Computer Architecture and Engineering - 3rd year Bachelor in Informatics - ETSINF - UPV

Exercises and solutions of Unit 1. “Introduction to computer architecture”

Exercise 1.1.

Two alternative architecture designs for a computer (let's name them A and B) are available. Both affect the clock frequency and the number of *l*oad/store instructions in programs. Taking into account that:

- The clock frequency of A is 5% faster than the one of B;
- A program compiled for A has 30% of *load/store* instructions, while the same program compiled for B reduces in 1/3 the number of such type of instructions;
- CPI is 1 for all instructions;

Decide which architecture design is better.

Solution:

Relating the execution time of both alternatives:

$$\frac{t_A}{t_B} = \frac{I_A \cdot CPI_A \cdot T_A}{I_B \cdot CPI_B \cdot T_B}$$

Let us see how can we compute the different parameter of the equation:

$$I_B = I_A - \frac{1}{3} \cdot \frac{30}{100} \cdot I_A = \frac{90}{100} I_A$$

$$CPI_B = CPI_A = 1$$

$$\frac{T_B}{T_A} = \frac{f_A}{f_B} = \frac{105}{100}$$

as a result:

$$\frac{t_A}{t_B} = \frac{100 \cdot 100}{90 \cdot 105} = 1.058$$

Conclusion: Machine B is 5.8% faster than machine A.

□

Exercise 1.2.

In a *Load/Store* computer, it has been measured the frequency of occurrence, and the CPI, of the following types of instructions:

Operation	%	CPI
ALU	43	1
LOAD	21	2
STORE	12	2
Branches	24	2

It has also been observed that 25 % of ALU instructions have a source operand that is never reused. Thus, it is proposed to enrich the instruction set of the computer with new arithmetic instructions having one source operand in memory, as it is shown hereafter:

```
        ld r1,o(r10)
        dadd r3,r1,r2
```

can be replaced by:

```
        daddm r3,o(r10),r2
```

New instructions have a CPI = 2 and their use increases the CPI of branch instructions by one. Determine quantitatively the interest of this change considering that the cost of the computer remains the same.

Solution:

This exercise evaluates the impact of an ISA modification on the execution time of a particular program. The modification affects the number of instructions and the CPI. Considering a the original processor and b de modified one, one can express both execution times as a function of the number of instructions I_a in the original program and the clock period t_c.

For the original processor a it is enough to compute the average CPI_a considering the provided data:

$$\begin{aligned} T_a &= I_a \cdot CPI_a \cdot t_c \\ &= I_a \cdot (0.43 \cdot 1 + 0.21 \cdot 2 + 0.12 \cdot 2 + 0.24 \cdot 2) \cdot t_c \\ &= 1.57 \cdot I_a \cdot t_c \end{aligned}$$

For processor b it is necessary to compute the number of instructions I_b and the average CPI_b. The number of instructions is different since one must decrement I_a by the 25% of ALU instructions:

$$I_b = (1 - (0.25 \cdot 0.43))I_a = (1 - 0.1075) \cdot I_a = 0.8925 I_a$$

For computing the CPI, the frequency of each type of instruction must be normalized:

$$\begin{aligned} CPI_b &= \frac{(0.43 - (0.25 \cdot 0.43)) \cdot 1 + (0.25 \cdot 0.43) \cdot 2 + ...}{0.8925} \\ & \frac{... + (0.21 - (0.25 \cdot 0.43)) \cdot 2 + 0.12 \cdot 2 + (0.24 \cdot 3)}{0.8925} \\ &= \frac{1.7025}{0.8925} = 1.91 \end{aligned}$$

The resulting execution time is:

$$T_b = I_b \cdot CPI_a \cdot t_c = 1.703 \cdot I_a \cdot t_c$$

The modification is not interesting, since the resulting speed-up is lower than 1:

$$S = \frac{1.57 \cdot I_a \cdot t_c}{1.703 \cdot I_a \cdot t_c} < 1$$

$\square$

Exercise 1.3.

A program *P* is compiled for a *MIPS R2000* computer working at 100 MHz and integrating a coprocessor. *P* performs two millions floating point operations. The compiler translates each floating point operation into either a coprocessor instruction or a routine of integer instructions attending to the provided set of compilation options. Changing these options, *P* is compiled twice, thus generating two different executables: P_h (code for *MIPS R2000* with coprocessor) and P_s (code for *MIPS R2000* without coprocessor). Both versions are executed and these are the obtained results:

Program	Execution time	Average measured CPI
P_h	92 miliseconds	3.1 cycles
P_s	1.2 seconds	1.2 cycles

Answer the following questions:

1. The number of instructions executed per unit of time for both program versions, expressed in *MIPS* and the total number of executed instructions in each case.
2. The average number of integer instructions replacing each floating point operation in P_s.
3. The throughput of both programs expressed in *MFLOPS*.

