Instruction-level parallelism: Tomasulo - A loop example

Dr. Tao Xie

These slides are adapted from notes by Dr. David Patterson (UCB)

Question 4 of HW3

- No instruction in the given code is labeled as the branch instruction target, so the target label must be elsewhere in the program.
- The answer “n/a” is ok for the storage location involved in a control dependence.
- Some are not hazards, some are not hazards if forwarding, some are hazards no matter forwarding or not.

	Dependence type	Independent instruction	Dependent instruction	Storage location
1	data	LD	DADD	R1

A Loop Example in C

This code multiplies a scalar to a vector:

```
for (i=100; i>0; i=i–1)
   x[i] = x[i] * s;
```

Tomasulo Loop Example

```
Loop: LD     F0   0    R1
      MULTD  F4   F0   F2
      SD     F4   0    R1
      SUBI   R1   R1   #8
      BNEZ   R1   Loop
```

- This time assume Multiply takes 4 clocks
- Assume 1st load takes 8 clocks (L1 cache miss), 2nd load takes 1 clock (hit)
- To be clear, will show clocks for SUBI, BNEZ
  - Reality: integer instructions ahead of Fl. Pt. Instructions
- Show 2 iterations

Loop Example

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1			
1	MULTD	F4	F0	F2			
1	SD	F4	0	R1			
2	LD	F0	0	R1			
2	MULTD	F4	F0	F2			
2	SD	F4	0	R1			

Iter-ation Count

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

Added Store Buffers

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2
SD    F4  0   R1
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Instruction Loop

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
0	80	Fu									

Value of Register used for address, iteration control

Loop Example Cycle 1

Instruction status:

ITER	Instruction	j	k	Issue	Exec Comp	Write Result
1	LD F0	0	R1	1		

	Busy	Addr	Fu
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Code:

```
LD    F0   0    R1
MULTD F4   F0   F2
SD    F4   0    R1
SUBI  R1   R1   #8
BNEZ  R1   Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
1	80	Fu	Load1								

Loop Example Cycle 2

Instruction status:

ITER	Instruction	*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
1	LD F0	0	R1	1		
1	MULTD F4	F0	F2	2		

	Busy	*Addr*	*Fu*
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	No		
Store2	No		
Store3	No		

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*Vj*	*S1 Vk*	*S2 Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load1	
	Mult2	No					

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2
SD    F4  0   R1
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
2	**80**	*Fu*	Load1		Mult1						

Loop Example Cycle 3

Instruction status:

ITER	Instruction	j	k	Issue	Exec Comp	Write Result
1	LD F0	0	R1	1		
1	MULTD F4	F0	F2	2		
1	SD F4	0	R1	3		

	Busy	Addr	Fu
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	S1 Vk	S2 Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load1	
	Mult2	No					

Code:

LD	F0	0	R1
MULTD	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
3	80	Fu	Load1		Mult1						

- Implicit renaming sets up data flow graph

Loop Example Cycle 4

Instruction status:

ITER	Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
1	LD	F0	0	R1	1		
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		

	Busy	*Addr*	*Fu*
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*Vj*	*S1 Vk*	*S2 Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load1	
	Mult2	No					

Code:

```
LD    F0  0  R1
MULTD F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8   <-
BNEZ  R1  Loop
```

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
4	**80**	*Fu*	Load1		Mult1						

- Dispatching SUBI Instruction (not in FP queue)

Loop Example Cycle 5

Instruction status:

ITER	Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
1	LD	F0	0	R1	1		
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		

	Busy	*Addr*	*Fu*
Load1	Yes	80	
Load2	No		
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*Vj*	*S1 Vk*	*S2 Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load1	
	Mult2	No					

Code:

```
LD    F0  0  R1
MULTD F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
5	**72**	*Fu*	Load1		Mult1						

- And, BNEZ instruction (not in FP queue)

Loop Example Cycle 6

Instruction status:

ITER	Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
1	LD	F0	0	R1	1		
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6		

	Busy	*Addr*	*Fu*
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*Vj*	*S1 Vk*	*S2 Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load1	
	Mult2	No					

Code:

```
LD    F0  0  R1   <-
MULTD F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
6	**72**	*Fu*	Load2		Mult1						

- Notice that F0 never sees Load from location 80

Loop Example Cycle 7

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1		
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6		
2	MULTD	F4	F0	F2	7		

	Busy	Addr	Fu
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	No		
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load1	
	Mult2	Yes	Multd		R(F2)	Load2	

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2   <-
SD    F4  0   R1
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
7	72	Fu	Load2		Mult2						

Observations?

- Register file completely detached from computation (how?)
- First and Second iteration completely overlapped

Loop Example Cycle 8

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1		
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6		
2	MULTD	F4	F0	F2	7		
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load1	
	Mult2	Yes	Multd		R(F2)	Load2	

Code:

```
LD    F0  0  R1
MULTD F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
8	72	Fu	Load2		Mult2						

Loop Example Cycle 9

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6		
2	MULTD	F4	F0	F2	7		
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	S2 Qk	RS
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	Multd		R(F2)	Load1		
	Mult2	Yes	Multd		R(F2)	Load2		

Code:

```
LD    F0  0  R1
MULTD F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8   <-
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
9	72	Fu	Load2		Mult2						

- Load1 completing: who is waiting?
- Can SUBI be dispatched?

Loop Example Cycle 10

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	?
2	MULTD	F4	F0	F2	7		
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	Vj	S1 Vk	S2 Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	Mult	M[80]	R(F2)		
	Mult2	Yes	Multd		R(F2)	Load2	

Code:

LD	F0	0	R1
MULTD	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
10	64	Fu	Load2		Mult2						

- Load2 completing: who is waiting?
- Can BNEZ be Dispatched?

Loop Example Cycle 11

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7		
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	Multd	M[80]	R(F2)		
4	Mult2	Yes	Multd	M[72]	R(F2)		

Code:

LD	F0	0	R1
MULTD	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
11	64	Fu	Load3		Mult2						

- Next load in sequence

Loop Example Cycle 12

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7		
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	S2 Qk	RS
	Add1	No						
	Add2	No						
	Add3	No						
2	Mult1	Yes	Multd	M[80]	R(F2)			
3	Mult2	Yes	Multd	M[72]	R(F2)			

Code:

```
LD    F0  0  R1
MULTD F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
12	64	Fu	Load3		Mult2						

- Why not issue third multiply?

Loop Example Cycle 13

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2		
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7		
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	S2 Qk	RS
	Add1	No						
	Add2	No						
	Add3	No						
1	Mult1	Yes	Multd	M[80]	R(F2)			
2	Mult2	Yes	Multd	M[72]	R(F2)			

Code:

LD	F0	0	R1
MULTD	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
13	64	Fu	Load3		Mult2						

- Why not issue third store?

Loop Example Cycle 14

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7		
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	S2 Qk	RS
	Add1	No						
	Add2	No						
	Add3	No						
0	Mult1	Yes	Multd	M[80]	R(F2)			
1	Mult2	Yes	Multd	M[72]	R(F2)			

Code:

```
LD    F0  0  R1
MULTD F4  F0 F2
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
14	64	Fu	Load3		Mult2						

- Mult1 completing. Who is waiting?

The third MULT is waiting (because of the structural hazard).

Loop Example Cycle 15

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	15
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	[80]*R2
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	Multd	M[72]		R(F2)	

Code:

```
LD    F0  0  R1
MULTD F4  F0 F2   <-
SD    F4  0  R1
SUBI  R1  R1 #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
15	64	Fu	Load3		Mult2						

- Mult2 completing. Who is waiting?

Store 2 is waiting.

