CS433 Midterm

Prof Josep Torrellas

October 16, 2014

Time: 1 hour + 15 minutes

Name:

Alias:

Instructions:

1. This is a closed-book, closed-notes examination.
2. The Exam has 4 Questions. Please budget your time.
3. Calculators are allowed.
4. Please write your answers neatly. Good luck!

Problem No.	Maxm Points	Points Scored
1	30	
2	20	
3	30	
3	20	
Total	100	

I Loop Unrolling + Software Pipelining [30 points]

Assume that the following loop is executed a great number of times. It runs on the MIPS architecture. Assume that ADD.D takes one cycle and that there is a branch delay slot of 1 cycle.

```
Sum:   L.D     R4, 0(R0)
       L.D.    R5, 0(R1)
       ADD.D   R4, R4, R5
       S.W.    R4, 0(R2)
       ADD.I   R0, R0, #4
       ADD.I   R1, R1, #4
       ADD.I   R2, R2, #4
       ADD.I   R3, R3, # -1
       BNE     R3, 0, Sum
```

a. [**4 points**] How many cycles does it take to run each iteration as is? Do not move any instructions, and put all the necessary stalls.

b. [**8 points**] Unroll the loop once (to get an order-2 loop, which means that two iterations execute in one new iteration) and reorder the instructions to make the loop run as fast as possible. With the unrolled loop, how many cycles does each of the old iterations take now?

c. [**2 points**] If you unroll the loop N times, will the sequence of code run N times faster? Select only one.

   a. Yes, because it is unrolled N times and there is a linear relation between performance and loop unrolling.
   b. Yes, but only in this particular code sequence.
   c. No, because this loop does not have many stalls to eliminate.
   d. No, because the loop has more instructions now.

d. [**2 points**] Suppose you have infinite registers, what do you gain by unrolling this particular loop beyond one time? Select only one.

   a. I can make the sequence run faster linearly with the number of times I unroll the loop.
   b. I will only reduce part of the overhead of the branches and address calculations.
   c. I gain nothing by unrolling it a great number of times.
   d. I gain in memory because the code occupies less space.

Now, you will software pipeline the loop. We will work one step at a time. Do not worry about the branch and address calculation.

e. [**4 points**] Divide the loop iteration into pipeline stages

f. **[4 points]** Write the steady state software pipelined loop

g. **[3 points]** Write the prolog instructions

h. **[3 points]** Write the epilog instructions

II Single Issue and Hardware Speculation [20 points]

This problem concerns Tomasulo algorithm with single issue and hardware speculation. Consider the following assumptions to solve this problem.

Fuctional Unit Type	Cycles in EX
Integer Mul	2
Integer Div	10
Integer Add	1

Table 1: Execution times

- Assume you have unlimited reservation stations, functional units, CDBs, and reorder buffer entries.
- Functional units are **not** pipelined.
- An instruction waiting for data on the CDB can move to its EX stage in the cycle after the CDB broadcast.
- Assume that integer instructions also follow Tomasulos algorithm (analogous to the floating point instructions) so they can be issued out of order and the result from the integer functional unit is also broadcast on the CDB and forwarded to dependent instructions through the CDB.
- Assume the processor can issue into the reservation stations (and reorder buffer) only one instruction per cycle.
- Assume it can **commit only** one instruction per cycle as well.

Complete the following table using Tomasulos algorithm with the above specifications. Fill in the cycle numbers in each pipeline stage for each instruction (CMT stands for the commit stage). For each instruction, indicate where its source operands are read from (use RF for register file, ROB for reorder buffer, and CDB for common data bus). The entries for the first instruction and for the issue stage are filled in for you.

Important: For each stall, **write** on the paper the reason for the stall. No explanation, no points.

	Instruction	IS	Src operand 1	Src operand 2	EX	WB	CM
1	MUL R2, ,R6, R12	1	RF	RF	2	4	5
2	DIV R1, R1, R2						
3	ADD R5, R1, R3						
4	ADDI R7, R5,4						
5	ADD R5, R6, R8						
6	ADDI R8, R8, 2						
7	ADD R9, R6, R9						
8	ADD R5, R5, R10						
9	ADD R6, R8, R5						

Table 2: Execution using Single Issue and Hardware Speculation

III **Pipelining [30 points]**

A. Modified MIPS Pipeline

We change the MIPS pipeline so that EX now takes 2 cycles, and both the branch outcome and target are computed in the first EX cycle. Normal operations take the 2 EX cycles to complete. Also, MEM takes two cycles, but the store *data* is only needed in the second MEM cycle.

```
IF   instruction fetch
ID   register read and instruction decode
EX1  first stage of execution. Branch condition and target are computed
     in this cycle. Normal instructions need the 2 EX cycles to
     execute.
EX2  second stage of execution; ALU results and effective
     address for loads and stores need the two EX cycles.
MEM1 first cycle of data memory access; only address is needed.
MEM2 second cycle of data memory access; store data needed at the beginning
     of this cycle, load data available at the end of this cycle.
WB   write back of results
```

Assume the register file operates on split cycles as discussed in class, so as to minimize bypassing requirements.

a) **[5 points]** How many branch delay slots are there? Why?

b) **[5 points]** Assuming all possible forwarding is supported, how many stall cycles we have in the following case? (draw a pipeline picture)

ADD R1 R2 R3

ADD R7 R1 R6

c) [**5 points**] Repeat b) for:
LOAD R1, 10(R5)
ADD R7 R1 R6

d) [**5 points**] Repeat b) for:
LOAD R1, 10(R5)
STORE R1, 8(R8)

B. **[10 points]** Now go back to the original MIPS Pipeline
Consider the following two J-type MIPS instructions:
JR Reg
J name
The first one is encoded as 6 bits for opcode and 5 bits for register. The second one is encoded as 6 bits for opcode and 26 bits for offset. Assume that these two instructions complete at the EX stage. Draw the modifications to the ID and EX stage of the MIPS pipeline that are needed to support these instructions. Add as little hardware as possible. Please draw two different figures; one for each instruction.

C. **Exceptions [20 points]**

a. [**4 points**] What does it mean that a pipeline supports precise exceptions?

b. [**4 points**] How does a pipeline support precise exceptions?

c. [**4 points**] List one good thing and one bad thing of precise exceptions?

d. [**4 points**] What is a statically and a dynamically scheduled machine?

e. [**4 points**] What is a superscalar and a VLIW? Are they static or dynamic machines?