Name: ____________________________

Directions: **Work only on this sheet** (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing. In order to get full credit, **SHOW YOUR WORK**.

All code is assumed to be run on our Linux PCs in CSIF.

1. (15) Consider the instruction

```
add $12, (%ecx).
```

Fill in the blanks: The source operand and destination operands are expressed in _______________ and _______________ modes, respectively. Not counting instruction fetch, the number of accesses made to memory for the execution of the instruction is _______________.

2. (15) Suppose address size is 16 bits and block size is 64 bytes. Then the address 0x0312 would be in block number _______________.

3. (15) Suppose we wish to change the subroutine **findmin()** in Section 8 of our PLN unit on Linux assembly language as follows. It will now search for the *first* element "x[j], j = i+1, i+2, ..." which is smaller than "x[i]." If such an element is found, the subroutine will place in ESI and EDI the address and value of the element, respectively; otherwise the it will place -1 in ESI. Show what new code must replace that in lines 35 and 36.

4. (10) Fill in the blank with a single-word adjective from our course: Pentium CPUs have the ability to execute the instructions

```
movl $12, %eax
addl %ebx, %ecx
```

simultaneously because Pentiums (Pentia?) are _______________.

5. (15) The assembly code

```
movb 'g', %al
```

would likely (i) generate a run-time error, (ii) generate a link-time error, (iii) generate an assembly-time error, (iv) not generate error messages at any time.

6. (15) Consider the code

```
main()
{  int x[5],y[20];
   x[0] = 12;
   strncpy(&y[19],"abcde",5);
   printf("%d\n",x[0]);
   y[0] = 1684234849;  // 0x64636261
   y[1] = 6382179;  // 0x00616263
   printf("%d\n",strstr(y,"bc"));
}
```

The C string library function **strstr()** searches for the first instance of the second argument within the first. If the search is successful, the function will return the address of the first match, and otherwise it will return 0.

What will be printed out? Assume that the address of **y[0]** is 200.

7. (15) What will be the floating-point value in EAX after this code is executed?

```
.data
x:     .float 1.625
.text
.globl _start
_start:
   movl x,%eax
   andl $0xffbfffff,%eax
```

Solutions:

1. immediate, indirect, 2

2. 12

3.

```
cmpl $-1, %esi
jz nexti
```

4. superscalar

5. The programmer obviously had intended to write

```
movb $'g', %al
```

i.e. to copy the ASCII code for 'g', 0x67, into the AL register. But without the dollar sign, it says to copy the byte at memory address 0x67 to AL. That part of memory is likely to be off limits to the program, thus causing a seg fault at run time. So, the answer is (i).

6. The **strcpy()** call will be 'e', 0x65, in the low byte of **X[0]** overwriting its previous contents, 0x0c. So, the first call to **printf()** will print out the decimal version of 0x65, 101.

In the call to **strstr()**, the function will look for the first instance, if any, of "bc", i.e. 0x62 followed by 0x63. In checking this ourselves, we have to keep in mind that (a) this is a little-endian machine and (b) when we describe an integer in hex, we list its most significant (hex) digits first. That means that the lowest-address byte of **y[0]** will contain 0x61, and the highest-address byte in that word will contain 0x64. So, **strstr()** will find "bc" at the second and third bytes of that word, so the answer is 201.

7. Note that 1.625, the number in this problem, was used as an example in our PLN unit on information representation and storage (and again in our PLN supplement on floating-point instructions). So, we know right away what the Mantissa and Exponent field values are for 1.625. The AND operation then changes the Mantissa from 10100000000000000000000 to 00100000000000000000000, i.e. from $\frac{1}{2}+\frac{1}{8}$ to $\frac{1}{8}$. Thus the floating-point number will now be 1.125.