CPSC 418	Advanced Computer Architecture	Spring 1995

Midterm Solution

Problem 1 (8 Points)

To keep up with the market (*e.g.* other companies), system performance needs to double every year. You are considering a design improvement that will increase performance by 10% but delay the release date of a system by two months.

In terms of the performance of the system with respect to other machines that will be on the market when it will be released, is it better to release the system on its current schedule, or add the improvement and move back the release by two months?

Performance must double each year, therefore the performance n years from now must be 2^n.

Two months from now is $\frac{2}{12} = \frac{1}{6}$ years.

$2^{\frac{1}{6}} = 1.122$ Therefore, two months from now, performance must be 12.2% greater than now.

The design improvement will only add a 10% performance increase. $10\% < 12\%$, so **it is better to stay with the current schedule**.

Problem 2 (10 Points)

This problem refers to the pipeline shown in Figure ??, where the stages perform their normal functions. In particular:

- Branch prediction is performed in the Decode stage.
- Branch instructions test a general purpose register.

2(a) (5 Points) What is the penalty (in number of cycles) for a branch mispredict?

Branch prediction happens in stage 2 (Decode).

Branches are resolved in stage 4 (Execute).

There are two cases to consider: predict branch taken and predict branch not-taken.

If a branch is predicted to be taken, then the following instruction (from address Bra+4), which is in FetchInst when the branch is in Decode, must be killed (resulting in a one-cycle penalty, but this penalty is not the concern of this question). If the prediction is wrong, (*i.e.* the branch is not taken) we will lose two cycles for the instructions fetched along the path of the branch. These instructions will be in the FetchInst and Decode stages when the branch instruction is in Execute. **Thus there is a two cycle penalty for a branch mispredict if the branch is predicted to be taken.**

If a branch is predicted to not be taken, then the instruction in FetchInst is correct, and we continue execution. If the prediction is wrong (*i.e.* the branch is taken), we will lose three cycles for the instructions fetched along the non-branch path (*e.g.* at addresses Bra+4, Bra+8, and Bra+C). These instructions will be in the FetchInst, Decode, and FetchOps stages when the branch instruction is in Execute. **Thus there is a three cycle penalty for a branch mispredict if the branch is predicted to be not-taken.**

Note: no one wrote down a full explanation similar to the above. I gave equal credit to answers of two and three clock cycles if the answer was explained.

2(b) (5 Points) Must all microprocessors in this architectural family have the same penalty for branch mispredictions? Justify your answer.

> The time from branch prediction to branch resolution varies between microprocessors in the same architectural family.
>
> Architectural issues define *correctness* of code, branch prediction penalties are a *performance* issue.

Problem 3 (10 Points)

Two microprocessors (R, for "rabbit" and T for "turtle") from different architectural families have the same memory hierarchy (levels of cache, sizes, speed of memory, *etc.*).

The clock speed for Micro-R is 1.5 times faster than the clock in Micro-T.

For the SPEC benchmarks on Micro-T:

3 cycles	average memory latency
2.4 cycles	average CPI for load instruction (including memory access time)

3(a) (5 Points)

Do you have enough information to calculate the average memory latency in clock cycles for the SPEC benchmarks running on Microprocessor R? If so, do so. If not, explain what additional information you need.

> The memory latency is the time for a read or write to/from memory to complete. This latency as measured in real time (wall time) is determined soley by the memory hierarchy and the size and distribution of code and data. By running roughly the same program (*i.e.* same source code, but different compiled code, because two different architectural families), we estimate that distribution and size of the code and data will be fairly similar for the two machines.
>
> The memory latency in *real time, or wall time* for the two microprocessors will be the same. Thus:
>
> $\mathrm{Clock_R} = Clock_T/1.5$, because the clock for Rabbit is faster, and therefore has a shorter period.

$$
\begin{aligned}
\mathrm{Latency_T} * \mathrm{Clock_T} &= Time \\
\mathrm{Latency_R} * \mathrm{Clock_R} &= Time \\
\\
\mathrm{Latency_R} * \mathrm{Clock_R} &= \mathrm{Latency_T} * \mathrm{Clock_T} \\
\mathrm{Latency_R} &= \frac{\mathrm{Latency_T} * \mathrm{Clock_T}}{\mathrm{Clock_R}} \\
&= \frac{\mathrm{Latency_T} * \mathrm{Clock_T}}{\frac{1.}{\mathrm{Clock_T}}} \\
&= \mathrm{Latency_T} * 1.5 \\
&= 3 * 1.5 \\
&= 4.5 cycles
\end{aligned}
$$

3(b) (5 Points)

Do you have enough information to calculate the average CPI for a load instruction (using the SPEC benchmarks) for Microprocessor R? If so, do so. If not, explain what additional information you need.

