ECE 431
Parallel Computer Architecture
Spring 2019

Dynamic Scheduling

Nikos Bellas

Electrical and Computer Engineering Department
University of Thessaly

Instruction Level Parallelism

- Instruction Level Parallelism can be exploited by hardware AND/OR software mechanisms
  - Typically, transparent to the user
- *Dynamic scheduling* refers to the arrangement of instruction execution by hardware mechanisms
  - Goal is to reduce stalls due to instruction dependences
  - Data dependences (RAW), Name dependences (WAW), Anti-dependences (WAR)
  - Maintain data flow and exception flow
- Dynamic scheduling handles successfully cases where data dependences are unknown at compile time
  - For example, when they involve pointer references

Dynamic Scheduling: the Idea

- Dynamic scheduling re-arranges instruction issue and execution to reduce pipeline stalls
- Consider the following code:

```
DIV. D     F0, F2, F4      ; Large latency
ADD.D      F10, F0, F8
SUB.D       F12, F8, F14
```

  - The instruction *ADD.D* has to wait for *DIV.D* due to the *F0* data dependency
  - *SUB.D* does not have to wait.
  - Dynamic scheduling executes *SUB.D* **out-of-order always obeying RAW dependences.**

WAW and WAR dependences

- WAW and WAR dependences may stall a static pipeline when the CPU has few registers
- Compilers require a lot of registers to do register renaming and avoid WAR and WAW stalls


```
fdiv.d f0,f2,f4          fdiv.d f0,f2,f4
fadd.d f6,f0, f8         fadd.d S,f0,f8
fsd f6,0(x1)             fsd S,0(x1)
fsub.d f8,f10,f14        fsub.d T,f10,f14
fmul.d f6,f10,f8         fmul.d f6,f10,T
```

RAW

WAW

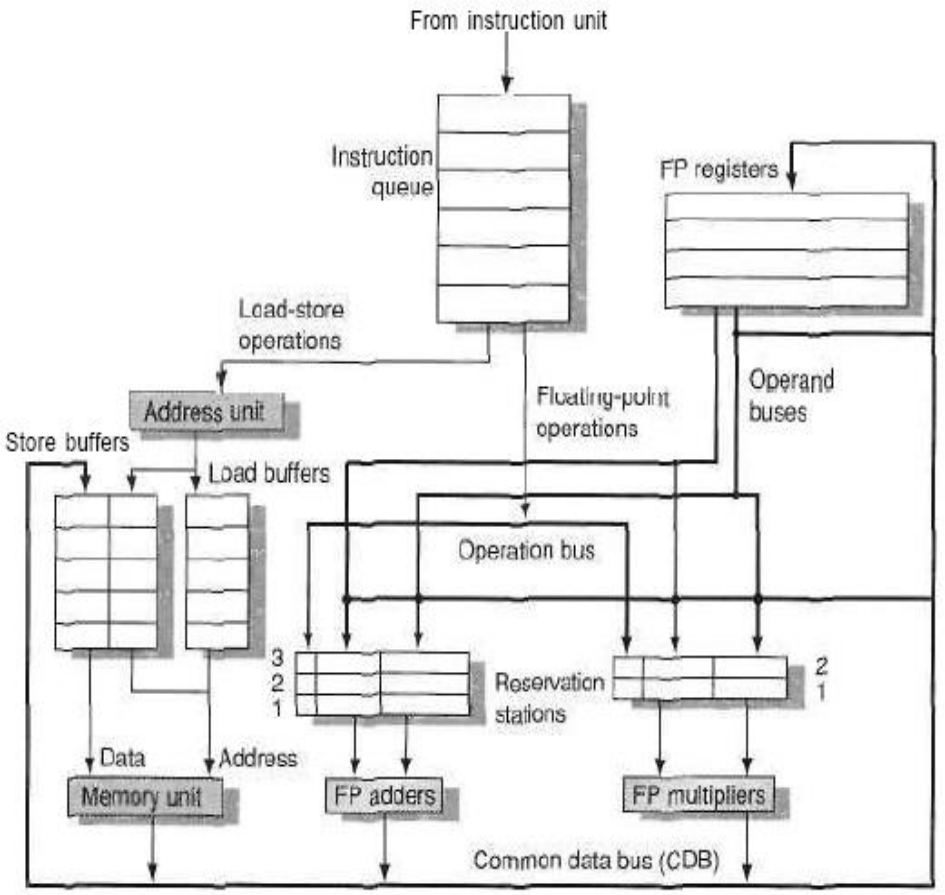
Not always possible to have two extra registers available (at compile time!)

