# High Performance Computer Architecture (CS60003)

## Quiz - 1 (Solutions)

1. (a) Number of Cycles required to execute each type of these instructions are:

$$\text{Arithmetic and Logic} = 1 \times 60\% \times 200,000 = 120,000$$
$$\text{Load/store with cache hit} = 2 \times 18\% \times 200,000 = 72,000$$
$$\text{Branch} = 4 \times 12\% \times 200,000 = 96,000$$
$$\text{Memory reference with cache miss} = 8 \times 10\% \times 200,000 = 160,000$$

Hence, total number of cycles required to execute all the 200,000 instructions = 120,000 + 72,000 + 96,000 + 160,000 = 448,000

Thus,

$$\text{Average CPI} = \frac{\text{Total Cycles}}{\text{Total Number of Instructions}} = \frac{448,000}{200,000} = 2.24$$

(b) Frequency of the processor ($f$) is 40MHz, i.e., $40 \times 10^6$ cycles/sec. Average CPI as calculated in (a) is 2.24, i.e., 2.24 cycles/inst. Number of instructions executed per second (IPS) by the processor = $\frac{f}{CPI}$. Hence,

$$\text{IPS} = \frac{f}{CPI} = \frac{40 \times 10^6}{2.24} = 17.86 \times 10^6$$

Hence, MIPS rate = 17.86.

2. Let us denote

(a) $CPI_{xM_1}$ - average CPI of a class X instruction on $M_1$.

(b) $CPI_{yM_1}$ - average CPI of a class Y instruction on $M_1$.

(c) $CPI_{xM_2}$ - average CPI of a class X instruction on $M_2$.

(d) $CPI_{yM_2}$ - average CPI of a class Y instruction on $M_2$.

Assume there are $n$ instructions of class X and class Y each.

Average cycles required to execute a single instruction (CPI) of both $M_1$ and $M_2$ for $B_1$ is $\frac{1GHz}{500MIPS} = \frac{1 \times 10^9}{500 \times 10^6} = 2$.

Average CPI on $M_1$ for $B_1$ can be expressed as:

$$\frac{n \times CPI_{xM_1} + n \times CPI_{yM_1}}{2n} = 2 \tag{1}$$

Average CPI on $M_2$ for $B_1$ can be expressed as:

$$\frac{n \times CPI_{xM_2} + n \times CPI_{yM_2}}{2n} = 2 \tag{2}$$

Equation (1) and (2) can be simplified to

$$CPI_{xM_1} + CPI_{yM_1} = 4 \tag{3}$$
$$CPI_{xM_2} + CPI_{yM_2} = 4 \tag{4}$$

After replacing half of the class X instructions with class Y instructions (suite $B_2$), average CPI on $M_1$ can be expressed as

$$\frac{(\frac{n}{2} \times CPI_{xM_1} + \frac{n}{2} \times CPI_{yM_1}) + n \times CPI_{yM_1}}{2 \times n} \tag{5}$$

and average CPI on $M_2$ can be expressed as

$$\frac{(\frac{n}{2} \times CPI_{xM_2} + \frac{n}{2} \times CPI_{yM_2}) + n \times CPI_{yM_2}}{2 \times n} \tag{6}$$

It is given that $M_1$'s running time is 70% of $M_2$. Therefore, we can relate these two average CPIs as

$$\frac{(\frac{n}{2} \times CPI_{xM_1} + \frac{n}{2} \times CPI_{yM_1}) + n \times CPI_{yM_1}}{2 \times n} = 0.7 \times \frac{(\frac{n}{2} \times CPI_{xM_2} + \frac{n}{2} \times CPI_{yM_2}) + n \times CPI_{yM_2}}{2 \times n} \tag{7}$$

This can be simplified to

$$CPI_{xM_1} + 3 \times CPI_{yM_1} = 0.7 \times CPI_{xM_2} + 2.1 \times CPI_{yM_2} \tag{8}$$

Similarly, for suite $B_3$, average CPI on $M_1$ can be expressed as

$$\frac{n \times CPI_{xM_1} + (\frac{n}{2} \times CPI_{xM_1} + \frac{n}{2} \times CPI_{yM_1})}{2 \times n} \tag{9}$$

and average CPI on $M_2$ can be expressed as

$$\frac{n \times CPI_{xM_2} + (\frac{n}{2} \times CPI_{xM_2} + \frac{n}{2} \times CPI_{yM_2})}{2 \times n} \tag{10}$$

It is given that $M_1$'s running time is 1.5 times of $M_2$. Therefore, we can relate these two quantities as

$$\frac{n \times CPI_{xM_1} + (\frac{n}{2} \times CPI_{xM_1} + \frac{n}{2} \times CPI_{yM_1})}{2 \times n} = 1.5 \times \frac{n \times CPI_{xM_2} + (\frac{n}{2} \times CPI_{xM_2} + \frac{n}{2} \times CPI_{yM_2})}{2 \times n} \tag{11}$$

