SOLUTION OF SELECTED PROBLEMS OF COMPUTER ORGANIZATION AND DESIGN

DAVID A. PATTERSON & JOHN L. HENNESY

ALDO NÚÑEZ TOVAR
aldo.nunez.2025@gmail.com

1 Computer Abstraction and Technology

Exercise 1.2
Consider the different configurations shown in the table

	Configuration	Resolution	Main Memory	Ethernet Network
a.	1	640 × 480	2 Gbytes	100 Mbps
	2	1280 × 1024	4 Gbytes	1 Gbps
b.	1	1024 × 768	2 Gbytes	100 Mbps
	2	2560 × 1600	4 Gbytes	1 Gbps

1.2.1 For a color display using `8 bits` for each of the primary colors (red, green, blue) per pixel, what should be the minimum size in bytes of the frame buffer to store a frame?
Solution:
The total number of pixels in a frame of `640x480` of resolution is: $640 \times 480 = 307,200$ pixels.

Each pixel consists of three colors: red, green and blue, and each color uses `8 bits` or `1 byte`. So, each pixel ocuppies `3 bytes`.

Then, to calculate the size in `bytes` of the `frame buffer`, we have to multiply the total number of pixels by three:

$$307,200 \times 3 = 921,600 \text{ bytes/frame}$$

Resolution	Frame buffer size (bytes/frame)
640x480	921,600
1280x1024	3,932,160
1024x768	2,359,296
2560x1600	12,288,000

1.2.2 How many frames could it store, assuming the memory contains no other information?
Solution:
1 Kbyte = 1024 bytes
1 Gbyte = 1024^3 bytes = 1,073,741,824 bytes $\approx 1 \times 10^9$ = 1 Gbyte

So, for a computer with 2 Gbytes, we have:
$2 \times 1,073,741,824 = 2,147,483,648$ bytes

Then, if every size of frame is 921,600 bytes/frame, the number of frames that can be stored is:

$$\frac{2,147,483,648 \text{ bytes}}{921,600 \text{ bytes/frame}} = 2,330.6 \text{ frames} \approx 2,330 \text{ frames}$$

The computer can store a total of 2,330 frames in its 2 GB RAM.

Resolution	Frame buffer size bytes/frame	Main Memory	Number of frames
640x480	921,600	2 GBytes	2,330
1280x1024	3,932,160	4 GBytes	1,092
1024x768	2,359,296	2 GBytes	910
2560x1600	12,288,000	4 GBytes	349

1.2.3 If a 256 Kbytes file is sent through the Ethernet connection, how long it would take?
Solution:
256 Kbytes = 256 x 1024 = 252,144 bytes
1 byte = 8 bits
8 x 252,144 = 2,097,152 bits
The size file in bits is: 2,097,152 bits
The formula to calculate the transmission time is:

$$\text{Time} = \frac{\text{Size}}{\text{Rate}}$$

For an Ethernet network of 100 Mbps

$$\text{Time} = \frac{2,097,152}{100 \cdot 10^6} = 0.02097152 \text{ s}$$

The time to sent the file is: 0.02097152 = 20.97 ms

Ethernet Network	Size file	Time
100 Mbps	2,097,152 bits	20.97 ms
1 Gbps	2,097,152 bits	2.097 ms

This calculation does not include the overhead that Ethernet includes in its operation.

1.1 *Performance*

Performance and Execution time

$$\texttt{Performance} = \frac{1}{\texttt{Execution time}} \tag{1}$$

Relative performance and Execution time

$$\frac{\texttt{Performance}_X}{\texttt{Performance}_Y} = \frac{\texttt{Execution time}_Y}{\texttt{Execution time}_X} = n, \quad n \geq 1 \tag{2}$$

CPI - Cycles Per Instructions

$$\texttt{CPI} = \frac{N_{cycles}}{N_{instr}} \tag{3}$$

CPU Performance Equation

$$T = N_{instr} \times \texttt{CPI} \times t_{cycle} = \frac{N_{instr} \times \texttt{CPI}}{f} \tag{4}$$

CPU clock cycles

$$N_{cycles} = \sum_{i=1}^{n} (\texttt{CPI}_i \times C_i) \tag{5}$$

MIPS

$$\texttt{MIPS} = \frac{N_{instr}}{T \times 10^6} = \frac{f}{\texttt{CPI} \times 10^6} \tag{6}$$

- T - CPU time, [s]
- N_{instr} - number of instructions executed (instructions count), [instr]
- CPI - Cycles per Instruction, [cycles/instr]
- C_i - Number of instructions type i executed in the program. [instr]
- t_{cycle} - duration of clock cycle, [s/cycles]
- f - clock frequency, [Hertz] = [cycles/s]
- n - Number of different instruction types in the program.

Exercise 1.3

Consider three different processors P1, P2, and P3 executing the same instruction set with the clock rates and CPIs given in the following table.

	Processor	Clock Rate	CPI
a.	P1	3 GHz	1.5
	P2	2.5 GHz	1.0
	P3	4 GHz	2.2
b.	P1	2 GHz	1.2
	P2	3 GHz	0.8
	P3	4 GHz	2.0

1.3.1 Which processor has the highest performance expressed in instructions per second?
Solution:

From (4) and (2) we have:

$$T = \frac{N_{instr} \times CPI}{f}$$
$$\frac{Performance_X}{Performance_Y} = \frac{Execution\ time_Y}{Execution\ time_X} = n$$

Performance is inversely proportional to execution time

a.

$$P1 : \frac{f_1}{CPI_1} = \frac{3 \cdot 10^9}{1.5} = 2 \cdot 10^9\ instr/s$$
$$P2 : \frac{f_2}{CPI_2} = \frac{2.5 \cdot 10^9}{1.0} = 2.5 \cdot 10^9\ instr/s$$
$$P3 : \frac{f_3}{CPI_3} = \frac{4 \cdot 10^9}{2.2} = 1.82 \cdot 10^9\ instr/s$$

