CS704 – Advanced Computer Architecture-II

Solution to Assignment 3

Instructions to Solve Assignments

The purpose of assignments is to give you hands on practice. It is expected that students will solve the assignments themselves. Following rules will apply during the evaluation of assignment.

- Cheating from any source will result in zero marks in the assignment.
- Any student found cheating in any two of the assignments submitted will be awarded "F" grade in the course.
- No assignment after due date will be accepted.

Question 1: Total Points (10+10=20)

Consider the speculative processor discussed in lectures. Since the reorder buffer contains a value field, you might think that the value field of the reservation stations could be eliminated.

(a) Show an example where this is the case and an example where the value field of the reservation stations is still needed. Use the speculative processor shown in Figure 1.1. Show MIPS code for both examples. How many value fields are needed in each reservation station?

(b) Find a modification to the rules for instruction commit that allows elimination of the value fields in the reservation station. What are the negative side effects of such a change?

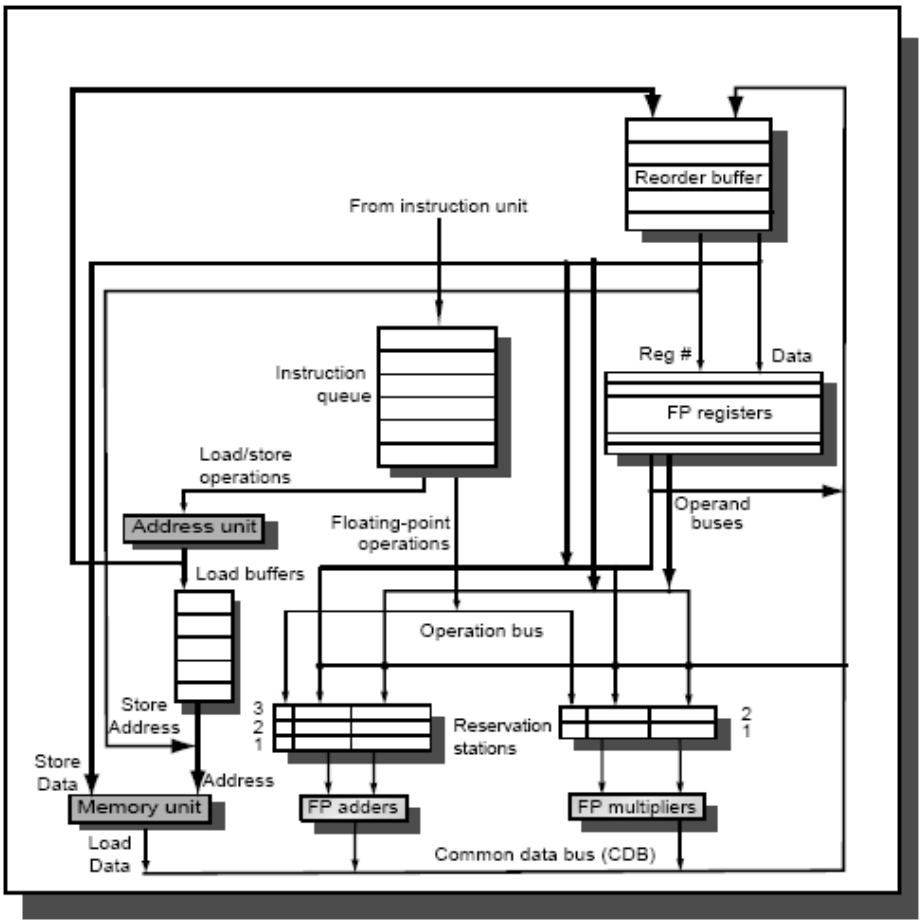

FIGURE 1.1 The basic structure of a MIPS FP unit using Tomasulo's algorithm and extended to handle speculation.

Solution:

(a) The following example shows a segment of code where issued instructions read their operand values directly from the ROB. This example assumes that the divide executes immediately and that the divide takes significantly long to

perform than the other FP operations. The register values of F2, F3, F4, and F6 are present in the register file for immediate use by the instructions. While the divide executes, the add and multiply issue. The multiply starts immediately when the divide finishes since the add finishes before the divide.

DIV.D F1, F2, F3
ADD.D F5, F6, F4
MULT.D F9, F1, F5

If the add moves to before the divide (as shown in the code below), then the ROB entry associated with the add retires before the divide finishes. The multiply, already in the reservation station, would have to read the value of F5 from the register file. This datapath does not exist in the reservation station architecture. Unless the rules of commit change, reservation stations require value storage.

ADD.D F5, F6, F4
DIV.D F1, F2, F3
MULT.D F9, F1, F5

(b) In order to remove the value fields from the reservation stations, a rule must be added to the commit phase. The ROB entry allocated to an instruction must not be permitted to commit until all issued instructions that are data dependant on that particular allocated space have start to execute. In the example above where the add is first, the add instruction may not commit until the multiply starts to execute. The ROB may start to fill with these lingering instructions. Once filled, issue must stall.

