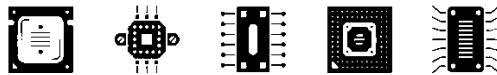


CSE 2006

MICROPROCESSOR AND INTERFACING



Task – 1

L11+L12 | SJT516

FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI

19BCE2105

The screenshot shows a Zoom meeting window. At the top, there's a status bar indicating "Recording has started". The main area displays a presentation slide from VIT (Vellore Institute of Technology). The slide title is "Exercise" and it contains a bullet point: "Write ALP codes for the following arithmetic operations:". Below this is a table with 8 rows of problems. Each row includes a problem number, arithmetic instructions, and the registers to be used. To the right of the slide is a sidebar showing a list of participants, including Arvind Kumar as the organizer. At the bottom of the screen, there are icons for various functions like chat, share screen, and a toolbar with participant initials.

Exercise

- Write ALP codes for the following arithmetic operations:

Problem No.	Arithmetic Instructions	Which registers are to be used?
1	12H + CAH	CL, DL
2	1A4CH + B1DEH	AX, BX
3	7AH – 4CH	CL, DL
4	3B7AH – C142H	BX, CX
5	1DH × 77H	AL, BL
6	EF1AH × CD50H	AX, BX
7	19H ÷ 03H	AL, BL
8	1927H ÷ 1201H	AX, BX

In each case interpret the results of different flags. Crosscheck your results by converting them into decimal numbers.

<https://sites.google.com/view/arvindk>

27

1. 12H + CAH (using CL, DL registers)

Code:

```
MOV CL,12h
MOV DL,0CAh
ADD CL,DL; 12H + CAH = (18+202) in decimal = 220 = DCh (in hexadecimal)
ret
```

The screenshot displays the emu8086 emulator interface. The main window shows assembly code with the following instructions:

```

01: MOV CL, 12h
02: MOV DL, 0CAh
03: ADD CL, DL; 12h + CAh = <18*202> in decimal = 220 = DCh <in hexadecimal>
04: ret
05:

```

The registers window shows the state of various registers. The CX register is highlighted with a value of 0012. The DS register is highlighted with a value of 000A. The registers window also shows the state of other registers like AX, BX, CX, DX, ES, etc.

The flags window shows the state of various flags. The CF (Carry Flag) is set to 0. The ZF (Zero Flag) is set to 0. The SF (Sign Flag) is set to 0. The OF (Overflow Flag) is set to 0. The PF (Parity Flag) is set to 0. The AF (Auxiliary Flag) is set to 1. The DF (Direction Flag) is set to 0.

