DUE: Do NOT hand in

READ:
i) Related portions of Chapter 4 (except Section 4.4)
ii) Related portions of Appendix B
iii) Related portions of Appendix C
iv) Related portions of Appendix D
v) Related portions of Chapter 1

ASSIGNMENT: There are six questions.

Solve all homework and exam problems as shown in class and past exam solutions.

In calculating figures, you may have to convert percentages to fractions. Or, you may have to convert fractions to percentages.

1) A program is run on machines M1 and M2 and the following execution statistics is obtained:

<table>
<thead>
<tr>
<th>Time on Machine 1</th>
<th>Time on Machine 2</th>
<th>Instructions executed on M1</th>
<th>Instructions executed on M2</th>
<th>Clock frequency of M1</th>
<th>Clock frequency of M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 seconds</td>
<td>1.5 seconds</td>
<td>$5 \times 10^9$</td>
<td>$6 \times 10^9$</td>
<td>4 GHz</td>
<td>6 GHz</td>
</tr>
</tbody>
</table>

Calculate the following:

i) The $\text{MIPS}_{\text{ave}}$ figures for the program on Machine M1 and on Machine M2.

ii) The $\text{CPI}_{\text{ave}}$ figures for the two machines when they run the program.

2) Solve Problem 1.14.1 of Chapter I.

You are asked to compute $\text{CPUtime}$ figures for the program on each processor. That is, you will obtain two $\text{CPUtime}$ figures to compare the performance of the two processors.

Note that $\text{CPI}_i$ figures are very low. This is because $\text{pipelining}$ is used to speed up the execution!
3) Solve Problem 1.14.4 of Chapter I.

You are asked to compute \( \text{MFLOPS}_{\text{ave}} \) figures for the program on each machine. That is, you will obtain two \( \text{MFLOPS}_{\text{ave}} \) figures to compare the performance of two machines.

To calculate the MFLOPS figures, you will calculate the FP execution time figures for the two programs. This is possible since the question specifies the CPI_{FP} figures for the machine.

4) Solve Problem 1.14.5 of Chapter I.

You are asked to compute \( \text{MIPS}_{\text{ave}} \) figures for the program on each machine. That is, you will obtain two \( \text{MIPS}_{\text{ave}} \) figures to compare the performance of two machines.

Note that to calculate the MIPS figures, you will also calculate the CPUtime figures for the two programs. This is possible since the question specifies separate CPI figures for different types of instructions for the machine.

5) Solve Problem 1.14.6 of Chapter I.

To solve the question you will use CPUtime to calculate the Performance where \( 1/\text{CPUtime} \) is the Performance. The question is asking you to mention if there is any correlation between Performance and MFLOPS and any correlation between Performance and MIPS.

6) Assume that a new processor is developed. The design of the processor and compiler are completed, and a decision has to be made whether to produce the current processor design as it stands or spend additional time to improve it. There are two options:

Option a:
Leave the design as it stands. Let’s call this base computer, Mbase. It has a clock rate of 500 MHz, and the following measurements have been made using a simulator:

<table>
<thead>
<tr>
<th>Instruction Class</th>
<th>CPI (_i)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>40%</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>25%</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>25%</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>10%</td>
</tr>
</tbody>
</table>
Option b:
Optimize the hardware so that the clock rate is increased to 600 MHz. Let’s call this computer, Mopt. The following measurements were made using a simulator for Mopt:

<table>
<thead>
<tr>
<th>Instruction Class</th>
<th>CPI_i</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>40%</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>25%</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>25%</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>10%</td>
</tr>
</tbody>
</table>

i) What is the CPI_{ave} for each computer?

The CPI_{ave} is calculated by using:

$$\text{CPI}_{ave} = \sum_{i=1}^{n} CPI_i \times \text{Fraction}_i$$

Fraction_i is the conversion of Frequency_i to a fraction. For example, if the frequency is 40%, then the fraction is 0.4.

ii) What is MIPS_{ave} for each computer?

iii) Which computer is better? Why?

Let’s try to visualize what the designers are going through as they think about the options:

According to the tables, instructions in class A are the most common and must immediately be targeted for further hardware optimization. However, Class A instructions already take a very short time to run: Two clock periods. It would be too expensive to shorten their run time further.

The next candidates are Class B and C instructions which are more common than Class D instructions to improve the run time. It turns out that it is the case....

