NAME: ______________________________ UID: ______________

CS 4400: Computer Systems

Midterm Exam 1
Fall 2010

Please give your solutions in the space provided on the exam. If you choose to show your work on the exam, be sure to clearly indicate your final solution to each problem.

The exam is open-book, but closed-notes. *In addition, no laptops, calculators, cell phones, or other electronic devices are allowed.*

The point value of each question is clearly marked, so allocate your time wisely. The exam is worth a total of 75 points.

You must complete all work by 2:45pm, there are no exceptions.

Make sure that you have 10 numbered pages.

Problem 1	/ 10 points
Problem 2	/ 13 points
Problem 3	/ 12 points
Problem 4	/ 8 points
Problem 5	/ 8 points
Problem 6	/ 8 points
Problem 7	/ 16 points
Total	/ 75 points

/ 10 points

1. Consider a **7-bit** two's complement representation. Fill in the empty boxes in the following table. You need not fill in entries marked "—".

Expression	Decimal Representation	7-bit Binary Representation
—	51	
—	−12	
—		`010 1010`
—		`101 0101`
`-4 << 5`		
`38 >> 3`		
TMin		
TMax		
TMin +1		
−TMax		
TMax − TMin		
TMin − TMax		

/ 13 points

2. Consider the following 6-bit floating point representation based on the IEEE floating-point format.

   - The most significant bit indicates the sign.
   - The next two bits are the exponent.
   - The last three bits are the fraction.
   - The representation encodes numbers of the form: $V = (-1)^s \times M \times 2^E$, where M is the significand and E is the biased exponent.

   Fill in the table below. The following are the instructions for each field.

   - **Hex:** The 6-bit binary representation, given in 2-digit hexadecimal.
   - **M:** The value of the significand. This should be a number of the form x or $\frac{x}{y}$, where x is an integer and y is an integral power of 2. Examples include 0 and $\frac{3}{2}$.
   - **E:** The integer value of the exponent.
   - **Value:** The numeric value represented.

   Note: You need not fill in entries marked with "—".

Description	Hex	M	E	Value
Negative zero		—	—	—
Positive infinity		—	—	—
	`0x3E`	—	—	—
—	`0x15`			
—				-0.25
One				1.0
Smallest denormalized > 0				
Largest normalized > 0				

/ 12 points

3. Match each of the three C functions on the left with one of the IA32 assembly-code routines on the right.

```
int baz1(int a, int b) {
  if(b < a)
    return b;
  return a;
}


int baz2(int a, int b) {
  if(a < b)
    return a;
  return b;
}


int baz3(int a, int b) {
  unsigned ua = (unsigned) a;
  if (ua < b)
    return b;
  return ua;
}
```

```
asm1:
  pushl %ebp
  movl  %esp, %ebp
  movl  8(%ebp), %edx
  movl  12(%ebp), %eax
  cmpl  %edx, %eax
  jge   .L6
  movl  %edx, %eax
.L6:
  popl  %ebp
  ret

asm2:
  pushl %ebp
  movl  %esp, %ebp
  movl  8(%ebp), %edx
  movl  12(%ebp), %eax
  cmpl  %eax, %edx
  jb    .L9
  movl  %edx, %eax
.L9:
  popl  %ebp
  ret


asm3:
  pushl %ebp
  movl  %esp, %ebp
  movl  8(%ebp), %edx
  movl  12(%ebp), %eax
  cmpl  %edx, %eax
  jle   .L2
  movl  %edx, %eax
.L2:
  popl  %ebp
  ret
```

C function `baz1` corresponds to assembly-code routine ____________________.

C function `baz2` corresponds to assembly-code routine ____________________.

C function `baz3` corresponds to assembly-code routine ____________________.

/ 8 points

4. Consider the following IA32 assembly code and corresponding C function `bar` containing a for loop.

```
bar:
  pushl %ebp
  movl  $1, %eax
  movl  %esp, %ebp
  movl  12(%ebp), %ecx
  pushl %esi
  movl  8(%ebp), %esi
  testl %ecx, %ecx
  jle   .L4
  xorl  %edx, %edx
.L5:
  incl  %edx
  imull %esi, %eax
  cmpl  %edx, %ecx
  jne  .L5
.L4:
  popl  %esi
  leave
  ret
```

Fill in the blanks to provide the functionality of the loop.

```
int bar(int x, int y) {
  int i, result;

  for(i = ____________, result = ____________; ____________; i++)

     ______________________;

  return result;
}
```

