CS 3843 Spring 2011 Final Exam

Answer the following questions in the space provided. You may use the back of the test as a scratch area. Make sure to write your name on your test.

True or False
Write T for True or F for False in the spaces provided.

1. __________ The gets() function is insecure and should not be used in production code.
2. __________ Calling a function can prevent the compiler from performing certain optimizations.
3. __________ In x86 assembly language, the %eax register is used as the return value from functions.
4. __________ The andl instruction adds two quantities.
5. __________ xorl %eax,%eax sets %eax to 1 if %ebx is odd.
6. __________ A P-type transistor is good as passing ones, but bad at passing zeroes.
7. __________ Stack-allocated objects include arguments and some local variables.
8. __________ Arguments to compiled C functions are passed on the stack.
9. __________ The decimal value 11 in binary is 1011.
10. __________ A P-type transistor is switched on when a one is placed on its gate terminal.
11. __________ The %ebp register points to the current stack frame.
12. __________ Under x86 Linux, a struct with an array of char is aligned on a memory address divisible by 8.
13. __________ A union containing an array of 20 ints would be 80 bytes long.
14. __________ Heap-allocated objects are placed on a special part of the stack.
15. __________ “Arithmetic shift right” is the same as “logical shift right.”
16. __________ A ripple-carry adder circuit that adds two n-bit numbers takes time proportional to $O(n)$.
17. __________ xorl %eax,%eax is more efficient than movl $0,%eax.
18. __________ Under Microsoft Windows, a struct with a long double is aligned on an address divisible by 12.
19. __________ The x86 architecture is big-endian.
20. __________ If %eax starts as 1, then leal 1(%eax,%eax,8),%eax changes it to 10.
21. Consider the following circuit diagram.

Give a truth table for the Boolean function of $a$ and $b$ computed by this circuit.
22. Consider the following code fragment:

```c
union {
    unsigned char c[4];
    unsigned int i;
} x;

x.c[0] = 0x21;
x.c[1] = 0x43;
x.c[2] = 0x65;
x.c[3] = 0x87;
printf ("%x\n", x.i);

x.c[0] = x.c[0] | 0x12;
  x.c[1] = x.c[1] & 0x34;
  x.c[2] = x.c[2] | 0x56;
  x.c[3] = x.c[3] & 0x78;
printf ("%x\n", x.i);
```

Recall that `%x` means “print hexadecimal” for `printf`.

(a) What is the output of this code on a little-endian machine?

(b) What is the output of this code on a big-endian machine?

(c) If we replaced the `&` operator with `&&`, what would be the value of `x.c[3]` at the end of this code?
Reading Comprehension

23. Consider the following assembly language program:

```assembly
f:  pushl %ebp
    movl %esp, %ebp
    movl 8(%ebp), %eax
    popl %ebp
    leal (%eax,%eax,2), %eax
    ret
.globl main
main: pushl %ebp
    movl %esp, %ebp
    pushl %esi
    xorl %esi, %esi
    pushl %ebx
    xorl %ebx, %ebx
    subl $4, %esp
.L2: movl %ebx, (%esp)
    addl $1, %ebx
    call f
    addl %eax, %esi
    cmpl $101, %ebx
    jne .L2
    movl %esi, 4(%esp)
    movl $.LC0, (%esp)
    call printf
    addl $4, %esp
    xorl %eax, %eax
    popl %ebx
    popl %esi
    movl %ebp, %esp
    popl %ebp
    ret
.LC0: .string "%d\n"
```

1. Describe, in 25 words or fewer, what this program does.

2. Write a C program that could plausibly be compiled to result in this assembly language program.
Alignment
24. Consider the following definition of an array A in C:

```c
struct {
    unsigned int a;
    char b;
    int c[2];
    double d[2];
    char e;
} x;
```

1. What is the size of \( x \) in bytes on x86 Linux?

2. What is the size of \( x \) in bytes on Microsoft Windows?

3. Rewrite the definition of \( x \) so that it contains the same fields with the same types but takes as few bytes as possible under x86 Linux.

4. How many bytes does your new \( x \) consume under x86 Linux?