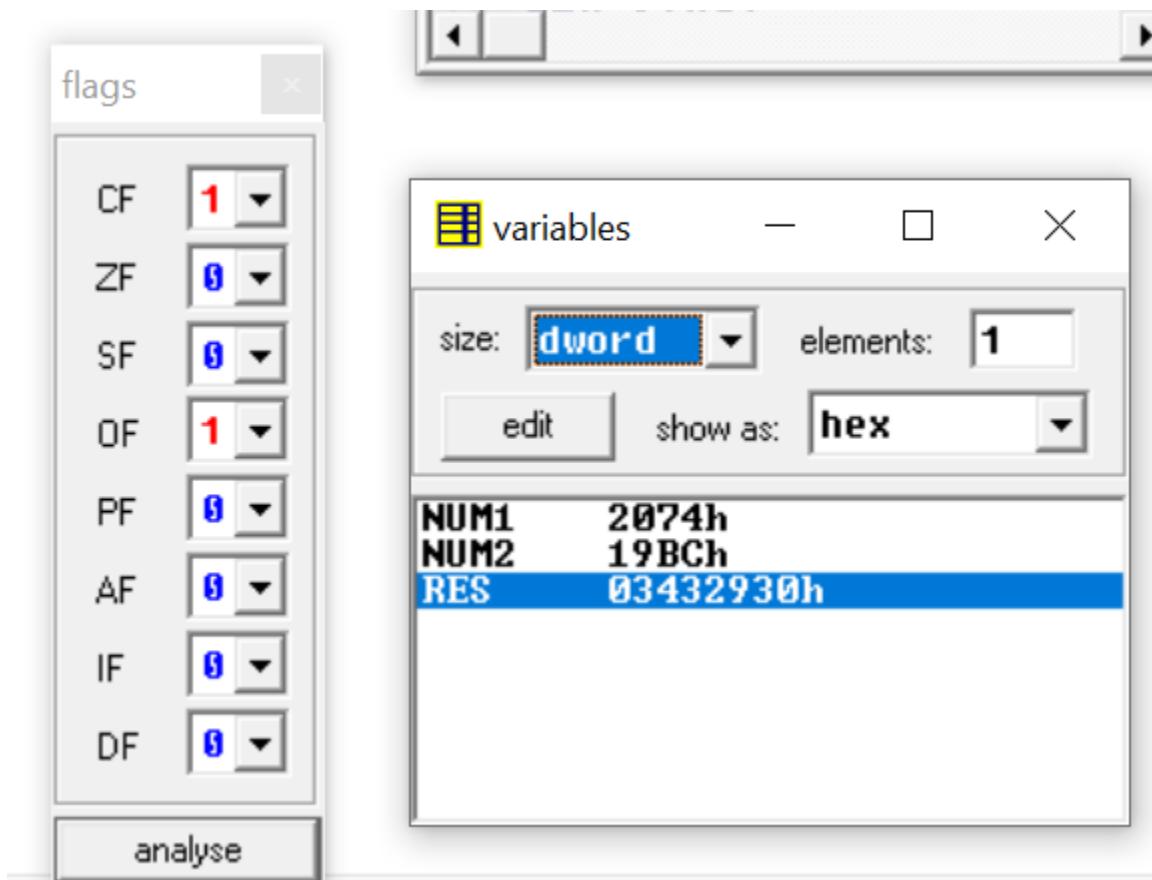
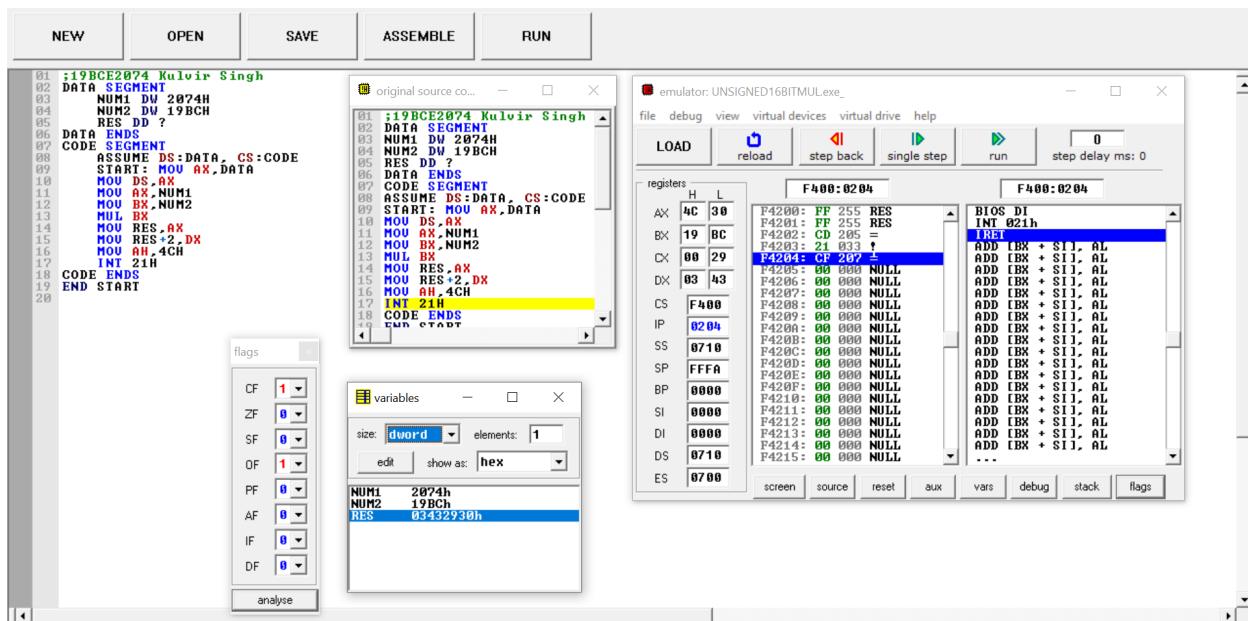
19BCE2024
KULVIR SINGH
Program: 16 bit Multiplication (UNSIGNED)