Performance:

$$P2 > P1 > P3$$

b.

$$P1 : \frac{f_1}{CPI_1} = \frac{2 \cdot 10^9}{1.2} = 1.67 \cdot 10^9\ instr/s$$
$$P2 : \frac{f_2}{CPI_2} = \frac{3 \cdot 10^9}{0.8} = 3.75 \cdot 10^9\ instr/s$$
$$P3 : \frac{f_3}{CPI_3} = \frac{4 \cdot 10^9}{2.0} = 2.0 \cdot 10^9\ instr/s$$

Performance:

$$P2 > P3 > P1$$

1.3.2 If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.
Solution:

We have (3) and (4):

$$CPI = \frac{N_{cycles}}{N_{instr}}$$
$$T = \frac{N_{instr} \times CPI}{f}$$

From (3) and (4) we have:

$$N_{cycles} = CPI \times N_{instr}$$
$$N_{instr} = \frac{T \times f}{CPI}$$

Finally, we get the formula to calculate the number of cycles:

$$N_{cycles} = T \times f$$

To calculate the number of instructions, from(3) we have:

$$N_{instr} = \frac{N_{cycles}}{CPI}$$

a.

P1:

$$N_{Cycles} = 10 \times 3 \cdot 10^9$$
$$N_{Cycles} = 30 \cdot 10^9 \text{ cycles}$$
$$N_{instr} = \frac{30 \cdot 10^9}{1.5}$$
$$N_{instr} = 20 \cdot 10^9 \text{ instr}$$

P2:

$$N_{Cycles} = 10 \times 2.5 \cdot 10^9$$
$$N_{Cycles} = 25 \cdot 10^9 \text{ cycles}$$
$$N_{instr} = \frac{25 \cdot 10^9}{1.0}$$
$$N_{instr} = 25 \cdot 10^9 \text{ instr}$$

P3:

$$N_{Cycles} = 10 \times 4 \cdot 10^9$$
$$N_{Cycles} = 40 \cdot 10^9 \text{ cycles}$$
$$N_{instr} = \frac{40 \cdot 10^9}{2.2}$$
$$N_{instr} = 18.18 \cdot 10^9 \text{ instr}$$

b.

P1:

$$N_{Cycles} = 10 \times 2 \cdot 10^9$$
$$N_{Cycles} = 20 \cdot 10^9 \text{ cycles}$$
$$N_{instr} = \frac{20 \cdot 10^9}{1.2}$$
$$N_{instr} = 16.67 \cdot 10^9 \text{ instr}$$

P2:

$$N_{Cycles} = 10 \times 3 \cdot 10^9$$
$$N_{Cycles} = 30 \cdot 10^9 \text{ cycles}$$
$$N_{instr} = \frac{30 \cdot 10^9}{0.8}$$
$$N_{instr} = 37.5 \cdot 10^9 \text{ instr}$$

P3:

$$N_{Cycles} = 10 \times 4 \cdot 10^9$$
$$N_{Cycles} = 40 \cdot 10^9 \text{ cycles}$$
$$N_{instr} = \frac{40 \cdot 10^9}{2.0}$$
$$N_{instr} = 20 \cdot 10^9 \text{ instr}$$

Exercise 1.14
Section 1.8 cites as a pitfall the utilization of a subset of the performance equation as a performance metric. To illustrate this, consider the following data for the execution of a program in different processors.

	Processor	Clock Rate	CPI	No. Instru.
a.	P1	4 GHz	0.9	5.00E+6
	P2	3 GHz	0.75	1.00E+6
b.	P1	3 GHz	1.1	3.00E+6
	P2	2.5 GHz	1.0	0.50E+6

1.14.1 One usual fallacy is to consider the computer with the largest clock rate as having the largest performance. Check if this is true for P1 and P2.
Solution:

From (4)

$$T = \frac{N_{instr} \times CPI}{f}$$

a.

For P_1:

$$T_1 = \frac{5 \cdot 10^6 \times 0.9}{4 \cdot 10^9}$$
$$T_1 = 1.125 \text{ ms}$$

For P_2:

$$T_2 = \frac{1 \cdot 10^6 \times 0.75}{3 \cdot 10^9}$$
$$T_2 = 0.25 \text{ ms}$$

Performance:

$$n = \frac{T_1}{T_2} = \frac{1.125 \text{ ms}}{0.25 \text{ ms}}$$
$$n = 4.5$$

P_2 is 4.5 times faster than P_1, even though it has a lower clock rate.

b.

For P_1:

$$T_1 = \frac{3 \cdot 10^6 \times 1.1}{3 \cdot 10^9}$$
$$T_1 = 1.1 \text{ ms}$$

For P_2:

$$T_2 = \frac{0.5 \cdot 10^6 \times 1}{2.5 \cdot 10^9}$$
$$T_2 = 0.2 \text{ ms}$$

Performance:

$$n = \frac{T_1}{T_2} = \frac{1.1 \text{ ms}}{0.2 \text{ ms}}$$
$$n = 5.5$$

P_2 is 5.5 times faster than P_1, even though it has a lower clock rate.

1.14.2 Another fallacy is to consider that the processor executing the largest number of instructions will need a larger CPU time. Considering that processor P1 is executing a sequence of 10^6 instructions and that the CPI of processors P1 and P2 do not change, determine the number of instructions that P2 can execute in the same time that P1 needs to execute 10^6 instructions.
Solution:
a.

For P_1:

$$T_1 = \frac{1 \cdot 10^6 \times 0.9}{4 \cdot 10^9}$$
$$T_1 = 0.225 \text{ ms}$$

For P_2:

$$N_{intr_2} = \frac{T_1 \times f_2}{CPI_2} = \frac{0.225 \cdot 10^{-3} \times 3 \cdot 10^9}{0.75}$$
$$N_{intr_2} = 9.0 \cdot 10^5 \text{ intr}$$

P2 executes $9.0 \cdot 10^5$ instructions in the same time P1 executes $1 \cdot 10^6$ instructions.

b.

