

CSC 3210
Computer Organization and Programming
Lab 4
Answer Sheet

Student Name: Hakan Can Gunerli (could also be under John Gunerli) 002504797
Section: 014

Debug through each line of code.
Take screenshot that includes code and register window.
Record the register content.
and explain the register contents.

Line number: 10,11,12,13

Instruction:

```
mov al 0F5h,  
mov bl 029h,  
mov cl 00BH,  
mov dl 0D7h
```

Register values:

```
eax 0019F9F5  
ebx 00241029  
ecx 009F100B  
edx 009F10D7
```

Screenshot:

The screenshot shows a debugger window with the following content:

Registers window:

```
EAX = 0019F9F5 EBX = 00241029 ECX = 009F100B EDX = 009F10D7 ESI = 009F1005 EDI = 009F1005 EIP = 009F1018 ESP = 0019F8E8 EBP = 0019F8F4 EFL = 00000246  
OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
```

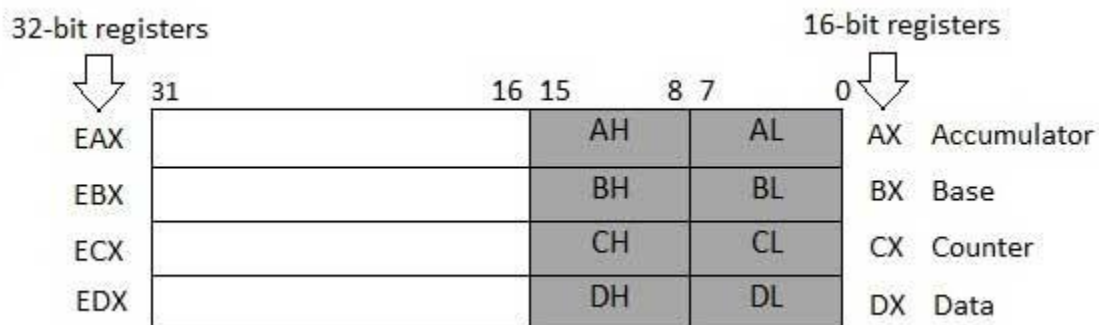
Disassembly window (main.asm):

```
1  .386  
2  .model flat,stdcall  
3  .stack 4096  
4  ExitProcess proto,dwExitCode:dword  
5  
6  
7  
8  .code  
9  main proc  
10 mov al, 0F5h  
11 mov bl, 029h  
12 mov cl, 00Bh  
13 mov dl, 0D7h  
14 sub al,dl ; 1ms elapsed  
15 sub cl,bl  
16 add al,cl  
17  
18 invoke ExitProcess,0  
19 main endp  
20 end main
```

Explanation:

These lines do the same action but for different registers, so I copied all into one explanation.

Each 32 bit register, `eax`, `ebx`, `ecx` and `edx` are consisted of four 16 bit data registers. These are called `AX` `BX` `CX` AND `DX`. These are also made of two 8 bit registers for each 16 bit. They are called `AH` `AL` for `AX`, `BH` `BL` for `BX`, `CH` `CL` for `CX` and finally `DH` `DL` for `DX`.



Although `ax` `bx` `cx` and `dx` also have their own reasons to be used, for this purpose I'll only be doing an arithmetic calculation.

In line 10 through 13 I'm moving the numbers I was instructed to do as hexadecimals.

`0F5` means 245

`029` means 41

`00B` means 11

And finally `0D7` means 241.

Line number: 14

Instruction: sub al,dl

Register values: eax:1E

Screenshot:



Explanation:

The al register's value which is F5 is subtracted from the dl's value which is 0D7, resulting in 1E which means 30.

Line number: 15

Instruction: sub cl,bl

Register values: ECX = 009F10E2

Screenshot:

The screenshot shows a debugger window with the following components:

- Registers:** EAX = 0019F91E, EBX = 00241029, ECX = 009F10E2, EDX = 009F10D7, ESI = 009F1005, EDI = 009F1005, EIP = 009F101C, ESP = 0019F8E8, EBP = 0019F8F4, EFL = 00000287. Below this, status flags are shown: OV = 0, UP = 0, EI = 1, PL = 1, ZR = 0, AC = 0, PE = 1, CY = 1.
- Disassembly:** The main.asm file is open, showing assembly instructions. Line 13, `mov d1, 007h`, is highlighted. The code includes a main procedure with several instructions: `mov al, 0F5h`, `mov bl, 029h`, `mov cl, 008h`, `mov d1, 007h`, `sub al, d1`, `sub cl, bl`, `add al, cl`, and `invoke ExitProcess, 0`.
- Diagnostic Tools:** A sidebar on the right contains sections for Events, Process Memory, CPU usage, and Memory Usage.

Explanation:

We are subtracting two values from each other. In this case, we have one signed value. (-41 bl), So the math we will do will represent the SIGNED value.

11-41 = -30 which is represented as E2 as a 8 bit binary converted to hexadecimal.

00001011+ 11010111= 11100010 which is a negative number. (-30)

Line number: 15

Instruction: sub al,cl

Register values:

Screenshot:

The screenshot shows a debugger window with two panes. The top pane, titled 'Registers', displays the current state of the CPU registers. The bottom pane, titled 'Disassembly', shows the assembly code being executed, with a red dot indicating the current instruction pointer.

Registers:

```
EAX = 0019F900 EBX = 00241029 ECX = 009F10E2 EDX = 009F10D7 ESI = 009F1005 EDI = 009F1005 EIP = 009F101E ESP = 0019F8E8 EBP = 0019F8F4 EFL = 00000257
```

OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 1 PE = 1 CY = 1

Disassembly:

```
1 .386
2 .model flat,stdcall
3 .stack 4096
4 ExitProcess proto,dwExitCode:dword
5
6
7
8 .code
9 main proc
10 mov al, 0F5h
11 mov bl, 029h
12 mov cl, 008h
13 mov dl, 007h
14 sub al,dl
15 sub cl,bl
16 add al,cl
17
18 invoke ExitProcess,0
19 main endp
20 end main
```

Explanation:

All the values in which we got from line 15 were stored in CL, and the all the values we got from al,dl subtraction were stored in line 14. Thus, we essentially add those two values together. E2 is signed so we need to make sure that we are doing our calculation with signed hexadecimal conventions, not unsigned. If you do it unsigned, you'll get 1C3 which will be over the value that AL can store.

The answer is 00h which is true if we do the math. $(245-215) + (11-41) = 0$.