

Final Exam



Spring 2025

This exam consists of 8 substantive questions. You have 3 hours – budget your time wisely. Assume the armlab/gcc217 environment unless otherwise stated in a problem.

Do all of your work on these pages. You may use the provided blank spaces for scratch space, however this exam is preprocessed by computer, so for your final answers to be scored you must write them inside the designated spaces and fill in selected circles and boxes completely (● and ■, not ✓ or ✕). Please make text answers dark and neat.

Name:

Sample Solutions

NetID:

Precept:

<input type="radio"/>	P01 - MW 1:30 Xiaoyan Li	<input type="radio"/>	P04 - TTh 12:30 Tolulope Oshinowo	<input type="radio"/>	P08 TTh 3:30 Yang Duan
<input type="radio"/>	P02 - MW 3:30 Xiaoyan Li	<input type="radio"/>	P06 - TTh 1:30 Indu Panigrahi	<input type="radio"/>	P09 TTh 7:30 Andrew Sheinberg
<input type="radio"/>	P03 - TTh 12:30 Lana Glisic	<input type="radio"/>	P07 - TTh 1:30 Ryan Oet		

This is a closed-book, closed-note exam, except you are allowed one two-sided study sheet. Please place items that you will not need out of view in your bag or under your working space at this time. Electronic devices such as cell phones, laptops, smartwatches except to check the time, etc. may not be used during this exam.

This examination is administered under the Princeton University Honor Code. Students should sit one seat apart from each other and refrain from talking to other students during the exam. All suspected violations of the Honor Code must be reported to [honor@princeton.edu](mailto:honor@princeton.edu).

In the box below, copy **and** sign the Honor Code pledge before turning in your exam:  
*"I pledge my honor that I have not violated the Honor Code during this examination."*

Exam stats:

Min: 24/70

Mean: 56/70 Standard Deviation: 10.31/70

Median: 59/70

Max: 70/70

X \_\_\_\_\_

## Question 0: *Et ego [N.] spondeo, voveo ac iuro.*

0 points

Please don't make the course staff's life harder: make sure you have filled out your name, NetID (i.e., armlab login – not PUID, not email alias), precept and the Honor Code pledge text on the front page. Sign your name once you have finished the exam.

## Question 1: A 2<sup>nd</sup> chance at midterm topics? *Accepto.*

5 points

For each of the code snippets below, indicate whether the equality operation always evaluates to true (1), always evaluates to false (0), or depends on the system because the behavior is not guaranteed by the C90 standard.

	TRUE	FALSE	DEPENDS
a. <code>sizeof(signed long) == sizeof(unsigned long)</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
b. <code>sizeof(size_t) == sizeof(unsigned long)</code>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
c. <code>sizeof(sizeof(char)) == sizeof(char)</code>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
d. <code>char ac[] = "muratarum"; ac[9] == (ac[8] - *ac)</code>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
e. <code>int ai[] = {0, 4}; int *pi = &amp;ai[1]; (pi - ai) == 4;</code>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

## Question 2: ~~Four ballots per day~~ stages per build

5 points

Which stage of the build process (**P**reprocessor, **C**ompiler, **A**sembler, or **L**inker) is responsible for each of the following operations:

	P	C	A	L
a. Combines object files' contents	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
b. Resolves references to external library implementations	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
c. Optimizes to use callee-saved registers	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
d. Handles macro replacement, e.g., EOF (aka substitution or expansion)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
e. Checks for unterminated compound statements (aka blocks)	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

### Question 3: *Urbī et Orbi* and all the places variables live 12 points

Consider the following program:

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

char fmt[] = "%lu\n";

size_t fun(char *x) {
    static size_t i;
    return ++i;
}

int main(int argc, char **argv) {
    char *pc;
    size_t ulResult = 0;
    if(argc < 2)
        return EXIT_FAILURE;

    pc = argv[1];
    while(*pc)
        ulResult = fun(pc++);
    printf(fmt, ulResult);
    return 0;
}
```

a. Fill in the memory section and size on armlab for each variable:

	SECTION	# BYTES
argv	Stack	8
x	Stack	8
pc	Stack	8
i	BSS	8
fmt	Data	5

b. Replace the loop in main with a single statement that calls a function from the C string library, such that the same result would be printed in the subsequent printf:

```
ulResult = strlen(pc);
```

## Question 4: *Quo nomine vis vocari?*

11 points

Consider the following C function, whose identifier naming leaves much to be desired:

```
struct Node {
    int payload;
    struct Node *next;
};

struct Node *mystery(struct Node *arg) {
    struct Node *var;

    if(!arg) return NULL;

    var = mystery(arg->next);
    if(!var) return arg;

    arg->next->next = arg;
    arg->next = NULL;
    return var;
}
```

- a. Determine what the function does, then for each specified identifier give a more semantically meaningful name to replace its current vague identifier:

mystery

var

arg

reverseList

newHead

head

(Many options were possible, of course.)

- b. Describe three cases that would be good function boundary tests for mystery, in less than 10 words each.

- Full credit for 3 reasonable cases given, among these:
  - \* empty
  - \* single node
- - \* 2-node
  - \* 3-node
- - \* even number of nodes
  - \* odd number of nodes
  - \* cyclic

## Question 5: Leo XIV has none of the long name issues 7 points

`$ ./a.out`  
 What is your name?  
*Christopher Moretti*  
 Thank you, Christopher Mor  
 tti.  
 The answer to life, the universe, and everything is 6911092

??? (!)  
 (Note: this is just the number that's actually printed when you run the code. It's not an attempt to Easter egg a phone number or anything like that. Please don't try to call it. Doing so almost certainly won't give you the answer to life, the universe, and everything.)

Lecture 22 gave an example of a buffer overrun, in which an `int` variable was corrupted with character input that overran the buffer. The slide above shows the result, where printing the overwritten variable results in four consecutive chars stored in memory – the characters: `'t', 't', 'i', '\0'` – being interpreted as the `int` value `0x00697474`.

- a. This result relies on many non-portable factors, e.g., the stack memory layout of the function and that the size of four chars is also the precise size of an `int`. List two more armlab properties that are not guaranteed by the C standard that caused the bytes `{'t', 't', 'i', '\0'}` to be interpreted as `0x00697474`.

The best answers were: ASCII; Little endian  
 Answers that did not earn credit included: properties listed in problem, signedness of char, `'\0'` encoded as 0 (guaranteed by C standard), different architecture/ML/etc.

- b. I asked ChatGPT to convert `0x00697474` to decimal and it got the wrong answer (6900092, instead of the number on the slide). Prove yourself more reliable than ChatGPT by converting the sum of the hex addition `0xC05 + 0x217` to decimal:

**3612 (converted from 0xE1C)**

You may refer to this abbreviated ARM assembly language reference for Q6 – Q8.

Instruction(s)	Description
{add,sub,lsl,sdiv} dst, src1, src2	dst = src1 {+, -, <<, / } src2
{beq,bne} label	Go to label if comparison was {"equal", "not equal"}
{b,bl} label	{Unconditionally go to , Call function at} label
cmp first, second	Compare first with second, setting bits in PSTATE
ldr dst, [src]	Load 4 or 8 bytes pointed to by src into dst
ldrb dst, [src]	Load 1 byte pointed to by src into dst
str src, [dst]	Store 4 or 8 bytes in src to memory pointed to by dst
mov dst, src	Copy contents of register src to register dst
ret	Return to address pointed to by x30
R0 – R7 and R0 (w or x)	Used for arguments to and return value from functions
R0 – R7 and R9 – R15 (w or x)	Caller-saved scratch registers

## Question 6: Breaking a Cardinal rule of function calls 6 points

Consider the following function, modified from the Euclid program you saw in precept:

1	.equ LABS2, 8	27	// lTemp = lAbs1 % lAbs2
2	.equ LABS1, 16	28	// rem = (dividend -
3	.equ LTEMP, 24	29	// (quotient * divisor))
4	.equ L2, 32	30	ldr x0, [sp, LABS1]
5	.equ L1, 40	31	ldr x1, [sp, LABS2]
6		32	sdiv x2, x0, x1
7	gcd:	33	mul x3, x2, x1
8	sub sp, sp, 48	34	sub x4, x0, x3
9	str x0, [sp, L1]	35	str x4, [sp, LTEMP]
10	str x1, [sp, L2]	36	
11		37	// lAbs1 = lAbs2
12	// lAbs1 = labs(l1)	38	ldr x0, [sp, LABS2]
13	ldr x0, [sp, L1]	39	str x0, [sp, LABS1]
14	bl labs	40	
15	str x0, [sp, LABS1]	41	// lAbs2 = lTemp
16		42	ldr x0, [sp, LTEMP]
17	// lAbs2 = labs(l2)	43	str x0, [sp, LABS2]
18	ldr x0, [sp, L2]	44	
19	bl labs	45	// goto gcdLoop
20	str x0, [sp, LABS2]	46	b gcdLoop
21		47	
22	gcdLoop:	48	loopEnd:
23	// if (lAbs2 == 0) goto loopEnd	49	ldr x0, [sp, LABS1]
24	ldr x0, [sp, LABS2]	50	add sp, sp, 48
25	cmp x0, 0	51	ret
26	beq loopEnd		

- a. I asked ChatGPT what the next instruction after the ret on line 51 is executed will be, and it answered that this function returns to the next instruction after the bl gcd in its caller. That's not the case here – there's a bug! Why does this function not return to its caller, and to where does it return instead? Be specific.

The function doesn't save its initial x30, so when bl changes x30, the original x30 value is lost. Thus, the ret will return to the most recent function call's return location, which is L20.

- b. Imagine unconditional branch (b) instructions were no longer a part of ARM assembly language. Replace line 46 with no more than 2 instructions that would result in the same behavior as the existing b gcdLoop instruction.

There were many reasonable approaches:

(cmp xk, xk OR cmp xzr, 0) then beq gcdLoop

cmp xzr, #AnyNonZeroImmediate then bne gcdLoop

cbz xzr, gcdLoop

just bne gcdLoop (because L25 must compare not equal or else we'd break on L26)

## Question 7: Does ChatGPT need more *Scrutatoribus*? 14 points

There were many potentially correct orderings, with two broad patterns for handling the ulCount variable:

- \* the "faithful" C translation -- read, update, and set ulCount on the stack inside the loop, such that L, E, and N all exist between B and F.

- \* load ulCount into x3 before the loop, then ulCount in register x3 within the loop, with the str of the contents back into memory either within or after the loop, such that L comes before B, and N comes anywhere after E.

Key ordered subcomponents (not necessarily contiguous depending on design decisions, as noted above) were:

- \* MJG (the while loop conditional)
- \* MIHEC (the if statement)
- \* DFA (the loop update step)
- \* AK (setting the return value)

The most common sequences for each of the two approaches above were:

- \* BMJGIHLENCDFAK
- \* LBMJGIHECDFANK

