CMPEN431
Rutvij Dhotey
Homework 4

Q1)

a. Data dependencies are as follows :

$t1 : I1 <-> I2 : **RAW**
$t1 : I1 <-> I3 : **RAW**
$t1 : I1 <-> I3 : **WAW**
St2 : I1 <-> I2 : **WAR**
$t2 : I2 <-> I3 : **RAW**

b. Total no. of such dependencies are **3.**

c. Register renaming can help eliminate WAW and WAR type data dependencies. We can reschedule the code as follows –

I1: or $t1, $t2, $t3 #$t1 <- $t2 | $t3
I2: or **$t6**, $t1, $t4
I3: or **$t5**, $t1, $t2

Q2)

a) loop: lw $2, 40($6)

lw $3, 60($6)
add $4, $3, $2
sll $2, $2, 2
sw $4, 80($7)
sw $2, 40($6)
addi $6, $6, 4
addi $7, $7, 4
addi $1, $1, -1
bnez $1, loop

Rescheduling the Instruction bundling: 2 way

ALU/Branch Instruction	Load/Store
addi $1, $1, -1	lw $2, 40($6)
addi $7, $7, 4	lw $3, 60($6)
sll $2, $2, 2	
add $4, $3, $2	sw $2, 40($6)
addi $6, $6, 4	sw $4, **76**($7)
bnez $1, loop	

Therefore, the total 10 instructions at the 6th cycle.
CPI = 0.6

b) Once unrolled loop with register renaming will look as below :

```
loop:   lw $2, 40($6)
        lw $3, 60($6)
        lw $8, 44($6)
        lw $9, 64($6)

        add $4, $3, $2
        sll $2, $2, 2
        add $10,$8, $9
        sll $8, $8, 2

        sw $4, 80($7)
        sw $2, 40($6)
        sw $10,84($7)
        sw $8, 44($6)

        addi $6, $6, 8
        addi $7, $7, 8

        addi $1, $1, -2

        bnez $1, loop
```

addi $1, $1, -2	lw $2, 40($6)
addi $7, $7, 8	lw $3, 60($6)
sll $2, $2, 2	lw $8, 44($6)
add $4, $3, $2	lw $9, 64($6)
sll $8, $8, 2	sw $2, 40($6)
add $10, $8, $9	sw $4, 72($7)
addi $6, $6, 8	sw $8, 44($6)
bnez $1, loop	sw $10,76($7)

CPI: 8 Instructions / 16 cycles = **0.5** ,the best possible case with 2 way VLIW machine.

Q3)

i. Inter Bundle Load Use Cross Checks:

a. **4 way VLIW Data Path** :

ID Stage Instruction:

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

EX Stage Instruction:

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

The dst register in one Instruction in EX stage requires 2 comparisons per Instruction in ID stage corresponding to 2 src fields.

Since there are 4 Instructions in both EX & ID stage, so total no. of comparisons required is 2x 4x 4 = **32 Load Use Checks.**

b. 6 way VLIW Data Path :

ID stage Instruction:

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

EX stage Instruction:

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

Total Load Use checks = **2 x 6 x 6** = **72 Load Use Checks.**

ii. RAW Intra Bundle dependency Checks

a. **4 way VLIW Datapath** :

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

The dst of oldest instruction here in the packet needs to be compared against the 2 src per instruction packeted in the same bundle.

For a 4 way VLIW, this means 6 + 4 +2 = **12 RAW**

b. **6 way VLIW Data path** :

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

For a 6 way VLIW, this means 10 + 8 + 4 + 2 = **24 RAW**

iii. **Intra Bundle WAW checks** :

a. 4 Way VLIW :

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

The dst of oldest instruction here in the packet needs to be compared against the dst of each instruction packeted in the same bundle.

For a 4 way VLIW, this means 3 + 2 +1 = **6 WAW** checks

b. 6 Way VLIW :

dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2	dst, src1,src2

For a 6 way VLIW, this means 5 + 4 + 3 +2 +1 = **15 WAW** checks

---

Q4)

a) With Threaded Execution :

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15
A1	IF	ID	EX	M	WB										
B1		IF	ID	EX	M	M	M	WB							
A2			IF	ID	EX	-	-	M	WB						
B2				IF	ID	-	-	EX	M	WB					
A3					IF	-	-	ID	EX	EX	M	WB			
B3								IF	ID	-	EX	M	WB		
A4									IF	-	ID	EX	M	WB	
B4											IF	ID	EX	M	WB

The total number of Clock Cycles required to complete the execution :**15**

Issue Slots wasted due to Hazards is **3**.

b) With Independent Thread Execution :

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16
A1	IF	ID	EX	M	WB											
A2		IF	ID	EX	M	WB										
A3			IF	ID	EX	EX	M	WB								
A4				IF	ID	-	EX	M	WB							
B1					IF	-	ID	EX	M	M	M	WB				
B2						-	IF	ID	-	-	-	EX	M	WB		
B3								IF	-	-	-	ID	EX	M	WB	
B4												IF	ID	EX	WB	M

Total no. of clock cycles to complete the instructions **:16**

Total no. of issue slots wasted due to Hazards: **4.**

Q5)

Simulation Output:

		bzip2	equake	hmmer	mcf	milc	sjeng
Baseline	sim_total_insn	2500000	2500000	2500000	2500000	2500000	2500000
	sim_IPC	0.2313	0.2856	0.4151	0.3358	0.3929	0.3575
	ifq_occupancy	0.899	0.5744	0.8248	0.65	0.8701	0.8823
	ruu_occupancy	1.6894	1.0651	1.422	1.1265	1.4686	1.5875
	lsq_occupancy	1.0782	0.4443	0.5253	0.5819	0.8431	1.0032
SSS2	sim_total_insn	2500001	2500000	2500000	2500000	2500000	2500001
	sim_IPC	0.3283	0.3411	0.5444	0.3851	0.4553	0.4291
	ifq_occupancy	1.7639	1.0367	1.5451	1.2257	1.7062	1.7211
	ruu_occupancy	3.5543	1.314	1.9179	1.3353	2.0181	2.2208
	lsq_occupancy	2.054	0.5313	0.6854	0.6735	0.9915	1.452
DSS2	sim_total_insn	3010491	3351628	3563117	3592664	3620784	3594683
	sim_IPC	0.3479	0.4092	0.7206	0.4808	0.627	0.5482
	ifq_occupancy	1.6514	0.8156	1.3985	0.9843	1.5938	1.6447
	ruu_occupancy	4.5992	2.5578	3.5723	2.5342	4.1499	3.6858
	lsq_occupancy	2.5487	0.9103	1.2918	1.1264	1.7857	2.134
DSS4	sim_total_insn	4423770	4178298	5572206	4593814	4712388	5111530
	sim_IPC	0.3981	0.4764	0.8039	0.5713	0.8248	0.6638
	ifq_occupancy	2.9184	1.1869	2.658	1.5115	2.9277	3.1399
	ruu_occupancy	9.4538	3.7639	6.7355	4.0405	8.1024	7.2407
	lsq_occupancy	4.7962	1.3476	2.128	1.7306	2.9387	4.1857

Conclusion: As we can see from above table, IPC increases progressively when Baseline machine is changed to 2 way Static Superscalar to 2 way Dynamic Superscalar to 4 way Dynamic Superscalar.