Name: ____

CSE 378 Winter 2009 Final Exam
Machine Organization & Assembly Language

Write your answers on these pages. Additional pages may be attached (with staple) if necessary. Please ensure that your answers are legible. Please show your work. Write your name at the top of each page.

Problem	Points
1	/ 20
2	/ 10
3	/ 15
4	/ 5
5	/ 5
6	/ 10
7	/ 10
8	/ 15
9	/ 10
TOTAL	/ 100

1. [20 points] **MIPS Assembly Programming**

   Write an assembly function `ITOA` that takes a 32-bit integer stored in `$a0` and translates it into a null-terminated decimal string of ASCII characters stored at the memory location pointed at by `$a1`. Some of the code has already been provided, including function `REVERSE_STRING`, which reverses a null-terminated string in memory.

   You may assume that all integers passed to your function will be greater than or equal to 1.

   HINT: Use the `div` and `mfhi` instructions.

```
ITOA:
    subu $sp, $sp, 16       # allocate stack frame
    sw $ra, 0($sp)          # save return address

    # constants you may find useful
    addi $v0, $0, 48   # ASCII character code for '0'
    addi $v1, $0, 10

    add  $t0, $0, $a0  # move integer into temporary register
LOOP:









DONE:
    add $a0, $0, $a1        # argument for string_reverse
    jal REVERSE_STRING
    lw $ra, 0($sp)          # restore return address
    addiu $sp, $sp, 16      # pop stack
    jr $ra
```

2. [10 points] **Pointers**.

   Are the following two pieces of code the same?

```
int x;
while(a[i] != 0) {
x = a[i];
  i++;
}

...

int x;
while(*a != 0) {
  x = *a;
  a++;
}
```

   If so, why? If not, why not?

3. [15 points] **Performance**

Instruction Type	CPI
load/store	2.5
ALU	1
branch	2

Application	IPC	% Loads/Stores	% ALU ops	% Branches
A	.625			
B	.8			

(a) Given the IPC for each application *A* and *B*, and the CPI costs for each instruction type (in the tables above), give a percentage breakdown for each of the instruction types that constitute that application (Note: there are multiple correct answers). Ensure that your percentages sum to 100% for each application. Hint: CPI $= \frac{1}{\text{IPC}}$.

(b) Supposing that the cost of ALU operations is now completely free (i.e., CPI = 0 for ALU ops), what is the speedup for each application? Feel free to leave your answers as fractions.

4. [5 points] **Interrupts**

   (a) Why do we use interrupts?

   (b) What is the difference between interrupts and exceptions?

   (c) What happens when an interrupt occurs?

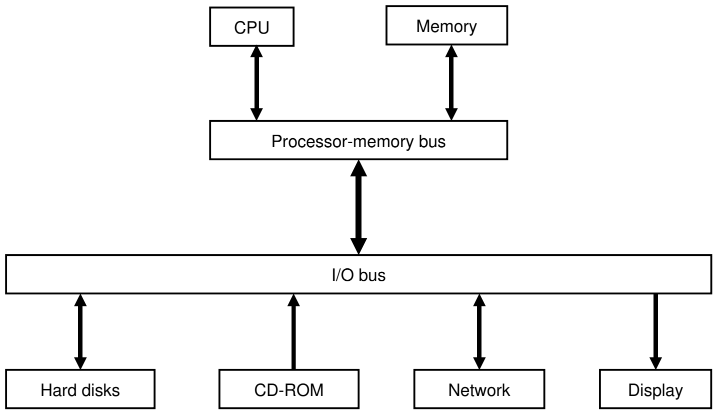
5. [5 points] **Code Optimization**

   Which program's code is faster? Why?

```
// Program A
for ( j = 0; j < 20; j++)
  for ( i = 0; i < 200; i++)
    x[i][j] = x[i][j] + 1;
```

```
// Program B
for ( i = 0; i < 200; i++)
  for ( j = 0; j < 20; j++)
    x[i][j] = x[i][j] + 1;
```

6. [10 points] **I/O**



(a) Explain why the buses in the figure above are split into a hierarchy.

(b) What is direct memory access? What is its benefit?

(c) A CPU and memory share a 64-bit bus running at 500MHz. The memory needs 40ns to access a 256-bit value from one address. What is the effective bandwidth?

7. [10 points] **Caching**

   The following C Program is run (with no optimization) on a processor with a cache that has 8-word (32-byte) blocks, and the cache holds a total of 256 bytes of data. A C `int` is 1 word in size. You **must** give your answers *as fractions*, but you can leave them unreduced.

```
int  i, j, c, stepsize, array[512];

for ( i = 0 ;  i < 100 ;  i++) {
  for ( j = 0 ;  j < 512 ; j=j + stepsize ) {
    c += array [j] + 17;
  }
}
```

   (a) If we consider only the cache activity generated by references to the array, what is the miss rate when the cache is direct mapped and `stepsize = 256`?

   (b) Again considering only the cache activity generated by references to the array, what is the miss rate when the cache is direct mapped but we change `stepsize` to `255`?

   (c) Again considering only the cache activity generated by references to the array, what is the miss rate when the cache is *two-way set associative* and `stepsize = 256`?

8. [15 points] **Mystery Cache**

   Given the following sequence of memory requests and their responses from a mystery cache M, give a possible block size (in bytes), associativity, and total size (in bytes) for M, as well as whether M is write allocate or write no-allocate. All addresses are *byte* addresses.

R/W	Address	Outcome
W	0	miss
R	1	hit
R	4	miss
R	2	miss
W	4	hit
R	3	hit
R	0	miss

9. [10 points] **Virtual Memory**

   (a) Assume a hierarchical page table of two levels. Pages in this system are 4KB in size, and page table entries are 4B each. Assume there is exactly one 2nd-level page table *P* in the system, and *P* occupies exactly one page of physical memory. If exactly half of *P*'s entries are valid, how many bytes of memory in our virtual address space actually reside in the physical memory? Do *not* include the space occupied by the page tables themselves in your answer.

   (b) What is a Translation Lookaside Buffer?