October 27, 2010 Name ______________________________

EECS 213
Introduction to Computer Systems
Midterm Exam

1. (16 pts total) Given the C code on the right:

```
int bitcnt(int n)
{
  unsigned m = 0;
  while ( n > 0 ) {
    m += n & 0x1;
  }
  return m;
}
```

a) (6 pts) **gcc –S** produces the assembly code below. Explain what each line does.

```
      pushl  %ebp              ______________________________

      movl   %esp, %ebp        ______________________________

      subl   $16, %esp         ______________________________

      movl   $0, -4(%ebp)      ______________________________

      jmp    L2                ______________________________

L3:

      movl   8(%ebp), %eax     ______________________________

      andl   $1, %eax          ______________________________

      addl   %eax, -4(%ebp)    ______________________________

L2:

      cmpl   $0, 8(%ebp)       ______________________________

      jg     L3                ______________________________

      movl   -4(%ebp), %eax    ______________________________

      leave                    ______________________________

      ret                      ______________________________
```

b) (6 pts) **gcc –S –O2** produces this assembly code. Explain what each line does.

```
      pushl %ebp               _____________________________
      movl  %esp, %ebp         _____________________________
      movl  8(%ebp), %eax      _____________________________
      testl %eax, %eax         _____________________________
      jg    L5                 _____________________________
      xorl  %eax, %eax         _____________________________
      popl  %ebp               _____________________________
      ret                      _____________________________
L5:
      jmp   L5                 _____________________________
```

c) (4 pts) Explain the optimizations made in version (b).

2. (6 pts) `strlen()` in C returns the length of a string. Its prototype is:

```
typedef unsigned int size_t;
size_t strlen(const char * s);
```

A student who didn't take EECS 213 wrote this code:

```
int is_longer_str(const char *s1, const char *s2)
{
  return strlen(s1) - strlen(s2) > 0;
}
```

Give an example where this will do the wrong thing, explain why, and give a simple fix. Be specific.

3. (13 pts) Fill in the following table for an IEEE floating point representation with 1 sign bit S, 3 exponent bits and 3 fraction bits,. M should be an integer or fraction, e.g., 0, 1, ¾. M, E and V should be base 10. V = $(-1)^S * M \cdot 2^E$

Binary	**M**	**E**	**V**
0 000 000			
1 110 110			
			1.75
0 000 011			
	–	–	∞

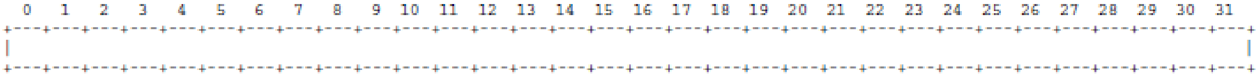
4. (19 pts) Fill in the table for a 5-bit two's complement integer representation.

Name	**Decimal**	**Binary**
–	14	
–	9	
–	-9	
–		0 1100
–		1 0100
TMax		
TMin		
Tmin + Tmax		
TMin + 1		
TMax + 1		
-TMax		
-TMin		

5. (15 pts) Given:

```
typedef struct {
  char c;
  double p;
  float d;
  short s;
  int *i;
} Struct1;
```

A. Use vertical lines and labels to indicate clearly how data would be allocated for each element of a structure of type `Struct1` on an IA32 (x86) machine using Linux alignment rules. Crosshatch areas that are allocated but not used.

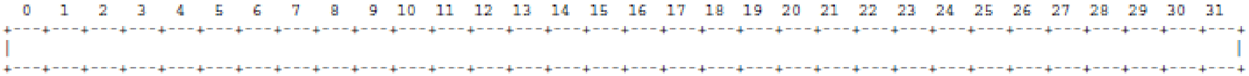


```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                                                                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

B. How many bytes are allocated for an object of type `Struct1`?

C. What alignment is required for an object of type `Struct1`? I.e., if an object must be aligned on an *x*-byte boundary, then say what *x* is.

D. Do (A) again, with the fields of `Struct1` re-ordered to use the least number of bytes. Crosshatch areas that are allocated but not used.



```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                                                                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

6. (14 pts) Assume the variables a and b are signed integers. Assume two's complement representation. Assume that MAX_INT is the maximum integer, MIN_INT is the minimum integer, and W is word length minus one, e.g., W = 31 for 32-bit integers. Next to each item on the left., write the letter of the code on the right that best matches it.

Description	Choice	Code
a		**a**. ˜(˜a | (b ˆ (MIN_INT + MAX_INT)))
a & b		**b.** ((a ˆ b) & ˜b) | (˜(a ˆ b) & b)
a * 7		**c.** a >> 3
a / 8		**d.** ˜((a >> W) << 1)
(a < 0) ? 1 : -1		**e.** ((a < 0) ? (a + 7) : a) >> 3
a * 14		**f.** ((~a & b) | a) & ((~a & b) | ~b)
a ^ b		**g.** ˜((a | (˜a + 1)) >> W) & 1
		h. (a << 3) + (a << 2) + (a << 1)
		i. 1 + (a << 3) + ˜a