JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

Computer Science 605.611

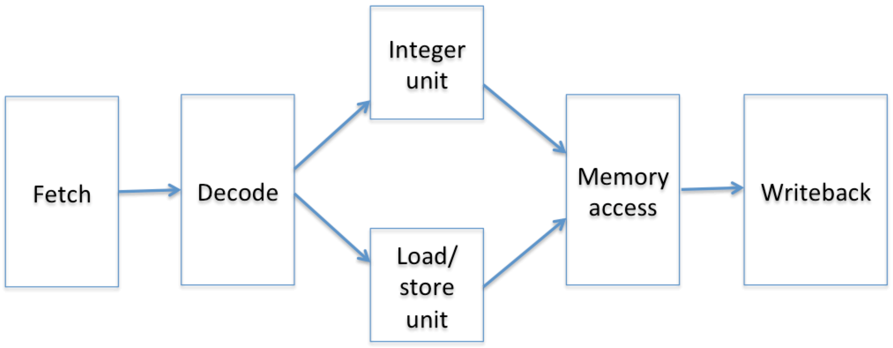
Problem Set 8 Answers

1. (5) Recall that our 5-stage MIPS pipeline is an example of a "scalar" pipeline since at most one instruction can occupy any stage at one time. Which of the following types of hazards does this pipeline never experience and why?

   a) Data hazard
   b) resource hazard
   c) branch hazard
   d) control hazard

   A resource hazard would never occur, since at most one instruction would use the same resource at the same time.

2. The diagram below represents a degree 2 superscalar version of our 5-stage MIPS pipelined system. The decode stage includes a buffer to hold instructions that have been decoded but cannot yet execute. The execute stage includes an integer unit and a load/store unit. The system allows both out-of-order execution and out-of-order completion. Each stage requires 1 clock cycle.



a) (5) How many clock cycles does the following short instruction sequence consume on this superscalar system?

1.	add	\$5,\$4,\$3
2.	lw	\$2,16(\$6)
3.	sw	\$9,8(\$7)
4.	lw	\$10,48(\$12)
5.		

After the first two instructions are fetched and decoded, they can execute in parallel on the two execution units. However the final sw and lw must execute sequentially since they both require the load/store unit. Hence the first two instructions complete in cycle 5, the sw completes in cycle 6 and the lw completes in cycle 7.

b) (3) How many clock cycles does the same instruction sequence consume on the original 5-stage scalar system?
On the original scalar system, the sequence would require 8 cycles. The add completes in cycle 5 with one additional cycle required for each of the remaining 3 instructions.

c) (3) How many read ports are required for the register file to support the superscalar system? Recall that a separate read port is required for each register input operand.
Since this is a degree 2 superscalar system, up to 2 instructions may have to read a pair of input registers at the same time, so 4 register read ports are required.

d) (3) How many write ports are required for the register file to support the superscalar system? Write ports receive the register results produced by instructions.
Two register write ports are required, one for each of the two instructions that can complete in a single cycle.

3. Consider the following instruction sequence that executes on our 5-stage scalar pipelined system with no code rearrangement and no branch prediction, but with a hazard unit as well as a forwarding unit:

```
            ori    $4,$0,2
loop:       add    $6,$7,$8
            and    $9,$4,$7
            bne    $4,$0,loop          ; delayed branch
            srl    $4,$4,1
            addi   $3,$0,1
            sll    $3,$3,1
```

a) (5) The ori instruction is fetched in cycle 1. During which clock cycle does the sll instruction complete the write-back stage if no early branch condition evaluation or branch prediction is used?

The ori instruction initializes $4 to the value 2. Each time that the bne instruction is executed, the srl in the delay slot is executed whether the branch is taken or not. Register $4=2 the first time through the loop, $4=1 the second time, $4=0 the final (third) time through the loop.
The bne takes effect when it is in the memory stage. When the bne branches, the instructions in the decode and fetch stages will be flushed causing two bubbles. The third time through the loop, no flushing occurs since the branch is not taken. The sll completes its write-back stage in cycle 23 since the instructions flow through the pipeline as follows:

Cycle	Fetch	Decode	Exec	Mem	Write-back
1	ori				
2	add	ori			
3	and	add	ori		
4	bne	and	add	ori	
5	srl	bne	and	add	ori
6	addi	srl	bne	and	add
7	sll	addi	srl	bne	and
8	add	bubble	bubble	srl	bne
9	and	add	bubble	bubble	srl
10	bne	and	add	bubble	bubble
11	srl	bne	and	add	bubble
12	addi	srl	bne	and	add
13	sll	addi	srl	bne	and
14	add	bubble	bubble	srl	bne
15	and	add	bubble	bubble	srl
16	bne	and	add	bubble	bubble
17	srl	bne	and	add	bubble
18	addi	srl	bne	and	add
19	sll	addi	srl	bne	and
20		sll	addi	srl	bne
21			sll	addi	srl
22				sll	addi
23					sll

b) (5) The ori instruction is fetched in cycle 1. There is no early branch condition evaluation. During which clock cycle does the sll instruction complete the write-back stage if branch prediction based on a decode history table (DHT) is used? Assume that the DHT always predicts that the branch will be taken.

When the branch is predicted in stage 2, the instruction in the delay slot (srl) will already be in the fetch stage. The next instruction that is fetched into the pipeline behind the srl instruction will be from the branch target address (i.e., the add instruction). The first two predictions will be correct, but the third is incorrect. When the bne actually branches, no flushing occurs. Flushing only occurs when the prediction is wrong. The incorrect branch prediction will be detected and acted on when the bne is in the MEM stage, which causes the two instructions (add, and) along the incorrect path to be flushed. Hence the instructions flow through the pipeline as follows:

Cycle	Fetch	Decode	Exec	Mem	Write-back
1	ori				
2	add	ori			
3	and	add	ori		
4	bne	and	add	ori	
5	srl	bne	and	add	ori
6	add	srl	bne	and	add
7	and	add	srl	bne	and
8	bne	and	add	srl	bne
9	srl	bne	and	add	srl
10	add	srl	bne	and	add
11	and	add	srl	bne	and
12	bne	and	add	srl	bne
13	srl	bne	and	add	srl
14	add	srl	bne	and	add
15	and	add	srl	bne	and
16	addi	bubble	bubble	srl	bne
17	sll	addi	bubble	bubble	srl
18		sll	addi	bubble	bubble
19			sll	addi	bubble
20				sll	addi
21					sll

Therefore the sll completes its write-back stage in cycle 21.

4. (5) Our MIPS system treats all branch instructions as delayed branches. Which one of the following is the main reason for using delayed branches?
a) to reduce the number of data hazards
b) to provide time to compute the branch target address
c) to reduce the number of pipeline bubbles
d) to provide time to predict the branch behavior
The main reason is to reduce the number of pipeline bubbles that result from flushing instructions that should not execute from the pipeline. With a decode history table the branch prediction occurs in stage 2 but the actual branch behavior is not known until stage 4 after the condition has been tested. When the branch instruction is in stage 4 there are 3 instructions behind it in the pipeline. Due to the delayed branch, only the first 2 stages have to be flushed resulting in a maximum branch penalty of 2 cycles.

5. a) (3) Once a conditional branch instruction is brought into the pipeline, what is the earliest stage within the pipeline in which a branch prediction can be made if the prediction is based on a branch history table?
Since only the address in the PC is required, the prediction can be made as early as the fetch stage (stage 1). There is no stage 0.

b) (3) When the prediction for a conditional branch instruction is found in a decode history table, what is the maximum branch penalty (i.e., the maximum number of pipeline bubbles that must be inserted) if the prediction turns out to be incorrect and the instruction is a delayed branch.
With a decode history table the branch prediction occurs in stage 2 but the actual branch behavior is not known until stage 4 after the condition has been tested. When the branch instruction is in stage 4 there are 3 instructions behind it in the pipeline. Due to the delayed branch, only the first 2 stages have to be flushed (in the fetch and in the decode stage) resulting in a maximum branch penalty of 2 cycles.

6. Consider the following code containing nested loops:

```
Loop1:      …

            …

Loop2:      …

            …

            …

            bne    $t2,Loop2

            …

            …

            beq    $t3,Loop1

            …
```

The bne and beq instructions are the only branch instructions within the code sequence. The outer loop (Loop1) contains an inner loop (Loop2). Assume that the beq at the end of the outer loop transfers control 120 consecutive times before the loop is exited. Also assume that for each iteration of the outer loop, the bne instruction at the end of the inner loop transfers control 100 consecutive times before the inner loop exits. That is, the inner loop iterates 100 times for each of the 120 iterations of the outer loop. Branch prediction is used for both branch instructions. What is the total number of mispredictions for each branch instruction, if:

a) the prediction is based on a separate single branch prediction bit for each branch instruction?