classmate
Date _____
Page _____

```
DATA SEGMENT
    NUM1 DW 2024H ; Number1 to be multiplied
    NUM2 DW 198CH ; Number2 to be multiplied
    RES DD ? ; 32 bit result variable
DATA ENDS

CODE SEGMENT
    ASSUME DS: DATA, CS: CODE
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, NUM1 ; AX ← (NUM1) → 2024H
    MOV BX, NUM2 ; AX, BX ← (NUM2) → 198CH
    MUL BX ; perform unsigned multiplication
    MOV AH, RES ; LSB of multiplied result
    MOV RES+2, DX ; MSB of multiplied result
    MOV AH, 4CH ; to terminate
    INT 21H
CODE ENDS
END START
```

Screenshots :



Inference :

The program can successfully multiply two 16 bit UNSIGNED numbers and store the correct result in the variable.

6) 16 Bit Multiplication SIGNED

Aim :

Write a program in 8086 Assembly Language to multiply 2 16 bit SIGNED numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

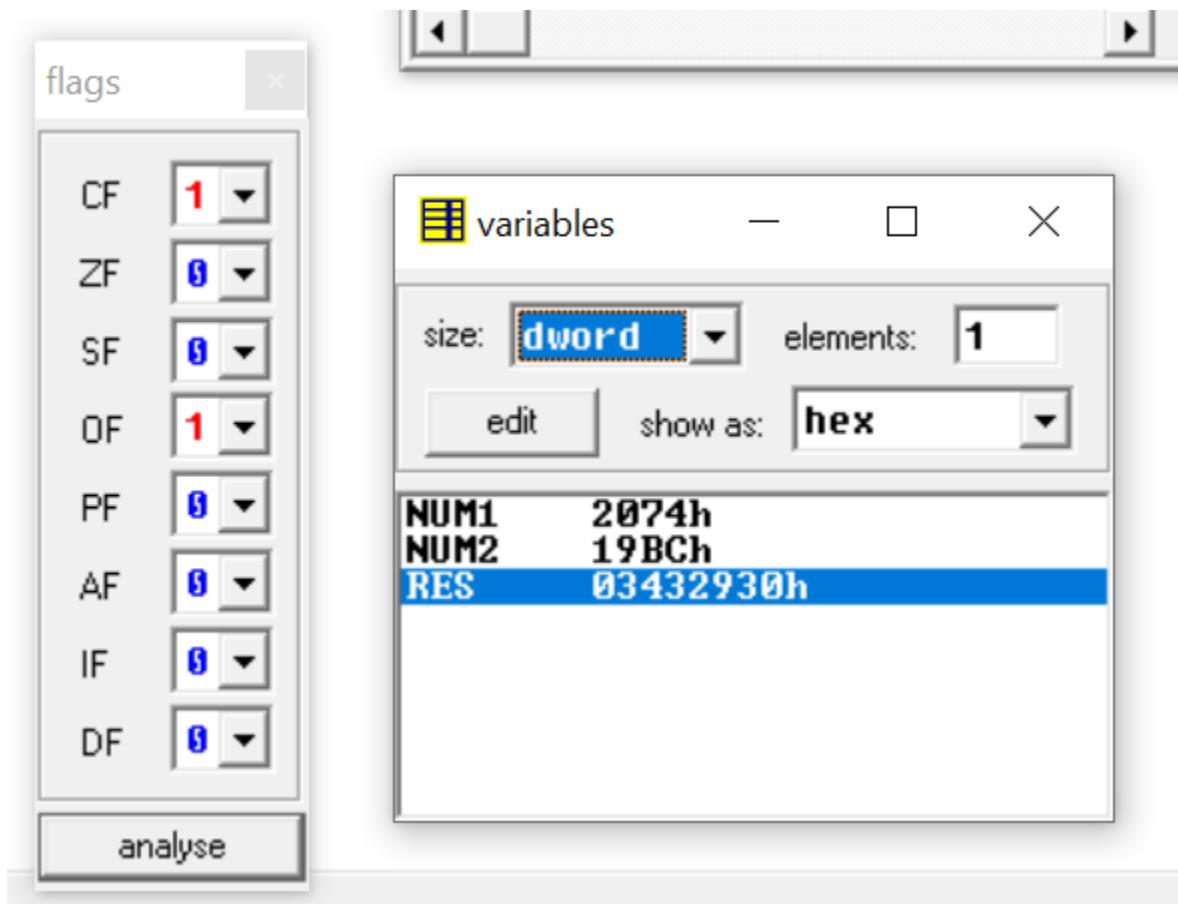
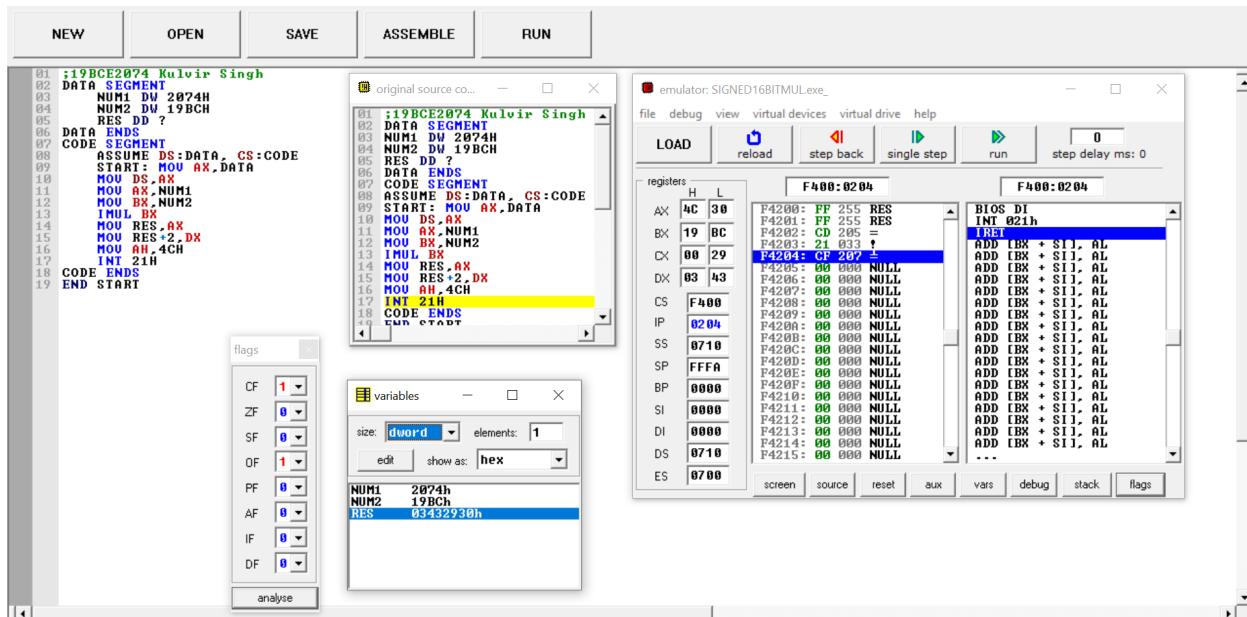
Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

