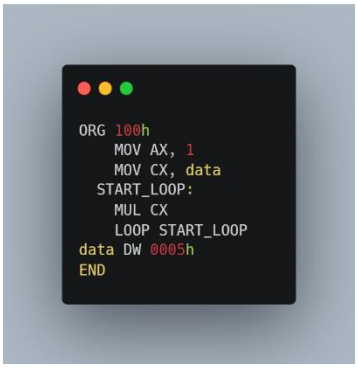
Alexandria University
Faculty of Engineering Computer and Systems Engineering Dept.
Computer Organization

Lab #1 Report

Name	ID
يوسف وليد عبد الوهاب	23011017

Problem 1:



Tracing this code, I can tell that it calculates the factorial of the number inside C_X. It defines a variable data = 5_{16} , Puts 1 in the Accumulator register and puts the value of data inside the C_X register, then it multiplies C_X by the value inside Ax, puts the result in Ax and decrements C_X until it's 0_{16}.

After execution: A_X = 00 78 which is 5! = 120_{10} and C_X = 00 00

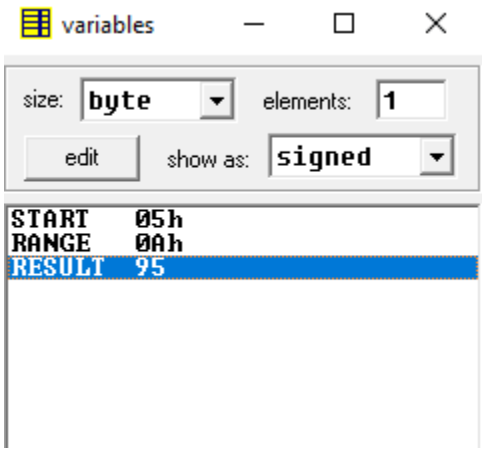
NOTE: the code is missing a HLT or RET to exit after getting the factorial, but it does the job in single step mode.

Problem 2 Code:

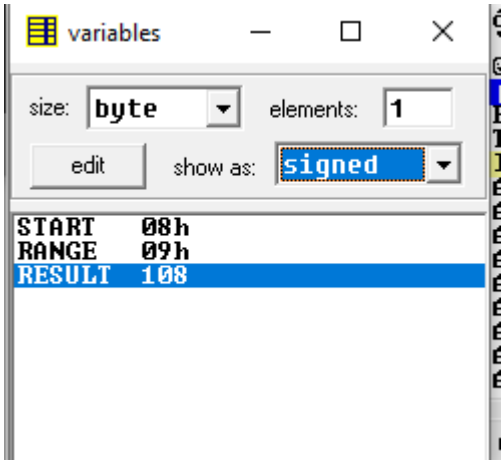
```
ORG 100h

;Reading range and start from user
XOR BX, BX

ReadStart:

MOV AH, 01h
INT 21h

CMP AL, 13
JE start_Done

SUB AL, 30h

MOV DL, AL
MOV AL, 10
MUL BL
MOV BL, AL
ADD BL, DL

JMP ReadStart


start_Done:
MOV start,BL


XOR BX, BX

ReadRange:

MOV AH, 01h
INT 21h

CMP AL, 13
JE range_Done

SUB AL, 30h

MOV DL, AL
MOV AL, 10
MUL BL
MOV BL, AL
ADD BL, DL

JMP ReadRange


range_Done:
MOV range,BL

XOR AX,AX

;Start of the Summation logic
MOV CL,range

CMP CL, 0
JLE INVALID_RANGE


MOV DL,start
MOV AL,start
ADD AL, range
JO OVERFLOW
MOV AL,0

LOOP_START:
ADD AL,DL
JO OVERFLOW
INC DL
LOOP LOOP_START
MOV result,AL
OVERFLOW:
INVALID_RANGE:
HLT

start DB ?
range DB ?
result DB ?

END
```

This code reads inputs from user for the start then range by reading character by character inside a loop and adding them accordingly to form the required byte. Then it loops till the range is done summing the numbers from start to start+range-1, The result is stored in a variable called result. This code handles checking that the range is more than zero, start + Range doesn't overflow the byte and Result overflowing, but for the sake of simplicity the user input assumes that the input is always a positive number.

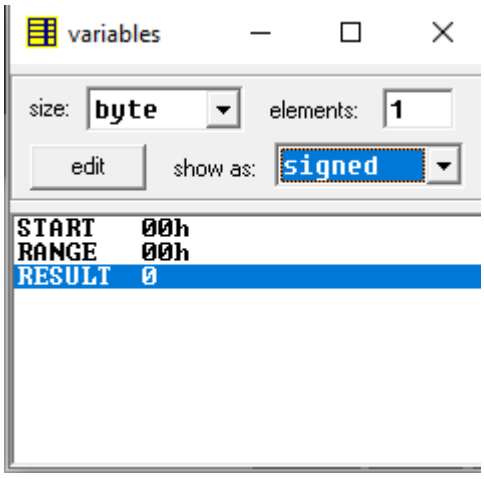
Running the example in the lab:



