EXERCISES

Problem 1.1

Two improvements are considered to a base machine with a Load/Store ISA and in which floating-point arithmetic instructions are implemented by software handlers. The first improvement is to add hardware floating-point arithmetic units to speed up floating-point arithmetic instructions. It is estimated that the time taken by each floating-point instruction can be reduced by a factor 10 with the new hardware. The second improvement is to add more first-level data cache to speed up the execution of Loads and Stores. It is estimated that, with the same amount of additional on-chip cache real-estate as for the floating-point units, Loads and Stores can be speeded up by a factor 2 over the base machine.

Let F_{fp} and F_{ls} be the fraction of execution time spent in floating-point and Load/Store instructions respectively. The executions of these two sets of instructions are non overlapping in time.

a. Using Amdahl's speedup, what should be the relation between the fractions F_{fp} and F_{ls} such that the addition of the floating-point units is better than the addition of cache space.

b. Suppose that, instead of being given the values of fractions F_{fp} and F_{ls}, you are given the fraction of floating-point instructions and the fraction of Loads and Stores. You are also given the average number of cycles taken by floating-point operations and Loads/Stores. Can you still find out which improvement is better based on these numbers? Explain why and how. Can you still estimate the maximum speedups for each improvement using Amdahl's law? Why?

c. What are fractions F_{fp} and F_{ls} such that a speedup of 50% (or 1.5) is achieved for each improvement deployed separately?

d. It is decided to deploy the floating point unit first and to add cache space later on. In the original workload, fractions F_{fp} and F_{ls} are 30% and 20% respectively. What is the maximum speedup obtained by upgrading to the floating point units? Assuming that this maximum speedup is achieved by the floating-point unit upgrade, what is the maximum speedup of the cache upgrade with respect to the floating point unit upgrade?

Problem 1.2

A multiprocessor machine has 1024 processors. On this machine we map a computation in which N iterate values must be computed and then exchanged between the processors. Values are broadcast on a bus after each iteration. Each iteration proceeds in two phases. In the first phase each processor computes a subset of the N iterates. Each processor is assigned the computation of K=N/P iterates where P is the number of processors involved. In the second, communication phase each processor broadcasts its results to all other processors, one by one. Every processor waits for the end of the communication phase before starting a new computation phase. Let T_c be the time to compute one iterate and let T_b be the time to broadcast one value on the bus. We define the computation-to-communication ratio R as T_c/T_b. Note that, when P=1, no communication is required.

At first, we use the premise of Amdhal's speedup (i.e., the same workload spread across an increasing number of processors). Under these conditions:
a. Compute the speedup as a function of P and R, for K=1,2,..1024.
b. Compute the maximum possible speedup as a function of P and R.
c. Compute the minimum number of processors needed to reach a speed up greater than 1 as a function of P and R?

Second, we use the premise of Gustasfon's law, namely that the uniprocessor workload grows with the number of processors so that the execution time on the multiprocessor is the same as on the uniprocessor. Assume that the uniprocessor workload computes 1024 iterates.

d. What should be the size of the workload (as a number of iterates) when P processors are used, as a function of P and R. Pick the closest integer value for the number of iterates.

e. Reconsider a-c above in the context of growing workload sizes, according to Guvstafon's law.

Third, we now consider the overhead needed to broadcast values over the bus. Because of software and bus protocol overheads, each bus transfer requires a fixed amount of time, independent of the size of the transfer. Thus the time needed to broadcast K iterate values on the bus by each processor at the end of each iteration is now $T_o + K \text{x} T_b$.

f. Using constant workload size assumption (as in Amdhal's law), what the maximum possible speedup?

g. Using growing workload size assumption (as in Gustafson's law), what is the maximum possible speedup?

Problem 1.3

This problem is about the difficulty of reporting average speedup numbers for a set of programs or benchmarks. We consider three ways of reporting average speedups of several programs:
- Take the ratio of average execution times, S_1
- Take the arithmetic means of speedups, S_2
- Take the harmonic means of speedups, S_3
- Take the geometric means of speedups, S_4

Consider the two improvements of a base machine in Problem 1.1, one improving floating-point performance and one improving memory performance. Three programs are simulated: one with no floating-point operations (Program 1), one dominated by floating-point operations (Program 2) and one with balance between memory accesses and floating-point operations (Program 3). The execution time of each program on the three machines is given in Table 1.4.

