ECE 4750 PSET 4

Tim Yao (ty252)

Nov 25, 2015

1 Tree Network Topologies

1.a Baseline I3L Microarchitecture

Cycle:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
mul r1, r2, r3	F	D	I	Y0	Y1	Y2	Y3	W																					
mul r4, r1, r5		F	D	I	I	I	I	Y0	Y1	Y2	Y3	W																	
div r6, r7, r8			F	D	D	D	D	I	Z	Z	Z	Z	W																
div r9, r10, r11				F	F	F	F	D	I	I	I	I	Z	Z	Z	Z	W												
div r12, r13, r14								F	D	D	D	D	I	I	I	I	Z	Z	Z	Z	W								
mul r15, r12, r16									F	F	F	F	D	D	D	D	I	I	I	I	Y0	Y1	Y2	Y3	W				
mul r17, r15, r18													F	F	F	F	D	D	D	D	I	I	I	I	Y0	Y1	Y2	Y3	W

Figure 1: Pipeline Diagram for Baseline I3L Architecture

The total issue to commit cycle count is 27.

1.b Schedule Oldest Ready Instruction First on IO2L Microarchitecture

Cycle:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
mul r1, r2, r3	I	Y0	Y1	Y2	Y3	W	C																	
mul r4, r1, r5					I	Y0	Y1	Y2	Y3	W	C													
div r6, r7, r8		I	Z	Z	Z	Z	W	r	r	r	r	C												
div r9, r10, r11						I	Z	Z	Z	Z	W	r	C											
div r12, r13, r14										I	Z	Z	Z	Z	W	C								
mul r15, r12, r16														I	Y0	Y1	Y2	Y3	W	C				
mul r17, r15, r18																		I	Y0	Y1	Y2	Y3	W	C

Figure 2: Pipeline Diagram for IO2L Architecture

The total issue to commit cycle count is 24.

1.c Optimal Scheduling on IO2L Microarchitecture

Cycle:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
mul r1, r2, r3		I	Y0	Y1	Y2	Y3	W	C										
mul r4, r1, r5						I	Y0	Y1	Y2	Y3	W	C						
div r6, r7, r8					I	Z	Z	Z	Z	W	r	r	C					
div r9, r10, r11									I	Z	Z	Z	Z	W	C			
div r12, r13, r14	I	Z	Z	Z	Z	W	r	r	r	r	r	r	r	r	r	C		
mul r15, r12, r16							I	Y0	Y1	Y2	Y3	W	r	r	r	r	C	
mul r17, r15, r18											I	Y0	Y1	Y2	Y3	W	r	C

Figure 3: Pipeline Diagram for IO2L Architecture with Optimized Scheduling

The total issue to commit cycle count is 18.

1.d Scheduling Comparison

The optimized scheduler is able to achieve higher performance due to better exploitation of ILP. In the instruction sequence, there are two main sequences of data dependencies, so the optimal schedule will be to interleave these two sequences, along with the two independent div instructions, to maximize commits per cycle. Also because we have an un-pipelined divider, we will also want to start the divide instructions as early as possible (which is what the optimal schedule does). We might be able to implement this in hardware by using a mixed priority scheduling algorithm. The scheduler will have to place divide instructions at the highest priority (with the oldest div instruction issuing first). The next highest priority will be the oldest ready instruction that is not a divide. This will implicitly interleave instructions with data dependencies.

2 Register Renaming

2.a Architectural RAW, WAW, and WAR Dependencies

```
1 mul  r1, r2, r3
2 mul  r4, r1, r5
3 addu r6, r7, r8
4 mul  r1, r2, r5
5 addu r6, r6, r9
```

2.b Pipeline Diagram with Register Renaming

Cycle:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
mul r1, r2, r3	F	D	I	Y0	Y1	Y2	Y3	W	C							
mul r4, r1, r5		F	D	i	i	i	I	Y0	Y1	Y2	Y3	W	C			
addu r6, r7, r8			F	D	I	X	W	r	r	r	r	r	r	C		
mul r1, r2, r5				F	D	I	Y0	Y1	Y2	Y3	W	r	r	r	C	
addu r6, r6, r9					F	D	i	I	X	W	r	r	r	r	r	C

