Computer Architecture - Problem-set-1

Context, Objectives and Organization

This problem set covers material from Lectures 1 (introduction to CA and technology trends), 2 (CPU performance equations) and 3 (Instruction Set Architecture). The main goals of this tutorial is to give you some quantitative experience with the CPU performance equation.

Problem 1

Problem

After graduating, you are asked to become the lead computer designer at Hyper Computers Inc. Your study of usage of high-level language constructs suggests that procedure calls are one of the most expensive operations. You have invented a scheme that reduces the loads and stores normally associated with procedure calls and returns. The first thing you do is run some experiments with and without this optimization. Your experiments use the same state-of-the-art optimizing compiler that will be used with either version of the computer. These experiments reveal the following information:

- The clock rate of the unoptimized version is 5% higher.
- 30% of the instructions in the unoptimized version are loads or stores.
- The optimized version executes 2/3 as many loads and stores as the unoptimized version. For all other instructions the dynamic counts are unchanged.
- All instructions (including load and store) take one clock cycle.

Which is faster? Justify your decision quantitatively.

Problem 2

Problem

Several researchers have suggested that adding a register-memory addressing mode to a load-store machine might be useful. The idea is to replace sequences of:

```
LOAD    Rx,0(Rb)
ADD     Ry,Ry,Rx
```

by

```
ADD     Ry,0(Rb)
```

Assume this new instruction will cause the clock period of the CPU to increase by 5%. Use the instruction frequencies for the gcc benchmark on the load-store machine from Table 1. The new instruction affects only the clock cycle and not the CPI.

1. What percentage of the loads must be eliminated for the machine with the new instruction to have at least the same performance?

2. Show a situation in a multiple instruction sequence where a load of a register (say Rx) followed immediately by a use of the same register (Rx) in an ADD instruction, could not be replaced by a single ADD instruction of the form proposed.

Vijay Nagarajan, 2023

Instruction	**Frequency**
load	22.8%
store	14.3%
add	14.6%
sub	0.5%
mul	0.1%
div	0%
compare	12.4%
load imm	6.8%
cond. branch	11.5%
uncond. branch	1.3%
call	1.1%
return, jump ind.	1.5%
shift	6.2%
and	1.6%
or	4.2%
other (xor, not)	0.5%

Table 1: Instruction frequencies for gcc (cc1) (from H&P (2/e), Figure 2.26, p.105)