Solutions to HW #1
19 Sept., 96

1.1 (a) Use *Amdahl's Law* as given on page 30, with $speedup_{enhanced} = 20$. The equation reduces to

$$speedup_{overall} = \frac{20}{20 - 19 \times fraction_{enhanced}}$$

Plotting $speedup_{overall}$ versus $fraction_{enhanced}$ yields the plot shown in the figure.

Note that the exponential nature of the curve shows that the optimization does not begin to pay off until large percentages of the code can be run in the vector mode.

1.1 (b) Solving

$$2 = \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{20}}$$

yields $fraction_{enhanced} = 0.53$.

1-1 (c) One-half the maximum speedup attainable using the vector made is 10. Solving

$$10 = \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{20}}$$

yields $fraction_{enhanced} = 0.95$.

1-1 (d) If we double the speed, the speedup relative to the scalar code for a 70% vectorizable code is:

$$speedup_{faster\ vector} = \frac{1}{(1 - fraction_{enhanced} + \frac{fraction_{enhanced}}{2 \times speedup_{original\ vector}}}$$

If we plug in $speedup_{original\ vector} = 20$ and $fraction_{enhanced} = 70\%$ we get a speedup of 3.1.

If we try to improve the vectorization to get the same speedup, we get

$$speedup_{more\ vectorization} = \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{20}} = 3.1$$

solving yields $fraction_{enhanced} = 72\%$.

Thus, with codes which are only 70% vectorized, doubling hardware speed provides a speed improvement equivalent to vectorizing just 2% more of the code. In general, this means that for partially vectorized codes, large improvements in the speed of vector hardware yields only small speed improvement.

1-2 (a) Since we cannot use Amdahl's law, we start from the basics.

$$speedup = \frac{time_{unenhanced}}{time_{enhanced}}$$

Since the enhanced mode is used 50% of the time and improves performance by a factor of 10

$$0.5 \times time_{enhanced} = \frac{percent_{original} \times time_{unenhanced}}{10}$$

where $percent_{original}$ is the percentage of the original system that the portion of the code that has been enhanced took originally. Also, $time_{enhanced}$ can be computed as the inverse of the speedups of the original unenhanced time, i.e

$$time_{enhanced} = (\frac{percent_{original}}{10} + (1 - percent_{original})) \times time_{unenhanced}$$

Eliminating $percent_{original}$ from the previous 2 equations, we get

$$\frac{time_{unenhanced}}{time_{enhanced}} = 5.5 = speedup$$

1-2 (b) Using Amdahl's law

$$speedup_{new} = 5.5 = \frac{1}{(1 - percent_original) + \frac{percent_{original}}{10}}$$

yields $percent_{original} = 91\%$.

1-3 In the example on page 31, we consider the performance impact of either speeding up the FPSQRT operation by a factor of 10, or making all FP operations faster by a factor of 2. Likewise, in the example on page 33, our alternatives are to reduce the CPI (and thus the CPU time) of the FPSQRT instruction from 20 to 2 (a factor of 10) or to reduce the CPU time all FP operations from 4 to 2 (a factor of 2). The two examples are equivalent.

1-4 We evaluate the *load-store* machine in terms of the average CPI.

Instruction type	**Frequency**	**Clock cycles**	**CPI**
ALU	43%	1	0.43
Loads	21%	2	0.42
Stores	12%	2	0.24
Branches	24%	2	0.48

The net CPI, CPI_{orig} is thus $0.43 + 0.42 + 0.24 + 0.48 = 1.57$. The CPU time is given by

$$time_{orig} = 1.57 \times IC_{orig} \times Clk$$

In the new machine, 25% of the ALU instructions have a *paired load instruction that can be removed* by taking advantage of the new register-memory ALU instruction format. As the ALU instructions constitute 43% of the instruction mix, the new version has $25\% \times 43\% = 11\%$ fewer instructions than the original version. This implies we have to normalize the frequencies of the instructions to this new count. We evaluate the CPI of each class again.

Instruction type	**Frequency**	**Clock cycles**	**CPI**
ALU - reg-memory	25% of 43%/89%	2	0.24
ALU - reg-reg	75% of 43%/89%	1	0.36
Loads	$\frac{21\% - (25\% \times 43\%)}{89\%}$	2	0.23
Stores	12%/89%	2	0.27
Branches	24%/89%	**3**	0.81

The net CPI, CPI_{new} is thus $0.24+0.36+0.23+0.27+0.81 = 1.91$. The CPU time is given by

$$time_{new} = IC_{new} \times 1.91 \times Clk = (89\% \times IC_{orig}) \times 1.91 \times Clk = 1.70 \times IC_{orig} \times Clk$$

Because $time_{new}$ is larger than $time_{orig}$, adding register-memory ALU instructions would not be profitable from a performance standpoint for this instruction mix.