Loop Example Cycle 16

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	15
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	[80]*R2
Store2	Yes	72	[72]*R2
Store3	No		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	S2 Qk	RS
	Add1	No						
	Add2	No						
	Add3	No						
4	Mult1	Yes	Multd		R(F2)	Load3		
	Mult2	No						

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2
SD    F4  0   R1
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
16	64	Fu	Load3		Mult1						

Loop Example Cycle 17

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	15
1	SD	F4	0	R1	3		
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8		

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	[80]*R2
Store2	Yes	72	[72]*R2
Store3	Yes	64	Mult1

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd	R(F2)	Load3		
	Mult2	No					

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2
SD    F4  0   R1   <-
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
17	64	Fu	Load3		Mult1						

Loop Example Cycle 18

Instruction status:

ITER	Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	15
1	SD	F4	0	R1	3	18	
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8		

	Busy	*Addr*	*Fu*
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	Yes	80	[80]*R2
Store2	Yes	72	[72]*R2
Store3	Yes	64	Mult1

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*Vj*	*S1 Vk*	*S2 Qj*	*RS Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load3	
	Mult2	No					

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2
SD    F4  0   R1
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
18	**64**	*Fu*	Load3		Mult1						

Loop Example Cycle 19

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	15
1	SD	F4	0	R1	3	18	19
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8	19	

	Busy	Addr	Fu
Load1	No		
Load2	No		
Load3	Yes	64	
Store1	No		
Store2	Yes	72	[72]*R2
Store3	Yes	64	Mult1

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	S2 Qk	RS
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	Multd		R(F2)	Load3		
	Mult2	No						

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2
SD    F4  0   R1
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
19	56	Fu	Load3		Mult1						

Loop Example Cycle 20

Instruction status:

ITER	Instruction		j	k	Issue	Exec Comp	Write Result
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	15
1	SD	F4	0	R1	3	18	19
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8	19	20

	Busy	Addr	Fu
Load1	Yes	56	
Load2	No		
Load3	Yes	64	
Store1	No		
Store2	No		
Store3	Yes	64	Mult1

Reservation Stations:

Time	Name	Busy	Op	Vj	S1 Vk	S2 Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load3	
	Mult2	No					

Code:

```
LD    F0  0   R1
MULTD F4  F0  F2
SD    F4  0   R1
SUBI  R1  R1  #8
BNEZ  R1  Loop
```

Register result status

Clock	R1		F0	F2	F4	F6	F8	F10	F12	...	F30
20	56	Fu	Load1		Mult1						

Observation?

- **Once again: In-order issue, out-of-order execution and out-of-order completion.**

Why can Tomasulo overlap iterations of loops?

- Register renaming
  - Multiple iterations use different physical destinations for registers (dynamic loop unrolling).
- Reservation stations
  - Permit instruction issue to advance past integer control flow operations
  - Also buffer old values of registers - totally avoiding the WAR stall that we saw in the scoreboard.
- Other perspective: Tomasulo building data flow dependency graph on the fly.

Tomasulo's scheme offers 2 major advantages

- The distribution of the hazard detection logic
  - distributed reservation stations and the CDB
  - If multiple instructions waiting on single result, & each instruction has other operand, then instructions can be released simultaneously by broadcast on CDB
  - If a centralized register file were used, the units would have to read their results from the registers when register buses are available.
- The elimination of stalls for WAW and WAR hazards

Summary

- Reservations stations: *implicit register renaming* to larger set of registers + buffering source operands
  - Prevents registers as bottleneck
  - Avoids WAR, WAW hazards of Scoreboard
  - Allows loop unrolling in HW
- Not limited to basic blocks
  (integer units gets ahead, beyond branches)
- Today, helps cache misses as well
  - Don’t stall for L1 Data cache miss
- Lasting Contributions
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- 360/91 descendants are Pentium III; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264

Instruction Dependence Example

- For the following code identify all data and name dependence between instructions and give the dependency graph

```
1   L.D     F0, 0 (R1)
2   ADD.D   F4, F0, F2
3   S.D     F4, 0(R1)
4   L.D     F0, -8(R1)
5   ADD.D   F4, F0, F2
6   S.D     F4, -8(R1)
```

Please find dependencies!

