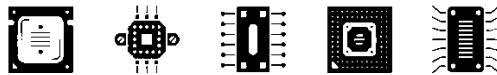


CSE 2006

MICROPROCESSOR AND INTERFACING



Task – 3

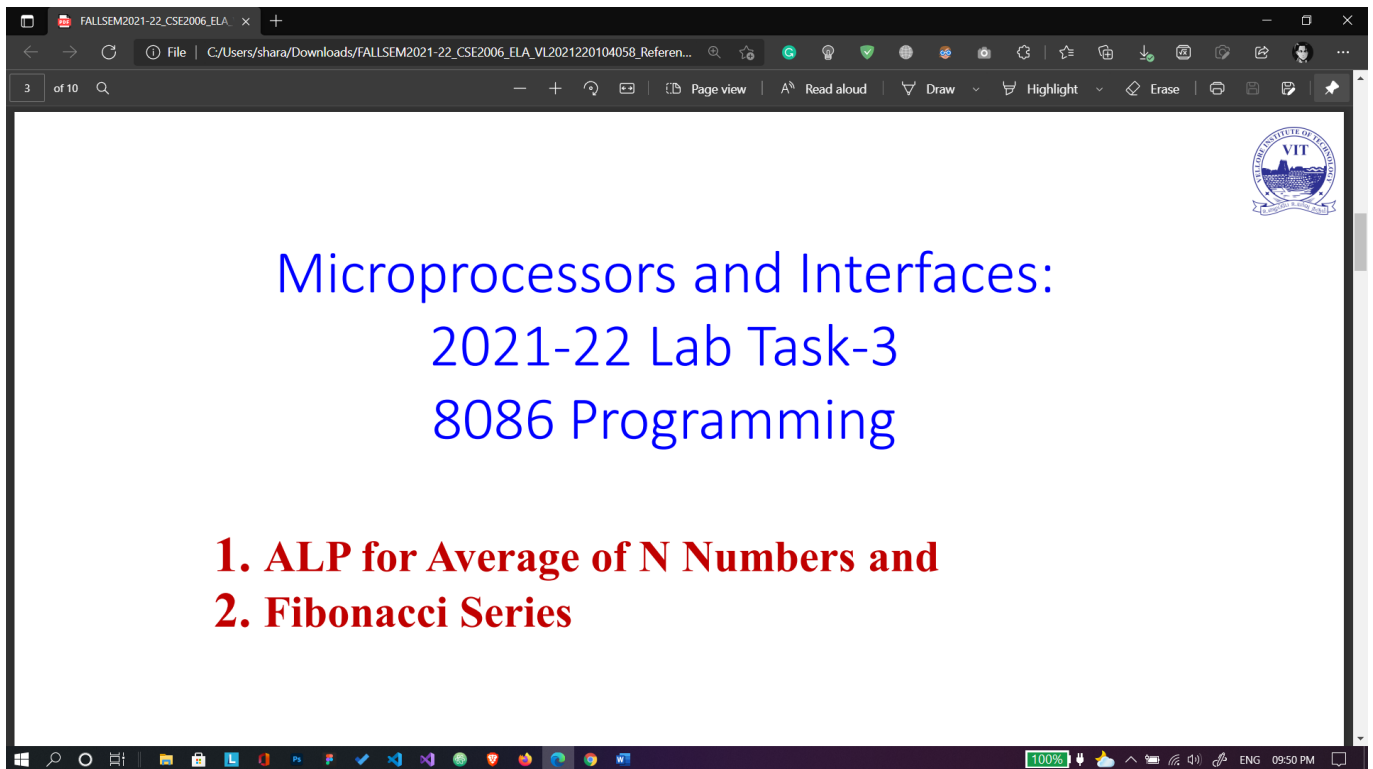
L11+L12 | SJT516

FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI

19BCE2105



Microprocessors and Interfaces:
2021-22 Lab Task-3
8086 Programming

1. ALP for Average of N Numbers and
2. Fibonacci Series

Experiment 1: ALP for Avg of N numbers

Aim: To find the average of N numbers stored in 8086 ALP.

