CSCI-UA.0201-003

Computer Systems Organization

Exam 1 Fall 2019 (time: 60 minutes)

Last name: First name: NetID:

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.**
- **The exam consists of 5 pages, 5 questions, and a total of 50 points. Last paper is left intentionally blank, for you to use as draft if you want.**
- **You answer on this question sheet.**

---

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) N bits can present 2^N different numbers

1.For signed numbers 2. For unsigned numbers

3 For either signed and unsigned. 4. Depends on whether 32-bit or 64-bit machine

(B) Suppose we have an array **int a[10]** of 10 integers stored in a little-endian machine.

1. a[0] is stored in memory before a[1]
2. a[0] is stored in memory after a[1]
3. depends on the OS
4. depends on the programming language

(C) The smallest signed number that can be presented with five bits is:

1. 11111 2. 11110

3. 10000 4. 11000

(D) Suppose we have a 64-bit machine. The size of "**char ***" is:

1. 4 bytes 2. 8 bytes

3. 2 bytes 4. 1 byte

(E) If we see a file that can be opened by a text editor, then it has been generated by:

1. the loader 2. the assembler 3. the linker 4.the compiler

2. Suppose we have the following 6-bit binary number: **101010**

a) [2 points] Suppose this number is interpreted as signed-number, write it in hexadecimal

Each four binary digits form one hexadecimal. This number is 6 bits. So, it needs two hexadecimal digits. It is interpreted as signed, so we will do sign-extension.
101010 → 10 1010 → 1110 1010 → 0xEA

b) [2 points] Suppose this number is interpreted as unsigned-number, write it in hexadecimal

In this case it will be zero-extended.
101010 → 10 1010 → 0010 1010 → 0x2A

c) [2 points] Suppose this number is interpreted as signed-number, write it in decimal and show all steps.

101010 → MSB is 1 so negative number in two's complement form
-(two's complement of 101010) = -(010110) = - ($2^4 + 2^2 + 2^1$) = -22

d) [2 points] Suppose this number is interpreted as unsigned-number, write it in decimal and show all steps.

101010 → $2^5 + 2^3 + 2^1 = 32 + 8 + 2 = 42$

e) [1 point] The computer stores data in binary. Why do we need hexadecimals? Answer in one sentence.

To be able to present long binary strings in a concise and easy to read form.

f) [2 points] For a 5-bit signed number, what is the range of numbers that can be presented?

-2^{5-1} → $+2^{5-1}-1$ That is from -16 to +15

g) [2 points] For a 5-bit unsigned number, what is the range of numbers that can be presented?

0 → 2^5-1 That is from 0 to 31

3. Suppose we have the following piece of code and assume x is an unsigned char.

```
if ( x & mask)  {
   x is not divisible by four
}
else {
  x is divisible by four
}
```

a) [1 point] what is mask in binary?

mask must test that the two least significant bits are 0 for the number to be divisible by 4. And since x is char, which is 8 bits then mask must also be 8 bits.
Therefore mask = 00000011

b) [1 point] Which one do you think is faster: if(x%4 != 0) of the one stated in the problem above?

The one stated in the problem is much faster.

c) [2 points] Justify your answer in b.

The remainder involves division which is much slower than bitwise operations.

d) [1 point] Will your answer in a) be different if x was signed char instead of unsigned char?

No

e) [2 points] Justify your answer in d) above.

Because the two least significant bits won't be affected. Try -8 for example.
+8 = 01000 -8=11000

4. Suppose we have the following piece of C code:

```
char x = 5;
while(x)
{
   do something
  x >> = 1;
}
```

a) [4 points] How many iterations the while loop will be executed? Justify

Three iterations because it requires three shifting to the right to get rid of all the non-zero bits since x = 5 which is 00000101

b)[4 points] Suppose we had **x = -5** instead of the above. How many iterations the while-loop will be executed? Justify.

Will be an infinite loop.
Because each time a shift right is performed, a one, not zero, is inserted from the left. x is signed.

5. We studied pointers in class.
a) [2 points] Write a C statement to declare x as a pointer to double precision floating point.

double * x;

b) [2 points] Write one C statement to dynamically allocate 20 elements of a double floating point using the x that you declared in a) above.

x = (double *)malloc(20 * sizeof(double));

c) [1 point] A pointer always has the same size no matter what it is pointing to. Then, why do we need to declare the type of elements it will be pointing to?

To allow the compiler to make correct pointer arithmetic when arrays are accessed.

d) [1 point] If we have a 64-bit machine, how big, in terms of bytes, is x?

8 bytes

e) [1 point] Suppose y is another pointer to a double precision floating point. Write a statement to make y point to element number 5 of the array pointed to by x (i.e. the array you allocated in b) above).

y = &x[5];

6. Suppose we have the following C declaration:

```
struct node{
 int x;
 char y;
};
```

a) [1 point] How many bytes will the machine reserve after the above declaration?

Zero

b) [2 points] Justify your answer in a)

The code above defines a structure to the compiler but no variables are allocated.

c) [1 point] Write a statement to declare a variable temp of the above struct.

struct node temp;

d) [1 point] Write a statement to declare a pointer ptr to a struct of the type declared in the problem above.

struct node * ptr;

e) [1 point] Write a statement to make ptr point to temp.

ptr = &temp;

f) [4 points] State two reasons why we may want to use pointers in our programs

- **To pass data structures of any size as argument to a function.**
- **To dynamically allocate data structures.**
- **To allow a function to modify its arguments.**