Instruction Dependence Example

- For the following code identify all data and name dependence between instructions and give the dependency graph

```
1    L.D     F0, 0 (R1)
2    ADD.D   F4, F0, F2
3    S.D     F4, 0(R1)
4    L.D     F0, -8(R1)
5    ADD.D   F4, F0, F2
6    S.D     F4, -8(R1)
```

True Data Dependence:

Instruction 2 depends on instruction 1 (instruction 1 result in F0 used by instruction 2), Similarly, instructions (4,5)
Instruction 3 depends on instruction 2 (instruction 2 result in F4 used by instruction 3), Similarly, instructions (5,6)

Name Dependence:

Output Name Dependence (WAW):

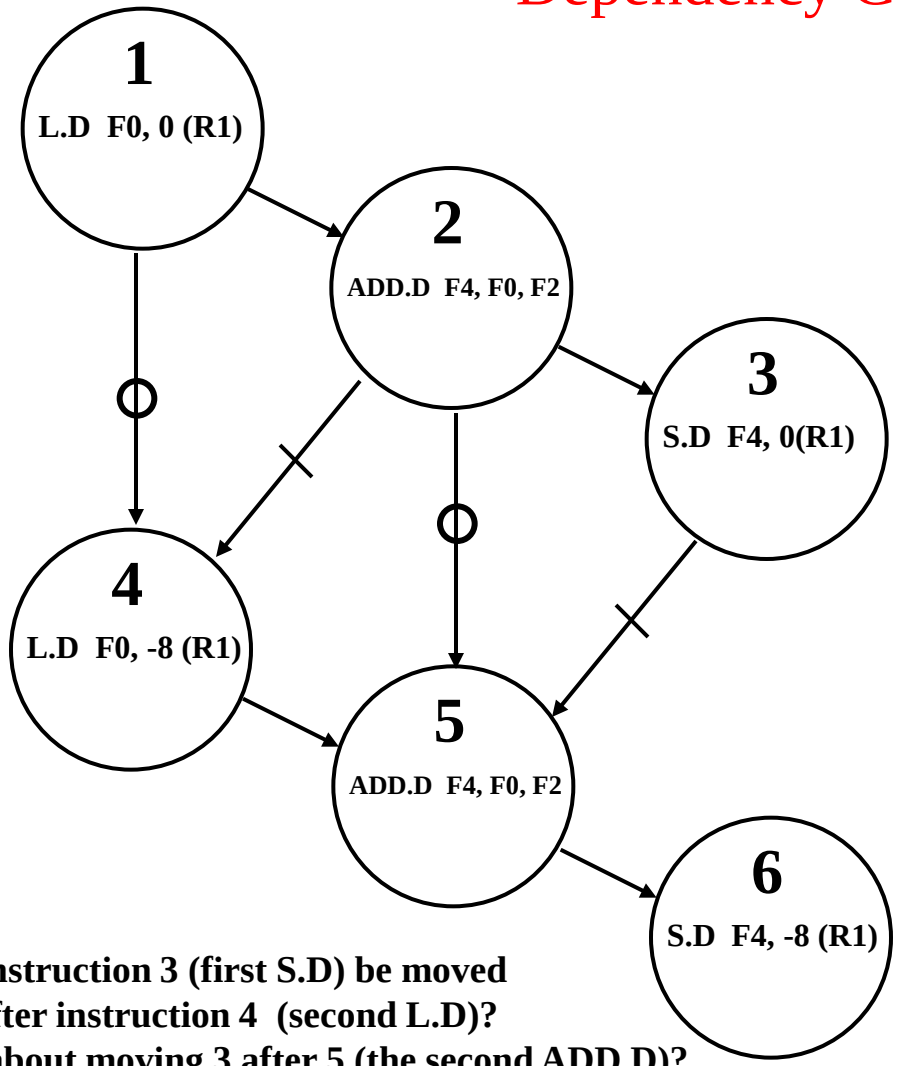
Instruction 1 has an output name dependence over result register (name) F0 with instructions 4
Instruction 2 has an output name dependence over result register (name) F4 with instructions 5

