Pipelined Vs. Superscalar processors

Example 3

E. Sanchez

Computer Architectures

Example 3

Considering the following pseudo-code:

```
for (i = 0; i < 100; i++) {
        Y[i] = X[i]^2 + X[i] / Z[i]
}
```

Suppose that vectors X[i] and Z[i] contain 100 FP numbers, were previously saved in memory, and Z[i] ≠ 0.

Assume the MIPS64 architecture presented below:

- **Integer ALU: 1 clock cycle**
- **Data memory: 1 clock cycle**
- **FP multiplier unit: pipelined 4 stages**
- **FP arithmetic unit: pipelined 2 stages**
- **FP divider unit: not pipelined unit, requiring 4 clock cycles**
- **branch delay slot: 1 clock cycle (disabled)**
- **forwarding is enabled**

Example 3 – [cont]

1. **Write an assembly program for the MIPS64 architecture able to perform the previously presented pseudo-code**
2. **show the timing of the developed loop-based program and compute how many cycles does this program take to execute**
3. **using all the static optimization techniques, re-write the developed code in order to eliminate the most data hazards**
4. **show the timing development of the new optimized program and compute how many clock cycles does this program take to execute.**

Example 3 – [cont]

5. **Complete the table reported below for the first two iterations of the loop, assuming the following superscalar MIPS processor architecture implementing multiple-issue strategy with speculation:**
   - **Issue 2 instructions per clock cycle**
   - **Jump instructions require 1 issue**
   - **2 instructions commit per clock cycle**
   - **Jump prediction is always correct**
   - **There are no cache misses**
   - **There are 2 CDB (Common Data Bus)**
   - **There are different funtional for FP and integer instructions.**

Example 3 – [cont]

Timing facts for the following separate functional units:

- **Memory address 1 clock cycle**
- **Integer ALU 1 clock cycle**
- **Jump unit 1 clock cycle**
- **FP multiplier unit, which is not pipelined: requiring 4 clock cycles**
- **FP divider unit, which is not pipelined: requiring 4 clock cycles**
- **FP Arithmetic unit, which is not pipelined: requiring 2 clock cycles.**

1. Write an assembly program for the MIPS64 architecture able to perform the previously presented pseudo-code

```
;********** MIPS64          **********
;--------------
; Y = x^2 + x /z
; Z ≠ 0.
;--------------
                    .data
vetX:    .double 2, 4, 6, 8…
vetZ:    .double 1, 2, 3, 4…
vetY:    .double 0, 0, 0, 0…
```

1)

	.text		CC
MAIN:	daddui	R1,R0,100	
	daddui	R2,R0,vetX	
	daddui	R3,R0,vetZ	
	daddui	R4,R0,vetY	
loop:	l.d	F1,0(R2)	
	l.d	F2,0(R3)	
	mul.d	F3,F1,F1	
	div.d	F4,F1,F2	
	add.d	F5,F3,F4	
	s.d	F5,0(R4)	
	daddi	R2,R2,8	
	daddi	R3,R3,8	
	daddi	R4,R4,8	
	daddi	R1,R1,-1	
	bnez	R1,loop	
	halt		

1)

	.text		CC
MAIN:	daddui	R1,R0,100	5
	daddui	R2,R0,vetX	1
	daddui	R3,R0,vetZ	1
	daddui	R4,R0,vetY	1
loop:	l.d	F1,0(R2)	1
	l.d	F2,0(R3)	1
	mul.d	F3,F1,F1	4
	div.d	F4,F1,F2	1
	add.d	F5,F3,F4	2
	s.d	F5,0(R4)	1
	daddi	R2,R2,8	1
	daddi	R3,R3,8	1
	daddi	R4,R4,8	1
	daddi	R1,R1,-1	1
	bnez	R1,loop	2
	halt		1

8

17

2)

8

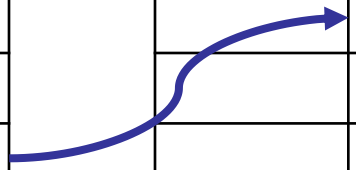
+

17 x 100

1708

3)

