CSE 471 Autumn 2002

Computer Design and Organization

Midterm

Wednesday November 6th

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.

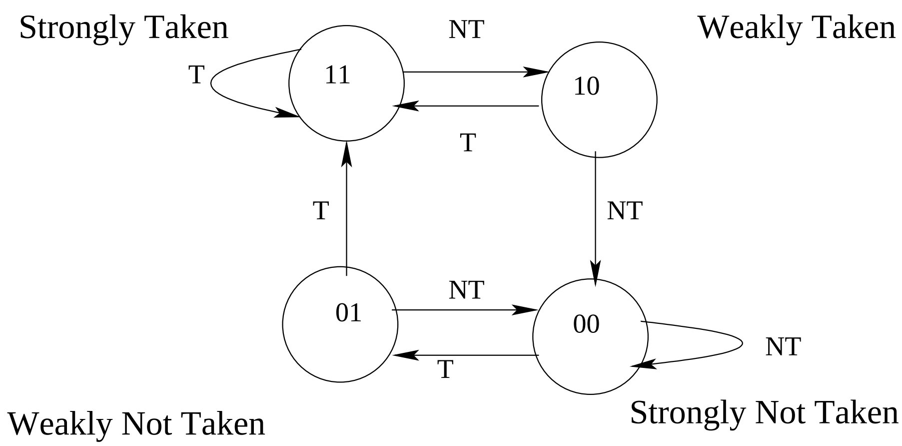
1. 20 points
   - (a) 3 points ______________
   - (b) 3 points ______________
   - (c) 2 points ______________
   - (d) 2 points ______________
   - (e) 5 points ______________
   - (f) 5 points ______________
2. 20 points ______________
   - (a) 10 points ______________
   - (b) 10 points ______________

1. Branch Prediction (20 points)

Processor XYZ performs branch prediction with:

- A 256 entry, 2-way set associative Branch Target Buffer (BTB) that records only the branches that are predicted taken (T). A valid bit is included in each entry to that effect (i.e., if the branch was entered in the BTB and is subsequently predicted to be not taken (NT) the valid bit is turned off). On a miss to the BTB, the default prediction is NT.
- A table of 1K 2-bit saturating counters

(a) (3 points)
Complete the state diagram of a 2-bit saturating counter shown below by showing labelled transitions between states.



Note: This is a gray code counter

(b) (3 points)
Give the layout, i.e., the various fields and their lengths, of a BTB entry. You can assume a 32-bit ISA and that all instructions are of the same length (4 bytes).

Each entry consists of:

- *a tag (23 bits) since we need to have an index of 7 bits to access one of the 128 sets and the last 2 bits of the PC are a "displacement".*
- *a next PC of 30 bits with the 2 rightmost bits assumed 00.*
- *a valid bit*

(c) (2 points)
At stage 1 of execution, corresponding to IF, the BTB is accessed. How is the BTB look-up performed and what are the possible outcomes? In this sub-question and the remainder of Question #1 you can assume that there are no indirect jumps and that on a valid hit to the BTB the target address in the BTB is correct.

The BTB is accessed via an index of 7 bits (bits 2-8) of the PC. The 23 high-order bits of the PC are compared with the two tags of the indexed set. There are 3 possibilities:

- *No match. The prediction is NT. The next instruction (at PC + 4) is fetched.*
- *Match and invalid bit. The prediction is NT. The next instruction (at PC + 4) is fetched.*
- *Match and valid bit. The prediction is T. The next PC of the matching entry becomes PC and the instruction at that PC will be fetched on the next cycle.*

(d) (2 points)
At stage 2, corresponding to ID, the instruction is decoded and the address of the target instruction is computed. If the opcode indicates a conditional branch the BPB is accessed. How is the BPB look-up performed and what are the possible outcomes?

The BPB is accessed by the 10 rightmost bits of the PC (excluding the rightmost 2) i.e. bits 2-11. It returns a 2-bit "value" that is interpreted as predict T or predict NT according to the diagram of the Figure in answer (a).

(e) (5 points)
Assume that the prediction of the BPB has priority over that of the BTB. Indicate what happens when:

1. Predictions in BTB and BPB agree.
2. Predictions in BTB and BPB disagree.

(At this point the target address returned by the BTB, if any, should be checked, but we have assumed it's always correct).

Case 1: Predictions agree. Proceed with the current flow of instructions.

Case 2: Predictions disagree. The instructions fetched after the branch need to be squashed. Since BPB has priority:

- *If BPB predicted T, the PC must be set to the target instruction address computed during this cycle.*
- *If BPB predicted NT, the PC must be set to the PC of the branch + 4*

Note: we have assumed that IF and ID might take more than one stage in the pipeline or that more than one instruction can issue per cycle. If there is only one stage and a single instruction issue replace "The instructions fetched" by "The instruction fetched".

(f) (5 points)
At stage 7 (there is no special significance in the number 7), corresponding to the end of the EX stage, the branch outcome is available. Indicate what happens in the 4 cases of the cross product
(Predict Taken, Predict Not Taken) * (Outcome Taken, Outcome Not Taken). Don't forget the updates to BTB and BPB.

In all cases, the BPB is updated, i.e., the 2-bit counter corresponding to the branch is incremented if the outcome is T and decremented if the outcome is NT.