Question 2: Total Points (5+5+5+5=20)

For this problem, assume that you have a processor with a cache connected to main memory via a bus. A cache access takes 1 cycle. A successful cache access (a hit) finishes within that cycle. On an unsuccessful access (a miss) additional work must be performed to fetch a block from main memory over the bus. A bus transaction consists of one cycle to send the address to memory, four cycles of idle time for main-memory access, and then one cycle to transfer each word in the block to the cache. (Assume that the processor continues execution only after the last word of the block has arrived.) In other words, if the block size is B words (at 32 bits/word), a cache miss will cost 1 + 4 + B cycles.
Following table gives the average cache miss rates of a 1 megabyte cache for various block sizes.

Block size(B)	Miss ratio(m) %
1 word	3.40%
4 words	1.00%

16 words	0.40%
64 words	0.25%
256 words	0.19%

(a) Write an expression for the average memory access time for a 1-MByte cache and a B-word block size (in terms of m and B).
(b) What block size yields the best average memory access time?
(c) If bus contention adds three cycles to the main-memory access time, which block size yields the best average memory access time?
(d) Ignoring bus contention, if the bus width is doubled to 64 bits, what is the optimal block size?

Solution:

(a)

Average Memory Access Time = 1 + m(1 + 4 + B) cycles

(b)

For B == 1 1 + .0340(1+4+1) = 1.204
For B == 4 1 + .0100(1+4+4) = 1.09
For B == 16 1 + .0040(1+4+16) = 1.084 block size of 16 is best
For B == 64 1 + .0025(1+4+64) = 1.1725
For B == 256 1 + .0019(1+4+256) = 1.4959

(c)

For B == 1 1 + .0340(3+1+4+1) = 1.306
For B == 4 1 + .0100(3+1+4+4) = 1.12
For B == 16 1 + .0040(3+1+4+16) = 1.096 block size of 16 is best
For B == 64 1 + .0025(3+1+4+64) = 1.18
For B == 256 1 + .0019(3+1+4+256) = 1.5016

(d)

For B == 1 1 + .0340(1+4+1) = 1.204
For B == 4 1 + .0100(1+4+2) = 1.07
For B == 16 1 + .0040(1+4+8) = 1.052 block size of 16 is best
For B == 64 1 + .0025(1+4+32) = 1.0925
For B == 256 1 + .0019(1+4+128) = 1.2527

Question 3: Total Points (10+5+10=25)

You have a computer with two levels of cache memory and the following specifications:

- Processor: 2GHz, 64-bit RISC CPU
- On-chip L1 caches
  - split instruction & data cache, blocking, single-ported
  - write-through & non-write allocate
  - 1 CPU cycle access time
  - Block size = 32 bytes
- Off-chip L2 cache off-chip
  - unified single-ported cache, blocking
  - write-back
  - 10 CPU cycles access time (L1 miss penalty) for both reads and writes
  - Block size = 32 bytes
- Main memory:
  - Bus: 64-bit data transfers at 400 MHz
  - Latency: 15+5+5+5 CPU cycles access time for 32 bytes (L2 miss penalty – includes latency of both DRAM and memory bus)

(a) What is the:

- Peak L1 data cache bandwidth available to CPU (assuming 0% L1 misses)?
- Peak L2 cache bandwidth available to L1 cache (assuming 0% L2 misses)?
- Main memory bandwidth available to L2 cache?

[Calculate the bandwidths in Gbytes/sec, i.e. 2^{30} bytes/sec.]

(b) You are given the following L1 cache statistics for a program executing on this system

Metrics	**Access Type:**				
	Total	**Instrn**	**Data**	**Loads**	**Stores**
Accesses	10000000	7362210	2637790	1870945	766845
Misses	52206	8466	43740	36764	6976

Words Read From Lower-levels	180920 (i.e. 45230 cache lines)
Words Written-back to Lower-levels	766845
Total Traffic	947765

How long does an average instruction take to execute (in ns), assuming 1 clock cycle per instruction in the absence of memory hierarchy stalls, no write buffering at the

L1 cache level, and 0% L2 miss rate? Ignore register dependencies between instructions.