Algorithm:

$$Average = \frac{\sum_{i=1}^N i}{N}$$

Input: Values in decimal: 03, 04, 05, 08

Code:

```
org 100h

MOV [1100H],03
MOV [1101H],04
MOV [1102H],05
MOV [1103H],08
MOV AX,0000H
MOV SI,1100H
MOV CX,04H ;count of numbers inserted
MOV DX,00H

loop:ADD AL,[SI]
INC SI
INC DX
CMP CX,DX
JNZ loop
DIV CL
```

```

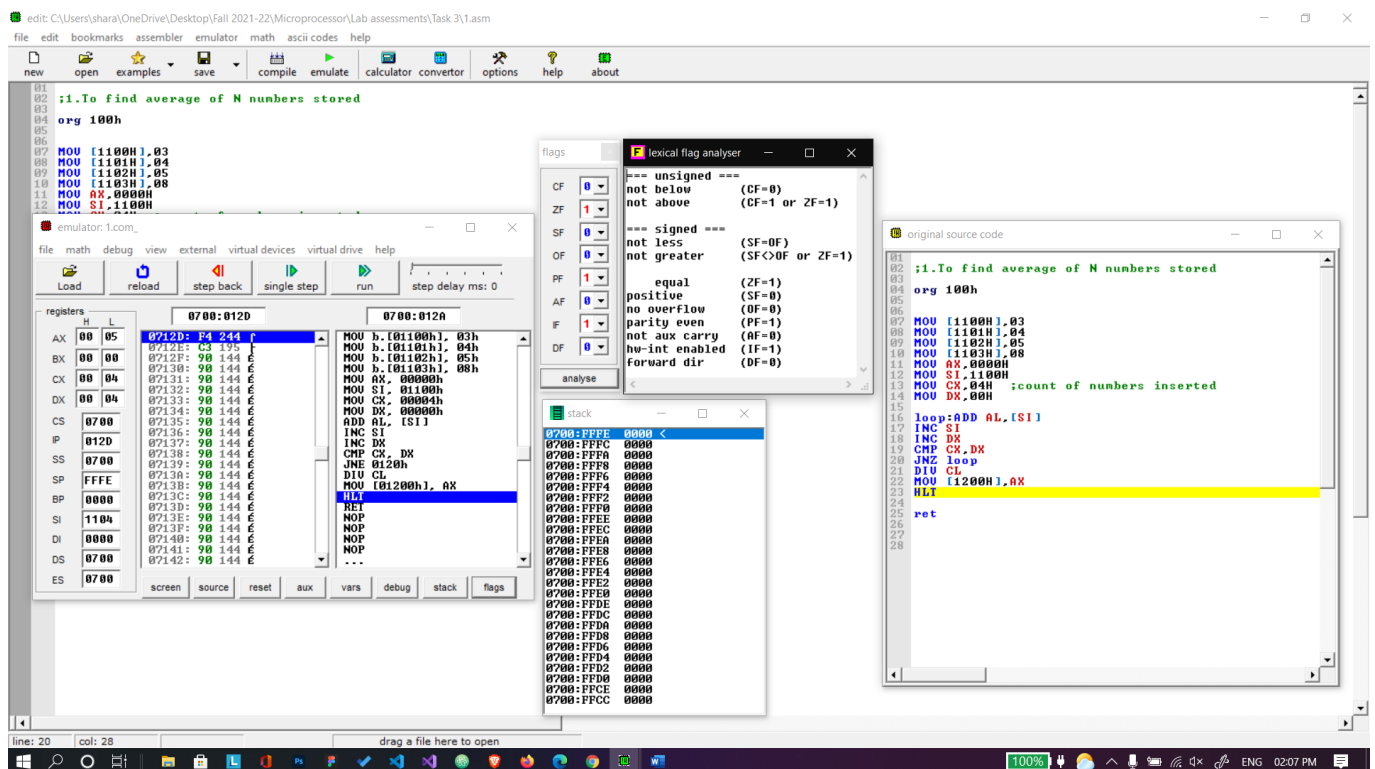
MOV [1200H], AX
HLT

ret

```

Output: Result in decimal: 05
 SI register location: 1100H
 Result location: 1200H

Screenshot:



Review Questions:

1. What if the accumulator is not initialized to zero?

If the accumulator was not initialised to zero (i.e. MOV AX,0000H), then the sum of the numbers would be increased beyond their original sum due to the newly inserted number in the accumulator. For example, if MOV AX,0100H, then the output for the above code would have been $(3+4+5+8+256(100H \text{ in decimal})/4) = 69$, which is obviously the wrong output. We add zero for a continuous sum due to its additive property to keep the result unaltered.

2. Does it have any effect on output result if the accumulator is not initialized to zero?

No, it does have an effect on the output. Otherwise, the output would have been $(3+4+5+8+0)/4 = 5$, instead of $(3+4+5+8+256)/4 = 69$ [If we consider MOV AX,0100H].