Note also that the clock frequency of machine Mopt is increased which improves the run time of all instruction classes: A, B, C and D.
RELEVANT QUESTIONS AND ANSWERS

Q1) The following piece of mnemonic machine language program is run on the EMY computer:

```
400100  LW  R8, 0(R9)
400104  ADDI R8, R8, 1
400108  SW  R8, 0(R9)
40010C  ADDI R9, R9, 4
400110  ??? # Decrement R10 by 1
400114  ??? # If R10 is not equal to zero, branch to location 400100
```

a) Based on the comments made, specify the last two instructions precisely.

b) The clock frequency is 200 MHz. Initially for the program, registers R9 and R10 contain 10005000 and (100)\(_{10}\), respectively. The CPI\(_i\) of each instruction is as follows:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Traced states and CPI(_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>0, 1, 2, 3, 4 =&gt; 5</td>
</tr>
<tr>
<td>SW</td>
<td>0, 1, 2, 5 =&gt; 4</td>
</tr>
<tr>
<td>ADDI</td>
<td>0, 1, 16, 17 =&gt; 4</td>
</tr>
<tr>
<td>M[110]</td>
<td>? =&gt; 4</td>
</tr>
<tr>
<td>M[114]</td>
<td>? =&gt; 3</td>
</tr>
</tbody>
</table>

Calculate how long it will take to run the above code. Also, calculate : the CPI\(_{ave}\) and the MIPS\(_{ave}\)

c) Assume that **clock doubling** is used. Repeat part (b)

A1) a)

```
400110  ADDI R10, R10, (-1)\(_{10}\) # Decrement R10 by 1
400114  BNE R10, R0, (-6)\(_{10}\)  # If R10 is not equal to zero, branch to location 400100
```

b) There is a loop which is executed “R10” times or (100)\(_{10}\) times. The loop contains 6 instructions. Therefore, the number of instructions executed is 600.

We can calculate the number of clock periods for the program then:

\[
\text{Number of clock periods} = \sum_{i=1}^{6} \text{CPI}_i \times L_i = (100 \times 5) + (100 \times 4) + (100 \times 4) + (100 \times 4) + (100 \times 3) = 2,400
\]

The time to execute the code is the number of clock periods for the code multiplied by the clock period duration:

\[
\text{clock period} = \frac{1}{200 \times 10^6} = 5 \times 10^{-9} = 5\text{ns}
\]

Time to execute the code = The number of clock periods for the code * The clock period duration

\[
= 2400 \times 5 \times 10^{-9} = 12000 \times 10^{-9} = 12 \times 10^3 \times 10^{-9} = 12 \times 10^{-6} = 12\mu\text{seconds}
\]
The new instruction execution timings in terms of clock periods are as follows:

\[
\begin{align*}
\text{CPI}_{\text{ave}} &= \frac{\text{number of clock periods}}{\text{NI}} = \frac{2400}{600} = 4 \\
\text{MIPS}_{\text{ave}} &= \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{600}{12 \times 10^{-6} \times 10^6} = \frac{600}{12} = 50
\end{align*}
\]

c) The new instruction execution timings in terms of clock periods are as follows:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Traced states and CPI\textsubscript{i}</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>0*, 1, 2, 3*, 4 =&gt; 7</td>
</tr>
<tr>
<td>SW</td>
<td>0*, 1, 2, 5* =&gt; 6</td>
</tr>
<tr>
<td>ADDI</td>
<td>0*, 1, 16, 17 =&gt; 5</td>
</tr>
<tr>
<td>BNE</td>
<td>0*, 1, 8 =&gt; 4</td>
</tr>
</tbody>
</table>

The new number of clock periods for the program is then:

\[
\text{Number of clock periods} = \sum_{i=1}^{6} \text{CPI}_i \times I_i = (100 \times 7) + (100 \times 5) + (100 \times 6) + (100 \times 5) + (100 \times 5) + (100 \times 4) = 3,200
\]

The time to execute the code is then:

\[
\text{clock period} = \frac{1}{400 \times 10^6} = 2.5 \times 10^{-9} = 2.5\text{ns}
\]

Time to execute the code = The number of clock periods for the code * The clock period duration

\[
= 3200 \times 2.5 \times 10^{-9} = 8000 \times 10^{-9} \text{seconds} = 8 \times 10^3 \times 10^{-9} \text{seconds} = 8 \times 10^{-6} \text{seconds} = 8\mu\text{seconds}
\]

\[
\text{CPI}_{\text{ave}} = \frac{\text{number of clock periods}}{\text{NI}} = \frac{3200}{600} = 5.33
\]

\[
\text{MIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{600}{8 \times 10^{-6} \times 10^6} = \frac{600}{8} = 75
\]

