



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

Lab #1 Report

Name	ID
يوسف وليد عبد الوهاب	23011017

Problem 1:

```
ORG 100h
MOV AX, 1
MOV CX, data
START_LOOP:
    MUL CX
    LOOP START_LOOP
data DW 0005h
END
```

Tracing this code, I can tell that it calculates the factorial of the number inside Cx. It defines a variable data = 5₁₆ , Puts 1 in the Accumulator register and puts the value of data inside the Cx register, then it multiplies Cx by the value inside Ax, puts the result in Ax and decrements Cx until it's 0₁₆.

After execution: Ax = 00 78 which is 5! = 120₁₀ and Cx = 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

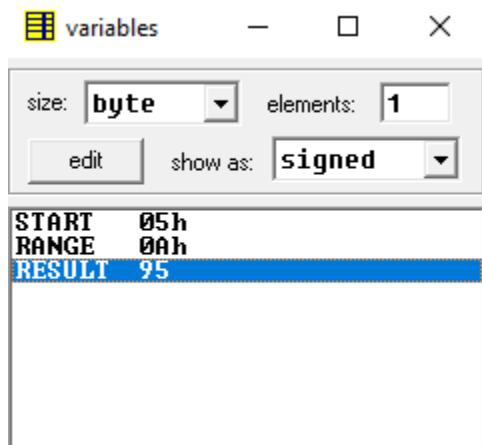
LOOP_START:
ADD AL,DL
JO OVERFLOW
INC DL
LOOP LOOP_START
MOV result,AL
OVERFLOW:
INVALID_RANGE:
HLT

start DB ?
range DB ?
result DB ?

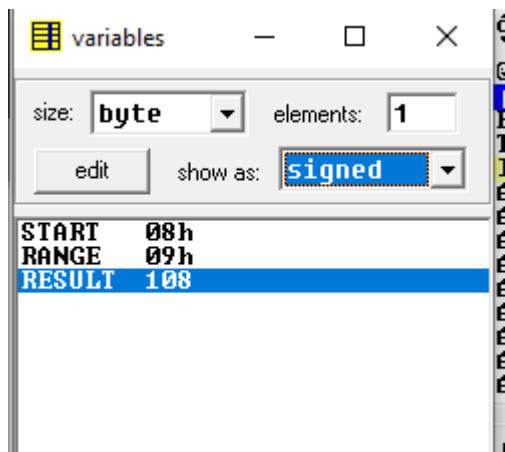
END
```

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.

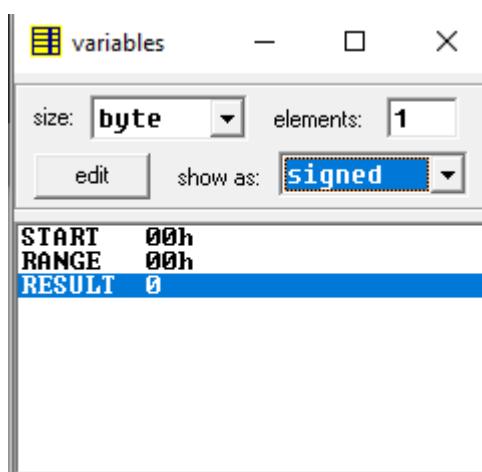
Running the example in the lab:



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 500₁₆

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

```

emulator: p2.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 1
registers H L
AX 00 01
BX 05 00
CX 00 00
DX 00 00
CS 0700
IP 011E
SS 0700
SP FFFE
BP 0000
SI 011E
DI 0128
DS 0700
ES 0700
ds : 0500 0700:011E
07500: 01 001 @
07501: 0F 015 *
07502: BD 189 u
07503: 8D 141 i
07504: 2D 045 -
07505: FF 255 RES
07506: 00 000 NULL
07507: BB 187 l
07508: 4F 079 o
07509: 85 133 à
0750A: D4 212 e
0750B: 00 000 NULL
0750C: 00 000 NULL
0750D: 00 000 NULL
0750E: 00 000 NULL
0750F: 00 000 NULL
HLT
TRET
MOU BP, 00F22h
ADD b.[BP] + 04Eh, 047h
XCHG AX, SI
MOU w.[BX + SI] + 00h, 01
JL 01E9h
INSW
...

```

Registers:

	H	L
AX	00	01
BX	05	00
CX	00	00
DX	00	00
CS	0700	
IP	011E	
SS	0700	
SP	FFFE	
BP	0000	
SI	011E	
DI	0128	
DS	0700	
ES	0700	

Buttons: screen, source, reset, aux, vars, debug, stack, flags

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

```

emulator: p3.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 1
registers H L
AX 00 00
BX 05 00
CX 00 00
DX 00 00
CS 0700
IP 011E
SS 0700
SP FFFE
BP 0000
SI 011E
DI 0128
DS 0700
ES 0700
ds : 0500 0700:011E
07500: 00 000 NULL
07501: 11 017 ←
07502: 12 018 ↑
07503: 13 019 ::!
07504: 14 020 ¶
07505: 15 021 §
07506: 16 022 -
07507: 17 023 ±
07508: 18 024 ↑
07509: 19 025 ↓
0750A: 1A 026 →
0750B: 00 000 NULL
0750C: 00 000 NULL
0750D: 00 000 NULL
0750E: 00 000 NULL
0750F: 00 000 NULL
HLT
ADD [BP + SII], AX
ADD AX, [SII]
ADD AX, 00706h
OR [BX + DI1], CL
OR DL, [BX + SII]
ADC [BX + SII], DL
ADC [BX + SII], DL
...

```

Registers:

	H	L
AX	00	00
BX	05	00
CX	00	00
DX	00	00
CS	0700	
IP	011E	
SS	0700	
SP	FFFE	
BP	0000	
SI	011E	
DI	0128	
DS	0700	
ES	0700	

Buttons: screen, source, reset, aux, vars, debug, stack, flags

Example 3:

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

emulator: p3.com_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 1

registers		DS:500	0700:011E
AX	00 01	07500: 01 001 ⊗	MOU SI, 00128h
BX	05 00	07501: FF 255 RES	MOU DI, 00132h
CX	00 00	07502: FF 255 RES	MOU BX, 0050Ah
DX	00 00	07503: FF 255 RES	MOU CX, 0000Ah
CS	0700	07504: FF 255 RES	CLC
IP	011E	07505: FF 255 RES	MOU AL, [SI]
SS	0700	07506: FF 255 RES	ADC AL, [DI]
SP	FFFF	07507: FF 255 RES	MOU [BX], AL
BP	0000	07508: FF 255 RES	DEC SI
SI	011E	07509: FF 255 RES	DEC DI
DI	0128	0750A: FE 254 ■	DEC BX
DS	0700	0750B: 00 000 NULL	LOOP 010Dh
ES	0700	0750C: 00 000 NULL	MOU AL, 00h
		0750D: 00 000 NULL	ADC AL, 00h
		0750E: 00 000 NULL	MOU [BX], AL
		0750F: 00 000 NULL	HLT
		07510: 00 000 NULL	BIOS DI
		07511: 00 000 NULL	BIOS DI
		07512: 00 000 NULL	BIOS DI
		07513: 00 000 NULL	BIOS DI
		07514: 00 000 NULL	BIOS DI
		07515: 00 000 NULL	BIOS DI
		07516: 00 000 NULL	BIOS DI
		07517: 00 000 NULL	BIOS DI
			...

screen source reset aux vars debug stack flags

