

# Exam

Prof. Tak Auyeung

March 11, 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 *cannot* let your calculator compile a program, or to communicate with others.

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.

*New rules, read this!* As of 2003.09.22, I no longer deduct points for wrong answers. 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.

As a result, I also need to adjust the letter grade assignment break points. For your individual examination, “A” means at least 95%, “B” means at least 75%, “C” means at least 55%, “D” means at least 35% and “F” means below 35%. This exam. makes up 20% of your final grade.

Please note that this change does not affect your letter grade at all, it is just a number game to make some people feel better about guessing.

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

The baseline is 12, there are 15 questions.

1 Which of the following `movw` has a problem when it is assembled?

- A `movw $1, 1`
- B `movw %bx, %ax`
- C `movw %ax, $1`
- D `movw -1(%eax),%bx`
- E none of the above has any problem

2 What is the an alias of `vash+2` given the symbols are defined as follows?

```
.lcomm vash 4
.lcomm knives 2
.lcomm seeds 10
```

- A `seeds-14`
- B `seeds-10`
- C `knives-2`
- D `knives+4`
- E `seeds-8`

3 What are the byte values at `outthere` when the following code is about to execute the `nop` instruction? The byte values are expressed in hexadecimal representation, from low address (left) to high address (right).

```
.lcomm outthere 4
.text
.global _start
_start:
    movl $outthere, %eax
    movb $0x04, %c1

L1:
    addl $1, %eax
    subb $1, %c1
    movb %c1, (%eax)
    jnz L1
```

```
    nop
# the usual exit code
```

- A this is an infinite loop, we'll never execute `nop`
- B the byte values cannot be determined
- C 03 02 01 00
- D 10 10 10 10
- E 10 11 12 13

4 Select a choice to substitute \_\_\_ so that the S flag is set immediately after the `addb` instruction in the following program:

```
movb $-120, %al
addb ___, %al
```

- A \$-120
- B \$-9
- C \$0
- D \$120
- E \$127

5 Unlike `add` and `sub`, `cmp` does not have a special version to utilize the carry flag. Given the following definitions, how can we compare the two 8-byte integers? Let's say we want to compare `var1` to `var2`, and `%eax` is at our disposal. Select the code fragment that changes the flags as if the compare is done by a single compare instruction. The choice must work for all cases except for one.

```
.lcomm var1 8
.lcomm var2 8
```

- A `movl var2,%eax`  
`cmpl %eax,var1`  
`movl var2+4,%eax`  
`jnc L1`  
`subl $1,%eax`  
L1:  
`cmpl %eax,var1+4`
- B `movl var2,%eax`  
`cmpl %eax,var1`  
`movl var2+4,%eax`  
`jc L1`  
`addl $1,%eax`  
L1:  
`cmpl %eax,var1+4`
- C `movl var2,%eax`  
`cmpl %eax,var1`  
`movl var2+4,%eax`  
`jc L1`  
`subl $1,%eax`  
L1:  
`cmpl %eax,var1+4`
- D `movl var2,%eax`  
`cmpl %eax,var1`  
`movl var2+4,%eax`  
`jnc L1`  
`addl $1,%eax`  
L1:  
`cmpl %eax,var1+4`

E we cannot determine the code because we don't know if the numbers are signed or not.

6 If `%ax` is less than `%bx` when interpreted unsigned, which of the following combinations of status flags is not possible? If a flag is listed, it has a value of 1, if it is not listed, it has a value of 0. Assume the following instruction is used to compare them:

```
cmpw %bx, %ax
```

- A CZ
- B Z
- C O
- D SO
- E all of the above are impossible

7 With the x86 architecture, which instruction can replace the block of code enclosed by `# begin block1` and `# end block1`? Assume L1 and L2 are only visible within block1.

```
cmpw %bx, %ax
# begin block1
jz L1
js L2
jno label1
L2:
jo label1
L1:
# end block1
```

