CS33 Fall Midterm Exam, 2025 Name: _____

UID: _____

(-1 points if gradescope can't read your uid or name :p)

This is an open book, open notes exam, but you cannot share books/notes. Please follow the university guidelines in reporting academic misconduct.

Notes:

- **If the architecture of the machine is not specified, assume that the question is being asked in the context of a 64-bit little endian x86 machine, running Linux.**
- **Some problem's correct answers may be blank (e.g. multiple choice), please simply leave them blank.**
- **There is an ASCII table on the last page.**

Please wait until everyone has their exam to begin. We will let you know when to start.

Good luck! (ANSWERS ARE IN WHITE TEXT, highlight or copy)

	Points Possible (extra credit)
1. Connect my Call	5
2. Union	12
3. Equivalence Checking	8
4. A Bit more manipulation	9 (+2)
5. Derpy Float	8
6. All or nothing	18
7. Summon the Dragon!	7
8. Data structure Matching	(5)
9. No Cheat Codes	(5)
Total	67 (+12EC)

Question 1. Connect my Call? (5 pts)

Here's a function and its assembly with missing prologue and epilogue (the part of the function where the stack is allocated or deallocated).

```
int function(int* p) {
    int array[10];
    int temp = *p;
    return func1(array,p) + temp;
}
```

```
    movq    %rdi, %rsi
    movl    (%rdi), %ebx
    movq    %rsp, %rdi
    call    func1
    addl    %ebx, %eax
```

Here are some possible prologues/epilogues, separated by "---".

Answer A	Answer B	Answer C	Answer D	Answer E

`pushq %rbx`
`subq $48, %rsp`
`---`
`addq $48, %rsp`
`popq %rbx`
	`pushq %rdi`
	`---`
	`popq %rdi`
		`subq $10, %rsp`
		`pushq %rbx`
		`---`
		`popq %rbx`
		`addq $10, %rsp`
			`pushq %rdi`
			`pushq %rbx`
			`subq $48, %rsp`
			`---`
			`addq $48, %rsp`
			`popq %rbx`
			`popq %rdi`
				`pushq %ebx`
				`subq $400, %rsp`
				`---`
				`addq $400, %rsp`
				`popq %ebx`

Which of the above is functionally correct and the most efficient?
(write below in the box as a CAPITAL letter, e.g. "A")

Question 2. Unions (12 pts)

Let's discuss some properties of this arbitrary union. As per usual, make sure to assume little endian.

```
union a {
        char a;
        short b;
        float c;
        char arr[10];
} thing;
```

	Answer Here
a) What is the **size of the** biggest primitive in this union?	
b) What is the sizeof(thing)?	

Suppose the union starts off initialized to all zeroes. Then suppose we copied the string "?" into arr, like this:

```
strcpy(thing.arr,"?");
```

c)What would this print? `printf("%c\n",thing.a); //print character`	
d)What would this print? `printf("%x\n",thing.b); //print hex`	
e) What would this print? (PLEASE don't do the hard math, just list the integer this is closest to) `printf("%f\n",thing.c); //print float value`	
f) Just for fun, but no actually for points, what would this value print if we pretend we're on a big endian machine? `printf("%f\n",thing.c); //print float value`	

Question 3. Equivalence Checking (8 pts)

For each pair of assembly snippets, list if they always return the same thing.

		YES or NO (all caps)
`leal (%rdi,%rdi), %eax`	`mov (,%rdi,%2), %eax`	
`movl %edi, %eax` `shrl $2, %eax`	`movl %edi, %eax` `sarl $2, %eax`	

`movl %edi, %eax`
`imull $33, %eax`
	`movl %edi, %eax`
	`sall $5, %eax`
	`leal (%rax,%rdi), %eax`	

`func:`
`cmpl %esi, %edi`
`jl .L4`
`movl $0, %eax`
`ret`
`.L4:`
`leal (%rdi,%rsi), %eax`
`ret`
	`func:`
	`leal (%rdi,%rsi), %eax`
	`cmpl %esi, %edi`
	`movl $0, %edx`
	`cmovge %edx, %eax`
	`ret`	

Question 4. A Bit more manipulation (9 + 2 Extra Credit)

Your friend gave you the solution to three of the datalab questions (what a nice friend!), but they neglected to tell you which they were. Try to decipher them! **(3 pts each)**

Function	***What does this function do?*** *You must use 5 or fewer words, in legible handwriting, to describe each function.*
`return ~(1 << 31);`	
`return (x>>31) & 0x1;`	

