University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Summer 2009 Instructor: Jeremy Huddleston 2009-07-20

☹ CS61C MIDTERM ☺

After the exam, indicate on the line above where you fall in the emotion spectrum between "sad" & "smiley"...

Last Name	Standard
First Name	Grading
Student ID Number	
Login	cs61c-
Login First Letter (please circle)	a b c d e f
Login Second Letter (please circle)	a b c d e f g h i j k l m n o p q r s t u v w x y z
*The name of your **LAB** TA (please circle)*	James Josh Paul
Name of the person to your Left	
Name of the person to your Right	
*All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. **(please sign)***	

Instructions (Read Me!)

- This booklet contains 12 numbered pages including the cover page. Put all answers on these pages; don't hand in any stray pieces of paper.
- Please turn off all pagers, cell phones & beepers. Remove all hats & headphones. Place your backpacks, laptops and jackets at the front. Sit in *every other* seat. Nothing may be placed in the "no fly zone" spare seat/desk between students.
- You have 80 minutes to complete this exam. No computers, PDAs, calculators, or other electronics. You may reference two books and a collection of personal notes (within reason, we have the final say!)
- There may be partial credit for incomplete answers; write as much of the solution as you can. We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided. "IEC format" refers to the mebi, tebi, etc prefixes.
- If the question asks for a response as a numerical expression, you should express the answer in as reduced a form as possible (i.e. show what you would enter into a calculator if you had one available)
- Assume variable size and padding is consistent with the MIPS architecture we've been studying in class.
- If a sign is ever omitted, the value is assumed positive.

Question	**0**	**1**	**2**	**3**	**4**	**5**		**Total**
Minutes	1	14	10	20	20	15		80
Points	**2**	**25**	**19**	**39**	**39**	**26**		**150**
Score								

Name:______________________________ Login: cs61c- ________

Question 0: Hi, I'm your midterm. Who are you? (2 Pts, 1 Min)

Take a minute to write your name and login on every page of this exam. Make sure you are not missing any pages. 2 pts for Success

red: rubrick
green: answer.

Question 1: Just a lil bit… just a lil bit... (25 Pts, 14 Min)

2^n: 2 points
n^2, 2^n-1: 1 point
Other: 0

a) How many different things can we represent with N bits? 2^n

b) How many **more** bits (than N) would we need if we want to **double** the number of things we can represent? 1: 2 pts 1

c) How many **more** bits (than N) do we need if we want to instead **triple** the number of things we can represent? 2: 2 pts 2

d) Fill in the table below to show how the following bits can be interpreted in different ways. If a particular field has no solution, answer "N/A".

bit pattern	1111 1111	0010 0100	0110 0000	0000 0000
an unsigned hexaddecimal number	−½ minor arithmetic errors 0xFF246000			
an IEEE 32bit float (normalized binary scientific notation) $-1.0100100011 \times 2^{127}$	−½ for bad mantissa, sign, exponent, or saying $\times 10^{exp}$ instead of 2^{exp}			
4 ASCII characters	N/A	$	(	NULL \0
4 sign and magnitude bytes (in decimal)	-127	36	96	0
4 one's complement bytes (in decimal)	0	36	96	0
4 two's complement bytes (in decimal)	-1	36	96	0

(change of base was OK)

−½ minor arithmetic errors
−1 conceptual/major errors

1: lookup error: −½
2: lookup errors: −1.5
3: lookup errors: −2
3 lookup errors and 0 instead of NULL: −3
Blank: −4

special cases:
a) −4 if you complemented the n's complement positive numbers
b) −7 if you did the above w/ sign magnitude, too.

Name:______________________________ Login: cs61c- ________

Question 2: Where have all the cowboys gone? (19 Pts, 10 Min)

Clint and John are trying to organize a collection of all their old cowboy buddies. These 21st century cowboys are always on the road and don't like to be tied down, so there's no need to store street address, just cell numbers. Use the code they came up with (below) to answer the questions that follow.