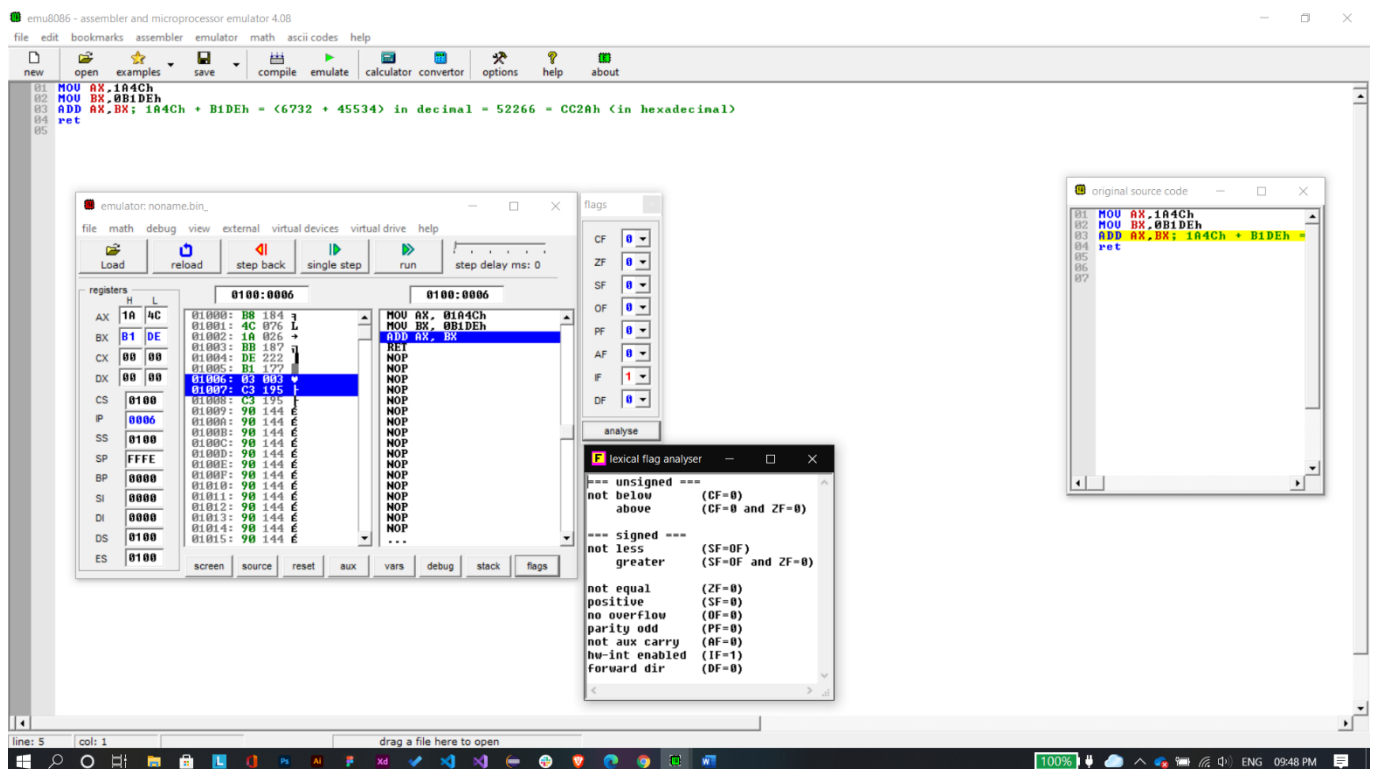
The original source code window shows the assembly code being executed, with the instruction ADD CL, DL; 12h + CAh = <18*202> in decimal = 220 = DCh <in hexadecimal> highlighted.

The lexical flag analyser window shows the analysis of the instruction ADD CL, DL; 12h + CAh = <18*202> in decimal = 220 = DCh <in hexadecimal>. The analysis shows that the instruction is unsigned, not below, above, signed, not less, greater, not equal, positive, no overflow, parity odd, not aux carry, hw-int enabled, and forward dir.

2. 1A4Ch + B1DEh (using AX, BX registers)

Code:

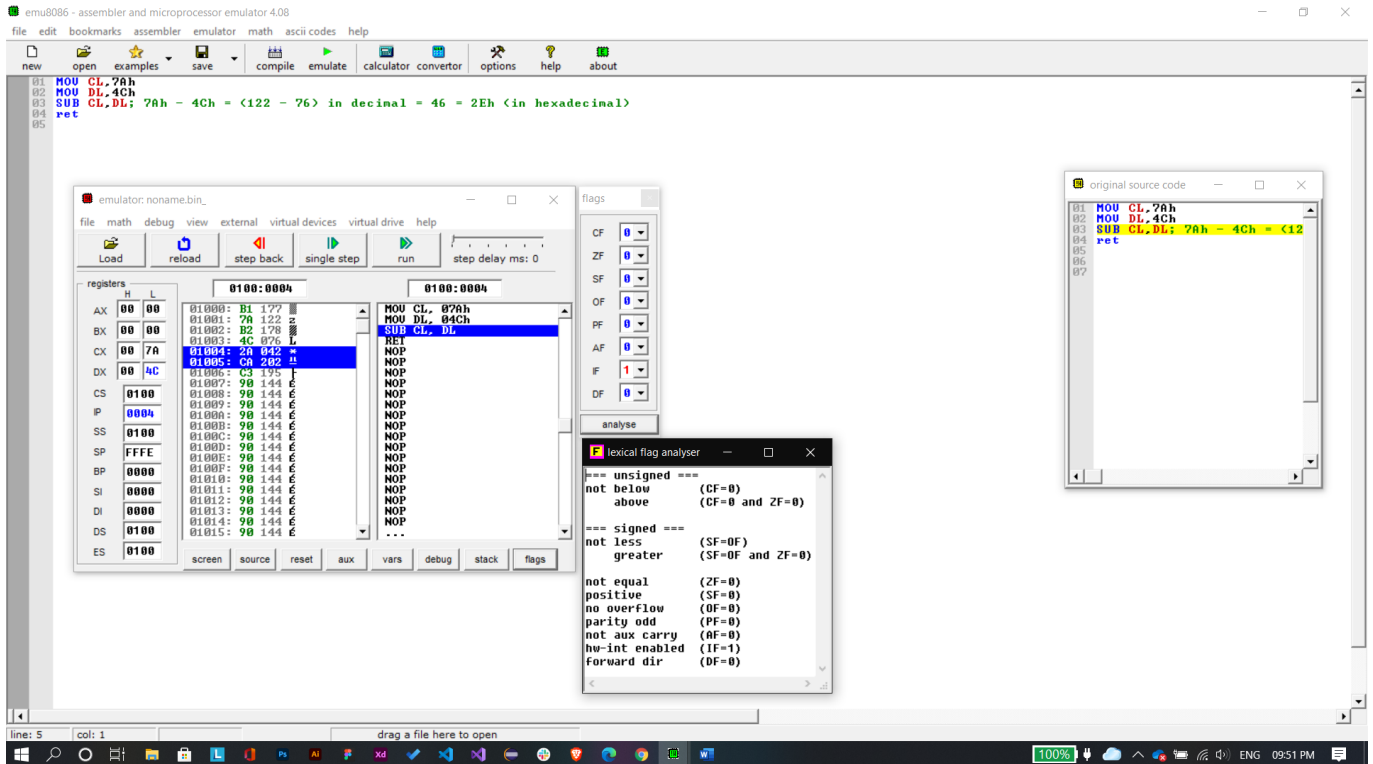
```
MOV AX,1A4Ch
MOV BX,0B1DEh
ADD AX,BX; 1A4Ch + B1DEh = (6732 + 45534) in decimal = 52266 = CC2Ah (in hexadecimal)
ret
```