/ 8 points

5. Consider the C code and its corresponding IA32 assembly code below. `M` and `N` are constants declared with `#define`.

```
int arr1[M][N];
int arr2[N][M];

void scale(int i, int j, int s) {
  arr1[i][j] *= s;
  arr2[j][i] *= s;
}
```

```
scale:
  pushl %ebp
  movl  %esp, %ebp
  subl  $8, %esp
  movl  8(%ebp), %ecx
  movl  %ebx, (%esp)
  movl  12(%ebp), %eax
  movl  16(%ebp), %edx
  movl  %esi, 4(%esp)
  leal  (%eax,%ecx,14), %ebx
  movl  arr1(,%ebx,4), %esi
  leal  (%eax,%eax,2), %eax
  leal  (%ecx,%eax,4), %eax
  imull %edx, %esi
  imull arr2(,%eax,4), %edx
  movl  %esi, arr1(,%ebx,4)
  movl  (%esp), %ebx
  movl  %edx, arr2(,%eax,4)
  movl  4(%esp), %esi
  movl  %ebp, %esp
  popl  %ebp
  ret
```

What are the values of `M` and `N`?

`M` =

`N` =

/ 8 points

6. Match each of the two IA32 assembly-code routines on the right with one of the five C functions on the left.

```
int foo1(int *ptr1, int *ptr2, int *ptr3) {
  int x = *ptr1;
  int y = *ptr2;
  int z = *ptr3;
  return x + y + z;
}

int foo2(int *ptr1, int *ptr2, int *ptr3) {
  int x = *ptr2;
  int y = *ptr3;
  int z = *ptr1;
  return x + y + z;
}

int foo3(int *ptr1, int *ptr2, int *ptr3) {
  int y = *ptr2;
  *ptr1 += *ptr3;
  return y;
}

int foo4(int *ptr1, int *ptr2, int *ptr3) {
  int x = *ptr1;
  *ptr3 += *ptr2;
  return x;
}

int foo5(int *ptr1, int *ptr2, int *ptr3) {
  int z = *ptr3;
  *ptr1 += *ptr2;
  return z;
}
```

```
asm1:
  pushl %ebp
  movl  %esp, %ebp
  movl  12(%ebp), %ecx
  movl  8(%ebp), %edx
  movl  16(%ebp), %eax
  movl  (%ecx), %ecx
  movl  (%eax), %eax
  addl  %ecx, (%edx)
  popl  %ebp
  ret
```

```
asm2:
  pushl %ebp
  movl  %esp, %ebp
  movl  12(%ebp), %eax
  movl  8(%ebp), %edx
  movl  (%eax), %eax
  addl  (%edx), %eax
  movl  16(%ebp), %edx
  popl  %ebp
  addl  (%edx), %eax
  ret
```

Assembly-code routine `asm1` corresponds to C function ____________________.

Assembly-code routine `asm2` corresponds to C function ____________________.

/ 16 points

7. Reconstruct C code based on declarations of C structures and unions and the corresponding IA32 assembly code.

```
struct s1 {
  char a[3];
  union u1 b;
  int c;
};


struct s2 {
  struct s1 *d;
  char e;
  int f[4];
  struct s2 *g;
};


union u1 {
  struct s1 *h;
  struct s2 *i;
  char j;
};
```

You may find it helpful to diagram these data structures in the space below.

For each IA32 assembly-code routine below on the left, fill in the missing portion of the corresponding C source code on the right.

(a)

```
proc1:
    pushl %ebp
    movl  %esp,%ebp
    movl  8(%ebp),%eax
    movl  8(%eax),%eax
    movl  %ebp,%esp
    popl  %ebp
    ret
```

```
int proc1(struct s1 *x) {

  return x->___________________ ;

}
```

(b)

```
proc2:
    pushl %ebp
    movl  %esp,%ebp
    movl  8(%ebp),%eax
    movl  12(%eax),%eax
    movl  %ebp,%esp
    popl  %ebp
    ret
```

```
int proc2(struct s2 *x) {

  return x->___________________ ;

}
```

(c)

```
proc3:
    pushl %ebp
    movl  %esp,%ebp
    movl  8(%ebp),%eax
    movl  4(%eax),%eax
    movl  20(%eax),%eax
    movl  %ebp,%esp
    popl  %ebp
    ret
```

```
int proc3(struct s1 *x) {

  return x->___________________ ;

}
```

(d)

```
proc4:
    pushl  %ebp
    movl   %esp,%ebp
    movl   8(%ebp),%eax
    movl   (%eax),%eax
    movl   24(%eax),%eax
    movl   (%eax),%eax
    movsbl 1(%eax),%eax
    movl   %ebp,%esp
    popl   %ebp
    ret
```

```
char proc4(union u1 *x) {

  return x->___________________ ;

}
```