CS 2130 Exam 2

Name _____

You MUST clearly write your computing ID on **EACH** page and put your name on the top of this page, too. Please write legibly.

Write your answers in the box labeled "Answer" when provided. In any multiple choice answer, **fill in the circle completely** for credit; checkmarks, circles, lines, or other marks will be graded as an empty circle.

If you are still writing when "pens down" is called, your exam will not be graded – even if you are still signing the honor pledge. So please do that first. Sorry to have to be strict on this!

There are 6 pages to this exam. Once the exam starts, please make sure you have all the pages. Questions are worth different amounts of points, so be sure to look over all the questions and plan your time accordingly.

This exam is CLOSED text book, closed-notes, closed-cell phone, closed-smart watch, closed-computer, closed-neighbor, etc. You may **not** discuss this exam with anyone until after the grades have been released. Please sign the honor pledge below.

Notes: Callee-saved registers: `%rbx`, `%rsp`, `%rbp`, `%r12`, `%r13`, `%r14`, `%r15`

On my honor as a student, I have neither given nor received aid on this exam.
I will not discuss the content of this exam, even in vague terms, with *anyone* other than current course staff, until *after* grades have been released.

_____

Segmentation fault (core dumped)

Page 2: Endianness, Backdoors, Patents

1. [8 points] The following 4 bytes are stored sequentially: `[0xa1, 0x7c, 0x93, 0x40]`. If these bytes represent an array of two 16-bit values, what is the value of the second element in the array given each of the following assumptions? Answer in hexadecimal.

   A. Assume little-endian storage.

Answer

   B. Assume big-endian storage.

Answer

2. [4 points] In class, we discussed copyright and patents. Why would copyright of the website describing our Toy ISA from class not be sufficient intellectual property protection to prevent others from creating clones?

   ◯ It allows copying of the description and re-implementation of the functionality

   ◯ It restricts re-implementation of the functionality, but not copying of the description

   ◯ It restricts copying of the description, but not re-implementation of the functionality

   ◯ It would actually be sufficient protection

3. [4 points] We discussed a method to create a backdoor in our Toy ISA processor. What was true of our malicious payload? (Fill in the circle completely for all that apply.)

   ◯ It must be executed by the user.

   ◯ It must be loaded into memory and read in order.

   ◯ It contains a passcode and code we want to run.

   ◯ It contains only a passcode.

   ◯ It starts with the code we want to run.

4. [4 points] What is true of hardware backdoors like the one we discussed?

   ◯ They require a large number of gates relative to the number of other gates on the chip.

   ◯ They are easy to identify upon inspection of the hardware.

   ◯ Any code the exploit runs must be compiled into the hardware.

   ◯ The backdoor might only be found after it is exploited.

Page 3: Reading Assembly

5. [24 points] Assume the first eight registers and the given segment of memory have the following values before the next few instructions.

Register	Value (hex)
rax	0x100000040
rcx	0x1000000ff
rdx	0x4
rbx	0x2130000000
rsp	0x8fffb8
rbp	0x8fffb0
rsi	0x10
rdi	0x1025

Mem Addr.	Value (hex)
0x8fffb0	0x43
0x8fffb1	0x4f
0x8fffb2	0x15
0x8fffb3	0x1a
0x8fffb4	0xab
0x8fffb5	0x8a
0x8fffb6	0xef
0x8fffb7	0x42
0x8fffb8	0x11

Mem Addr.	Value (hex)
0x8fffb9	0x34
0x8fffba	0x05
0x8fffbb	0x45
0x8fffbc	0xbf
0x8fffbd	0x19
0x8fffbe	0x33
0x8fffbf	0x27
0x8fffc0	0x9a
0x8fffc1	0x4f

Which program registers are modified, and to what values, by the following instructions? Leave spaces blank if fewer registers change than there are lines. If no registers are changed, write "none" in the first register box with no new value. *Each instruction below is independent; do not use the result of one as input for the next.* (4 points each)