This page is reproduced at the end of the test for you to tear off and reference!

```
#define NUM_FRIENDS     100
#define MAX_PHONE_LEN    13

struct entry {
    char   *name;
    char   phone_number[MAX_PHONE_LEN];
    int    favorite;
};
typedef struct entry entry_t;

unsigned int num_buddies = 0;
entry_t *buddies[NUM_FRIENDS];

void add_buddy(char *name, char *phone_number, int favorite) {
     int i = num_buddies++;
     static int initialized = 0;

     if(!initialized) {
          for(int j=0; j < NUM_FRIENDS; j++)
               buddies[j] = NULL;
          initialized = 1;
     }

     buddies[i] = (entry_t *)malloc(sizeof(entry_t));
     buddies[i]->name = name;
     strlcpy(buddies[i]->phone_number, phone_number, MAX_PHONE_LEN);
     buddies[i]->favorite = favorite;

     printf("%u\n", sizeof(name));
     printf("%u\n", sizeof(entry_t));
     printf("%u\n", sizeof(buddies));
     printf("%u\n", sizeof(*buddies));
     printf("%u\n", sizeof(buddies[i]->name));
     printf("%u\n", sizeof(buddies[i]->phone_number));

     printf("%u\n", *name);
     printf("%u\n", buddies[i]->name == name ? 1 : 2);
     printf("%u\n", buddies[i]->phone_number == phone_number ? 3 : 4);
     printf("%u\n", *buddies[i]->phone_number == *phone_number ? 5 : 6);
}

int main() {
     add_buddy("Dean", "5105551234", 1);
     ...
}
```

Name:________________________________ Login: cs61c- _________

a) Fill in the table below with the value printed as a result of each `printf` statement in the code. Assume `add_buddy` is being called for the first time as shown in `main()`.

Code	Printed Result
`printf("%u\n", sizeof(name));`	4
`printf("%u\n", sizeof(entry_t));`	24
`printf("%u\n", sizeof(buddies));`	400
`printf("%u\n", sizeof(*buddies));`	4
`printf("%u\n", sizeof(buddies[i]->name));`	4
`printf("%u\n", sizeof(buddies[i]->phone_number));`	13
`printf("%u\n", *name);`	68
`printf("%u\n", buddies[i]->name == name ? 1 : 2);`	1
`printf("%u\n", buddies[i]->phone_number == phone_number ? 3 : 4);`	4
`printf("%u\n", *buddies[i]->*phone_number == *phone_number ? 5 : 6);`	5

1 pt each

Name:______________________________ Login: cs61c- ________

b) Match the expressions on the left with their final location in memory on the right (assume that add_buddy has been called at least once as shown in main).

__E__ `struct entry`

__B__ `num_buddies`

__A__ `strlcpy`

__D__ `*buddies`

__C__ `name`

__D__ `buddies[0]->phone_number`

__B__ `initialized`

__A__ `add_buddy`

__B__ `buddies`

A. Code (text)
B. Static (global, data)
C. Stack
D. Heap
E. None / Other

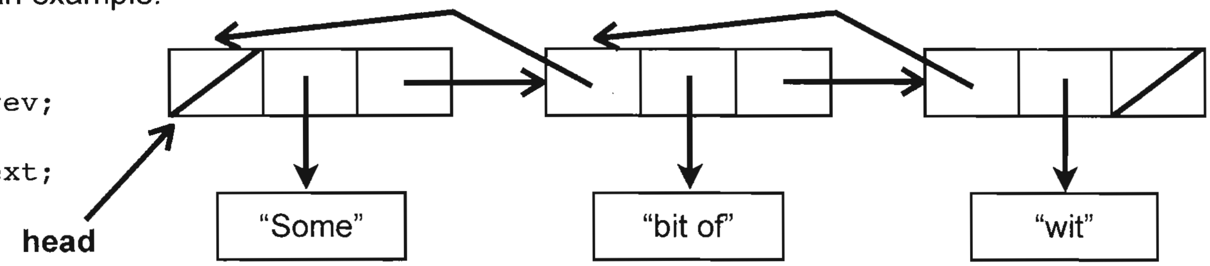
1 pt each

Name:________________________________ Login: cs61c- ________

Question 3: Insert wit here (39 Pts, 20 Min)

We have a simple doubly linked list that consists of witty comments (as C strings). The structure appears below with an example:

```
struct node {
  struct node* prev;
  char *payload;
  struct node* next;
};
```



The first node has its prev field set to NULL. The last node has its next field being set to NULL.