Basic hardware structure for Tomasulo Algorithm

Tomasulo's algorithm widely used for dynamic scheduling in wide-issue machines

Three steps for instruction executi

- ✓ Issue (or dispatch)
- ✓ Execute (Not pipelined)
- ✓ Write back



EXAMPLE CODE

```
LD    F6, 34(R2)      ; 2 cycle latency
LD    F2, 45(R3)      ; 2 cycles
MULTD F0, F2, F4      ; 10 cycles
SUBD  F8, F6, F2      ; 2 cycles
DIVD  F10, F0, F6     ; 40 cycles
ADDD  F6, F8, F2      ; 2 cycles
```

Tomasulo Example

Instruction stream

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2			
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

3 Load/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					

FU count down

3 FP Adder R.S.

2 FP Mult R.S.

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
0	FU								

Clock cycle counter

Tomasulo Example Cycle 1

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	No	
Load3	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					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
1	FU				Load1					

Tomasulo Example Cycle 2

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1		
LD	F2	45+	R3	2		
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	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					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
2	FU		Load2		Load1					

Note: Can have multiple loads outstanding

Tomasulo Example Cycle 3

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	Yes	34+R2
Load2	Yes	45+R3
Load3	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(F4)	Load2	
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
3	FU	Mult1	Load2		Load1					

- Note: registers names are removed ("renamed") in Reservation Stations; MULT issued
- Load1 completing; what is waiting for Load1?

Tomasulo Example Cycle 4

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	Yes	45+R3
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	SUBD	M(A1)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	Load2	
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
4	FU	Mult1	Load2		M(A1)	Add1				

- **Load2 completing; what is waiting for Load2?**

Tomasulo Example Cycle 5

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2			

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
2	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	No					
	Add3	No					
10	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
5	FU	Mult1	M(A2)		M(A1)	Add1	Mult2			

- **Timer starts down for Add1, Mult1**

Tomasulo Example Cycle 6

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
9	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
6	FU	Mult1	M(A2)		Add2	Add1	Mult2			

- **Issue ADDD here despite name dependency on F6?**

Tomasulo Example Cycle 7

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	Add3	No					
8	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
7	FU	Mult1	M(A2)		Add2	Add1	Mult2			

- **Add1 (SUBD) completing; what is waiting for it?**

Tomasulo Example Cycle 8

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
2	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
7	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
8	*FU*	Mult1	M(A2)			Add2	(M-M)	Mult2		

Tomasulo Example Cycle 9

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*
	Add1	No					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
6	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
9	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

Tomasulo Example Cycle 10

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
0	Add2	Yes	ADDD	(M-M)	M(A2)		
	Add3	No					
5	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
10	*FU*	Mult1	M(A2)		Add2	(M-M)	Mult2			

- **Add2 (ADDD) completing; what is waiting for it?**

Tomasulo Example Cycle 11

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

- **Write result of ADDD here**
- **All quick instructions complete in this cycle!**

Tomasulo Example Cycle 12

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
12	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 13

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
13	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 14

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	M(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
14	*FU*	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 15

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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(A2)	R(F4)		
	Mult2	Yes	DIVD		M(A1)	Mult1	

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU	Mult1	M(A2)		(M-M+M	(M-M)	Mult2			

- **Mult1 (MULTD) completing; what is waiting for it?**

Tomasulo Example Cycle 16

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

- **Just waiting for Mult2 (DIVD) to complete**

Faster than light computation (skip a couple of cycles)

Tomasulo Example Cycle 55

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
55	FU	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

Tomasulo Example Cycle 56

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	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	DIVD	M*F4	M(A1)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(A2)		(M-M+M	(M-M)	Mult2			

- **Mult2 (DIVD) is completing; what is waiting for it?**

Tomasulo Example Cycle 57

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	57
ADDD	F6	F8	F2	6	10	11

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
56	FU	M*F4	M(A2)		(M-M+M	(M-M)	Result			

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

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 4 clocks (hit)
- To be clear, will show clocks for SUBI, BNEZ
  - Reality: integer instructions ahead of FP Instructions
- Show a few 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*	*Vj*	*S1 Vk*	*S2 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*	*S1 Vk*	*S2 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*	*Comp*	*Result*
1	LD	F0	0	R1	1		
1	MULTD	F4	F0	F2	2		

Exec Write

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

Reservation Stations:

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

S1 *S2* *RS*

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*	*Vk*	*Qj*	*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						

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

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	S1 Vj	S2 Vk	RS Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	Multd	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?