Solution:

1. Computing the throughput in MIPS, one must consider the execution time of the program:

$$\text{Prod(MIPS)} = \frac{I}{T_{ej} \cdot 10^6} = \frac{I}{I \cdot CPI \cdot T \cdot 10^6} = \frac{1}{CPI \cdot T \cdot 10^6}$$

From this expression, since we know the CPI of both programs and the clock period ($\frac{1}{f} = \frac{1}{100 \cdot 10^6} = 10$ns), we can compute the MIPS. Additionally, since we also know the global execution time T_{ej} in both cases, the total number of instructions can be computed as:

$$I = \text{Prod(MIPS)} \cdot T_{ej} \cdot 10^6$$

Replacing, one obtains:

Program	Throughput (MIPS)	number of instructions
P_h	32.26	2.97 millions
P_s	83.33	100 millions

2. In P_s all instructions operate on integers, but some are devoted to the emulation of floating point operations and the rest are used for other data types and other program needs. The latter integer instructions can be also found in P_h, and can be computed as follows: 2.97 million instructions $-$ 2 million floating point instructions = 0.97 million integer instructions. As a result, the number of instructions in P_s performing floating point computations is 100 million $-$ 0.97 million = 99.03 million.

   In summary, 2 million floating operation correspond to 99.03 integer instructions, so the requested relation is:

$$\frac{\text{99.03 million}}{\text{2 million}} = 49.65 \text{ instructions}$$

3. For this computation it is enough to apply the definition. In both case, the number of operation carried out is the same, but the execution times differ. The result is:

Program	Throughput (MFLOPS)
P_h	21.74
P_s	1.67

Such values for *MFLOPS* are quite similar to the real performance of these programs. Program P_h is 13 times faster than P_s according to their respective execution times. The throughput ratio in *MFLOPS* is quite similar. This is very different in the case of considering the throughput ratio in *MIPS*, since the faster program is P_s.

□

Exercise 1.4.

An MP3 compression program has been written in C in order to compare two processors: A (the older one) and B (the new one). Processor A works at 200 MHz. Its instruction set only includes integer instructions and its CPI is 1. On the other hand, processor B works at 900 MHz. Its instruction set extends the A's ISA with multimedia instructions (*MI*). The average CPI provided by processor B varies according to the executed application: integer instructions takes 1 cycle, while multimedia ones take 3 cycles. Depending on the compilation options, the compiler generates either executables containing *MI* instructions or executables coping with the same task but using only integer instructions. This means that each *MI* instruction has an equivalent set of integer instructions for performing the same operation. Two different executables are generated from the compilation of the same compression algorithm: *H* (using *MI* instructions) and *S* (using only integer ones). Results issued from their execution are: processor A needs 50 seconds to compress a song when running code *S* , while processor B performs the same task in 8 seconds when executing code *H*. In addition, it has been observed that with code *H*, the number of executed instructions is 36% less than for code S.

1. Which is the average CPI of processor B when executing *H*?
2. Which is the percentage of *MI* instructions in *H*?
3. How many integer instructions are equivalent to one *MI* instruction?
4. How long will it take for processor B in order to compress the same song running *S*?

Solution:

1. The execution time for code H in processor B is

$$T_{H,B} = I_H \cdot CPI_{H,B} \cdot t_B,$$

where

- The number of executed instructions I_H depends on code H.
- The average CPI $CPI_{H,B}$ depends on the distribution of instructions in H and the CPI of each instruction in B.
- The clock period t_B depends on the type of processor (B).

From this equation we can obtain $CPI_{H,B}$:

$$T_{H,B} = I_H \cdot CPI_{H,B} \cdot t_B \rightarrow 8s = I_H \cdot CPI_{H,B} \cdot 1.11ns \rightarrow CPI_{H,B} = \frac{8s}{I_H \cdot 1.11ns}.$$

On the other hand, we know that code H executes 36% less instructions than code S. Therefore,

$$I_H = 0.64 \cdot I_S \rightarrow CPI_{H,B} = \frac{8s}{0.64 \cdot I_S \cdot 1.11ns}.$$

.

I_S can be obtained from the equation of the execution time of code S in processor A. Thus,

$$T_{S,A} = I_S \cdot CPI_{S,A} \cdot t_A \rightarrow 50s = I_S \cdot 1 \cdot 5ns \rightarrow I_S = \frac{50s}{5ns} = 10 \cdot 10^9.$$

Therefore,

$$CPI_{H,B} = \frac{8s}{0.64 \cdot 10 \cdot 10^9 \cdot 1.11ns} = 1.125.$$

Alternatively, we can express the speedup as a function of the execution times of the programs. Thus,

$$S = \frac{50}{8} = 6.25 = \frac{I_S \cdot CPI_{S,A} \cdot t_A}{I_H \cdot CPI_{H,B} \cdot t_B} = \frac{I_S \cdot 1 \cdot 5ns}{0.64 \cdot I_S \cdot CPI_{H,B} \cdot 1.11ns}.$$

From this expression we obtain $CPI_{H,B} = 1.125$.

2. The average CPI previously computed corresponds to a mix of instructions. A fraction x of these instructions are of type *MI*. They consume 3 cycles. The rest, $(1 - x)$, are integer instructions consuming one cycle. Computing x from the following equation

$$CPI_{H,B} = 1.125 = (1 - x) \cdot 1 + x \cdot 3$$

we obtain that the percentage of *MI* instructions is 6.25 %

3. Out of 100 instructions in S there are 64 instructions in H. Among these instructions, 6.25% are *M*I instructions (that is, $0.0625 \cdot 64 = 4$ instructions). As a result, for each 100 instructions in S, there are 60 integer and 4 *M*I in H. Therefore, there are 40 (100 total - 60 shared) integer instructions in *S* that will be replaced with 4 *MI* instructions en H. So, one *M*I instruction replaces 10 integer instructions.

