

Manit Gandhi
6280004

Problem 3:

Consider the following C functions and assembly code:

```
int fun1(int i, int j)
{
    if(i+3 != j)
        return i+3;
    else
        return j*16;
}

int fun2(int i, int j)
{
    if(i+3 != (unsigned)j)
        return i;
    else
        return j*4;
}

int fun3(int i, int j)
{
    if(i+3 <= (unsigned)j)
        return i;
    else
        return j>>2;
}
```

```

pushl    %ebp
movl     %esp, %ebp
movl     8(%ebp), %eax
movl     12(%ebp), %ecx
leal     3(%eax), %edx
cmpl     %ecx, %edx
jne      .L4
leal     0(,%ecx,4), %eax
.L4:
popl     %ebp
ret
```

Which of the functions compiled into the assembly code shown?

fun 2

Problem 5:

This problem tests your understanding of how `for` loops in C relate to IA32 machine code. Consider the following IA32 assembly code for a procedure `dog()`:

```
dog:
    pushl    %ebp
    movl     %esp, %ebp
    movl     12(%ebp), %ecx
    movl     $1, %eax
    movl     8(%ebp), %edx
    cmpl     %ecx, %edx
    jge      .L7
.L5:
    imull    %edx, %eax
    addl     $2, %edx
    cmpl     %ecx, %edx
    jl       .L5
.L7:
    popl     %ebp
    ret
```

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

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

    result = 1;

    for (i = result; x < y; x+2) {
        result = result * x;
    }
    return result;
}
```

Problem 7:

This problem tests your understanding of how `switch` statements in C relate to IA32 machine code. Consider the following IA32 assembly code for a procedure `frog()`:

```
frog:
    pushl    %ebp
    movl     %esp, %ebp
    movl     8(%ebp), %edx
    movl     12(%ebp), %eax
    cmpl     $7, %edx
    ja       .L8
    jmp      *.L9(, %edx, 4)
    .section      .rodata
    .align 4
    .align 4
.L9:
    .long     .L8
    .long     .L4
    .long     .L8
    .long     .L5
    .long     .L8
    .long     .L4
    .long     .L6
    .long     .L2
    .text
.L4:
    movl     $7, %eax
    jmp      .L2
.L5:
    decl     %eax
    jmp      .L2
.L6:
    incl     %eax
    jmp      .L2
.L8:
    movl     $-1, %eax
.L2:
    popl     %ebp
    ret
```

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

```
int frog(int a, int b)
{
    int result;

    switch(  a  )
    {
        case   1  :

        case   5  :

            result =   7  ;
            break;

        case   3  :

            result =   b--  ;
            break;

        case   7  :

              break  ;

        case   0:6  :
            result =   b++  ;
            break;

        default:

            result =       ;
    }

    return result;
}
```