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

S1 *S2* *RS*

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?

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

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

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

Tomasulo's algorithm advantages

- It achieves high performance without requiring the compiler to target code to a specific pipeline
  - ✓ It eliminates the need to re-compile all code when a new architecture arrives
  - ✓ Compiler independence is by far the most important reason for the commercial success of dynamic scheduling
- It can detect instruction independence at run-time when the compiler cannot
- It is very flexible
  - ✓ It continuously schedules instructions if there are no data dependences and no structural hazards
  - ✓ Based on data-flow execution

Hardware-based speculative execution

- BUT: Tomasulo's algorithm cannot be applied in its original form
- Two problems:
- No provision for control dependencies
  - We should not commit results before we are certain that the path will be taken – For example, loop iterations
  - Overcoming control dependencies is critical for wide, superscalar processors
- Precise exception model cannot be supported with out -of-order instruction completion
  - What if a later instruction *InsB* which causes an exception commits BEFORE an instruction *InsA*

InsA
....
InsB

Precise Exceptions in Speculative Execution

- Exceptions are unexpected situations that occur during program execution
  - *Faults* are due to the execution of a particular instruction (e.g. Divide By Zero or page fault)
  - *Traps* are user-invoked O/S calls (e.g. fwrite)
  - *Interrupts* are asynchronous external hardware events

Precise Exceptions in Speculative Execution

- An exception is *precise* if the saved processor state corresponds to the sequential mode of program execution where one instruction execution ends before the next begins

- All instructions before the faulty one have committed and all instructions after have not modified the state of the machine and should start executing from the beginning

- The effect of the faulty instruction on the state of the system depends on definition of the architecture and the cause of the exception

Precise Exceptions in Speculative Execution

- There are terminating and non-terminating exceptions.
  - The former are usually program errors

- For example, timer or I/O exceptions are non-terminating. The execution of the instruction should *restart* after servicing the exception.
- Privileged instructions can only be executed in supervisor mode and will cause a fault.
- Unimplemented opcodes can be emulated in software and also cause a fault.

Precise Exceptions in Speculative Execution

- Exceptions should never be raised on instructions that will not execute

- Control dependent speculative instructions
  - e.g. Speculative memory instructions

