ECE 4750 PSET 4

Tim Yao (ty252)

Nov 25, 2015

Worked with Gautam Ramaswamy, Gaurab Bhattacharya, and Sacheth Hegde.

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
div r12, r13, r14	I	Z	Z	Z	Z	W	C									
mul r1, r2, r3		I	Y0	Y1	Y2	Y3	W	C								
div r6, r7, r8					I	Z	Z	Z	Z	W	C					
mul r15, r12, r16						I	Y0	Y1	Y2	Y3	W	C				
mul r4, r1, r5							I	Y0	Y1	Y2	Y3	W	C			
div r9, r10, r11									I	Z	Z	Z	Z	W	C	
mul r17, r15, r18										I	Y0	Y1	Y2	Y3	W	C

Figure 3: Pipeline Diagram for IO2L Architecture

The total issue to commit cycle count is 16.

1.d Scheduling Comparison

TODO!

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													
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				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
8				1	p3			p1		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
9			5		p3			p1		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
10			4		p3			p1		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
11			2		p3			p1		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
12				2	p3			·		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
13				3	p3			p1		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
14				4	·			p1		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
15				5				p1		p4				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6
16								p1		·				p4/p2*/r9	p0*/r1	p1*/r4	p2*/r6	p3*/r1	p4*/r6

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

3 In-Order Superscalar Processors

3.a Pipeline Diagram for Single-Issue PARCv1 Processor

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
lw r1 , 0(r2)	F	D	X	M	W													
lw r3 , 0(r4)		F	D	X	M	W												
mul r1, r1, r6			F	D	X	M	W											
mul r3, r3, r7				F	D	X	M	W										
addu r8, r1, r3					F	D	X	M	W									
addu r9, r9, r8						F	D	X	M	W								
addiu r2, r2, 4							F	D	X	M	W							
addiu r4, r4, 4								F	D	X	M	W						
addiu r10, r10, -1									F	D	X	M	W					
bne r10, r0, loop										F	D	X	M	W				
opA											F	D	-	-	-			
opB												F	-	-	-	-		
lw r1 , 0(r2)													F	D	X	M	W	
lw r3 , 0(r4)														F	D	X	M	W

Figure 8: Pipeline Diagram for Single-Issue PARCv1 Processor

As shown by the bold vertical lines, each loop takes 12 cycles to execute. The CPI is therefore 12/10 = 1.2. The IPC is 1/CPI = 0.833.
CPI = 1.2 cycles/instruction
IPC = 0.83 instructions/cycle

3.b Pipeline Diagram for Dual-Issue PARCv1 Processor

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
lw r1 , 0(r2)	F	D	B0	B1	W												
lw r3 , 0(r4)	F	D	D	B0	B1	W											
mul r1, r1, r6		F	F	D	A0	A1	W										
mul r3, r3, r7		F	F	D	D	A0	A1	W									
addu r8, r1, r3				F	F	D	A0	A1	W								
addu r9, r9, r8				F	F	D	D	B0	B1	W							
addiu r2, r2, 4						F	F	D	A0	A1	W						
addiu r4, r4, 4						F	F	D	B0	B1	W						
addiu r10, r10, -1								F	D	B0	B1	W					
bne r10, r0, loop								F	D	D	A0	A1	W				
opA									F	F	D	-	-	-			
opB									F	F	D	-	-	-			
opC											F	-	-	-	-		
opD											F	-	-	-	-		
lw r1 , 0(r2)												F	D	B0	B1	W	
lw r3 , 0(r4)												F	D	D	B0	B1	W

Figure 9: Pipeline Diagram for Dual-Issue PARCv1 Processor

As shown by the bold vertical lines, each loop takes 11 cycles to execute. The CPI is therefore 11/10 = 1.1. The IPC is 1/CPI = 0.910.
CPI = 1.1 cycles/instruction
IPC = 0.91 instructions/cycle

3.c Optimized Pipeline Diagram for Dual-Issue PARCv1 Processor

```
1  lw r1 , 0(r2)
2  addiu r2, r2, 4
3  lw r3 , 0(r4)
4  addiu r4, r4, 4
5  mul r1, r1, r6
6  addiu r10, r10, -1
7  mul r3, r3, r7
8  addu r8, r1, r3
9  addu r9, r9, r8
10 bne r10, r0, loop
```