Figure 4: Pipeline Diagram with Register Renaming

2.c Register Renaming with Pointers in the IQ/ROB

	Stage				**RT**										
Cycle	**D**	**I**	**W**	**C**	**r1**	**r2**	**r3**	**r4**	**r5**	**r6**	**r7**	**r8**	**r9**	**Free List**	**IQ**
0					p0	p1	p2	p3	p4	p5	p6	p7	p8	p9,pA,pB,pC,pD	
1	1				:	:	:	:	:	:	:	:	:	p9,pA,pB,pC,pD	
2	2	1			p9*	:	:	:	:	:	:	:	:	pA,pB,pC,pD	p9/p1/p2
3	3				:	:	:	pA*	:	:	:	:	:	pB,pC,pD	pA/p9*/p4
4	4	3			:	:	:	:	:	pB*	:	:	:	pC,pD	pB/p6/p7
5	5	4			pC*	:	:	:	:	:	:	:	:	pD	pC/p1/p4
6		2	3		:	:	:	:	:	pD*	:	:	:		pD/pB*/p8
7		5	1		:	:	:	:	:	:	:	:	:		
8				1	:	:	:	:	:	:	:	:	:		
9			5		:	:	:	:	:	:	:	:	:	p0	
10			4		:	:	:	:	:	pD	:	:	:	p0	
11			2		pC	:	:	:	:	:	:	:	:	p0	
12				2	:	:	:	pA	:	:	:	:	:	p0	
13				3	:	:	:	:	:	:	:	:	:	p0,p3	
14				4	:	:	:	:	:	:	:	:	:	p0,p3,p5	
15				5	:	:	:	:	:	:	:	:	:	p0,p3,p5,p9	
16					:	:	:	:	:	:	:	:	:	p0,p3,p5,p9,pB	

Figure 5: Microarchitectural State (RT/FL/IQ) for Reg Renaming with Pointers in the IQ/ROB

	ROB				
Cycle	**0**	**1**	**2**	**3**	**4**
0					
1					
2	p9*/r1/p0				
3	p9*/r1/p0	pA*/r4/p3			
4	p9*/r1/p0	pA*/r4/p3	pB*/r6/p5		
5	p9*/r1/p0	pA*/r4/p3	pB*/r6/p5	pC*/r1/p9*	
6	p9*/r1/p0	pA*/r4/p3	pB*/r6/p5	pC*/r1/p9*	pD*/r6/pB*
7	p9*/r1/p0	pA*/r4/p3	pB/r6/p5	pC*/r1/p9*	pD*/r6/pB
8	p9/r1/p0	pA*/r4/p3	pB/r6/p5	pC*/r1/p9	pD*/r6/pB
9	p9/r1/p0	pA*/r4/p3	pB/r6/p5	pC*/r1/p9	pD*/r6/pB
10	p9/r1/p0	pA*/r4/p3	pB/r6/p5	pC*/r1/p9	pD/r6/pB
11	p9/r1/p0	pA*/r4/p3	pB/r6/p5	pC/r1/p9	pD/r6/pB
12	p9/r1/p0	pA/r4/p3	pB/r6/p5	pC/r1/p9	pD/r6/pB
13	p9/r1/p0	pA/r4/p3	•	pC/r1/p9	pD/r6/pB
14	p9/r1/p0	pA/r4/p3	•	•	pD/r6/pB
15	p9/r1/p0	pA/r4/p3	•	•	•

Figure 6: Microarchitectural State (ROB) for Reg Renaming with Pointers in the IQ/ROB

