CSCI-UA.0201-003

Computer Systems Organization

Midterm Exam Fall 2016 (time: 60 minutes)

Last name: First name:

Notes:

- **If you perceive any ambiguity in any of the questions, state your assumptions clearly.**
- **Questions vary in difficulty; it is strongly recommended that you do not spend too much time on any one question.**

---

1. (5 points) Circle the correct answer among the choices given. If you circle more than one answer, you will lose the grade of the corresponding question.

(A) The compiler is:
1. machine dependent 2. language dependent [✓ 3. both] 4. none

(B) Assume we have the following statement in C: *A = B; . Also assume A is in ebx and B is in ecx. Which of the following is a correct x86 translation to the C statement?
1. movl %ebx, %ecx 2. movl %ecx, %ebx 3. movl %ebx,(%ecx)
4. movl (%ecx), %ebx 5. movl (%ebx), %ecx [✓ 6. movl %ecx, (%ebx)]

(C) The instruction *movq (%rax), %rbx* accesses the memory:
[✓ 1. once] 2.twice 3. 0 times 4. depends on whether 32-bit or 64-bit

(D) Which of the following data types is not affected by the byte ordering of the machine (i.e. big endian vs little endian)?
[✓ 1. char] 2. int 3. float 4.double

(E) Which of the following pointers has a larger size (in terms of bytes)?
1. pointer in a 32-bit machine [✓ 2. pointer in a 64-bit machine]
3. pointer in a 32-bit machine pointing to an array of 100 integers

2. [2 points] Suppose the variable a is an unsigned int and has a value of x. What will be the final value calculated (in terms of x) after the following expression? (Hint "~" is the bitwise not). Explain your thinking to get full credit.

1 + (a << 3) + ˜a

(a<<3) = 2^3*a = 8*a = 8x
~a+1 = 2's complement of a = -x
Final result = 8x – x = <u>7x</u>
The above solution assumes signed number, yet the problem states unsigned!
Let's see whether there is a difference. We have 7x + 1 + ~x = 6x + 0xFFFFFFFF + 1
(Did you realize that x + ~x in unsigned is 11111…1111 ? Try it!)

3. [2 points] In all 64-bit machines, a pointer is always 64 bits in length. Why do we need to specify the type of the variable the pointer is pointing to? That is, why don't we just declare x as a pointer instead of declaring it as pointer to int for example?

This is needed so that the compiler can generate the correct assembly code for pointer arithmetic. For example to interpret this *(a+2) the compiler needs to know whether a is pointing to a char for example, in which case it will be interpreted by incrementing a by 2, or whether a is int so it will get incremented by 8.

4. [2 points] Suppose we have the following decimal number: -15

a) Write that number in an 8-bit binary number. To get full credit, show all the steps.

First, let's get the binary of +15 → 0000 1111
Then we get the binary of -15 by getting the 2's complement of the above number → <u>1111 0001</u>

b) Translate the number you calculated in a) above to hexadecimal.

<u>0xF1</u>

5. [2 points] Suppose x is an integer. We want to test whether both the most significant and the least significant bits of x are 1 or not (i.e. the right most and left most bits), so we wrote the C expression:

if(….)
{ *tests successful and the two bits are 1* }
else
{ *at least one bit of the least significant or most significant is 1*}

What will you put between the parenthesizes in order to test that condition?

To get the correct condition you must ensure that:
(The most significant bit is 1) AND (The least significant bit is also 1)
So

```
sol 1:  if(  (x & 0x80000000)  &&  (x & 0x00000001) )

sol 2: if( (x < 0)  && ( x%2 != 0)  )
```

6. Suppose that we have the following number: 0x4C

a) [1 point] Write this number in binary:

0100 1100

b) [2 points] Suppose that this number is interpreted as unsigned number, what is the decimal equivalent (note: you don't have to write a final decimal number, you can leave it in the format of $2^x+2^y+ \ldots$). To get full score, show all the steps.

We have 8 bits. So each bit will be multiplied by 2^x where x depends on the position of the bit. We start form the far right (least significant bit) where x =0 till we reach the far left (most significant bit) where x = 7.
$\underline{\mathbf{2^6 + 2^3 + 2^2}}$

c) [2 points] Suppose that this number is interpreted as signed number, what is the decimal equivalent (note: you don't have to write a final decimal number, you can leave it in the format of $2^x+2^y+ \ldots$). To get full score, show all the steps.

The above number is positive so the result is like part b) above.

7. [2 points] Suppose “a” is a pointer to unsigned integer
(i.e. it was declared as *unsigned int * a;*) and points to the following array of unsigned integers: {3,2,2,1}.
How many times the body of the following loop will be executed? Justify

while((*a++) & 0x1) { .... loop body …. }

That loop checks whether the corresponding array element is odd. If it is odd, it will advance to the next element (a is incremented and, by pointer arithmetic, will move to the next element). Therefore, the loop body will be executed <u>once</u>.