HW 4

Sunday, October 25, 2015 4:29 PM

1)

a)

Or	$t1	$t2	$t3
Or	$t2	$t1	$t4
Or	$t1	$t1	$t2

RAW	$t1 in dst of line 1 and $t1 in src1 of line 2
WAR	$t2 in src1 of line 1 and $t2 in dst of line 2
RAW	$t2 in dst of line 2 and $t2 in src2 of line 3
WAR	$t1 in src1 of line 2 and $t1 in dst of line 3
WAW	$t1 in dst of line 1 and $t1 in dst of line 3
RAW	$t1 in dst of line 1 and $t1 in src1 of line3

b) 3, the two RAW data dependencies.

c)

Or	$t1	$t2	$t3
Or	$t6	$t1	$t4
Or	$t5	$t1	$t5

2)

a)

Line #	ALU/Branch	Load/Store
1	addi $1, $1, -1	lw $2, 40($6)
2	addi $7, $7, 4	lw $3, 60($6)
3	sll $t2, $2, 2	NOP
4	add $4, $3, $2	sw $2, 36($6)
5	addi $6, $6, 4	sw $4, 76($7)
6	bnez $1, loop	

We executed 10 instructions in 6 cycles therefore the CPI = .6

b)

Line #	ALU/Branch	Load/Store
1	addi $1, $1, -2	lw $2, 40($6)
2	addi $7, $7, 8	lw $3, 60($6)
3	sll $t2, $2, 2	lw $8, 44($6)
4	add $4, $3, $2	lw $9, 64($6)
5	sll $t8, $8, 2	sw $2, 40($6)
6	addi $10, $8, 2	sw $4, 72($7)
7	addi $6, $6, 8	sw $8, 44($6)
8	bnez $1, loop	sw $10, 76($7)

We executed 16 instructions in 8 cycles therefore the CPI = .5

3)

	4 -way	6-way
Load-use	32	72
WAR	12	30

WAW	6	15
Read ports	8	12
Write ports	4	6

4)

With FGMT

Cycle	Instruction
1	X -- A1
2	Y -- B1
3	X -- A2
4	X -- A3
5	NOP
6	Y -- B2
7	X -- A4
8	Y -- B3
9	Y -- B4

- Thread X took 7 cycles to finish
- Thread Y took 9 cycles to finish
- It took 10 cycles in total to finish both threads
- 1 issue slots were wasted due to hazards

Without FGMT

Cycle	Thread X
1	A1
2	A2
3	A3
4	NOP
5	NOP
6	A4

Cycle	Thread Y
1	B1
2	NOP
3	NOP
4	NOP
5	B2
6	B3
7	B4

- It took 7 cycles to finish thread X and 8 cycles to finish thread Y

5)

Baseline					
	sim_total_insn	sim_IPC	ifq_occupancy	ruu_occupancy	lsq_occupancy
BZIP2	2500000	0.2313	0.899	1.6894	1.0782
MCF	2500000	0.3358	0.65	1.1265	0.5819
HMMR	2500000	0.4151	0.8248	1.4219	0.5253
SJENG	2500000	0.3575	0.8823	1.5875	1.0032
MILC	2500000	0.3929	0.8701	1.4686	0.8431
EQAKE	2500000	0.2856	0.5744	1.0651	0.4443
Second Run	***2-way, static superscalar datapath***				
	sim_total_insn	sim_IPC	ifq_occupancy	ruu_occupancy	lsq_occupancy
BZIP2	2500001	0.3283	1.7639	3.5543	2.054
MCF	2500000	0.3851	1.2257	1.3353	0.6735
HMMR	2500000	0.5444	1.5452	1.9179	0.6856
SJENG	2500001	0.4291	1.7211	2.2208	1.452

MILC	2500000	0.4553	1.7062	2.0181	0.9915
EQAKE	2500000	0.3411	1.0367	1.314	0.5313
Third Run	***2-way, dynamic superscalar datapath***				
	sim_total_insn	sim_IPC	ifq_occupancy	ruu_occupancy	lsq_occupancy
BZIP2	3010491	0.3479	1.6514	4.5992	2.5487
MCF	3592664	0.4808	0.9843	2.5342	1.1264
HMMR	3563185	0.7205	1.3986	3.5731	1.2921
SJENG	3594683	0.5482	1.6447	3.6858	2.134
MILC	3620784	0.627	1.5938	4.1499	1.7857
EQAKE	3351628	0.4092	0.8156	2.5578	0.9103
Fourth Run	***4-way, dynamic superscalar datapath***				
	sim_total_insn	sim_IPC	ifq_occupancy	ruu_occupancy	lsq_occupancy
BZIP2	3956133	0.3597	3.1594	5.769	2.9723
MCF	4179167	0.5375	1.6781	3.0427	1.2716
HMMR	5400667	0.7951	2.6718	5.7772	1.7031
SJENG	4605584	0.5932	3.2318	4.5153	2.4648
MILC	4013065	0.7348	3.0445	5.7532	1.9979
EQAKE	4040701	0.4434	1.3882	3.4118	1.1886

- IPC and total # of instruction executed increased progressively as the baseline machine changed from static, to 2-way dynamic, and to 4-way dynamic. The increased seen in static superscalar is due to the fact that the machine can now operate on 2 instructions per clock cycle. The increased seen in dynamic superscalar is due to the fact that the machine can now reorder instructions, which means that it can get rid of more NOPs. As the # of ways for the dynamic superscalar increases the number of instructions the machine is able to execute increases also; therefore, the number of # of instructions executed and IPC increases as the # of ways increases.