<u>Does anyone have any other practice exams from friends who've taken this class before? (please)</u>
/ "Pain is weakness leaving the body" -USMC
^imagine Schwarzenegger saying it
"If you're going through hell keep going" -Winston Churchill
"Carpe Diem" -- some professor
"Drink from the water fountains" -water fountains
"Don't drink from the water fountains" - anti water fountain gang aka city of Atlanta
"The only water fountain in the building has been down since I moved in." - a resident of GLC
"I can confirm the above statement is true." - another resident of GLC
"Gang gang gang gang." -- 21 Savage
"They did surgery on a grape." -- teh internet
"I came. I saw. I failed." --a student in CS2110
But u came tho
^ NO.
^ (͡° ͜ʖ ͡°) yeh
^ if lectures had attendance i'd get a 1%
I came, I saw, I yeeted.
And yet I left defeated :(

"What is wrong with this code:"

Ans: It's written on a piece of paper. It'll never run.
^LMAO
DO CIOS. SECTION A HAS ONLY 67 % NOW.
WITH 1 % additional point, it is same to get 4 % extra point on final exam.
]"Fill out the CIOS if you're in section A" - dying 2110 student : https://cios.zucc.io/ | http://gatech.smartevals.com

If you doubt/want to contend an answer, open a comment and/or ask our lovely TAs in follow-up discussions

1.

Expression	Resulting Type
a	pointer to char
***a**	char
a[0]	Char

c.c3	pointer to char
***f(1,2)**	pointer to int
f(3,4)	pointer to pointer to int
(*d).c3[1]	char
d->c2	float
g	pointer to array of pointer to float
***g**	array of pointer to float
h[3]	array 5 of array 3 of int. AKA array of array of int.
h[3][4][2]	int

2.

Expression	Value
a+1	0x20004
&a[0]	0x20000
a[0]	1
b	Address of ‘s’
b[2]	'r'
cx	0x2001C
&cx[2]	0x2003C
&cx[3].c2	0x20054
cx+2	0x2003C
cp+1	0x2002C
&cp[1]	0x2002C

3. Add pd pointer to table (&d)

	a	b	c	d	e	f	pa	pb	pc	pd	ppc	ppd	pf	ppf
	12	***9***	***3.5***	***-2.7***	***badly***	***w***	***&a***	***&b***	***&c***	***&d***	***&pc***	***&pd***	***&f***	***&pf***
***pa = 5**	5													
***pb = *pa+2**		7												
pa = pb							&b							
***pa= 1**		1												
pd = pc										&c				
****ppd = 4.0**			4.0											
****ppc = 1.0**			1.0											
****ppf = 'X'**						X								
pf = e													e	
(*ppf)[3] = 0					bad									

4.

```
struct mac {
        unsigned short macaddr[3];
};
int ctoh(char **buf) {
        int n = 0, hdigit;
        char *p;
        while (1) {
                if (**buf >= '0' && **buf <= '9') {
                        hdigit = **buf – '0';
                } else if (**buf >= 'a' && **buf <= 'f') {
                        hdigit = **buf – 'a' + 10;
                } else if (**buf >= 'A' && **buf <= 'F')
                        hdigit = **buf – 'A' + 10;
                } else {
                        break;
                }
                n = n * 16 + hdigit;
                (*buf)++;
        }
        return n;
}
int mactonum(char *in, struct mac *out) {
        //first get two character each hex digit
        unsigned short sum = 0;
        int count = 0;

        //for how many shorts  (TA Answer from 6 PM josh: // case: 08:00:20:f5:cc:a1)

        while (count != 3){
            //extract each short
                if (in[0] == ';') { return 0;}
                sum += ctoh(&in);       //sum = 8  08:00:20:f5:cc:a1 -> :00:20:f5:cc:a1
                sum = sum * 16 * 16;  //0800 add space for 00

                if (in[0] == ':') {
                        in++;
                        sum += ctoh(&in);   // x0800
                } else {
                        return 0; //malformed
                }
                out-> macaddr[count] = sum;
                sum = 0;
```

```
        In++;  // :20:f5:cc:a1 -> 20:f5:cc:a1
        count++;
        }
return 1; //i think after ur fin, u return 1

}
```

//TA Answer: Preston

```
int mactonum(char * in, struct mac * out) {
int array[6];
for (int i = 0; i < 6 && *in ! = '\0'; i ++) {
//loop through all 6 pairs of inputs, placing the integer value of each pair into array[]
        array[i] = ctoh(&in);
        if (i < 5 && *in != ":") {
//badly formed address if the character that in points to after it runs through ctoh is not ":"
                return 0;
        }
        in++;
}
out->macaddr[0] = (array[0] << 8) + array[1];
out->macaddr[1] = (array[2] << 8) + array[3];
out->macaddr[2] = (array[4] << 8) + array[5];
return 1;
}
```