`int xn = x>>31;`
`int yn = y>>31;`
`return !((xn & !yn) |`
`((!(xn ^ yn)) & (~y+x)>>31));`	

`(2 pts extra credit)` `return (!!x) & (!(x+x));`	

Question 5. Derpy Float (8pts)
Suppose we have a the smallest possible floating point format, where sign, exp, and frac fields all take 1 bit. Following the rules of IEEE floating point we learned in class, including for special values, what are all the possible values of each bit pattern:

Number (binary)	What does it represent?
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	



Question 6. All or nothing

For the following multiple choice questions, **select all applicable answers (if there are any).** Put your answers in the table on each page. No partial credit 😛

1. One byte corresponds to how many hex digits?
    A. 1
    B. 2
    C. 3
    D. 4
    E. 8
2. What restrictions does the calling convention impose?
    A. Whether memory is accessed in the function
    B. What registers are used for function arguments
    C. Maximum size of the stack
    D. What register holds the return value
    E. Function deallocates any stack space it allocated before returning
    F. Which registers are preserved across function calls
3. Which of the following instructions could be removed, and just done by a combination of some other existing instructions?
    A. lea
    B. pop
    C. push
    D. ret
    E. call
4. Suppose you use objdump -D to print out the disassembly of a file, and you see this instruction. Which of the following hex numbers could get pushed onto the stack as a result of the call?

```
4a4c11:       e8 6a f0 f5 ff     call   403c80 <someFunc>
```

    A. 4a4c12
    B. 4a4c13
    C. 4a4c16
    D. 403c80
    E. 403c81
    F. 403c85

Answer Table (list any correct answers IN ALL CAPS: A,B,...)

1	
2	
3	
4	

5. Which registers could be used as a temporary register if we have a leaf function (i.e. a function that calls no other functions?)
    A. rax
    B. rbx
    C. rcx
    D. rdx
6. Which registers could be used as a temporary register if we have a non-leaf function (i.e. a function that has calls to other functions?)
    A. rax
    B. rbx
    C. rcx
    D. rdx
7. Which of the following castings could lose information?
   (i.e. the original number can't be recovered)

```
int i= ...
unsigned int ui = ...
unsigned long ul = ...
float f = ...
double d = ...
```

    A. (unsigned int) i
    B. (int) ui
    C. (int) f
    D. (float) ui
    E. (double) ul
8. Which of the following number systems have a unique representation for all representable numbers?
    A. Unsigned
    B. Sign-magnitude (use most-significant digit just to represent the sign, rest of digits represent the number)
    C. Bias (e.g. where bias=$2^{k-1} - 1$)
    D. Two's Compliment
    E. IEEE Floating Point
9. Which of the following statements are true?
    A. In C, integer operations are associative, but floats are not
    B. In C, floats do not wrap around, unlike some integers
    C. In C, unsigned overflow is well-defined, but signed overflow is not

5	
6	
7	
8	
9	

(protip: if in doubt, just choose all the answers)

Question 7. Summon the Dragon! (6 pts)

After weeks of perilous descent into the Obsidian Dungeon, the adventurers have reached the final chamber — where the evil Archmage awaits. Their only hope is to channel their inner energy — their power level — to awaken the ancient guardian dragon bound within the ruins. Only the correct balance of energy will summon the guardian in time to defeat the Archmage and escape the dungeon alive.

```
invoke_spell:
        pushq   %rbx
        movl    %edi, %ebx
        cmpl    $5, %edi
        ja      .L4
        movl    %edi, %eax
        jmp     *.L6(,%rax,8)
.L6:
        .quad   .L4
        .quad   .L4
        .quad   .L8
        .quad   .L5
        .quad   .L4
        .quad   .L10
.L10:
        movl    $3, %ebx
.L5:
        subl    $1, %ebx
.L4:
        cmpl    $2, %ebx
        je      .L8
.L3:
        popq    %rbx
        ret
.L8:
        movl    $0, %eax
        call    summon_dragon
        jmp     .L3
```

The power level is the input argument to invoke_spell. For which values does the dragon awaken to help the heroes?

