Exam 2 Review Problems

The exam is over the TOY machine language and the following sections from Chapter 3.

- Chapter 3, Sections 3.1, 3.2, 3.3, 3.4, 3.5.1–3.5.3, 3.6.1–3.6.5, 3.6.7, 3.7, and 3.8.1–3.8.2.

1. You should solve the following "Practice Problems" from the textbook (the solutions are on pages 325–340).

   Practice Problem 3.1 (page 182)
   Practice Problem 3.2 (page 185)
   Practice Problem 3.5 (page 189)
   Practice Problem 3.8 (page 194)
   Practice Problem 3.14 (page 205)
   Practice Problem 3.16B (page 212) (just Part B)
   Practice Problem 3.23 (page 222-223)
   Practice Problem 3.32 (page 244-245)
   Practice Problem 3.34 (page 252-253)

2. Assume the following values are stored at the memory addresses and registers indicated by the first two tables. Fill in the third table showing the values for the indicated operands.

Address	Value
0x100	0xFF
0x104	0xAB
0x108	0x13
0x10C	0x11

Register	Value
%eax	0x104
%ecx	0x101
%edx	0x3

Operand	Value
%eax	
0x10C	
$0x104	
(%eax)	
4(%eax)	
8(%ecx, %edx)	

3. Assume the following values are stored at the indicated memory addresses and registers.

Address	Value
0x100	0x7F
0x104	0xAB
0x108	0x13
0x10C	0x11

Register	Value
%eax	0x100
%ecx	0x1
%edx	0x3

Fill in the following table showing the effects of the following instructions, both in terms of the register or memory location that will be updated and the resulting value.

Instruction	Destination	Value
`addl %ecx, (%eax)`		
`orl %edx, 4(%eax)`		
`imull $8, 5(%eax, %edx)`		
`incl %eax`		

4. The assembler routine on the right was generated from the equivalent C function on the left. Explain carefully why there is a `subl` instruction in the assembly language listing. (Note: The variable `x` is stored in the location denoted by `8(%ebp)`.)

```
int fun1(int x)
{
    return 15 * x;
}
```

```
fun1:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     sall $4,%eax
     subl 8(%ebp),%eax
     movl %ebp,%esp
     popl %ebp
     ret
```

5. Consider the following C functions and assembly code:

```
int fun2(int x)
{
    return x * 30;
}

int fun3(int x)
{
    return x * 34;
}

int fun4(int x)
{
    return x * 18;
}
```

```
pushl %ebp
movl %esp,%ebp
movl 8(%ebp),%eax
sall $4,%eax
addl 8(%ebp),%eax
addl %eax,%eax
movl %ebp,%esp
popl %ebp
ret
```

Which of the functions compiled into the assembly code shown? Explain how you can tell.

6. (a) Explain the meaning of the terms "caller save registers" and "callee save registers." (You do not need to remember which registers are of which kind in an Intel CPU.)

   (b) Why do we have both kinds of registers? In particular, what would be the disadvantage if all registers were caller save? Is there any disadvantage to all the registers being callee save? (Be sure to answer all three questions and be sure to give these questions a bit of thought.)

7. Describe the steps involved in creating a stack frame. Be sure to specify what steps are done by the caller and what steps are done by the callee. Draw a rough sketch of what a stack frame looks like.

8. Consider the following C functions and assembly code. Which function compiled into the assembly code shown? Explain how you can tell. (Hint: Look at the C code first. How do these two functions differ? Use that bit of information when analyzing the assembly code. Also, C functions push their arguments onto the stack from right to left, i.e., the rightmost argument is pushed first.)

```
int fun1(int a, int b)
{
    if (a < b)
        return a;
    else
        return b;
}

int fun2(int a, int b)
{
    if (b < a)
        return b;
    else
        return a;
}
```

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

9. Consider the following assembly code for a C for-loop.

```
loop:
        pushl %ebp
        movl %esp,%ebp
        movl 8(%ebp),%ecx
        movl 12(%ebp),%edx
        xorl %eax,%eax
        cmpl %edx,%ecx
        jle .L4
.L6:
        decl %ecx
        incl %edx
        incl %eax
        cmpl %edx,%ecx
        jg .L6
.L4:
        incl %eax
        movl %ebp,%esp
        popl %ebp
        ret
```

Based on the assembly code above, fill in the blanks below in its corresponding C source code. (Note: you may only use the symbolic variables `x`, `y`, and `result` in your expressions below — *do not use register names.*)

```
int loop(int x, int y)
{
   int result;

   for (____________; __________; result++ ) {

      _________;

      _________;
   }

   _________;

   return result;
}
```