Another example



Another example

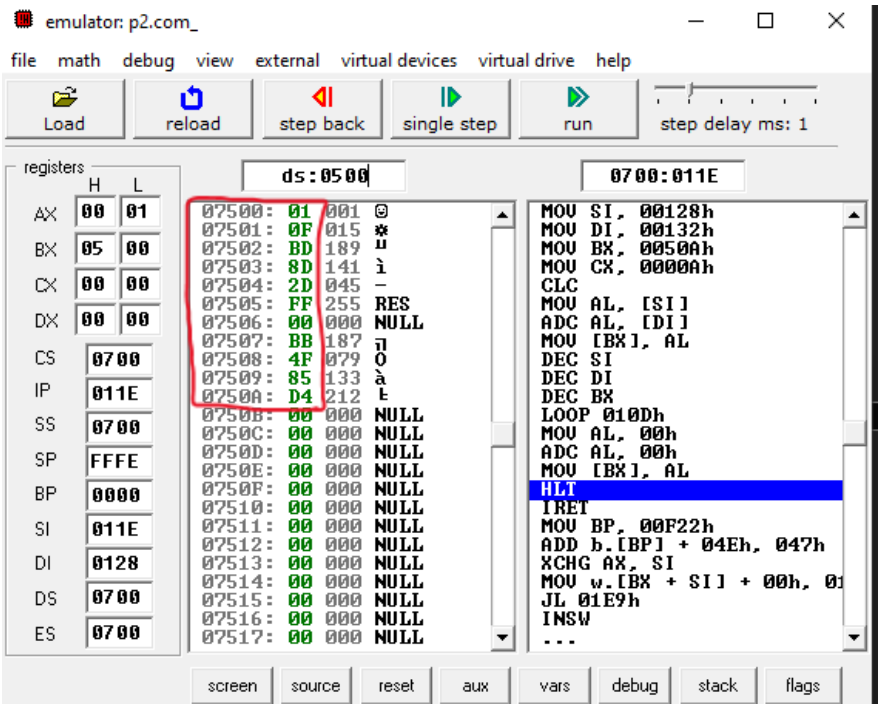


Problem 3 Code:

```
ORG 100h



MOV SI, OFFSET first+9
MOV DI, OFFSET second+9
MOV BX, OFFSET [0500h]+10
MOV CX, 10
CLC
ADD_LOOP:
MOV AL,[SI]
ADC AL,[DI]
MOV [BX],AL

DEC SI
DEC DI
DEC BX
LOOP ADD_LOOP

MOV AL,0
ADC AL,0
MOV [BX],AL

HLT
first DB
0CFh,0BDh,022h,0Fh,082h,046h,04Eh,047h,
096h,0C7h
second DB
040h,000h,06Bh,01Eh,07Ch,0BAh,06Dh,007h
,0EFh,0Dh

END
```

“Assuming the required address to store the result is 500_{16}

Running the example in lab this is the result in memory:

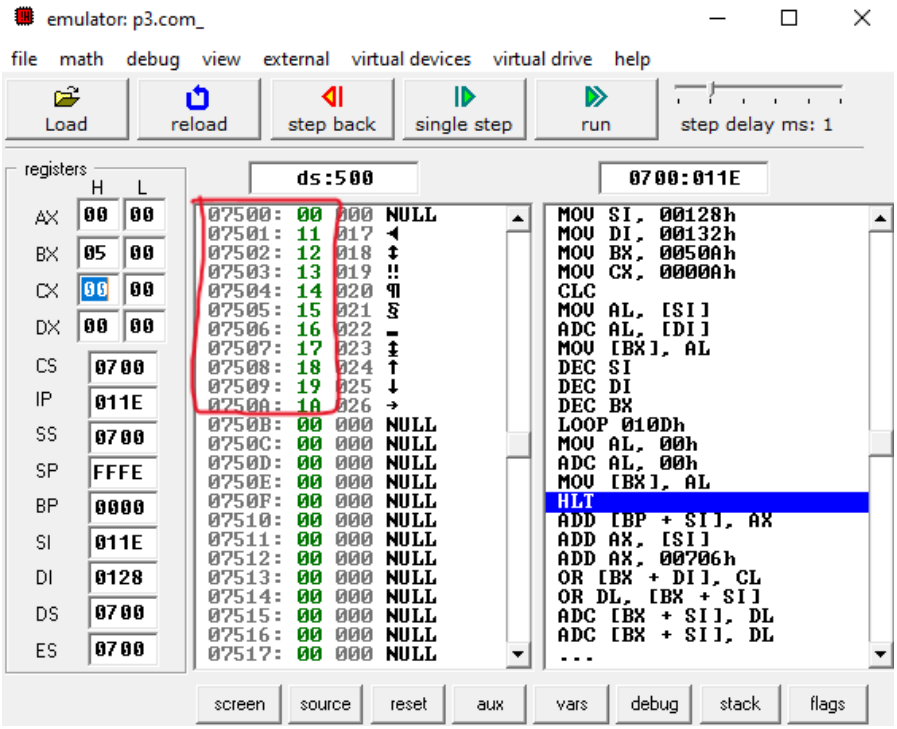


Example 2:

01 02 03 04 05 06 07 08 09 0A

+10 10 10 10 10 10 10 10 10 10

=00 11 12 13 14 15 16 17 18 19 1A



Example 3:

FF FF FF FF FF FF FF FF FF FF

+ FF FF FF FF FF FF FF FF FF FF

= 01 FF FF FF FF FF FF FF FF FF FE