4. The execution time for code S in processor B is

$$T_{S,B} = I_S \cdot CPI_{S,B} \cdot t_B = 10 \cdot 10^9 \cdot 1 \cdot 1.11ns = 11.1s.$$

Alternatively, we know that, independently of the processor, the number of executed instructions for code S is I_S. We also know that $CPI_{S,A} = CPI_{S,B}$ since S exclusively contains integer instructions, which have CPI equal to 1 in both processors. Therefore, only the clock period changes. From the ratio between clock frequencies we get $S = 900/200 = 4.5$, and thus, the execution time will be $t = 50/4.5 = 11.1$ seconds.

□

Exercise 1.5.

Integer instructions of a CPU execute in 1 clock cycle, while floating point instructions take 5 cycles. Most programs to execute include 20% of floating point instructions. From a performance and cost analysis point of view, to what extent is it interesting to redesign the floating point unit of this CPU to make it 5 times faster if this redesign doubles the cost of the CPU? Justify your answer.

Solution:

The goal is to compare an initial design, named A, to an alternative design, called B. The reference for this comparison is a program that will be used in both designs. The unit of time used in the comparison is the clock cycle, which is the same in both designs.

The solution will be searched considering two strategies: the equation of the execution time and Amdahl's law.

- Use of the equation of execution time

  As factors I and T do not change, only the CPI of each one has to be computed:

$$CPI_A = 0.2 \cdot 5 + 0.8 \cdot 1 = 1.8$$

$$CPI_B = 0.2 \cdot 1 + 0.8 \cdot 1 = 1$$

  Relating execution times,

$$S = \frac{T_A}{T_B} = \frac{I \cdot 1.8 \cdot T}{I \cdot 1 \cdot T} = 1.8$$

  The new CPU design is 1.8 times faster, but its cost doubles. As a result, the proposal is not interesting from a performance-cost viewpoint.

- Applying Amdahl's law

  Using the general expression

$$S' = \frac{1}{(1-F) + \frac{F}{S}}$$

  where S is the proposed improvement, and F is the percentage of time where the optimization can be used. The speed-up $S = 5$ is provided, but not the factor F. The fraction of time is not the portion of instructions affected by the changes (20 %), since each type of instruction has a different duration. The relevant fraction of time is the one devoted by the CPU A to the execution of floating point instructions wrt the total. If a program integrates n instructions, the 20% representing the floating point instructions takes $5 \cdot (0.2 \cdot n)$ cycles, which should be compared with the total execution time, $1.8 \cdot n$ cycles in this case.

$$F = \frac{1}{1.8} = 0.55$$

  Applying Amdahl's law:

$$S' = \frac{T_A}{T_B} = \frac{1}{(1-0.55) + \frac{0.55}{5}} = 1.8$$

  As previously stated, option *B* is 1.8 times faster than option *A*.

□

Exercise 1.6.

A computer coprocessor improves the processing of floating point operations by a factor of 5. The execution time of a certain program is 1 minute when the coprocessor is available and it raises to 2.5 minutes without the coprocessor . Compute the percentage of execution time devoted by the program to the execution of floating point operations when the coprocessor is not installed.

Solution:

The goal here is to apply the law of Amdahl in the reverse way. Two speed-ups are available: the one relating to the modified part (S) and the resulting one S'. Computing F according to Amdahl's law:

$$F = \frac{S' \cdot S - S}{S' \cdot S - S'}$$

As a result:

$$F = \frac{2.5 \cdot 5 - 5}{2.5 \cdot 5 - 2.5} = 0.75 = 75\%$$

which is the requested percentage of time.

□

Ejercicio 1.7.

Exercise 1.8.

A computer has a load/store processor integrating a single level of (internal) cache L1. Memory access instructions are 30% of the total number of executed instructions. 5% of such memory access instructions lead to cache misses requiring a main memory access that takes 10 clock cycles, which must be added to the time required for serving a cache hit. ThIn absence of cache misses, the processor CPI is 1.

Study the interest of including a second level of (external) cache L2. Experiments performed have shown that L2 solves 90% of L1 misses, thus reducing the penalty induced by such misses to only 2 clock cycles. As a result, only 10% of L1 misses are penalized with 10 clock cycles.

1. Which is the fraction of time spent by the processor accessing main memory when only L1 is available?
2. In average, which is the penalty imposed to the processor in each L1miss when L2 is available?
3. Which is the global speed-up resulting from the inclusion of L2 in the computer?
4. Which is the fraction of time spent by the processor accessing main memory when both L1 and L2 are available?
5. If the computer costs 1000 euros, and adopting a cost-performance point of view, which is the maximum acceptable cost for the processor integrating the second level of cache?

