Wawrzynek & Weaver CS 61C Sp 2018 Great Ideas in Computer Architecture MT 1

PRINT your name: __________________________, __________________________
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct will be reported to the Center for Student Conduct, and may result in partial or complete loss of credit.

SIGN your name: ____

PRINT your class account login: `cs61c-`______ and SID: ________________________

Your TA's name: ____

Your section time: ____

Exam # for person sitting to your left: ______________ Exam # for person sitting to your right: ______________

You may consult one sheet of paper (double-sided) of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted.

You have 110 minutes. There are 6 questions, of varying credit (90 points total). The questions are of varying difficulty, so avoid spending too long on any one question. Parts of the exam will be graded automatically by scanning the **bubbles you fill in**, so please do your best to fill them in somewhat completely. Don't worry—if something goes wrong with the scanning, you'll have a chance to correct it during the regrade period.

If you have a question, raise your hand, and when an instructor motions to you, come to them to ask the question.

Do not turn this page until your instructor tells you to do so.

Question:	1	2	3	4	5	6	Total
Points:	10	20	10	20	15	15	90

Figure 1: This Space Deliberately Left Blank

Exam Clarifications

Please Sit Every Other Seat (at least)
Put away your phone/electronics
Now is 10:35
Question 1
(a)Write your 8 bit number in hex for your final answer
(b) x is equivalent to x in C (flip every bit).
Question 2
(a) The last } closes the function definition, not the while loop.
Question 3
Assume malloc always succeeds
(a) (iv) Should be &song2
Question 4
(a) write either character/ASCII code, memory location, OR value of the variable respectively
(c) By correct we mean no undefined behavior
Question 6:
#TODO: epilogue line and mv a0, s0 line should be swapped

Problem 1 *Number Representation* (10 points)

(a) Translate the following decimal numbers into 8-bit two's complement and unsigned binary representation in the table below. If a translation is not possible, please write "N/A". **Write your final answer in hexadecimal format.**

Solution:

Decimal Number	Two's Complement	Unsigned Number
10	0x0A	0x0A
129	None	0x81
-12	0xF4	None

(b) Suppose that we define the negative of x to be just $\overline{x}$. We will call this new number representation scheme **one's complement**. Note that the top bit of a one's complement number still denotes the number's sign (0 for positive, 1 for negative).

Translate the following decimal numbers into 8-bit one's complement binary representation. If the translation is not possible, please write "N/A". **Write your final answer in hexadecimal format.**

Solution:

Decimal Number	One's Complement
13	0x0D
-6	0xF9

(c) What is the range of integers (in decimal format) that we can represent with an n-bit one's complement binary number?

Solution: $[-2^{n-1}+1, 2^{n-1}-1]$

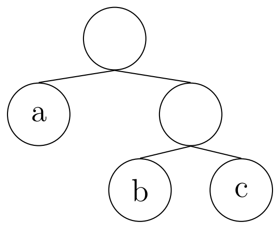
Problem 2 ***C Coding*** **(20 points)**

(a) **Find the End:** Given a **acyclic** singularly linked list, return a pointer to the last non-NULL node of the linked list. We have defined the `list_node` struct for the linked list below. Note that each node contains a string `str` as data. Return NULL if the list is empty. You may or may not need every blank but you may not add additional lines.

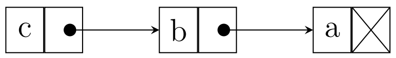
```
typedef struct list_node {
    char *str;
    struct list_node *next;
} list_node;
```

Solution:
```
list_node * find_end(list_node *start) {

    if (start == NULL) {

        return NULL;

    }

    while (start->next != NULL) {

        start = start->next;

    }

    return start;
}
```

(b) **Flatten:** Given a binary search tree (BST) **with data only stored in the leaf nodes**, generate a linked list with elements of the tree in reverse order (since this is a BST, the linked list will necessarily be in descending sorted order). The new linked list should have a **full copy** of each string in the BST. Return NULL if the BST is empty. For example, the following BST:



Should be turned into a linked list as such:



You are given a BST `tree_node` struct, the `list_node` struct and `find_end` (assume it's correctly implemented). You may use any C standard library functions and may assume that memory allocation always succeeds. Fill in the blanks to implement `flatten`. You may not need all the lines but you must not add additional lines.

```
typedef struct tree_node {
    char *str;
    struct tree_node *left, *right;
} tree_node;
char *strcpy(char *dest, const char *src);
list_node * find_end(struct list_node *start); // from part (a)
```

Solution:

```
list_node * flatten(tree_node *root) {
    if (root == NULL)
        return NULL;
    if (root->str != NULL) {
        list_node *curr = malloc(sizeof(list_node));
        curr->str = malloc(strlen(root->str)+1));
        strcpy(curr->str, root->str);
        return curr;
    }
    list_node *left_list = flatten(root->left);
    list_node *right_list = flatten(root->right);
    list_node *rend = find_end(right_list);
    if (rend == NULL) {
        return left_list;
    } else {
        rend->next = left_list;
        return right_list;
    }
}
```

Problem 3 ***C Analysis*** **(10 points)**

The CS61C Staff is creating songs in preparation of the grading party. Consider the following program:

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

typedef struct Song {
    char *title;
    char *artist;
} Song;

Song * createSong() {
    Song* song = (Song*) malloc(sizeof(Song));
    song->title = "this old dog";
    char artist[100] = "mac demarco";
    song->artist = artist;
    return song;
}


int main(int argc, char **argv) {
    Song *song1 = createSong();
    printf("%s\n", "Song written:");
    printf("%s\n", song1->title); // print statement #1
    printf("%s\n", song1->artist); // print statement #2

    Song song2;
    song2.title = malloc(sizeof(char)*100);
    strcpy(song2.title, song1->title);
    song2.artist = "MAC DEMARCO";
    printf("%s\n", "Song written:");
    printf("%s\n", song2.title); // print statement #3
    printf("%s\n", song2.artist); // print statement #4

    return 0;
}
```

(a) What type of address does each value evaluate to? Fill in the entire bubble.

i. `song1`

○ Stack address ○ Static address
[✓ Heap address] ○ Code address

ii. `song1->title`

○ Stack address [✓ Static address]
○ Heap address ○ Code address

iii. `song1->artist`

[✓ Stack address] ○ Static address
○ Heap address ○ Code address

iv. `&song2`

[✓ Stack address] ○ Static address
○ Heap address ○ Code address

v. `song2.title`

○ Stack address ○ Static address
[✓ Heap address] ○ Code address

(b) Will all of the print statements execute as expected?

○ Yes [✓ No]

If you answered yes, leave this blank. If you answered no, write the number(s) of the print statement(s) which will not execute as expected.

Solution: print statement #2

Problem 4 ***Sorting With Pointers*** **(20 points)**

You are given the following implementation of insertion sort for strings:

```
1 void string_insertion_sort(char *str) {
2     char tmp;
3     int inner, outer;
4     outer = 1;
5     while (str[outer] != '\0') {
6         inner = outer;
7         while (inner != 0 && str[inner] > str[inner - 1]) {
8             tmp = str[inner - 1];
9             str[inner - 1] = str[inner];
10            str[inner] = tmp;
11            inner--;
                     <------ breakpoint here
12        }
13        outer++;
14     }
15 }
```

It is called like so:

```
char str[4] = "bdf";
string_insertion_sort(str);
```

(a) **Trace Execution:** Suppose that I've set a breakpoint after line 11 (such that the breakpoint triggers after line 11 executes). Fill in the table for the memory contents at the breakpoint for the first 2 times this breakpoint triggers. The initial state refers to the contents of memory on line 2 (right when we enter the function). You may write either the character or the ASCII code, memory location, OR variable value respectively.

Solution:

Addr	C Variable	Initial State	1st Break	2nd Break
0xA	str[0]	'b' (98)	'd' (100)	'd' (100)
0xB	str[1]	'd' (100)	'b' (98)	'f' (102)
0xC	str[2]	'f' (102)	'f' (102)	'b' (98)
0xD	str[3]	'\0' (0)	'\0' (0)	'\0' (0)
0xE	str	0xA	0xA	0xA
0x11	tmp	uninitialized	'b' (98)	'b' (98)
0xF	inner	uninitialized	0	1
0x10	outer	uninitialized	1	2

(b) How many more times will the breakpoint trigger?

○ 0 [✓ 1] ○ 2 ○ 3 ○ 4 ○ 5

(c) Is this program correct for all null-terminated strings?

○ Yes [✓ No]

If you answered yes, leave this blank. If you answered no, provide a string that would serve as a counterexample.

Solution: '\0', as well as any statically defined string.

Suppose that we revise the above insertion sort algorithm to support a new character encoding that uses 16 bits (similar to Unicode). These characters are stored in a 16 bit unsigned integer type (called `uni_t`). The function is identical except that all variables are now of type `uni_t` instead of `char`. We will call this function `uni_string_insertion_sort`. We call this new sort in a similar fashion:

```
uni_t str[4] = "Ωβα";
uni_string_insertion_sort(str);
```

Suppose that the first character in `str` is stored at address `0xFFF8`. Answer the following questions assuming `uni_string_insertion_sort` is run from the start. If the answer cannot be determined or would cause an error from the provided information, write "Unknown".

(d) After line 4 has executed, what is the value of `str + outer`?

Solution: 0xFFFA (0xFFF8 + `sizeof(uni_t)*1`)

(e) After line 4 has executed, what is the value of `*(str + outer)`?

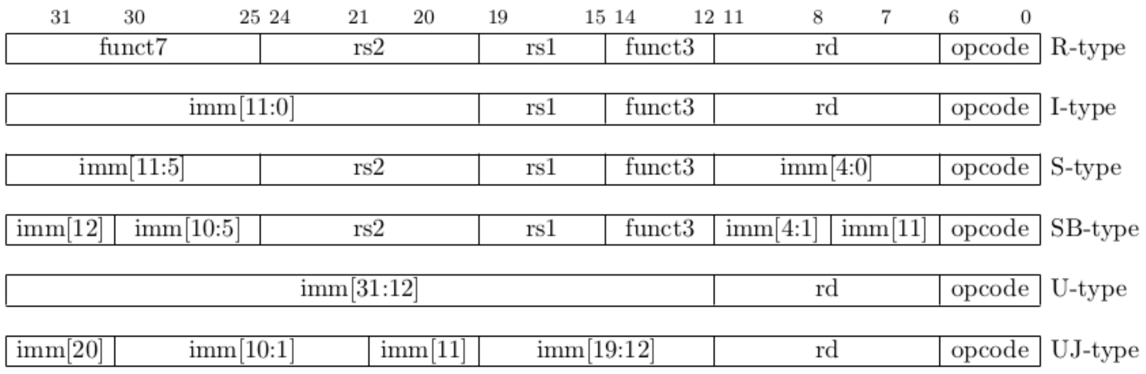
Solution: β

(f) After line 4 has executed, what is the value of `str[inner]`?

Solution: Unknown, `inner` is uninitialized.

Problem 5 ***RISC-U ISA*** **(15 points)**

Here are the standard 32-bit RISC-V instruction formats taught in lecture for your reference:


31 30 25 24 21 20 19 15 14 12 11 8 7 6 0
funct7 | rs2 | rs1 | funct3 | rd | opcode R-type
imm[11:0] | rs1 | funct3 | rd | opcode I-type
imm[11:5] | rs2 | rs1 | funct3 | imm[4:0] | opcode S-type
imm[12] | imm[10:5] | rs2 | rs1 | funct3 | imm[4:1] | imm[11] | opcode SB-type
imm[31:12] | rd | opcode U-type
imm[20] | imm[10:1] | imm[11] | imm[19:12] | rd | opcode UJ-type

Considering the standard 32-bit RISC-V instruction formats, convert `lw t5, 17(t6)` to machine code:

(a) **Solution:** 0x011FAF03

Prof. Wawrzynek decides to design a new ISA for his ternary neural network accelerator. He only needs to perform 7 different operations with his ISA: XOR, ADD, LD, SW, LUI, ADDI, and BLT. He decides that each instruction should be 17 bits wide, as he likes the number 17. There are no funct7 or funct3 fields in this new ISA.

(b) What is the minimum number of bits required for the opcode field?

Solution: $\lceil log_2 7 \rceil = 3$

(c) Suppose Prof. Wawrzynek decides to make the opcode field 6 bits. If we would like to support instructions with 3 register fields, what is the maximum number of registers we could address?

Solution: $\lfloor (17 - 6)/3 \rfloor = 3$ bits per register field which means 8 registers we could address

(d) Given that the opcode field is 6 bits wide and each register field is 2 bits wide in the 17 bit instruction, answer the following questions:

(i) Using the assumptions stated in the description of part (d), how many bits are left for the immediate field for the instruction BLT (Assume it takes opcode, rs1, rs2, and imm as inputs)?

Solution: 17-6-2-2 = 7

(ii) Let `n` be your answer in part (i). Suppose that BLT's branch immediate is in units of instructions (i.e. an immediate of value 1 means branching 1 instruction away). What is the maximum number of **bits** a BLT instruction can jump **forward** from the current PC using these assumptions? Write your answer in terms of `n`.

Solution: $(2^{n-1} - 1) * 17$

(iii) Using the assumptions stated in the description of part (d), what is the most negative immediate that could be used in the ADDI instruction (Assume it takes opcode, rs1, rd, and imm as inputs)?

Solution: -64

(iv) For LUI, we need opcode, rd, and imm as inputs. Using the assumptions stated in the description of part (d), how many bits can we use for the immediate value?

Solution: 17-6-2 = 9

Problem 6 ***RISC-V to C Magic*** **(15 points)**

Assume we have two arrays `input` and `result`. They are initialized as follows:

```
int *input = malloc(8*sizeof(int));
int *result = calloc(8, sizeof(int));
for (int i = 0; i < 8; i++) {
    input[i] = i;
}
```

You are given the following RISC-V code. Assume register `a0` holds the address of `input` and register `a2` holds the address of `result` when `MAGIC` is called by `main`.

```
main:
    ...
    # Start Calling MAGIC
    addi a1, x0, 8
    jal ra, MAGIC     # a0 holds input, a2 holds result
    # Checkpoint:  finished calling MAGIC
    ...
exit:
    addi a0, x0, 10
    add a1, x0, x0
    ecall     # Terminate ecall
MAGIC:
    # TODO: prologue.  What registers need to be stored onto the stack?
    mv s0, x0
    mv t0, x0
loop:
    beq t0, a1, done
    lw t1, 0(a0)
    add s0, s0, t1
    slli t2, t0, 2
    add t2, t2, a2
    sw s0, 0(t2)
    addi t0, t0, 1
    addi a0, a0, 4
    jal x0, loop
done:
    mv a0, s0
    # TODO: epilogue.  What registers need to be restored?
    jr ra
```

(a) Consider the function `MAGIC`. The prologue and epilogue for this function are missing. Which registers should be saved/restored in `MAGIC`'s prologue/epilogue? Select all that apply.

○ t0	○ a1
○ t1	○ a2
○ t2	○ ra
[✓ s0]	○ x0
○ a0	

(b) Assume you have the prologue and epilogue correctly coded. You set a breakpoint at "Checkpoint: finish calling `MAGIC`" and call `main`. What does `result` contain when your program pauses at the breakpoint? Please write the 8 numbers starting at `result` in the blanks below.

Solution: 0 1 3 6 10 15 21 28

(c) Translate `MAGIC` into C code. You may or may not need all of the lines provided below.

Solution:

```
// sizeof(int) == 4
int MAGIC(int *a, int b, int *c) {
    int sum = 0;
    for (int i = 0; i < b; i++) {
        sum += a[i];
        c[i] = sum;
    }
    return sum;
}
```