For P_1:

$$T_1 = \frac{1 \cdot 10^6 \times 1.1}{3 \cdot 10^9}$$
$$T_1 = 0.367 \text{ ms}$$

For P_2:

$$N_{intr_2} = \frac{T_1 \times f_2}{CPI_2} = \frac{0.367 \cdot 10^{-3} \times 2.5 \cdot 10^9}{1}$$
$$N_{intr_2} = 9.17 \cdot 10^5 \text{ intr}$$

P2 executes $9.17 \cdot 10^5$ instructions in the same time P1 executes $1 \cdot 10^6$ instructions.

1.14.3 A common fallacy is to use MIPS (millions of instructions per second) to compare the performance of two different processors, and consider that the processor with the largest MIPS has the largest performance. Check if this is true for P1 and P2.
Solution:
a.

For P_1:

$$MIPS_1 = \frac{5 \cdot 10^6}{1.125 \cdot 10^{-3} \times 10^6}$$
$$MIPS_1 = 4.44 \cdot 10^3 \text{ MIPS}$$

For P_2:

$$\text{MIPS}_2 = \frac{1 \cdot 10^6}{0.25 \cdot 10^{-3} \times 10^6}$$
$$\text{MIPS}_2 = 4 \cdot 10^3 \text{ MIPS}$$

Although the MIPS of P1 are higher than the MIPS of P2, the performance of P2 is 4.5 times that of P1, as calculated above.

b.

For P_1:

$$\text{MIPS}_1 = \frac{3 \cdot 10^6}{1.1 \cdot 10^{-3} \times 10^6}$$
$$\text{MIPS}_1 = 2.73 \cdot 10^3 \text{ MIPS}$$

For P_2:

$$\text{MIPS}_2 = \frac{0.5 \cdot 10^6}{0.2 \cdot 10^{-3} \times 10^6}$$
$$\text{MIPS}_2 = 2.5 \cdot 10^3 \text{ MIPS}$$

Although the MIPS of P1 are higher than the MIPS of P2, the performance of P2 is 5.5 times that of P1, as calculated above.

Exercise 1.4
Consider two different implementations of the same instruction set architecture. There are four classes of instructions, A, B, C, and D. The clock rate and CPI of each implementation are given in the following table.

		Clock Rate	CPI Class A	CPI Class B	CPI Class C	CPI Class D
a.	P1	2.5 GHz	1	2	3	3
	P2	3 GHz	2	2	2	2
b.	P1	2.5 GHz	2	1.5	2	1
	P2	3 GHz	1	2	1	1

1.4.1 Given a program with 10^6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which implementation is faster?
Solution:

From (4)

$$T = \frac{N_{instr} \times CPI}{f}$$

From (3), (4) and (5), we have:

$$T = \frac{\sum_{i=1}^{n} (CPI_i \times C_i)}{f}$$

Calculate the number of instructions:

- Class A : 10% = 10^5 instr
- Class B : 20% = $2 \cdot 10^5$ instr
- Class C : 50% = $5 \cdot 10^5$ instr
- Class D : 20% = $2 \cdot 10^5$ instr

a.
P1:

$$N_{cycles} = \sum_{i=1}^{4} (CP_i \times C_i) = 1 \times 1 \cdot 10^5 + 2 \times 2 \cdot 10^5 + 3 \times 5 \cdot 10^5 + 3 \times 2 \cdot 10^5$$

$$N_{cycles} = (1 + 4 + 15 + 6) \cdot 10^5$$

$$T_{P1} = \frac{26 \cdot 10^5}{2.5 \cdot 10^9}$$

$$T_{P1} = 1.04 \text{ ms}$$

P2:

$$N_{cycles} = \sum_{i=1}^{4} (CP_i \times C_i) = 2 \times 1 \cdot 10^5 + 2 \times 2 \cdot 10^5 + 2 \times 5 \cdot 10^5 + 2 \times 2 \cdot 10^5$$

$$N_{cycles} = (2 + 4 + 10 + 4) \cdot 10^5$$

$$T_{P2} = \frac{20 \cdot 10^5}{3 \cdot 10^9}$$

$$T_{P2} = 0.67 \text{ ms}$$

b.
P1:

$$N_{cycles} = \sum_{i=1}^{4} (CP_i \times C_i) = 2 \times 1 \cdot 10^5 + 1.5 \times 2 \cdot 10^5 + 2 \times 5 \cdot 10^5 + 1 \times 2 \cdot 10^5$$

$$N_{cycles} = (2 + 3 + 10 + 2) \cdot 10^5$$

$$T_{P1} = \frac{17 \cdot 10^5}{2.5 \cdot 10^9}$$

$$T_{P1} = 0.68 \text{ ms}$$

P2:

$$N_{cycles} = \sum_{i=1}^{4} (CP_i \times C_i) = 1 \times 1 \cdot 10^5 + 2 \times 2 \cdot 10^5 + 1 \times 5 \cdot 10^5 + 1 \times 2 \cdot 10^5$$

$$N_{cycles} = (1 + 4 + 5 + 2) \cdot 10^5$$

$$T_{P2} = \frac{12 \cdot 10^5}{3 \cdot 10^9}$$

$$T_{P2} = 0.4 \text{ ms}$$

1.4.2 What is the global CPI for each implementation?

Extra problems

E1.1 A program consist of `5,000` floating point instructions and `25,000` integer instructions. `Processor A` has clock rate of `2.0 GHz`. Floating point instructions take `7 cycles` and integer instructions take `1 cycle`. How long does it take for this processor to run the program?
Solution:

Time for floating point instructions:

$$\begin{aligned} N_{instr_{fp}} &= 5,000\ instr \\ CPI_{fp} &= 7\ cycles/instr \\ f &= 2\ GHz \end{aligned}$$

from (4)

$$\begin{aligned} T &= \frac{N_{instr} \times CPI}{f} \\ T_{fp} &= \frac{5,000\ instr \times 7\ cycles/instr}{2 \times 10^9 cycles/s} = 17.5 \times 10^{-6} s \\ T_{fp} &= 17.5\ \mu s \end{aligned}$$