Solution:

1. If one considers 100 instructions, 30 are load/store and 1,5 will provoke a page miss, thus introducing a penalty of 15 cycles, which will be added to the 100 base cycles. So, from each 115 cycles, 15 are related to the introduced penalty:

$$F = \frac{15}{115} = 13\%$$

2. From each 100 L1 misses, 90 can be solved in 2 cycles and the 10 remaining in 10 cycles. So, the average penalty is:

$$P = \frac{90 \cdot 2 + 10 \cdot 10}{100} = 2.80\ cycles$$

3. Amdahl's law can be applied here, considering $F = 13\%$ and $S = \frac{10}{2.8} = 3.57$

$$S' = \frac{1}{1 - 0.13 + \frac{0.13}{3.57}} = 1.10$$

4. Each 100 instructions require $1.5 \cdot 2.8$ cycles of penalty, i.e. 104.2 cycles in total. From them, $1.5 \cdot 10\% \cdot 10$ cycles will correspond to memory accesses. As a result:

$$F = \frac{1.5 \cdot 0.1 \cdot 10}{104.2} = 1,4\%$$

5. If the speed-up is of 10%, one can invest up to 100 €

$\square$

Exercise 1.9.

A program P takes $120\ s$ to execute on a computer C. The following improvements are under evaluation in order to accelerate the execution of this program:

1) Purchase a RAID disk controller card, which would decrease the disk access time to 40% of its original value, reducing at the same time the total execution time of P to $84\ s$. This card costs 90 €.

2) Incorporate a video accelerator card, whose cost is 125 €. This decision is motivated by the fact that P devotes 30% of its execution time to graphics processing. The integration of this accelerator card in the system would reduce the time related to graphics processing to $1/3$ of its original value.

Taking into account these considerations, answer the following questions:

1. If the original computer cost was 300 €, which of the presented improvements are really interesting? Study each option separately from a cost/performance viewpoint. Justify your answer.

2. Which would had been the original cost of computer C to make worthwhile the simultaneous inclusion of both improvements? Adopt a cost/performance viewpoint in your analysis and justify your answer.

Solution:

1. In order to apply an analysis from the viewpoint of performance and cost, it is necessary to quantify the global effect of each particular improvement (S'_d for the disk and S'_v for the video subsystem) and compare it with the corresponding increment of price.

   In the case of the disk:

$$S'_d = \frac{T_{\text{original}}}{T_{\text{nuevo}}} = \frac{120}{84} = 1.43$$

   If the original cost was of $C_o = 300$ € and the cost with the improvement is of $C_d = 390$ €, the relation is:

$$\frac{C_d}{C_o} = \frac{390}{300} = 1.3$$

   So the improvement introduced by the disk controller is interesting in this case.

   S'_v can be computed using the law of Amdahl, considering $F_v = 30\%$ and the speed-up $S_v = 3$.

$$S'_v = \frac{1}{(1 - F_v + \frac{F_v}{S_v}} = \frac{1}{(1-0.3) + \frac{0.3}{3}} = 1.25$$

The relation of costs is:

$$\frac{C_v}{C_o} = \frac{425}{300} = 1.42$$

So, in this case, buying the video cards is not recommended. coste/prestaciones.

2. The global improvement, S'_{dv}, can be obtained using Amdahl's law:

$$S'_{dv} = \frac{1}{1 - F_d - F_v + \frac{F_d}{S_d} + \frac{F_v}{S_v}}$$

where S_d and F_d must be determined. In order to compute S_d

$$S_d = \frac{1}{0.4} = 2.5$$

and F_d can be obtained from:

$$S'_d = \frac{1}{(1-F_d) + \frac{F_d}{S_d}} \Rightarrow 1.43 = \frac{1}{(1-F_d) + \frac{F_d}{2.5}}$$

$$F_{\text{disco}} = \frac{0.75}{1.5} = 0.5$$

As a result:

$$S'_{dv} = \frac{1}{1 - F_d - F_v + \frac{F_d}{S_d} + \frac{F_v}{S_v}}$$

$$S'_{dv} = \frac{1}{1 - 0.5 - 0.3 + \frac{0.5}{2.5} + \frac{0.3}{3}} = \frac{1}{0.2 + 0.2 + 0.1} = \frac{1}{0.5} = 2$$

The global improvement costs $90 + 125 = 215$ €. The increment of cost must be equal or lower than such global speed-up in order to accept the incorporation of both modification. Accordingly, the limit of the cost of the original computer must be:

$$\begin{aligned}
C_{\text{improved}} &\leq S'_{\text{disk+video}} \cdot C_{\text{original}} \\
C_{\text{original}} + 215 &\leq 2 \cdot C_{\text{original}} \\
1 \cdot C_{\text{original}} &\geq 215 \\
C_{\text{original}} &\geq 215 \text{ €}
\end{aligned}$$

□

Exercise 1.10.

The execution of a given application has been monitored. It has been determined that the application executes in 10 minutes, spending most of its execution time running *P1* and *P2* procedures. For the sake of reducing the application execution time, the code of both procedures is modified. In the ideal case, if the resulting speed-up for both procedures (*P1* and *P2*) is infinite, then the execution time of the application is reduced to 2 minutes. However, the coded finally produced makes *P1* to carry out its work 5 times faster than before, while *P2* runs 100% faster than originally. In such conditions, the resulting execution time for the application is 4.5 minutes.

Determine the amount of time corresponding to *P1* and *P2* in the original application.

