CS61C, Fall 2000
Midterm 1
Professor D. Patterson

Question #0: Sign-in Question (-1 point if not followed): *Fill out the front page correctly and write your login at the top of each of the pages.* **Circle your login initials so we can read them.**

Question #1: I am a computer (Parts of a Computer) (3 points)

(a) Name the five components of a computer:	**(b)** Consider your body, and how it is like a computer system. For each component you listed to the left, give one example of a body part that corresponds to that component. You may not use the same body part more than twice.
1.	**1.**
2.	**2.**
3.	**3.**
4.	**4.**
5.	**1.**

Question #2: Minding your p's and q's (Pointers) (5 points)
Below is a segment of C code:

```
char *p, *q, x = 'a', y = 'b';

p = &x;
q = &y;
*q = *p++;
q = p;
```

To answer the following questions, you may use any `$t` register. Assume the address of `x` is 1000, address of `y` is 1004, `p` is associated with register `$s0, q` with `$s1`. Please keep your code brief.

i. **[1 point]** What is the MAL code for `p = &x;`

ii. **[1 point]** What is the MAL code for `q = &y;`

iii. **[1 point]** What is the MAL code for `*q = *p++;`

iv. **[1 point]** What is the MAL code for `q = p;`

v. **[1 point]** What is the value of `q` at then end of this section of code?

Question #3: My numbers are changing! (Pilable Data) (5 points)

A = 0x20C89000

(a) [1 point] Give the binary equivalent of A:

_____

For the following 3 questions (b, c, d) you may leave parts of your answers as signed sums of powers of two [e.g. -(2^2+...2^19)]. Do **not** leave any part of your answers in hex. **No hex!**

(b) [1 point] Give the decimal equivalent of A if A is interpreted as an integer in 2's-complement notation:

_____

(c) [1 point] Give the decimal equivalent of A if A is interpreted as a single-precision IEEE 754 floating point number:

_____

(d) [2 points] Give the MAL instruction equivalent of A if A is interpreted as an instruction in machine language format:

_____

Remember, **no hex!**

Question #4: Can a C integer float? (Floating Point) (2 points)

(a) Can all *int* in C (for the MIPS Instruction Set) be converted to a **single precision** floating point number and back to integer without changing the value of the integer? Why or why not?

(b) Can an *int* in C (for the MIPS Instruction Set) be converted to a **double precision** floating point

number and back to integer without changing the value of the integer? Why or why not?

Question #5: Is the Internet really this fast? (Networks) (10 points)

You are writing a program to send an array X from machine A to machine B over a network. Let us assume that ints are 32 bits and chars are 8 bits, and the operating system on computer A implements a function

```
void send(char *machineName, char *sendbuf, int nbytes);
```

that sends to the machine indicated by the first argument `(machineName)` `nbytes` of data starting at the byte pointed to by `sendbuf`.

You are considering two ways to implement the program:

```
/* First way */
int main(int argc, char *argv[]) {
  int X[500,000];
  int I, *Y = X;
  for (I=0; I<500; I++, Y+=1000)
    send("B", (void *)Y, 4000);
  exit(0);
}

/* Second way */
  int X[500000];
  send("B", (void *)X, 2000000);
  exit(0);
}
```

The questions are on the next page.

You do not have to show your work, but if your answer is incorrect, showing your work may help us to determine what you understand.

Assume that the bandwidth of the network link between machine A and machine B is 0.5 million bytes per second (**not** 524288 bytes per second), and what each call to `send()` incurs an overhead of 1ms per call plus 10 microseconds per byte to copy the data to the network interface.
Assume that the `send()` overhead described above cannot be overlapped with the actual sending of the data over the network, i.e. that a call to `send()` does not return until the data has been copied to the network interface and then sent out (but not necessarily received on the other end) over the network by

the network interface.

[2 points] (a) How long does it take to send the X array from machine A to machine B using the "first way" piece of code above? **Circle or box your final answer.**

[2 points] (b) How long does it take to send the X array from machine A to machine B using the "second way" piece of code above? **Circle or box your final answer.**

Now assume that the call to `send()` operates exactly as above (copying data to the network interface, the network interface sending the data out, and `send()` then returning) but the call to `send()` does not return until the network interface of machine A receives an acknowledgment mesage from B letting A know that B has received the data. Assume that the acknowledgment message is 10000 bytes in size, that the acknowledgment message is sent directly by the network interface on B without involving B's CPU or memory, and that it takes the network interface on B 1ms to receive a message from A and prepare the acknowledgment message before sending it.

[3 points] (a) How long does it take to send the X array from machine A to machine B using the "first way" piece of code above? **Circle or box your final answer.**

[3 points] (a) How long does it take to send the X array from machine A to machine B using the "second way" piece of code above? **Circle or box your final answer.**

Question #6: M'Piero behind the scenes of gmake (Starting a Program) (2 points)

Here are five events, A-E that occur while M'Piero is working on proj1. But, they are out of order! You must put them in the correct order and **write your answers in the boxes below.**
Assume that M'Piero's MIPS system uses the classic four discrete steps for transforming a C program into a program running on a computer; M'Piero's system does not combine steps.

A)
```
addiu $a0, $a0, 0x456789ab
is converted into lui $at, 0x4567
ori $at, 0x89ab
addu $a0, $a0, $at
```
(The second line uses an ori, not an addiu, because addiu sign-extends, but ori does not.)

B) The executable file, philspel, is written.

C) hastable.o and philspel.o are produced independently, each with a symbol table and relocation information. As a result, future editing of philspel.c will not require recompilation of hashtable.c.

D) References from philspel.o to the function findData are patched. The jump target corresponding to readDictionary is changed as well.

