

COS 471A, COS 471B/ELE 375 Assignment #1

Due Oct 7th at 5pm, 2005

Fall 2005, Princeton University

Homework Submission Instructions:

Your course enrollment (COS 471A, COS 471B/ELE 375) should be listed with your name on the front page of your submission. Please submit your assignment using the envelope outside CS 004. **LATE SUBMISSIONS:** Record the date and time of your submission.

Do the following:

1. In MIPS assembly (documented thoroughly in Appendix A), write an assembly language version of the following C code segment:

```
for(i = 0; i < 98; i ++){
    C[i] = A[i + 1] - A[i] * B[i + 2];
}
```

Arrays A, B and C start at memory location $A000_{hex}$, $B000_{hex}$ and $C000_{hex}$ respectively. Try to reduce the total number of instructions and the number of expensive instructions such as multiplies.

2. P&H 3rd ed 2.15.
3. (roughly P&H 2nd ed 3.10, 3rd ed 2.37) Pseudo-instructions are not part of the MIPS instruction set but often appear in MIPS programs. For each pseudo-instruction in the following table, produce a minimal sequence of actual MIPS instructions to accomplish the same thing. You may need to use `$at` for some of the sequences. In the following table, `big` refers to a specific number that requires 32 bits to represent and `small` to a number that can be expressed using 16 bits.

Pseudo-instruction	What it accomplishes
<code>move \$t5, \$t3</code>	<code>\$t5 = \$t3</code>
<code>clear \$t5</code>	<code>\$t5 = 0</code>
<code>li \$t5, small</code>	<code>\$t5 = small</code>
<code>li \$t5, big</code>	<code>\$t5 = big</code>
<code>lw \$t5, big(\$t3)</code>	<code>\$t5 = Memory[\$t3 + big]</code>
<code>addi \$t5, big(\$t3)</code>	<code>\$t5 = \$t3 + big</code>
<code>beq \$t5, small, L</code>	<code>if(\$t5 = small) go to L</code>
<code>beq \$t5, big, L</code>	<code>if(\$t5 = big) go to L</code>
<code>ble \$t5, \$t3, L</code>	<code>if(\$t5 <= \$t3) go to L</code>
<code>bgt \$t5, \$t3, L</code>	<code>if(\$t5 > \$t3) go to L</code>
<code>bge \$t5, \$t3, L</code>	<code>if(\$t5 >= \$t3) go to L</code>

4. (P&H 2nd ed 3.12, 3rd ed 2.38) Given your understanding of PC-relative addressing, explain why an assembler might have problems directly implementing the branch instruction in the following code sequence:

```
here:      beq $t1, $t2, there
          . . .
there:    add $t1, $t1, $t1
```

Show how the assembler might rewrite this code sequence to solve these problems.

5. Assume the following instruction mix for a MIPS-like RISC instruction set: 10% stores, 30% loads, 15% branches, 35% integer arithmetic, 5% integer shift, and 5% integer multiply. Given that load instructions require 2 cycles, branches require 4 cycles, integer ALU and store instructions require one cycle, and integer multiplies require 10 cycles, compute the overall CPI.

6. A designer wants to improve the overall performance of a given machine with respect to a target benchmark suite and is considering an enhancement X that applies to 55% of the original dynamically-executed instructions, and speeds each of them up by a factor of 3. The designer's manager has some concerns about the complexity and the cost-effectiveness of X and suggests that the designer should consider an alternative enhancement Y . Enhancement Y , if applied only to some (as yet unknown) fraction of the original dynamically-executed instructions, would make them only 75% faster. Determine what percentage of all dynamically-executed instructions should be optimized using enhancement Y in order to achieve the same overall speedup as obtained using enhancement X .