2.d Register Renaming with Values in the IQ/ROB

	Stage				RT										ROB				
Cycle	**D**	**I**	**W**	**C**	**r1**	**r2**	**r3**	**r4**	**r5**	**r6**	**r7**	**r8**	**r9**	**IQ**	**0**	**1**	**2**	**3**	**4**
0																			
1	1																		
2	2	1			p0*									p0/r2/r3	p0*/r1				
3	3				p0*			p1*						p1/p0*/r5	p0*/r1	p1*/r4			
4	4	3			p0*			p1*		p2*				p2/r7/r8	p0*/r1	p1*/r4	p2*/r6		
5	5	4			p3*			p1*		p2*				p3/r2/r5	p0*/r1	p1*/r4	p2*/r6	p3*/r1	
6		2	3		p3*			p1*		p4*				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
7		5	1		p3*			p1*		p4*					p0*/r1	p1*/r4	p2/r6	p3*/r1	p4*/r6
8				1	p3*			p1*		p4*					p0*/r1	p1*/r4	p2/r6	p3*/r1	p4*/r6
9			5		p3*			p1*		p4*						p1*/r4	p2/r6	p3*/r1	p4*/r6
10			4		p3*			p1*		p4						p1*/r4	p2/r6	p3*/r1	p4/r6
11			2		p3			p1*		p4						p1*/r4	p2/r6	p3*/r1	p4/r6
12				2				·		p4						p1/r4	p2/r6	p3*/r1	p4/r6
13				3						p4							p2/r6	p3*/r1	p4/r6
14				4	·					p4							•	p3*/r1	p4/r6
15				5						p4								•	p4/r6
16										·									•

Figure 7: Microarchitectural State for Reg Renaming with Values in the IQ/ROB

3 In-Order vs. Out-of-Order Superscalar Processors

3.a Performance of In-Order Dual-Issue Processor

Cycle:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
lw r1 , 0(r2)	F	D	I	L0	L1	W	C							
mul r3, r1, r4	F	D	I	I	I	Y0	Y1	Y2	Y3	W	C			
sw r3, 0(r5)		F	D	D	D	D	D	D	I	S	W	C		
addiu r2, r2, 4		F	D	D	D	D	D	D	I	A	W	C		
addiu r5, r5, 4			F	F	F	F	F	F	D	I	A	W	C	
addiu r6, r6, -1			F	F	F	F	F	F	D	I	B	W	C	
bne r6, r0, loop									F	D	I	A	W	C
opA									F	*	*	*	*	*

Figure 8: Pipeline Diagram for In-Order Dual-Issue Processor

As shown by the bold vertical lines, during steady state, each loop takes 9 cycles to execute. The W stage of the lw instruction is included because during looping, the W stages causes an extra cycle of "delay" between the last commit of the previous iteration and the first commit of the current iteration.

Therefore, the total number of cycles it takes to execute 64 iterations is 9*64 = 576 cycles.

3.b Performance of Out-of-Order Dual-Issue Processor

Cycle:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
lw r1 , 0(r2)	F	D	I	L0	L1	W	C																
mul r3, r1, r4	F	D	i	i	I	Y0	Y1	Y2	Y3	W	C												
sw r3, 0(r5)		F	D	i	i	i	i	i	I	S	W	C											
addiu r2, r2, 4		F	D	I	A	W	r	r	r	r	r	C											
addiu r5, r5, 4			F	D	I	A	W	r	r	r	r	r	C										
addiu r6, r6, -1			F	D	i	I	B	W	r	r	r	r	C										
bne r6, r0, loop				F	D	i	I	A	W	r	r	r	r	C									
opA				F	*	*	*	*	*	*	*	*	*	*									
lw r1 , 0(r2)					F	D	I	L0	L1	W	r	r	r	C									
mul r3, r1, r4					F	D	i	i	I	Y0	Y1	Y2	Y3	W	C								
sw r3, 0(r5)						F	D	i	i	i	i	i	I	S	W	C							
addiu r2, r2, 4						F	D	i	i	I	A	W	r	r	r	C							
addiu r5, r5, 4							F	D	i	I	B	W	r	r	r	r	C						
addiu r6, r6, -1							F	D	i	i	I	A	W	r	r	r	C						
bne r6, r0, loop								F	D	i	i	I	A	W	r	r	r	C					
opA								F	*	*	*	*	*	*	*	*	*						
lw r1 , 0(r2)									F	D	i	I	L0	L1	W	r	r	C					
mul r3, r1, r4									F	D	i	i	i	I	Y0	Y1	Y2	Y3	W	C			
sw r3, 0(r5)										F	D	i	i	i	i	i	i	I	S	W	C		
addiu r2, r2, 4										F	D	i	i	I	A	W	r	r	r	r	C		
addiu r5, r5, 4											F	D	i	i	I	A	W	r	r	r	r	C	
addiu r6, r6, -1											F	D	i	i	I	B	W	r	r	r	r	C	
bne r6, r0, loop												F	D	i	i	I	A	W	r	r	r	r	C
opA												F	*	*	*	*	*	*	*	*	*	*	*