E) M'Piero writes proj1 in C. M'Piero uses C, instead of assembly, because M'Piero knows that optimizing compilers can produce assembly language programs almost as well as human experts, and because memory is cheaper these days.

The order of events is **(write your letters in the boxes)**:

_

1	2	3	4	5
_	_	_	_	_

Question #7: If only I could write in C? (MIPS) (9 points)

Executing this program in spim, will it reach the *next* label? If not, why not, and if so, what will the value of *$t1* be (in hex) and why? Assume all registers are initially zero. This is a correct translation from MAL to TAL; both versions are there only to assist you with the problem.

(a) [3 points]

```
Address          TAL                 MAL
[0x00400000] lui    $1, 64       __start:  la    $s0, loop
[0x00400004] ori    $16, $1, 12
[0x00400008] lui    $8, 1                  addiu $t0, $0, 0x10000
[0x0040000c] bne    $0, $11, 12  loop:     bne   $0, $t3, next
[0x00400010] lw     $10, 0($16)            lw    $t2, 0($s0)
[0x00400014] addu   $9, $9, $8             addu  $t1, $t1, $t0
[0x00400018] jr     $16                    jr    $s0
[0x0040001c] ori    $2, $0, 10   next:     done
[0x00400020] syscall
```

$t1 = 0x_____

Why/Why not? _____

We've added a couple of instructions. Now, if the program reaches the *next* label, what will the value of *$t1* be (in hex) and why? If not, why not?

(b)[6 points]

Address	TAL		MAL		
[0x00400000]	lui	$1, 64	__start:	la	$s0, loop
[0x00400004]	ori	$16, $1, 12			
[0x00400008]	lui	$8, 1		addiu	$t0, $0, 0x10000
[0x0040000c]	bne	$0, $11, 20	loop:	bne	$0, $t3, next
[0x00400010]	lw	$10, 0($16)		lw	$t2, 0($s0)
[0x00400014]	addu	$10, $10, $8		addu	$t2, $t2, $t0
[0x00400018]	sw	$10, 0($16)		sw	$t2, 0($s0)
[0x0040001c]	addu	$9, $9, $8		addu	$t1, $t1, $t0
[0x00400020]	jr	$16		jr	$s0
[0x00400024]	ori	$2, $0, 10	next:	done	
[0x00400028]	syscall				

$t1 = 0x_____

Why/Why not? _____

Question #8: Can you help me find my integer? (Pointers in C and MIPS) (14 points)

In your first project, philspel, we provided you with a typical hash table implementaion. A similar, albeit simplified, hashtable is defined below. **Note that the keys and values stored in this hashtable are simply integers:**

```
/* hashBuckets store the individual entries of the hash table.
 * Since it is possible for two different keys to hash to the
 * same value, the buckets can be chained via the next pointer
 * they contain. The next pointer of the final bucket in a
 * chain has value NULL, and, in this implementation, you may
 * assume that NULL has value 0
 */
typedef struct HashBucket {
  int key;
  int value;
  struct HashBucket *next;
} HashBucket;
```

```
/* The hashtable itself is an array of pointers to HashBuckets.
 * TABLE_SIZE is an arbitrary constant. hashTable is passed in
 * to findData
 */
HashBucket *hashTable[TABLE_SIZE];

/* The findData method looks up the given key in the given hash
 * table, and returns the value associated with that key
 */
int findData(int key, HashBucket **hashTable) {
  unsigned int location = hashFunction(key) % TABLE_SIZE;
  HashBucket *lookAtMe = hashTable[location];
  while (lookAtMe != NULL) {
    if (lookAtMe->key == key) {
      return lookAtMe->value;
    }
    lookAtMe = lookAtMe->next;
  }
  return ERROR_NOT_FOUND;
}
```

A possible implementation of findData in MIPS is on the following page. Your task is to fill in the missing instructions. You should also add a relevant comment for each instruction you fill in. **You do not have to follow the given MIPS code, you can write your own function from scratch, however you are more likely to make mistakes by not following the framework code. Use the last page if writing your own findData function.**

You will likely find the following information, regarding the layout of C structs in memory, useful: The members of a structure are laid out in memory one after another in the order of their declarations. The address of a structure is the address of its first member. You may assume that there are no gaps between members. (Actually, we've simplified things a bit for this question. After the exam, refer to K&R for details (not now!).)

Hint: Your MIPS solution should closely follow the C solution above.

```
findData:

 addiu $sp, $sp, -28
 sw    $a0, 16($sp)
 sw    $a1, 20($sp)
 sw    $ra, 24($sp)
```

```
jal     hashFunction
lw      $a0, 16($sp)
lw      $a1, 20($sp)
lw      $ra, 24($sp)
addiu $t0, $0, TABLE_SIZE # Pretend TABLE_SIZE is a constant
divu   $v0, $t0
mfhi   $t0                      # $t0 = hashFunction(key) % TABLE_SIZE

_____________________________  #_____________________________

_____________________________  #_____________________________

_____________________________  #_____________________________

_____________________________  #_____________________________

loop:

_____________________________  #_____________________________

_____________________________  #_____________________________

 bne $t1, $a0, next_bucket # if (lookAtMe->key == key). . .
                           # The above comment is somewhat
                           # misleading
                           # if (lookAtMe->key != key)
                           # might be more accurate

_____________________________  #_____________________________

j fin

next_bucket:

_____________________________  #_____________________________

j loop
```

```
not_found:

 addiu $v0, $v0, ERROR_NOT_FOUND # Pretend ERROR_NOT_FOUND is a
                                 # constant

fin:
addiu $sp, $sp, 28
jr $ra
```
