Microprocessor and Assembly Language Lab

Lab Material 7_2 for CSE 312 (M&AL Lab)

Dr. Shah Murtaza Rashid Al Masud
Associate Prof.
Dept. of CSE, UAP

FLOW CONTROL INSTRUCTIONS

Today we will see how jump and loop instructions can be used in assembly language programming to make decisions or repeat sections of code

Table: Different Types of Jump Instructions

Type	Instruction		Meaning (jump if)	Condition
Unconditional	JMP		unconditional	None
Comparisons	JA	jnbe	above (not below or equal)	CF = 0 and ZF = 0
	JAE	jnb	above or equal (not below)	CF = 0
	JB	jnae	below (not above or equal)	CF = 1
	JBE	jna	below or equal (not above)	CF = 1 or ZF = 1
	JE	jz	equal (zero)	ZF = 1
	JNE	jnz	not equal (not zero)	ZF = 0
	JG	jnle	greater (not lower or equal)	ZF = 0 and SF = OF
	JGE	jnl	greater or equal (not lower)	SF = OF
	JL	jnge	lower (not greater or equal)	(SF xor OF) = 1 i.e. SF ≠ OF
	JLE	jng	lower or equal (not greater)	(SF xor OF or ZF) = 1
	JCXZ	loop	CX register is zero	(CF or ZF) = 0
Carry	JC		Carry	CF = 1
	JNC		no carry	CF = 0
Overflow	JNO		no overflow	OF = 0
	JO		overflow	OF =1
Parity Test	JNP	jpo	no parity (parity odd)	PF = 0
	JP	jpe	parity (parity even)	PF = 1
Sign Bit	JNS		no sign	SF = 0
	JS		sign	SF = 1
Zero Flag	JZ		zero	ZF = 1
	JNZ		non-zero	ZF = 0

High-level Language Branching Structures with Compound Conditions

AND Conditions

- AND

  An AND condition is true if and only if condition_1 and condition_2 are both true.

  Syntax

  condition_1 AND condition_2

Read a character, and if it's an uppercase letter, display it.

```
Read a character (into AL)
IF ('A' <= character) and (character <= 'Z')
   THEN
        display character
END_IF
```

Read a character, and if it's an uppercase letter, display it.