Solution:

Consider $F1$ and $F2$ as the time consumed by each one of the procedures, and R the time consumed by the rest of the code. One knows that:

$F1 + F2 + R = 10$

When $P1$ and $P2$ will be improved, the program will consume a time $F1'$ and $F2'$ in each one of them. Considering an infinite improvement within these procedures ($F1' = 0$ and $F2' = 0$), the execution time will become of 2 minutes:

$R = 2$

As a result:

$F1 + F2 + 2 = 10 \rightarrow F1 + F2 = 8$

On the other hand, one knows that improving $P1$ and $P2$ in 5 and 2 times, respectively, the execution time will become of 4.5 minutes:

$\frac{F1}{5} + \frac{F2}{2} + 2 = 4.5$

This results in a system of equations in two variables:

$$\left.\begin{array}{rcl} F1 + F2 & = & 8 \\ \frac{F1}{5} + \frac{F2}{2} & = & 2.5 \end{array}\right\}$$

Solving such systems results in:

$F1 = 5$ min y $F2 = 3$ min

□

Exercise 1.11.

Farmax Ltd. has a HAL supercomputer devoted to chemical analysis tasks. The computer integrates a RIX/300 processor working at 300 MHz. The system is only used for running the SpectroQuimix program, which makes an intensive use of floating point instructions. After monitoring the system, the following information becomes available:

- The fraction of time devoted by the processor to the execution of floating point instructions is 75%.
- Frequency of floating point instructions in SpectroQuimix is: (see table).
- CPI of each type of floating point instruction is: (see table)

Frequency (%)	Operation	CPI
30	add	5
10	sub	5
40	mult	10
20	div	40

The Farmax direction ask to the head of computer engineers for increasing the computation power of HAL. After checking the alternatives with providers, the following proposal is under study: Replace the processor by the new RIX/400E, which is binary-compatible with the existing processor. It works at 400 MHz and integrates an improved floating point unit. In this new processor, the CPI of integer instructions remains the same, but CPI of floating point ones change as follows:

Operation	CPI
add	3
sub	3
mult	7
div	25

Compute:

1. The speed-up resulting from the execution of integer instructions.
2. The average CPI of floating point instructions of RIX/300 and RIX/400 processors when executing the *SpectroQuimix* program.
3. The throughput (expressed in MFLOPS) resulting from the execution of the *SpectroQuimix* program on the RIX/300 processor.
4. The speed-up resulting from the replacement of the processor when execution floating-point instructions.
5. The global speed-up resulting from the execution of the *SpectroQuimix* program.
6. The fraction of time taken by the RIX/400 processor for the execution of floating-point instructions.
7. The throughput, expressed in MFLOPS, resulting from the execution of the *SpectroQuimix* program on the RIX/400 processor.

Solución:

This problem requires the analysis of the performance of both processors executing integer and floating point instructions.

Let's call A the current RIX/300-based computer and B its RIX/400E alternative,

1. Restricting the analysis to integer instructions, the execution time of the integer instructions in the *Espectroquimix* program is

$$T_{int} = I_{int} \cdot CPI_{int} \cdot t_C$$

As both processors are binary compatible, the number of integer instructions does not change from one to another, since they execute the same program. As the average (unknown) CPI of integer instruction does not change either, the speed-up should rely only on the change of the clock frequency.

$$S_{int} = \frac{t_{C(A)}}{t_{C(B)}} == \frac{f_B}{f_A} = \frac{400}{300} = 1.333$$

RIX/400E is 1.33 times faster then the RIX/300.

2. Frequency of instructions does not change, but their respective CPI do.

$$CPI_{fpA} = 0.30 \cdot 5 + 0.10 \cdot 5 + 0.40 \cdot 10 + 0.20 \cdot 40 = 14 \text{ cycles}$$

$$CPI_{fpB} = 0.30 \cdot 3 + 0.10 \cdot 3 + 0.40 \cdot 7 + 0.20 \cdot 25 = 9 \text{ cycles}$$

3. If the clock cycle of one processor is t_C, or its frequency is $f = frac1t_C$, the number of instructions executed per second (IPS) is

$$IPS = \frac{1}{t_C \cdot CPI} = \frac{f}{CPI}$$

Each floating point instruction carries out an operation. It is important to note that the floating point computation only happens during the fraction of time $F_{cf} = 75\%$.

$$ThroughputFP_A = \frac{F_{fp} \cdot f_A}{CPI_{fpA}} = \frac{0.75 \cdot 300}{14} = 16.1 \text{ MFLOPS}$$

4. In the case of the floating point unit, two considerations must be taken into account: CPI_{fp} reduction and the increase of the clock frequency.

$$S_{fp} = \frac{CPI_{fpA}}{CPIfpB} \cdot \frac{f_B}{f_A} = \frac{14}{9} \cdot \frac{400}{300} = 2.07$$

5. Here one must separately consider integer and floating point processing; for each on there is an average fraction of time and a local speed-up is applied. Using Amdahl, the resulting global speed-up S_g can be computed as follows:

$$S_g = \frac{1}{1 - F_{int} - F_{fp} + \frac{F_{int}}{S_{int}} + \frac{F_{fp}}{S_{fp}}} = \frac{1}{\frac{0.25}{1.33} + \frac{0.75}{2.07}} = 1.82$$

6. The new fraction will be

$$F' = F_{fp} \cdot \frac{S_g}{S_{fp}} = 0.75 \cdot \frac{1.82}{2.074} = 0.66$$

7. The floating point throughput will be

$$FP_Throughput_A = \frac{0.66 \cdot 400}{9} = 29.3\text{MFLOPS}$$

□

Exercise 1.12.