Table 1.4. Execution times of three programs

Machines	Program 1	Program 2	Program 3
Base machine	1sec	10msec	10sec
Base + FP units	1sec	2msec	6sec
Base + cache	0.7sec	9msec	8sec

a. Compute the four average speedups to the base machine for both improvements. Which conclusions would you draw about the best improvement if you were to consider each average speedup individually?

b. To remove the bias due to the difference in execution times, we first normalize the execution

time of the base machine to 1, yielding the normalized execution times in Table 1.5.

Table 1.5. Normalized execution times of three programs

Machines	Program 1	Program 2	Program 3
Base machine	1	1	1
Base + FP units	1	0.2	0.6
Base + cache	0.7	0.9	0.8

Compute the four average speedups to the base machine for both improvements. Which conclusions would you draw about the best improvement if you were to consider each average speedup individually?

Problem 1.4

In a machine M1 clocked at 100MHz it was observed that 20% of the computation time of integer benchmarks is spent in the subroutine Multiply(A,B,C) which multiplies integer A and B and returns the result in C. Furthermore, each invocation of Multiply takes 800 cycles to execute. To speed up the program it is proposed to introduce a new instruction MULT to improve the performance of the machine on integer benchmarks. Please answer the following questions, if you have enough data. If there is not enough data simply answer "not enough data".

a. How many times is the Multiply routine executed in the set of programs?

b. An implementation of the MULT instruction is proposed for a new machine M2. MULT executes the multiplication in 40 cycles (which is an improvement over the 800 cycles needed in M1.) Besides the Multiplies, all other instructions which were not part of the multiply routine in M1 have the same CPI in M1 and M2. Because of the added complexity however, the clock rate of M2 is only 80MHz? How much faster (or slower) is M2 over M1?

c. A faster hardware implementation of the MULT instruction is designed and simulated for a proposed machine M3, also clocked at 80MHz. A speedup of 10% over M1 is observed. Is this possible or is there a bug in the simulator? If it is possible, how many cycles does the MULT instruction take in this new machine? If it is not possible, why is this so?

Problem 1.5

In this problem we compare two styles of conditional branches. We start with the MIPS BEQ. If you do not know of this class of instructions please refer to chapter IV. This instruction is used in conjunction with the SLT instruction. For example, the following code branches when the value in R1 is less than the value in R2:

```
SLT R3,R1,R2              /set R3 to 1 if R1<R2 otherwise set R3 to 0
BNEZ R3, target           /branch to target when R3=0
```

This takes two instructions. Alternatively we could use branch instructions that execute both the test and the branch itself. This approach saves one arithmetic/logic instruction. However, because it is more complex, it increases the cycle time.

For example:

```
BLT R1,R2,target          /branch to target if R2<R1
```

It is suggested to add instructions such as BLT to the MIPS ISA (BGE, BGT, etc).

a. First consider the base machine with BEQZ and BNEZ only. The dynamic instruction mix in Table 1.6 is observed. Compute the average CPI.

b. After adding BLT-type branch instructions the clock cycle has been raised by 5% but the compiler is able to remove the SLT instructions associated with 50% of branches. Compute the new average CPI.
c. Is the addition of BLT-like instructions a good idea?

Table 1.6. Dynamic instruction mix in the base machine

Instructions	Frequency	Cycles
Arithmetic/logic	40%	1
Loads	25%	2
Stores	10%	1
Branches (Untaken)	8%	1
Branches (Taken)	12%	3
Miscellaneous	5%	1

Problem 1.6

A baseline microprocessor is enhanced as follows.

1) The core is replicated 16 times to form a 16-way CMP.

2) A floating-point co-processor is added to each core. This co-processor speeds up all floating-point operations in each core by a factor 4 and is attached to the core. While the floating point co-processor is active, the host core is idle and the presence of the co-processor does not affect instructions that are not part of floating point operations.

We observe the following on the **NEW** machine:
1) the floating-point co-processor is used during 30% of the execution time of each core.
2) 25% of the time only one core is active and 75% of the time the four cores are active.

What is the speedup of the new machine over the base machine?