ECSE 425 Lecture 15: More Dynamic Scheduling

H&P Chapter 2

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer
Textbook figures © 2007 Elsevier Science

Administrative Notes

- Midterm 1
  - 50 minutes, in class, October 12
    - I won't be there, but Alex will
  - Chapter 1, Appendix A, Chapter 2.1-2.3
  - One page crib sheet
    - Use both sides
    - Produce it however you like
    - Hand it in with your exam

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Last Time

- Dynamic Scheduling
  - Tomasulo's Algorithm

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Today

- Wrap up Dynamic Scheduling
  - Chapter 2.4
  - Tomasulo's Algorithm and Loops
  - Memory disambiguation
- Midterm Review

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Tomasulo Loop Example

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

- New assumptions
  - Multiply takes 4 cycles
  - Assume 1st load takes 8 cycle (L1 cache miss)
  - Assume 2nd load takes 1 cycle (L1 cache hit)
- To be clear, we will show clocks for SUBI, BNEZ
- We'll execute two iterations

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

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*	*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	

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*	*Vj*	*Vk*	*S1 Qj*	*S2 Qk*	*RS*
	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					

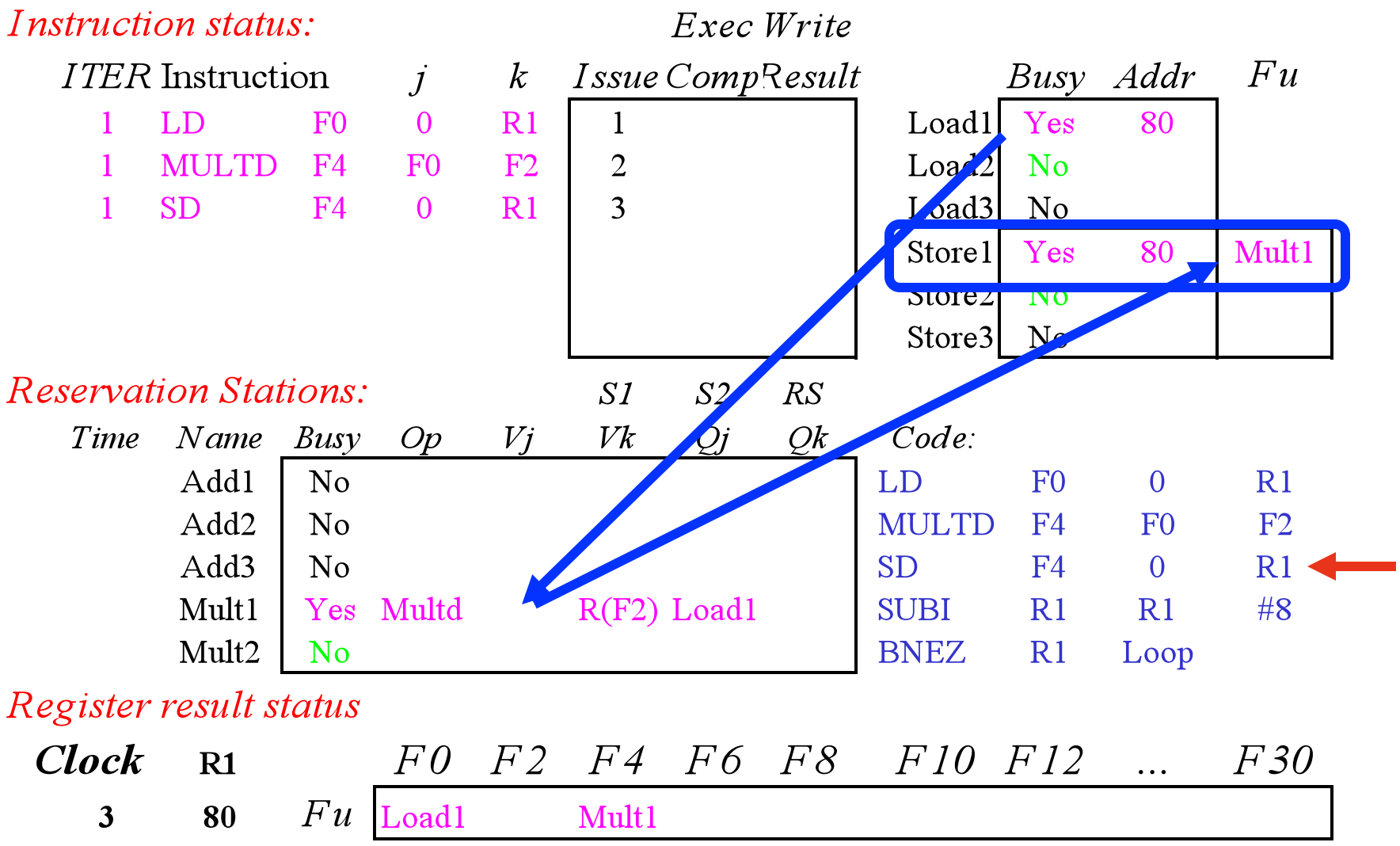
Code:

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



- Implicit renaming sets up data flow graph

Loop Example Cycle 4

Instruction status:

					Exec	Write
ITER	Instruction		*j*	*k*	*Issue*	*Comp*	*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:

					Exec	Write
ITER	Instruction		*j*	*k*	*Issue*	*Comp*	*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	S2 Vk	RS Qj	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						

- Register file completely detached from computation
- First and Second iteration completely overlapped

Loop Example Cycle 8

Instruction status:

						Exec	Write
ITER	Instruction		*j*	*k*	*Issue*	*Comp*	*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*	*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*
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	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
9	72	Fu	Load2		Mult2						

- Load1 completing: who is waiting?
- Note: Dispatching SUBI

Loop Example Cycle 10

Instruction status:

					Exec		Write				
ITER	Instruction		*j*	*k*	*Issue*	*Comp*	*Result*		*Busy*	*Addr*	*Fu*
1	LD	F0	0	R1	1	9	10	Load1	No		
1	MULTD	F4	F0	F2	2			Load2	Yes	72	
1	SD	F4	0	R1	3			Load3	No		
2	LD	F0	0	R1	6	10		Store1	Yes	80	Mult1
2	MULTD	F4	F0	F2	7			Store2	Yes	72	Mult2
2	SD	F4	0	R1	8			Store3	No		

Reservation Stations:

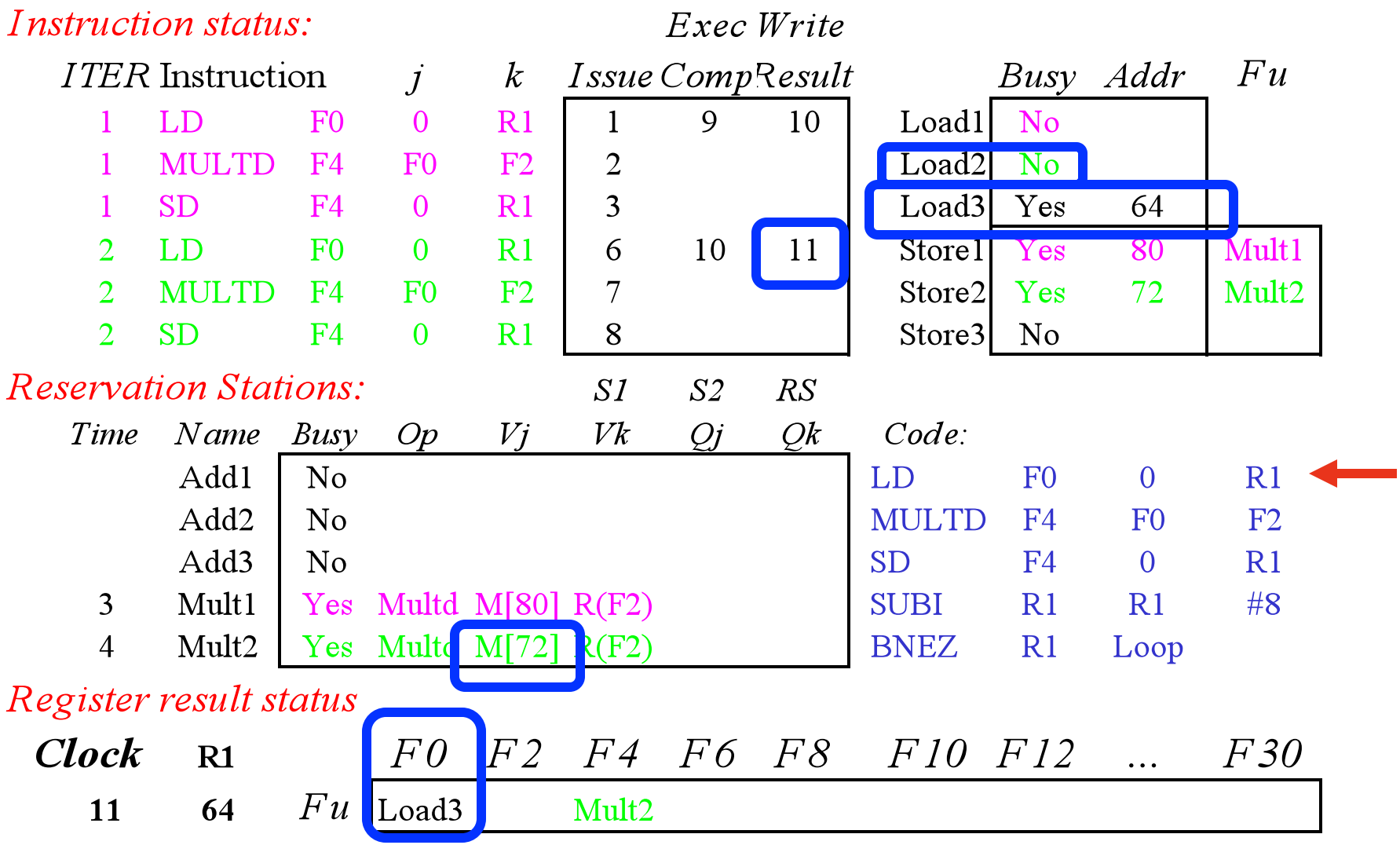
					S1	*S2*	*RS*	
Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*Code:*
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
4	Mult1	Yes	Multd	M[80]	R(F2)			SUBI R1 R1 #8
	Mult2	Yes	Multd		R(F2)	Load2		BNEZ R1 Loop