1-5 The CPI for an instruction is given by

$$CPI_i = CPI_{ideal} + CPI_{stalls}$$

where CPI_{ideal} is the number of cycles required for the instruction in the presence of a perfect cache, and CPI_{stalls} is given by

$$CPI_{stalls} = (acc_i \times freq_{i-miss} \times pen_{i-miss}) + (acc_d \times freq_{d-miss} \times pen_{d-miss})$$

where
acc_i, acc_d: number of instruction and data accesses,
$freq_{i-miss}, freq_{d-miss}$: frequency of instruction and data misses, and
$pen_{i-miss}, pen_{d-miss}$: the two penalties in terms of the number of cycles.

Plugging in the values given in the problem, we get

$$\begin{aligned} freq_{i-miss} \times pen_{i-miss} &= 0.05 \times 40 = 2 \\ freq_{d-miss} \times pen_{d-miss} &= 0.10 \times 40 = 4 \end{aligned}$$

We need to determine the number of instruction and data accesses for each type of instruction. The results are shown in the table.

Instruction Class	**Freq.**	**Cycles (CPI_{ideal})**	**Instruction Accesses**	**Data Accesses**	**CPI_{stalls}**	**CPI_i**
ALU	43%	1	1	0	2	3
Loads	21%	2	1	1	6	8
Stores	12%	2	1	1	6	8
Branches	24%	2	1	0	2	4

We can now calculate the two overall CPIs.
$CPI_{perfect} = (1 \times 43\%) + (2 \times 21\%) + (2 \times 12\%) + (2 \times 24\%) = 1.6$
$CPI_{real} = (3 \times 43\%) + (8 \times 21\%) + (8 \times 12\%) + (4 \times 24\%) = 4.9$
The speedup is given by

$$speedup = \frac{CPI_{real}}{CPI_{perfect}} = \frac{4.9}{1.6} = 3.1$$

Therefore, with a perfect cache, the machine is 3.1 times faster than it is with a real cache.

1-12 We need to modify Amdahl's Law so that it can handle multiple enhancements. The key feature that makes extending Amdahl's Law straightforward is the fact that only one enhancement can be used at one time. We extend Amdahl's Law as follows:

$$Speedup = [(1 - \sum_i FE_i) + \sum_i \frac{FE_i}{SE_i}]^{-1}$$

where FE_i is the fration of time enhancement i could be used in the unoptimized system, and SE_i is the speedup from enhancement i. *Note* that this form of Amdahl's Law works only if the enhancements are non-overlapping, i.e. at most *one* enhancement is usable at any point of time.

1-12 (a) Plugging the values into the modified version of Amdahl's Law gives

$$\begin{aligned} Speedup &= \{[1 - (FE_1 + FE_2 + FE_3)] + \frac{FE_1}{SE_1} + \frac{FE_2}{SE_2} + \frac{FE_3}{SE_3}\}^{-1} \\ 10 &= \{[1 - (0.30 + 0.30 + FE_3)] + \frac{0.30}{30} + \frac{0.30}{20} + \frac{FE_3}{10}\}{-1} \\ 0.10 &= 0.425 + (0.10 - 1)FE_3 \\ &\Rightarrow FE_3 = 0.36 \end{aligned}$$

Therefore, the third enhancement must be usable in the unenhanced system 36% of the time to achieve an overall speedup of 10.

1-12 (b) Using the modified Amdahl's Law yields

$$\begin{aligned} Speedup &= \{[1 - (0.3 + 0.3 + 0.2)] + \frac{0.3}{30} + \frac{0.3}{20} + \frac{0.2}{10}\}^{-1} \\ &= \frac{1}{0.245} \end{aligned}$$

This implies that $time_{enhanced} = 0.245 \times time_{unenhanced}$. Also, the term corresponding to the time spent in the unenhanced mode in the enhanced system is 0.2(1 - (0.3 + 0.3 + 0.2)). Thus, the unenhanced mode accounts for $\frac{0.2}{0.245} = 82\%$ of the execution time on the enhanced system.

1-12 (c) The speedups obtained using only *one* enhancement are:

$$
\begin{aligned}
Speedup_1 &= (1 - 0.20 + \tfrac{0.20}{30})^{-1} = 1.240 \\
Speedup_2 &= (1 - 0.20 + \tfrac{0.20}{20})^{-1} = 1.235 \\
Speedup_3 &= (1 - 0.70 + \tfrac{0.70}{10})^{-1} = 2.703
\end{aligned}
$$

The third enhancement would be the choice since it yields the highest speedup.

The speedups obtained using a pair of enhancements (using the modified Amdahl's Law) are:

$$
\begin{aligned}
Speedup_{12} &= [(1 - 0.20 - 0.20) + (\tfrac{0.20}{30} + \tfrac{0.20}{20})]^{-1} = 1.622 \\
Speedup_{13} &= [(1 - 0.20 - 0.70) + (\tfrac{0.20}{30} + \tfrac{0.70}{10})]^{-1} = 5.660 \\
Speedup_{23} &= [(1 - 0.20 - 0.70) + (\tfrac{0.20}{20} + \tfrac{0.70}{10})]^{-1} = 5.556
\end{aligned}
$$

The choice would be the first and third enhancements.