A small Internet server, devoted to service requests arriving through the network, includes a motherboard with two quad-core CPUs based on the AMD Barcelona core. Each CPU has two memory controllers. The application running on the server consists of many independent tasks running concurrently. Each task spends 50% of the execution time executing instructions on the CPU, and 30% of the execution time doing memory accesses that cannot be overlapped with instruction execution (the processor stalls during those accesses).

After several years of operation, the company needs to update the server. Among the available options, there are two configurations: a) a motherboard with two Intel Nehalem processors or b) a motherboard with two AMD Magny-Cours processors. Each Intel processor features 8 cores and 8 memory controllers while each AMD processor features 12 cores and 4 memory controllers. Assuming that all cores in the Intel and the AMD processor are equally fast, that each of those cores is 20% faster than a Barcelona core, and that each memory controller provides the same memory bandwidth and latency for all the processors, which motherboard will deliver the highest speed-up?

Solución:

The fraction of total execution time consumed by the CPU and the memory is 50% and 30%, respectively. Since tasks are independent from each other, a larger number of cores achieves a linear CPU speedup. Thus, the speedup achieved by Intel and AMD cores is $S_{IntelCPU} = 1.2 * 8/4 = 2.4$ and $S_{AMDCPU} = 1.2 * 12/4 = 3.6$, respectively. Since memory controllers are identical, the speedup is exclusively due to a larger number of memory controllers. Therefore, the speedup achieved by the Intel and AMD memory controllers in the new chips is $S_{IntelRAM} = 8/2 = 4$ and $S_{AMDRAM} = 4/2 = 2$, respectively. Applying Amdahl's law to the combination of CPU and memory, the total speedup achieved by Intel processors is:

$$S_{Intel} = \frac{1}{(1 - 0.5 - 0.3) + \frac{0.5}{2.4} + \frac{0.3}{4}} = 2.069 \tag{1}$$

The total speedup achieved by AMD processors is:

$$S_{AMD} = \frac{1}{(1 - 0.5 - 0.3) + \frac{0.5}{3.6} + \frac{0.3}{2}} = 2.045 \tag{2}$$

Thus, Intel motherboards are slightly better for this application.

Note: If the 1.2 factor is not considered then S_{Intel} = 1,905 and S_{AMD}=1,935.
In this case, AMD motherboards will be better for this application.

□

Exercise 1.13.

A 500 MHz *l*oad/store processor is modified to add a carry flag in its arithmetic operators. The modification affects both its ALU and instruction set. The analysis of the instruction set and the average *CPI* of the original processor is:

Type	Frequency	*CPI*
arithmetic	50 %	1
load	20 %	2
store	10 %	1.2
branch	20 %	1.2

The modification earns 1 over 10 arithmetic instructions, but their *CPI* increases to 1.2, while the *CPI* of branches goes to 1.5. On the other hand, the redesign of the decoder, the ALU and the hazard detection circuitry have reduced the maximum reachable clock frequency to 400 MHz.

Study the proposed modification according to the following steps:

1. Compute the *CPI* exhibited by the original processor.
2. Compute the *CPI* exhibited by the modified processor.
3. Determine which design is faster and quantify your answer.

Solución:

1. From the table it can be deduced that *CPI*(original) = 1.26 cycles.
2. Take into account that the modification reduces in $10\% \cdot 50\% = 5\%$ the number of necessary instructions.

$$\mathit{CPI}(modified) = \frac{0.45 \cdot 1.2 + 0.2 \cdot 2 + 0.1 \cdot 1.2 + 0.2 \cdot 1.5}{0.95} = 1.43$$

3. Program execution times must be compared in both processors. If I is the number of instructions in a program in the original processor, the comparison will be:

$$\frac{T(\text{modified})}{T(\text{original})} = \frac{(I \cdot 0.95) \cdot 1.43 \cdot 2.5}{I \cdot 1.26 \cdot 2} = 1.35$$

where clock cycles are expressed in ns. The original design is 35% faster than the modified one.

□

Exercise 1.14.

A 32-bit load/store processor is available. The processor integrates instructions to work with *bytes*, *halfwords* and *words*. The execution of TEX in this processor shows that 11% of data references are performed on *bytes* and *halfwords*, and the rest are carried out on *words*. It is also observed that 36% of program instructions access memory, *load* instructions doubling the frequency of occurrence of *store* ones. Finally, the measured CPI is 1.

Load instructions are the same provided by the *MIPS* instruction set: (*lb*, *lbu*, *lh*, *lhu* and *lw*). The same applies to store instructions (*sb*, *sh* and *sw*).

In order to improve performance, the following architecture modifications are under study:

- Remove from the instruction set those instructions providing access to *bytes* and *halfwords*. As a result, programs requiring the removed instructions will use the following alternatives:

Code with *bytes* access	Code without *bytes* access

`lb/lbu/lh/lhu r,A`	`lw r,A′`
	`extract n,r`

`sb/sh r,A`	`lw r′,a′`
	`insert n,r,r′`
	`sw r′,a′`

New insertion (*insert*) and extraction (*extract*) instructions can work with either *bytes* and *halfwords* and signed and unsigned data.

