CMPUT 429 / CMPE 382 Term Exam B1

Monday, April 17/00
Mike MacGregor

2 hours, open book

Section 1. Short answer – 1.5 marks each, total of 30%

Use 1 or at most 2 sentences to answer each of the following questions.

1. What is the point of Amdahl's Law?

2. Define latency. Does pipelining reduce latency? Explain.

3. Define throughput. Does pipelining improve throughput? Explain.

4. How does an exception affect a pipeline?

5. What is the most significant difference between the DLX FP multiplier and the FP / integer divider?

6. Why are RAW hazards called true dependencies?

7. What is the benefit of loop unrolling?

8. When is software pipelining of greater benefit than loop unrolling?

9. What is the goal of a superscalar design?

10. What is the appropriate number of instructions to issue per cycle in a processor using Tomasulo's method? Why?

11. Why is cache memory used?

12. Why are L1 caches generally smaller and simpler than L2 caches?

13. What phenomenon makes caches useful?

14. If you had to choose between having L1 tags or L1 data on the same chip as the CPU, which would you choose and why?

15. In a cache, why do we use low-order address bits for the index and high-order bits for the tag instead of vice versa?

16. What are the two major performance figures to be considered in designing a main memory subsystem?

17. What is the main benefit of virtual memory?

18. Why is a TLB important?

19. Does RAID-0 improve availability? Explain.

20. What improvement does RAID-5 offer over RAID-1 ?

Section 2. Short calculation – 8 marks each, total of 40%

1. Look at Fig. 3.37 on p. 177 of the text (a copy is attached to the exam). First, in two or three sentences, and in your own words summarize what is being presented in this figure. Then answer the following: If we use static branch prediction, what is the effective size of a basic block?

2. Look at the block diagram for the Intel P6 family micro-architecture (the figure is attached to the exam). Is it possible for this processor to support precise exceptions ? Explain your answer.

3. You have a machine with four 1-bit branch predictors. These predictors are indexed using address bits two and three in the address of the branch being predicted (bits are numbered starting at zero). Look at the attached code labeled "Code for question 2.3". First, make a table listing the branches by address, and for each branch show which predictor it maps to. Next, assuming that all predictors are initialized "Taken" show the state of all four predictors at the end of the second iteration.

4. You have designed a machine with 8-way interleaved memory. Each bank supports from 2 to 8 SIMMs (single inline memory modules). Each SIMM supports from 2 to 8 DRAMs. The available DRAMs are 64 Mb x 1 or 256 Mb x 1. What is the minimum amount of memory that can be added to this system? What is the maximum amount of memory this system can carry?

5. Propose an experiment that you could perform that would time one or more loops to determine whether a particular machine has a write-back vs. write-through cache.

Section 3. Design problem – 30%

You have to choose one of three configurations for a Level 1 cache:

- Cache 1: Direct-mapped with one-word blocks.
- Cache 2: Direct-mapped with four-word blocks.
- Cache 3: Two-way set associative with four-word blocks.

You have measured the following miss rates:

- Cache 1: Instruction miss rate is 4%, data miss rate is 8%.
- Cache 2: Instruction miss rate is 2%, data miss rate is 5%.
- Cache 3: Instruction miss rate is 2%, data miss rate is 4%..

In your benchmark, 50% of the instructions contain a data reference. Assume that the cache miss penalty is (6+Block size in words). The CPI for this benchmark was measured on a machine with Cache 1, and was found to be 2.0.

a) Determine which configuration will spend the fewest cycles on cache misses.

b) With cache 1 or 2, the cycle time of the machine can be 2 ns, while cache 3 pushes the cycle time out to 2.4 ns. Which cache yields the fastest machine?

