Introduction to Computer Architecture

Assignment 1 - Solution

Due April 9, 2015

1. [20 = 10 x 2] You are designing a system for a real-time application in which specific deadlines must be met. Finishing the computation faster gains nothing. You find that your system can execute the necessary code, in the worst case, twice as fast as necessary.

a. How much energy do you save if you execute at the current speed and turn off the system when the computation is complete?
b. How much energy do you save if you set the voltage and frequency to be half as much?

Solution
lec03: P17, P20, P21

a. 50%
b. Energy = ½ load × V^2. Changing the frequency does not affect energy–only power. So the new energy is ½ load × (½ V)^2, reducing it to about ¼ the old energy.

2. [10] Discuss why ideal pipelining with equal-length pipe stages yields the highest speedup?

Solution:
Open discussion.

3. [25 = 5 x 5] When parallelizing an application, the ideal speedup is speeding up by the number of processors. This is limited by two things: percentage of the application that can be parallelized and the cost of communication. Amdahl's law takes into account the former but not the later.

a. What is the speedup with *N* processors if 80% of the application is parallelizable, ignoring the cost of communication?
b. What is the speedup with 8 processors if, for every processor added, the communication overhead is 0.5% of the original execution time?
c. What is the speedup with 8 processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?
d. What is the speedup with *N* processors if, for every time the number of processors

is doubled, the communication overhead is increased by 0.5% of the original execution time?
e. Write the general equation that solves this question: What is the number of processors with the highest speedup in an application in which *P*% of the original execution time is parallelizable, and, for every time the number of processors is doubled, the communication is increased by 0.5% of the original execution time?

Solution

a. $1/(.2 + .8/N)$

b. $1/(.2 + 8 \times 0.005 + 0.8/8) = 2.94$

c. $1/(.2 + 3 \times 0.005 + 0.8/8) = 3.17$

d. $1/(.2 + \log N \times 0.005 + 0.8/N)$

e. $d/dN(1/((1 - P) + \log N \times 0.005 + P/N)) = 0$

4. [32 = 8 x 4] For the following assume that values A, B, and C reside in memory. Also assume that instruction operation codes are represented in 8 bits, memory addresses are 64 bits, and register addresses are 6 bits.

For each instruction set architecture shown in Figure A.2 (*stack*, *accumulator*, *register-memory*, *register-register*), how many addresses, or names, appear in each instruction for the code to compute C = A + B, and what is the total code size?

Solution
lec04: P10-15

1) Stack:
Push A // one address appears in the instruction, code size = 8 bits (opcode) + 64 bits (memory address) = 72 bits;
Push B // one address appears in the instruction, code size = 72 bits;
Add // zero address appears in the instruction, code size = 8 bits;
Pop C // one address appears in the instruction, code size = 72 bits;

Total code size = 72 + 72 + 8 + 72 = 224 bits.

2) Accumulator
Load A // one address appears in the instruction, code size = 8 bits (opcode) + 64 bits (memory address) = 72 bits;
Add B // one address appears in the instruction, code size = 72 bits;
Store C // one address appears in the instruction, code size = 72 bits;

Total code size = 72 + 72 + 72 = 216 bits.

3) Register-memory
Load R1, A // two addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 64 bits (memory address) = 78 bits;
Add R3, R1, B // three addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 6 bits (register address) + 64 bits (memory address) = 84 bits;
Store R3,C // two addresses appear in the instruction, code size = 78 bits;

Total code size = 78 + 84 + 78 = 240 bits.

4) Register-register
Load R1, A // two addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 64 bits (memory address) = 78 bits;
Load R2, B // two addresses appear in the instruction, code size = 78 bits;
Add R3, R1, R2 // three addresses appear in the instruction, code size = 8 bits (opcode) + 6 bits (register address) + 6 bits (register address) + 6 bits (register address) = 26 bits;
Store R3, C // two addresses appear in the instruction, code size = 78 bits;

Total code size = 78 + 78 + 26 + 78 = 260 bits.

5. [8] Briefly introduce research projects (if any) you have participated. Are you interested in the Research Warmup component? Any research topic you would like to explore?

Solution:
Open discussion.

6. [5] If you were the lecturer, what would you do to improve students' study experience of this course?

Solution:
Open discussion.