Question 8. Data structure Matching (Extra Credit, 5pts)

	Assembly

A	`movslq %edi, %rdi`
	`movq arr(,%rdi,8), %rax`
	`movslq %esi, %rsi`
	`movq (%rax,%rsi,8), %rax`
	`ret`

B	`movslq %edi, %rdi`
	`movslq %esi, %rax`
	`imulq $60, %rax`
	`movq arr(%rip), %rdx`
	`addq (%rdx,%rdi,8), %rax`
	`ret`

C	`movslq %esi, %rsi`
	`movslq %edi, %rdi`
	`leaq (%rdi,%rdi,2), %rax`
	`addq %rsi, %rax`
	`movl arr(,%rax,4), %eax`
	`ret`

D	`movslq %edi, %rdi`
	`movslq %esi, %rsi`
	`leaq (%rsi,%rsi,2), %rax`
	`salq $3, %rax`
	`addq arr(,%rdi,8), %rax`
	`ret`

E	`movslq %esi, %rsi`
	`movslq %edi, %rdi`
	`leaq (%rdi,%rdi,2), %rax`
	`addq %rsi, %rax`
	`movq arr(,%rax,8), %rax`
	`ret`

Code	**Which Assembly?**

`int *arr[5][3];`
`int* func(int x, int y) {`
`  return arr[x][y];`
`}`	

`int **arr[5][3];`
`int** func(int x, int y) {`
`  return arr[x][y];`
`}`	

`int *(*arr[5])[3];`
`int** func(int x, int y) {`
`  return arr[x][y];`
`}`	

`int (**arr[5])[3];`
`int (*func(int x, int y))[3] {`
`  return arr[x][y];`
`}`	

`int (**arr)[5][3];`
`int (*func(int x, int y))[3] {`
`  return arr[x][y];`
`}`	

Protip: If you answer C for all codes, you should get at least one right?

Question 9. No Cheat Codes, Chat GPT can't save you now! (Extra Credit, 5pts)

```
phase_k:
        cmpb    $0, 10(%rdi)
        jne     .L4
        movq    %rdi, %rax
        movzbl  (%rdi), %esi
        cmpb    $117, %sil
        jne     .L4
        movzbl  4(%rdi), %edx
        cmpb    $108, %dl
        jne     .L4
        movzbl  1(%rdi), %ecx
        cmpb    %cl, %sil
        jne     .L4
        movzbl  2(%rdi), %esi
        cmpb    3(%rdi), %sil
        jne     .L4
        movl    %edx, %edi
        andl    %ecx, %edi
        cmpb    %dil, %sil
        jne     .L4
        movzbl  5(%rax), %esi
        movl    %edx, %edi
        xorb    6(%rax), %dil
        movl    %esi, %r8d
        xorb    7(%rax), %r8b
        orb     %r8b, %dil
        jne     .L4
        movsbl  %dl, %edx
        addl    $6, %edx
        movsbl  %sil, %esi
        cmpl    %esi, %edx
        jne     .L4
        movsbl  %cl, %ecx
        addl    %ecx, %ecx
        andl    %ecx, %esi
        movsbl  8(%rax), %edx
        cmpl    %edx, %esi
        jne     .L4
        movsbl  9(%rax), %eax
        subl    $1, %edx
        cmpl    %edx, %eax
        jne     .L4
        ret
.L4:
        subq    $8, %rsp
        call    explode_bomb
```

What string will defuse this bomblab phase?

(we'll give you some partial if you get close…)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

This was the source code for bomb lab question:

```
void phase_k(char *input) {
    if (input[10] != '\0') { goto explode; }

    // check for uudd
    if (input[0] != 'u') { goto explode; }
    if (input[4] != 'l') { goto explode; }
    if(input[0] != input[1]) { goto explode; }
    if (input[2] ^ input[3]) { goto explode; }
    if( input[2] != (input[1] & input[4])) { goto explode; }

    // check for lrlr
    if ((input[4] ^ input[6]) |
        (input[5] ^ input[7])) { goto explode; }

    if (input[4]+'r'-'l' != input[5]) { goto explode; }

    // r AND (u << 1)  -> b
    if ((input[5] & (input[1] << 1)) != input[8]) { goto explode; }

    if (input[9] != input[8] - 1) { goto explode; }

    return;
    explode: explode_bomb();
}
```

The source code for the dungeon question:

```c
#include <stdio.h>

void summon_dragon() {
    printf("✨ magic happens!\n");
}

void invoke_spell(int level) {
    switch (level) {
        case 5: level = 3;
        case 3: level--; break;
        case 2: summon_dragon(); break;
        default: break;
    }
    if (level == 2) {
        summon_dragon();
    }
}
```