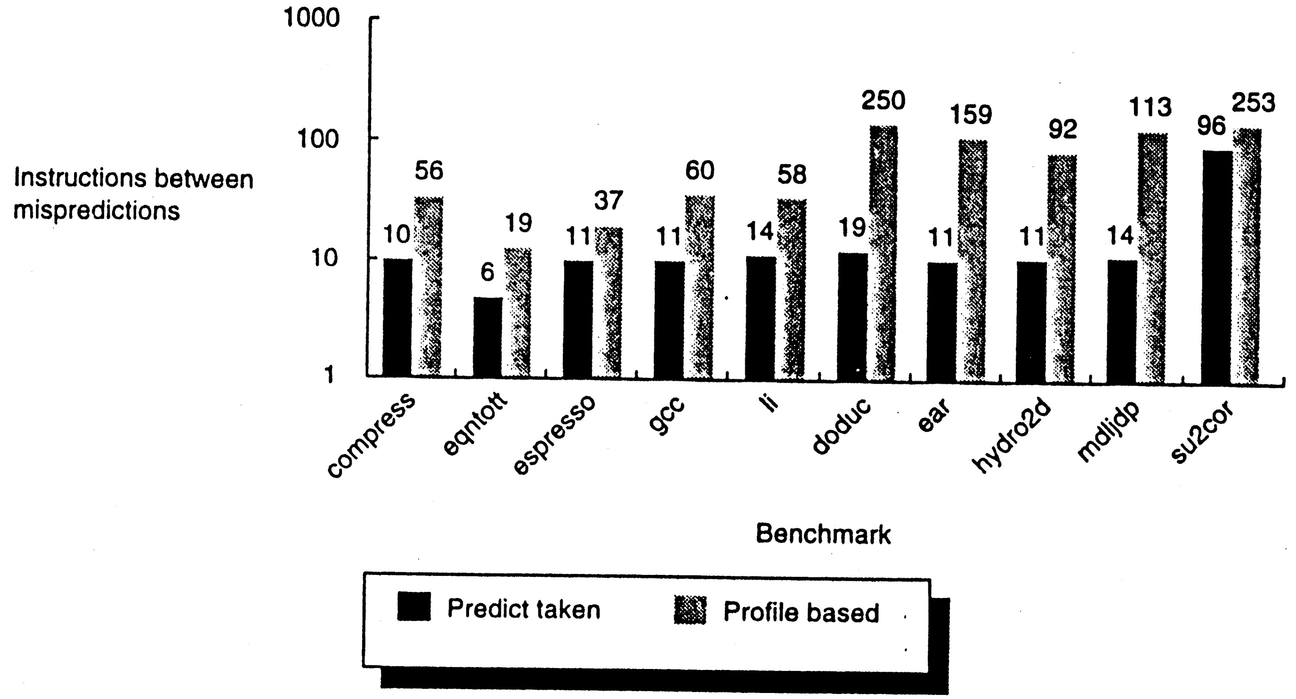
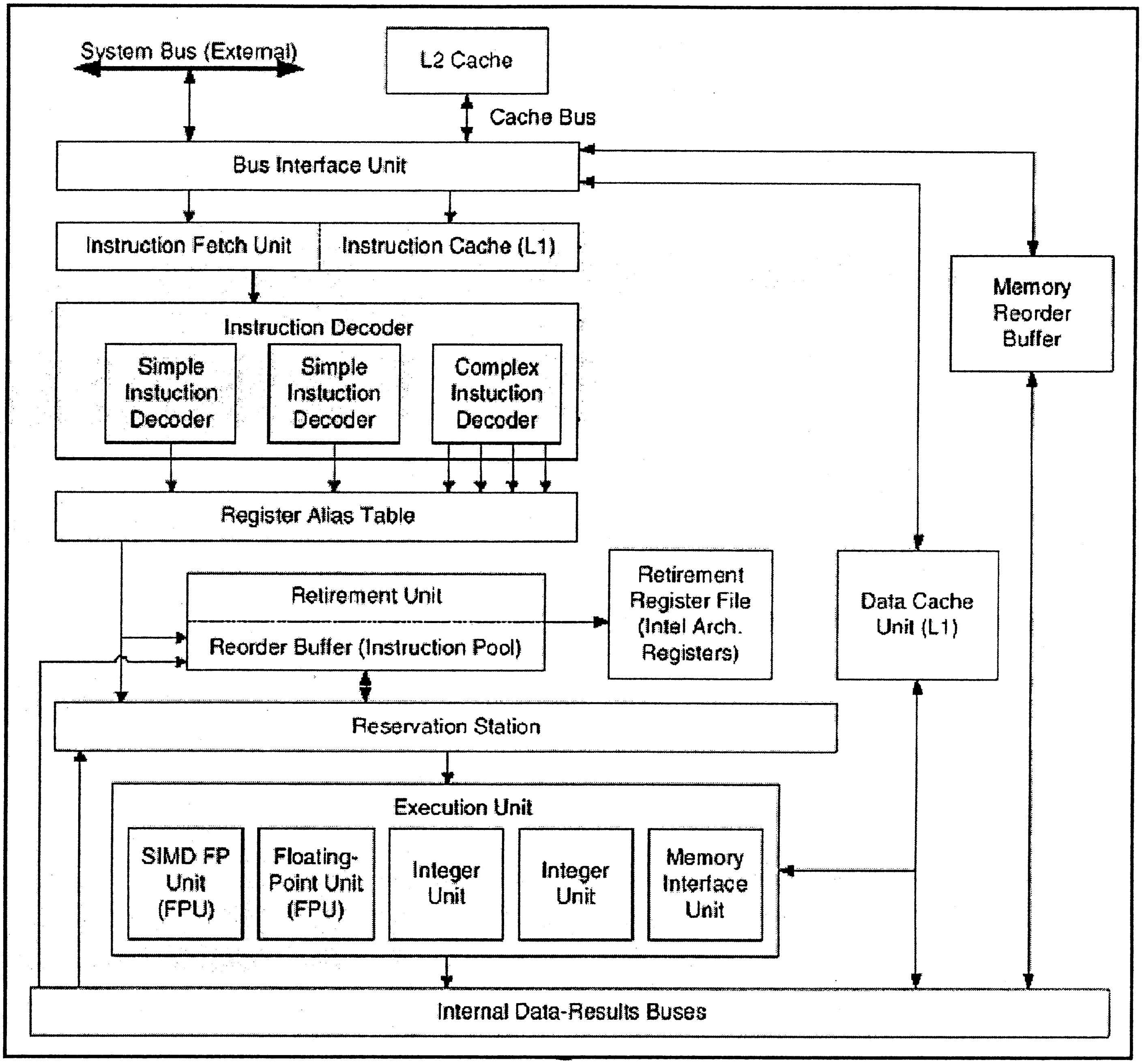