5.

```
void *realloc(void *mem, size_t newsize);
int *getInts(size_t *n, size_t max) {
        // Declare any necessary variables here.
        int temp;
        int *retval = NULL;
        *n = 0;

        size_t buffCount = 0;
        size_t buffMax = 1; // start with random size
        size_t buffSize = buffMax * sizeof(int);
        Int scannedValue;
        retval = realloc(revtal, buffSize);

        If (retval==NULL){return NULL;}

        printf("Int> ");
        while((temp = scanf("%d", &scannedValue)) != EOF) {
                // Your code here
                // essentially you are give a number of int(unknown) occasionally realloc
                 // place code plz;
                If (buffCount + 1 > max){
                        printf("Buffer can't hold anymore!);
                        free(retval);
                        *n = max + 1;
                        return NULL;
                }

                if(buffcount+1> buffMax){
                        buffMax=buffMax*2;
                        buffSize = buffMax*(sizeof(int));
                        Int * new = realloc(retval, buffSize);
                        // why bother make a new placeholder? -So you can call free(retval) if it
                fails
                        if(new ==NULL){
                                free(retval);
                                *n = max + 1;
                                return NULL;
                        }
                        retval=new;
                }
                retval[buffCount++] = scannedValue;
                printf("Int> ");
```

```
        }

        *n = buffCount;

        // Don't forget n!!!

        return retval;
}
```

```
//a slightly more interesting solution
Int allocated_size = 0;

while(scanf("%d",&temp)  !!= EOF){
        *n ++;
        if(*n >. max){
                free(retval);
                *n=max+1;
                Return NULL;

        }
        if(allocated_size < *n){
                Retval = (int*)realloc(retval,2*(*n) * sizeof(int));
                Allocated_size = 2* (*n);
                if(retval == NULL){
                        free(retval);
                        *n=max+1;
                        Return NULL;
                }
        }
        retval[*n-1] = temp;


}

Return retval;
```

6.

a) **Problem:** *The problem is new string will only survive in the life cycle of the stack, it will be torn off the stack. Solution: store the new string in a heap.*

   Solution: *char * newString = (char*) malloc(sizeof(char)*1024);*

b) **Problem:** *pass by value, type mismatch*

   Solution: *pass in their address (int * x, int * y)*

c) **Problem:** *1. S is just a pointer that is not initialized, can not be access causes Segfault. 2. While loop is logically off*

   Solution: *1. Malloc for the char pointer. 2. && i < 1024 needs to be added to while loop*
   What does EOF mean again? End of File Lol

7.

Assembly Instruction	R1	R2	R3	0x30f4	0x3102
LEA R1, -3	0x30f4				
ADD R2, R1, 14		0x3102			
ST R2, -5				0x3102	
AND R2, R2, 0		0			
ADD R2, R2, 5		5			
STR R2, R1, 14					5
LDI R3, -9			5		

Helpful LC-3/Assembly Instruction Video:
https://www.youtube.com/watch?v=6hhNNj67w1E&t=1369s

8.

a) **NOT R3, R3**
ADD R3, R3, 1

b) **NOT R1, R1**
NOT R2, R2
AND R0, R1, R2
NOT R0, R0

c) **AND R0, R0, 0**
AND R1, R1, 1
BRP END
ADD R0, R0 1
END

Alternative solution:
AND R0, R0, 0
ADD R1, R1, 1
AND R0, R1, 1

Why not:
ADD R0, R1, 1
AND R0, R0, 1

What about:
AND R0, R1, 1
NOT R0, R0
(all of the bits to the left would be 1 instead of 0 so this is wrong)
Instead:
NOT R4, R1
AND R0, R4, 1

How about:
AND R0, R1, 1
(last bit = 1 if odd, 0 if even
R1 & 1 extracts the first bit (1 & 1 = 1, 0 & 1 = 0) and stores that in R0)

d) **ADD R3, R3, R3**
ADD R3, R3, R3

e) **AND R0, R1, R2**
NOT R0, R0

f) **ADD R0, R1, 0**
BRZP END
NOT R0, R0
ADD R0, R0, 1
END NOP

g) **NOT R1, R1**
ADD R1, R1, 1
ADD R0, R0, R1

9.

a) There are multiple SEXTs connected between IR and ADDR2MUX because there are three different types of offset bits pulled from the IR that will be doing different things based on different instructions. The offset bits are in increments of 6, 9, and 11, but ADDR2MUX takes in 16 bits so these offsets must be sign extended to hold the same value but be 16 bits. For example, the LEA instruction involves a PC offset expressed in 9 bits, so the SEXT [8:0] is needed to extend the offset to 16 bits, which can then be processed properly by the ADDR2MUX. The LDR instruction involves an offset expressed in 6 bits, so SEXT[5:0] is needed in this case.

b) There are 3 different offsets/addressing modes, offset 6 (Base register), 9 (PC-relative), and 11 (PC-Relative). They're connected to a MUX so it can choose the proper value and pass it along to whichever instruction is being carried out.