Figure 9: Pipeline Diagram for Out-of-Order Dual-Issue Processor

As shown by the bold vertical lines, during steady state, it takes 9 cycles to execute 2 iterations of the loop. The first commit (by lw) of the second iteration is not included because the commit is part of previous iteration's last cycle (this pattern is assumed to repeat; for example, the commit stage of the fourth iteration will be in the same cycle as the commit stage of the bne instruction at the end of the third iteration).

Therefore, the total number of cycles it takes to execute 64 iterations is 9*32 = 288 cycles.

3.c Dual-Issue In-Order versus Dual-Issue Out-of-Order

4 Branch Prediction

4.a Two-Bit Saturating Counter Branch History Table

		Branch B0			Branch B1			Branch B2		
i	src[i]	**BHT**	**P**	**A**	**BHT**	**P**	**A**	**BHT**	**P**	**A**
0	0	WT	T	T	WT	T	T	WT	T	T
1	0	ST	T	T	ST	T	T	ST	T	T
2	12	ST	T	NT	ST	T	NT	ST	T	T
3	15	WT	T	NT	WT	T	NT	ST	T	T
4	0	WNT	NT	T	WNT	NT	T	ST	T	T
5	0	WT	T	T	WT	T	T	ST	T	T
6	11	ST	T	NT	ST	T	NT	ST	T	T
7	17	WT	T	NT	WT	T	NT	ST	T	T
8	0	WNT	NT	T	WNT	NT	T	ST	T	T
9	0	WT	T	T	WT	T	T	ST	T	T
10	11	ST	T	NT	ST	T	NT	ST	T	T
11	13	WT	T	NT	WT	T	NT	ST	T	T
12	9	WNT	NT	T	WNT	NT	NT	ST	T	T
13	0	WT	T	T	SNT	NT	T	ST	T	T
14	12	ST	T	NT	WNT	NT	NT	ST	T	T
15	15	WT	T	NT	SNT	NT	NT	ST	T	T
16	0	WNT	NT	T	SNT	NT	T	ST	T	T
17	8	WT	T	T	WNT	NT	NT	ST	T	T
18	12	ST	T	NT	SNT	NT	NT	ST	T	T
19	18	WT	T	NT	SNT	NT	NT	ST	T	NT