3. 7Ah – 4Ch (using CL, DL registers)

Code:

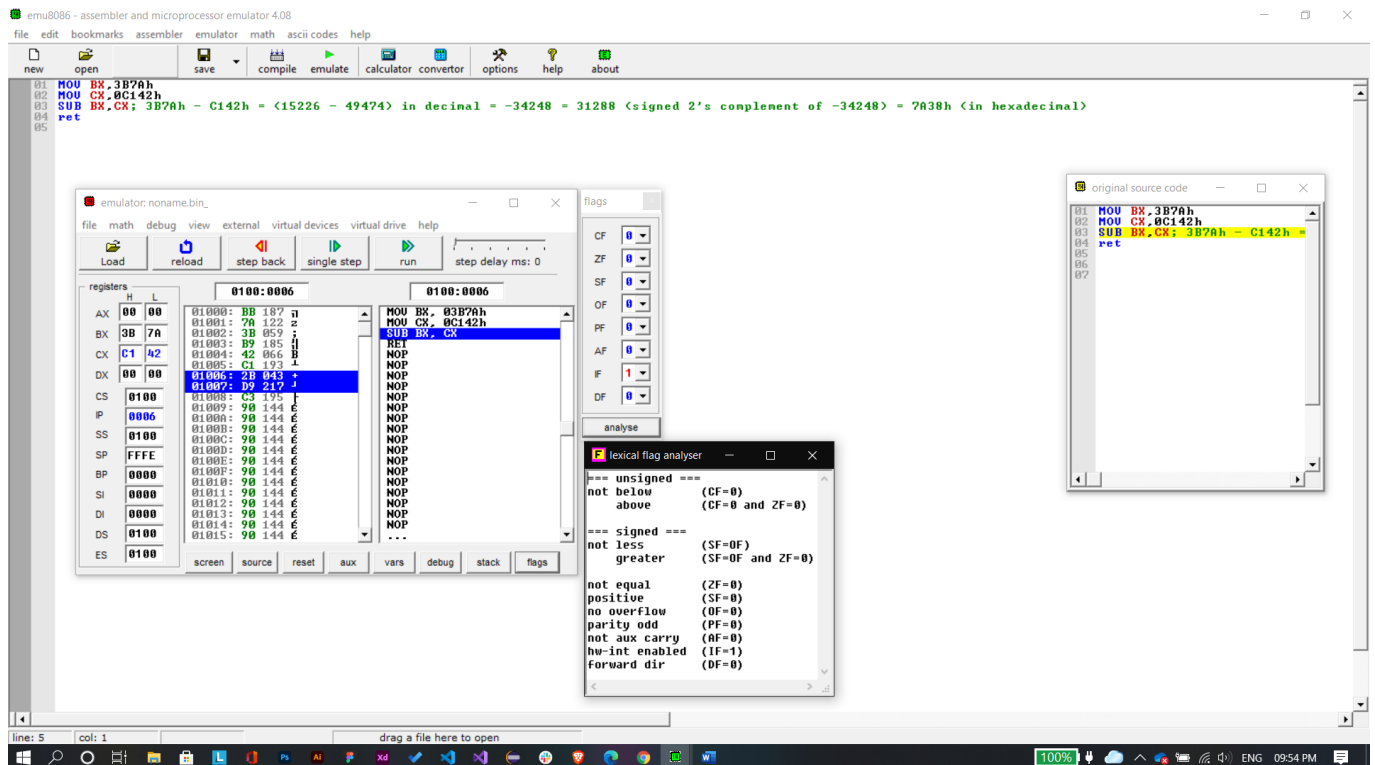
```
MOV CL,7Ah
MOV DL,4Ch
SUB CL,DL; 7Ah - 4Ch = (122 - 76) in decimal = 46 = 2Eh (in hexadecimal)
ret
```