Q2) A CPU clock frequency is 100 MHz. A program is run on EMY. Then, the same program is run on the same CPU with clock doubling. For both types of the CPU, calculate:

a) Number of clock periods the program takes,

b) The CPU time,

c) The CPI\textsubscript{ave},

d) The MIPS\textsubscript{ave}.
The following table contains the statistics:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Number of times run</th>
<th>$CPI_i$</th>
<th>$CPI_i$ (clock doubling)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>2000</td>
<td>0, 1, 2, 3, 4 =&gt; 5</td>
<td>0*, 1, 2, 3*, 4 =&gt; 7</td>
</tr>
<tr>
<td>SW</td>
<td>500</td>
<td>0, 1, 2, 5 =&gt; 4</td>
<td>0*, 1, 2, 5* =&gt; 6</td>
</tr>
<tr>
<td>ADD</td>
<td>12000</td>
<td>0, 1, 16, 17 =&gt; 4</td>
<td>0*, 1, 16, 17 =&gt; 5</td>
</tr>
<tr>
<td>SUB</td>
<td>4500</td>
<td>0, 1, 2, 5 =&gt; 4</td>
<td>0*, 1, 2, 5* =&gt; 5</td>
</tr>
<tr>
<td>BEQ</td>
<td>140</td>
<td>0, 1, 8 =&gt; 3</td>
<td>0*, 1, 8 =&gt; 4</td>
</tr>
<tr>
<td>J</td>
<td>50</td>
<td>0, 1, 16 =&gt; 3</td>
<td>0*, 1, 16 =&gt; 4</td>
</tr>
</tbody>
</table>

A2) a) 

$$\text{number of clock periods} = \sum_{i=1}^{6} N_{I_i} CPI_i = (2000 \times 5) + (500 \times 4) + (12000 \times 4) + (4500 \times 4) + (140 \times 3) + (50 \times 3) = 78,570$$

b) 

$$\text{clock period} = \frac{1}{100 \times 10^6} = 10 \times 10^{-9} = 10 \text{ns}$$

$$\text{CPU time} = \text{number of clock periods} \times \text{clock period} = 78570 \times 10 = 785700 \text{ns} = 785.7 \mu s$$

c) 

$$\text{NI} = \sum_{i=1}^{6} N_{I_i} = 2000 + 500 + 12000 + 4500 + 140 + 50 = 19,190$$

$$\text{CPI ave} = \frac{\text{number of clock periods}}{\text{NI}} = \frac{78570}{19190} = 4.09$$

$$\text{CPI ave} = \frac{\text{number of clock periods}}{\text{NI}} = \frac{100260}{19190} = 5.22$$

d) 

$$\text{MIPS ave} = \frac{\text{NI}}{\text{CPU time} \times 10^6} = \frac{19190}{785.7 \times 10^{-6} \times 10^6} = \frac{19190}{785.7} = 24.42$$

$$\text{MIPS ave} = \frac{\text{NI}}{\text{CPU time} \times 10^6} = \frac{19190}{501.3 \times 10^{-6} \times 10^6} = \frac{19190}{501.3} = 38.28$$
Q3) The following piece of program is run on the EMY computer:

```
400500  I1  # Instruction I1 is executed 5000 times. Its CPIi is asked below
400504  I2  # Number of times instruction I2 is executed is asked below. The CPIi is 4.
400508  I3  # Instruction I3 is executed 5000 times. The CPIi is 3
40050C  I4  # Instruction I4 is executed once. The CPIi is 3
```

The clock period is 100MHz. You are given the following exact measurements:
- NI = 15001
- CPU time = 0.55003 * 10^-3 seconds

Calculate:

i) the number of times I2 is executed,
ii) the number of clock periods the program takes,
iii) the CPIi of I1,
iv) the CPIave,
v) the MIPSave.

A3) i) NI = 15001 = NI1 + NI2 + NI3 + NI4 = 5000 + NI2 + 5000 + 1 => NI2 = 15001 - 10001 = 5000

ii)
```
clock period = \frac{1}{100 \times 10^{-6}} = 10 \times 10^{-9} = 10ns
```

```
0.55003 \times 10^{-3} = \text{number of clock periods} \times 10^{-8}
```

CPUtime = number of clock periods \times clock period
```
\text{number of clock periods} = \frac{0.55003 \times 10^{-3}}{10^{-8}} = 0.55003 \times 10^{5} = 55003
```

iii) number of clock periods = 55003 = 5000*CPII1 + 5000*4 + 5000*3 + 1*3
```
CPII1 = \frac{[55003 - (20000 + 15000 + 3)]}{5000} = 20000/5000 = 4
```

iv) 
```
CPIave = \frac{\text{number of clock periods}}{NI} = \frac{55003}{15001} = 3.67
```

v) 
```
MIPSave = \frac{NI}{\text{CPUtime} \times 10^{6}} = \frac{15001}{550.03 \times 10^{-6} \times 10^{6}} = \frac{15001}{550.03} = 27.27
```