	.text	
MAIN:	daddui	R1,R0,100
	daddui	R2,R0,vetX
	daddui	R3,R0,vetZ
	daddui	R4,R0,vetY
loop:	l.d	F1,0(R2)
	l.d	F2,0(R3)
	mul.d	F3,F1,F1
	div.d	F4,F1,F2
	add.d	F5,F3,F4
	s.d	F5,0(R4)
	daddi	R2,R2,8
	daddi	R3,R3,8
	daddi	R4,R4,8



			cc
MAIN:	daddui	R1,R0,25	
	daddui	R2,R0,vetX	
	daddui	R3,R0,vetZ	
	daddui	R4,R0,vetY	
loop:	l.d	F1,0(R2)	
	l.d	F2,0(R3)	
	mul.d	F3,F1,F1	
	div.d	F4,F1,F2	
	daddi	R2,R2,8	
	add.d	F5,F3,F4	
	s.d	F5,0(R4)	
	daddi	R3,R3,8	
	daddi	R4,R4,8	

3)

			CC																				
MAIN:	daddui	R1,R0,25	5	F	D	E	M	W															
	daddui	R2,R0,vetX	1		F	D	E	M	W														
	daddui	R3,R0,vetZ	1			F	D	E	M	W													
	daddui	R4,R0,vetY	1				F	D	E	M	W												
loop:	l.d	F1,0(R2)																					
	l.d	F2,0(R3)																					
	mul.d	F3,F1,F1																					
	div.d	F4,F1,F2																					
	daddi	R2,R2,8																					
	add.d	F5,F3,F4																					
	s.d	F5,0(R4)																					
	daddi	R3,R3,8																					
	daddi	R4,R4,8																					

3)

			CC																				
MAIN:	daddui	R1,R0,25	5	F	D	E	M	W															
	daddui	R2,R0,vetX	1		F	D	E	M	W														
	daddui	R3,R0,vetZ	1			F	D	E	M	W													
	daddui	R4,R0,vetY	1				F	D	E	M	W												
loop:	l.d	F1,0(R2)	1					F	D	E	M	W											
	l.d	F2,0(R3)	1						F	D	E	M	W										
	mul.d	F3,F1,F1	4							F	D	E	E	E	E	M	W						
	div.d	F4,F1,F2	1								F	D	E	E	E	E	M	W					
	daddi	R2,R2,8	0									F	D	E	M	W							
	add.d	F5,F3,F4	2										F	D	S	S	E	E	M	W			
	s.d	F5,0(R4)	1											F	S	S	D	E	S	M	W		
	daddi	R3,R3,8	1														F	D	S	E	M	W	
	daddi	R4,R4,8	1															F	S	D	E	M	W
			12																				

3+)

			CC	
MAIN:	daddui	R1,R0,25	5	8
	daddui	R2,R0,vetX	1	
	daddui	R3,R0,vetZ	1	
	daddui	R4,R0,vetY	1	
loop:	l.d	F1,0(R2)	1	12
	l.d	F2,0(R3)	1	
	mul.d	F3,F1,F1	4	
	div.d	F4,F1,F2	1	
	daddi	R2,R2,8	0	
	add.d	F5,F3,F4	2	
	s.d	F5,0(R4)	1	
	daddi	R3,R3,8	1	
	daddi	R4,R4,8	1	
	l.d	F1,0(R2)	1	12
	l.d	F2,0(R3)	1	
	mul.d	F3,F1,F1	4	
	div.d	F4,F1,F2	1	
	daddi	R2,R2,8	0	
	add.d	F5,F3,F4	2	
	s.d	F5,0(R4)	1	
	daddi	R3,R3,8	1	
	daddi	R4,R4,8	1	

	l.d	F1,0(R2)	1	12
	l.d	F2,0(R3)	1	
	mul.d	F3,F1,F1	4	
	div.d	F4,F1,F2	1	
	daddi	R2,R2,8	0	
	add.d	F5,F3,F4	2	
	s.d	F5,0(R4)	1	
	daddi	R3,R3,8	1	
	daddi	R4,R4,8	1	
	l.d	F1,0(R2)	1	15
	l.d	F2,0(R3)	1	
	mul.d	F3,F1,F1	4	
	div.d	F4,F1,F2	1	
	daddi	R2,R2,8	0	
	add.d	F5,F3,F4	2	
	s.d	F5,0(R4)	1	
	daddi	R3,R3,8	1	
	daddi	R1,R1,-1	1	
	daddi	R4,R4,8	1	
	bnez	R1,loop	1	
	halt		1	