Anti-dependence (WAR):

Instruction 2 has an anti-dependence with instruction 4 over register (name) F0 which is an operand of instruction 1 and the result of instruction 4
Instruction 3 has an anti-dependence with instruction 5 over register (name) F4 which is an operand of instruction 3 and the result of instruction 5

Instruction Dependence Example

Dependency Graph



Example Code

1	L.D	F0, 0 (R1)
2	ADD.D	F4, F0, F2
3	S.D	F4, 0(R1)
4	L.D	F0, -8(R1)
5	ADD.D	F4, F0, F2
6	S.D	F4, -8(R1)

Date Dependence:
(1, 2) (2, 3) (4, 5) (5, 6)

Output Dependence:
(1, 4) (2, 5)

Anti-dependence:
(2, 4) (3, 5)

Can instruction 3 (first S.D) be moved just after instruction 4 (second L.D)? How about moving 3 after 5 (the second ADD.D)? If not what dependencies are violated?

Can instruction 4 (second L.D) be moved just after instruction 1 (first L.D)? If not what dependencies are violated?

Question?

- For the True Data Dependence on last slide, are they only applicable to two immediately adjacent instructions? No
- Why Instr 5 is not dependent on Instr 1 ?
- Is there a limit to how far apart the independent and dependent instructions can be from each other?

Because instruction 5 doesn't need the value of F0 that was generated by instruction 1. Instead, it needs the value produced by instruction 4.

As long as the independent and the dependent instructions will be executed in parallel, then there is a WAW or WAR hazard no matter how far away of the two instructions

A Scoreboard Example

The following code is run on the MIPS with a scoreboard given earlier with:

Functional Unit (FU)	# of FUs	EX cycles
Integer	1	1
Floating Point Multiply	2	10
Floating Point add	1	2
Floating point Divide	1	40

All functional units are not pipelined
(similar to CDC6600)

