ELEC 5200-001/6200-001 Computer Architecture and Design
Spring 2017
Homework 5 Solution

Problem 1:

(a) Run times for four programs (A, B, C and D) are recorded on three processors (X, Y and Z):

Program	Run time (seconds)		
	Processor X	Processor Y	Processor Z
A	4	3	3
B	20	50	20
C	700	700	700
D	1200	800	1000

Compute the mean run time for each processor. Which is the fastest processor?

(b) Suppose X is a single processor with an average power consumption of 10W and Y is chip multiprocessor (CMP) that uses dynamic voltage and frequency scaling (DVFS) for power management and consumes 5W. Compute the efficiencies of the two processors. What is the relative efficiency of Y with respect to X? Which processor is more efficient?

Solution:

(a) Using the geometric mean formula, we obtain the relative performances:

Run time (X) = $(4 \times 20 \times 700 \times 1200)^{1/4}$ = 90.54 s

Run time (Y) = $(3 \times 50 \times 700 \times 800)^{1/4}$ = 95.74 s

Run time (Z) = $(10 \times 50 \times 500 \times 1000)^{1/4}$ = 80.50 s

Z is the fastest processor.

(b) Energy efficiencies are as follows:

$\eta(X) = [(1/20)(1/200)(1/7000)(1/12000)]^{1/4}$ = 0.00131/J

$\eta(Y) = [(1/15)(1/150)(1/3500)(1/4000)]^{1/4}$ = 0.00209/J

Relative efficiency, $\eta(Y/X) = 209/131 = 1.6$

Processor Y is more energy efficient.

Problem 2: Suppose in a typical target applications of a processor 30% of execution time is spent on floating point additions, 25% on floating point multiplications, 25% on floating point divisions, and the remaining time is used by other operations including integer math. To create a higher performance processor, a design team examines three possible hardware changes, each costing about the same in design effort and manufacturing: (i) Make floating point adder twice as fast, (ii) Make floating point multiplier 20 times as fast, (iii) Make floating point divider 100 times as fast. What will the speedup be if these design changes are incorporated? To hold the cost down, if we can include only one of the three enhancements, then which one should that be?

Solution: Amdahl's law states,

$$\text{Speedup} = 1/[(1 - f) + f/n]$$

Where f is the enhanced fraction and n is the speed up factor for the enhanced fraction.
For all three enhancements,

$$\text{Speedup} = 1/[1 - 0.3 - 0.25 - 0.25 + 0.3/2 + 0.25/20 + 0.25/100]$$

$$= 2.74$$

The enhanced fraction f = 0.30, 0.25 and 0.25 for addition, multiplication and division, respectively. Corresponding speed up factor n = 2, 20 and 100. Speed up ratios for the three options are:

i.	Speedup for adder redesign	= 1/ [0.70 + 0.30/2]	= 1.1765
ii.	Speedup for multiplier redesign	= 1/ [0.75 + 0.25/20]	= 1.3115
iii.	Speedup for divider redesign	= 1/ [0.75 + 0.25/100]	= 1.3289

The divider redesign is the best choice.

Problem 3: Consider n processor cores in a distributed memory system. The compiler distributes a program to processors such that independent executions are possible. However, after execution of instructions the processors must communicate to keep the data consistent. Given that the inter-processor communication network has a finite bandwidth the total amount of communication through the network grows as more processors are used by a program. Suppose we model the communication overhead for n processors as $\beta(n - 1)$, where constant β, the inter-processor communication overhead factor, has some small positive value. Show that the speed up can be expressed as,

$$\text{Speedup} = \frac{1}{(1/n) + \beta(n - 1)} \qquad (1)$$

Also show that:

(a) The fraction f_{enh} of Amdahl's law in this system equals $1 - n\beta$.
(b) For non-zero β, maximum speed up occurs when $n = 1/\sqrt{\beta}$.
(c) For a very small value of the communication overhead $\beta > 0$, the speed up, which is always less than n, attains the maximum value when it is n/2.

Solution: We assume that K instructions are equally distributed to n processors. Then,

$$\text{Speedup} = \frac{K}{(K/n) + K\beta(n-1)} = \frac{1}{(1/n) + \beta(n-1)} \qquad (1)$$

(a) We compare (1) with Amdahl's original formula written below:

$$\text{Speedup} = \frac{1}{1 - f_{enh} + f_{enh}/n} \qquad (2)$$

Equating the denominators of (1) and (2), we get $f_{enh} = 1 - \beta n$.

(b) To find n for maximum speed up, we differentiate the denominator of (1) with respect to n and set it to 0. Thus,

$$-(1/n^2) + \beta = 0, \text{ or } n^2 = 1/\beta, \text{ or } n = 1/\sqrt{\beta}$$

This minimizes the denominator in (1) because the second derivative $2/n^3$ is positive.

(c) Substituting the optimum number of processors in (1) and for very small β,

$$\text{Maximum speedup} = \frac{1}{\sqrt{\beta} + \sqrt{\beta} - \beta} \approx \frac{1}{2\sqrt{\beta}} = n/2, \text{ because } n = \sqrt{(1/\beta)}$$

Problem 4: For an n-processor system, we define parallelizing efficiency as the ratio of actual speedup to the ideal speedup n. For the system of Problem 3 find the maximum speedups achievable with $\beta = 0.001$ and at least 80% parallelizing efficiency.

Solution: As n increases, speedup increases but at a slower rate than n. Thus, while speedup increases with n, parallelizing efficiency drops. From the given definition of parallelizing efficiency and using equation (1) from Problem 3,

$$\text{Parallelizing efficiency} = \text{Speedup}/n = \frac{1/n}{(1/n) + \beta(n-1)} = \frac{1}{1 + \beta(n-1)n} \geq 0.8$$

Therefore, $n(n-1) \leq 1/(4\beta)$

For $\beta = 0.001$ we get the following result: $n(n-1) \leq 250$, $n = 16$, speedup = 12.9

Problem 5:

(a) For two instructions A and B where B follows A, define RAW, WAR and WAW dependencies? How do they influence the out-of-order issuing of B.

(b) Examine the following sequence of four instructions for out-of-order issue:

#1 R6 = R5 + R6
#2 R1 = R5 + R7
#3 R2 = R1 + R6
#4 R3 = R5 – R6

Can #2 be issued in the same cycle as #1? If not when should #2 be issued? Can #3 be issued before #1 or #2 is retired? If not, when should #3 be issued? Can #4 be issued either together with or before #1, 2 or 3? Explain.

Answer:

(a) RAW dependence: If a source register of instruction B is being written to by instruction A, then B has a RAW dependence on A. Thus, B must not be issued until after A is retired.

WAR dependence: If the destination register of instruction B is being read by instruction A, then B has a WAR dependence on A. Thus, B must not be issued until A is retired.

WAW dependence: If the destination register of B is being written by A, then B has a WAW dependence on A. Thus, B must not be issued until A is retired.

(b) #2 has no dependence on #1 and hence #1 and #2 can be issued together in the same cycle.

#3 has RAW dependences on both #1 and #2 and so #3 cannot be issued until after #1 and #2 are both retired.

#4 has a RAW dependence on #1 but no dependence on #2 or #3. So, #4 should be issued only after #1 is retired.