```
#include <stddef.h>
#include <assert.h>
size_t Str_count(const char* pcStr, char c) {
    size_t ulCount = 0;
    assert(pcStr != NULL);
    while(*pcStr != '\0') {
        if(*pcStr == c)
            ulCount++;
        pcStr++;
    }
    return ulCount;
}
```

```
.section .text
.global Str_count
Str_count:
    sub sp, sp, 16
    str xzr, [sp]
    str x30, [sp,8]

    // Your instructions go here

    ldr x30, [sp, 8]
    add sp, sp, 16
    ret
```

These are the 14 instructions you will use to fill the body of Str\_count. Fill in one letter on each blank below, in the order in which they should appear in the function.

A	Done:
B	Loop:
C	NoCount:
D	add x0, x0, 1
E	add x3, x3, 1
F	b Loop
G	beq Done

H	bne NoCount
I	cmp w2, w1
J	cmp w2, wzr
K	ldr x0, [sp]
L	ldr x3, [sp]
M	ldrb w2, [x0]
N	str x3, [sp]

## Question 8: *Habemus iterum "Iterum"*

10 points

I asked ChatGPT to solve the Fall 2022 final exam “Encore” problem, which required generating the machine code for the instruction `adr x1, label12`. In that problem, `label12`’s address was `0x217217` and the address of the `adr` instruction itself was `0x214127`. The correct answer was `0x10018781`.

Here was the response from ChatGPT:

The `adr` instruction encodes the difference between the label address and the current PC as a signed 19-bit immediate in multiples of 4 (because ARM instructions are 4 bytes wide).

### Step 1: Compute the byte offset:

csharp

Copy

Edit