Q4) A program is run on the EMY computer and the following set of statistics is obtained:
- The total number of memory accesses for the program: 150 million
- The number of memory accesses for data: 50 million
- The fraction of Load instructions run: 0.4
- The fraction of integer A/L instructions run: 0.4
- The MFLOPSave = 0.0
- The clock frequency = 500 MHz

Obtain as many figures as you can from the given set of statistics.
A4) The total number of memory accesses for the program is the sum of the number of memory accesses for instructions and the number of memory accesses for data. The number of memory accesses for instructions is the number of instructions run for the program (NI):

\[ 150M = NI + 50M \Rightarrow NI = 100M \]

\[
\text{clock period} = \frac{1}{\text{clock frequency}} = \frac{1}{500 \times 10^6} = 2 \times 10^{-9} \text{ seconds} = 2\text{ ns}
\]

The number of L/S instructions run = the number of memory accesses for data = 50M
The number of Load instructions run = 0.4*NI = 0.4*100M = 40M
The number of Store instructions run = 50M - 40M = 10M
The number of A/L instructions run = 0.4*NI = 0.4*100M = 40M
The number of FP instructions run = 0 since MFLOP_{ave} = 0.0
The number of control instructions run = NI - #A/L - #L/S = 100M - 40M - 50M = 10M
The number of clock periods for the program = #A/L*CPI_{A/L} + #L*CPI_{L} + #S*CPI_{S} + #control*CPI_{control}

\[ = 40M*4 + 40M*5 + 10M*4 + 10M*3 = 160M + 200M + 40M + 30M = 430M \]

\[ \text{CPUtime} = \frac{\text{#clock periods for program} \times \text{clock period}}{\text{clock frequency}} = 430 \times 10^6 \times 2 \times 10^{-9} = 0.860 \text{ sec} \]

\[ \text{CPI}_{ave} = \frac{\text{#clock periods for program}}{ NI } = \frac{430 \times 10^6}{100 \times 10^6} = 4.3 \]

\[ \text{MIPS}_{ave} = \frac{NI}{\text{CPUtime} \times 10^6} = \frac{100 \times 10^6}{0.860 \times 10^6} = 116.28 \]

Q5) A benchmark suit is run on EMY and the following execution time statistics is obtained:

- A/L instruction frequency : 35% with CPI_i = 4.4
- data transfer instruction frequency : 25% with CPI_i = 4.8
- control instruction frequency : 15% with CPI_i = 3.3
- FP instruction frequency : 25% with CPI_i = 7.2
- Time to execute all FP instructions : 6 seconds
- clock frequency = 200MHz

Calculate : CPI_{ave}, CPUtime, MIPS_{ave}, MFLOPS_{ave}, total number of memory accesses made, the number of memory accesses for data only.

A5) 

\[ \text{CPI}_{ave} = \sum_{i=1}^{4} CPI_i \times I_i = (4.4 \times 0.35) + (4.8 \times 0.25) + (3.3 \times 0.15) + (7.2 \times 0.25) = 5.035 \]

\[ \text{#clock periods for FP instructions} = \text{time for FP instructions} \times \text{clock frequency} = 6 \times 200 \times 10^6 = 1.2 \times 10^9 \]
It is given that 25% of all instructions run are FP instructions and each is a FP operation. Thus, 166.7*10^6 FP instructions are run. Then,

\[
\text{NI} = \frac{100 \times 166.7 \times 10^6}{25} = 667 \times 10^6
\]

CPU\text{time} = \frac{\text{NI} \times \text{CPI}_{\text{ave}}}{\text{clock frequency}} = \frac{667 \times 10^6 \times 5.035}{200 \times 10^6} = 16.79\text{seconds}

MIPS_{\text{ave}} = \frac{\text{NI}}{\text{CPU\text{time}} \times 10^6} = \frac{667 \times 10^6}{16.79 \times 10^6} = 39.72

MFLOPS_{\text{ave}} = \frac{\text{#FP operations}}{\text{FP\text{time}} \times 10^6} = \frac{166.7 \times 10^6}{6 \times 10^6} = 27.78

The percentage of data transfer instructions that generate memory data accesses is 25. Then,

\[
\text{#data transfer instructions} = \frac{25 \times 667 \times 10^6}{100} = 166.7 \times 10^6
\]

A data transfer instruction generates only one data access on the memory. Therefore, there are 166.7*10^6 memory accesses for data.