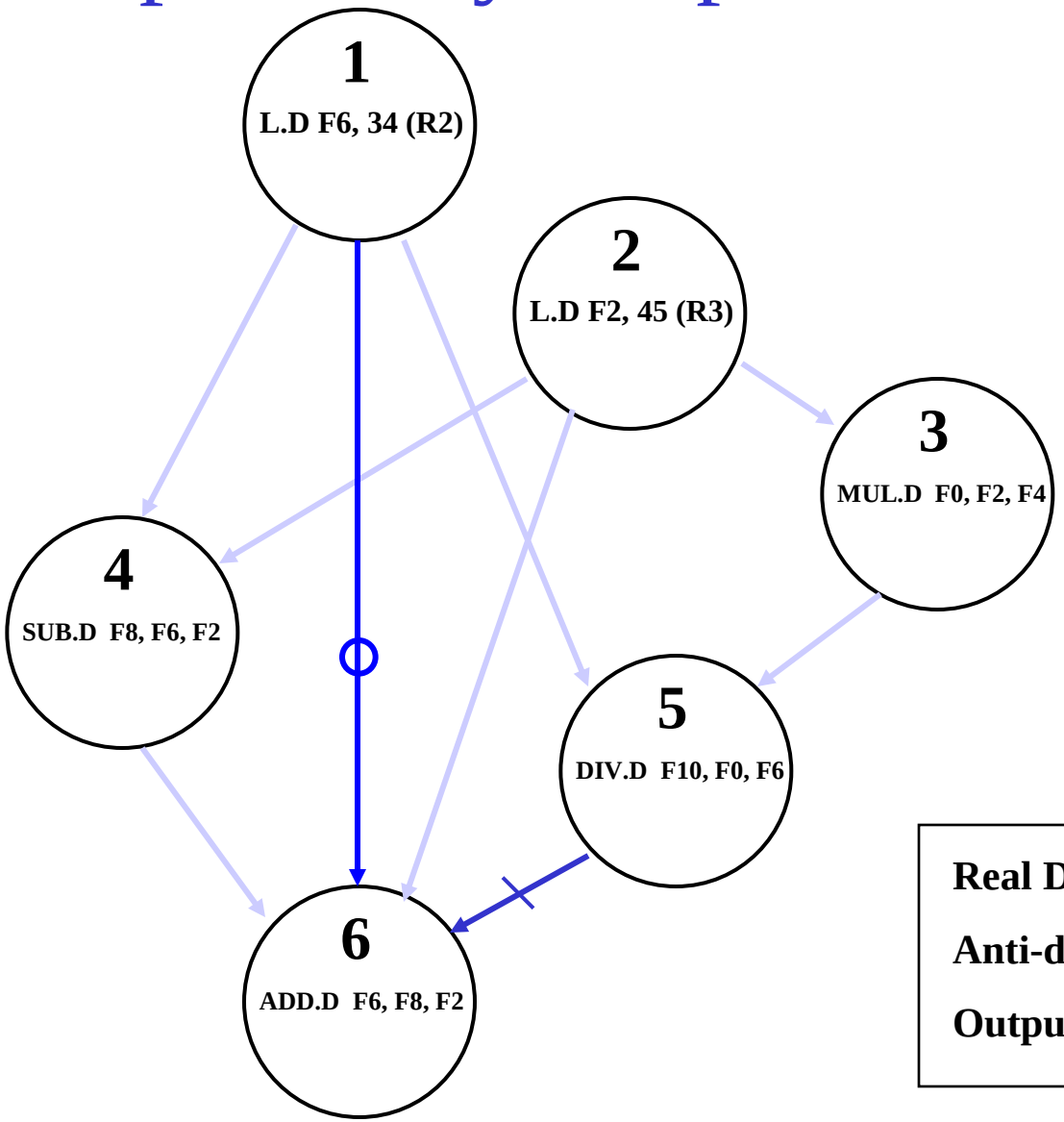
```
L.D      F6, 34(R2)
L.D      F2, 45(R3)
MUL.D    F0, F2, F4
SUB.D    F8, F6, F2
DIV.D    F10, F0, F6
ADD.D    F6, F8, F2
```

Real Data Dependence (RAW)

Anti-dependence (WAR)

Output Dependence (WAW)

Dependency Graph For Example Code



Example Code

1	L.D	F6, 34(R2)
2	L.D	F2, 45(R3)
3	MUL.D	F0, F2, F4
4	SUB.D	F8, F6, F2
5	DIV.D	F10, F0, F6
6	ADD.D	F6, F8, F2

Date Dependence:
(1, 4) (1, 5) (2, 3) (2, 4)
(2, 6) (3, 5) (4, 6)

Output Dependence:
(1, 6)

Anti-dependence:
(5, 6)

Real Data Dependence (RAW)

Anti-dependence (WAR)

Output Dependence (WAW)

Question

Looks like there you missed one anti dependency (WAR) for the given example

4 SUB.D F8, **F6**, F2
5 DIV.D F10, F0, F6
6 ADD.D **F6**, F8, F2

ADD is writing to F6 where as 4 and 5 are reading you missed to mention about (4,6)

Anti-dependence:
(5, 6)

Don’t need to worry about (4,6) as anti-dependency because there is a true-data dependency between the two instructions. Therefore, F6 will never be written by instruction 6 before instruction 4 reads F6.