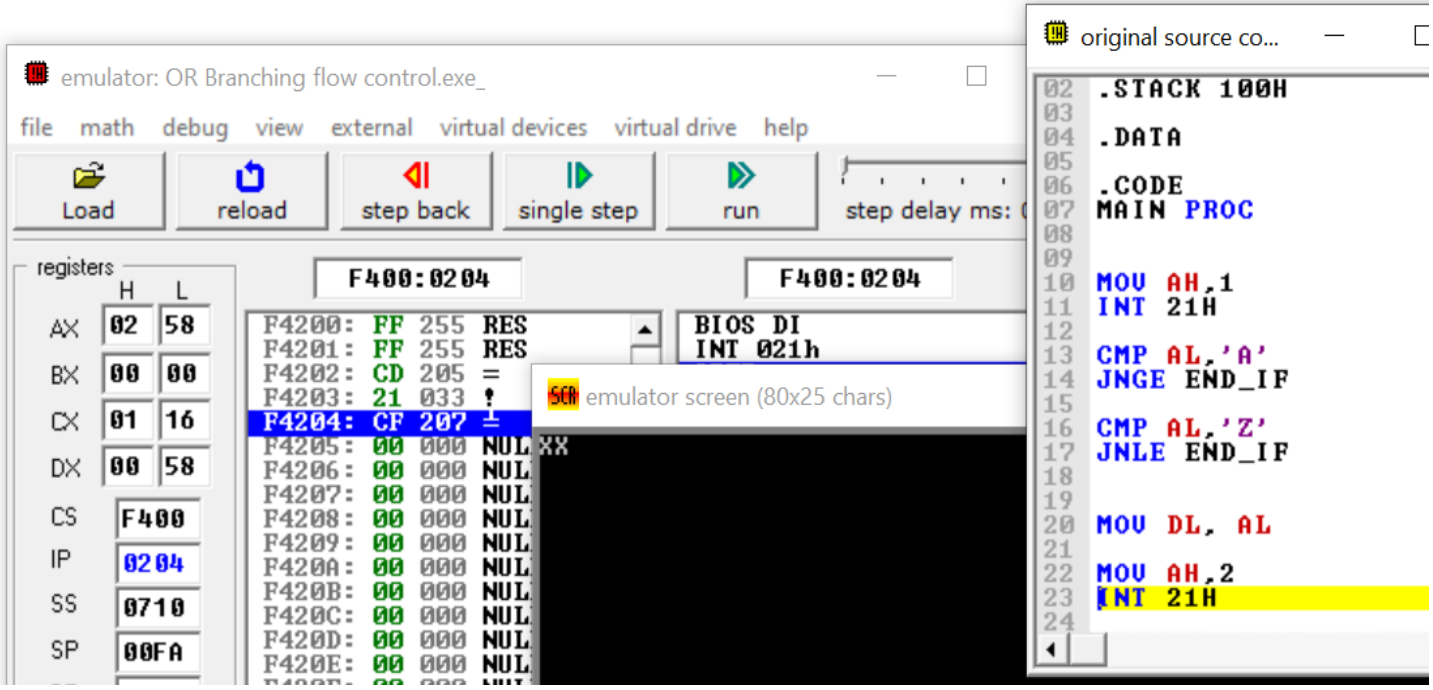
```
; read a character
            MOV     AH, 1                   ; prepare to read
            INT     21H                     ; char in AL
; if ('A' <= char) and (char >= 'Z')
          CMP       AL, 'A'                 ; char >= 'A'?
          JNGE      END_IF                  ; no, exit
             CMP    AL, 'Z'                 ; char <= 'Z'?
             JNLE   END_IF                  ; no, exit
; then display char
             MOV    DL, AL                  ; get char
             MOV    AH, 2                   ; prepare to display
          INT       21H                     ; display char
END_IF:
```

Another Example (AND execution)

```
.MODEL SMALL
.STACK 100H
.DATA

.CODE
MAIN PROC

    ;Read a character and if it is an uppercase letter
    ;display it.

    MOV AH,1
    INT 21H

    CMP AL,'A'
    JNGE RETURN

    CMP AL,'Z'
    JNLE RETURN

    MOV AH, 2
    MOV DL, AL
    INT 21H

    RETURN:
    MOV AH,4CH
    INT 21H



MAIN ENDP
```

Another Example (AND execution)

```
  .MODEL SMALL
 .STACK 100H

 .DATA

 .CODE
   MAIN PROC


MOV AH,1
INT 21H

CMP AL,'A'
JNGE END_IF

CMP AL,'Z'
JNLE END_IF


MOV DL, AL

MOV AH,2
INT 21H

END_IF:


          MOV AH, 4CH
     INT 21H
   MAIN ENDP
 END MAIN
```



OR Conditions

An OR condition is true if at least one of condition between condition_1 and condition_2 are true.

Syntax

condition_1 OR condition_2

Read a character, and if it is “y” or “Y”, display it; otherwise, terminate the program.

Read a character (into AL)
IF (character = ‘y’) OR (character = ‘Y’)
    THEN
        display it
    ELSE
        terminate the program
END_IF

Read a character, and if it is “y” or “Y”, display it; otherwise, terminate the program.

```
; read a character
        MOV  AH, 1              ; prepare to read
        INT    21H              ; char in AL
; if (character = ‘y’) or (character = ‘Y’)
        CMP    AL, ‘y’          ; char = ‘y’?
        JE       THEN           ; yes, go to display it
        CMP    AL, ‘Y’          ; char = ‘Y’?
        JE       THEN           ; yes, go to display it
        JMP     ELSE_           ; no, terminate
```

Read a character, and if it is “y” or “Y”, display it; otherwise, terminate the program.