This can be simplified to

$$3 \times CPI_{xM_1} + \times CPI_{yM_1} = 4.5 \times CPI_{xM_2} + 1.5 \times CPI_{yM_2} \tag{12}$$

Solve the 4-variable system of equations formed by Equations (3), (4), (8) and (12). This gives the required CPIs

(a) $CPI_{xM_1} = 2.49$
(b) $CPI_{yM_1} = 1.50$
(c) $CPI_{xM_2} = 0.99$
(d) $CPI_{yM_2} = 3.00$

3. (a) Here, 60% of the computation time can be used by the floating-point processor. Hence, $F_{enh} = 0.6$. The speedup of the floating-point processor is 40% faster. Hence, $S_{enh} = 1.4$. Thus, according to Amdahl's Law,

$$\begin{aligned}\text{Overall Speedup} &= \frac{1}{(1 - F_{enh}) + \frac{F_{enh}}{S_{enh}}} \\ &= \frac{1}{(1 - 0.6) + \frac{0.6}{1.4}} \\ &= \frac{1}{0.4 + 0.429} \\ &= 1.206\end{aligned}$$

(b) Take **Cost/Speedup** ratio to quantitatively compare between the two options. We will select the Option having lower value of this ratio.

**Option 1:** Here, 70% of the computation time can be used by the floating-point processor. Hence, $F_{enh} = 0.7$. $S_{enh} = 1.4$ as before. Thus, according to Amdahl's Law,

$$\begin{aligned}\text{Overall Speedup} &= \frac{1}{(1 - F_{enh}) + \frac{F_{enh}}{S_{enh}}} \\ &= \frac{1}{(1 - 0.7) + \frac{0.7}{1.4}} \\ &= \frac{1}{0.3 + 0.5} \\ &= 1.25\end{aligned}$$

**Cost/Speedup** $= 50/1.25 = 40$

**Option 2:** Here, 50% of the computation time can be used by the floating-point processor. Hence, $F_{enh} = 0.5$. The speedup of the floating-point processor, in this case, is 100% faster. Hence, $S_{enh} = 2$. Thus, according to Amdahl's Law,

$$\begin{aligned}\text{Overall Speedup} &= \frac{1}{(1 - F_{enh}) + \frac{F_{enh}}{S_{enh}}} \\ &= \frac{1}{(1 - 0.5) + \frac{0.5}{2}} \\ &= \frac{1}{0.5 + 0.25} \\ &= 1.33\end{aligned}$$

**Cost/Speedup** $= 60/1.33 = 45.11$

Therefore, **Option 1** is better because it has a smaller **Cost/Speedup** ratio.

4. The 5 stage pipelined execution of the two programs are illustrated in the following two figures

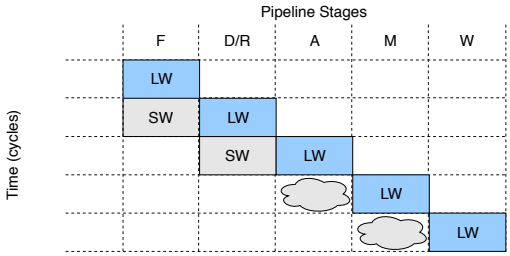


Figure 1: 5-stage pipelined execution of Program 1

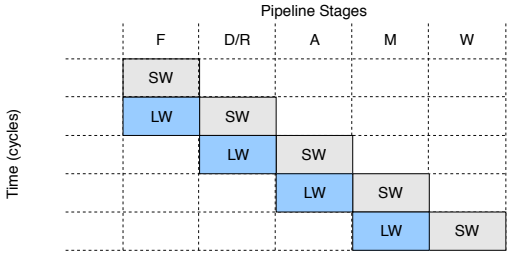


Figure 2: 5-stage pipelined execution of Program 2

In a 5 stage pipelined processor, source register access is done at the `D/R` (Decode/Register Access) stage. It can be observed that, in case of Program 1, the `SW` instruction has to wait until the `LW` instruction has been completed the write-back stage. This is because the `SW` instruction's `D/R` stage (Decode/Register access) requires the updated `R1` value, but `R1` is updated only at the `W` (Write Back stage) of the `LW` instruction. Hence, `SW` has to stall the pipeline until `R1` has been updated. This introduces bubbles in the pipeline and incurs additional clock cycles to complete the instructions. However, in case of Program 2, the `SW` instruction accesses `R1` register at `D/R` stage, and `LW` instruction accesses `R1` at `W` stage, which comes later. Therefore, pipeline stalling is not required in this case. Thus, it requires less number of clock cycles than Program 1 implying that Program 2 would execute faster.