ID________________________________ Name________________________________

Assume all operations are based on 64-bit instruction operated on the Intel CPU and represent a number in a hexadecimal form unless specified.

1. For each of the following sentences, you are to indicate whether the expression always yields "**true**". Otherwise, answer the question with a "**false**". If you write no answer you will get **-1** points. If you write the right answer, you will get **+2** points. You write the wrong answer, you will get **-2** points. Justify your answer if you choose false.

```
int t1 = random(); double d1 = (double)t1;
int t2 = random(); double d2 = (double)t2;
int t3 = random(); double d3 = (double)t3;
long t4 = random(); double d4 = (double)t4;
float f2 = (float)d2; float f3 = (float)d4;
double d5 = (d1 + d2) + d3;
double d6 = d1 + (d2 + d3);
double d7 = (long)t3;
```

Do not include special values of the floating point.

a) **t3** and **d7** are the same value.

b) **t4** and **d4** are the same value.

c) **f2** and **d2** are the same value.

d) **f3** and **d4** are the same value.

e) **d5** and **d6** are the same value.

f) **d4** has a multiplicative inverse.

g) **t4** has a multiplicative inverse.

h) **f2** has a multiplicative inverse.

i) The pipelining simultaneously dispatches multiple instructions to different execution units.

j) Each program running on 32-bit Intel CPU machine can address a maximum of 4 GB of memory.

k) The main memory is addressed using virtual addresses.

l) In the current implementation, the virtual address is represented by 64 bits. So, an address can specify a byte over a range of 2^{64}.

m) The size of instructions is variable and ranges in length one to 15 bytes.

n) An immediate in the regular **movq** instruction can only be a source operand.

```
9. int *xp; int x = *xp;
```

o) Dereferencing a pointer in C as shown above involves copying that pointer into a register and using this register to find a value in the memory. (line 9)

p) The C equivalent of **mov 0x10(%rax,%rcx,4),%rdx** is (line 10)

```
10. rdx = rax + rcx * 4 + 10;
```

2. Answer to the following question with a **short** answer. Show your work if applicable.

a) What are the four instruction cycle in the Von Neumann machine?

b) What is the primary reason we want to a cache near the CPU?

c) Why does the CPU have too many cache misses?

d) What does the program counter (PC) do and what is the name of the PC register?

e) If you have 2KB page size, N-bit virtual address space and M-bit physical address space what is the number of pages a program can have?

f) Round `11.111100` to nearest `1/8` using a default rounding mode.

```
11. int idivXOverTwoPowerY( int x, int y ) {
12.   return x/(2^y);
13. }
```

g) Convert above a division function, **idivXOverTwoPowerY**, to a shift function (≫, ≪). The two functions must produce the same result. (line 11 thru 13)

3. Consider an 8 bit floating-point representation based on the IEEE floating point format, with **one** sign bit, **four** exponent bits and **three** fraction bit. All answers should be in a decimal format.

a) What is the bias?

b) What is the largest value, other than infinity, that can be represented?

c) What is the smallest denormalized value that can be represented?

d) Fill in the following table. If a number is too big or too small to represent, use the representation of infinity.

Description	Binary	M	E	Value
Minus zero				-0.0
Negative Infinity				−∞
	1011 1010			
One				1.0
				-17/16
				6/512
				251.0

4. Consider following sequential instructions.

```
14. movabsq $0x11223344556677, %rax
```

a) What is the value of register `%rax`? (line 14)

```
15. movq $0x808A14E1, %rdx
```

b) What is the value of register `%rdx`? (line 15)

```
16. movl $0x808A14E1, %rdx
```

c) What is the value of register `%rdx`? (line 16)

```
17. movw $0x17EE, %dx
```

d) What is the value of register `%rdx`? (line 17)

```
18. movb $0x98, %dh
```

e) What is the value of register `%rdx`? (line 18)

```
19. movswl %dx, %eax
```

f) What is the value of register `%rax`? (line 19)

```
20. cltq
```

g) What is the value of register `%rax`? (line 20)

5. Each of the following instruction generates an error message. Explain what is wrong with each line.

```
21. movq $0x1873FAB77, %rax
22. movb $0x1FFF, (%eax)
23. movabsq $0x0011223344556677, (%rax)
24. movabsq $0x1F, %r8d
25. movabsq %rax, %rdx
26. movq %rax, $0011223344556677
27. movq (%rax), 8(%rsp)
28. cltq %eax
```

6. Bit manipulation. Function **`howManyBits`** should return the minimum number of bits required to represent **x** in two's complement.

Fill the following four blanks in the code below.
Legal operators are given as follows: **! ~ & ^ | + << >>**

```
int howManyBits( int x ) {
    int sign, pos, bias;
    sign = x >> (a)(___);

    x = (sign (b)(___)) | (~sign (c)(___));

    bias = !( x^0 );
    pos = ( !!( x >> 16 ) ) << 4;
    pos |= ( !!( x >> ( pos + 8 ) ) ) << 3;
    pos |= ( !!( x >> ( pos + 4 ) ) ) << 2;
    pos |= ( !!( x >> ( pos + 2 ) ) ) << 1;
    pos |= x >> ( pos + 1 );

    return ( pos + ( d )(___) + ( ~bias + 1 ) );
}
```