3. Repeat the problem with five numbers of 16-bit data of your choice. What are the changes you had to make in your code?

```

org 100h

MOV [1100H],69
MOV [1101H],96
MOV [1102H],256
MOV [1103H],121
MOV [1104H],482
MOV AX,0000H
MOV SI,1100H
MOV CX,05H ;count of numbers inserted
MOV DX,00H

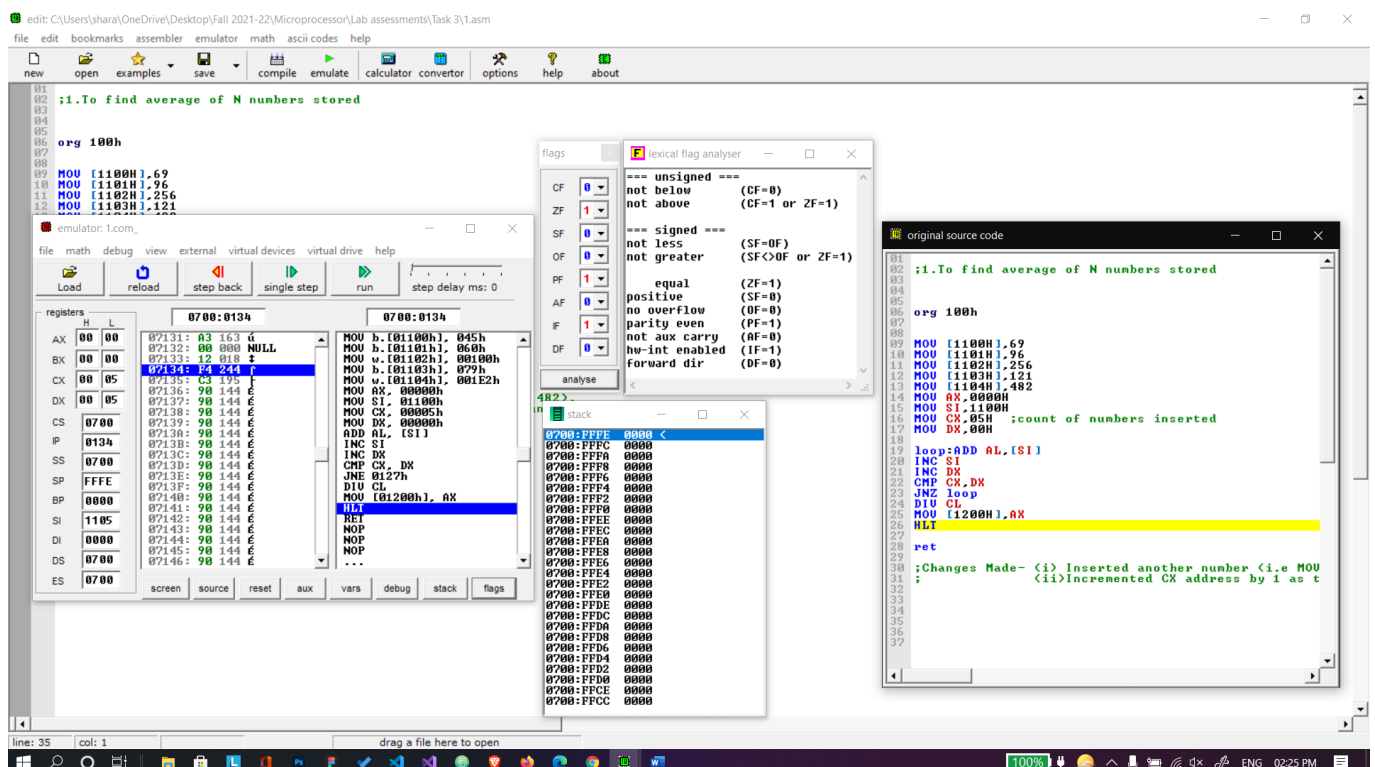
loop:ADD AL,[SI]
INC SI
INC DX
CMP CX,DX
JNZ loop
DIV CL
MOV [1200H],AX
HLT

ret

```

Changes made:

- Inserted another number (i.e MOV [1104H],482).
- Incremented CX address by 1 as total count of numbers is equal to 5.



Experiment 2: Fibonacci Series

Aim: To generate Fibonacci series using an 8086 program.

Algorithm:

- To generate Fibonacci sequence, I'm putting the 00H and 01H into memory at first.
- Then we are taking the limit from location offset 500.
- The limit is decreased by 2 at first, because 00H and 01H is already present there.
- Now I'm taking number from previous location, then add it with the value of current location, after that storing the result into next location.

Input:

Address	Data
...	...
500	E
...	...

