Computer Architecture - tutorial 1

Context, Objectives and Organization

This worksheet covers material from Lectures 1 (introduction to CA and technology trends), 2 (CPU performance equations) and 3 (the first half of the section on Instruction Set Architecture).

The main goals of this tutorial is to give you some quantitative experience with the CPU performance equation, and to stimulate a discussion where your group can explore some issues related to modern instruction set design.

This tutorial consists of two activities. The first involves quantitative problem solving. The exercises are taken from the H&P book 2nd or 3rd editions. The suggested number of students per group and duration of the activity are given with each exercise. The second main activity involves qualitative discussions. The suggested format is to allow for some discussion in small groups, followed by an exchange of ideas with the whole class.

E1: H&P (2/e) 1.6 p.61 – individual exercise, 15 mins.

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.

E2: H&P (2/e) 2.6 p.164 – in groups of 2, 10 mins.

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    R1,0(Rb)
ADD     R2,R2,R1
```
   by
```
ADD     R2,0(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 R1 followed immediately by a use of R1 (with some type of opcode) could not be replaced by a single instruction of the form proposed, assuming that the same opcode exists.

D1: Discussion – in groups of 4, 15 mins.

In the early years of the RISC versus CISC dispute, the total number of different instructions and their variations in the ISA was a common indication of the "simplicity" of an ISA (lesser the number, greater the simplicity). Modern RISC instruction sets contain almost as many instructions as old CISC instruction sets. Discuss whether modern "RISC" processors are no longer RISC (as envisioned in the 80's). If they are still RISC, then what features in the instruction set best define the simplicity of an ISA? (e.g. memory access instructions, fixed and simple instruction encoding, register-oriented instructions, simple data types, etc?)

D2: Discussion – in groups of 4, 10 mins.

Even though the Intel x86 ISA is a clear example of a CISC ISA, modern implementations of it (e.g. Core and Xeon) use many RISC ideas: register-based micro-instructions, pipelining, simple branch micro-instructions, fixed length micro-instructions, etc. Some say that, since at the low level the Pentium *behaves* like a RISC, it is RISC. Others say that, since at the software interface (compiler) it *is seen* like a CISC, it is CISC. Discuss what level we should measure the instructions executed by a processor implementation? What are the implications of considering the ISA at each level? Is the Pentium RISC?

<div align="right">

Vijay Nagarajan 2014. thanks to Nigel Topham and Marcelo Cintra.

</div>

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)