Finally, the total number of memory accesses is the sum of the memory accesses for instruction fetches and the memory data accesses. Since there are 667*10^6 instructions executed, total number of memory accesses = 667*10^6 + 166.7*10^6 = 833.7*10^6

Q6) Assume that a program is run on EMy. It is observed that for this program 10 billion (10^{10}) instructions are executed and CPI_{\text{ave}} = 6.37. 30% of all instructions run are Load and Store, 40% of all instructions run are single-precision floating-point and 30% of all instructions run are Integer A/L instructions. The clock frequency is 3 GHz.

Calculate the following figures :

i) CPU\text{time}

ii) MIPS_{\text{ave}}

iii) MFLOPS_{\text{ave}}

iv) Total number of all memory accesses made for the program

A6) i) The CPU\text{time} : First, we have to obtain the clock period : clock period = \frac{1}{3 \times 10^9} = 0.33 \times 10^{-9} = 1\text{ns}

CPU\text{time} = \text{NI} \times \text{CPI}_{\text{ave}} \times \text{clock period duration} = 10^{10} \times 6.37 \times 0.33 \times 10^{-9} = 21.23\text{seconds}
ii) The \(\text{MIPS}_{\text{ave}}\) for the program:

\[
\text{MIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{10^{10}}{21.23 \times 10^{-9} \times 10^6} = 471.03
\]

iii) For the \(\text{MFLOPS}_{\text{ave}}\), we first calculate the number of FP operations for the program. It is given that 40% of all instructions are FP. Then, the number of FP instructions run is \(4 \times 10^{10} = 4 \times 10^9\). The \(\text{MFLOPS}_{\text{ave}}\) for the program:

\[
\text{MFLOPS}_{\text{ave}} = \frac{\text{Number of FP operations}}{\text{CPUtime} \times 10^6} = \frac{4 \times 10^9}{21.23 \times 10^{-9} \times 10^6} = 188.41
\]

iv) We know that every instruction requires a memory access to fetch the instruction. In addition, Load and Store instructions require a memory access for data. It is given that 30% of instructions executed are Load and Store instructions. Then, the total number of memory accesses is calculated by adding the number of instruction fetches and the number of Load and Store instructions requiring data accesses: \(10^{10} + 0.3 \times 10^{10} = 1.3 \times 10^{10}\).

Q7) Consider the following piece of EMY mnemonic machine language program:

\[
\begin{align*}
400000 & \text{ LW} \ R10, \ 0(R8) \quad \# \ R8 \ \text{points at array A and initially has 10000000} \\
400004 & \text{ LW} \ R11, \ 0(R9) \quad \# \ R9 \ \text{points at array B and initially has 10002000} \\
400008 & \text{ SLT} \ R12, \ R10, \ R11 \quad \# \ \text{Compare R10 and R11} \\
400010 & \text{ SW} \ R11, \ 0(R8) \quad \# \ \text{Store the 2nd number in the first} \\
400014 & \text{ ADDI} \ R8, \ R8, \ 4 \quad \# \ \text{Update the array A pointer} \\
400018 & \text{ ADDI} \ R9, \ R9, \ 4 \quad \# \ \text{Update the array B pointer} \\
40001C & \text{ ADDI} \ R13, \ R13, \ (-1)_{10} \quad \# \ \text{The loop-end counter, R13, has 2 initially} \\
400020 & \text{ BNE} \ R13, \ R0, \ (-9)_{10} \quad \# \ \text{If not the end, go back to 400000} \\
\end{align*}
\]

i) If the clock frequency is 1 GHz, determine how long it takes to run the above piece of program as done in class? Note that each memory access takes one clock period.

ii) Assume that a new machine language instruction is created to perform the above program faster, SLTM:

\[
\text{SLTM} \ Rd, (Rs), (Rt) \quad \# \ \text{If M[Rs] < M[Rt] then Rd} \quad 1, \ \text{else Rd} \quad 0
\]