Code:

```
org 100h

MOV [600H],10 ;setting a limit for the loop instruction to run
accordingly
MOV AL,00H
MOV SI,500H
MOV [SI],AL
INC SI
ADD AL,01H
MOV [SI],AL
MOV CX,[600H]
SUB CX,0002H

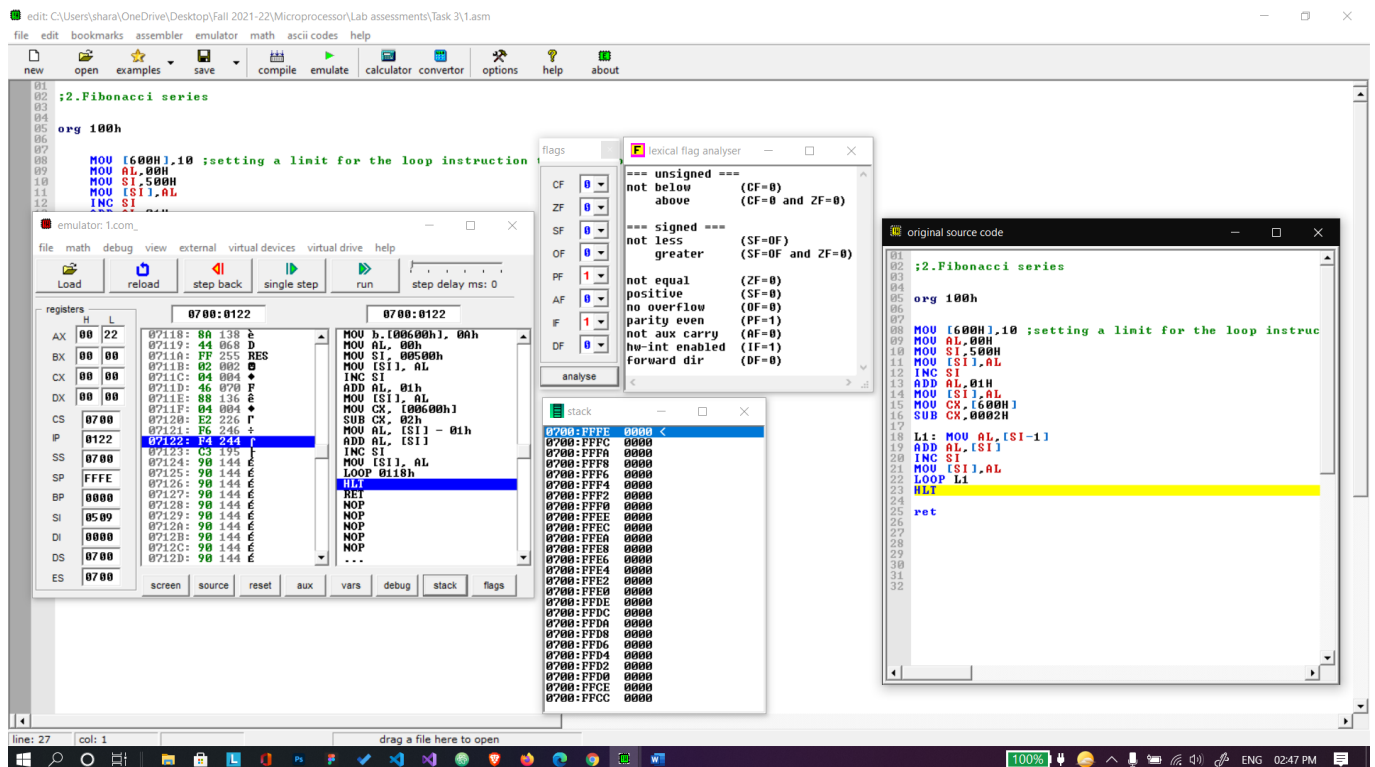
L1: MOV AL,[SI-1]
ADD AL,[SI]
INC SI
MOV [SI],AL
LOOP L1
HLT

ret
```

Output:

Address	Data
...	...
600	00
601	01
602	01
603	02
604	03
605	05
606	08
607	0D
608	15
609	22
60A	37
60B	59
60C	90
60D	E9
...	...

Screenshot:



Review questions:**1. Why is sub instruction used?**

We use the SUB instruction in order to explicitly load the first 2 elements of the sequence (i.e., 0 and 1).

2. What is the replaced number instead of "xx" and why is it so?

The replaced number instead of "xx" is 00h. This is assigned to the accumulator to use the additive property in order to produce an unaltered continuous sum until the loop exits.

3. Which addressing mode is used in this programming?

Immediate Addressing Mode is use in this ALP.

4. What is the role of LOOP instruction in this ALP?

Function of the LOOP is as follows:

L1: This defines the start of the loop (A label is created)

MOV AL, [SI-1]: Moves the element in the (i-1)th position into AL

ADD AL, [SI]: Moves the ith element with the (i-1)th element already present in AL

ADD SI, 1: Increment SI to point to the next position

MOV [SI], AL: Store the sum in the new position

LOOP L1: The instructions between label L1 and this LOOP instruction are executed "CX" times