> No, we don't have enough information. We need to know the internal configuration of the Micro-R pipeline, the different addressing modes that are supported, *etc.*, *etc.*. The CPI for a load is determined both by the memory hierarchy and how the instruction goes through the pipeline.

Problem 4 (12 Points)

In the table below list:

1. three different architectures
2. two different microprocessors for each architecture

Note, these are just examples.

	Intel x86	MIPS Rx000	PowerPC
Architecture	80386	R3000	601
Microprocessor 1	80486	R4400	603
Microprocessor 2	Pentium	T5 / R10000	620

For each architecture, list one distinguishing feature of the architecture or of a microprocessor in the architecture.

- Distinguishing feature for architecture 1
  CISC instruction set, special purpose registers, world's best selling architecture.
- Distinguishing feature for architecture 2
  MIPS T5 has a large data cache, capable of issuing four instructions per cycle, uses register renaming. Rx000 architecture has delayed branch and delayed load instructions.
- Distinguishing feature for architecture 3
  PowerPC (and IBM Power) are RISC, but have some complex instructions, such as decrement-and-branch and floating-point multiply/add.

Problem 5 (35 Points)

You have a workstation that is used almost exclusively for floating-point intensive programs, such as Spice.

Currently, it does not have a cache, and you are using a simple, non-optimizing compiler.

You can afford to spend up to \$700 on upgrading the computer. You can either buy SRAM for a cache or an optimizing compiler, which sells for \$500.

5(a) (25 Points)

Using the tables and figures at the end of the test, find the relative speedup for using the optimizing compiler and upgrading the cache with the maximum amount of SRAM you can buy for \$700.

From H&P pp9:
Speedup $= \frac{\text{Enhancedperformance}}{\text{Unenhancedperformance}}$
The optimizing compiler will reduce the dynamic instruction count.

$$\begin{aligned} \text{Unoptimizedinstructioncount} &= InstBRA + InstLS + InstALU \\ &= 6 + 52 + 59 \\ &= 117 \end{aligned}$$

$$\begin{aligned} \text{Optimizedinstructioncount} &= InstBRA + InstLS + InstALU \\ &= 4 + 31 + 38 \\ &= 73 \end{aligned}$$

Instruction count is inversely proportional to performance.

Speedup for cache = 117/73 = 1.60

Maximum amount of SRAM for $700 is 1024k for $600.

Going from no cache to 1M cache takes CPI from 2.6 to 1.2.

CPI is inversely proportional to performance.

Speedup for cache = 2.6/1.2 = 2.167

Note, due to typos, errors, and general confusion, I was very lenient in terms of the definition of "speedup" that people used.

The correct formula for "X is n percent faster than Y" is:
$\frac{TimeY - TimeX}{TimeX}$
there is a typo in the solutions for HW-5 with this formula. Also, the definitions of "speedup" and "n percent faster" are related, but not identical.

5(b) (10 Points)

In terms of the price/performance ratio, is it better to buy the cache or the compiler?

Price / Performance for compiler: $\frac{\$500}{1.60} = 312$

Price / Performance for cache: $\frac{\$600}{2.17} = 277$

The price/performance ratio for the cache is better, so we should buy the memory.

Problem 6 (25 Points)

For the pipeline in Figure ??:

- divides spend a total of two cycles in the divide stage.
- load/stores use the load/store stage once

- stores do not use the writeback stage

Using the standard style used in class, complete the execution table in Figure ?? for the following program:

```
; R1=A, R2=B, R3=C, R4=D, R5=E
1000   ADD R1, R3, R1
1004   DIV R3, R2, R1
1008   ADD R3, R1, R5
100C   LDW R2, R5
```

	0	1	2	3	4	5	6	7	8	9	10	11
PC	1000	1004	1008	100C								
FI		ADD R1,R3,R1	DIV R3,R2,R1	ADD R3,R1,R5	LDW R2,R5							
DI			R1=R3+R1	R3=R2/R1	R3=R1+R5	R2=M[R5]	R2=M[R5]	R2=M[R5]				
FO				R1=S1+S2	R3=S1/BY1	R3=S1+S2	R3=S1+S2	R3=S1+S2	R2=M[S1]			
EX					R1=F	R3=__			R3=I	R2=___		
LS											R2=J	
DIV							R3=__	R3=H				
WB						R1=F			R3=H	R3=I		R2=J
BY1					F							
BY2											J	
S1				C	B	F	F	F	E			
S2				A	A	E	E	E				
R1						F						
R2												J
R3									H			

Figure 1: Execution table solution