(3) Number of mispredictions for the bne = ___240_____

A misprediction occurs the first time the bne is executed and the bit is changed to 1 to indicate taken. So for each of the next 98 executions the prediction is correct. On the 100th execution the branch is not taken, and a second misprediction occurs. Hence there are 2 mispredictions for the bne for each of the 100 iterations of the inner loop (on the first and 100th). But the inner loop iterates 100 times for each iteration of the outer loop. So the total number of mispredictions for the bne = 2*120 = 240.

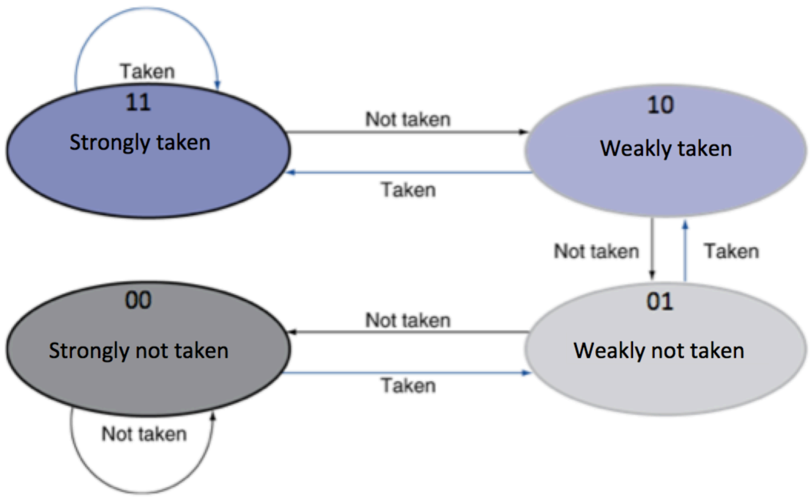
(3) Number of mispredictions for the beq = ____2_____

A misprediction occurs on the first execution of the beq instruction and on the final execution of the beq instruction (i.e., on the 120th execution).

So the total number of mispredictions for the beq = 2.

The initial value for the branch prediction bit for each branch instruction is 0 (not taken) and the bit is inverted each time that the prediction is incorrect (i.e., mispredicted).

b) the prediction is instead based on a separate pair of branch prediction bits for each branch instruction. The initial value of the branch prediction bits for each branch instruction = 00 (strongly not taken) and the bits are updated based on the diagram below:



(5) Number of mispredictions for the bne = ___122__

A misprediction occurs for the first execution of the bne instruction and the bits change from 00 to 01. On the second execution of the bne another misprediction occurs and the bits change from 01 to 10. The prediction is now for the bne to be taken so on the third execution of the bne the prediction is correct and the bits change from 10 to 11 (strongly taken). Hence for the next 96 executions of the bne the prediction will be correct. On the 100th iteration there is a misprediction and the bits go from 11 back to 10 (weakly taken). Hence for the first iteration of the outer loop, the bne at the end of the inner loop is misprediction three times. But for the remaining iterations of the outer loop, the first 99 predictions for the bne at the end of the inner loop will be correct and only the final iteration of the inner loop will have a misprediction. This is because the prediction bits for the bne are left at 10 (taken) each time the inner loop completes. So the total number of mispredictions for the bne = 3 + 119 = 122.

(5) Number of mispredictions for the beq = _____3___ There is a misprediction for the beq instruction on the first and second iteration of the outer loop. On the first misprediction the bits change from 00 to 01. On the second misprediction the bits change from 01 to 10 (weakly taken). On the third iteration the prediction is correct and the bits change from 10 to 11 where they remain for the next 116 iterations. On the 120th iteration there is a misprediction but the bits go from 11 back to 10. So the total number of mispredictions for the beq instruction is 3.

