

Assembly Report (Lab 1)

Problem 1

- **The functionality of program:** calculates factorial using AX as an accumulator and CX as multiplier and loop counter
- **What happens to the AX and CX registers:** When the program starts, AX is initialized to 1 and CX is initialized to 5 (the number stored at 'data'). During each iteration of the loop, AX is multiplied by the current value of CX and CX is automatically decremented by 1 by the LOOP instruction. This continues until CX becomes zero. At the end of execution, AX contains 120 (0078h), which is the factorial of 5, and CX contains 0000h, indicating the loop has finished.

```
05 ORG 100h
06 MOU AX, 1
07 MOU CX, data
08 START_LOOP:
09 MUL CX
10 LOOP START_LOOP
11 data DW 0005h
12 END
13
```

- Code:

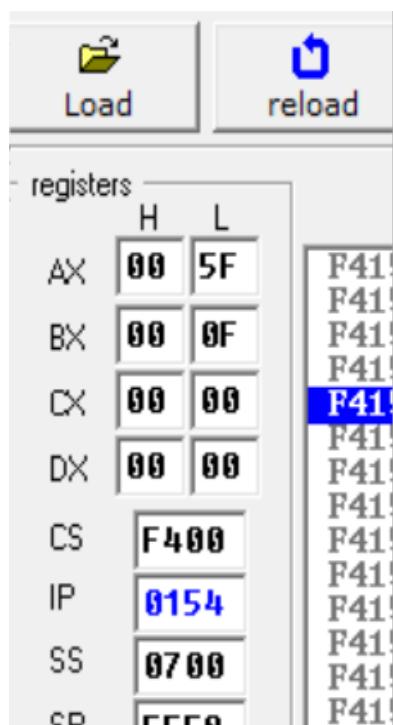
| registers | | |
|-----------|----|----|
| | H | L |
| AX | 00 | 78 |
| BX | 00 | 00 |
| CX | 00 | 00 |

| registers | | |
|-----------|----|----|
| | H | L |
| AX | 00 | 78 |
| BX | 00 | 00 |
| CX | 00 | 00 |
| DX | 00 | 00 |
| CS | 07 | 00 |
| IP | 01 | 0B |
| SS | 07 | 00 |
| SP | FF | FE |
| BP | 00 | 00 |
| SI | 00 | 00 |
| DI | 00 | 00 |
| DS | 07 | 00 |
| ES | 07 | 00 |

- Values of registers after execution:

Problem 2

Code and execution:



The screenshot shows a debugger interface with assembly code and register values. The assembly code is as follows:

```
; you may customize this and other sta
; The location of this template is c:\

org 100h

MOU AX,0      ;Sum
MOU BX,5      ;Start
MOU CX,10     ;Range

CMP CX,0       ;check if range=0?
JE END_PROGRAM ;If range=0,skip loop

SUM_LOOP:
ADD AX,BX      ;AX=AX+BX
INC BX         ;BX++
LOOP SUM_LOOP;CX--

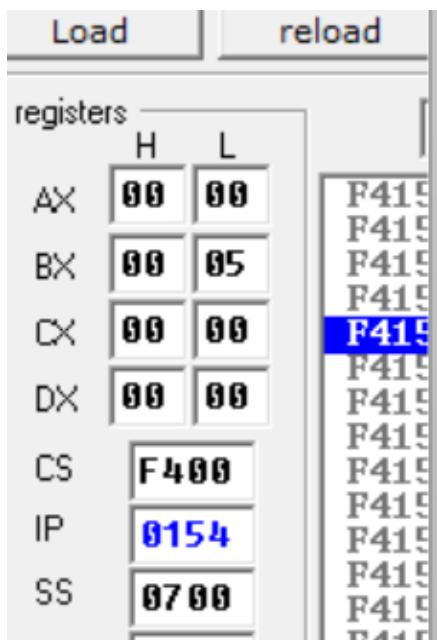
END_PROGRAM:
INT 20h        ;=Stop program
```