4)

8

+

51 x 25

1283

3+) *branch delay slot* enabled

			cc
MAIN:	daddui	R1,R0,25	
	daddui	R2,R0,vetX	
	daddui	R3,R0,vetZ	
	daddui	R4,R0,vetY	
loop:	l.d	F1,0(R2)	
	l.d	F2,0(R3)	
	mul.d	F3,F1,F1	
	div.d	F4,F1,F2	
	add.d	F5,F3,F4	
	s.d	F5,0(R4)	

			cc
MAIN:	daddui	R1,R0,25	
	daddui	R2,R0,vetX	
	daddui	R3,R0,vetZ	
	daddui	R4,R0,vetY	
loop:	l.d	F1,0(R2)	
	l.d	F2,0(R3)	
	mul.d	F3,F1,F1	
	div.d	F4,F1,F2	
	add.d	F5,F3,F4	
	s.d	F5,0(R4)	
→	l.d	F1,8(R2)	
→	l.d	F2,8(R3)	
	mul.d	F3,F1,F1	
	div.d	F4,F1,F2	
	add.d	F5,F3,F4	
→	s.d	F5,8(R4)	

3)

			CC	
MAIN:	daddui	R1,R0,25	5	
	daddui	R2,R0,vetX	1	**8**
	daddui	R3,R0,vetZ	1	
	daddui	R4,R0,vetY	1	
loop:	l.d	F1,0(R2)	1	
	l.d	F2,0(R3)	1	
	mul.d	F3,F1,F1	4	**10**
	div.d	F4,F1,F2	1	
	add.d	F5,F3,F4	2	
	s.d	F5,0(R4)	1	
	l.d	F1,8(R2)	1	
	l.d	F2,8(R3)	1	
	mul.d	F3,F1,F1	4	**10**
	div.d	F4,F1,F2	1	
	add.d	F5,F3,F4	2	
	s.d	F5,8(R4)	1	