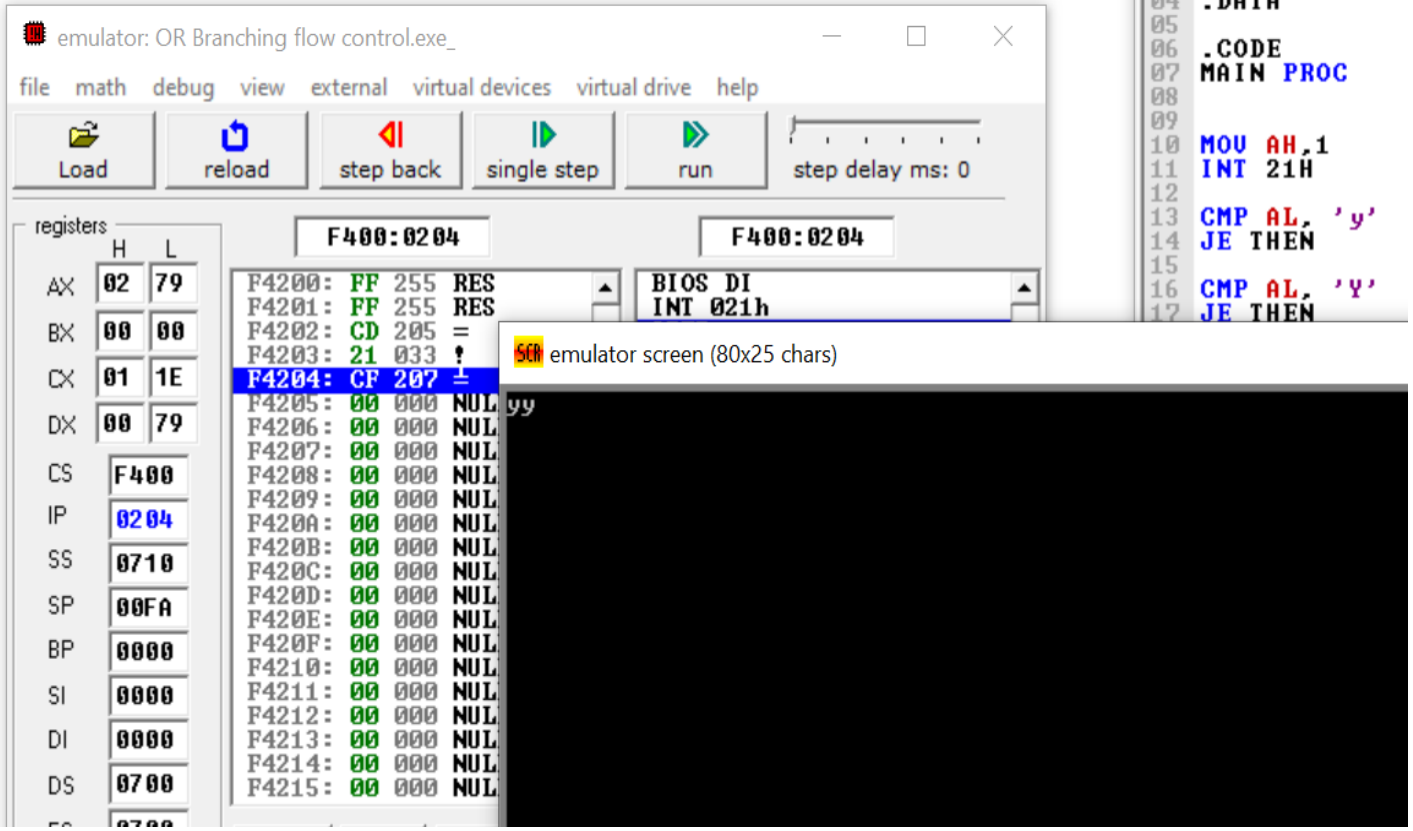
```
THEN:
        MOV AH, 2          ; prepare to display
        MOV DL, AL         ; get char
        INT   21H          ; display it
        JMP   END_IF       ; end exit
ELSE_:
        MOV AH, 4CH
        INT   21H          ; DOS exit
END_IF:
```

