CS 2130 Exam 2

Name

You MUST write your e-mail ID on **EACH** page and put your name on the top of this page, too.

If you are still writing when "pens down" is called, your exam will not be graded – even if you are still writing 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 all the exam times have ended. Please write and sign the honor pledge below.

Segmentation fault (core dumped)

Page 2: Endianness, Backdoors

1. [10 points] Suppose an array of two 32-bit values (`[0x12345678, 0xfedcba90]`) is stored at address `0x800`. What byte is stored at address `0x807` given each of the following assumptions? Answer in hexadecimal.

   A. Assume little-endian storage.

   Answer

   fe

   B. Assume big-endian storage.

   Answer

   90

2. [5 points] We discussed a method to create a backdoor in our Toy ISA processor. Which of the following is true of our design? (Fill in the circle for all that apply.)

   ○ The exploit could be made to run any arbitrary code.

   ○ The code the exploit runs must be compiled into the hardware of the exploit.

   ○ The exploit runs only when the user executes code containing the passcode.

   ○ The payload must be loaded into memory to work.

   A,D

3. [5 points] Suppose we want to protect the intellectual property of the Toy ISA from class. Which should we use: patent or copyright? Why is that the right choice?

   Full credit for patent (and a valid reason) or neither (and a valid reason for not protecting it)

Page 3: Assembly

4. [24 points] Assume the first eight registers and the given segment of memory have the following values before the next few questions.

Register	Value (hex)
rax	0x100000040
rcx	0x12345
rdx	0x8
rbx	0x2130
rsp	0x79ffe0
rbp	0x79fff0
rsi	0x42
rdi	0x99

Mem Addr.	Value (hex)
0x79ffdf	0x00
0x79ffe0	0x42
0x79ffe1	0x15
0x79ffe2	0x1a
0x79ffe3	0xab
0x79ffe4	0x8a
0x79ffe5	0xef
0x79ffe6	0x42
0x79ffe7	0xab

Mem Addr.	Value (hex)
0x79ffe8	0x01
0x79ffe9	0x23
0x79ffea	0x45
0x79ffeb	0x67
0x79ffec	0x00
0x79ffed	0x00
0x79ffee	0x00
0x79ffef	0x00
0x79fff0	0x1f

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.*

```
leaq -0x8(%rbp), %rdi
```

rdi, 0x79ffe8

```
movl 0x4(%rsp), %ebx
```

rbx/ebx, 0xab42ef8a

```
pushq %rbx
```

rsp, 0x79ffd8

```
cmpq %rdi, %rsi
```

none (or conditional flags)

```
subl %edx, %eax
```

rax/eax, 0x38

```
retq
```

rsp, 0x79ffe8

Page 4: C and Assembly

5. [24 points] Consider the following C code snippet:

```
long reprint(const char *c, long n) {
    long i = 0;
    while (i < n) {
        puts(c);
        i += 1;
    }
    return i;
}
```

Rearrange the following assembly instructions so that they implement the code above. Write the number corresponding to each instruction on the lines provided to the right; you do not need to rewrite the entire instruction. Some order has been provided for you. *Each instruction is only used once.*

1. `addq $0x1,%rbx`	16 `reprint:`
2. `callq puts`	13 `pushq %rbp`
3. `cmpq %rbp, %rbx`	14 `pushq %rbx`
4. `jge label2`	___ 18 or 9
5. `jmp label1`	___ 9 or 18
6. `label1:`	___ 6
7. `label2:`	___ 3
8. `movq %rbx, %rax`	___ 4
9. `movq %rsi, %rbp`	___ 15
10. `popq %rbp`	2 `callq puts`
11. `popq %rbx`	___ 12
12. `popq %rdi`	___ 1
13. `pushq %rbp`	___ 5
14. `pushq %rbx`	___ 7
15. `pushq %rdi`	___ 8
16. `reprint:`	___ 11
17. `retq`	___ 10
18. `xorl %ebx, %ebx`	17 `retq`

Page 5: C

6. [8 points] Consider the following main function:

```
int main() {
    int x[6] = {11, 12, 13, 14, 15, 16};
    int y[2] = {21, 22};
    int *z[2] = {x, y};
    int *w = z[0] + 3;
    int a = *w;
    printf("%d", a);
    return 0;
}
```

What is printed? If the program would crash or seg fault, write **crash**.

Answer

14

7. [10 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:

- **L**exing — 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. Missing a variable name, such as: `int = x + 2;`

Answer

B. Declaring an array as `char c[25];` then accessing: `c[124]`

Answer

P, none

Page 6: Writing C

8. [14 points] Complete the following C function that counts the number of spaces (i.e., " ") in a given string (`str`). For example, if given the string "`This is exam 2`", the function would return `3`.

```
________ countSpaces(______________________________ str) {
    int count = 0;
    // Complete this function






    return ________________;
}
```

```
int countSpaces(const char *str) {
    int count = 0;
    // Complete this function
    char *s = str;
    while (*s != '\0') {
        if (*s == ' ')
            count++;
        s += 1;
    }
    return count;
}

int countSpaces(const char *str) {
    int count = 0;
    // Complete this function
    int i = 0;
    while (str[i] != '\0') {
        if (str[i] == ' ')
            count++;
        i += 1;
    }
    return count;
}
```

Nothing below this line will be graded