CSE 410 Midterm Exam 4/27/07

Name ________________________________

Do **not** write your id number or any other confidential information on this page.

There are 8 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

You will want to use a copy of the “green card” from the textbook. We have additional copies if you do not have one. Other than that, the exam is closed book, closed notes, etc.

Please wait to turn the page until everyone is told to begin.

Score ___________________ / 100

1.	_______ / 6
2.	_______ / 4
3.	_______ / 10
4.	_______ / 10
5.	_______ / 16
6.	_______ / 20
7.	_______ / 20
8.	_______ / 14

Question 1. (6 points) Suppose the 32-bit quantity `0x12345678` is stored at address 1000.

(a) If we interpret these bits as a MIPS machine instruction, what is it? (Use symbolic forms for the opcode and register names, but if the instruction contains a 16-bit immediate or displacement field, you can leave that as a hex number and don't need to convert it to decimal.)

(b) If this 32-bit word is stored at address 1000 on a big-endian machine, what is the value of the byte stored at address 1000?

Question 2. (4 points) If we interpret the 32-bit quantity `0x446f6821` as a sequence of ASCII characters, what are they?

Question 3. (10 points) Convert the following decimal numbers to 16-bit 2’s complement numbers. Write the results in both binary and hexadecimal notation.

(a) 410

(b) -90

Question 4. (10 points) For each of the following 16-bit two’s complement hexadecimal numbers, (i) write the value as a binary number, and (ii) convert the value to a decimal integer and give its value.

(a) 0x030F

(b) 0xFEED

Question 5. (16 points) Translate each of the following MIPS assembly instructions to machine code. Show the results as both binary and hexadecimal 32-bit numbers.

If the instruction is a MIPS assembly-language pseudo-instruction, first show the actual MIPS instructions that implement it, then translate those instructions to binary and hexadecimal.

(a) `add  $v0,$a2,$t4`

(b) `addi $s1,-3`

(c) `move $s3,$ra`

(d) `sw   $t3,32($a1)`

Question 6. (20 points) Consider the following fragment of MIPS code.

```
loop:
     lw     $t2,0($t0)
     add    $t3,$t3,$t2
     sw     $t3,0($t0)
     addi   $t0,4
     addi   $t1,-1
here:
     bne    $t1,$zero,loop
done:
```

Now assume that registers and memory are initialized as shown in the "initial" column of the following tables (all values are decimal numbers, not hex).

Register	Initial	1st "here"	2nd "here"	etc.		
$t0	1000					
$t1	3					
$t2	-1					
$t3	0					

Memory	Initial	1st "here"	2nd "here"	etc.		
1000	2					
1004	4					
1008	6					
1012	8					

(a) Trace the execution of the MIPS code above. Write down the contents of the listed registers and memory locations each time execution reaches the label "`here:`" in the program. If you need an extra column, go ahead and add it.

(b) Below, describe what the program fragment does in a short sentence or two.

Question 7. (20 points) Write a function in MIPS assembly language to locate and return the largest value in an integer array. In Java the function would be specified as follows (C, C++, and other languages are similar):

```
/* Given an array A containing n integers, return */
/* the largest integer found in A[0]...A[n-1]     */
int findmax(int[] A, int n) { ... }
```

Your function should use the standard MIPS calling and register conventions. For example, the following code could be used to call the function to find the largest integer in a 10-element array starting at location `nums`:

```
la      $a0,nums        # $a0 = array address
li      $a1,10          # $a1 = number of array elements
jal     findmax         # call function findmax
```

Write your code on the next page. You may assume that the array has at least 1 element (i.e., the number of elements in `$a1` will be greater than 0).

Suggestion: Use the rest of this page to sketch out your solution, perhaps by writing fairly low-level Java, C, or other code. Once you've organized your thoughts, write your answer on the next page.

(continue on the next page)

Question 7. (cont.) Write your answer below.

Question 8. (14 points) The following MIPS code could be used to increment the value of two integer variables (similar to "`i++; j++`" in Java).

```
li   $t0,1            # load 1 into $t0
lw   $t1,100($t5)     # load first variable
add  $t1,$t0,$t1      # increment
sw   $t1,100($t5)     # store
lw   $t2,104($t5)     # load 2nd variable
add  $t2,$t0,$t2      # increment
sw   $t2,104($t5)     # store
```

(a) Draw a diagram showing how these instructions are executed in the standard 5-stage MIPS pipeline. You should assume that words loaded from memory are available immediately after the MEM cycle of the instruction (i.e., data is immediately available from the cache). You may also assume that the usual forwarding of results from the EX cycle of one instruction immediately into another is available Your diagram should show all of the remaining stall cycles due to dependencies in the code.

(continued next page)

Question 8 (cont). Code repeated for reference below:

```
addi $t0,1            # load 1 into $t0
lw   $t1,100($t5)     # load first variable
add  $t1,$t0,$t1      # increment
sw   $t1,100($t5)     # store
lw   $t2,104($t5)     # load 2nd variable
add  $t2,$t0,$t2      # increment
sw   $t2,104($t5)     # store
```

(b) Can you reorder the instructions to reduce the total number of cycles needed to perform the same computation? If so, show the new order and draw a new diagram showing how the reordered instructions are executed in the pipeline.

(c) How many total cycles were needed to execute the instructions in the original order? How many cycles are needed in the new order?