CSC 3210

Computer Organization and Programming

Lab 3 (b)

Answer Sheet

Student Name: Aparna Mandapaka

Section:

Debug through each line of code and explain the register content.
(We already answered line 10 to 13 for your reference. Start writing your answer from Line 14)

Line: 10
Instruction: mov eax, 12345678h
Register value: EAX = 12345678
Explanation: 12345678 is a hexadecimal value which is 32-bit in binary. EAX register is also 32-bit.

Line 11:
Instruction: mov ax, 1122h
Register value: EAX = 12341122h
Explanation: 1122 is hexadecimal and it is 16-bit in binary. this mov instruction only updates AX (16 bit) register, a part of EAX register. That's why you can see that the upper portion of EAX register is NOT updated.

Line 12:
Instruction: mov bl, al
Register value: EBX = **00D6F0**22
Explanation: AL register is 8-bit long. When you mov the content of al register (22) to BL register, it only updates the first 8-bit of the EBX register. The rest contains the garbage value.

Line 13:
Instruction: mov bl, ah
Register value: EBX = 00D6F011
Explanation: Ah register is 8-bit long. When you mov the content of AH register (11) to BL register, it only updates the first 8-bit of the EBX register. The rest contains the garbage value.

Line 14:
Instruction: mov al, 89h
Register value of EAX register, after executing line 14: 1234**1189**
Explain the content of the EAX register. The value 1189 is a hexadecimal and it is a 8 bit value in binary. The mov instruction on the visual studio updates the AX register, since the last number is 8 bit, so it only updated the last 8 bit values and the rest of the number would be the overflow values. Here we are updating the al register. Since "al" is an 8-bit register, when we

use the ‘mov 89h’ operation to the ‘al’ register, it updates the least significant byte of the EAX register with ‘89h’ at al position

Line 15:
Instruction: add al, 10h
Register value of EAX, after executing line 15: **12341199**
Explanation: **here we can say that the hexadecimal ‘1199’ is a 8-bit register, so when we add 10h to the al register, it updates the least significant value or byte of the EAX register with the sum of 89h + 10h, which gives us 99h at the al spot.**
Show the step of the hexadecimal addition.

```
  8  9  h
+ 1  0  h
---------
  9   9 h
```

So therefore, we can see that when we perform the addition between the al, which from pervious step is 89h and then add 10h to it, it gives us 99h, which is the last bit value that we get when we execute the add al, 10h operation.

Line 16:
Instruction: sub al, al
What Register value of EAX, after executing line 15? 1234**1100**
Explanation: **so the ‘al’ is an 8 – bit register, so when we subtract the previous value 99h from 99h, we end up with 00 or 00h. it only updates the last two digits because it targets the least significant byte of the EAX.**
Show the step of the hexadecimal subtraction.

```
 9    9   h
-9    9   h
-----------
0     0   h
```

So we can see that when we add the value 89h + 10 h, we got the value of 99h and we stored that value in al, now in line 16, we are subtracting the al from al, which would be 99h-99h, which gives us 00h, which is the last two EAX value that we got.

Line 17, 18:
Instruction:

    mov al, 98h
    add al, 89h

Register value of EAX, after executing line 17 and 18:

Register value of EAX, after executing line 17:123411**98**
Register value of EAX, after executing line 18: 123411**21**
Show the step of the hexadecimal addition.

```
 9  8  h
+ 8 9 h
----------
 2  1 h
```

21h = 21h , it results extra carry 1 CF =1