COMP.2030 Assembly LAB

NAME \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. A C function declared as

pp:

pushq %rbp

movq %rsp, %rbp

movq %rdi, -8(%rbp)

cmpq $0, -8(%rbp)

je .L1

movq -8(%rbp), %rax

addq $4, %rax

movq %rax, -8(%rbp)

.L1:

popq %rbp

ret

void pp (int \*p){ ….}

is compiled to the MIPS code on the right.

Write the C code that performs the same operation as the MIPS code labeled ‘pp’ in **no more than two** C statements.

void pp (int \*p) {

if (p) {

\*p++;

}

}

prob: pushq %rbp

movq %rsp, %rbp

movq %rdi, -8(%rbp)

movq %rsi, -16(%rbp)

movq -16(%rbp), %rax

movl (%rax), %edx

movq -8(%rbp), %rax

movl (%rax), %eax

addl %eax, %edx

movq -16(%rbp), %rax

movl %edx, (%rax)

popq %rbp

ret

1. A C function prob, declared as

void prob(int \*p1, int \*p2){ … }

is compiled to the x86 code on the right.

Write the C function prob in **ONE or TWO** C statements that does the same thing as the x86 code labeled ‘prob’.

void prob(int \*p1, int \*p2){

\*p2 += \*p1;

}

1. For a function with the prototyped

# *x = %rdi, y = %rsi, z = %rdx*

decode2:

subq %rdx, %rsi

imulq %rsi, %rdi

movq %rsi, %rax

salq $63, %rax

sarq $63, %rax

xorq %rdi, %rax

ret

long decode2(long x, long y, long z);

gcc generates the assembly code on the right.

Fill in C code below that does the same thing as the x86 assembly code labeled ‘decode2’ using only arithmetic operations and conditions (i.e., no bitwise operations like &, ^, ~, or shifts >>, <<).

long decode2(long x, long y, long z) {

y -= z;

x \*= y;

if (y % 2 == 0) {

return x;

} else {

return -x-1;

}

}

%rax (=y) shifted left by 63 bits and right shifted by 63 bits will be either 0xFFFF FFFF FFFF FFFF or 0 depending on whether the least significant bit of %rax (=y) is equal to 0 or 1. XORing against all 1’s will complement the number, which is equivalent to -x-1.