Figure 10: Two-Bit Saturating Counter BHT Execution

4.b Two-Level Adaptive Branch Predictor to Exploit Temporal Correlation

		Branch B0				Branch B1				Branch B2			
i	src[i]	**BHSRT**	**BHT**	**P**	**A**	**BHSRT**	**BHT**	**P**	**A**	**BHSRT**	**BHT**	**P**	**A**
0	0	000	WT	T	T	000	WT	T	T	000	WT	T	T
1	0	001	WT	T	T	001	WT	T	T	001	WT	T	T
2	12	011	WT	T	NT	011	WT	T	NT	011	WT	T	T
3	15	110	WT	T	NT	110	WT	T	NT	111	WT	T	T
4	0	100	WT	T	T	100	WT	T	T	111	ST	T	T
5	0	101	WT	T	T	101	WT	T	T	111	ST	T	T
6	11	011	WNT	NT	NT	011	WNT	NT	NT	111	ST	T	T
7	17	110	WNT	NT	NT	110	WNT	NT	NT	111	ST	T	T
8	0	100	ST	T	T	100	ST	T	T	111	ST	T	T
9	0	101	ST	T	T	101	ST	T	T	111	ST	T	T
10	11	011	SNT	NT	NT	011	SNT	NT	NT	111	ST	T	T
11	13	110	SNT	NT	NT	110	SNT	NT	NT	111	ST	T	T
12	9	100	ST	T	T	100	ST	T	NT	111	ST	T	T
13	0	101	ST	T	T	000	ST	T	T	111	ST	T	T
14	12	011	SNT	NT	NT	001	ST	T	NT	111	ST	T	T
15	15	110	SNT	NT	NT	010	WT	T	NT	111	ST	T	T
16	0	100	ST	T	T	100	WT	T	T	111	ST	T	T
17	8	101	ST	T	T	001	WT	T	NT	111	ST	T	T
18	12	011	SNT	NT	NT	010	WNT	NT	NT	111	ST	T	T
19	18	110	SNT	NT	NT	100	ST	T	NT	111	ST	T	NT

Figure 11: Two-Level BHT for Temporal Correlation Execution

4.c Two-Level Adaptive Branch Predictor to Exploit Spatial Correlation

		Branch B0				**Branch B1**				**Branch B2**			
i	src[i]	**BHSR**	**BHT**	**P**	**A**	**BHSR**	**BHT**	**P**	**A**	**BHSR**	**BHT**	**P**	**A**
0	0	0	WT	T	T	1	WT	T	T	1	WT	T	T
1	0	1	WT	T	T	1	ST	T	T	1	ST	T	T
2	12	1	ST	T	NT	0	WT	T	NT	0	WT	T	T
3	15	1	WT	T	NT	0	WNT	NT	NT	0	ST	T	T
4	0	1	WNT	NT	T	1	ST	T	T	1	ST	T	T
5	0	1	WT	T	T	1	ST	T	T	1	ST	T	T
6	11	1	ST	T	NT	0	SNT	NT	NT	0	ST	T	T
7	17	1	WT	T	NT	0	SNT	NT	NT	0	ST	T	T
8	0	1	WNT	NT	T	1	ST	T	T	1	ST	T	T
9	0	1	WT	T	T	1	ST	T	T	1	ST	T	T
10	11	1	ST	T	NT	0	SNT	NT	NT	0	ST	T	T
11	13	1	WT	T	NT	0	SNT	NT	NT	0	ST	T	T
12	9	1	WNT	NT	T	1	ST	T	NT	0	ST	T	T
13	0	1	WT	T	T	1	WT	T	T	1	ST	T	T
14	12	1	ST	T	NT	0	SNT	NT	NT	0	ST	T	T
15	15	1	WT	T	NT	0	SNT	NT	NT	0	ST	T	T
16	0	1	WNT	NT	T	1	ST	T	T	1	ST	T	T
17	8	1	WT	T	T	1	ST	T	NT	0	ST	T	T
18	12	1	ST	T	NT	0	SNT	NT	NT	0	ST	T	T
19	18	1	WT	T	NT	0	SNT	NT	NT	0	ST	T	NT

Figure 12: Two-Level BHT for Spatial Correlation Execution

4.d Branch Predictor Comparison

	Two-Bit FSM Accuracy	**Two-Level Temporal Accuracy**	**Two-Level Spatial Accuracy**
Branch B0	30%	90%	30%
Branch B1	50%	65%	85%
Branch B2	95%	95%	95%
All Branches	58%	83%	70%

Figure 13: Summary of Branch Predictor Accuracies

5 Connections to Classic Architectures

5.a IBM System/360 Model 91 with the Tomasulo Algorithm

5.b Register Renaming in the MIPS R10K and the Intel P6 Microarchitectures