CPI_{SLTM} is 6 since we trace states 0, 1, 16, 17, 18 and 19

The above program is rewritten by using the SLTM instruction as follows:

\[
\begin{align*}
400000 & \text{ SLTM} \ R12, \ (R8), \ (R9) \quad \# \ \text{Compare array A and B elements} \\
400004 & \text{ BEQ} \ R12, \ R0, \ 2 \quad \# \ \text{Skip next instruction if array A element is not less than array B element} \\
400008 & \text{ LW} \ R11, \ 0(R9) \quad \# \ \text{Read array B element} \\
40000C & \text{ SW} \ R11, \ 0(R8) \quad \# \ \text{Store array B element in array A} \\
400010 & \text{ ADDI} \ R8, \ R8, \ 4 \quad \# \ \text{Update the array A pointer} \\
400014 & \text{ ADDI} \ R9, \ R9, \ 4 \quad \# \ \text{Update the array B pointer} \\
400018 & \text{ ADDI} \ R13, \ R13, \ (-1)_{10} \quad \# \ \text{The loop-end counter, R13, has 2 initially} \\
40001C & \text{ BNE} \ R13, \ R0, \ (-9)_{10} \quad \# \ \text{If not the end, go back to 400000} \\
\end{align*}
\]

The SLTM instruction replaces the first three instructions.

Determine the new execution time of the program as done in class. Again, assume that the clock frequency is 1 GHz and each memory access takes one clock period.
A7) i) The loop compares arrays A and B. If an array A element is less than the corresponding array B element, the array A element is replaced with the array B element by executing a SW. Out of two comparisons (iterations), only one requires the execution of the SW instruction. Therefore, the following instructions are run: LW, LW, SLT, BEQ, SW, ADDI, ADDI, ADDI, BNE, LW, LW, SLT, BEQ, ADDI, ADDI, ADDI, BNE.

The execution timings of the six different instructions in the loop are as follows:

- LW : 0, 1, 2, 3, 4 => 5 cp
- SLT : 0, 1, 6, 7 => 4 cp
- BEQ : 0, 1, 8 => 3 cp
- SW : 0, 1, 2, 5 => 4 cp
- ADDI : 0, 1, 2, 16 => 4 cp
- BNE : 0, 1, 16 => 3 cp

The number of clock periods to run the 17 instructions is 5 + 5 + 4 + 3 + 4 + 4 + 3 + 3 + 5 + 5 + 4 + 4 + 4 + 4 + 3 = 68 clock periods.

\[
\text{clock period} = \frac{1}{1 \times 10^9} = 1 \times 10^{-9} \text{sec} = 1 \text{ns}
\]

The execution time is 68 * 1 = 68ns

ii) We execute the following 14 instructions : SLTM, BEQ, LW, SW, ADDI, ADDI, ADDI, BNE, SLTM, BEQ, ADDI, ADDI, ADDI, BNE.

The number of clock periods to run the 14 instructions is 6 + 3 + 5 + 4 + 4 + 4 + 3 + 6 + 3 + 4 + 4 + 4 + 3 = 57 clock periods. Then, the execution time is 57 * 1 = 57ns. The speedup is 68/57 = 1.19 or 19%

Compared with the old code, the new code has one less instruction, not two. An extra LW instruction is needed to bring the array B element to the CPU to write to array A even if the SLTM instruction brings the same element to the CPU. This is because the SLTM instruction does not keep the element in an architectural register that can be used by other instructions.

In general, if data read from the memory is used again, it must be kept on architectural registers as long as possible. The RISC concept of having only LW and SW to access the memory for data supports that idea. CISC A/L instructions accessing memory for data keep the data in organizational registers, not visible to the machine language programmer. It means if the same data element is needed an additional instruction is needed!

Q8) Assume that a program is run on Emy. The following table contains the statistics of the execution:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Number of times run</th>
<th>CPI_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>1M</td>
<td>5</td>
</tr>
<tr>
<td>SW</td>
<td>0.25M</td>
<td>4</td>
</tr>
<tr>
<td>ADD</td>
<td>2M</td>
<td>4</td>
</tr>
<tr>
<td>BEQ</td>
<td>0.25M</td>
<td>3</td>
</tr>
<tr>
<td>FPADD</td>
<td>10M</td>
<td>7</td>
</tr>
<tr>
<td>FPMUL</td>
<td>10M</td>
<td>11</td>
</tr>
<tr>
<td>FPDIV</td>
<td>10M</td>
<td>20</td>
</tr>
</tbody>
</table>

The clock frequency is 2 GHz.

a) Calculate the following figures:

i) The CPU time
ii) The CPI_{ave}
iii) The MIPS_{ave}
iv) The MFLOPS_{ave}

b) Assume that we decide to improve the execution of FPDI

V instructions: we execute a FPDIV instruction in 10 clock periods. Calculate the Speedup_{overall} figure.
A8) clock period = \( \frac{1}{2 \times 10^9} = 0.5 \times 10^{-9} = 0.5 \text{ns} \)

a) i) First, we need to get the number of clock periods the program takes:

number of clock periods for the program = \( \sum_{i=1}^{7} I_i \cdot CPI_i \)

