ECGR 4101/5101 - Assignment 2

1. A function fun_a, has the following overall structure:

```
int fun_a(unsigned x)
{
        int val = 0;
        while (_________) {
                _________;
        }
        return _________;
}
```

The GCC C compiler generates the following assembly code:

```
# x at %ebp+8

movl 8(%ebp), %edx
movl $0, %eax
testl %edx, %edx
je .L7
.L10:
xorl %edx, %eax
shrl %edx # shift right by 1
jne .L10
.L7:
andl $1, %eax
```

Reverse engineer the operation of this code and then do the following:
A. Use the assembly-code version to fill in the missing parts of the C code.
B. Describe in English what this function computes.

2. A function fun_b has the following overall structure:

```
int fun_b(unsigned x) {
        int val = 0;
        int i;
        for(_________, _________, _________;) {
                _________;

        }
        return val;
}
```

The GCC C compiler generates the following assembly code;

```
# x at %ebp + 8
movl 8(%ebp), %ebx
movl $0, %eax
movl $0, %ecx
.L13:
 leal (%eax, %eax), %edx
 movl %ebx, %eax
 andl $1, %eax
 orl %edx, %eax
 shrl %ebx
 addl $1, %ecx
 cmpl $32, %ecx
 jne .L13
```

Reverse engineer the operation of this code and then do the following:
A. Use the assembly code version to fill in the missing parts of the C code.
B. Describe in English what this function computes.

3. In the following C function, we have left the definition of OP incomplete:

```
#define OP _________ /*Unknown operator */

int arit(int x) {
        return x OP 4;
}
```

When compiled, GCC generates the following assembly code:

```
leal 3(%edx), %eax
testl %edx, %edx
cmovns %edx, %eax
sarl $2, %eax # Return value in %eax
```

A. What operation is OP?
B. Annotate the code to explain how it works?

4. Given the C function

```
int proc(void)
{
         int x,y;
         scanf(”%x %x”, &y, &x);
         return x-y;
}
```

```
proc:
pushl %ebp
movl %esp,%ebp   # line 1
subl $40, %esp   # line 2
leal -4(%ebp), %eax
movl %eax, 8(%esp)
leal -8(%ebp), %eax
movl %eax, 4(%esp)
movl $.LCO, (%esp) # Pointer to string ”%x %x”
call scanf
movl -4(%ebp), %eax
subl -8(%ebp), %eax
leave
ret
```

Assume that procedure proc starts executing with the following register values:

```
%esp 0x800040
%ebp 0x800060
```

Suppose proc calls scanf and that scanf reads values 0x46 and 0x53 from the standard input. Assume that the string ”%x %x” is stored at memory location 0x300070.

A. What value does %ebp get set to on line 1?
B. What value does %esp get set to on line 2?
C. At what addresses are local variables x and y stored?
D. Draw a diagram of the stack frame for proc right after scanf returns. Include as much information as you can about the addreses and the contents of the stack frame elements.
E. Indicate the regions of the stack frame that are not used by proc.

5. For a C function having the general structure

```
int rfun(unsigned x) {
         if (_______)
```

```
                return ______;
        unsigned nx = _________;
        int rv = rfun(nx);
        return ______;

}
```

GCC generates the following code (with setup and completion code omitted)

```
movl 8(%ebp), %ebx
movl $0, %eax
testl %ebx, %ebx
je .L3
movl %ebx, %eax
shrl %eax
movl %eax, (%esp)
call rfun
movl %ebx, %edx
andl $1, %edx
leal (%edx, %eax), %eax
.L3:
```

A. What value does rfun store in the callee-save register %ebx?

B. Fill in the missing expression in the C code show above.

C. Describe in English what function this code computes.

6. Consider the following source code, where M and N are constants with #define:

```
int mat1[M][N];
int mat2[N][M];

int sum_element(int i, int j) {
        return mat1[i][j] + mat2[j][i];
}
```

In compiling this program, GCC generates the following assembly code:

```
# i at %ebp+8, j at %ebp+12

movl 8(%ebp), %ecx
movl 12(%ebp), %edx
leal 0(,%ecx,8), %eax
subl %ecx, %eax
```

```
addl %edx, %eax
leal (%edx,%edx,4), %edx
addl %ecx, %edx
movl mat1(,%eax,4), %eax
addl mat2(,%edx,4), %eax
```

Use your reverse engineering skills to determine the values of M and N based on this assembly code.