Another Example (OR execution)

```
01  .MODEL SMALL
02  .STACK 100H
03  .DATA
04  .CODE
05  MAIN PROC
06
07      ;Read a character and if it is 'y' or 'Y' display it
08      ;otherwise terminate the program.
09
10      MOV AH,1
11      INT 21H
12
13      CMP AL, 'Y'
14      JE DISPLAY
15
16      CMP AL, 'y'
17      JE DISPLAY
18
19      JMP RETURN
20
21
22      DISPLAY:
23      MOV AH,2
24      MOV DL,AL
25      INT 21H
26
27
28      RETURN:
29      MOV AH,4CH
30      INT 21H
31
32
33
34  MAIN ENDP
```

Another Example (OR execution)

```
.MODEL SMALL
.STACK 100H

.DATA

.CODE
  MAIN PROC


MOV AH,1
INT 21H

CMP AL, 'y'
JE THEN

CMP AL, 'Y'
JE THEN

JMP ELSE_

THEN:

MOV AH,2
MOV DL, AL
INT 21H

JMP END_IF


ELSE_:
MOV AH,4CH
INT 21H
END_IF:



        MOV AH, 4CH
    INT 21H
  MAIN ENDP
END MAIN
```



High-level Language Looping Structures

- FOR

  This is a loop structure in which the loop statements are repeated a known number of times. The counter for the loop is the register CX which is initialized to loop_count. Execution of LOOP instruction causes CX to be decremented automatically.

  Syntax

  LOOP destination_label

The LOOP Instructions:

The LOOP instruction is a combination of a DEC and JNZ instructions.

It causes execution to branch to the address associated with the LOOP instruction. The branching occurs a number of times equal to the number stored in the CX register.

Like the **conditional and unconditional jump instructions** which can be used to simulate the **IF-Then- Else structure** of any programming language, the **Loop instructions** can be used to simulate the **Repeat- Until and While-Do loops**.

LOOP Instruction (other than condition instruction)

LOOP destination_label

- The counter for the loop is the register *CX which is initialized to loop_count*
- Execution of LOOP instruction *causes CX to be decremented automatically.*

LOOP Instruction (for Loop)

Example:
Write a count-controlled loop and display a row of 80 stars.

Pseudocode Algorithm	**Assembly Code**

FOR 80 times DO
display '*'
END_FOR
	MOV CX,80
	MOV AH,2
	MOV DL,'*'
	TOP:
	INT 21H
	LOOP TOP

Write a count-controlled loop to display a row of 80 stars.

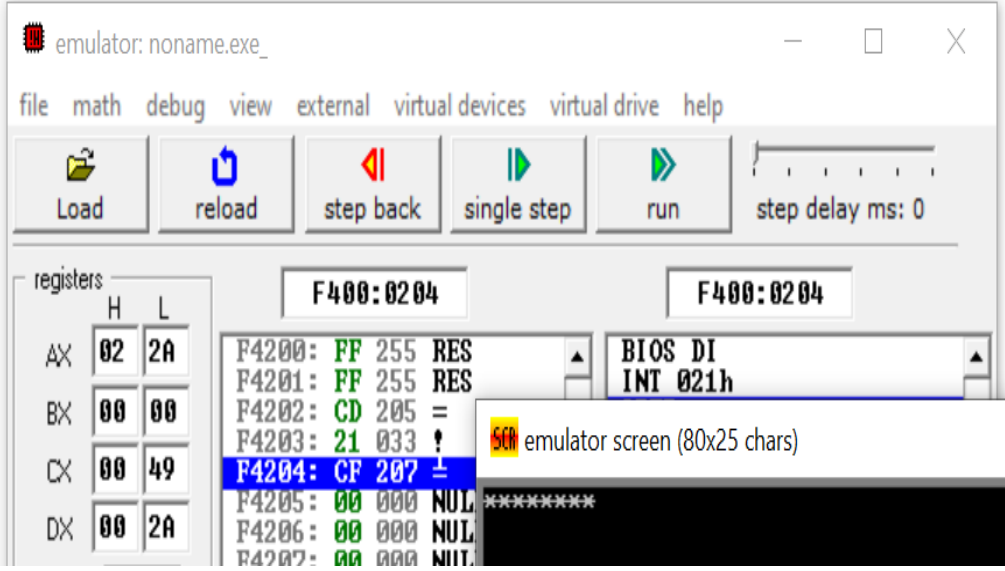
```
        MOV  CX, 80        ; number of stars to display
        MOV  AH, 2         ; display character function
        MOV  DL, '*'       ; character to display
TOP:
        INT  21h           ; display a star
        LOOP TOP           ; repeat 80 times
```

