CS 2130 Exam 2

Name _____ **Computing ID** _____

You MUST clearly write your computing ID and name on the top of this page. Do this **BEFORE** you begin. 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, closed generative AI, closed smart glasses, etc. You may **not** discuss this exam with anyone until after the grades have been released. Please sign the honor pledge below.

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: Backdoors, Patents, Compilation Pipeline

1. [6 points] True/False questions on copyrights and patents. For each of the statements below, fill in the **T** circle *completely* if you think the statement is True. If you think it's False, fill in the **F** circle *completely*.

   (T) (F) Copyrighting our ToyISA website is sufficient intellectual property protection to prevent others from creating clones of our ToyISA because it restricts re-implementation of the functionality.

   (T) (F) We do not need a fully implemented version of an algorithm before patenting it.

2. [12 points] True/False questions on backdoors and the method we discussed to create a backdoor in our Toy ISA processor. For each of the statements below, fill in the **T** circle *completely* if you think the statement is True. If you think it's False, fill in the **F** circle *completely*.

   (T) (F) Our backdoor provided a way to run any arbitrary code provided by the malicious programmer.

   (T) (F) Our malicious payload contained a passcode that must be executed by the processor to run the exploit.

   (T) (F) Backdoors, like ours, are easy to identify by inspecting the hardware.

   (T) (F) Backdoors can be added in hardware, but not in software.

3. [9 points] For each of the following bugs, fill in the circle completely for the stage of compilation that would find it (or none of the above). 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
   - L**I**nking — attaching library files to code
   - **N**one of the above

   (L) (P) (T) (C) (I) (N) Missing a variable or integer, such as: `int x = + 2;`

   (L) (P) (T) (C) (I) (N) Defining `int a[4];` followed by `a[26] = 5;`

   (L) (P) (T) (C) (I) (N) Given `int foo(const char *x);` then later calling `foo(25);`

Page 3: Assembly Instructions

4. [8 points] Suppose an array of three 16-bit values (`[0x1234, 0xabcd, 0x5678]`) is stored at address `0xf700`. What byte is stored at address `0xf704` given each of the following assumptions? Answer in hexadecimal.

   A. Assume little-endian storage.

Answer

   B. Assume big-endian storage.

Answer

5. [16 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	0x200000010
rcx	0x100000020
rdx	0x8
rbx	0x7fffea04
rsp	0x7fffeb0c
rbp	0x7fffeb00
rsi	0x20
rdi	0x200

Mem Addr.	Value (hex)
0x7fffeb00	0x12
0x7fffeb01	0x56
0x7fffeb02	0x9a
0x7fffeb03	0x10
0x7fffeb04	0xde
0x7fffeb05	0xad
0x7fffeb06	0x45
0x7fffeb07	0x7c
0x7fffeb08	0x33

Mem Addr.	Value (hex)
0x7fffeb09	0xe4
0x7fffeb0a	0x59
0x7fffeb0b	0xbf
0x7fffeb0c	0x6d
0x7fffeb0d	0x91
0x7fffeb0e	0xfa
0x7fffeb0f	0x27
0x7fffeb10	0xb3
0x7fffeb11	0x4e

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)

`leaq 0x80(%rbx,%rsi,4), %rdi`

Register	New Value

`movl 0x2(%rbp,%rdx), %edx`

Register	New Value

`testq %rax, %rax`

Register	New Value

`pushl %eax`

Register	New Value

Page 4: C and Writing Assembly

6. [12 points] True/False questions on C, based on the following C code in the file `exam2.c`. For each of the statements below, fill in the **T** circle *completely* if you think the statement is True. If you think it's False, fill in the **F** circle *completely*.

```
1  #include <stdio.h>
2
3  typedef struct {
4      char *name;
5      int stats[2];
6  } profile;
7
8  int main() {
9      profile p = {.name = "Ada", .stats = {42, 67}};
10     puts(p.name + 1);
11     printf("%ld\n", *(long *)stats);
12     return 0;
13 }
```

(T) (F) After compiling your C program `exam2.c` using `clang -g exam2.c`, you can debug it on the CS portal nodes by running the command `lldb exam2.c`.

(T) (F) The size of struct `p` is at least 16 bytes and `sizeof(p.stats) = sizeof(p.name)`.

(T) (F) Line 10 prints `da`.

(T) (F) Line 11 prints `42`.

7. [20 points] Consider the following C code snippet:

```
1  int examcount(const char *a, long b) {
2      long c = 1;
3      long n = c;
4      while (c <= b) {
5          long k = n;
6          n = n + c;
7          c = k;
8      }
9      printf(a, c);
10     return 0;
11 }
```

Rearrange the x86-64 assembly instructions on the next page to implement the `examcount` 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 `examcount`.

Page 5: Writing Assembly (continued)

Possible Instructions	Proper Ordering
1. `examcount:`	1 `examcount:`
2. `movq %r12, %rax`	____
3. `movq %rsi, %r13`	____
4. `movq $1, %r12`	____
5. `pushq %r12`	____
6. `pushq %r13`	____
7. `addq %r12, %rax`	____
8. `cmpq %r13, %r12`	____
9. `jg label2`	11 `label1:`
10. `jge label2`	____
11. `label1:`	____
12. `pushq %rax`	____
13. `popq %rax`	
14. `popq %r12`	
15. `callq printf`	____
16. `cmpq %r13, %r12`	____
17. `jl label1`	____
18. `jle label1`	____
19. `label2:`	15 `callq printf`
20. `movq %r12, %rsi`	____
21. `popq %r12`	21 `popq %r12`
22. `popq %r13`	22 `popq %r13`
23. `retq`	23 `retq`
24. `xorl %eax, %eax`	

Page 6: Writing C drafted by Xinyao, please review and change it if needed.

8. [17 points] Write a C function named `smoothArray` that takes two parameters: an integer array `arr` and an unsigned integer length `n`. The function should replace each element in `arr` with the average of the next two elements, leaving the last two elements in the array of size `n` unchanged. If there are less than three elements, all elements should remain unchanged. The function should return the sum of all averages calculated (i.e., sum of all values that were updated in the array) as a 64-bit signed integer.

   Examples: If `arr = {2, 4, 6, 8, 10}` and `n = 5`, then the function updates `arr` to `{5, 7, 9, 8, 10}` and returns `21`. If `arr = {1, 6, 7, 10}` and `n = 4`, then it becomes `{6, 8, 7, 10}` and the function returns `14`

Answer

Nothing below this line will be graded

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