

Exam 1

Prof. Tak Auyeung

October 5, 2004

Instructions: You may bring any material that is hand-written or printed *prior* to the examination to help you. You can also bring a calculator if you think it may help you. However, you can only use the calculator for numerical computations only.

You, as an individual, are expected to do your own work. This means you cannot seek, receive or otherwise acquire any assistance except clarifications from the professor during an examination. Any communication involving the contents of the subject matter or the examination is considered cheating. Do not initiate or accept such communication, or the result of your examination is automatically voided.

Each correct answer is worth one point, each wrong answer is worth zero point, and each unanswered question is also worth zero point. This means you *should* guess and leave no question unanswered.

Make sure you write down your name on the upper right corner *first*, otherwise I cannot give points to anonymous students!

- 1 Which of the following instructions potentially changes the value of a general purpose register? Assume labels are defined so that all lines assemble.

- A `movl $0xff, (%eax)`
- B `movl $0xff, eax`
- C `movl $0xff, 6324523`
- D `movb $0xff, (,%eax)`
- E None of the other choices modify a general purpose register

- 2 Observe the following code:

```
movb $0x80,%al
subb $____,%al
```

Select a value for `____` such that the carry (C) flag is set after the `subb` instruction executes.

- A 0x00
- B 0x10
- C 0x7f
- D 0x80
- E 0xff

- 3 The following are byte values at location `somevar`:

0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

Assume that `%eax` has the address of `somevar`. What is the content of `somevar` after the following code executes?

```
movl (%eax),%ebx
addl $1,%eax
movl %ebx,(%eax)
```

- A 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08
- B 0x01 0x01 0x03 0x04 0x05 0x06 0x07 0x08
- C 0x01 0x01 0x02 0x03 0x04 0x05 0x06 0x07
- D 0x01 0x01 0x02 0x03 0x04 0x06 0x07 0x08
- E 0x01 0x02 0x03 0x04 0x01 0x02 0x03 0x04

- 4 Which of the following code compares two 64-bit number, and branches to `label1` if and only if `val1` is greater than `val2`? Assume `val1` and `val2` are signed, and they are properly allocated space. Select an answer that works in general.

- A

```
movl val1,%eax
cmpl val2,%eax
jg label1
```
- B

```
movl val1+4,%eax
cmpl val2+4,%eax
jg label1
movl val1,%eax
cmpl val2,%eax
jg label1
```
- C

```
movl val1,%eax
cmpl val2,%eax
jg label1
movl val1+4,%eax
cmpl val2+4,%eax
jg label1
```
- D

```
movl val1,%eax
cmpl val2+4,%eax
jg label1
movl val1+4,%eax
cmpl val2,%eax
jg label1
```
- E

```
movl val1,%eax
cmpl val2+4,%eax
```

```

jg  label1
movl val1+4,%eax
cmpl val2,%eax
jg  label1

```

- B 7
- C 6
- D 4
- E 0

5 Assume that `var1` and `var1_end` are defined as follows:

```

.data
var1: .org +20
      var1_end = .

```

The direction `.org +20` means increase the memory location counter by 20 (bytes). `var1_end` is effectively defined to be the address of the byte immediately following the 20 bytes allocated by the first line.

What does the following code do to the allocated memory locations?

```

      movl $var1,%eax
L1:   cmpl $var1_end,%eax
      jb  L2
      movb $0,(%eax)
      addl #1,%eax
      jmp L1

```

L2:

- A nothing
- B the first 19 bytes are initialized to 0
- C all 20 allocated bytes are initialized to 0
- D this is an infinite loop, `%eax` always alternates between 0 and 1
- E all 20 allocated bytes are initialized to 0, but one additional byte is also written to 0.

6 Assume the bytes starting at location `var1` are as follows:

0x01 0x02 0x03 0x04 0x00 0x05 0x06 0x07 0x08 0x09'

What is the value of `%ecx` when the following code completes?

```

      movl $10,%ecx
      movl $var1,%eax
L1:   cmpb $0,(%eax)
      jz  L2
      cmpl $0,%ecx
      jz  L2
      addl $1,%eax
      subl $1,%ecx
      jmp L1

```

L2:

- A 10

7 Read the assembly code in the previous question. Select a matching pseudocode representation. Recall that `*eax` means the location pointed to by `eax`. `&var1` means the address of `var1`.

- A `eax ← &var1`
`ecx ← 10`
while $((ecx) = 0) \wedge (*(eax) = 0)$ **do**
 `eax ← eax + 1`
 `ecx ← ecx - 1`
end while
- B `eax ← &var1`
`ecx ← 10`
while $((ecx) \neq 0) \vee (*(eax) \neq 0)$ **do**
 `eax ← eax + 1`
 `ecx ← ecx - 1`
end while
- C `eax ← &var1`
`ecx ← 10`
while $((ecx) = 0) \wedge (*(eax) \neq 0)$ **do**
 `eax ← eax + 1`
 `ecx ← ecx - 1`
end while
- D `eax ← &var1`
`ecx ← 10`
while $((ecx) \neq 0) \wedge (*(eax) \neq 0)$ **do**
 `eax ← eax + 1`
 `ecx ← ecx - 1`
end while
- E `eax ← &var1`
`ecx ← 10`
while $((ecx) \neq 0) \vee (*(eax) = 0)$ **do**
 `eax ← eax + 1`
 `ecx ← ecx - 1`
end while

8 What is the value of register `eax` after the following code completes?

```

movl $1, %eax
addl %eax, %eax
addl %eax, %eax

```

- A 1
- B 2
- C 3
- D 4
- E 8

9 How do we interpret the following assembly instructions?

```
cmpl %eax, %ebx
jna label1
```

- A jump to label1 iff register eax is not greater than ebx, signed interpretation
- B jump to label1 iff register ebx is not greater than eax, unsigned interpretation
- C jump to label1 iff register ebx is not greater than eax, signed interpretation
- D jump to label1 iff register eax is not greater than ebx, unsigned interpretation
- E jump to label1 iff register eax is greater than ebx, unsigned interpretation

10 Which flag(s) (Carry, Sign, Overflow and Zero) is/are set after the following code?

```
movl $-1, %eax
movl $-1,%ebx
addl %eax,%ebx
```

- A C, S
- B C, S, O
- C S, O
- D C, O
- E S

11 A
B
C
D
E

12 A
B
C
D
E