CS341l Fall 2009 Test #1

Friday, 18 September 2009 10-10:50am

Name: _____**Key**_____

CS 341l Fall 2009, Test #1. 100 points total, number of points each question is worth is indicated in parentheses. Answer all questions. Be as concise as possible while still answering the question. **You have 50 minutes exactly, I'll collect the tests at 10:50am.** Closed book, closed notes, closed calculator, closed everything.

#1. (24 points, 4 points each) Circle T for true or F for false.

* MIPS assembly is strongly typed, so if you move a signed integer into a register you can only use it as a signed integer, not an unsigned integer or a pointer, etc.

A register is a 32-bit pattern that can mean anything you want. F

Memory is the same way, it's just a bit pattern until you interpret it as something.

* An assembler takes machine code and compiles it into microcode. F

It takes assembly code and assembles it into machine code.

* Non-volatile memory retains data even in the absence of a power source.

This was one of the blue terms in the margin of the book. T

* The `jal Label` instruction differs from the `j Label` instruction in that it saves the return pointer in memory on the stack to support subroutine calls.

It save the return pointer into $ra, you have to put it on the stack. F

* The instruction in a branch delay slot never gets executed, it's more of a "filler."

It is <u>always</u> executed. F

* In floating point numbers, I can always assume that (a + b) + c = a + (b + c), but not necessarily that "a + b = a" is not true when b does not equal 0.

Can't assume either of these things, due to rounding F

#2. (16 points) Draw lines to divide the following MIPS code into basic blocks, and then indicate below the total number of basic blocks. Three points for each division line you draw in a correct place and the rest of the points for getting the total right.

```
strcpy:
        move    t0, a0
strcpy_top:
        lb      t1, 0(a0)
        beq     t1, zero, strcpy_bottom
        sb      t1, 0(a1)
        addi    a0, a0, 1
        addi    a1, a1, 1
        j       strcpy_top
strcpy_bottom:
        sb      zero, 0(a1)
        move    v0, t0
        jr      ra
```

How many basic blocks are there total? ___**4**___

#3. Multiple choice (40 points, 5 points each). Circle only the one correct answer.

* Which of the following is not an R-type instruction?

a. add $t0, $t0, $t1

b. sll $t1, $t2, 3

c. addi $t1, $t2, 3

d. jr $ra

* Which of the following is not an I-type instruction?

a. bne $t1, $t0, SomePlace

b. sll $t1, $t2, 3

c. addi $t1, $t2, 3

d. slti $t1, $t2, 5

* What type of compiler optimization would the compiler most likely be applying if it removed a loop with 10 iterations from my code and replaced it with the body of the loop repeated 10 times in a row (and no more loop)?

a. Constant folding

b. Dead code elimination

c. Loop unrolling

d. Strength reduction

* What type of compiler optmization would the compiler most likely be applying if it took an invariant statement inside a loop, e.g, “i = 5;” and moved it outside the loop?

a. Constant folding

b. Common subexpression elimination

c. Code motion

d. Dead code elimination

* Which of these ways of representing numbers in a machine would I most likely choose when I needed a large range of numbers, from quantum sized fractions all the way up to astronomically large numbers larger than the number of atoms in the universe?

a. Saturating fixed point

b. Unsaturating fixed point

c. Two's complement

d. Floating point

* Which of these will cause an exception in C?

a. Dividing a floating point number by 0.0 *(NaN)*

b. Dividing an interger by 0

c. Overflowing an unsigned integer *(no warning in C)*

d. Overflowing a signed integer *(C always compiles using unsigned int operations in MIPS due to the C standard)*

* Within a function in a typical C/UNIX program, which of these would typically go on the process' stack, specifically, within the function's stack frame?

a. Local variables that are local to the function

b. Global variables

c. Dynamically allocated memory managed with malloc() and free()

d. Constant string literals such as, “Hello World”

* In a typical C/UNIX program, which of these would typically go on the process' heap?

a. Local variables that are local to the main() function

b. Global variables

c. Dynamically allocated memory managed with malloc() and free()

d. Constant string literals such as, “Hello World”

#4 (13 points, 2 points each, and another point for specifying the range that that addressing mode allows you to address, where two of the ranges are done for you as examples) Your answers should have a one-to-one correspondence with MIPS' five addressing modes and be the best fit, so while it's true that, *e.g.*, loads and stores use multiple forms of addressing, choose the best one-to-one mapping of possible answers.

*Which MIPS addressing mode allows me to choose among the 32 registers?

_____**__Register addressing_____**

Range?: *0..31*

*Which MIPS addressing mode allows me to specify a constant in the instruction that can be directly used by the CPU without needing any data from the registers or memory?

_____**__Immediate addressing_____**

Range?: **Immedaite value can be -2^15 to 2^15-1**

*Which MIPS addressing mode is used for conditional branches?

_____PC-relative addressing_____

Range?: *About +/- 2^16 instructions, or about +/- 2^18 bytes*

*Which MIPS addressing mode is used for jumps?

___Pseudo-direct addressing_____

Range?: W**ithin the 256MB where the PC currently points**

*Which MIPS addressing mode is used for load and store instructions?

____Base+offset addressing_____

Range?: **Anywhere in memory, or the base +/- about 2^15**

(Either answer is acceptable)

#5 (4 points) What is the difference between registers and memory (your answer should be based on the load-store machine concept)?

Registers are special locations built directly in hardware (see page 80 of the textbook), while memory is where programs and data are kept. For the CPU to manipulate values to do operations such as addition, the values must be loaded from memory into the registers, manipulated in the registers, and stored back out to memory.

#6 (1 points) Who is often credited with the stored program concept, and has the type of machine where code and data occupy the same memory named after him?

John von Neumann

#7 (1 points) Which IBM machine, that was developed for Los Alamos in the late 50's/early 60's (and was a predecessor to the IBM/360), was highly influential on modern instruction set architectures (such as MIPS) and architectural paradigms in general (such as memory protection and pipelining)?

#8 (1 point) Give a sound and precise algorithm that can disassemble the full x86 instruction set, with arbitrary obfuscation, and is gauranteed to terminate in the general case.

Not possible.

#9 (no points, just for fun) Which 1977 Italian horror film, directed by Dario Argento and filmed in technicolor with anamorphic lenses, features dream-like depictions of scenes in a German dance school that turns out to be run by witches?

Suspiria.