CIS 2107
Computer Systems and Low-Level Programming
Spring 2011
Final Exam

May 10, 2011

Name: _____

Page	Points	Score
1	16	
2	9	
3	10	
4	13	
5	5	
6	14	
7	9	
8	6	
9	10	
10	8	
Total:	100	

Instructions
The exam is closed book, closed notes. You may *not* use a calculator, cell phone, etc.

For the short-answer questions, please use the amount of space given as a guide for how long your answer should be.

For each of the questions of this quiz, you can assume the following sizes for C data types:

type	**bytes**
char	1
short	2
int	4
long	8
float	4
double	8
void*	4

1. When I run a particular binary file through a disassembler, I see the following output:

```
000000d5 <is_empty>:
    d5:   55          push %ebp
    d6:   89 e5       mov %esp,%ebp
    d8:   8b 45 08    mov 0x8(%ebp),%eax
    db:   8b 00       mov (%eax),%eax
    dd:   85 c0       test %eax,%eax
    df:   0f 94 c0    sete %al
    e2:   0f b6 c0    movzbl %al,%eax
    e5:   5d          pop %ebp
    e6:   c3          ret
```

(2 points) (a) What, specifically, is represented in the first two columns? (Hint: columns 1 and 2 correspond to columns 1 and 2 of every SIM program that you've seen.)

For the next two questions, please do not write `gcc`. You're being asked about a specific component of gcc.

(2 points) (b) What type of program will translate C code into the sort of code you see in column 3?

(b) _____

(2 points) (c) What type of program translates the sort of code found in column 3 to the sort of code found in column 2?

(c) _____

(2 points) 2. What is done by the C preprocessor?

(2 points) 3. Other than the problem of differing libraries, explain why it is that I can't take a Windows executable, and run it on an Intel Mac or a machine running Linux on Intel. Please do not write, "because they're different operating systems". Be more specific about what the biggest problem is.

(2 points) 4. Where do we store an integer return value from a function in x86 assembly?

(2 points) 5. Memory allocated with `malloc( )` is stored in what memory segment?

(2 points) 6. When creating an executable file, what is the biggest difference between statically linking and dynamically linking?

(2 points) 7. In the steganography lab, describe how it is that there appeared to be no difference in the image after the secret message was inserted.

8. What are the key characteristics of each of the following memory allocation schemes:

(1 point) (a) Implicit free lists

(1 point) (b) Explicit free lists

(1 point) (c) Segregated storage

(2 points) 9. The first and last lines of a function in assembly are something like:

```
pushl %ebp
movl  %esp, %ebp

     /* body of the function here */

movl %ebp, %esp
popl %ebp
```

Very briefly describe what's going on. Be sure to mention what is stored in %ebp and %esp.

(2 points) 10. A program calling the following function will crash. What causes the crash? Please say more than, "You forgot the base case". Missing the base case makes the program incorrect. What makes it crash?

```
int bad_fact(int n) {
   return n*bad_fact(n-1);
}
```

11. **Some bit operations** If we have `char i = 0xD9, j = 0x8C;`, what is the result of the following operations? Your answer must be in the form of exactly two hex digits[1].

(1 point) (a) (!!i)^(j & 0xFF)

(a) _____

(1 point) (b) i|j

(b) _____

(1 point) (c) i>(j | 0xF0)

(c) _____

(1 point) (d) i<<2

(d) _____

(1 point) 12. In hex, what is -1 as an 8-bit two's complement int?

12. _____

(1 point) 13. In hex, what is the largest integer that can be represented by a 8-bit two's complement int?

13. _____

(1 point) 14. In hex, what is the largest integer that can be represented by a 8-bit unsigned int?

14. _____

(1 point) 15. In hex, what is the smallest integer that can be represented by a 8-bit two's complement int?

15. _____

(2 points) 16. What is $111011011_2 + 10011010_2$ in base 2?

$$\begin{array}{rcccccccc} & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1_2 \\ + & & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0_2 \\ \hline \end{array}$$

[1] Forget about the possibility of the values being promoted to 32-bits. Just behave as though we're living in the land of 8-bit arithmetic.

(2 points) 17. What is $A7935B_{16} + 5C8D3_{16}$ in base 16?

	A	7	9	3	5	B_{16}
+		5	C	8	D	3_{16}

(4 points) 18. How would the number 197.6875_{10} be stored in a 32-bit C float variable?