```
0x217217 - 0x214127 = 0x2AF0 = 10992 (decimal)
```

### Step 2: Divide by 4 to get the immediate field:

yaml

Copy

Edit

```
10992 / 4 = 2748
```

This value (2748) is placed in bits [23:5] of the instruction.

### Step 3: Register `x1` is encoded in bits [4:0] as `000012` (or just 1 decimal).

### Step 4: The top 8 bits of an `adr` instruction are always `0b00010000` → `0x10`.

Putting it all together:

- Top bits: `0x10 << 24 = 0x10000000`
- Immediate field: `2748 << 5 = 0x35A00`
- Destination register: `x1 = 0x1`

Final instruction:

Copy

Edit

```
0x10000000 | 0x35A00 | 0x1 = 0x10035A01
```

✅ Final Answer:

Copy

Edit

```
0x10035A01
```

Your questions and a reference sheet for `adr` appear on the next page.



See page 10 for discussion on part a.

- a. ChatGPT made major errors in (1) the initial paragraph, (2) Step 1, (3) Step 4 bold header, and (4) a bullet point in Step 4. Find and circle each of these four errors in the ChatGPT response on the previous page – be precise as to which portion constitutes the error.

For (3), the error is **not** the use of `0b...` as a shorthand for a binary number. This notation isn't valid C90 but is valid in some languages and eminently reasonable.

Note that each error is identified only once, even if it cascades through the answer. For example, if ChatGPT claimed  $1 + 1 = 3$ , you would circle that claim, not subsequent uses of the erroneous 3 later in the calculation.

Hint: there's also a minor error in Step 2: dividing by 4 gets the immediate hi field

- b. One last time, prove yourself better than ChatGPT. In the box below, give the (correct!) hex machine code for the instruction `adr x26, label3` where `label3`'s address is `0x4217A7` and the address of the `adr` instruction itself is `0x402170`. (ChatGPT says the answer to this question is `0x073F2E9A` ... do you trust it?)

0x700FB1BA

-Offset: 0x1F637

7: from rightmost 2 bits of the offset, surrounded by opcode

00: remainder of opcode and leftmost 4 bits of orange field

FB1: 12 bits in the middle hexits of orange field

B: three rightmost bits of orange field (3rd through 5th of offset), followed by leftmost bit of register.

A: four rightmost bits of register

msb: bit 31

lsb: bit 0

$\downarrow$

0 i i 1    0 0 0 0    i i i i    i i i i    i i i i    i i i i    i i i r    r r r r     $\downarrow$

Specifies *relative* offset of label (data location)

19 High-order bits of offset in bits 5-23

2 Low-order bits of offset in bits 29-30

Destination register in bits 0-4

As a reminder, the relative offset calculation is the label address minus the address of the instruction's address – ChatGPT *did* manage to get that order right in its response on the previous page.

*Fumus albus!* Question 8 was the last question!

*Extra omnes!* I hope you take with you from COS 217 many useful skills and a good deal of pride for having persevered through it, even if perhaps you'd like to lock the memories of the midterm exam, late night debugging sessions, etc. away *cum clave* and throw away the key.

(The space below is intentionally left blank. You may use it for scratch work, but any answers given below will not be graded.)

The 4 errors were:

(1) the immediate value is 21 bits, and does not have an alignment requirement that allows it to be in multiples of 4.

(2) the subtraction should yield 0x30F0, not 0x2AF0.

(3) The middle bits of the first hexit are not always 00.  
(This would disregard the immlo field.)

(4)  $2748 \ll 5$  is not 0x35A00, but rather 0x15780.

(Note that the use of the term "immediate field" here should be "immediate hi field", which is a cascaded error from a previous step, as mentioned in the problem.)

Q0: And I, [name], so promise, pledge and swear.

Q1: I accept; surrounded by walls.

Q3: To the City and the World.

Q4: By what name do you wish to be called?

Q7: Scrutineers – vote checkers.

Q8 (taking some liberties): We have, again, "Encore"

This page: White smoke!; Everyone out!; with a key.