Pre-Lecture 4

Due Aug 28 at 9am **Points** 14 **Questions** 8 **Available** until Aug 28 at 9am
Time Limit None **Allowed Attempts** 2

Instructions

Take this quiz ***after you have watched the required videos and/or read the associated sections of the textbook***. See Lecture 4: Representing control flow.

You may attempt this quiz twice. Incorrect responses are marked after each attempt. Correct answers are revealed at the start of class for this lecture.

Carefully note the deadline for responses. Submissions are not accepted after the deadline, and there is no grace period.

This quiz was locked Aug 28 at 9am.

Attempt History

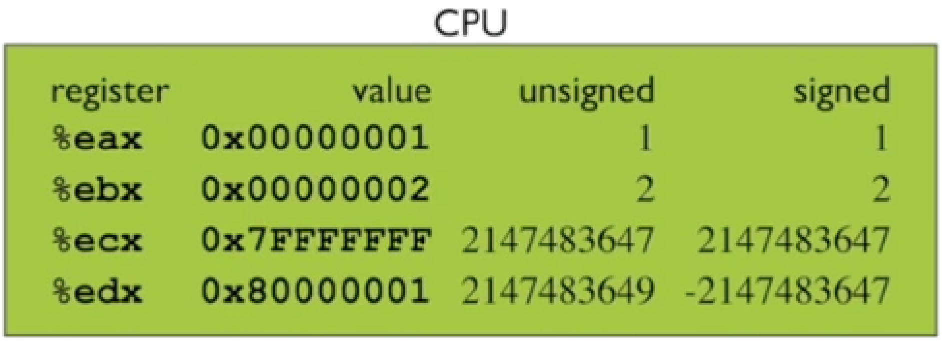
	Attempt	Time	Score
LATEST	Attempt 1	9,120 minutes	14 out of 14

Score for this attempt: **14** out of 14
Submitted Aug 27 at 9:29pm
This attempt took 9,120 minutes.

Question 1

1 / 1 pts

Suppose the following contents of registers %eax, %ebx, %ecx, and %edx:



CPU

register	value	unsigned	signed
%eax	0x00000001	1	1
%ebx	0x00000002	2	2
%ecx	0x7FFFFFFF	2147483647	2147483647
%edx	0x80000001	2147483649	-2147483647

Which of the condition codes are set by the instruction below? (Select all that apply.)

```
addl %eax, %ecx
```

- ☐ CF
- ☐ ZF
- Correct! ✓ SF
- Correct! ✓ OF

Question 2

1 / 1 pts

Consider again Question 1. Suppose the instruction there has been executed, and next to be executed is the following instruction:

```
setg %al
```

Give the the exact contents of register %al after execution, in hexadecimal. Pay close attention to the number of digits used.

%al: 0x 01

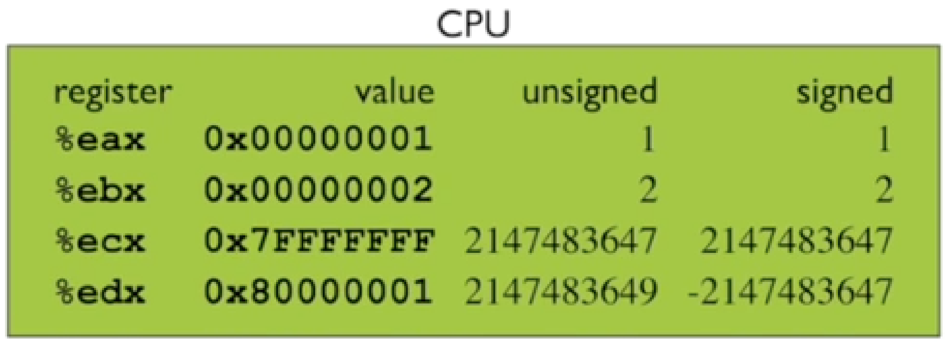
Answer 1:

Correct! 01

Question 3

1 / 1 pts

Suppose the following contents of registers %eax, %ebx, %ecx, and %edx:



CPU

register	value	unsigned	signed
%eax	0x00000001	1	1
%ebx	0x00000002	2	2
%ecx	0x7FFFFFFF	2147483647	2147483647
%edx	0x80000001	2147483649	-2147483647

Which of the condition codes are set by the instruction below? (Select all that apply.)

```
subl %ebx, %edx
```

- CF
- ZF
- SF
- [✓ OF]

Correct!