(c) You are considering replacing the L2 cache with a victim cache. Given the information provided to you, compute a measure of "speed" for each alternative and indicate which is the faster solution. Justify the metric you choose to compare the two alternatives and state your assumptions. Assume the performance statistics are:

- L2 cache local miss ratio= 0.18
- Victim cache miss ratio = 0.23
- Victim cache transport time from L1 miss = 2 CPU clock

Solution:

(a)

Peak L1 data cache bandwidth:

8 bytes / 1 CPU cycle access time = 8 bytes/0.5ns = 16 Gbytes/sec

Peak L2 cache bandwidth:

32 bytes / 10 CPU cycle access time = 32 bytes/5ns = 6.4 Gbytes/sec

Peak memory bandwidth:

32 bytes / 30 CPU cycle access time = 32 bytes/15ns = 2.13 Gbytes/sec

(b)

Any instruction that hits in the cache will not be penalized by the misses (Unless of course there are data dependencies, but since the problem tells us to ignore this). Thus, we just need to find average miss penalty of an instruction since that will incur extra latency. Note, that since we have a 0% L2 miss rate, we never incur any main memory accesses.

Average instruction latency =

$latency_{ideal}$ + extra $latency_{stalls}$= 1 + instruction stall cycles + data stall cycles

Instruction stall cycles = ICache miss rate * L1 miss penalty

Data stall cycles = (load rate * load miss rate * L1 miss penalty) + (store rate * write penalty)

The L1 miss penalty is 10 cycles. Since L1 is write-through, we assume the write penalty is the same as the L1 miss penalty which is 10 cycles. Also, since the L2 is single-ported, there may be the case where an instruction miss and a data miss occurs at the same time and there will be a structural hazard. In that case, one of the misses will incur an extra 10 cycle penalty. However the chance of this happening is

almost zero: ICache miss rate * DCache miss rate = 0.115%*1.66% ~= 0%. Therefore we've neglected this in our equation.

The ICache miss rate is given in the table as: 8466/7362210 = 0.115%
The load rate is given in the table as: 1870945/7362210 = 25.41%
The load miss rate is given in the table as: 36764/1870945 = 1.965%
The store rate is given in the table as: 766845/7362210 = 10.42%

Thus,
Instruction stall cycles = 0.115% * 10 = 0.0115 cycles
Data stall cycles = 25.41% * 1.965% * 10 + 10.42% * 10 = 1.09 cycles
Average instruction latency = 1 + 0.0115 + 1.09 = 2.1015 cycles
So, average instruction time = Average latency * cycle time = 2.1015 * 0.5ns = 1.05ns

(c)

Given the information provided, it's clear that the common case is a cache read. The reads, i.e. instruction fetch and loads, account for (7362210+1870945)/10000000 = 92.3% of total L1 cache accesses. Also, the write miss rate is much lower than the load miss rate. Thus, one metric that we can use for comparison then is Average Memory Access Time (AMAT) of a read.
The L1 read miss rate = (instruction & load misses) / (instruction & load accesses) = (8466 + 36764) / (7362210 + 1870945) = 0.4899%

1. L2 Cache,
$AMAT_{L2}$ = 10 + L2 miss rate * 30 = 10 + 0.18*30 = 15.4 cycles
$AMAT_{L1}$ = 1 + L1 read miss rate * AMATL2 = 1 + 0.004899*15.4 = 1.0754 cycles

2. Victim Cache,
$AMAT_{VC}$ = 2 + VC miss rate * 30 = 2 + 0.23*30 = 8.9 cycles
$AMAT_{L1}$ = 1 + L1 read miss rate * $AMAT_{L2}$ = 1 + 0.004899*8.9 = 1.0436 cycles

So, it would seem that the victim cache would be a better choice in this case.

Question 4: Total Points (10)

A large amount (more than a third) of DRAM power can be due to page activation (see http://download.micron.com/pdf/technotes/ddr2/TN4704.pdf and http://www.micron.com/systemcalc). Assume you are building a system with 1 GB of memory using either 4-bank 512 Mbit × 4 DDR2 DRAMs or 8-bank 1 Gbit × 8 DRAMs, both with the same speed grade. Both use a page size of 1 KB. Assume DRAMs that are not active are in precharged standby and dissipate negligible power. Assume the time to transition from standby to active is not significant. Which type of DRAM would be expected to result in lower power? Explain why.

Solution:

The power required to drive the output lines is the same in both cases, but the system built with the x4 DRAMs would require activating banks on 18 DRAMs, versus only 9 DRAMs for the x8 parts. The page sizes activated on each x4 and x8 part are the same and take roughly the same activation energy. Thus, since there are fewer DRAMs being activated in the x8 design option, it would have lower power.

Question 5: Total Points (5+10=15)

Read the paper "**Future Cache Design using STT MRAMs for Improved Energy Efficiency: Devices, Circuits and Architecture**" and answer the following questions:

(a) What is STT MRAM and how is it different from SRAM?

(b) Evaluate STT MRAM and SRAM caches based on the following parameters:

- Cache utilization
- Energy consumption

Solution:

Research-paper based question.

Question 6: Total Points (10)

Read the paper **"Energy Reduction for STT-RAM Using Early Write Termination"** and elaborate how Early Write Termination (EWT) improves energy efficiency of STT-RAM cache?

Solution:

Research-paper based question.