35/36 Questions Answered
1 question with unsaved changes

Midterm

Q1 Getting Started
0 Points

Welcome to the CSE 351 Midterm, Spring 2022!

Note: It is STRONGLY recommended that you record your answers outside of Gradescope as a backup. You may resubmit the exam as many times as you wish before the exam closes at 11:59pm SHARP on Wed 5/03/22.

We will note any corrections/clarifications on this Ed post so please **check this before submitting**!

Relax :-) You got this.
We hope this will be good learning experience! Good Luck!

The CSE 351 Staff

Q1.1 Exam Policies
0 Points

I have read the exam policies and I understand that while discussion with my group members is allowed, what I am submitting is my own work that I fully understand, in my own words.

[✓ I have questions about this policy I will clarify before continuing]

I understand

Save Answer   Last saved on **May 03 at 5:56 PM**

Q1.2 "Infinitely Surprising Alligator" ♾️ 😲 🐊

0 Points

What does ISA **really** stand for?!?!

Instruction set architecture

Save Answer Last saved on **May 03 at 5:56 PM**

Q2 Number Representation

7 Points

Q2.1 Interpreting Bits

3 Points

Interpret the 32-bit numeral `0xC0500000` in the following ways:

A signed integer:

-1068498944

An unsigned integer:

3226468352

A floating point number:

-3.25

Save Answer Last saved on **May 03 at 6:02 PM**

Q2.2 To Overflow or Not To Overflow

2 Points

For each of the following 8-bit additions, indicate which type(s) of arithmetic overflow occurred, if any.

```
  0b 11001110
+ 0b 11111110
-------------
```

[✓ Unsigned Overflow]
Signed Overflow
Both
Neither

```
  0b 10101010
+ 0b 10101010
-------------
```

Unsigned Overflow
Signed Overflow
[✓ Both]
Neither

Save Answer Last saved on **May 03 at 7:34 PM**

Q2.3 7-bits

2 Points

Assume a 7-bit two's complement integer representation.

What is -37 in this number representation (give your answer in binary and don't forget the prefix)?

0b1011011

What is the smallest magnitude negative number you can add to -37 to cause signed overflow (give your answer in binary and don't forget the prefix)?

0b1100100

Save Answer Last saved on **May 03 at 7:34 PM**

Q3 Closet Conundrum

7 Points

Veronica's closet is a mess and she wants to organize it!

Her clothes can be split into 3 main categories: tops, bottoms, and shoes. She assigns each item a 4-bit ID so it's easy to keep track of, but instead of storing items separately, she wants a way to store outfits that work well together.

One potential representation for an outfit is as follows:

```
/*----------------------------------------------------------------------
Outfit Layout:

An outfit consists of a top ID, bottom ID, and shoes ID stored in 12 bits

          MSB                                       LSB
           +-------------+-------------+-------------+
           |   top ID    |  bottom ID  |   shoes ID  |
           +-------------+-------------+-------------+
           |             |             |             |
bit offset: 12           8             4             0
  ----------------------------------------------------------------------*/
```

For example, an outfit with top #15, pants #10, and shoes #1 would be represented as follows:

```
/*----------------------------------------------------------------------

Example outfit:
          MSB                                       LSB
           +-------------+-------------+-------------+
           | 1  1  1  1  | 1  0  1  0  | 0  0  0  1  |
           +-------------+-------------+-------------+
           |             |             |             |
bit offset: 12           8             4             0
  ----------------------------------------------------------------------*/
```

Q3.1 How many outfits?

1 Point

How many unique outfits can we represent with this scheme? Two outfits are only considered the same if they have the same top, bottoms, and shoes.

4096

Save Answer Last saved on **May 04 at 3:09 PM**

Q3.2 Invert

6 Points

Given the representation for an outfit shown above:

```
/*---------------------------------------------------------------------
Outfit Layout:

An outfit consists of a top ID, bottom ID, and shoes ID stored in 12 bits

           MSB                                     LSB
            +-------------+-------------+-------------+
            |   top ID    |  bottom ID  |   shoes ID  |
            +-------------+-------------+-------------+
            |             |             |             |
bit offset: 12            8             4             0
  ---------------------------------------------------------------------*/
```

Veronica wants to have matching outfits with her friend Betty, so she assigns IDs to each of Betty's clothes that match one of hers. She wants these IDs to be related to hers, so she decides to make Betty's IDs the bitwise negation of hers.

For example, if Veronica's top has the ID `0100`, then Betty's matching top has the ID `1011`.