FIGURE 3.37 Accuracy of a predict-taken strategy and a profile-based predictor as measured by the number of Instructions executed between mispredicted branches and shown on a log scale. The average number of instructions between mispredictions is 20 for the predict-taken strategy and 110 for the profile-based prediction; however, the standard deviations are large: 27 instructions for the predict-taken strategy and 85 instructions for the profile-based scheme. This wide variation arises because programs such as su2cor have both low conditional branch frequency (3%) and predictable branches (85% accuracy for profiling), while eqntott has eight times the branch frequency with branches that are nearly 1.5 times *less* predictable. The difference between the FP and integer benchmarks as groups is large. For the predict-taken strategy, the average distance between mispredictions for the integer benchmarks is 10 instructions, while it is 30 instructions for the FP programs. With the profile scheme, the distance between mispredictions for the integer benchmarks is 46 instructions, while it is 173 instructions for the FP benchmarks.



Intel P6 family microarchitecture

Code for question 2.3

```
                        ;*** Initialization
0x00000160      main:           addi            r1,r0,0         ;r1 indexes one past the top
0x00000164                      addi            r2,r0,2         ;r2 is the current value

                        ;*** Can R2 be divided by a value in the table?
0x00000168      NextValue:      addi            r3,r0,0         ;r3 is index of entry tested
0x0000016c      Loop:           sub             r12,r1,r3       ;Are we at end of the table?
0x00000170                      addi            r4,r0,0         ;Clear r4
0x00000174                      beqz            r12, Endit      ;Branch if we're at the end
0x00000178                      addi            r4,r0,1         ;Otherwise set r4

0x0000017c      Endit:          beqz            r4,IsPrim       ;Branch - R2 is prime number

0x00000180                      lw              r5,Table(R3)    ;Put the entry into r5
0x00000184                      divu            r6,r2,r5        ;Divide r2 by r5
0x00000188                      multu           r7,r6,r5        ;Multiply the result by rr5
0x0000018c                      subu            r8,r2,r7        ;And subtract from r2
                                                ;If r8==0, r2 is evenly divisible by r5

0x00000190                      beqz            r8,IsNoPrim     ;Branch - not a prime
0x00000194                      addi            r3,r3,4         ;Increment the table index
0x00000198                      j               Loop

                        ;*** Write value into Table and increment index
0x0000019c      IsPrim:         sw              Table(r1),r2
0x000001a0                      addi            r1,r1,4

                        ;*** 'Count' reached?
0x000001a4                      lw              r9,Count
0x000001a8                      srli            r10,r1,2        ;Number of entries is index/4
0x000001ac                      sub             r11,r9,r10
0x000001b0                      beqz            r11,Finish      ;Branch if done

                        ;*** Check next value
0x000001b4      IsNoPrim:       addi            r2,r2,1         ;increment R2
0x000001b8                      j               NextValue

                        ;*** end
0x000001bc      Finish:         trap            0
```