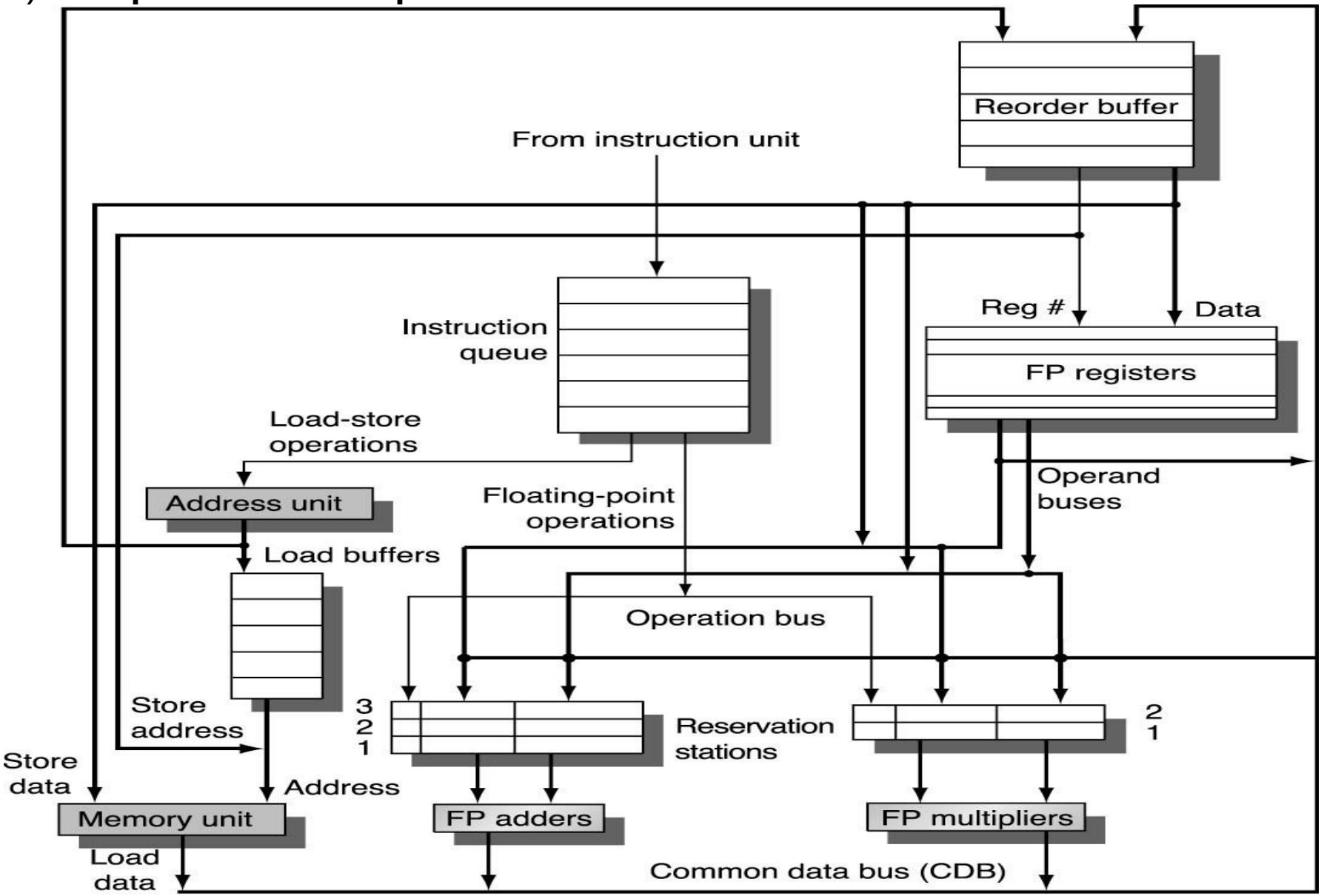
	l.d	F1,16(R2)	1	
	l.d	F2,16(R3)	1	
	mul.d	F3,F1,F1	4	**10**
	div.d	F4,F1,F2	1	
	add.d	F5,F3,F4	2	
	s.d	F5,16(R4)	1	
	l.d	F1,24(R2)	1	
	l.d	F2,24(R3)	1	
	mul.d	F3,F1,F1	4	
	div.d	F4,F1,F2	1	
	daddi	R2,R2,32	0	
	add.d	F5,F3,F4	2	**14**
	s.d	F5,24(R4)	1	
	daddi	R1,R1,-1	1	
	daddi	R3,R3,32	1	
	bnez	R1,loop	1	
	daddi	R4,R4,32	1	
	Halt		1	**1**

4)

8
+
44 x 25
+
1

1109

5) **Multiple-issue with speculation**



5)

	Instruction		Issue	EXE	READ MEM	CDBx2	COMMIT
1	l.d	F1,0(R2)					
	l.d	F2,0(R3)					
	mul.d	F3,F1,F1					
	div.d	F4,F1,F2					
	add.d	F5,F3,F4					
	s.d	F5,0(R4)					
	daddi	R2,R2,8					
	daddi	R3,R3,8					
	daddi	R4,R4,8					
	daddi	R1,R1,-1					
	bnez	R1,loop					
2	l.d	F1,0(R2)					
	l.d	F2,0(R3)					
	mul.d	F3,F1,F1					
	div.d	F4,F1,F2					
	add.d	F5,F3,F4					
	s.d	F5,0(R4)					
	daddi	R2,R2,8					
	daddi	R3,R3,8					
	daddi	R4,R4,8					
	daddi	R1,R1,-1					
	bnez	R1,loop					

5)