7. a) (3) What type of data hazard (RAW, WAR, WAW, or NONE), requiring a stall, is caused by the following pair of instructions on a scalar pipeline?

```
sw     $5,44($2)
add    $10,$5,$2
```

There is no data hazard since the sw instruction does not write to $5. So the answer is NONE.

b) (3) What type of data hazard (RAW, WAR, WAW, or NONE), requiring a stall, is caused by the following pair of instructions on a scalar pipeline?

```
sw     $5,44($2)
lw     $5,44($2)
```

None. The sw does not modify $5 and the lw does not use $5 as an input. Since instructions execute in-order on the scalar pipeline, the sw completes its write to memory before the lw reads from the location. Also since the instructions execute in the order listed, the lw can't overwrite $5 before the sw reads $5

c) (3) What type of data hazard (RAW, WAR, WAW, or NONE) can be caused by the following two instructions if instead of the scalar system they are executed on a superscalar pipeline with multiple load/store units that allows out-of-order execution?

```
sw     $5,44($2)
lw     $5,44($2)
```

On a superscalar system the lw could be allowed to execute before the sw (for example, if the sw was stalled waiting on a result in $5). If so, the lw would change the value in $5 that the sw would write to memory. This would be a WAR data hazard on register $5.

Also a RAW data hazard exists on the shared memory word since the lw must read from memory after the sw writes the same memory word. Otherwise the lw would read the previous contents of the memory word.

8. (10) On a certain VLIW system, each long instruction word is a packet or bundle of six individual machine instructions. The hardware system contains three integer units and three floating point units. Therefore, up to 3 integer operations and 3 floating point operations can be performed in parallel in the same clock cycle. However, due to the mix of available instructions and possible dependencies, the system may not be able to make use of all the execution units in every cycle. In that case, one or more nop instructions must be inserted into the instruction bundle (i.e., into the long instruction word) to fill any unused slots.
Using the following format to show the instructions within each long instruction word:

Int unit1	Int unit2	Int unit3	Flt unit1	Flt unit2	Flt unit3

List the contents of the all long instruction words or bundles that are required for the following group of instructions using the minimum number of nop instructions:

```
add        $11,$2,$3
add        $4,$5,$11
mtc1       $4,$f12
cvt.s.w    $f12,$f12
sub.s      $f14,$f16,$f18
add.s      $f8,$f14,$f12
sll        $6,$9,5
sub.s      $f8,$f14,$f8
```

Based on the available execution units and the dependencies among the instructions, the following 6 long instruction words will be produced:

Int unit1	Int unit2	Int unit3	Flt unit1	Flt unit2	Flt unit3
add $11,$2,$3	sll $6,$9,5	nop	sub.s $f14,$f16,$f18	nop	nop
add $4,$5,$11	nop	nop	nop	nop	nop
mtc1 $4,$f12	nop	nop	nop	nop	nop
nop	nop	nop	cvt.s.w $f12,$f12	nop	nop
nop	nop	nop	add.s $f8,$f14,$f12	nop	nop
nop	nop	nop	sub.s $f8,$f14,$f8	nop	nop

Floating point instructions must execute on a floating point unit and CPU instructions (integer instructions) must execute on an integer unit. The mtc1 instruction is a CPU instruction.
The add must write $4 before the mtc1 reads $4. The mtc1 most copy $4 into $f12 before the cvt.s.w can convert the contents of $f12. The cvt.s.w must write $f12 before the add.s reads $f12 and the add.s must write $f8 before the sub.s reads $f8.

9. (5) What is the main advantage of using separate reservation stations versus using a centralized instruction window to supply instructions to the execution units within a superscalar system?

When a reservation station becomes full, only later instructions that require the same reservation station are stalled. However, a full instruction window causes later instructions of all types to be stalled. So there is a greater chance that one or more execution units will be idle with a full instruction window. One or more full reservation stations still allow further instructions to be processed as long as those instructions map to execution units whose reservation station still has room. The instruction window can allow a single instruction type to monopolize the window and starve some execution units for work.

10. (5) Which type of data hazard (RAW, WAR or WAW) is best handled by using a reorder buffer (ROB)? Explain why.

The answer is WAW, since instructions are extracted from the ROB in the Writeback stage, so that the result registers are written in the correct program order.

11. (5) For each of the 3 types of data hazard: RAW, WAR and WAW, indicate whether register renaming can or can not be used to avoid stalling the pipeline and explain why.

Register renaming can be used to handle both WAR and WAW hazards, but not RAW hazards.

12. a) (5) What is the minimum width (in bits) required for the CPU-to-instruction memory bus in a MIPS degree 4 superscalar system?

The system must be able to fetch four instructions at a time. So the bus must be at least 4*32 = 128 bits wide.

b) (5) Does hardware or does software select the instructions that execute together in the same clock cycle on a:

i. VLIW system

The compiler selects the instructions that it combines into a VLIW instruction or bundle as it translates the source program. So the answer is software.

ii. superscalar system.

The hardware controller selects, at runtime, the instructions that execute together during the same cycle.