`movl 0x8(%rbp), %edx`

Register	New Value

`leaq 0x8(%rbp), %rdx`

Register	New Value

`testq %rdx, %rdi`

Register	New Value

`andl -0x10(%rsp,%rdx,2), %ecx`

Register	New Value

`popw %ax`

Register	New Value

`callq foo`

Register	New Value

Page 4: Compilation Pipeline and Writing Assembly

6. [8 points] For each of the following bugs, indicate the stage of compilation that would find it. If it would not be found until run-time, write "none". The stages are:

   - **Le**xing — breaking the input into words and related tokens
   - **P**arsing — making a parse tree (an abstract syntax tree (AST))
   - **T**ype-checking — annotating the AST with data types, etc
   - **C**ode generation — creating assembly
   - **A**ssembling — turning assembly into machine code
   - **Li**nking — attaching library files to code

   A. Adding an int to a string, such as:
   ```
   const char *a = "hello";
   int x = a + 2;
   ```
   Answer

   B. Calling a function, such as `mynewfunction()`, that does not exist at all.
   Answer

7. [19 points] Consider the following C code snippet:

```
long exammath(long a, const char *c) {
    long i = 0;
    while (i <= 30) {
        a -= a;
        i += 1;
    }
    printf(c, i);
    return a;
}
```

Rearrange the x86-64 assembly instructions on the next page to implement the `exammath` function. The instructions have been grouped into three sections; for each section, reorder the instructions by writing the number of the appropriate instruction on the lines provided to the right. Some order has been provided for you and the order of instructions in one section *may* influence the ordering of instructions in another section. *Each instruction may be used at most once and only within its group. Some instructions may not be used. All blanks will be filled.* When complete, your reordered instructions (under "Proper Ordering") should fully implement `exammath`.

Page 5: Writing Assembly (continued)

Possible Instructions	Proper Ordering
1. `exammath:`	1 `exammath:`
2. `movq $30, %r12`	____
3. `pushq %r12`	____
4. `pushq %rbx`	____
5. `xorl %ebx, %ebx`	____
6. `addq $1, %rbx`	____
7. `cmpq %rbx, %r12`	____
8. `jge label1`	____
9. `jle label1`	____
10. `label1:`	____
11. `subq %rdi, %rdi`	
12. `callq printf`	____
13. `movq %rbx, %rsi`	____
14. `movq %rsi, %rdi`	____
15. `popq %r12`	12 `callq printf`
16. `popq %rax`	____
17. `popq %rbx`	17 `popq %rbx`
18. `pushq %rdi`	15 `popq %r12`
19. `retq`	19 `retq`

Page 6: C

8. [12 points] Consider the following C code:

```
char first[5] = {'f', 'y', 'i', '!', '\0'};
char *second = strdup("hello");
char *both[2] = {first, second};
```

What is printed for each of the following lines? If the program would crash or seg fault, write **crash**. *Hint:* `printf("%c", x);` *means "print the char stored in variable* `x`*."*

A. `printf("%c", (*both)[1]);` Answer

B. `printf("%c", *(both[1]));` Answer

C. `puts(&both[0][2]);` Answer

9. [5 points] Assuming that you have written C code in a file named `exam2.c`, what is the full command on the CS portal to compile your code with debugging information?

`mst3k@portal01:~$` _____

10. [12 points] Complete the following C function that takes an array of signed integers (`arr`) and a length (`n`) as parameters, replaces any of the first `n` elements in the array which are less than 0 with increasing powers of two, and returns the next power of two. For example, if `n` = 5 and `arr` contains `[0, -9, 5, -3, -2]`, the function would update `arr` to have values `[0, 1, 5, 2, 4]` and return `8`.

```
_______ powersOfTwo(___________arr, unsigned int n) {
    int twos = 1;
    // Complete this function









    return twos;
}
```