- Increase the clock frequency. This improvement can be considered since the previous modification simplifies the design of the interface of the processor core with its cache.

How much should the computer clock frequency be increased to make the incorporation of the aforementioned modifications worthwhile?

Solución:

The increase in the number of instruction due to the change in the ISA must be evaluated. Once this information is computed, the equation of the execution time of programs will enable the obtention of the increment of the clock frequency compensating the change.

The number of added instructions depends on the frequency of instructions accessing memory and the type of data being manipulated. For instance, the frequency f_{hb} representing the percentage of load instructions referring to *bytes* and *halfwords* (*lb*, *lbu*, *lh* and *lhu*) is 11% of all loads, and $2/3$ of memory accesses are loads. Since memory accesses are 36% of instructions, one can express this frequency as follows: $f_{lhb} = 2/3 \cdot 0.11 \cdot 0.36$ with respect to all instructions. Similarly, one can compute the frequency of stores affected by the change as $f_{shb} = 1/3 \cdot 0.11 \cdot 0.36$

Each deleted load is replaced by 2 instructions: one load *lw* and the corresponding *extract* instruction. On the other hand, each deleted store is replaced by three instructions: load *lw*, insertion *insert* and store *sw*.

Final balance: the gross increment resulting from the replacement of load instructions is $0.11 \cdot \frac{2}{3} \cdot 0.36 \cdot 1 = 2.6\%$, and the one resulting from the replacement of stores is $0.11 \cdot \frac{1}{3} \cdot 0.36 \cdot 2 = 2.6\%$

The number of instructions incremented is 5.2%

From the equation of the program execution time one can conclude that it is necessary an increment of, at least, 5.2% of the clock frequency.

□

Exercise 1.15.

A team of computer architects considers the improvement of the design of the instructions set of a *MIPS*- like computer. The considered modification consist in including in the instructions set instructions to handle the stack: `push` and `pop`. The existence of these new instructions will enable the replacement of sequences of instructions such as:

```
...
sw r1,0[sp]    ; push r1 y r2
sw r2,-4[sp]
subi sp,sp,8
...
lw r4,0[sp]    ; pop r4
addi sp,sp,4
...
```

by the following ones:

```
...
push r1  ; push r1 y r2
push r2
...
pop r4   ; pop r4
...
```

It is known this computer only uses the stack to support subprogram calls. A program call requires passing the incoming parameters and saving and restoring the values of registers used within subprograms for local variable management.

Main program:

```
...
sw r1,0[sp]      ; subprogram call
sw r2,-4[sp]     ; (two parameters)
subi sp,sp,8
jal subprograma
addi sp,sp,8
...
```

Subprogram:

```
; Subprogram entry point
sw r1,0[sp],     ; create space for three
sw r2,-4[sp]     ; local variables
sw r3,-8[sp]
sub sp,sp,#12
...              ; start the code ...
...
lw r3,0[sp]      ; return to the main program
lw r2,4[sp]
lw r1,8[sp]
add sp,sp,#12
jr r31
```

According to results obtained from benchmarking experiments, 1% of instructions are subprogram calls. Regarding the number of local variables and parameters used in subprograms, measures states that:

Loc. var	%	Parámeters	%
0	30	0	45
1	25	1	20
2	20	2	15
3	15	3	10
4	10	4	10

In order to implement the two instructions it is necessary to reduce the clock frequency. It is known that such modification will not impact the average CPI. If in the original design the clock frequency was 100 MHz, which should be the minimum clock frequency of the modified design to make the incorporation of the aforementioned instructions worthwhile?

Solución:

The proposed change in the ISA impacts on two factors of the equation of the execution time: the number of instructions I and the clock cycle t_C. Let's call T_{ex} the execution time exhibited by the original design, and T'_{ex} the execution time after applying the proposed change.

Let's the impact on the number of instructions of the modified I' wrt the original I:

- Parameter passing: an instruction is earned if there is any parameter. This happens in 55% of the procedure calls.
- Local variables: Two instructions are earned if there is any local variable. This happens in 70% of the cases.

The number of instructions is $I' = I \cdot (1 - (0.55 + 2 \cdot 0.7) \cdot 0.01) = 0.98 \cdot I$

Thus the condition that must be satisfy the clock of the modified processor is

$$\frac{T_{ex}}{T'_{ex}} = \frac{I \cdot 1 \cdot 10}{0.98 \cdot\ t'_C} > 1$$

So to keep the execution time, the clock of the modified design must work at 98 MHz

□

Exercise 1.16.

Study the interest of including a new indexed addressing mode for *load* and *store* instructions in the *MIPS*. The address of the operand located in memory is computed by adding the contents of two registers and a displacement of 11 bits. As a result the following sequence of instructions:

```
dadd r1,r1,r2
ld rd,o(r1)
```

can be replaced by this other one:

```
ldi rd,o(r1,r2)
```

The CPI is not affected, but the clock period of the improved MIPS is 5% longer than the original. Measures obtained from the original MIPS reflects that 24 % of the executed instructions are of type *load/store*, and 10% of them can be replaced by the new instructions.

Analyze which of both machines is faster and quantify its speed wrt the slowest machine.