Register result status

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

- Load2 completing: who is waiting?
- Note: Dispatching BNEZ

Loop Example Cycle 11



- Next load in sequence

Loop Example Cycle 12

Instruction status:

					Exec		Write
ITER	Instruction		*j*	*k*	*Issue*	*Comp*	*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:

				S1	*S2*	*RS*	
Time	*Name*	*Busy*	*Op*	*Vj* *Vk*	*Qj*	*Qk*	*Code:*
	Add1	No					LD F0 0 R1
	Add2	No					MULTD F4 F0 F2
	Add3	No					SD F4 0 R1
2	Mult1	Yes	Multd	M[80] R(F2)			SUBI R1 R1 #8
3	Mult2	Yes	Multd	M[72] R(F2)			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?
- Load3 is completing, who is waiting?

Loop Example Cycle 13

Instruction status:

					Exec		Write
ITER	Instruction		*j*	*k*	*Issue*	*Comp*	*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	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

				S1	*S2*	*RS*	
Time	*Name*	*Busy*	*Op*	*Vj* *Vk*	*Qj*	*Qk*	*Code:*
	Add1	No					LD F0 0 R1
	Add2	No					MULTD F4 F0 F2
	Add3	No					SD F4 0 R1
1	Mult1	Yes	Multd	M[80] R(F2)			SUBI R1 R1 #8
2	Mult2	Yes	Multd	M[72] R(F2)			BNEZ R1 Loop

Register result status

Clock	**R1**		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
13	**64**	*Fu*	M[64]		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	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					
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	M[64]		Mult2						

- Mult1 completing. Who is waiting?

Loop Example Cycle 15

Instruction status:

					Exec		Write
ITER	Instruction		*j*	*k*	*Issue*	*Comp*	*Result*
1	LD	F0	0	R1	1	9	10
1	MULTD	F4	F0	F2	2	14	15
1	SD	F4	0	R1	3	15	
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	No		
Store1	Yes	80	[80]*R2
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					
	Mult1	Yes	Multd	R(F0)	R(F2)		
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*	M[64]		Mult2						

- Mult2 completing. Who is waiting?
- Issue the third multiply.

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	15	16
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8	16	

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

Reservation Stations:

Time	Name	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	Multd	R(F0)	R(F2)		
	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	M[64]		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	15	16
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8	16	17

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

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	Multd	R(F0)	R(F2)		
	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*	M[64]		Mult1						

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

Overlapping Loop Iterations in TA

- Register renaming
  - Dynamic loop unrolling
  - Multiple iterations use different physical registers
- Reservation stations (buffers)
  - Issue FP instructions, wait for control flow operations
  - Buffer old register values to avoiding WAR stall
- In other words, TA builds the dynamic data flow dependency graph

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Advantages of TA

- Distributed hazard detection logic
  - Distributed reservation stations and the CDB
  - The register file doesn't force serialization
- RS and the CDB
  - If multiple instructions wait on one result, and
  - If each instruction has its other operand
  - Then the instructions can be released simultaneously
- Centralized RF
  - Issue of instructions to FUs serialize on access to the RF
- Elimination of WAR and WAW stalls

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Dynamic Memory Disambiguation

- WAR and WAW hazards are eliminated by Tomasulo's algorithm by register renaming
  - Easy to do since the names are exposed
- What if two instructions share the same memory address ?

```
L.D   F1, 40(R6)
S.D   F4, 64(R3)
```

- What if 40(R6) = 64(R3)?

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Dynamic Memory Disambiguation

- WAR hazard if the load and store are re-ordered

```
L.D   F1, 40(R6)
S.D   F4, 64(R3)
```

- RAW hazard instead if the original order is reversed

```
S.D   F4, 64(R3)
L.D   F1, 40(R6)
```


© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Dynamic Memory Disambiguation

- Loads with the same effective address (EA) can proceed out of order relative to one another
- When loads and stores share the same EA
  - Later instructions must wait until earlier ones complete
- How do we know? Calculate EA in program order
  - Loads: check the store buffer for the same EA
  - Stores: check both load and store buffers

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Summary

- Tomasulo's Algorithm
  - Instructions are issued to reservation stations (RS)
  - Execution begins when all operands have arrived
  - Results are broadcast to all potentially waiting RS
- HW register renaming
  - Compensates for ISA's limited number of registers
- Dynamic latency tolerance
  - Compensates for compiler's inability to predict delay

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science

Next Time

- No class on Monday, October 10
- Midterm 1 on Wednesday, October 12
- Class resumes on Friday, October 14
  - Hardware Speculation (Chapter 2.5)

© 2011 Patterson, Gross, Hayward, Arbel, Vu, Meyer; © 2007 Elsevier Science