Given an outfit as represented in the 12-bit scheme described previously, implement this function that only bitwise negates the ID of the top.

```
/*
 *   Return outfit with the 4 bits that represent a top ID bitwise negated
 *   (i.e., turn 0 into 1 and vice versa) and the rest left
 *   unchanged.
 *
 *   Example: reverseTopID(0b 100001101111) = 0b 011101101111
 *            reverseTopID(0b 000010100011) = 0b 111110100011
*/
unsigned short reverseTopID(unsigned short outfit) {
  // bitwise negated top ID in the least-significant 4 bits
  unsigned short invertedTopID = [blank 1];

  // bitwise negated top ID in the correct place, bottom and shoes IDs
  // set to 0
  unsigned short shiftedInvertedTopID = invertedTopID << 8;

  // outfit with the top ID field zeroed out
  unsigned short outfitWithoutTopID = outfit & [blank 2];

  return [blank 3];
}
```

Blank 1:

(~(outfit >> 8)) & 0x0F

Blank 2:

outfit & 0x0FF

Blank 3:

outfitWithoutTopID |
shiftedInvertedTopID

Save Answer Last saved on **May 04 at 3:40 PM**

Q4 Pointers & Memory
7 Points

Both parts of this question use the following snippet of C code with the current state of memory (values in hex) shown below. Assume a 64-bit x86-64 machine (little-endian). It may be helpful for you to review **hw3**.

```
char* cp = 0xA2;
int** nums = 0xAC;
int* ip = 0xBC;
```

Word Addr	+0	+1	+2	+3	+4	+5	+6	+7
0xA0	E4	00	48	69	20	52	75	74
0xA8	68	21	00	BA	11	37	66	72
0xB0	E0	00	FE	CA	FE	F0	0D	FF
0xB8	84	3F	02	4F	8E	F3	F6	E5
0xC0	CC	00	00	00	A0	00	00	00
0xC8	00	00	00	00	DE	AD	BE	EF

Q4.1 Memory Usage
1 Point

How many bytes are allocated by the declarations and initializations given above? Answer with just the number of bytes in decimal.

24

Save Answer Last saved on **May 04 at 3:43 PM**

Q4.2 What is it?
1 Point

If we treat `cp` as a string literal (i.e. an array of ASCII characters), what is its string value? Link to ASCII table

Hi Ruth!

Save Answer Last saved on **May 04 at 3:43 PM**

Q4.3 C and Memory: exp1

1 Point

Compute the C type and value for the following C expression, **making sure to pay attention to bit widths**:

`*ip`

Expression 1 Type:

int

Expression 1 Value: (in hex, with prefix)

0xE5F6F38E

Save Answer Last saved on **May 04 at 7:04 PM**

Q4.4 C and Memory: exp2

1 Point

Compute the C type and value for the following C expression, **making sure to pay attention to bit widths**:

`cp + 2`

Expression 2 Type:

char pointer

Expression 2 Value: (in hex, with prefix)

0xA4

Save Answer Last saved on **May 04 at 7:04 PM**

Q4.5 C and Memory: exp3

1 Point

Compute the C type and value for the following C expression, **making sure to pay attention to bit widths**:

`*(nums[3])`

Expression 3 Type:

int

Expression 3 Value: (in hex, with prefix)

0x694800E4

Save Answer Last saved on **May 04 at 7:04 PM**

Q4.6 C and Memory: exp4

1 Point

Compute the C type and value for the following C expression, **making sure to pay attention to bit widths**:

`cp[-1] + 4`

Expression 4 Type:

char

Expression 4 Value: (in hex, with prefix)

0x04

Save Answer Last saved on **May 04 at 4:00 PM**

Q4.7 C and Memory: exp5

1 Point

Compute the C type and value for the following C expression, **making sure to pay attention to bit widths**:

`*((short*) ip - 5)`

Expression 5 Type:

short

Expression 5 Value: (in hex, with prefix)

0xCAFE

Save Answer Last saved on **May 04 at 4:00 PM**

Q5 Pointers & Memory Design

4 Points

Tom & Angela want to design a new programming language that is similar to C, but with a different implementation of pointers. Pointers are no longer typed: instead, there is a single data type called `ptr` that only holds an address. (So `int*` and `float*` no longer exist, instead it's just `ptr`.)

```
char* x = 0xA2;  // => ptr x = 0xA2;
int*  y = 0xAC;  // => ptr y = 0xAC;
int** z = 0xBC;  // => ptr z = 0xBC;
```

What are two disadvantages of implementing pointers this way?

Disadvantage 1 & Explanation (two sentences max):

int** z means z is a pointer to a pointer, but if we use ptr z, then it just says z is a pointer. They will have different value if we deference them.

Disadvantage 2 & Explanation (two sentences max):

Using ptr will always dereference whatever data store in that address, the compiler won't know if it should read how many bytes in the ram.