a) Fill in the code below to implement the `insert_after` function. This function inserts a node in the linked list after the `given_node` and **copies** the given string `data` into the `payload` field. The function returns a pointer to the `new_node` or `NULL` if the operation can not be completed. You can assume that the input is always valid.

```
struct node* insert_after(struct node* given_node, char * data) {
     // Allocate space
     struct node *new_node = (struct node*) malloc(sizeof(struct node)); +1
                                                              -1/2 if No struct
     if (new_node==NULL) +1 // Check for errors
          return NULL +1;
                                        not required
     // Setup the payload          +1                       +2           +2
     new_node->payload = (char *) malloc (sizeof(char)*(strlen(data)+1));
+2 for any malloc
     if (new_node->payload==NULL) {
               free (new_node); +1
               return NULL;     +1
     }
     strcpy(new_node->payload, data); +1

     // Setup Links
     new_node->next = given_node->next +1;     new_node->prev = given_node; +1
     if (new_node->next!=NULL or given_node->next!=NULL) +1
          given_node->next->prev or new_node->next->prev = new_node; +1
     +1 given_node->next = new_node;

     return new_node +1;
}
```

+19

~~20~~

Name:______________________________ Login: cs61c- ________

b) The `free_list` function takes a pointer to the head of a doubly linked list and frees all memory that was allocated for the list. You may assume it is correctly given the head of the doubly linked list.

```
void free_list(struct node* head) {
     free(head->payload);
     free(head);
     free_list(head->next);
}
```

Describe all the bugs in the given `free_list` function. Please enumerate ("number") your answer.

+5 1) Calls free(head) before free_list(head->next). can not dereference head after freeing.

+5 2) No base case on recursion. will free a null pointer eventually.

-2 for hand wavy solutions.
-1 or -2 for answers that were wrong, but would break things.

Implement a version of `free_list` which corrects all the bugs.

```
void free_list(struct node* head) {
    if (head == NULL) return;
    free(head->payload);
    free_list(head->next);
    free(head);
```

or

```
    free(head->payload);
    if (head->next != NULL)
        free_list(head->next);
    free(head);
```

+2 free_list() before free() correct
+2 free payload before free(the head) correct
+2 free head
+2 free payload
+2 handle base case
various deductions for incorrect behavior.

These were the two most common solutions, others were accepted.

```
}
```

Name:______________________________ Login: cs61c- ________

Question 4: Assemble this! (39 Pts, 20 Min)

a) Below is a section of MAL code. Convert this code into its TAL equivalent using the table below and label the address in memory where the TAL instructions start. The address for `start` is 0x1020.

```
start:      subiu $a0, $v0, 0xDEADBEEF
            move $a1, $s0
            jal check_value
            beqz $v0, start             # branch if $v0 is zero
check_value: ...
```

Start Address	MAL	TAL	
0x1020	subiu $a0, $v0, 0xDEADBEEF	lui $at 0x DEAD ori $at $at 0xBEEF subu $a0 $v0 $at	6pts
0x102c	move $a1, $s0	or $a1 $0 $s0	3pts
0x1030	jal check_value	jal check_value	3 pts
0x1034	beqz $v0, start	beq $v0 $0 start	3pts

b) What toolchain stage (compiler, assembler, linker, loader) would be responsible for **finishing** the conversion from MAL to TAL for each of the 4 instructions?

___A___ `subiu $a0, $v0, 0xDEADBEEF` ___Li___ `jal check_value` 2 pts

___A___ `move $a1, $s0` ___A___ `beqz $v0, start`

4 pts

c) Convert the following TAL instructions into their machine level representation in hex. Show your work for full credit.

i) `lw $t0 4($sp)` Hex: 0x 8FA80004

1000 11|11 101|0 1000 | 0000 0000 0000 0100

ii) `add~~i~~u $s0 $t1 $a0` Hex: 0x 01248021

0000 00|01 001|0 0100| 1000 0|000 00|10 0001

iii) `beq $t0 $t1 -4` Hex: 0x 1109FFFC

0001 00|01 000|0 1001| 1111 1111 1111 1100

5 pts each

Name:______________________________ Login: cs61c- ________

Question 6: Gi'me, gi'me more, gi'me more, gi'me, ... (26 Pts, 15 Min)

Amy recognized that rounding errors occurred when adding together two of her floating point numbers. Bob, one of her friends, suggested that she could increase the precision and express more numbers by increasing the size of the significand/mantissa field.