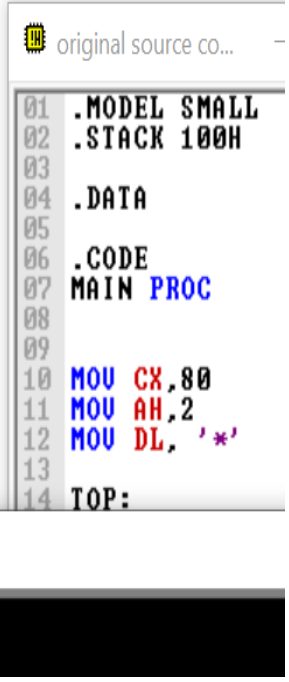
LOOP Instruction (for Loop)

```
01  .MODEL SMALL
02  .STACK 100H
03  .DATA
04  .CODE
05  MAIN PROC
06
07      ;display a row of 80 stars
08
09      MOV CX, 80
10
11      MOV AH, 2
12      MOV DL, '~'
13
14
15      TOP:
16      INT 21H
17
18      LOOP TOP
19
20
21
22  MAIN ENDP
```

LOOP Instruction (for Loop)

```
 .MODEL SMALL
.STACK 100H

.DATA

.CODE
  MAIN PROC


MOV CX,80
MOV AH,2
MOV DL, '*'

TOP:

INT 21H
LOOP TOP



      MOV AH, 4CH
    INT 21H


     MAIN ENDP
```





Examples

Example2:

The following program calculate the expression below and store the result in AL.

1+3+5+7+9→AL

```
 .MODEL SMALL
.DATA
.CODE
MAIN PROC
      MOV BL,1
      MOV  CX, 5
      MOV AL,0
L1: ADD AL,BL
      ADD BL,2
      LOOP L1
     NOP            ;AL=……..
     .EXIT
MAIN ENDP
END MAIN
```