SOMEONE PLEASE ADD MORE. Not sure if these answers are fleshed out enough
This explanation is fine. But if you wanna still add more go ahead.

What's a good resource for more info about the LC-3 components?
I think the textbook is a good resource! Also lots of good youtube videos out there (https:// www.youtube.com/watch?v=6hhNNj67w1E) is good!!

10.

A	B	C	D	S
0	**0**	**0**	**0**	1
0	**0**	**0**	**1**	0
0	**0**	**1**	**0**	1
0	**0**	**1**	**1**	0
0	**1**	**0**	**0**	0
0	**1**	**0**	**1**	1
0	**1**	**1**	**0**	1
0	**1**	**1**	**1**	1
1	**0**	**0**	**0**	0
1	**0**	**0**	**1**	1
1	**0**	**1**	**0**	1
1	**0**	**1**	**1**	0
1	**1**	**0**	**0**	0
1	**1**	**0**	**1**	0
1	**1**	**1**	**0**	0
1	**1**	**1**	**1**	1

11.

	Statically Addressed	Stack	Heap	Visible to all	Visible within C file	Visible within function
Title	X			X		
error status	X			X		
stats	X				X	
a		X				X
b		X				X
result		X				X
count	X					X

help us out with this one!

Three types of storage duration: static, auto (on stack), dynamic (on heap)

- **Static objects last for the duration of the program**
- **Auto objects come and go with stacks**
- **Dynamic objects are created/destroyed at runtime (with malloc/free)**

Scopes/Visibilities: global (within program), static (within file), local (within block)

12.

Question	True/False?	Explanation
a: The BR and JMP LC-3 instructions use the same addressing modes	False	BR uses PCOffset to get an address, JMP uses a base register to get an address
b: The instruction TRAP x27 loads hexadecimal 27 into the PC	False	Loads the address at 0x27 in the trap vector table
c: Representing the number -81 in twos-complement requires at least 8 bits	True	10101111
d: An 8-bit twos-complement integer can represent numbers in the range -127 to +127	False	Two's complement goes from -2^(n - 1) to 2^(n - 1) - 1. In this case it would be from -128 to 127
e: A twos-complement integer can be multiplied by two by shifting it one bit to the left, filling the low-order bit with zero	True	Left shifting does this
f: Any Boolean function can be calculated by a proper combination of NOR gates	True	I'm actually not sure how to show this, Cliff says it's true though You can get NOT by putting the same input into a NOR (a NOR a = ~a). Once you have that: OR = ~NOR a AND b = ~a NOR ~b NAND = ~AND And you've got them all, anything else can be built from these
g: In C, computing (14 & 19) gives the result of 2	True	1. Convert each # to binary 2. AND them together (see which bits are the same) 3. Return the value of all the same bits
h: The hexadecimal number 3F07 can be represented in octal as	True	Convert 0x3F07 to binary, convert that back into octal

37407		
i: Sign-extending the 8-bit twos-complement integer 0xA7 to 32 bits yields 0xFFFFFFA6	False	The correct answer is 0xFFFFFFA7
j: The twos-complement of 0x0000 is 0xFFFF	False	2's complement of 0x0000: 1) convert 0x0000 to binary: (0000 0000 0000 0000) 2) flip all bits: (1111 1111 1111 1111) 3) add one: (0000 0000 0000 0000) 4) Convert back to hex: 0x0000 2's complement of 0x0000 is itself

Best of luck folks.

Section A, fill out the CIOS!

http://gatech.smartevals.com

this is a magic cat

```
  /\
  / \
.ʌ＿ʌ
( ･ω･。)つ━☆・*。
⊂　ノ　　・゜+.
しーＪ　　°。+ *
```

send this to ur 2110 friends and you will ace your final. if you don't you're fucked (just like the stack)
R6 pointer has been decremented 6261836183728

Luckily i dont have any 2110 friends so i will ace without sending

This is now a procrastination thread

yes haha

www.poptropica.com
Who knew google docs would be the new yik yak
I miss yik yak :(I was the muffin man guy