19BCF2074 KULVIR SINGH
Program : 16 Bit Multiplication (SIGNED)

```
DATA SEGMENT
    NUM1 DW 2024H ; Number1 to be multiplied
    NUM2 DW 19BC H ; Number2 to be multiplied
    RES DD ? ; 32 bit result variable
DATA ENDS
CODE SEGMENT
    ASSUME DS:DATA, CS:CODE
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, NUM1 ; AX ← (NUM1) → 2024H
    MOV BX, NUM2 ; BX ← NUM2 → 19BC H
    IMUL BX, AX ; perform signed multiplication
    MOV RES, AX ; ←LSB of multiplied result
    MOV AX, RES+2, DX ; MSB of multiplied result
    MOV AH, 4CH ; terminate
    INT 21H
CODE ENDS
END START
```

Screenshots :



Inference :

The program can successfully multiply two 16 bit SIGNED numbers and store the correct result in the variable.

7) 16 Bit Division UNSIGNED

Aim :

Write a program in 8086 Assembly Language to divide 2 16 bit UNSIGNED numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

* Program : 16 bit division (UNSIGNED)

```

DATA SEGMENT
    NUM1    DD    ?
    NUM2    DW    ?
    QUOTIENT DW ? ; store the quotient
    REM     DW ? ; store the remainder
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
       MOV DS, AX ; VALUE OF DATA STORED IN AX
       MOV AX, NUM1 ; LSB OF NUM1 stored in AX
       MOV DX, NUM1+2 NUM1+2 ; MCB Q NUM1 stored in AX
       MOV BX, NUM2 ; divisor stored in BX
       DIV BX ; Unsigned division Q=AX/BX
       MOV QUOTIENT, AX ; result of division stored
       MOV REM, DX ; remainder of division stored
       MOV AH, 4CH ; terminated
       INT 21H
CODE ENDS
END START

```

Screenshots :

The screenshot shows a debugger interface with the following components:

- Assembly View:** Displays the assembly code for the program. The code initializes variables NUM1 and NUM2, performs division, and stores the quotient and remainder in variables QUOTIENT and REM.
- Registers View:** Shows the state of CPU registers. The AX register contains the value 13, BX contains 0A, CX contains 20, and DX contains 03. The CS register points to the code segment, and IP points to the instruction at address F400:0204, which is the INT 21H instruction.
- Stack View:** Displays the stack contents starting with BIOS:DI and INT 021h.
- Flags View:** Shows the current state of various flags: CF (0), ZF (0), SF (0), OF (0), PF (0), AF (0), IF (0), and DF (0).
- Variables View:** Shows the values of variables NUM1 (000000C1h), NUM2 (0013000Ah), QUOTIENT (0013h), and REM (0003h). The REM variable is currently selected.

Inference :

The program can successfully divide two 16 bit UNSIGNED numbers and store the correct result in the variable.

8) 16 Bit Division SIGNED

Aim :

Write a program in 8086 Assembly Language to divide 2 16 bit SIGNED numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

19BCF2074 KULVIR SINGH
Program : 16 Bit Division (SIGNED)

```
DATA SEGMENT
    NUM1 DD ?
    NUM2 DW ?
    QUOTIENT DW ? ; store the quotient
    REM DW ? ; store the remainder
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, NUM1 ; LSB of NUM1 stored in AX
    MOV DX, NUM1+2 ; MSB of NUM1 stored in DX
    MOV BX, NUM2 ; divisor to be stored in BX
    IDIV DX ; perform NUM1 / NUM2 signed
    MOV QUOTIENT, AX ; result of idiv stored in AX to quo
    MOV REM, DX ; remainder stored in DX to rem
    MOV AH, 4CH ; terminate
    INT 21H
CODE ENDS
END START
```