Time for integer instructions:

$$\begin{aligned} N_{instr_{int}} &= 25,000\ instr \\ CPI_{int} &= 1\ cycles/instr \\ f &= 2\ GHz \end{aligned}$$

From (4)

$$\begin{aligned} T &= \frac{N_{instr} \times CPI}{f} \\ T_{int} &= \frac{25,000\ instr \times 1\ cycles/instr}{2 \times 10^9 cycles/s} = 12.5 \times 10^{-6} s \\ T_{int} &= 12.5\ \mu s \end{aligned}$$

So, the total time is:

$$\begin{aligned} T &= T_{fp} + T_{int} = 17.5\ \mu s + 12.5\ \mu s \\ T &= 30\ \mu s \end{aligned}$$

The processor takes $30\ \mu s$ to run the program.

E1.2 What is the average `CPI` for this processor for the given program?
Solution:

From (3), we have:

$$\texttt{CPI} = \frac{\texttt{N}_{\texttt{cycles}}}{\texttt{N}_{\texttt{instr}}}$$

Calculate the number of cycles for floating point instructions:

$$\begin{aligned}
\texttt{CPI}_{\texttt{fp}} &= 7\ \texttt{cycles/inst} \\
\texttt{N}_{\texttt{instr}_{\texttt{fp}}} &= 5,000\ \texttt{instr} \\
\texttt{N}_{\texttt{cycles}_{\texttt{fp}}} &= \texttt{CPI}_{\texttt{fp}} \times \texttt{N}_{\texttt{instr}_{\texttt{fp}}} = 7\ \texttt{cycles/instr} \times 5,000\ \texttt{instr} \\
\texttt{N}_{\texttt{cycles}_{\texttt{fp}}} &= 35,000\ \texttt{cycles}
\end{aligned}$$

Calculate the number of cycles for integer instructions:

$$\begin{aligned}
\texttt{CPI}_{\texttt{int}} &= 1\ \texttt{cycles/instr} \\
\texttt{N}_{\texttt{instr}_{\texttt{int}}} &= 25,000\ \texttt{instr} \\
\texttt{N}_{\texttt{cycles}_{\texttt{int}}} &= \texttt{CPI}_{\texttt{int}} \times \texttt{N}_{\texttt{instr}_{\texttt{int}}} = 1\ \texttt{cycles/instr} \times 25,000\ \texttt{instr} \\
\texttt{N}_{\texttt{cycles}_{\texttt{int}}} &= 25,000\ \texttt{cycles}
\end{aligned}$$

So, calculate the CPI average:

$$\begin{aligned}
\texttt{CPI} &= \frac{\texttt{N}_{\texttt{cycles}_{\texttt{fp}}} + \texttt{N}_{\texttt{cycles}_{\texttt{int}}}}{\texttt{N}_{\texttt{instr}_{\texttt{fp}}} + \texttt{N}_{\texttt{intr}_{\texttt{int}}}} = \frac{35,000\ \texttt{cycles} + 25,000\ \texttt{cycles}}{5,000\ \texttt{instr} + 25,000\ \texttt{instr}} \\
&= \frac{60,000\ \texttt{cycles}}{30,000\ \texttt{instr}} \\
\texttt{CPI} &= 2\ \texttt{cycles/instr}
\end{aligned}$$

The average CPI is: 2 cycles/instr.

E1.3 Processor A runs Program 2 consisting of 100,000 floating point instructions and 50,000 integer instructions. What is the average CPI for this program?
Solution:

$$\begin{aligned}
\texttt{N}_{\texttt{cycles}_{\texttt{fp}}} &= \texttt{CPI}_{\texttt{fp}} \times \texttt{N}_{\texttt{instr}_{\texttt{fp}}} = 7\ \texttt{cycles/instr} \times 100,000\ \texttt{instr} \\
\texttt{N}_{\texttt{cycles}_{\texttt{fp}}} &= 700,000\ \texttt{cycles} \\
\texttt{N}_{\texttt{cycles}_{\texttt{int}}} &= \texttt{CPI}_{\texttt{int}} \times \texttt{N}_{\texttt{instr}_{\texttt{int}}} = 1\ \texttt{cycles/instr} \times 50,000\ \texttt{instr} \\
\texttt{N}_{\texttt{cycles}_{\texttt{int}}} &= 50,000\ \texttt{cycles} \\
\texttt{CPI} &= \frac{\texttt{N}_{\texttt{cycles}_{\texttt{fp}}} + \texttt{N}_{\texttt{cycles}_{\texttt{int}}}}{\texttt{N}_{\texttt{instr}_{\texttt{fp}}} + \texttt{N}_{\texttt{intr}_{\texttt{int}}}} = \frac{700,000\ \texttt{cycles} + 50,000\ \texttt{cycles}}{100,000\ \texttt{instr} + 50,000\ \texttt{instr}} \\
\texttt{CPI} &= \frac{750,000}{150,000}\texttt{cycles/instr} \\
\texttt{CPI} &= 5\ \texttt{cycles/instr}.
\end{aligned}$$

E1.4 Processor B has an average CPI for Program 2 of 3.5. Its clock rate is 1.8 GHz. How much time does it take to execute the program?
Solution:

From (4)

$$\begin{aligned} T &= \frac{N_{instr} \times CPI}{f} \\ &= \frac{150,000 \times 3.5}{1.8 \times 10^9} \\ T_B &= 291.7\ \mu s \end{aligned}$$

E1.4 Which Processor is faster and by how much?
Solution:

For processor A, we have:

$$\begin{aligned} f_A &= 2\ GHz \\ N_{instr_A} &= 150,000\ instr \\ CPI_A &= 5\ cycles/instr \end{aligned}$$

From (4)

$$\begin{aligned} T &= \frac{N_{instr} \times CPI}{f} \\ T_A &= \frac{150,000 \times 5}{2 \times 10^9} \\ T_A &= 375\ \mu s \end{aligned}$$

From the former problem, we have:

$$T_B = 291.7\ \mu s$$

Then, we have:

$$T_A > T_B$$

From (2), we have:

$$\frac{Performance_X}{Performance_Y} = \frac{Execution\ time_Y}{Execution\ time_X} = n$$

$$\frac{Execution\ time_A}{Execution\ time_B} = \frac{375\ \mu s}{291.7\ \mu s} = 1.29$$

So,

$$\frac{Performance_B}{Performance_A} = 1.29$$

Hence, Processor B is 1.29 times faster than Processor A.