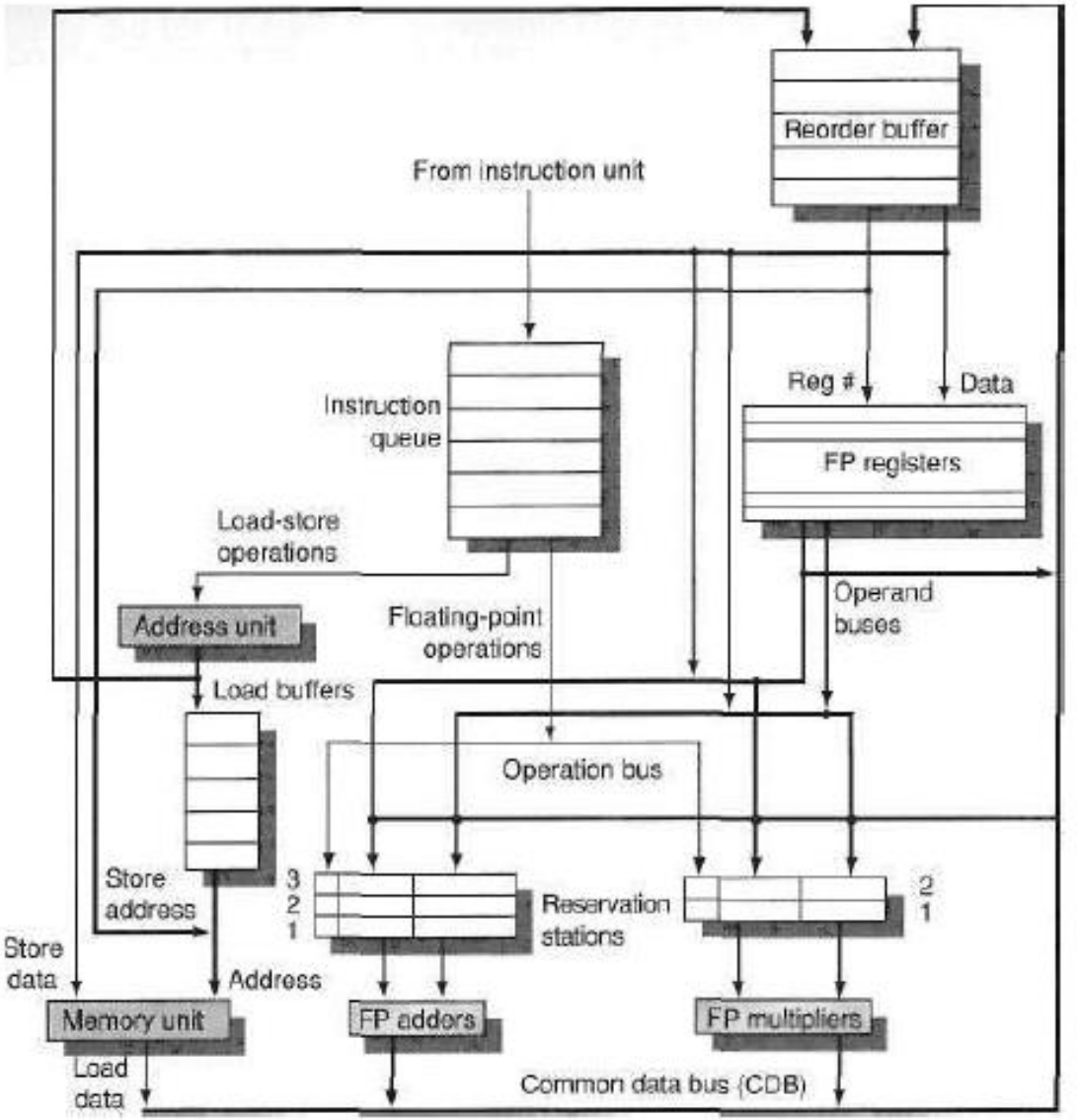
Hardware-based speculative execution

Four steps for instruction execution

- ✓ Issue (or dispatch)
- ✓ Execute
- ✓ Write back
- ✓ Commit (or Retire)

Do not modify the architectural state unless you are certain that the instruction will execute

A new data structure: Reorder Buffer (ROB)



Speculative execution

- Multiply takes 4 clocks
- Assume 1st load takes 8 clocks
  (L1 cache miss), 2nd load takes 4 clocks (hit)
- We will show clocks for branching overhead instructions

Speculative execution

Instruction stream

Reorder Buffer

Instruction status:

Instruction		j	k	*Issue*	*Exec Comp*	*Write Result*
LD	F0	0	R1			
MULD	F4	F0	F2			
SD	F4	0	R1			
DADDIU	R1	R1	#-8			
BNE	R1	R2	Loop			

Reorder Buffer

	Busy	*Instruction*	*State*	*Dest*	*Value*
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

Reservation Stations:

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS Qj*	*RS Qk*	*Dest*
	Load1							
	Load2							
	Add1							
	Add2							
	Add3							
	Mult1							
	Mult2							

3 FP Adder R.S.

2 FP Mult R.S.

Register result status:

Clock	*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
Reorder #									
Busy									

Clock cycle counter

Speculative execution: Cycle 1

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F0	0	R1	1		
MULD	F4	F0	F2			
SD	F4	0	R1			
DADDIU	R1	R1	#-8			
BNE	R1	R2	Loop			

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Issue	F0	
2					
3					
4					
5					
6					
7					
8					
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	0+R1			#1	
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	No						
	Mult2	No						

Register result status:

Clock 1

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	1								
Busy	Yes								

Speculative execution: Cycle 2

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F0	0	R1	1		
MULD	F4	F0	F2	2		
SD	F4	0	R1			
DADDIU	R1	R1	#-8			
BNE	R1	R2	Loop			

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3					
4					
5					
6					
7					
8					
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	0+R1	0	0	#1	
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD	R(F2)	#1		#2	
	Mult2	No						

Register result status:

Clock **2**

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	1		2						
Busy	Yes		Yes						

Speculative execution: Cycle 3

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F0	0	R1	1		
MULD	F4	F0	F2	2		
SD	F4	0	R1	3		
DADDIU	R1	R1	#-8			
BNE	R1	R2	Loop			

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4					
5					
6					
7					
8					
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	0+R1			#1	
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD		R(F2)	#1	#2	
	Mult2	No						