Solución:

In the original processor, the execution time of the considered program is:

$$T_{ex} = I \cdot CPI \cdot t_c$$

After the modification, the execution time will become

$$T'_{ex} = I' \cdot CPI' \cdot t'_c = (1 - 0.24 \cdot 0.1) \cdot I \cdot CPI \cdot (1.05 \cdot T_{clk})$$

The speed-up resulting from the modification will be:

$$S = \frac{T'_{ex}}{T_{ex}} = 1.024$$

Conclusion: the original processor is 2.4% faster than the improved one.

□

Ejercicio 1.17.

MIPS64 SIMD extensions are known as *MIPS64 SIMD Architecture* (MSA). They add a new register file including 32 registers of 128-bit each, named `w0`, `w1`, etc. Depending on the instruction, each register can be handled as a 16-byte vector (8 bits per vector element), as a vector of 8 *halfwords* (16 bits per vector element), as a 4-*words* vector (32 bits per element), or as a 2-*doublewords* vector (64 bits per element).

In order to specify the type of element, SIMD instructions should include one of the following sufixes: `.b`, `.h`, `.w`, or `.d`; for referring to bytes, halfwords, words, and doublewords, respectively. So, the instruction `addv`**`.w`** `w5,w1,w2` adds all 4 **word** elements in vector registers `w1` and `w2` (the first element of `w1` is added to the first element of `w2`, the second one to the second one, and so on). Then, the resulting vector components are stored into register `w5`.

MSA extensions wants to be used in order to accelerate the execution of the following algorithm, which computes the vector operation $\vec{z} = A \cdot \vec{x} + \vec{y}$:

```
    addi r10,r0,N    ; N is the vector size
    addi r11,r0,0
    addi r12,r0,A
loop:
    lw r20,X(r11)
    lw r21,Y(r11)
    mul r20,r20,r12
    add r21,r21,r20
    sw r21,Z(r11)
    addi r11,r11,+4
    addi r10,r10,-1  ; 1 iteration processes 1 element of vectors x⃗, y⃗, z⃗
    bne r10,r0,loop
```

In order to cope with that goal two versions of the original algorithm, both using MSA extensions, are implemented. The first one (MSA1) executes the following code (MSA instructions are bolded):

```
    addi r10,r0,N    ; N is the vector size
    addi r11,r0,0
    ldi.w w12,A
loop:
    ld.w w20,X(r11)
    ld.w w21,Y(r11)
    mulv.w w20,w20,w12
    addv.w w21,w21,w20
    st.w w21,Z(r11)
    addi r11,r11,+16
    addi r10,r10,-4  ; 1 iteration processes 4 element of vectors x⃗, y⃗, z⃗
    bne r10,r0,loop
```

Considering that each loop iteration in MSA1 processes 4 elements (in stead of only 1 in the original algorithm), answer the following questions:

1. Assume N is a multiple of 4, how many instructions does MSA1 execute in total?

2. Considering that the CPI of all instructions is 1 and MSA extensions do not affect the processor frequency, which is the speed up resulting from using MSA1 in stead of the original algorithm?

3. A second version of the algorithm (MSA2) improves MSA1 by using a the instruction maddv, which La segunda versión (MSA2) mejora MSA1 mediante el uso de la instrucción `maddv`, which combines instructions `mulv.w` and `addv.w`. As a result, the following code:

```
mulv.w w20,w20,w12
addv.w w21,w21,w20
```

becomes:

```
maddv.w w21,w20,w12
```

However, the CPI of the instruction `maddv.w` is 3 due to data hazards with `ld.w` instructions, and the fact that the instruction carries out with 2 operations. Attending to this, how much should be increased the processor frequency to make the execution of MSA2 better than the one of MSA1?

Solución:

1. How many instructions does MSA1 execute in total?

$$I_{MSA1} = 3 + \frac{8 \cdot N}{4} \approx 2 \cdot N$$

2. which is the speed up resulting from using MSA1 in stead of the original algorithm?

$$T_{MSA1} = I_{MSA1} \cdot CPI \cdot t = 2 \cdot N \cdot t$$

On the other hand:

$$I_{orig} = 3 + 8 \cdot N \approx 8 \cdot N \rightarrow T_{orig} = 8 \cdot N \cdot t$$

As a result:

$$S = \frac{T_{orig}}{T_{MSA1}} = \frac{8 \cdot N \cdot t}{2 \cdot N \cdot t} = 4$$

3. How much should be increased the processor frequency to make the execution of MSA2 better than the one of MSA1?

$$I_{MSA2} = 3 + \frac{7 \cdot N}{4} \approx 1.75 \cdot N$$

$$CPI_{MSA2} \approx \frac{6}{7} \cdot 1 + \frac{1}{7} \cdot 3 = \frac{9}{7} \approx 1.28$$

$$T_{MSA2} = I_{MSA2} \cdot CPI_{MSA2} \cdot t_{MSA2} = 1.75 \cdot N \cdot 1.28 \cdot t_{MSA2} = 2.24 \cdot N \cdot t_{MSA2}$$

MSA2 will offer better performance than MSA1 when:

$$T_{MSA2} < T_{MSA1} \rightarrow 2.24 \cdot N \cdot t_{MSA2} < 2 \cdot N \cdot t \rightarrow \frac{2.24}{2} < \frac{t}{t_{MSA2}} \rightarrow 1.12 < \frac{t}{t_{MSA2}}$$

As a result, the clock must be at leat 12% faster.

□