Computer Architecture, Section 379: Homework #5

Yousef Alaa Awad

November 25, 2025

1

Given: *Given the following code:*

```
Loop:
  lw   $t0, 0($s1)
  add  $t0, $t0,   $s2
  sub  $t0, $t0,   $s3  # extra dependency
  and  $t4, $t0,   $s4  # added dependent instruction
  or   $t5, $t4,   $s5  # added dependent instruction
  sw   $t5, 0($s1)
  addi $s1, $s1,   -4
  bne  $s1, $zero, Loop
```

A) Arrange the loop in the two-issue Slot 1 / Slot 2 table format shown in slide 20, with respect to all hazards. Compute the IPC.

cycle		ALU/branch	Load/Store
1	Loop:	nop	lw $t0, 0($s1)
2		addi $s1, $s1, -4	nop
3		add $t0, $t0, $s2	nop
4		sub $t0, $t0, $s3	nop
5		and $t4, $t0, $s4	nop
6		or $t5, $t4, $s5	nop
7		bne $s1, $zero, Loop	sw $t5, 4($s1)

Performance Calculation:

$$\text{IPC} = \frac{8 \text{ instructions}}{7 \text{ cycles}} \approx 1.143 \; IpC$$

B) Unroll the loop three times (as opposed to four in slide 23). Show the three copies. Rescheduled loop unrolled using the Slot 1 / Slot 2 table and compute IPC.

cycle		ALU/branch	Load/Store
1	Loop:	nop	lw $t0, 0($s1)
2		nop	nop
3		add $t0, $t0, $s2	nop
4		sub $t0, $t0, $s3	nop
5		and $t4, $t0, $s4	nop
6		or $t5, $t4, $s5	lw $t0, -4($s1)
7		nop	sw $t5, 0($s1)
8		add $t0, $t0, $s2	nop
9		sub $t0, $t0, $s3	nop
10		and $t4, $t0, $s4	nop
11		or $t5, $t4, $s5	lw $t0, -8($s1)
12		nop	sw $t5, -4($s1)
13		add $t0, $t0, $s2	nop
14		sub $t0, $t0, $s3	nop
15		and $t4, $t0, $s4	nop
16		or $t5, $t4, $s5	nop
17		addi $s1, $s1, -12	sw $t5, -8($s1)
18		bne $s1, $zero, Loop	nop

Performance Calculation:

$$\text{IPC} = \frac{20 \text{ instructions}}{18 \text{ cycles}} \approx 1.111 \; IpC$$

C) Apply register renaming following the method shown in slide 26. Use distinct temporaries for each unrolled copy and highlight index changes. Reschedule and compute the IPC.

cycle		ALU/branch	Load/Store
1	Loop:	addi $s1, $s1, -12	lw $t0, 0($s1)
2		nop	lw $t6, 8($s1)
3		add $t0, $t0, $s2	lw $t9, 4($s1)
4		sub $t0, $t0, $s3	nop
5		and $t4, $t0, $s4	nop
6		or $t5, $t4, $s5	nop
7		add $t6, $t6, $s2	nop
8		sub $t6, $t6, $s3	nop
9		and $t7, $t6, $s4	nop
10		or $t8, $t7, $s5	nop
11		add $t9, $t9, $s2	nop
12		sub $t9, $t9, $s3	nop
13		and $t10, $t9, $s4	sw $t5, 12($s1)
14		or $t11, $t10, $s5	sw $t8, 8($s1)
15		bne $s1, $zero, Loop	sw $t11, 4($s1)

Performance Calculation:

$$\text{IPC} = \frac{\text{20 instructions}}{\text{15 cycles}} \approx 1.333\ IpC$$

D) Compute the speedup achieved from part (b) to part (c).

The performance speedup is as follows:

$$\frac{IpC}{IpC} = \frac{1.333}{1.111} \approx 1.2 \text{ times faster}$$

2

Given: *Given the following code:*

```
addi $s1, $s0,   16
lw   $t0, 0($s1)
addi $s2, $s0,   200
lw   $t1, 0($s2)
mul  $t2, $t0,   $t1 # hazard chain extension
add  $t3, $t2,   $t4
and  $t5, $t3,   $t6
or   $t7, $t1,   $t5
```

A) Identify the hazards and draw the dependency structure in the format used in slide 37.

The dependencies are as follows...

Register	Lines
$s1	Lines 1, 2
$t0	Lines 2, 5
$s2	Lines 3, 4
$t1	Lines 4, 5
$t1	Lines 4, 8
$t2	Lines 5, 6
$t3	Lines 6, 7
$t5	Lines 7, 8

B) Assume both load instructions experience cache misses. Rewrite the code and annotate each line as in slide 38: “cache miss”, “on hold”, or “OK to execute.”

Instruction	Status
`addi  $s1, $s0, 16`	ok
`lw     $t0, 0($s1)`	cache miss
`addi $s2, $s0, 200`	ok
`lw     $t1, 0($s2)`	cache miss
`mul  $t2, $t0, $t1`	on hold $t1
`add  $t3, $t2, $t4`	on hold $t2
`and  $t5, $t3, $t6`	on hold $t3
`or   $t7, $t1, $t5`	on hold $t5

C) Before the misses resolve, show the out-of-order execution that can occur, and list the instructions in the reorder buffer with destination, ready/not-ready, and commit status.

- **Line 1:** Executes successfully.
- **Line 2:** Encountered a cache miss.
- **Line 3:** Executes out-of-order.
- **Line 4:** Encountered a cache miss.
- **Lines 5–8:** Currently on hold/dependent.

#	Instruction	Destination	Ready/Not Ready	Commit Status
1	`addi $s1, $s0, 16`	$s1	Ready	**Committed**
2	`lw   $t0, 0($s1)`	$t0	Not Ready	Not Committed
3	`addi $s2, $s0, 200`	$s2	Ready	Not Committed
4	`lw   $t1, 0($s2)`	$t1	Not Ready	Not Committed
5	`mul  $t2, $t0, $t1`	$t2	Not Ready	Not Committed
6	`add  $t3, $t2, $t4`	$t3	Not Ready	Not Committed
7	`and  $t5, $t3, $t6`	$t5	Not Ready	Not Committed
8	`or   $t7, $t1, $t5`	$t7	Not Ready	Not Committed

Status of instructions in the Reorder Buffer / Pipeline