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
3									
Reorder #	1		2						
Busy	Yes		Yes						

Speculative execution: Cycle 4

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F0	0	R1	1		
MULD	F4	F0	F2	2		
SD	F4	0	R1	3		
DADDIU	R1	R1	#-8	4		
BNE	R1	R2	Loop			

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	Issue	R1	
5					
6					
7					
8					
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	0+R1			#1	
	Load2	No						
	Add1	Yes	DADDIU	R1	-8		#4	
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD		R(F2)	#1	#2	
	Mult2	No						

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
4									
Reorder #	1		2						
Busy	Yes		Yes						

Speculative execution: Cycle 5

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result
LD	F0	0	R1	1		
MULD	F4	F0	F2	2		
SD	F4	0	R1	3		
DADDIU	R1	R1	#-8	4	5	
BNE	R1	R2	Loop	5		

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	Exec	R1	
5	Yes	BNE	Issue		
6					
7					
8					
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	0+R1			#1	
	Load2	No						
	Add1	Yes	DADDIU	R1	-8		#4	
	Add2	Yes	BNE		R2	#4		
	Add3	No						
	Mult1	Yes	MULD		R(F2)	#1	#2	
	Mult2	No						

Register result status:

Clock **5**

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	1		2						
Busy	Yes		Yes						

Speculative execution: Cycle 6

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1			
MULD	F4	F0	F2	2			
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6		
LD	F0	0	R1	6			
MULD	F4	F0	F2				
SD	F4	0	R1				
DADDIU	R1	R1	#-8				
BNE	R1	R2	Loop				

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	Issue		
6	Yes	LD	Issue	F0	
7					
8					
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	80			#1	
	Load2	Yes	LD			#4	#6	
	Add1	Yes	DADDIU	R1	-8			#4
	Add2	Yes	BNE		R2	#4		
	Add3	No						
	Mult1	Yes	MULD		R(F2)	#1		#2
	Mult2	No						

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
6	Reorder #	6		2						
	Busy	Yes		Yes						

Speculative execution: Cycle 7

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1			
MULD	F4	F0	F2	2			
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6			
MULD	F4	F0	F2	7			
SD	F4	0	R1				
DADDIU	R1	R1	#-8				
BNE	R1	R2	Loop				

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	No	DADDIU	WrRes	R1	72
5	Yes	BNE	Exec		
6	Yes	LD	Exec	F0	
7	Yes	MULD	Issue	F4	
8					
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	80				#1
	Load2	Yes	LD	72				#6
	Add1							
	Add2	Yes	BNE	72	R2			
	Add3	No						
	Mult1	Yes	MULD		R(F2)	#1		#2
	Mult2	Yes	MULD		R(F2)	#6		#7

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
7	Reorder #	6		7					
	Busy	Yes		Yes					

Speculative execution: Cycle 8

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1			
MULD	F4	F0	F2	2			
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6			
MULD	F4	F0	F2	7			
SD	F4	0	R1	8			
DADDIU	R1	R1	#-8				
BNE	R1	R2	Loop				

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	Exec	F0	
7	Yes	MULD	Issue	F4	
8	Yes	SD	Issue	72	#7
9					
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	80			#1	
	Load2	Yes	LD	72			#6	
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD		R(F2)	#1	#2	
	Mult2	Yes	MULD		R(F2)	#6	#7	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
8	6		7						
Reorder #									
Busy	Yes		Yes						

Speculative execution: Cycle 9

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1	9		
MULD	F4	F0	F2	2			
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6			
MULD	F4	F0	F2	7			
SD	F4	0	R1	8			
DADDIU	R1	R1	#-8	9			
BNE	R1	R2	Loop				

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	Exec	F0	
2	Yes	MULD	Issue	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	Exec	F0	
7	Yes	MULD	Issue	F4	
8	Yes	SD	Issue	72	#7
9	Yes	DADDIU	Issue	R1	
10					

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	80			#1	
	Load2	Yes	LD	72			#6	
	Add1	Yes	DADDIU	72	-8		#9	
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD		R(F2)	#1	#2	
	Mult2	Yes	MULD		R(F2)	#6	#7	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
9	Reorder #		6						
	Busy		Yes						

Speculative execution: Cycle 10

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1	9	10	
MULD	F4	F0	F2	2			
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6	10		
MULD	F4	F0	F2	7			
SD	F4	0	R1	8			
DADDIU	R1	R1	#-8	9	10		
BNE	R1	R2	Loop	10			