4. 3B7AH + C142H (using BX, CX registers)

Code:

```
MOV BX, 3B7Ah
MOV CX, 0C142h
SUB BX, CX; 3B7Ah - C142h = (15226 - 49474) in decimal = 34248 = 31288 (signed 2's complement of -34248) = 7A38h (in hexadecimal)
ret
```



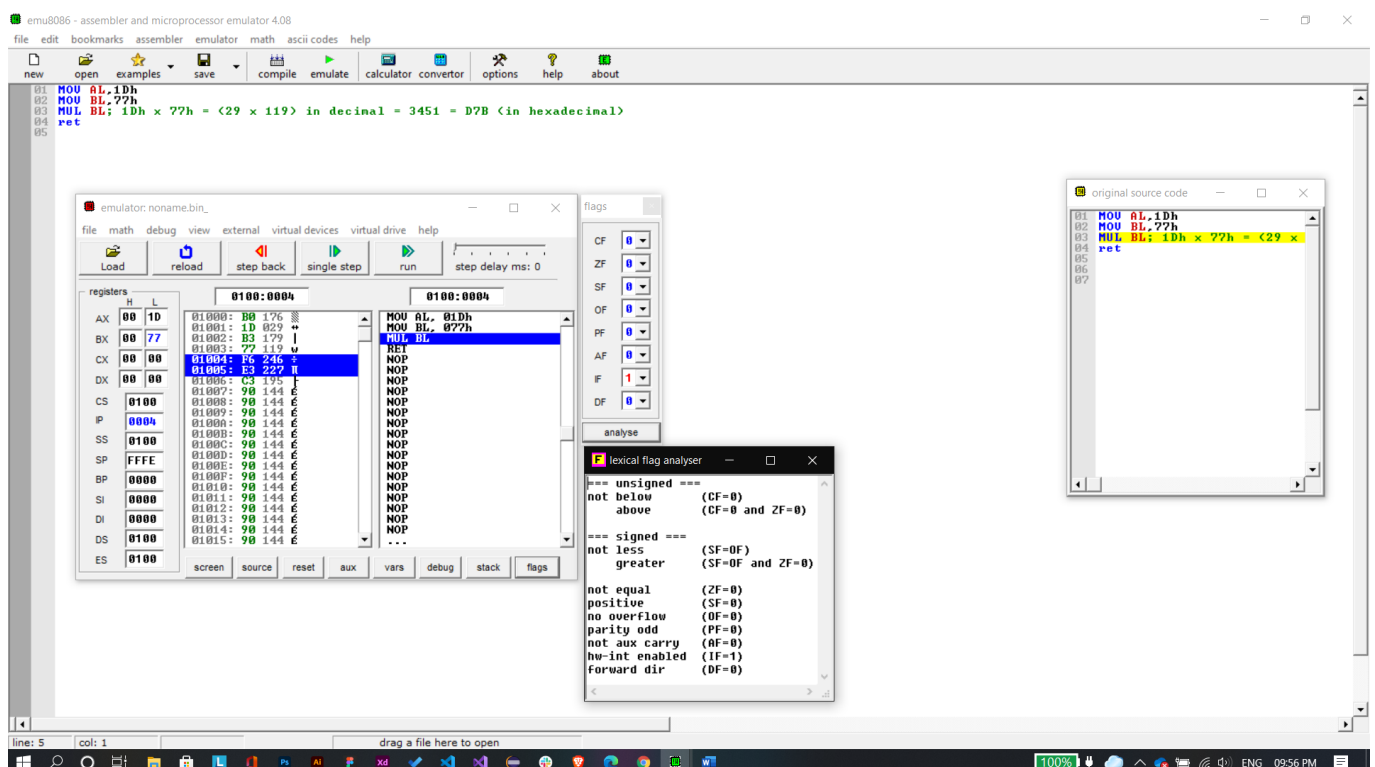
5. 1DH x 77H (using AL, BL registers)