E2. A given application written in Java runs 15 seconds on a desktop processor. A new Yourself Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler? Pick the right answer from the three choices below

a.) $\frac{15 \times 0.6}{1.1} = 8.2 \text{ sec}$
b.) $\frac{15 \times 0.6}{1.1} = 9.9 \text{ sec}$
c.) $\frac{15 \times 0.6}{1.1} = 27.5 \text{ sec}$

Solution:

Be T_1 the time using the old compiler and T_2 using the new

$$
\begin{aligned}
T_1 &= \frac{N_{instr_1} \times CPI_1}{f} \\
T_2 &= \frac{N_{instr_2} \times CPI_2}{f} \\
N_{intr_2} &= 0.6 \cdot N_{intr_1} \\
CPI_2 &= 1.1 \cdot CPI_2 \\
T_2 &= \frac{0.6 \cdot N_{instr_1} \times 1.1 \cdot CPI_1}{f} \\
T_2 &= \frac{0.6 \cdot 1.1 N_{instr_1} \times CPI_1}{f} = 0.66 \cdot T_1 \\
T_1 &= 15 \text{ s}
\end{aligned}
$$

Then, we have:

$$T_2 = 0.66 \cdot 15 \text{ s} = 9.9 \text{ s}$$

With the new Java compiler the application runs in 9.9 s.

E3. Consider the following performance measurements for a program:

Measurement	Computer A	Computer B
Instruction count	10 billon	8 billon
Clock rate	4 GHz	4 GHz
CPI	1.0	1.1

a. Which computer has the higher MIPS rating?
b. Which computer is faster?

Solution:
a. ¸

From (6), we have:

$$\text{MIPS} = \frac{f}{CPI \times 10^6}$$

MIPS for computer A:

$$
\begin{aligned}
\text{MIPS}_A &= \frac{4 \cdot 10^9}{1.0 \times 10^6} \\
\text{MIPS}_A &= 4.0 \cdot 10^3 \text{ MIPS}
\end{aligned}
$$

MIPS for computer B:

$$\begin{aligned}\texttt{MIPS}_\texttt{B} &= \frac{4 \cdot 10^9}{1.1 \times 10^6} \\ \texttt{MIPS}_\texttt{B} &= 3.64 \cdot 10^3 \ \texttt{MIPS}\end{aligned}$$

Computer A has a higher MIPS than computer B.

b.)

From (4), we have

$$\texttt{T} = \frac{\texttt{N}_\texttt{instr} \times \texttt{CPI}}{\texttt{f}}$$

Time for computer A:

$$\begin{aligned}\texttt{T}_\texttt{A} &= \frac{10 \cdot 10^9 \times 1.0}{4 \cdot 10^9} \\ \texttt{T}_\texttt{A} &= 2.5 \ \texttt{s}\end{aligned}$$

Time for computer B:

$$\begin{aligned}\texttt{T}_\texttt{B} &= \frac{8 \cdot 10^9 \times 1.1}{4 \cdot 10^9} \\ \texttt{T}_\texttt{B} &= 2.2 \ \texttt{s}\end{aligned}$$

From (2)

$$\begin{aligned}\frac{\texttt{Performance}_\texttt{X}}{\texttt{Performance}_\texttt{Y}} &= \frac{\texttt{Execution time}_\texttt{Y}}{\texttt{Execution time}_\texttt{X}} = \texttt{n}, \quad \texttt{n} \geq 1 \\ \frac{\texttt{Performance}_\texttt{B}}{\texttt{Performance}_\texttt{A}} &= \frac{\texttt{Execution time}_\texttt{A}}{\texttt{Execution time}_\texttt{B}} = \frac{2.5 \ \texttt{s}}{2.2 \ \texttt{s}} = 1.14\end{aligned}$$

The performance of computer B is 1.14 times the performance of computer A , even though MIPS of computer A is higher than MIPS of computer B.

2 Instructions: Language of the Computer

Exercise 2.1
The following problems explore translating from C to MIPS. Assume that the variables `f, g, h,` and `i` are given and could be considered 32-bit integers as declared in a C program.
2.1.1 For the C statements above, what is the corresponding MIPS assembly code? Use a minimal number of MIPS assembly instructions.

Solution:
a. Let's assume that the C variables: `f, g` and `h` are assigned to the registers `$s0`, `$s1` and `$s2` respectively.

```
f = g - h;
```

Then, we have:

```
sub $s0, $s1, $s2    # register $s0 contains g - h
```

b. Let's assume that the C variables: `f, g` and `h` are assigned to the registers `$s0`, `$s1` and `$s2`, respectively, and a temporary register `$t0`.

```
f = g + (h - 5);
```

Then, we have

```
addi $t0, $s2, -5    # register $t0 contains h - 5
add  $s0, $s1, $t0   # register $s0 contains g + (h - 5)
```

2.1.2 For the C statements above, how many MIPS assembly instructions are needed to perform the C statement?

Solution:
In the case of section **a.** only one instruction is needed.
In the case of section **b.** two instructions are needed.