	Instruction		Issue	EXE		READ MEM	CDBx2	COMMIT
	l.d	F1,0(R2)	1	2	m	3	4	5
	l.d	F2,0(R3)	1	3	m	4	5	6
	mul.d	F3,F1,F1	2	5	x		9	10
	div.d	F4,F1,F2	2	6	d		10	11
	add.d	F5,F3,F4	3	11	a		13	14
1	s.d	F5,0(R4)						
	daddi	R2,R2,8						
	daddi	R3,R3,8						
	daddi	R4,R4,8						
	daddi	R1,R1,-1						
	bnez	R1,loop						
	l.d	F1,0(R2)						
	l.d	F2,0(R3)						
	mul.d	F3,F1,F1						
	div.d	F4,F1,F2						
	add.d	F5,F3,F4						
2	s.d	F5,0(R4)						
	daddi	R2,R2,8						
	daddi	R3,R3,8						
	daddi	R4,R4,8						
	daddi	R1,R1,-1						
	bnez	R1,loop						

5)

	Instruction		Issue	EXE		READ MEM	CDBx2	COMMIT
1	l.d	F1,0(R2)	1	2	m	3	4	5
	l.d	F2,0(R3)	1	3	m	4	5	6
	mul.d	F3,F1,F1	2	5	x		9	10
	div.d	F4,F1,F2	2	6	d		10	11
	add.d	F5,F3,F4	3	11	a		13	14
	s.d	F5,0(R4)	3	4	m			14
	daddi	R2,R2,8	4	5	i		6	15
	daddi	R3,R3,8	4	6	i		7	15
	daddi	R4,R4,8	5	7	i		8	16
	daddi	R1,R1,-1	5	8	i		9	16
	bnez	R1,loop	6	10	j			17
2	l.d	F1,0(R2)						
	l.d	F2,0(R3)						
	mul.d	F3,F1,F1						
	div.d	F4,F1,F2						
	add.d	F5,F3,F4						
	s.d	F5,0(R4)						
	daddi	R2,R2,8						
	daddi	R3,R3,8						
	daddi	R4,R4,8						
	daddi	R1,R1,-1						
	bnez	R1,loop						

5)

	Instruction		Issue	EXE		READ MEM	CDBx2	COMMIT
1	l.d	F1,0(R2)	1	2	m	3	4	5
	l.d	F2,0(R3)	1	3	m	4	5	6
	mul.d	F3,F1,F1	2	5	x		9	10
	div.d	F4,F1,F2	2	6	d		10	11
	add.d	F5,F3,F4	3	11	a		13	14
	s.d	F5,0(R4)	3	4	m			14
	daddi	R2,R2,8	4	5	i		6	15
	daddi	R3,R3,8	4	6	i		7	15
	daddi	R4,R4,8	5	7	i		8	16
	daddi	R1,R1,-1	5	8	i		9	16
	bnez	R1,loop	6	10	j			17
2	l.d	F1,0(R2)	7	8	m	9	10	17
	l.d	F2,0(R3)	7	9	m	10	11	18
	mul.d	F3,F1,F1	8	11	x		15	18
	div.d	F4,F1,F2	8	12	d		16	19
	add.d	F5,F3,F4	9	17	a		19	20
	s.d	F5,0(R4)	9	10	m			20
	daddi	R2,R2,8	10	11	i		12	21
	daddi	R3,R3,8	10	12	i		13	21
	daddi	R4,R4,8	11	13	i		14	22
	daddi	R1,R1,-1	11	14	i		15	22
	bnez	R1,loop	12	16	j			23

5+)

3

Instruction		Issue	EXE		READ MEM	CDBx2	COMMIT
l.d	F1,0(R2)	13	14	m	15	16	23
l.d	F2,0(R3)	13	15	m	16	17	24
mul.d	F3,F1,F1	14	17	x		21	24
div.d	F4,F1,F2	14	18	d		22	25
add.d	F5,F3,F4	15	23	a		25	26
s.d	F5,0(R4)	15	16	m			26
daddi	R2,R2,8	16	17	i		18	27
daddi	R3,R3,8	16	18	i		19	27
daddi	R4,R4,8	17	19	i		20	28
daddi	R1,R1,-1	17	20	i		21	28
bnez	R1,loop	18	22	j			29