Amy decided to change the floating point spec to see if this would work. She chose to move two bits from the exponent field to the significand field:

1	6	25
S	EEEEEE	MMMMMMMMMMMMMMMMMMMMMMMMM

Answer the following questions below using these modified field sizes. Everything else follows the normal IEEE floating point standard. Where you may have a choice in algorithm, choose the *default* mode for IEEE floating point. If a question asks for a floating point value, **write it in normalized binary scientific notation.**

a) What is the exponent bias for Amy's new configuration? (3 pts)

Exponent bias $= 2^{\#\text{exp bits}-1} - 1 = 2^{6-1} - 1 = 2^{5} - 1 = 31$

-1 Math Errors / off By 1 Errors

-3 Solution that doesn't make any sense

b) What is the smallest positive number that Amy can accurately represent in **normalized** form? (5 pts)

S EEEEEE M . . . . . M

0 000001 0 0

$(-1)^0 \times 2^{1-\text{bias}} \times 1.0 = (-1)^0 \times 2^{1-31} \times 1.0$

$= 2^{-30}$

-1 off By 1 Errors

-5 bad solution (doesn't know difference between normalised & denormalized + other stuff)

Name:______________________________ Login: cs61c- ________

c) What is the implicit exponent for **denormalized** numbers in Amy's new configuration? (3 pts)

$-(\text{bias}-1) = 1-\text{bias} = 1-31 = -30$

-1 Off By 1 Errors

-3 Bad Solution (positive implicit exp
other nonsense)

d) What is the smallest positive number that Amy can accurately represent? (5 pts)

S EEEEEE M - - - - - M

0 000000 0.0 - - - - - 1

$\Downarrow$ denorm $\Downarrow$

$(-1)^0 \times 2^{-30} \times 2^{-25} = 2^{-55}$

-3 missing either 2^{-30} or 2^{-25}

-5 Bad Solution

e) What is the smallest positive number (in Amy's scheme) to which you can add the number 5.0 and have the result be unchanged? (ie: what is the smallest x such that x + 5.0 = x) (10 pts)

$5.0 = 101_2$

X = 1 0000 0000 0000 0000 0000 0000 0000
101

1 0000 0000 0000 0000 0000 0000 0|101

these bits get lost, since there are only 25 bits for the significand

$\boxed{X = 1.0 \times 2^{28}}$

-1 off by 1, with sufficient work

-2–8 other issues

-9 enough work to warrant a pity point

-10 blank / nonsense

Name:______________________________ Login: cs61c- ________

Reference: Where have all the cowboys gone?

Clint and John are trying to organize a collection of all their old cowboy buddies. These 21st century cowboys are always on the road and don't like to be tied down, so there's no need to store street address, just cell numbers. Use the code they came up with (below) to answer the questions that follow.

```
#define NUM_FRIENDS    100
#define MAX_PHONE_LEN   13

struct entry {
    char   *name;
    char   phone_number[MAX_PHONE_LEN];
    int    favorite;
};
typedef struct entry entry_t;

unsigned int num_buddies = 0;
entry_t *buddies[NUM_FRIENDS];

void add_buddy(char *name, char *phone_number, int favorite) {
     int i = num_buddies++;
     static int initialized = 0;

     if(!initialized) {
          for(int j=0; j < NUM_FRIENDS; j++)
               buddies[j] = NULL;
          initialized = 1;
     }

     buddies[i] = (entry_t *)malloc(sizeof(entry_t));
     buddies[i]->name = name;
     strlcpy(buddies[i]->phone_number, phone_number, MAX_PHONE_LEN);
     buddies[i]->favorite = favorite;

     printf("%u\n", sizeof(name));
     printf("%u\n", sizeof(entry_t));
     printf("%u\n", sizeof(buddies));
     printf("%u\n", sizeof(*buddies));
     printf("%u\n", sizeof(buddies[i]->name));
     printf("%u\n", sizeof(buddies[i]->phone_number));

     printf("%u\n", *name);
     printf("%u\n", buddies[i]->name == name ? 1 : 2);
     printf("%u\n", buddies[i]->phone_number == phone_number ? 3 : 4);
     printf("%u\n", *buddies[i]->phone_number == *phone_number ? 5 : 6);
}

int main() {
     add_buddy("Dean", "5105551234", 1);
     ...
}
```