= \((1 \text{M} \times 5) + (0.25 \text{M} \times 4) + (2 \text{M} \times 4) + (0.25 \times 3) + (10 \text{M} \times 7) + (10 \text{M} \times 11) + (10 \text{M} \times 20)\) = \(394.75 \times 10^6\)

The CPU time can now be calculated:

CPU time = clock periods for the program \times clock period duration = \(394.75 \times 10^6 \times 0.5 \times 10^{-9} = 0.197375 \text{ seconds} \)

time ii) The CPIave:

\( \frac{\text{NI}}{\text{CPIave}} = \frac{1 \text{M} + 0.25 \text{M} + 2 \text{M} + 0.25 \text{M} + 10 \text{M} + 10 \text{M} + 10 \text{M}}{7} = 33.5 \times 10^6 \)

CPIave = \(\frac{\text{number of clock periods for the program}}{\text{NI}}\) = \(\frac{394.75 \times 10^6}{33.5 \times 10^6} = 11.78\)

iii) The MIPSave:

\(\frac{\text{MIPSave}}{\text{CPUtime} \times 10^6} = \frac{33.5 \times 10^6}{0.197375 \times 10^6} = 169.73\)

iv) The MFLOPSave:

Number of FP instructions = NI_{FPADD} + NI_{FPMUL} + NI_{FPDIV} = 10 \text{M} + 10 \text{M} + 10 \text{M} = 30 \times 10^6

FPclockperiods = NIFPADD \times CPI_{FPADD} + NIFPMUL \times CPI_{FPMUL} + NIFPDIV \times CPI_{FPDIV} = 10 \text{M} \times 7 + 10 \text{M} \times 11 + 10 \text{M} \times 20

FPclockperiods = 380M

FP_time = FPClockperiods \times = 380 \times 10^6 \times 0.5 \times 10^{-9} = 190 \times 10^{-3} = 0.19 \text{ seconds}

MFLOPSave = \(\frac{\text{Number of FP operations}}{\text{CPUtime} \times 10^6}\) = \(\frac{30 \times 10^6}{0.19 \times 10^6} = 155.89\)

b) The new number of clock periods for the programs is as follows:

number of clock periods for the program = \(\sum_{i=1}^{7} I_i \cdot CPI_i \)

= \((1 \text{M} \times 5) + (0.25 \text{M} \times 4) + (2 \text{M} \times 4) + 0.25 \times 43 + (10 \text{M} \times 7) + (10 \text{M} \times 11) + (10 \text{M} \times 10)\) = \(294.75 \times 10^6\)

The new CPU time is as follows:

CPUtime = Number of clock periods for the program \times clock period

= \(294.75 \times 10^6 \times 0.5 \times 10^{-9} = 0.147375 \text{ seconds} \)
Consider the following piece of EMY mnemonic machine language program:

```
400000 ADDI R8, R0, 0    # R10 points at array A and initially has 10000000
400004 LW  R9, 0(R10)
400008 ADD  R8, R8, R9
40000C ADDI R10, R10, 4
400010 ADDI R11, R11, (-1)_{10}  # R11 is the loop-end counter
400014 BNE R11, R0, (-5)_{10}
400018 SW  R8, 0(R10)
```

Assume that the EMY CPU is unpipelined, as discussed in Chapter 5 of the textbook and there is a perfect memory, with no stalls. Assume also that R11 initially has (1000)_{10}.

The instruction execution timings for the above program are as follows: CPI_LW = 5, CPI_ADD = 4, CPI_ADDI = 4, CPI_BNE = 3 and CPI_SW = 4. Assume also that the clock frequency is 2 GHz.

Calculate the following figures:

i) CPI_{ave}  
ii) CPUtime  
iii) MIPS_{ave}  
iv) The total number of all memory accesses made for the program.

A9) We execute the following instructions for “k” iterations: ADDI + k(LW + ADD + ADDI + ADDI + BNE) + SW

i) “k” is given as (1000)_{10}. Then, the number of clock periods for the program based on the given CPIi figures is: 4 + 1000(5 + 4 + 4 + 4 + 3) + 4 = 20008.