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	Yes	LD	WrRes	F0	M(80)
2	Yes	MULD	Exec	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	Exec	F0	
7	Yes	MULD	Issue	F4	
8	Yes	SD	Issue	72	#7
9	Yes	DADDIU	Exec	R1	64
10	Yes	BNE	Issue		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	Yes	LD	80				#1
	Load2	Yes	LD	72				#6
	Add1	Yes	DADDIU	72	-8			#9
	Add2	Yes	BNE		R2	#9		
	Add3	No						
	Mult1	Yes	MULD	M[80]	R(F2)			#2
	Mult2	Yes	MULD		R(F2)	#6		#7

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
10									
Reorder #	6		7						
Busy	Yes		Yes						

Speculative execution: Cycle 11

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1	9	10	11
MULD	F4	F0	F2	2			
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6	10	11	
MULD	F4	F0	F2	7			
SD	F4	0	R1	8			
DADDIU	R1	R1	#-8	9	10	11	
BNE	R1	R2	Loop	10	11		

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	No	LD	Commit	F0	M(80)
2	Yes	MULD	Exec	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	WrRes	F0	M(72)
7	Yes	MULD	Exec	F4	
8	Yes	SD	Issue	72	#7
9	Yes	DADDIU	WrRes	R1	64
10	Yes	BNE	Exec		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	No						
	Load2	No						
	Add1	No						
	Add2	Yes	BNE		R2	#9		
	Add3	No						
	Mult1	Yes	MULD	M[80]	R(F2)		#2	
	Mult2	Yes	MULD	M[72]	R(F2)		#7	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
11									
Reorder #	M[72]		7						
Busy	Yes		Yes						

Speculative execution: Cycle 12

Instruction status:

	Instruction		j	k	Issue	Exec Comp	Write Result	Commit
1	LD	F0	0	R1	1	9	10	11
2	MULD	F4	F0	F2	2			
3	SD	F4	0	R1	3			
4	DADDIU	R1	R1	#-8	4	5	6	
5	BNE	R1	R2	Loop	5	6	7	
6	LD	F0	0	R1	6	10	11	
7	MULD	F4	F0	F2	7			
8	SD	F4	0	R1	8			
9	DADDIU	R1	R1	#-8	9	10	11	
10	BNE	R1	R2	Loop	10	11	12	

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	No	LD	Commit	F0	M(80)
2	Yes	MULD	Exec	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	WrRes	F0	M(72)
7	Yes	MULD	Exec	F4	
8	Yes	SD	Issue	72	#7
9	Yes	DADDIU	WrRes	R1	
10	Yes	BNE	WrRes		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	No						
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD	M[80]	R(F2)		#2	
	Mult2	Yes	MULD	M[72]	R(F2)		#7	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
12									
Reorder #			7						
Busy			Yes						

Speculative execution: Cycle 13

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1	9	10	11
MULD	F4	F0	F2	2			
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6	10	11	
MULD	F4	F0	F2	7			
SD	F4	0	R1	8			
DADDIU	R1	R1	#-8	9	10	11	
BNE	R1	R2	Loop	10	11	12	

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	No	LD	Commit	F0	M(80)
2	Yes	MULD	Exec	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	WrRes	F0	M(72)
7	Yes	MULD	Exec	F4	
8	Yes	SD	Issue	72	#7
9	Yes	DADDIU	WrRes	R1	64
10	Yes	BNE	WrRes		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	No						
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD	M[80]	R(F2)		#2	
	Mult2	Yes	MULD	M[72]	R(F2)		#7	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
13									
Reorder #			7						
Busy			Yes						

Speculative execution: Cycle 14

Instruction status:

	Instruction		j	k	Issue	Exec Comp	Write Result	Commit
1	LD	F0	0	R1	1	9	10	11
2	MULD	F4	F0	F2	2	14		
3	SD	F4	0	R1	3			
4	DADDIU	R1	R1	#-8	4	5	6	
5	BNE	R1	R2	Loop	5	6	7	
6	LD	F0	0	R1	6	10	11	
7	MULD	F4	F0	F2	7			
8	SD	F4	0	R1	8			
9	DADDIU	R1	R1	#-8	9	10	11	
10	BNE	R1	R2	Loop	10	11	12	