In all cases the BTB is checked. If the outcome is T and:

- *There is an entry in the BTB corresponding to the branch, set the bit to valid (even if it were already valid; it won't hurt)*
- *There is no matching entry in the BTB. Replace one of the existing ones in the set where the entry should be (e.g., replace one with the valid bit off, or if not replace one of the two "randomly") and set a new entry with current (tag, target address, valid bit) as in answer to question (b).*

If the outcome is NT and:

- *There is an entry in the BTB corresponding to the branch, then set the valid bit to invalid.*
- *There is no matching entry in the BTB. Do nothing.*

Finally, if the predictions and outcomes disagree and:

- *The outcome was T and the prediction was NT, squash all instructions after the branch, and reset PC to the computed target address.*
- *The outcome was NT and the prediction was T, squash all instructions after the branch, and reset PC to branchPC + 4*

2. Tomasulo's algorithm (20 points; 10 each for each "cycle")

In the next two pages, you'll find tables showing the state of the Reservation stations, Reorder buffer, and FP register status for an **out-of-order execution and in-order completion** processor implementing Tomasulo's algorithm at a given stage of execution of a MIPS code sequence. Filling of the tables has been done assuming the following:

- Only one instruction can issue per cycle
- The reorder buffer had depth 8 (results of loads are forwarded via the common data bus -CDB– to the reorder buffer)
- All functional units are pipelined
- There are 2 FP multiply reservation stations
- There are 3 FP add reservation stations
- There are 3 integer reservation stations
- Memory requests occur and complete in one cycle, i.e., barring structural hazards, loads issue in one cycle - say t -, "execute" in the next (t+1), "write" in the next (t+2), and a dependent instruction can start execution one cycle later, i.e., at t+3
- No exceptions occur during the execution of the code
- All integer operations (including loads and stores) require 1 execution cycle
- All FP multiplication operations require 4 execution cycles
- All FP addition operations require 2 execution cycles
- There is a single CDB and, on a CDB write conflict, the instruction issued earlier gets priority
- Execution for a dependent instruction can begin on the cycle after its operand is broadcast on the CDB
- All reservation stations, reorder buffer, and functional units were empty and not busy when the code started execution
- Integer registers are not shown in the tables and you don't have to show their state
- The "Value" column gets updated when the value is broadcast on the CDB

When filling up the tables, do not erase any information already present. Of course, you may overwrite some entries, e.g., from "not busy" to "busy" or any other one if some state/value change makes it necessary.

Given the sequence of MIPS code:

```
MUL.D     F0,F2,F4
ADD.D     F6,F6,F0
ADD.D     F2,F2,F8
L.D       F4, 0(R2)
DADDUI    R1,R1,8
MUL.D     F8,F10,F12
ADD.D     F4,F4,F10
DADDUI    R2,R2,1
```

The tables in the next two pages show the state at the end of the cycle in which the second MUL.D (the one in *italics*) issued.

If more than one instruction is ready to "write" all but the one writing stay in the "execute" state.

Execute*i* means the ith stage of the Execute of a given functional unit.

If all stages of a functional unit are busy and an instruction is in a reservation station of that unit, it is in state "issue".

On the next page, show the state after the next cycle. On the page after that, show the state after two cycles.

Name	Busy	op	V_j	V_k	Q_j	Q_k	Dest
Add1	Y	ADD.D	F6	(#1)			#2
Add2	N	ADD.D	F2	F8			#3
Add3	Y	ADD.D		F10	#4		#7
Mult1	N	MUL.D	F2	F4			#1
Mult2	Y	MUL.D	F10	F12			#6
Int1	Y	L.D	R2				#4
Int2	Y	DADDUI	R1	8			#5
Int3	N						

Table 1: Reservation stations; *Changes in Add2 and Add3*

Entry	Busy	Instruction	State	Dest	Value
1	N	MUL.D F0,F2,F4	Commit	F0	F2 * F4
2	Y	ADD.D F6,F6,F0	Execute1	F6	
3	Y	ADD.D F2,F2,F8	Write	F2	F2 * F8
4	Y	L.D F4, 0(R2)	Memory	F4	
5	Y	DADDUI R1,R1,8	Execute1	R1	
6	Y	MUL.D F8,F10,F12	Execute1	F8	
7	Y	ADD.D F4,F4,F10	Issue	F4	
8	N				

Table 2: Reorder buffer; *Changes in entries 1, 2,3,7*

Field	F0	F2	F4	F6	F8	F10	F12	...
Reorder #		3	7	2	6			
Busy	N	Y	Y	Y	Y			

Table 3: FP register status; *Changes in F0 and F4*

Name	Busy	op	V_j	V_k	Q_j	Q_k	Dest
Add1	Y	ADD.D	F6	(#1)			#2
Add2	N	ADD.D	F2	F8			#3
Add3	Y	ADD.D	(#4)	F10			#7
Mult1	N	MUL.D	F2	F4			#1
Mult2	Y	MUL.D	F10	F12			#6
Int1	N	L.D	R2				#4
Int2	Y	DADDUI	R1	8			#5
Int3	Y	DADDUI	R2	1			#8

Table 4: Reservation stations; *Changes in Add3, Int1 and Int3*

Entry	Busy	Instruction	State	Dest	Value
1	N	MUL.D F0,F2,F4	Gone	F0	F2 * F4
2	Y	ADD.D F6,F6,F0	Execute2	F6	
3	Y	ADD.D F2,F2,F8	Write	F2	F2 * F8
4	Y	L.D F4, 0(R2)	Write	F4	Mem(R2)
5	Y	DADDUI R1,R1,8	Execute1	R1	
6	Y	MUL.D F8,F10,F12	Execute2	F8	
7	Y	ADD.D F4,F4,F10	Issue	F4	
8	Y	DADDUI R2,R2,1	Issue	R2	

Table 5: Reorder buffer; *Changes in entries, 2,3,4,6,8*

Field	F0	F2	F4	F6	F8	F10	F12	...
Reorder #		3	7	2	6			
Busy	N	Y	Y	Y	Y			

Table 6: FP register status: *no change*