Universidade de Aveiro

Mestrado Integrado em Engenharia de Computadores e Telemática

Arquitectura de Computadores Avançada

Lesson 1: Performance Measurement

Academic year 2015/2016 — Nuno Lau/José Luís Azevedo

1. Consider a computer with a clock frequency of 100MHz. It has been determined that the execution of a particular program executes 1.2×10^9 instructions in 30 seconds. What is the average CPI (cycles per instruction) for this program on this computer?

2. A given computer has a clock frequency of 1 GHz and an average CPI of 1.8. The implementation of a new mode of branch (jump) instructions is being considered that allows reducing the number of instructions executed by a program by 20% which requires, however, decreasing the clock frequency to 0.7GHz. What is the speedup obtained with this implementation change (assuming the CPI remains the same)?

3. Consider two implementations of the same instruction set, M1 and M2. The clock frequency of implementation M1 and M2 is 50MHz and 100MHz, respectively. There are three classes of instructions, A, B and C. The CPI of each class of instructions is given by:

	CPI	
Class	M1	M2
A	1	3
B	2	2
C	3	4

   3.1. What is the peak performance of the implementation M1 (measured in MIPS)?

   3.2. Considering that the number of instructions executed by a given program is distributed equally among the three classes A, B and C, determine the CPI for M1 and M2.

   3.3. For the conditions of the preceding paragraph, determine the speedup of M2 with respect to M1.

4. The `licao1.tgz` archive (available in the UA moodle – http://elearning.ua.pt), contains the following files:

   - `dhrystone.c` – C code for an implementation of the synthetic Dhrystone benchmark (despite the disadvantages of this type of benchmarks - see pages 79 and 80 of [1] -, we will use it in some exercises of this lesson).
   - `prime.c` - C code of a program that calculates, using a simple algorithm, the number of prime numbers that exists up to 10^6. We will use this program as a toy benchmark.
   - `run_cc1.sh` – *Script* that executes the `cc1plus` compiler (version 2.95) taking as input the file `mips.i` (`mips.i` and `cc1plus` also stored in `licao1.tgz` archive).

Extract the `licao1.tgz` archive to your workspace using the Linux operating system (`tar xvzf licao1.tgz`).

4.1. Compile the `dhrystone` and `prime` programs without using optimization options. Execute both programs and take note of the respective execution times.

4.2. Recompile the same programs using maximum optimization options and take note, again, of the execution times.

4.3. Run the script `run_cc1.sh` using the `time` command. Take note of the respective execution times.

4.4. Compile the programs without optimization and with maximum optimization options. Execute the programs in Windows operating system and take note of the results.

4.5. Taking as reference machine your local computer using Windows, calculate the speedups obtained by running the unoptimized and optimized versions in Linux. Describe all relevant characteristics of the computer under test. Discuss the results.

5. Using the Linux program `gprof` it is possible to check the time spent by a program in each of its procedures. This operation, generally designated by profiling, allows knowing the points at which the program spends more time, thus guiding the effort to optimize to where it is needed. The steps to perform this analysis for the program `dhrystone.c` are as follows:

   - compile the program using the `-pg`:

     ```
     cc -pg dhrystone.c -o dhry_prof
     ```

   - run the program:

     ```
     ./dhry_prof
     ```

   - run `gprof` to summarize the results:

     ```
     gprof dhry_prof > gprof.out
     ```

   The file `gprof.out` now contains all the profiling information about the `dhrystone.c` program. In its first lines `gprof.out` presents a table that shows the time the program spent in each of its procedures.

   5.1. Execute the steps previously described.

   5.2. To optimize the `dhrystone` program, based on the `gprof` results, which procedure would you try to change in first place?

   5.3. What is the maximum speedup that can be achieved through the optimization of that procedure? Correlate the result with the Amdhal law.

Bibliography

[1] “Computer Organization and Design”, David Patterson and John Hennessy, 2nd Edition, Morgan Kauffman