- A `jnc label1`
- B `jb label1`
- C `ja label1`
- D `jna label1`
- E `jng label1`

8 Which assembly code fragment implements the following pseudocode? Assume  $x$  is already loaded into register  $\%ax$ . Also, assume the number is *unsigned*.

```

if  $x > 0$  then
    goto label1
end if

```

A    `cmpw $0, %ax`  
       `jnz label1`

B    `cmpw $0, %ax`  
       `jnl label1`

C    `cmpw $0, %ax`  
       `jna label1`

D    `cmpw $0, %ax`  
       `jnc label1`

E    `cmpw $0, %ax`  
       `jc label1`

9 In the following pseudocode, if we execute statement “do something useful”, which expression *must* have been evaluated? Assume we use short-circuited evaluation from left to right.

```

if  $(C \wedge D) \vee ((\neg C) \wedge (\neg D))$  then
    do something useful
end if

```

A only  $C$

B only  $D$

C both  $C$  and  $D$

D neither  $C$  nor  $D$

E none of the above is the answer

10 Which of the following assembly code fragments correspond(s) to this pseudocode? Assume  $x$  and  $y$  are loaded into registers  $\%ax$  and  $\%bx$ , respectively.

```

while  $x \neq y$  do
    block1
end while

```

A    `cmpw %bx, %ax`  
       `jz L1`  
       # code for block1  
       L1:

B L1:  
       `cmpw %ax, %bx`  
       `jnz L1`  
       # code for block1

C L1:  
       `cmpw %bx, %ax`  
       `jz L1`  
       `jmp L2`  
       # code for block1  
       L2:

D L1:  
       `cmpw %bx, %ax`  
       `jz L2`  
       # code for block1  
       `jmp L1`  
       L2:

E L1:  
       `cmpw %bx, %ax`  
       `jnz L1`  
       # code for block1  
       `jmp L2`  
       L2:

11 Assuming `%eax` can be changed, which of the following choices implement the same effects as the following instruction?

```
movw $0x1234, 200(, %eax, 4)
```

- A    `addl $200,%eax`  
       `addl $4,%eax`  
       `movw $0x1234,(%eax)`
- B    `addl $200,%eax`  
       `addl %eax,%eax`  
       `addl %eax,%eax`  
       `movw $0x1234,(%eax)`
- C    `addl $200,%eax`  
       `addl $200,%eax`  
       `addl $200,%eax`  
       `addl $200,%eax`  
       `movw $0x1234,(%eax)`
- D    `movl $200,%eax`  
       `addl %eax,%eax`  
       `addl %eax,%eax`  
       `addl %eax,%eax`  
       `addl %eax,%eax`  
       `movw $0x1234,(%eax)`
- E    `addl %eax,%eax`  
       `addl %eax,%eax`  
       `addl $200,%eax`  
       `movw $0x1234,(%eax)`

12 Not knowing the initial value of `%ax`, select a value to substitute `---` to guarantee the following code *not* to be an infinite loop.

```
L1:
  addw ____,%ax
  cmpw $0x8000,%ax
  jc L1
```

- A 1
- B 2
- C \$1
- D \$2
- E \$1 and \$2 both

13 Which of the following instructions causes an assembly time error?

- A `addb $0xff, %ah`
- B `addb (%eax),0xffff`
- C `addb 0xffff,%ah`
- D `addb $0xff,0xff`
- E `addb %eax, %bx`

14 Which flag is guaranteed to be set after the following instruction? We don't know anything about the initial value of `%ax`.

```
addw $65535,%ax
```

- A C
- B Z
- C S
- D O
- E one of the above must be set, but we don't know which one

15 Which combination(s) is(are) possible after the following instruction, regardless of the value of `%ax`? Select the most inclusive answer.

```
cmpw $0x0,%ax
```

- A C=1 and Z=0
- B C=0 and Z=1
- C C=1 and Z=1
- D C=0 and Z=0
- E combinations 15b and 15d are both possible