2.1.3 If the variables `f, g , h,` and `i` have values `1, 2, 3,` and `4`, respectively, what is the end value of `f`?
Solution:
a.

```
f <- 1, g <- 2, h <- 3, i <- 4
```

```
f = g - h
```

```
f <- 2 - 3
f <- -1
```

b.

```
f = g +  (h - 5)
```

```
f <- 2 + (3 - 5)
f <- 2 - 2
f <- 0
```

The following problems deal with translating from MIPS to C. For the following exercise, assume that the variables `g, h, i`, and `j` are given and could be considered 32-bit integers as declared in a C program.

a.	`addi f, f, 4`
b.	`add f, g, h` `sub f, i, f`

2.2.4 For the MIPS assembly instructions above, what is a corresponding C statement?

Solution:

a.

```
f = f + 4;
```

b.

```
f = i - (g + h);
```

Exercise 2.3

The following problems explore translating from C to MIPS. Assume that the variables `f` and `g` are given and could be considered 32-bit integers as declared in a C program. 2.3.1 For the C statements above,

a.	`f = -g - f;`
b.	`f = g + (-f - 5);`

what is the corresponding MIPS assembly code? Use a minimal number of MIPS assembly instructions.

Solution:

a.

Let's assume that the C variables: `f` and `g` are assigned to the registers `$s0` and `$s1`, respectively, and a temporary register `$t0`. Register `$zero` contains the constant value `0`.

Then, we have

```
sub $t0, $zero, $s1    # register $t0 contains -g
sub $s0, $t0, $s0      # register $s0 contains -g - f
```

b. Let's assume that the C variables: `f` and `g` are assigned to the registers `$s0` and `$s1`, respectively, and a temporary register `$t0`.

Then, we have

```
sub  $t0, $zero, $s0    # register $t0 contains -f
addi $t0, $t0, -5       # register $t0 contains -f - 5
add  $s0, $s1, $t0      # register $s0 contains g + (-f - 5)
```

2.3.2 For the C statements above, how many MIPS assembly instructions are needed to perform the C statement?
Solution:
Exercise 2.10
In the following problems, the data table contains bits that represent the opcode of an instruction. You will be asked to interpret the bits as MIPS instructions into assembly code and determine what format of MIPS instruction the bits represent.

a.	`0000 0010 0001 0000 1000 0000 0010 0000`$_{\text{two}}$
b.	`0000 0001 0100 1011 0100 1000 0010 0010`$_{\text{two}}$

2.10.1 For the binary entries above, what instruction do they represent?
Solution:
a.
`0000 0010 0001 0000 1000 0000 0010 0000`$_{\text{two}}$
If bits from `31` to `26` are `0`, then the instruction is an R-format instruction.
Dividing the binary word by fields:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
000000	10000	10000	10000	00000	100000

Field	Binary	MIPS	Range	Size
opcode	000000	R-type	31:25	6 bits
rs	10000	$s0	25:21	5 bits
rt	10000	$s0	20:16	5 bits
rd	10000	$s0	15:11	5 bits
shamt	00000	not used	10:6	5 bits
funct	100000	add	5:0	6 bits

```
add $s0, $s0, $s0
```

b.
0000 0001 0100 1011 0100 1000 0010 0010_{two}
If bits from 31 to 26 are 0, then the instruction is an R-format instruction.
Dividing by fields:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
000000	01010	01011	01001	00000	100010

Field	Binary	MIPS	Range	Size
opcode	000000	R-type	31:25	6 bits
rs	01010	$t2	25:21	5 bits
rt	01011	$t3	20:16	5 bits
rd	01001	$t1	15:11	5 bits
shamt	00000	not used	10:6	5 bits
funct	100010	sub	5:0	6 bits

```
sub $t1, $t2, $t3
```

2.10.2 What type (I-type, R-type, J-type) instruction do the binary entries above represent?
Solution:
Both are R-type intructions.
2.10.3 If the binary entries above were data bits, what number would they represent in hexadecimal?
Solution:
a.
0000 0010 0001 0000 1000 0000 0010 0000_{two}
0 2 1 0 8 0 2 0_{hex}
b.
0000 0001 0100 1011 0100 1000 0010 0010_{two}
0 1 4 B 4 8 2 2_{hex}
In the following problems, the data table contains MIPS instructions. You will be asked to translate the entries into the bits of the opcode and determine the MIPS instruction format.

a.	`addi $t0, $t0, 0`
b.	`sw $t1, 32($t2)`

2.10.4 For the instructions above, show the binary then hexadecimal representation of these instructions.
Solution:
a.

```
addi $t0, $t0, 0
```

addi is an I-type instruction. So, the

op	rs	rt	constant
6 bits	5 bits	5 bits	16 bits
001000	01000	01000	0000 0000 0000 0000

Field	Binary	MIPS	Range	Size
opcode	001000	I-type	31:25	6 bits
rs	01000	$t0	25:21	5 bits
rt	01000	$t0	20:16	5 bits
constant	000..0		15:0	16 bits

$$\begin{array}{cccccccc} 0010 & 0001 & 0000 & 1000 & 0000 & 0000 & 0000 & 0000_{\text{two}} \\ 2 & 1 & 0 & 8 & 0 & 0 & 0 & 0_{\text{hex}} \end{array}$$

b.

```
sw $t1, 32($t2)
```

sw is an I-type instruction. So, the

op	rs	rt	address
6 bits	5 bits	5 bits	16 bits
101011	01010	01001	0000 0000 0010 0000

Field	Binary	MIPS	Range	Size
opcode	101011	I-type	31:25	6 bits
rs	01010	$t0	25:21	5 bits
rt	01001	$t0	20:16	5 bits
constant	0...0100000	number	15:0	16 bits

$$\begin{array}{cccccccc} 1010 & 1101 & 0100 & 1001 & 0000 & 0000 & 0010 & 0000_{\text{two}} \\ A & D & 4 & 9 & 0 & 0 & 2 & 0_{\text{hex}} \end{array}$$

2.10.5 What type (I-type, R-type, J-type) instruction do the instructions above represent?
Solution:
Both of them are I-type instructions.