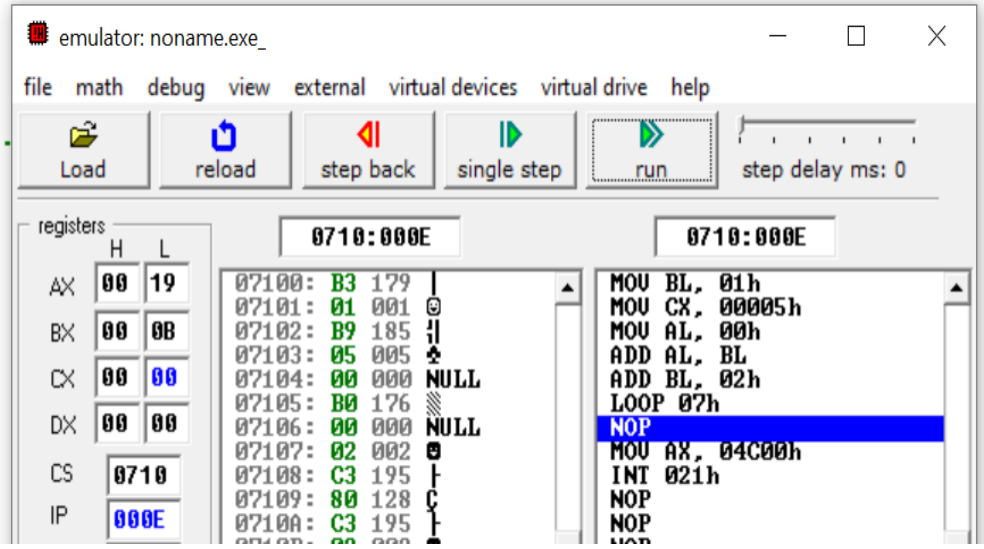
Examples

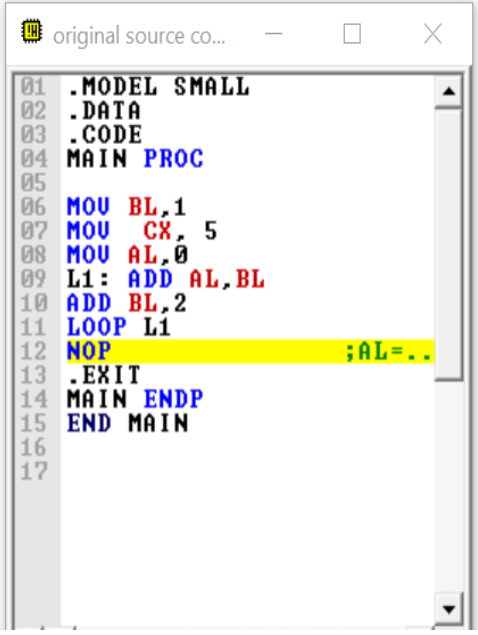
Example2:

The following program calculate the expression below and store the result in AL.

1+3+5+7+9→AL

```
.MODEL SMALL
.DATA
.CODE
MAIN PROC

        MOV BL,1
        MOV  CX, 5
        MOV AL,0
L1: ADD AL,BL
        ADD BL,2
        LOOP L1
       NOP                    ;AL=.....
        .EXIT
MAIN ENDP
END MAIN
```





Examples

Exercise 2:

The following program calculate the expression below and store the result in AX.

2*4*6→AL

Answer

```
TITLE ARRAY.ASM
.MODEL SMALL
.DATA
.CODE
MAIN PROC
      MOV AX, @DATA
      MOV DS, AX
      MOV BL,2
      MOV  CX, 3
      MOV AL,1
L1: MUL BL
      ADD BL,2
      LOOP L1
     NOP
     .EXIT
MAIN ENDP
END MAIN
```

Example

Summation of following series in 8086 microprocessor.
3+6+9+.....+42

Assembly code:

```
MOV AX,3
MOV BX,6
MOV CX,13
start:
ADD AX,BX
ADD BX,3
LOOP start
```