Screenshots :

File Edit Workbooks Assemble Help

NEW OPEN SAVE ASSEMBLE RUN

```

01 ;19BCE2074 Kulvir Singh
02 DATA SEGMENT
03 NUM1 DD FFFFFFFB5h
04 NUM2 DD 000Ch
05 QUOTIENT DW ?
06 REM DW ?
07 DATA ENDS
08 CODE SEGMENT
09 ASSUME CS:CODE, DS:DATA
10 START: MOV AX, DATA
11 MOU DS, AX
12 MOU AX, NUM1
13 MOU DX, NUM1+2
14 MOU BX, NUM2
15 IDIV BX
16 MOU AX, QUOTIENT, AX
17 MOU REM, DX
18 MOU AH, 04Ch
19 INT 21H
20 CODE ENDS
21 END START

```

original source code

DATA SEGMENT
NUM1 DD FFFFFFFB5h
NUM2 DD 000Ch
QUOTIENT DW ?
REM DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
MOU DS, AX
MOU AX, NUM1
MOU DX, NUM1+2
MOU BX, NUM2
IDIV BX
MOU AX, QUOTIENT, AX
MOU REM, DX
MOU AH, 04Ch
INT 21H
CODE ENDS
END START

flags

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

variables

size: word elements: 1

edit show as: hex

NUM1 FFFFFFFB5h
NUM2 000Ch
QUOTIENT 0FFFh
REM 0FFFh

Registers

AX	4C FA
BX	00 0C
CX	00 20
DX	FF FD
CS	F400
IP	0204
SS	0710
SP	FFFA
BP	0000
SI	0000
DI	0000
DS	0710
ES	0700

Memory Dump

F400:0204 F400:0204

F42001: FF 255 RES	B105 DI INT 021h
F42012: 00 255 RES	ADD BX + SI1, AL
F42023: CD 283 =	ADD BX * SI1, AL
F42034: 21 033 ;	ADD BX + SI1, AL
F42045: CE 207 ±	ADD BX * SI1, AL
F42056: 00 000 NULL	ADD BX * SI1, AL
F42067: 00 000 NULL	ADD BX * SI1, AL
F42078: 00 000 NULL	ADD BX * SI1, AL
F42089: 00 000 NULL	ADD BX * SI1, AL
F4209A: 00 000 NULL	ADD BX * SI1, AL
F420B0: 00 000 NULL	ADD BX * SI1, AL
F420C1: 00 000 NULL	ADD BX * SI1, AL
F420D2: 00 000 NULL	ADD BX * SI1, AL
F420E3: 00 000 NULL	ADD BX * SI1, AL
F420F4: 00 000 NULL	ADD BX * SI1, AL
F42105: 00 000 NULL	ADD BX * SI1, AL
F42116: 00 000 NULL	ADD BX * SI1, AL
F42127: 00 000 NULL	ADD BX * SI1, AL
F42138: 00 000 NULL	ADD BX * SI1, AL
F42149: 00 000 NULL	ADD BX * SI1, AL
F42150: 00 000 NULL	ADD BX * SI1, AL

Buttons: screen, source, reset, aux, vars, debug, stack, flags

Inference :

The program can successfully divide two 16 bit SIGNED numbers and store the correct result in the variable.

9) 8 Bit Division UNSIGNED

Aim :

Write a program in 8086 Assembly Language to divide 2 8 bit UNSIGNED numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