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13
lw r1 , 0(r2)	F	D	B0	B1	W								
addiu r2, r2, 4	F	D	A0	A1	W								
lw r3 , 0(r4)		F	D	B0	B1	W							
addiu r4, r4, 4		F	D	A0	A1	W							
mul r1, r1, r6			F	D	A0	A1	W						
addiu r10, r10, -1			F	D	B0	B1	W						
mul r3, r3, r7				F	D	A0	A1	W					
addu r8, r1, r3				F	D	D	B0	B1	W				
addu r9, r9, r8					F	F	D	B0	B1	W			
bne r10, r0, loop					F	F	D	A0	A1	W			
opA							F	D	-	-	-		
opB							F	D	-	-	-		
opC								F	-	-	-	-	
opD								F	-	-	-	-	
lw r1 , 0(r2)									F	D	B0	B1	W
addiu r2, r2, 4									F	D	A0	A1	W

Figure 10: Optimized Pipeline Diagram for Dual-Issue PARCv1 Processor

As shown by the bold vertical lines, each loop takes 8 cycles to execute. The CPI is therefore $8/10 = 0.8$. The IPC is $1/\text{CPI} = 1.25$.
CPI = 0.8 cycles/instruction
IPC = 1.25 instructions/cycle

3.d Optimized Pipeline Diagram for Quad-Issue PARCv1 Processor

```
1  lw r1 , 0(r2)
2  addiu r2, r2, 4
3  lw r3 , 0(r4)
4  addiu r4, r4, 4
5  mul r1, r1, r6
6  addiu r10, r10, -1
7  mul r3, r3, r7
8  addu r8, r1, r3
9  addu r9, r9, r8
10 bne r10, r0, loop
```

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12
lw r1 , 0(r2)	F	D	B0	B1	W							
addiu r2, r2, 4	F	D	A0	A1	W							
lw r3 , 0(r4)	F	D	H0	H1	W							
addiu r4, r4, 4	F	D	G0	G1	W							
mul r1, r1, r6		F	F	D	A0	A1	W					
addiu r10, r10, -1		F	F	D	B0	B1	W					
mul r3, r3, r7		F	F	D	G0	G1	W					
addu r8, r1, r3		F	F	D	D	H0	H1	W				
addu r9, r9, r8				F	F	D	B0	B1	W			
bne r10, r0, loop				F	F	D	A0	A1	W			
opA				F	F	D	G0	-	-			
opB				F	F	D	H0	-	-			
opC						F	D	-	-	-		
opD						F	D	-	-	-		
opE						F	D	-	-	-		
opF						F	D	-	-	-		
opG							F	-	-	-	-	
opH							F	-	-	-	-	
opI							F	-	-	-	-	
opJ							F	-	-	-	-	
lw r1 , 0(r2)								F	D	B0	B1	W
addiu r2, r2, 4								F	D	A0	A1	W
lw r3 , 0(r4)								F	D	H0	H1	W
addiu r4, r4, 4								F	D	G0	G1	W

Figure 11: Optimized Pipeline Diagram for Quad-Issue PARCv1 Processor

As shown by the bold vertical lines, each loop takes 7 cycles to execute. The CPI is therefore $7/10 = 0.7$. The IPC is $1/\text{CPI} = 1.43$.
CPI = 0.7 cycles/instruction
IPC = 1.43 instructions/cycle

3.e Instruction Level Parallelism

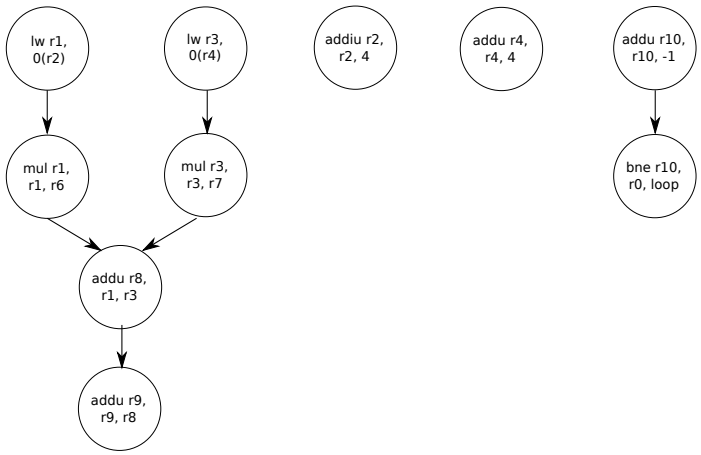

Figure 12: Instruction Dependency Graph for Single Iteration

The longest path contains 4 nodes. The ideal ILP for a single iteration is $10/4 = 2.5$.