2.10.6 What is the binary then hexadecimal representation of the opcode, Rs, and Rt fields in this instruction? For R-type instructions,

what is the hexadecimal representation of the `Rd` and funct fields? For I-type instructions, what is the hexadecimal representation of the immediate field?
Solution:

Exercise 2.13
In the following problems, the data table contains the values for registers `$t0` and `$t1`. You will be asked to perform several MIPS logical operations on these registers.

a.	`$t0 = 0xAAAAAAAA, $t1 = 0x12345678`
b.	`$t0 = 0xF00DD00D, $t1 = 0x11111111`

2.13.1 For the lines above, what is the value of `$t2` for the following sequence of instructions?

```
sll $t2, $t0, 4
or $t2, $t2, $t1
```

Solution:
The first instruction

```
sll $t2, $t0, 4
```

is the shift left logical (`sll`) instruction. In this case, the bits of register `$t0` are shifted four places to the left and the result is stored in register `$t2`.
The second instruction

```
or $t2, $t2, $t1
```

is the `OR` instruction. It's a logical bit-by-bit operation with two operands that calculates a `1` if there is a `1` in either operand.
`$t2 <- $t2 OR $t1`

$\texttt{AAAAAAAA}_{\text{hex}} = \texttt{1010 1010 1010 1010 1010 1010 1010 1010}_{\text{two}}$
$\texttt{12345678}_{\text{hex}} = \texttt{0001 0010 0011 0100 0101 0110 0111 1000}_{\text{two}}$

```
$t0 -  1010 1010 1010 1010 1010 1010 1010 1010
$t1 -  0001 0010 0011 0100 0101 0110 0111 1000

$t2 <- 1010 1010 1010 1010 1010 1010 1010 0000 # sll $t2, $t0, 4

$t2  -  1010 1010 1010 1010 1010 1010 1010 0000
        or
$t1 -   0001 0010 0011 0100 0101 0110 0111 1000
        ---------------------------------------
$t2 <-  1011 1010 1011 1110 1111 1110 1111 1000 # or $t2, $t2, $
    t1
```

So, the value of register `$t2` is:
$\texttt{BABEFEF8}_{\text{hex}} = \texttt{1011 1010 1011 1110 1111 1110 1111 1000}_{\text{two}}$

2.13.2 For the values in the table above, what is the value of `$t2` for the following sequence of instructions?

```
sll $t2, $t0, 4
andi $t2, $t2, -1
```

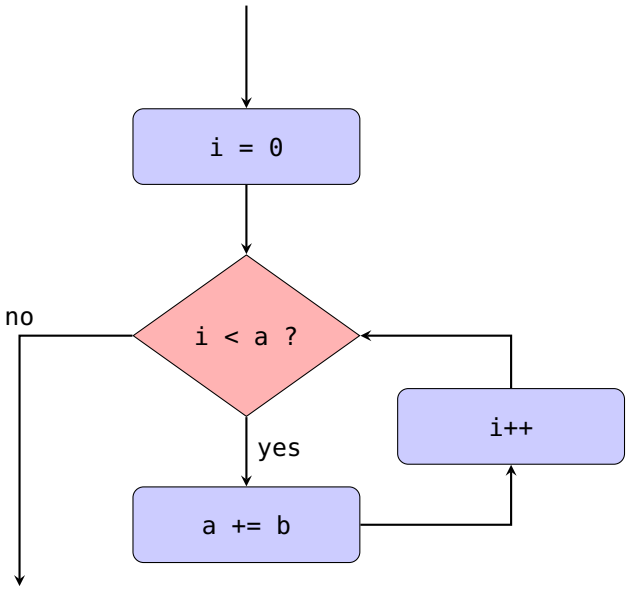
Exercise 2.18
For these problems, the table holds some `C` code. You will be asked to evaluate these `C` code statements in `MIPS assembly code`.

a.	`for(i=0; i<a; i++)` `  a += b;`

b.	`for(i=0; i<a; i++)`
	`  for(j=0; j<b; j++)`
	`  D[4*j] = i + j;`

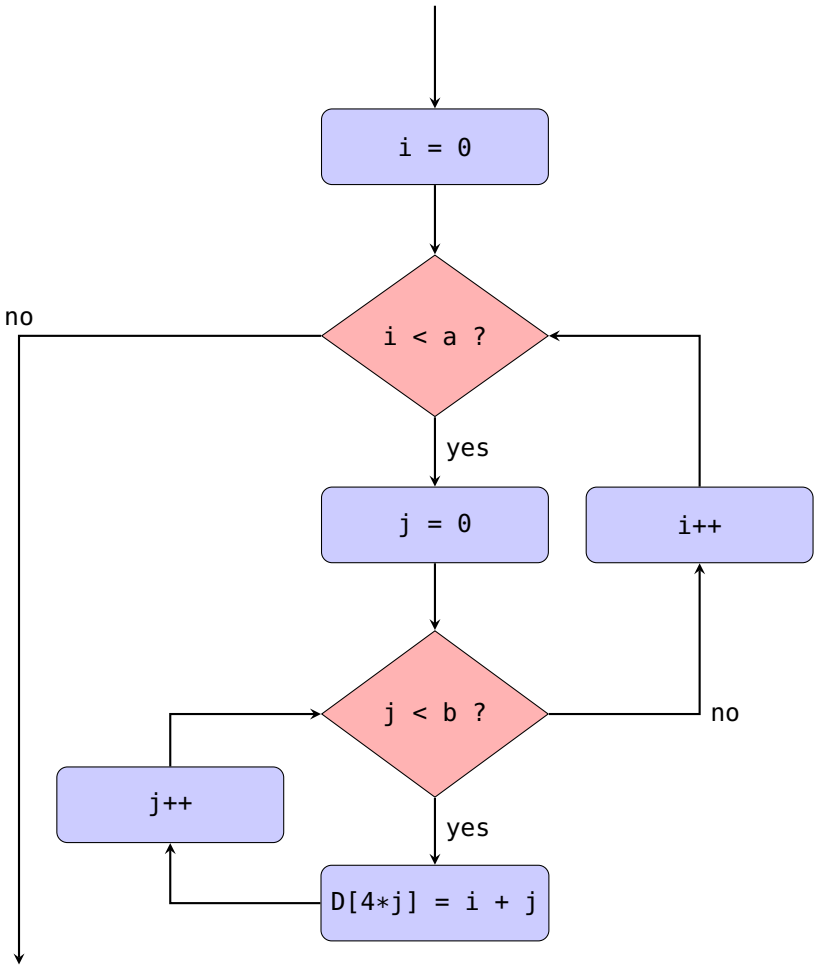
2.18.1 For the table above, draw a control-flow graph of the `C` code.
Solution:
a.

```
for(i = 0; i < a; i++)
    a += b;
```



b.

```
for(i = 0; i < a; i++)
    for(j = 0; j < b; j++)
        D[4*j] = i + j;
```