The registers window shows the following values:

| | H | L |
|----|------|----|
| AX | 00 | 5F |
| BX | 00 | 0F |
| CX | 00 | 00 |
| DX | 00 | 00 |
| CS | F400 | |
| IP | 0154 | |
| SS | 0700 | |
| CD | FFFF | |

After code execution: AX=5F (95 in decimal), BX=0F(15 in decimal), CX=0

- **Validations made** check if range=0 then Ax will be the same



The screenshot shows a debugger interface with assembly code and register values. The assembly code is identical to the previous one:

```
org 100h

MOU AX,0      ;Sum
MOU BX,5      ;Start
MOU CX,0      ;Range

CMP CX,0       ;check if range=0?
JE END_PROGRAM ;If range=0,skip loop

SUM_LOOP:
ADD AX,BX      ;AX=AX+BX
INC BX         ;BX++
LOOP SUM_LOOP;CX--

END_PROGRAM:
INT 20h        ;=Stop program
```

The registers window shows the following values:

| | H | L |
|----|------|----|
| AX | 00 | 00 |
| BX | 00 | 05 |
| CX | 00 | 00 |
| DX | 00 | 00 |
| CS | F400 | |
| IP | 0154 | |
| SS | 0700 | |
| CD | FFFF | |

Problem 3

Code and execution:

```
05
06 ORG 100h
07
08
09 MOU SI,500h ;SI=result memory
10 MOU BX,0 ;byte index
11 MOU CX,10 ;10 bytes to add
12 CLC ;Clear carry first
13
14 ADD_LOOP:
15 MOU AL,A[BX] ;AL=A[i]
16 ADC AL,B[BX] ;AL=AL+B[i]+carry
17 MOU [SI+BX],AL ;SI+i=AL
18 INC BX ;i++
19 LOOP ADD_LOOP ;CX--
20
21 ;Store final carry
22 MOU AL,0
23 ADC AL,0
24 MOU [SI+BX],AL
25
26 INT 20h
27
28 A DB 0C7h,096h,047h,04Eh,046h,082h,00Fh,022h,0BDh,0CFh
29 B DB 00Dh,0EFh,007h,06Dh,0BAh,07Ch,01Eh,06Bh,000h,040h
30
```

 Random Access Memory

500

update

table

list

| | | | | | | | | | | | | | | | | |
|-----------|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|----|-----|
| 0700:0500 | D4 | 85 | 4F | BB | 00 | FF | 2D | 8D-BD | 0F | 01 | 00 | 00 | 00 | 00 | 00 | EAC |
|-----------|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|----|-----|

- After code execution: result stored in memory address 500 is:
(1 0F BD 8D 2D FF 00 BB 4F 85 D4)

The result stored in memory appears reversed because the 8086 processor uses little-endian format, meaning the least significant byte is stored at the lowest memory address. Reversing the stored result yields the number in normal most-to-least significant byte order, matching the expected output

Different test Cases:

- A and B = 0:

```
27  
28 A DB 0h,0h,0h,0h,0h,0h,0h,0h,0h,0h  
29 B DB 0h,0h,0h,0h,0h,0h,0h,0h,0h,0h  
30
```

Random Access Memory

500 update table list

0700:0500 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00

- A and B = FF:

```
26 *** ***  
27  
28 A DB 0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh  
29 B DB 0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh  
30
```

Random Access Memory

500 update table list

0700:0500 FE FF FF FF FF FF FF FF-FF FF 01 00 00 00 00 00 00 00 00

- A and B = random values:

```
28 A DB 0C7h,096h,047h,04Eh,046h,082h,00Fh,022h,0BDh,0CFh  
29 B DB 00Dh,0EFh,007h,06Dh,0BAh,07Ch,01Eh,06Bh,000h,040h  
30
```

Random Access Memory

500 update table list

0700:0500 D4 85 4F BB 00 FF 2D 8D-BD 0F 01 00 00 00 00 00 00