Save Answer Last saved on **May 04 at 4:04 PM**

Q6 C & Assembly

17 Points

Given the following assembly for the function `sunny`, answer the questions below.

```
sunny:
        movl    $0, %r8d        # Line 1
        jmp     .L2             # Line 2
.L3:
        addl    $1, %r8d        # Line 3
.L2:
        cmpl    %esi, %r8d      # Line 4
        jge     .L5             # Line 5
        movslq  %r8d, %r10      # Line 6
        addq    %rdi, %r10      # Line 7
        cmpb    %dl, (%r10)     # Line 8
        jne     .L3             # Line 9
        movb    %cl, (%r10)     # Line 10
        jmp     .L3             # Line 11
.L5:
        ret                     # Line 12
```

Q6.1 Can I have a signature?

5 Points

The function `sunny` has the following function signature. Fill in the blanks with the appropriate types. For each blank, **in around 2 sentences**, justify the type you have chosen and **cite line numbers from the assembly code** that helped you reach your answer.

```
_____ sunny(___ a, ___ b, ___ c, ___ d)
```

Return Type

void

Return Type justification

%rax has never been accessed

Data type of `a`

char pointer

`a` Type justification

%rdi register is 8 byte in size, also the suffix of add is q which means 8 bytes. Since we need to dereference a, it has to be a pointer.

Data type of `b`

int

`b` Type justification

%esi is the lowest 4 bytes of %rsi, also the suffix of cmp is l which indicate the size is 4 bytes

Data type of `c`

char

`c` Type justification

%dl is the lowest byte of %rdx, also the suffix of cmp is b which indicate the size is one byte

Data type of `d`

char

`d` Type justification

%cl is the lowest byte of %rcx, also the suffix of mov is b which indicate the size is one byte

Save Answer   Last saved on **May 04 at 7:39 PM**

Q6.2 Loop-de-doo

5 Points

The function `sunny` shown above contains a loop. Given the code skeleton below, fill out the loop's initialization, condition, update statement, and body (note: the body may contain other control structures). If the loop contains variables local to its scope, you may name these variables anything you would like. Hint: the loop initialization statement occurs on `line 1`

```
for (<init> ; <condition> ; <update>) {
    <loop body>
}
```

Loop initialization

int i = 0

Loop Condition

```
i < b
```

Loop update

```
i++
```

Loop body

```
char* m = a
if (*(m + i) == c) {
    *(m + i) = d;
}
```

Save Answer ***Unsaved Changes**

Q6.3 What was the reason!?!?!

3 Points

At a **high level**, what does the function `sunny` do? (Note, we don't want to know that it contains a loop and takes in parameters, we want to know why?). **Keep answers to two sentences maximum.** Be sure to explain how each argument is used inside the function and what is returned if anything.

Enter your answer here

Save Answer Last saved on **May 04 at 10:35 PM**

Q6.4 LEA!

2 Points

Consider lines 6 and 7 in the assembly.

```
movslq  %r8d, %r10      # Line 6
addq    %rdi, %r10      # Line 7
```

One of your awesome TAs suggests these lines can be rewritten as a single `lea` instruction using only registers `%r8d`, `%r10`, and `%rdi`.

In two sentences maximum, explain why this is **not** possible.

movslq is move and sign-extend a value from a 32-bit source to a 64-bit destination. You cannot do the sign extension in lea instruction.

Save Answer Last saved on **May 04 at 4:20 PM**

Q6.5 Join the Rebellion

2 Points

`Line 5` contains a `jge` instruction.

We are feeling a bit rebellious today and want to change this to a `jle` instruction. Change **Lines 4 & 5 only** so that `Line 5` uses `jle` and the function would still have the **same behavior**.
Answer in the form:

```
Line 4: instruction operand1, operand2
Line 5: jle <label>
```

cmpl %r8d, %esi
jle .L5

Save Answer Last saved on **May 04 at 5:43 PM**

Q7 Can't spell function without "fun"!

7 Points

Download the `mystery` executable to run on either attu or the CSE VM using the following command:

```
wget https://courses.cs.washington.edu/courses/cse351/22sp/exams/mystery
```

To be able to run the file and trace though it you will also need to run the following command after you have downloaded the file: `chmod +x mystery` (in the same directory as the file).

You should trace through the file to find the needed information to answer the questions using `gdb`. If you would like, you can also run the following command `objdump -d mystery > mystery.s` to get a file called `mystery.s` that has the assembly code for this whole executable. You should first place a breakpoint in `main` which will call 2 functions, you are tasked with figuring out what those functions do.

The following questions are based on this assembly code. First we will examine the function `func1`.

Q7.1 Stack frames

1 Point

How many stack frames deep will the call to `func1` get at its deepest (including the stack frame for main)?