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	No	LD	Commit	F0	M(80)
2	Yes	MULD	Exec	F4	
3	Yes	SD	Issue	M(0+R1)	#2
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	WrRes	F0	M(72)
7	Yes	MULD	Exec	F4	
8	Yes	SD	Issue	72	#7
9	Yes	DADDIU	WrRes	R1	64
10	Yes	BNE	WrRes		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	No						
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	Yes	MULD M[80]	R(F2)			#2	
	Mult2	Yes	MULD M[72]	R(F2)			#7	

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
14									
Reorder #			7						
Busy			Yes						

Speculative execution: Cycle 15

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1	9	10	11
MULD	F4	F0	F2	2	14	15	
SD	F4	0	R1	3			
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6	10	11	
MULD	F4	F0	F2	7	15		
SD	F4	0	R1	8			
DADDIU	R1	R1	#-8	9	10	11	
BNE	R1	R2	Loop	10	11	12	

Reorder Buffer

	Busy	Instruction	State	Dest	Value
1	No	LD	Commit	F0	M(80)
2	Yes	MULD	WrRes	F4	M(80)*F0
3	Yes	SD	Issue	M(0+R1)	M(80)*F0
4	Yes	DADDIU	WrRes	R1	72
5	Yes	BNE	WrRes		
6	Yes	LD	WrRes	F0	M(72)
7	Yes	MULD	Exec	F4	
8	Yes	SD	Issue	72	#7
9	Yes	DADDIU	WrRes	R1	64
10	Yes	BNE	WrRes		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	No						
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	No						
	Mult2	Yes	MULD	M[72]	R(F2)			#7

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
15	Reorder #		7						
	Busy		Yes						

Speculative execution: Cycle 16

Instruction status:

Instruction		j	k	Issue	Exec Comp	Write Result	Commit
LD	F0	0	R1	1	9	10	11
MULD	F4	F0	F2	2	14	15	16
SD	F4	0	R1	3	16		
DADDIU	R1	R1	#-8	4	5	6	
BNE	R1	R2	Loop	5	6	7	
LD	F0	0	R1	6	10	11	
MULD	F4	F0	F2	7	15	16	
SD	F4	0	R1	8			
DADDIU	R1	R1	#-8	9	10	11	
BNE	R1	R2	Loop	10	11	12	

1 2 3 4 5 6 7 8 9 10

Reorder Buffer

Busy	Instruction	State	Dest	Value
No	LD	Commit	F0	M(80)
No	MULD	Commit	F4	M(80)*F2
Yes	SD	Exec	M(0+R1)	M(80)*F2
Yes	DADDIU	WrRes	R1	72
Yes	BNE	WrRes		
Yes	LD	WrRes	F0	M(72)
Yes	MULD	Exec	F4	M(72)*F2
Yes	SD	Issue	72	M(72)*F2
Yes	DADDIU	WrRes	R1	64
Yes	BNE	WrRes		

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk	Dest
	Load1	No						
	Load2	No						
	Add1	No						
	Add2	No						
	Add3	No						
	Mult1	No						
	Mult2	No						

Register result status:

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
16									
Reorder #			7						
Busy			Yes						

- **In-order issue, out-of-order execution and in-order completion (commit).**

Speculation and Dependences

The STORE instructions write to memory when they commit

The LOAD instructions load from memory at EXEC stage

RAW dependences

SD F0, 0(R1)

.....

LD F1, 0(R1)

Do not initiate a LOAD from memory if a STORE occupies an earlier active ROB entry with the same Address

WAW and WAR dependences

WAW and WAR hazards cannot happen since stores always execute in order when the store is at the head of the ROB

How do we exploit ILP?

- Previous examples only showed one commit per cycle
  - Although multiple instructions executed per cycle
- Dynamic scheduling can issue and commit multiple instructions per cycle
  - Multiple instruction issue per cycle is bottleneck (typically 4 instr/cycle)
  - Need to check for dependences and to update ROB and Res. Station in a single cycle
- All instructions ready to commit can do so provided they are in successive locations in the Reorder Buffer.

Issue Logic As ILP Bottleneck

6-issue CPU

```
add $t1, $t2, $t3
sub $t1, $t1, $t2
add $t2, $t1, $t2
sub $t1, $t3, $t2
add $t1, $t1, $t2
sub $t1, $t3, $t1
```

Reorder Buffer

	Busy	*Instruction*	*State*	*Dest*	*Value*
1	Yes	add	Issue	$t1	$t2+$t3
2					
3					
4					
5					
6					
7					
8					
9					
.					
.					