Question 4

3 / 3 pts

Consider the following partial function definition in C:

```
char ctest(int a, int b) {
 char t1 = a __ b;
 char t2 = (__) b __ (__) a;
 return t1 + t2;
}
```

The complete C code has been translated to this fragment of x86 assembly code (with comments):

```
# %edi contains a, %esi contains b
cmpl %esi, %edi          # Compare a-b
setge %al                # t1
cmpl %edi, %esi          # Compare b-a
setbe %cl                # t2
addb %al, %cl            # t2 += t1
movsbl %cl, %eax         # Convert type
```

Fill in the blanks below with ***exact*** C code (with no extra spaces) to complete the function definition above so that it matches the x86 code. Note that the blanks are ordered in the C code from top to bottom, left to right.

Blank 1: >=

Blank 2: unsigned

Blank 3: <=

Blank 4: unsigned

Answer 1:

Correct! >=

Answer 2:

Correct! unsigned

orrect Answer unsigned int

Answer 3:

Correct! <=

Answer 4:

Correct! unsigned

orrect Answer unsigned int

Question 5

1 / 1 pts

Consider the following partial function definition in C:

```
int ifstmt(int a, int b) {
 int x = 0;
 if(a __ b) {
   x = 2 * a;
```

```
      x += b;
    }
    else {
      x = b - 12;
      x -= a;
    }
    return x;
  }
```

The complete C code has been translated to this fragment of x86 assembly code (with comments):

```
    # %edi contains a, %esi contains b
    cmpl %esi, %edi              # Compare a-b
    jge .L2
    leal (%esi,%edi,2), %eax     # x
    ret
  .L2:
    leal -12(%esi), %eax         # x
    subl %edi, %eax              # x
    ret
```

Fill in the blank below with ***exact*** C code (with no extra spaces) to complete the function definition above so that it matches the x86 code.

Blank: <

Answer 1:

Correct!

<

Question 6

3 / 3 pts

Consider the following function definition in C:

```
  int loop(int start, int limit) {
    int x = 0;
    int y = 0;
    int i;

    for(i = start; i < limit; i += 4) {
      x += i;
      y++;
    }
    return x + y;
  }
```

The C code has been translated to this fragment of x86 assembly code:

```
        movl    $0, %eax
        movl    $0, %edx
        jmp     .L2
.L3:
        addl    %edi, %edx
        addl    $1, %eax
        addl    $4, %edi
.L2:
        cmpl    %esi, %edi
        jl      .L3
        addl    %edx, %eax
        ret
```

Match each register with the variable whose value it contains.

Correct!	**%eax**	y
Correct!	**%edx**	x
Correct!	**%edi**	i
Correct!	**%esi**	limit

Question 7

1 / 1 pts

For which of the following switch statements is a jump table more likely to be used?

```
switch(x) {
case 1:
  result = 39;
  break;
case 23:
  result = -2;
  break;
case 200:
  result = 600;
  break;
case 457:
  result = -90;
  break;
case 3011:
  result = 16;
  break;
default:
  result = 1;
}
```

Correct!

[✓]

```
switch(x) {
case 1:
  result = 39;
  break;
case 2:
  result = -2;
  break;
case 3:
  result = 600;
  break;
case 4:
  result = -90;
  break;
case 8:
  result = 16;
  break;
default:
  result = 1;
}
```

```
switch(x) {
case 0:
  result = 39;
  break;
case 1:
  result = -2;
  break;
default:
  result = 1;
}
```

Question 8

3 / 3 pts

Consider the following switch statement:

```
switch(x) {
 case 5:
 case 7:
   result = 2 * y;
   break;
 case 8:
   result += y - 4;
   break;
 case 9:
 case 10:
   result += 8;
   break;
 default:
   result = y;
 }
```

Suppose that the compiler generates this jump table:

```
.L4:
        .quad   .L3
        .quad   .L2
        .quad   .L3
        .quad   .L5
        .quad   .L6
        .quad   .L6
```

Match the each case of the switch to the correct label from the jump table.

Correct!

case 5

.L3

Correct!	**case 7**	.L3
Correct!	**case 8**	.L5
Correct!	**case 9**	.L6
Correct!	**case 10**	.L6
Correct!	**default**	.L2

Other Incorrect Match Options:

- .L4

Quiz Score: **14** out of 14