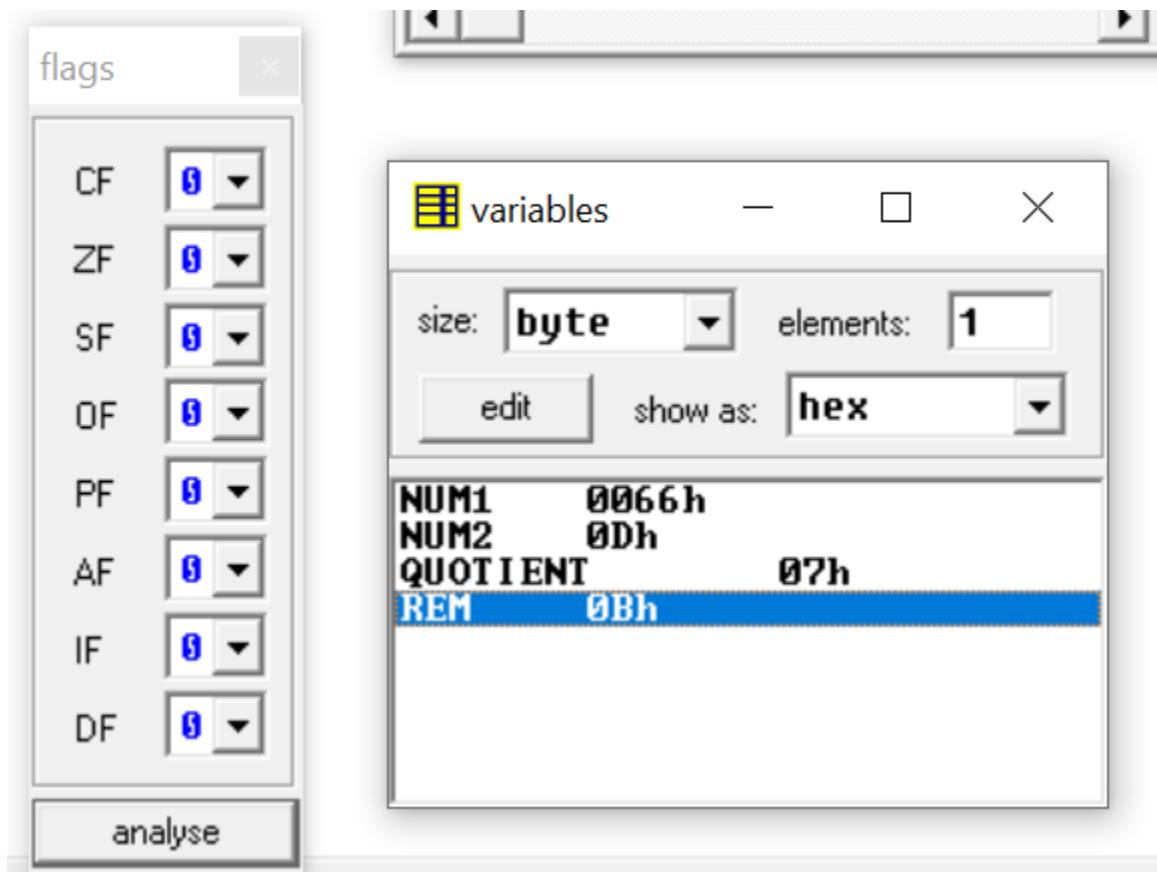
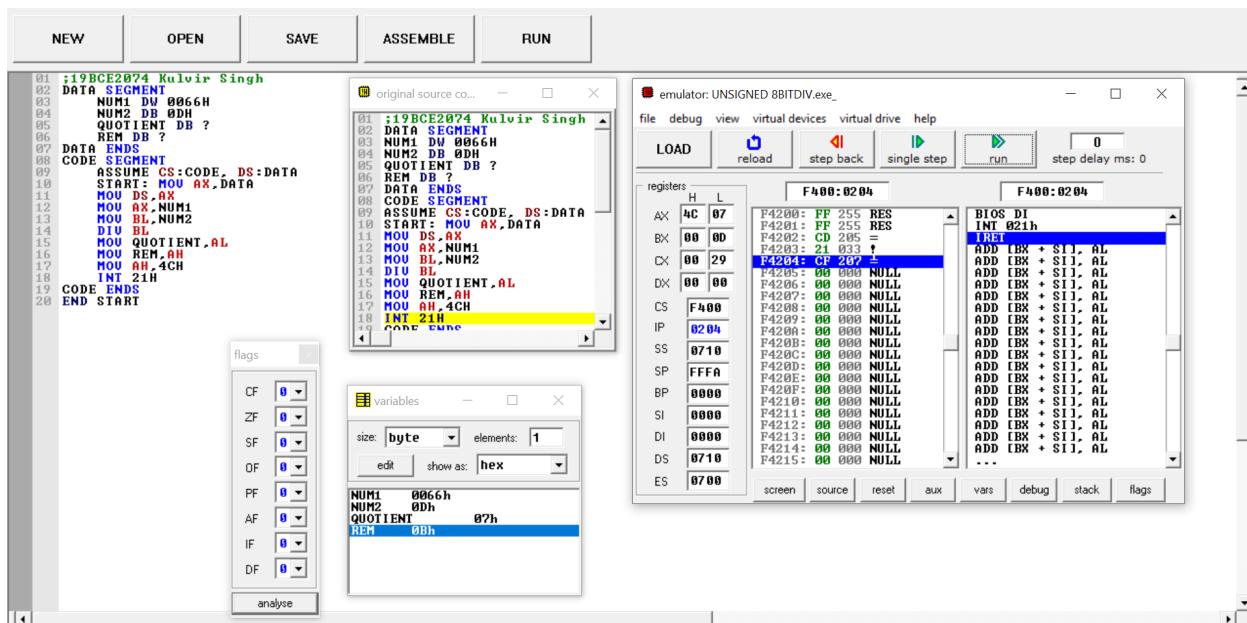
Handwritten Program :

