**CSC 373 Spring 2019 Prof. Lytinen**

**Final Exam**

**March 2020**

**Name: Abel Marin**

**Directions:**

This is a take-home exam. The exam is open book, open notes and computer usage is allowed. All work must be your own. Each of the 5 problems is worth 5 points, for a total possible score of 25. The exam is worth 25% of your overall grade for the quarter.

Please submit your solutions to the exam problems in the D2L submissions folder by Saturday, March 21 at 11:59 PM.

1. Assume that a C function **f** has the following prototype:

int f(char \*str, int x, char c);

Also assume that f is called as follows.

f("final", 5, ‘a’); // in ASCII, ‘a’ is 0x61

Give the value of each operand below as **f** begins to execute. Write these values **in hex.** When relevant, you should assume that the **str** array begins at memory address 0x7fffffff0000.

|  |  |
| --- | --- |
| ***Operand*** | ***Value*** |
| %rdx | 0x61 |
| 0x7fffffff0004 | 0x7fffffff0004 >>“l” of 0x6c |
| (%rdi) | 0x7fffffff0000 >>“f” or 0x66 |
| (%rdi, %rsi) | 0x7fffffff0005 >>“\0” or 0x00 |
| -2(%rdi, %rsi) | 0x7fffffff0003 >>“a” or 0x61 |

1. Consider the C function below, and the corresponding assembly language produced by gcc at level –O2.

|  |  |
| --- | --- |
| // computes x % y  int mod(int x, int y) {  if (x < y) return x;  return mod(x-y,y);  } | mod:  cmpl %esi, %edi  movl %edi, %eax  jl .L2  .L3:  subl %esi, %eax  cmpl %esi, %eax  jge .L3  .L2:  ret |

The assembly language produced by gcc at level -O2 (on the right) is not recursive. Write a non-recursive version of the C code on the left which better reflects the result of compilation. You can still use loops, if-else, etc.; goto statements are not required.

Code (also included in abelmarin\_final.c file):

int mod(int x, int y) {

while (x>=y) {

x = x - y;

}

return x;

}

1. Write a C function which mimics the behavior of the assembly language function below. Note that this time, the assembly language code is recursive (despite –O2 compilation), and your C code should likewise be recursive.

f:

cmpl $1, %edi

je .L3

xorl %eax, %eax

testl %edi, %edi

jle .L7

subq $8, %rsp

subl $1, %edi

call f

testl %eax, %eax

sete %al

addq $8, %rsp

movzbl %al, %eax

.L7:

ret

.L3:

movl $1, %eax

ret

Code (also included in abelmarin\_final.c file):

int f(int x) {

if (x == 1) return 1;

if (x <= 0) return 0;

x--;

int y = f(x);

if (y==0) return 1;

else return 0;

}

1. Below you see an interaction with gdb in which I have disassembled the functions **main** and **f**.and have also revealed some information about the contents of various memory addresses.

(gdb) disas main

Dump of assembler code for function main:

0x00000000004004e0 <+0>: sub $0x8,%rsp

0x00000000004004e4 <+4>: xor %eax,%eax

0x00000000004004e6 <+6>: callq 0x4005f0 <f>

0x00000000004004eb <+11>: mov $0x4006d5,%edi

0x00000000004004f0 <+16>: mov %eax,%esi

0x00000000004004f2 <+18>: add $0x8,%rsp

0x00000000004004f6 <+22>: xor %eax,%eax

0x00000000004004f8 <+24>: jmpq 0x4004a0 <printf@plt>

End of assembler dump.

(gdb) disas f

Dump of assembler code for function f:

0x00000000004005f0 <+0>: sub $0x18,%rsp

0x00000000004005f4 <+4>: mov $0x4006c0,%edi

0x00000000004005f9 <+9>: callq 0x400490 <puts@plt>

0x00000000004005fe <+14>: lea 0xc(%rsp),%rdx

0x0000000000400603 <+19>: lea 0x8(%rsp),%rsi

0x0000000000400608 <+24>: mov $0x4006d0,%edi

0x000000000040060d <+29>: xor %eax,%eax

0x000000000040060f <+31>: callq 0x4004c0 <\_\_isoc99\_scanf@plt>

0x0000000000400614 <+36>: mov 0x8(%rsp),%eax

0x0000000000400618 <+40>: add 0xc(%rsp),%eax

0x000000000040061c <+44>: add $0x18,%rsp

0x0000000000400620 <+48>: retq

(gdb) x/s 0x4006d5

0x4006d5: "%d\n"

(gdb) x/s 0x4006c0

0x4006c0: "Type your input"

(gdb) x/s 0x4006d0

0x4006d0: "%d%d"

1. How big is **f’s** stack frame? Explain your answer.

The stack frame of f is 24 or 0x18 bytes large. You can tell that the stack frame is this large by looking at f+0 which is sub $0x18,%rsp. This creates the size necessary for the function to run.

1. How many parameters are passed to **f**?

The function f is passed 2 parameters.

1. Assuming the user types **3 8**, write all input/output that is produced by this code.

The input would be: 3 8

The output would be: 11

1. What is the output of this program?

#include <stdio.h>

int f(int i) {

return (1 << i);

}

int main() {

int i;

for (i=0; i<8; i++)

printf("%#x\n", f(i));

}

OUTPUT:

0x1

0x2

0x4

0x8

0x10

0x20

0x40

0x80