7

Save Answer Last saved on **May 04 at 6:02 PM**

Q7.2 Return address #1

1 Point

What is the return address that gets stored on the stack when `func1` is called from main (in hex, with the prefix, give an absolute address, not a relative address)?

0x00401191

Save Answer Last saved on **May 04 at 7:45 PM**

Q7.3 Num Arguments

1 Point

How many argument(s) does `func1` take?

1

Save Answer Last saved on **May 04 at 7:46 PM**

Q7.4 Arguments

1 Point

What are the types of the argument(s) to `func1`? If they are pointer(s), what section of memory are they pointing to?

char pointer, literals

Save Answer Last saved on **May 04 at 10:36 PM**

Q7.5 Argument Articulation

1 Point

Which instruction(s) tell you about the parameters to `func1`? (Hint: They might be in main as well!)

```
mov    $0x402010,%edi (main)
cmpb   $0x0,(%rdi) (func1)
add    $0x1,%rdi  (func1)
```

Save Answer Last saved on **May 04 at 8:25 PM**

Q7.6 High Level
2 Points

At a **high level**, describe what the function `func1` accomplishes. **Keep answers to two sentences maximum.** Be sure to explain how each argument is used inside the function and what is returned if anything.

count how many characters in a string that we passed into the function

Save Answer Last saved on **May 04 at 10:50 PM**

Q8 Maybe you can spell function without "fun" (see: Method)
8 Points

The following questions are based on the same executable as the above question, focusing now on the function `func2`.

Q8.1 Procedure
1 Point

How big, in bytes, is the code for `func2` (give your answer in decimal)?

27

Save Answer Last saved on **May 04 at 8:31 PM**

Q8.2 Num Arguments
1 Point

How many argument(s) does `func2` take?

2

Save Answer Last saved on **May 04 at 8:32 PM**

Q8.3 Arguments

2 Points

What are the types of the argument(s) to `func2`? If they are pointer(s), what section of memory are they pointing to?

first argument is an int pointer and the second argument is an integer

Save Answer Last saved on **May 04 at 10:49 PM**

Q8.4 Opposite of Low Level

2 Points

At a **high level**, describe what the function `func2` accomplishes. **Keep answers to two sentences maximum.** Be sure to explain how each argument is used inside the function and what is returned if anything.

We pass in two parameter a and b, the function will output the integers stored in address a, a+1, a+2 all the way to a+b

Save Answer Last saved on **May 04 at 10:51 PM**

Q8.5 Return to sender

1 Point

What is the return address that gets stored on the stack when `func2` is called from `func2`?

0x0040113d

Save Answer Last saved on **May 04 at 8:38 PM**

Q8.6 Caller? Hardly know 'er!

1 Point

Why do we need the instruction `0x40112d push   %rbx` and the instruction `0x40113f pop    %rbx` ?

Pushing and poping registers save and restore the values of registers to memory temporarily so they don't get overwritten.

Save Answer Last saved on **May 04 at 10:40 PM**

Q9 Design Questions

4 Points

The following questions do not include any code for you to examine. Instead, these are open-ended design prompts without a single correct answer. What's most important is that you *explain your thought process* to demonstrate your understanding.

Q9.1 Return

2 Points

Consider our current convention of using `%rax` as the designated return register. **Supposing you could introduce a new register**, would you add a convention of a second return register if you could? Explain your reasoning, considering the advantages and disadvantages of your approach.

I would not, if the programmer really wants to return multiple value, they can create an array or struct outside the function and modify the value in the function.
Advantage: the compiler won't get confused which register value is used to return
Disadvantage: the compiler will spend extra time compling the data structure(array, struct)

Save Answer Last saved on **May 04 at 10:53 PM**

Q9.2 Return return

2 Points

Would a second return register be a caller or callee saved register? Explain your reasoning.

Hint: don't just say '____, because `%rax` *is ____', explain the reasoning for why a return register must be ____-saved.*

caller saved register, because the caller wants to restore the register value after procedural calls, this is not a temporary value.

Save Answer Last saved on **May 04 at 10:28 PM**

Q10 Wrap Up
1 Point

Q10.1 Collaboration
0.5 Points

Did you work on the exam alone or collaborate with others? If you collaborated with others, who did you collaborate with? Please list **full names** and **UWNetIDs** of everyone you collaborated with.

Total number of people in my group (including me) was:

2

Names and UWNetIDs (e.g. rea2000, NOT student number)

tianyl28, jzhang65

Save Answer Last saved on **May 04 at 8:39 PM**

Q10.2 Time
0.5 Points

Reminder: Did you check the corrections/clarifications Ed post? Please do so before submitting!

Finally, answer this question: How long did you personally spend taking the exam (in hours)? (not including time you spent studying beforehand)

12

Save Answer Last saved on **May 04 at 8:40 PM**

Save All Answers

Submit & View Submission