

## CSci 330, Fall 2006, Midterm

Name: \_\_\_\_\_

1. [10 pts] Distinguish between a load-store instruction set (such as x86) and a memory-register instruction set (such as MIPS). What are their relative performance advantages in today's technology?
2. [8 pts] The MIPS calling convention includes some caller-save registers, such as \$t0 and \$t1, whereas the calling conventions for the 8080 and the IA-32 have no caller-save registers. Explain why caller-save registers helps to improve program performance.
3. [20 pts] Suppose we have a subroutine `func` that, given an integer parameter  $x$ , returns some mysterious function of  $x$  (such as, say,  $n!$  or  $\lfloor 100 \sin n \rfloor$ ). The function works entirely with integers.  
At right, write a MIPS assembly language subroutine `sumFunc` that, given a parameter which is the address of the first element of an array, and a second parameter which is the number of elements in that array, determines the sum of the function values for the array entries. For example, if `func` happens to compute  $x^2$ , and the array specified in the parameters is  $\langle 1, 3, 6 \rangle$ , then `sumFunc` would return  $1^2 + 3^2 + 6^2 = 46$   
(This will be a fairly long subroutine...)

4. [10 pts] Translate each of the following MIPS assembly instructions into its 32-bit machine language equivalent. Note: \$t0 is encoded as register number 8.
  - a. `xor $t0, $t1, $t2`
  
  - b. `lw $t0, 4($t1)`
  
5. [8 pts] Define the term *pseudo-operation* as used in the context of assembly language programs.
  
  
  
  
  
  
  
  
  
  
6. [10 pts] Name and explain two advantages of message passing over shared memory.
  
  
  
  
  
  
  
  
  
  
7. [8 pts] What does the term *hyperthreading* or *simultaneous multithreading* mean in processor design?

8. [10 pts] Describe the *write invalidate* protocol for cache coherency in shared-memory systems.

9. [8 pts] What do the letters of the acronym *SIMD* stand for? What does the acronym mean?

10. [8 pts] Consider the following sequence of instructions to compute  $x + y + xy$  for two parallel vectors  $\$v0$  and  $\$v1$ .

```
addv    $v2, $v0, $v1
multv   $v3, $v0, $v1
addv    $v4, $v2, $v3
```

Let  $n$  represent the length of the vector supported on our processor, where  $n \geq 8$ . Assume that we have a vector processor with one vector multiplication unit whose latency is 7 and one vector addition unit whose latency is 6. If the processor does not avoid latency in subsequent instructions, but it does support forwarding between instructions (a.k.a. chaining), how many clock cycles will this instruction sequence take?