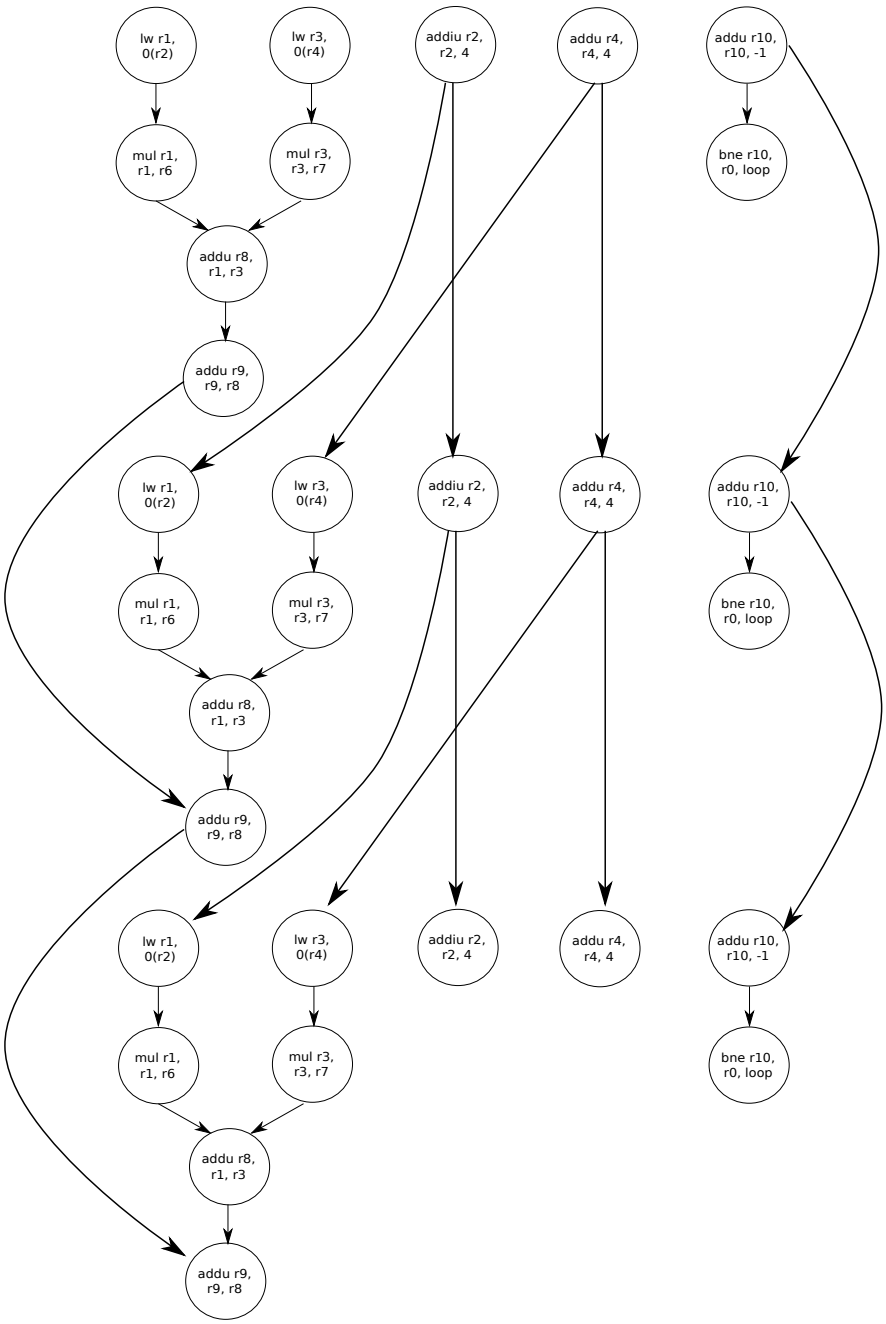

Figure 13: Instruction Dependency Graph for Three Iterations

The longest path contains 6 nodes. The ideal ILP for three iterations is $30/6 = 5$.

The ideal ILP for N iterations of the loop is simply $10N/(3+N)$.

The IPC of the quad-issue processor is less than the ideal ILP due to several different reasons. The first is that the quad issue processor can only execute at most 4 instructions simultaneously. This thereby limits the IPC to a max of 4. Then, the load word instructions are resolved in the second functional unit in the pipeline, which means that a RAW hazard on the next set of instructions will need to be stalled by 1 cycle. There is also a RAW hazard within a fetch block, so this requires 1 cycle of stalling. Finally, the branch instruction introduces another 2 cycles of delay. There are also 2 squashed instructions in the fetch block with the branch instruction, so this reduced the number of executed instructions for calculating the IPC. In the ideal case, we can execute 12 instructions in 3 cycles, but after adding in the squashed instructions and various delays, we are actually executing 10 instructions in 7 cycles.