Code:

```

MOV AL, 1Dh
MOV BL, 77h
MUL BL; 1Dh x 77h = (29 x 119) in decimal = 3451 = D7B (in hexadecimal)
ret

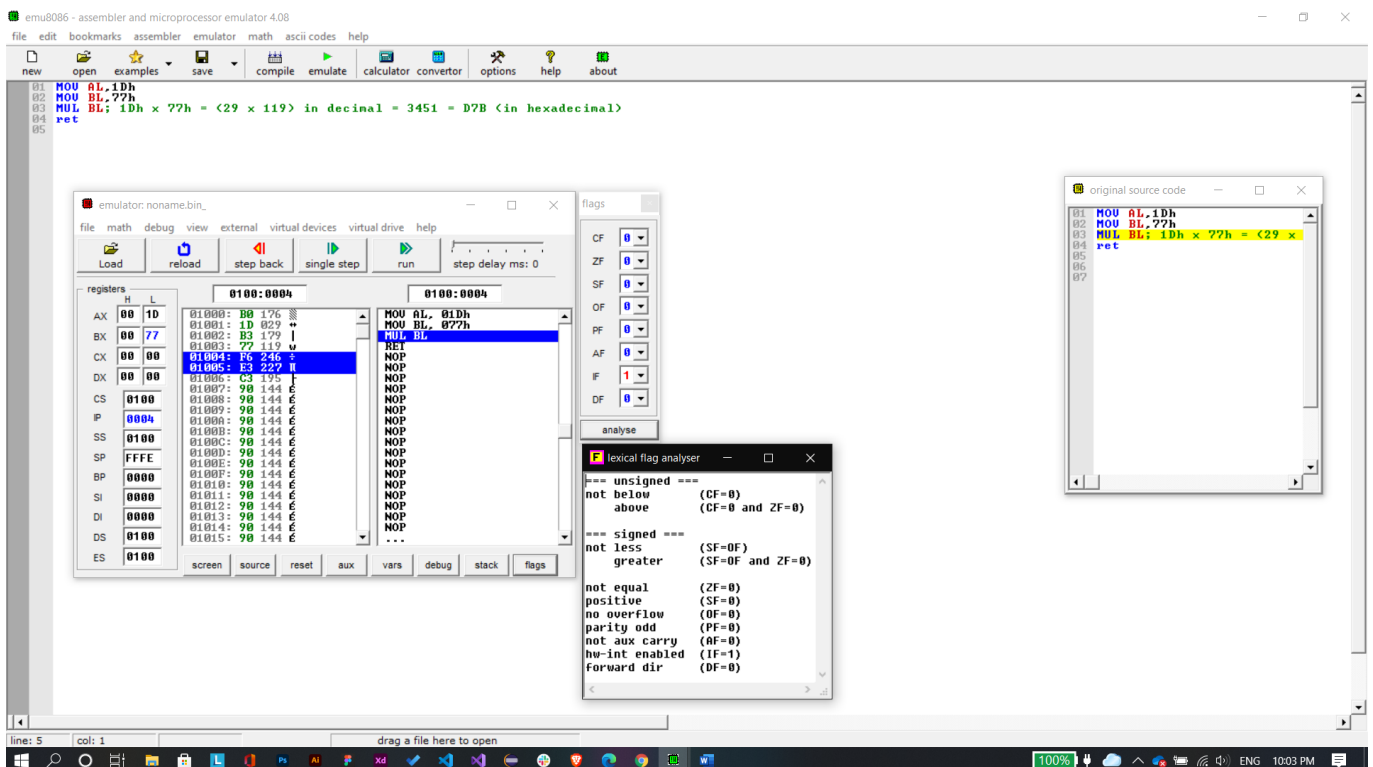
```



6. EF1AH x CD50H (using AX, BX registers) //product stored partly in DX and in AX

Code:

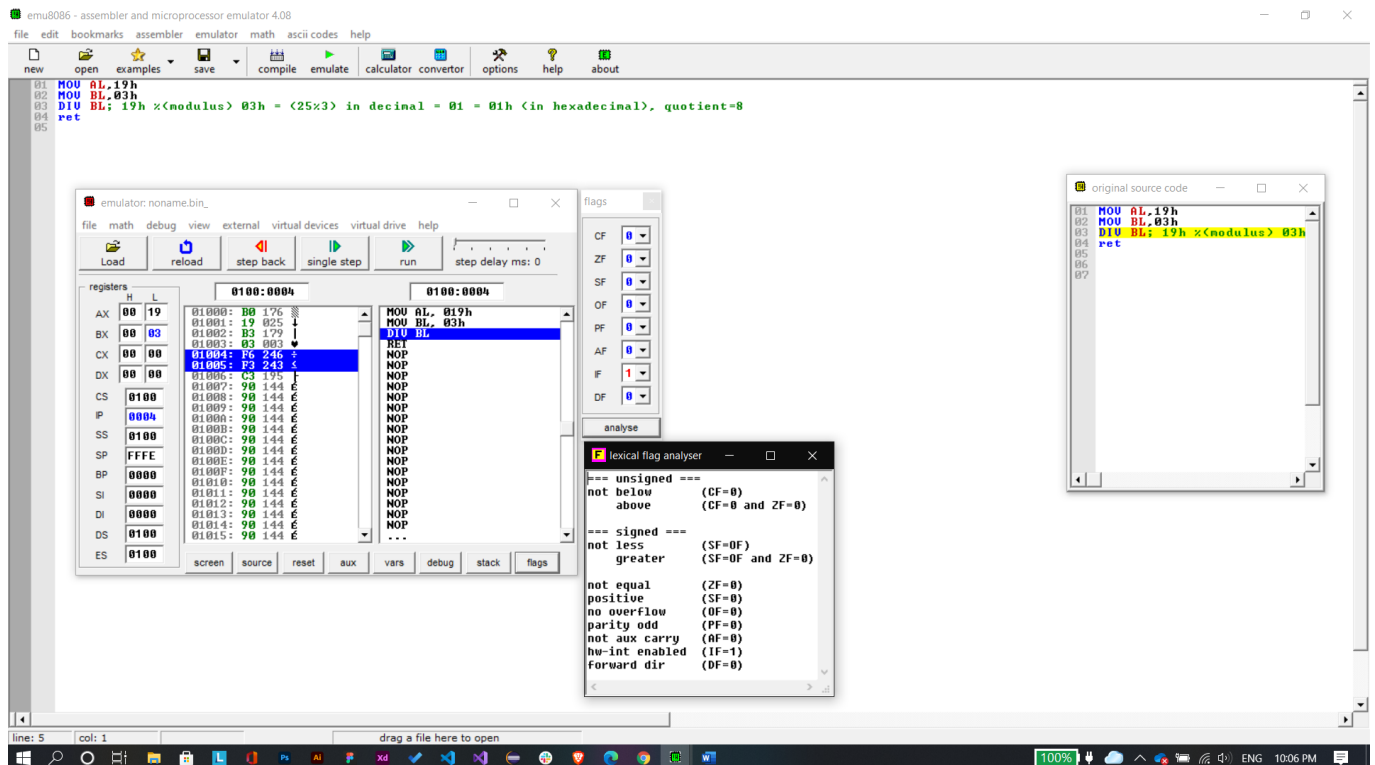
```
MOV AX,0EF1Ah
MOV BX,0CD50h
MUL BX; EF1Ah x CD50h = (61210 x 52560) in decimal = 3217197600 = BFC28A20h (in hexa
decimal)
ret
```



7. 19H ÷ 03H (using AL, BL registers) //here quotient is stored in AH and remainder in BL

Code:

```
MOV AL,19h
MOV BL,03h
DIV BL; 19h %(modulus) 03h = (25%3) in decimal = 01 = 01h (in hexadecimal), quotient
=8
ret
```



8. 1927H ÷ 1201H (using AX, BX registers) //here quotient is stored in AX; remainder in DX

```

MOV AX,1927h
MOV BX,1201h
DIV BX; 1927h %(modulus) 1201h = (6439%4609) in decimal = 1830 = 726h (in hexadecimal), quotient=1
ret

```

