CSE 471 Winter 2000

Computer Design and Organization

Midterm

Friday February 4th

NAME : ______________________

Do all your work on these pages. Do not add any pages. Use back pages if necessary. Show your work to get partial credit.

This exam is worth 40 points. After each question, you will find the number of points it is worth. You should spend approximately x minutes on a question worth x points. That will leave you with 10 minutes to read the statement of the problem and to look over your work.

1. 10 points ____________
2. 4 points ____________
3. 8 points ____________
4. 11 points ____________
5. 7 points ____________

Consider the pipelined processor of Figure 1 with common instruction fetch and issue stages and 3 pipelined functional units: one for floating-point operations (single and double precision), one for integer operations (including branches), and one for load-store operations (access to data cache). There are separate sets of integer and floating-point registers each 32-bit long. Many floating-point operations operate on "doubles", i.e., pairs of floating-point registers (64-bit operands).

The processor is a RISC machine, i.e., all operations except for Load and Store operate on registers.

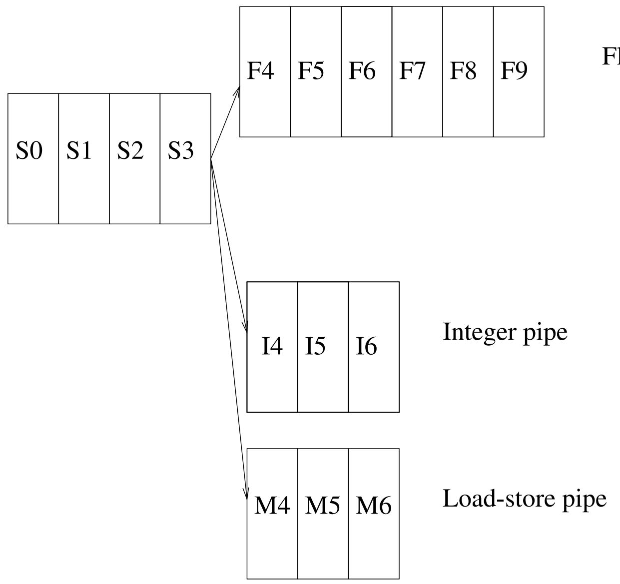

Figure 1. Pipeline 1.

More specifically:

- Stages S0 - S3, common to all instructions, perform:
  - In stage S0: instruction fetch.
  - In stage S1: branch target address computation and branch prediction. Prediction is done via 2-bit saturating counters associated with each instruction in the I-cache. These counters are initialized to "weak not-taken". (There is no Branch Target Buffer – BTB – in this processor.)
  - In stage S2: decode and read registers
  - In stage S3: check for resource conflict and data dependencies.

Once past S3, instructions run to completion unless there a is mispredicted branch (or an interrupt/exception).

Instructions are issued to their respective functional units, depending on their opcode, when their operands are ready.

- Stages F4 - F8 represent a multiple cycle "EXE" stage for the floating-point operations.
- Stage F9 writes the result in a floating-point (double) register.
- Stage I4 is an EXE stage. The result of a branch is known at the end of this stage so that, in the case of a misprediction, the correct instruction can be fetched in the next cycle. All arithmetic instructions, except "long shifts", compute their result in this stage.
- Stage I5 is used for "long shifts" only.
- Stage I6 writes the result in an integer register.
- Stage M4 computes the address of the memory location in a load/store instruction.
- Stage M5 accesses the data cache (the cache is virtually indexed so that data cache access and TLB check can be started in parallel).
- Stage M6 checks (i.e., receives the answer from the TLB) to see if there was a cache hit. In other words, forwarding or bypassing after a load cannot be done before the end of M6. In the case of a load the result is written in the corresponding integer or floating-point (double) register during M6.

1. (10 points)

Indicate 5 paths where forwarding can be performed. Indicate these paths directly on the figure, number them, and for each of them indicate the width of the data path (e.g., 1-bit, 8-bit, 32-bit, 64-bit, 1 Megabyte (!) etc.)

2. (4 points)

Let D2-D7 be floating point registers, R1-R4 be integer registers. Consider the sequence of instructions:

```
l.d     $D2, 100($R1)          # load 8 bytes from effective address
                               # R1+100 in registers D2 and D3
add.d   $D6,$D2,$D4            # {D6,D7} = {D2,D3} + {D4,D5}
                               # add double-precision floats
```

How many "bubbles" (cycle stalls) will be generated in this sequence (you can assume that there is always a cache hit and that no register is waiting for a result before the load instruction).

Answer the same question for the sequence

```
lw      $R2, 100($R1)          # load 4 bytes from effective address
                               # R1+100 in register R2
add     $R4,$R2,$R3            # R4  =  R2 + R3
```

3. (8 points)

Branch Prediction is done in Stage S1 (see above description). Indicate the number of "bubbles" that will be generated in the 4 possible combinations of Predict/Actual:

1. Predict: NT; Actual: NT. Penalty (Number of bubbles):
2. Predict: NT; Actual: T. Penalty (Number of bubbles):
3. Predict: T; Actual: NT. Penalty (Number of bubbles):
4. Predict: T; Actual: T. Penalty (Number of bubbles):

4. (11 points) (This question continues on the next page)

It has been observed that on this machine 20% of instructions are Loads and 30% are branches.

After instrumentation of a suite of benchmarks, it was observed that:

- 40% of the time the result of the load is needed 3 or more instructions after the load;
- 40% of the time the result of the load is needed 2 instructions after the load;
- 20% of the time the result of the load is needed for the instruction following the load.

and that

- One third of the branches are Not Taken and two thirds are Taken
- The Branch Prediction mechanism described earlier has a 90% prediction accuracy when predicting "Not Taken" and a 90% accuracy when predicting "Taken".

Assume an ideal CPI of 1.

What is the extra amount of CPI contributed by Load instructions, say CPI_{load}

What is the extra amount of CPI contributed by branch instructions, say CPI_{br}

If you wanted to have a more complete assessment of CPI contributions due to pipeline effects (i.e., not taking into account CPI contributions due to cache and TLB misses) you would need more information. List two items for which you need information.

5. (7 points)

The designers of the next generation of processors with the same ISA as the Processor depicted in Figure 1 have modified it as follows:

1. A Branch Target Buffer (BTB) has been included in the architecture and is tested during S0.
2. The result of an access to the data cache is available at the end of Stage M5. Stage M6 is still needed though for writing in the register sets.
3. Because of critical paths constraints, the result of a branch or of an integer arithmetic operation is only known now at the end of Stage I5 (instead of I4)

What is the new CPI_{load}

What is the new CPI_{br}

Would the changes have been worthwhile, even if they had not introduced a BTB? Is there some other information that you need in order to answer this question?