2.18.2 For the table above, translate the C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers $s0, $s1, $t0 and $t1, respectively. Also, assume that register $s2 holds the base address of the array D.

Solution:

a. C code:

```
for(i = 0; i < a; i++)
    a += b;
```

```
Registers:
    a - $s0, b - $s1, i - $t0, j - $t1, Base Address - $s2
```

MIPS assembly code:

```
        add  $t0, $zero, $zero    # i <- 0
Loop:
        slt  $t2, $t0, $s0        # i < a ? 1 : 0
        beq  $t2, $zero, Exit     # if i == 0 then Exit
        add  $s0, $s0, $s1        # a <- a + b
        addi $t0, $t0, 1          # i++
        j    Loop                 # goto Loop
Exit:
```

b.

```
Registers:
a - $s0, b - $s1, i - $t0, j - $t1, Base Address - $s2
```

C code:

```
for(i = 0; i < a; i++)
    for(j = 0; j < b; j++)
        D[4*j] = i + j;
```

MIPS assembly code:

```
        add  $t0, $zero, $zero    # i <- 0
Loop_i:                           # extern for loop
        slt  $t2, $t0, $s0        # i < a ? 1 : 0
        beq  $t2, $zero, Exit_i   # if i == 0 then Exit_i
        add  $t1, $zero, $zero    # j <- 0
Loop_j:                           # intern for loop
        slt  $t3, $t1, $s1        # j < b ? 1 : 0
        beq  $t3, $zero, Exit_j   # if j == 0 then Exit_j
        add  $t4, $t0, $t1        # t4 <- i + j
        sll  $t5, $t1, 4          # t5 <- 4*j
        add  $t5, $t5, $s2        # t5 <- 4*j + Base Address
        sw   $t4, 0($t5)          # D[4*j] <- i + j
        addi $t1, $t1, 1          # j++
        j    Loop_j               # goto Loop_j
Exit_j:
        addi $t0, $t0, 1          # i++
        j    Loop_i               # goto Loop_i
Exit_i:
```

2.18.3 How many MIPS instructions does it take to implement the C code? If the variables a and b are initialized to 10 and 1 and all elements of D are initially 0, what is the total number of MIPS instructions that is executed to complete the loop?
Solution:
a. In the first case it has 6 instructions lines. Labels: Loop: and Exit: don't count as instructions.

- **R-Type:**

  add, slt

- **I-Type:**

  beq, addi

- **J-Type:**

  j

b. In the second case it has 14 instructions lines. Labels: Loop_i:, Loop_j, Exit_i and Exit_j don't count as instructions.

- **R-Type:**

  `add, slt, sll`

- **I-Type:**

  `beq, sw, addi`

- **J-Type:**

  `j`

Now, if `a = 10`, `b = 1` and the elements of `D` array are `0`.

For these problems, the table holds `MIPS` assembly code fragments. You will be asked to evaluate each of the code fragments, familiarizing you with the different `MIPS` branch instructions.

a.	`addi $t1, $0, 50`
	`LOOP: lw $s1, 0($s0)`
	`add $s2, $s2, $s1`
	`lw $s1, 4($s0)`
	`add $s2, $s2, $s1`
	`addi $s0, $s0, 8`
	`subi $t1, $t1, 1`
	`bne $t1, $0, LOOP`

b.	`addi $t1, $0, $0`
	`LOOP: lw $s1, 0($s0)`
	`add $s2, $s2, $s1`
	`addi $s0, $s0, 4`
	`addi $t1, $t1, 1`
	`slti $t2, $t1, 100`
	`bne $t2, $0, LOOP`

2.18.4 What is the total number of `MIPS` instructions executed?
a.
The first line of code is executed once. The `Loop` has `7` lines of code that are executed `50` times. Hence, we have: $1 + 7 \times 50 = 351$. The total number of `MIPS` instructions executed are `351`.
b.
The first line of code is executed once. The Loop has `9` lines of code that are executed `50` times. Hence, we have: $1 + 9 \times 50 = 451$. The total number of `MIPS` instructions executed are `451`.

2.18.5 Translate the loops above into `C`. Assume that the `C-level` integer `i` is held in register `$t1`, `$s2` holds the `C-level` integer called `result`, and `$s0` holds the base address of the integer `MemArray`.
Solution:
a.

```
i - $t1, result - $s2, $s0 - BaseAdress of MemArray
```

```
int j = 0;
int result  = 0;
```

```
for(int i = 50; i > 0; i--)
{
    result += MemArray[j];
    result += MemArray[j + 1];
    j += 2;
}
```

or

```
int j = 0;
int result  = 0;
for (int i = 50; i > 0; i--)
{
    result += *(MemArray + j);
    result += *(MemArray + j + 1);
    j += 2;
}
```

b.

`i - $t1, result - $s2, $s0 - BaseAdress of MemArray`

```
int j = 0, result  = 0, int temp;

for (int i = 0; i < 100; i++)
{
    temp = *(MemArray + j);    // lw $s1, 0($s0)
    result += temp;            // add $s2, $s2, $s1
    j++;                       // addi $s0, $s0, 4
}
```