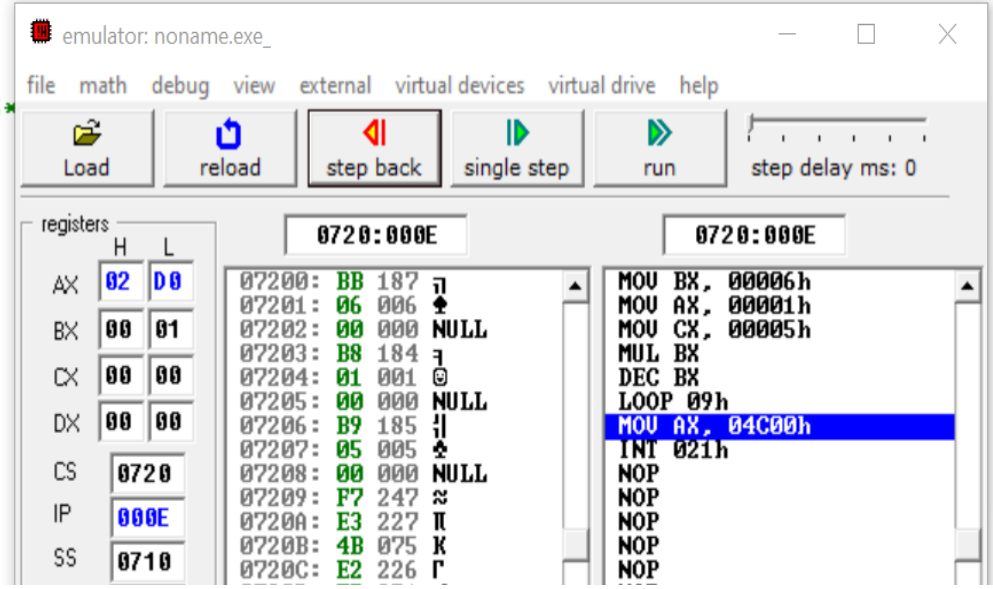
Example

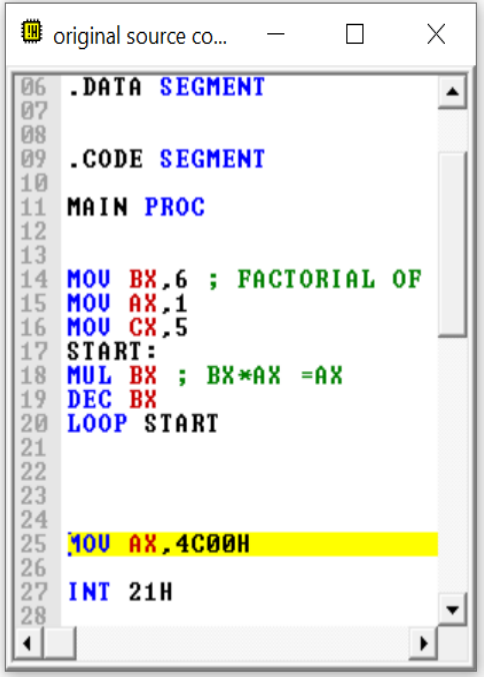
FIND THE FACTORIAL OF N in Intel 8086 microprocessor. ;
FACTORIAL OF N=1*2*3*...*N
Assembly code:

```
MOV BX,6 ; FACTORIAL OF 6!=1*2*3*4*5*6=720d MOV
AX,1
MOV CX,5
START:
MUL BX ; BX*AX =AX
DEC BX
LOOP START
```

Example

```
.MODEL SMALL
.STACK 100H
.DATA SEGMENT

.CODE SEGMENT
MAIN PROC

MOV BX,6 ; FACTORIAL OF 6!=1*2*3*4*5*
MOV AX,1
MOV CX,5
START:
MUL BX ; BX*AX =AX
DEC BX
LOOP START



        MOV AX,4C00H
        INT 21H

       MAIN ENDP

       END MAIN
```





WHILE LOOP and REPEAT LOOP

Example
Write some code to count the number of characters in an input line.

Pseudocode Algorithm	Assembly Code

Initialize count to 0
Read a character
WHILE character <> carriage_return DO
count=count+1
read a character
END_WHILE
	MOV DX,0
	MOV AH,1
	INT 21H
	WHILE_:
	CMP AL,0DH
	JE END_WHILE
	INC DX
	INT 21H
	JMP WHILE_
	END_WHILE:

Write some code to count the number of characters in an input line.

```
        MOV     DX, 0           ; DX counts characters (or CX)
        MOV     AH, 1           ; prepare to read
        INT     21H             ; character in AL
WHILE_:
        CMP     AL, 0DH         ; CR?
        JE      END_WHILE       ; yes, exit
        INC     DX              ; not CR, increment count
        INT     21H             ; read a character
        JMP     WHILE_          ; loop back
END_WHILE_:
```

Write some code to count the number of characters in an input line.

```
.MODEL SMALL
 .STACK 100H

 .DATA

 .CODE
   MAIN PROC


MOV CX,0

MOV AH,1
INT 21H

WHILE_:

CMP AL,0DH
JE END_WHILE

INC CX
INT 21H

JMP WHILE_

END_WHILE:

 MOV AH, 4CH
     INT 21H
   MAIN ENDP
 END MAIN
```