Indian Institute of Technology, Delhi

FALL, 2012

COMPUTER ARCHITECTURE

Homework 3

(DEADLINE: November 9th 11:59 PM (HARD DEADLINE))

1 Questions

Write a small architecture simulator in Java that simulates the branch predictor, inorder pipeline, and the cache. The details are as follows:

1.1 Branch Predictor

Design a tournament predictor, which contains a GShare predictor, and a bimodal predictor. The GShare predictor will use 12 bits of the PC, and will have a 6 bit BHR. You need to XOR the BHR with lower 6 bits of the PC. The bimodal predictor, will use 10 bits of the PC.

1.2 Inorder Pipeline

Design an inorder pipeline, which has the following stages:

1. Fetch: Get an instruction from the trace file. If the instruction is a branch, then predict the branch. If the prediction is correct, go ahead. Otherwise, stall the pipeline for two cycles.
2. Decode: Process the instruction
3. Execute: Train the branch predictor
4. Memory Access: Send the request to the cache and wait for the reply.
5. Writeback:

You don't have to execute the program. You just have to simulate the timing. We will give you a trace file of the form:

```
pc mem_address [1|0|-1]
Example:
0xE3FF0987 0x9000324D -1
```

Here the pc and the memory address are in hex. For the third field, 0 stands for branch not taken, 1 stands for branch taken, and -1 stands for the fact that the instruction is not a branch. Here all the fields might not be relevant for each instruction. Use your judgement. Along with the trace file, we have another instruction file containing the list of instructions. This has the following form:

```
pc type rs rt rd
```

1. pc : program counter in hex
2. type : 0 – load, 1 – store, 2 – ALU instruction, 3 – branch
3. rs : source register 1
4. rt : source register 2
5. rd : destination register

Instruction Format

Instruction	Format
Load	ld rd, 100(rs)
Store	st rt, 100(rs)
ALU Instruction	inst rd, rs, rt
Branch	br rs

Note that the traces are of actually executed instructions. Hence, upon encountering a branch, you don't need to fetch the branch target separately. It is the next instruction in the trace.

1.3 Cache

Implement the following caches:

Cache	Size	Block Size	Associativity	Access(Hit) Time	Replacement Policy
L1	32 KB	32 bytes	2	1 cycle	FIFO
L2	2 MB	128 bytes	8	8 cycles	LRU

Assume that main memory is infinite in size, and the L2 miss penalty is 200 cycles.

2 What to submit

Form groups of three people, and implement all the three subsystems. You need to then attend a demo session. Submit a .jar file that takes two arguments. The first is the trace file, and the second is the instruction file.

We will run your jar file on our sample inputs. We need the following statistics: IPC, L1 local miss rate, L2 local miss rate, branch prediction accuracy. Print all four of them in one line. An example output line is of the form:

```
0.76 10% 5% 94%
```

CONTINUED