19BCE2074 KULVIR SINGH

* RRD Program : 8 Bit Division (UNSIGNED)

```
DATA SEGMENT
    NUM1 DW 0066H ; 16 bit unsigned number
    NUM2 DB 0DH ; 8 bit divisor
    QUOTIENT DB ? ; 8 bit Quotient variable
    REM DB ? ; 8bit remainder variable
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
    START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, NUM1 ; AX ← (NUM1) ⇒ 0066H
        MOV BL, NUM2 ; BL ← (NUM2) ⇒ 0DH
        DIV BL ; AX = Divides NUM1 by NUM2
        MOV QUOTIENT, AL ; Result of DIV in AL moved to Quotient
        MOV AH, REM ; Remainder from AH moved to REM
        MOV AH, 04CH ; terminate
        INT 21H
CODE ENDS
END START
```

Screenshots :



Inference :

The program can successfully divide two 8 bit UNSIGNED numbers and store the correct result in the variable.

10) 8 Bit Division SIGNED

Aim :

Write a program in 8086 Assembly Language to divide 2 8 bit SIGNED numbers and store the result in a variable

Requirements :

8086 EMU - An emulator to run the 8086 Assembly Language Code

Operating System - Any valid operating system that can execute the emulator

Handwritten Program :

19BCE2074 KULVIR SINGH
Date _____
Page _____

* Program : 8 Bit Division (SIGNED)

```
DATA SEGMENT
    NUM1 DW 00FFH ; 16 bit Signed Number
    NUM2 DB 0DH ; 8 bit divisor
    QUOTIENT DB ? ; 8 bit Quotient variable
    REM DB ? ; 8 bit Remainder variable
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV AX, NUM1 ; AX ← (NUM1) → 00FFH
    MOV BL, NUM2 ; AX's BL ← (NUM2) → 0DH
    IDIV BL ; X; AX signed division NUM1 / NUM2
    MOV QUOTIENT, AL ; Result of IDIV moved to Quotient
    MOV AH, REM ; Remainder of IDIV moved to REM
    MOV AH, 04CH ; terminate
    INT 21H
CODE ENDS
END START
```

Screenshots :

The screenshot displays a debugger interface with several windows:

- Assembly Window:** Shows the assembly code for a program named "Kulvir Singh". The code performs division (NUM1 / NUM2) and stores the quotient (QUOTIENT) and remainder (REM) in memory.
- Registers Window:** Shows the CPU register values at address F400:0204. Registers include AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, and ES. The IP register shows the instruction F4204: CD 20h, which corresponds to the INT 21h interrupt call.
- Stack Window:** Shows the stack contents starting with BIOS DI INT 021h, followed by a series of ADD BX + SI, AL instructions.
- Flags Window:** Shows the current flag settings: CF=0, ZF=0, SF=0, OF=0, PF=0, AF=0, IF=0, and DF=0.
- Variables Window:** Shows the variable values: NUM1 = 00FFh, NUM2 = 0Dh, QUOTIENT = 13h, and REM = 08h.

Inference :

The program can successfully divide two 8 bit SIGNED numbers and store the correct result in the variable.