Register Result Status

	$t1	*$t2*	*$t3*	*$t4*
Reorder #	#1			
Busy	*Yes*	*No*	*No*	*No*

Issue Logic As ILP Bottleneck

6-issue CPU

```
add $t1, $t2, $t3
sub $t1, $t1, $t2
add $t2, $t1, $t2
sub $t1, $t3, $t2
add $t1, $t1, $t2
sub $t1, $t3, $t1
```

Reorder Buffer

	Busy	*Instruction*	*State*	*Dest*	*Value*
1	Yes	add	Issue	$t1	$t2+$t3
2	Yes	sub	Issue	$t1	#1 +$t2
3					
4					
5					
6					
7					
8					
9					
.					
.					

Register Result Status

	$t1	*$t2*	*$t3*	*$t4*
Reorder #	#2			
Busy	*Yes*	*No*	*No*	*No*

Issue Logic As ILP Bottleneck

6-issue CPU

```
add $t1, $t2, $t3
sub $t1, $t1, $t2
add $t2, $t1, $t2
sub $t1, $t3, $t2
add $t1, $t1, $t2
sub $t1, $t3, $t1
```

Reorder Buffer

	Busy	*Instruction*	*State*	*Dest*	*Value*
1	Yes	add	Issue	$t1	$t2+$t3
2	Yes	sub	Issue	$t1	#1 +$t2
3	Yes	add	Issue	$t2	#2 +$t2
4					
5					
6					
7					
8					
9					
.					
.					

Register Result Status

	$t1	*$t2*	*$t3*	*$t4*
Reorder #	#2	#3		
Busy	*Yes*	*Yes*	*No*	*No*

Issue Logic As ILP Bottleneck

6-issue CPU

```
add $t1, $t2, $t3
sub $t1, $t1, $t2
add $t2, $t1, $t2
sub $t1, $t3, $t2
add $t1, $t1, $t2
sub $t1, $t3, $t1
```

Reorder Buffer

	Busy	*Instruction*	*State*	*Dest*	*Value*
1	Yes	add	Issue	$t1	$t2 + $t3
2	Yes	sub	Issue	$t1	#1 + $t2
3	Yes	add	Issue	$t2	#2 + $t2
4	Yes	sub	Issue	$t1	$t3 + #3
5					
6					
7					
8					
9					
.					
.					

Register Result Status

	$t1	*$t2*	*$t3*	*$t4*
Reorder #	#4	#3		
Busy	*Yes*	*Yes*	*No*	*No*

Issue Logic As ILP Bottleneck

6-issue CPU

```
add $t1, $t2, $t3
sub $t1, $t1, $t2
add $t2, $t1, $t2
sub $t1, $t3, $t2
add $t1, $t1, $t2
sub $t1, $t3, $t1
```

Reorder Buffer

	Busy	*Instruction*	*State*	*Dest*	*Value*
1	Yes	add	Issue	$t1	$t2 + $t3
2	Yes	sub	Issue	$t1	#1 + $t2
3	Yes	add	Issue	$t2	#2 + $t2
4	Yes	sub	Issue	$t1	$t3 + #3
5	Yes	add	Issue	$t1	#4 + #3
6					
7					
8					
9					
.					
.					

Register Result Status

	$t1	*$t2*	*$t3*	*$t4*
Reorder #	#5	#3		
Busy	*Yes*	*Yes*	*No*	*No*

Issue Logic As ILP Bottleneck

ALL THESE IN A SINGLE CLOCK CYCLE

```
add $t1, $t2, $t3
sub $t1, $t1, $t2
add $t2, $t1, $t2
sub $t1, $t3, $t2
add $t1, $t1, $t2
sub $t1, $t3, $t1
```

Reorder Buffer

	Busy	*Instruction*	*State*	*Dest*	*Value*
1	Yes	add	Issue	$t1	$t2 + $t3
2	Yes	sub	Issue	$t1	#1 + $t2
3	Yes	add	Issue	$t2	#2 + $t2
4	Yes	sub	Issue	$t1	$t3 + #3
5	Yes	add	Issue	$t1	#4 + #3
6	Yes	sub	Issue	$t1	$t3 + #5
7					
8					
9					
.					
.					

Register Result Status

	$t1	*$t2*	*$t3*	*$t4*
Reorder #	#6	#3		
Busy	*Yes*	*Yes*	*No*	*No*