19. **Some tricky declarations**. Write a very brief description in English of what is declared. For example, if the question is `int func(int A[])`, you'd write, "func is a function which is passed an array of int and returns an int".

(1 point) (a) char (*p)[10];

(1 point) (b) int (*p[ ])( );

(1 point) (c) int (*p( ))[ ];

(4 points) 20. For each of the following, suppose that %eax contains the value x, %ecx contains y. What's stored in %edx after the each operation?

expression	result
leal 0xC(%eax), %edx	
leal (%eax,%ecx, 4), %edx	
leal 5(%eax,%eax,8), %edx	
leal 0xA(,%ecx,8), %edx	

21. What is printed?

In the following code, after the function f( ) is called, what is the value of each of the following?

```
1  #include <stdlib.h>
2
3  typedef struct {
4    int x;
5    int *p;
6    int A[100];
7  } Junk;
8
9  void f(Junk*, Junk, int[]);
10
11 int main(void)
12 {
13   int a = 1;
14   int b = 5;
15   Junk j1, j2;
16   int A[5]={4,0,0,0,0};
17
18   j1.x=2;
19   j1.p=&a;
20   j1.A[0]=3;
21
22   j2.x=6;
23   j2.p=&b;
24   j2.A[0]=7;
25
26   f(&j1, j2, A);
27
28   return 0;
29 }
30
31 void f(Junk *j1, Junk j2, int A[])
32 {
33   j1->x=222;
34   (j1->p)=(int*)malloc(sizeof(int));
35   *(j1->p)=111;
36   j1->A[0]=333;
37
38   j2.x=666;
39   *(j2.p)=555;
40   j2.A[0]=777;
41
42   A[0]=444;
43 }
```

(1 point) (a) j1.x

(a) _____

(1 point) (b) *(j1.p)

(b) _____

(1 point) (c) j1.A[0]

(c) _____

(1 point) (d) j2.x

(d) _____

(1 point) (e) *(j2.p)

(e) _____

points: _____
out of a possible 5

question 21 continues...

(1 point) (f) j2.A[0]

(f) _____

When we make the function call, how many bytes is passed to f( ) when we pass:

(1 point) (g) &j1

(g) _____

(1 point) (h) j2

(h) _____

(1 point) (i) A

(i) _____

(10 points) 22. Write a function which is passed an array of `int A[ ]` and its length. The function reverses the byte order of each of the elements of A. Do not use the `[ ]` operator. Do not assume that ints are 4 bytes. (Please be careful about what's being asked. You are not asked to reverse the array. You're asked to reverse the byte order of each element of A.)

(9 points) 23. Recall the Fibonacci sequence: (1,1,2,3,5,8,13,21, . . .) . Write a function f( ) which is passed an int n, and returns a pointer to an array of int containing the first n Fibonacci numbers. It is up to the caller to free the memory allocated by f( ).

(6 points) 24. On some big-endian system, dates are stored in a single 32-bit unsigned int.

year is stored in the low-order 12 bits

month is stored in the next 4 bits. (e.g., 0000 for January, 0001 for February, ..., 1011 for December.)

day is stored in the next 5 bits

Implement the following three functions used to read date information in this manner from an unsigned 32-bit int d:

```
/* extracts the day from d and returns
   it as an unsigned int */
unsigned int day(unsigned int d)
{









}
/* extracts the month from d and returns
   it as an unsigned int.  e.g., returns 0
   for January, 1 for February, ... ,
   11 for December */
unsigned int month(unsigned int d)
{









}
/* extracts the year from d and returns
   it as an unsigned int */
unsigned int year(unsigned int d)
{









}
```

(10 points) 25. Write a C function equivalent to the following assembly (no credit for an answer containing inline assembly).

```
1           .text
2   .globl mystery
3           .type    mystery, @function
4   mystery:
5           pushl    %ebp
6           movl     %esp, %ebp
7           movl     12(%ebp), %edx
8           movl     8(%ebp), %eax
9           subl     %edx, %eax
10          addl     16(%ebp), %eax
11          addl     $91, %eax
12          popl     %ebp
13          ret
```

(8 points) 26. The following are declarations that we've used in a linked implementation of a stack. Please write the `pop( )` function.

```
1  typedef struct stack_node {
2    void *data;
3    struct stack_node* next;
4  } stack_node;
5
6  typedef struct {
7    stack_node *top;
8  } Stack;
```

(extra space)