The number of instructions executes is: 1 + 1000(1 + 1 + 1 + 1 + 1) + 1 = 5002

\[ \text{CPI}_{\text{ave}} = \frac{\text{Number of clock periods for the program}}{\text{NI}} = \frac{20008}{5002} = 4 \]

ii) The CPUtime: First, we have to obtain the clock period:

\[ \text{clock period} = \frac{1}{2 \times 10^9} = 0.5 \times 10^{-9} = 0.5\text{ns} \]

\[ \text{CPUtime} = \text{NI} \times \text{CPI}_{\text{ave}} \times \text{clock period duration} \]
\[ = 5002 \times 4 \times 0.5 \times 10^{-9} \text{ seconds} = 10004 \times 10^{-9} \text{ seconds} = 10.004\text{μs} \]

iii) The MIPS_{ave} for the program:

\[ \text{MIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^6} = \frac{5002}{10.004 \times 10^{-6} \times 10^6} = 500 \]

iv) For every instruction a memory access is performed. In addition, Load and Store instructions require a memory access for data. Then, the total number of memory accesses is calculated by adding the number of instruction fetches and the number of data accesses due to Load and Store instructions: \(1 + 1000(2 + 1 + 1 + 1 + 1) + 2 = 6003.\)
Q10) Consider the following piece of EMY mnemonic machine language program:

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Register(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400C00</td>
<td>LW</td>
<td>R9, 0(R8)</td>
<td># R8 points at array A and initially has 10EA0000</td>
</tr>
<tr>
<td>400C04</td>
<td>ADD</td>
<td>R9, R9, R9</td>
<td></td>
</tr>
<tr>
<td>400C08</td>
<td>ADDI</td>
<td>R10, R9, AC</td>
<td></td>
</tr>
<tr>
<td>400C0C</td>
<td>SW</td>
<td>R10, 0(R8)</td>
<td></td>
</tr>
<tr>
<td>400C10</td>
<td>ADDI</td>
<td>R8, R8, 4</td>
<td></td>
</tr>
<tr>
<td>400C14</td>
<td>ADDI</td>
<td>R11, R11, (-1)\text{_{10}}</td>
<td># R11 is the loop-end counter</td>
</tr>
<tr>
<td>400C18</td>
<td>BNE</td>
<td>R11, R0, (-7)\text{_{10}}</td>
<td></td>
</tr>
</tbody>
</table>

Assume that the EMY CPU is unpipelined, as discussed in Chapter 5 of the textbook and there is a perfect memory, with no stalls. Assume also that R11 initially has \((10,000)_{\text{10}}\).

The instruction execution timings for the above program are as follows: \(\text{CPI}_{\text{LW}} = 5\), \(\text{CPI}_{\text{ADD}} = 4\), \(\text{CPI}_{\text{ADDI}} = 4\), \(\text{CPI}_{\text{SW}} = 4\) and \(\text{CPI}_{\text{BNE}} = 3\). Assume also that the clock frequency is 2 GHz.

Calculate the following figures:

i) \(\text{CPI}_{\text{ave}}\)

ii) \(\text{CPUtime}\)

iii) \(\text{GIPS}_{\text{ave}}\)

iv) The total number of all memory accesses made for the program.

A10) We execute the following instructions for “k” iterations: \((\text{LW} + \text{ADD} + \text{ADDI} + \text{SW} + \text{ADDI} + \text{ADDI} + \text{BNE})\)

i) “k” is given as \((10,000)_{\text{10}}\). Then, the number of clock periods for the program based on the given CPIi figures is: \(10000(5 + 4 + 4 + 4 + 4 + 4 + 3) = 280000\).

The number of instructions executed is: \(10000(1 + 1 + 1 + 1 + 1) = 70000\)

\[
\text{CPI}_{\text{ave}} = \frac{\text{Number of clock periods for the program}}{\text{NI}} = \frac{280000}{70000} = 4
\]

ii) The CPUtime: First, we have to obtain the clock period:

\[
\text{clock period} = \frac{1}{2 \times 10^9} = 0.5 \times 10^{-9} = 0.5\text{ns}
\]

\[
\text{CPUtime} = \text{NI} \times \text{CPI}_{\text{ave}} \times \text{clock period duration}
\]

\[
= 70000 \times 4 \times 0.5 \times 10^{-9} \text{ seconds} = 140000 \times 10^{-9} \text{ seconds} = 140\mu\text{seconds}
\]

iii) The GIPS_{ave} for the program:

\[
\text{GIPS}_{\text{ave}} = \frac{\text{NI}}{\text{CPUtime} \times 10^9} = \frac{70000}{140000 \times 10^{-9} \times 10^9} = 0.5
\]

iv) We know that for every instruction a memory access is performed. In addition, Load and Store instructions require a memory access for data. Then, the total number of memory accesses is calculated by adding the number of instruction fetches and the number of data accesses due to Load and Store instructions: \(10000(2 + 1 + 1 + 2 + 1 + 1 + 1) = 90000\)