Ch3--
ILP & its exploration

Ch3-2

- **Dynamic scheduling**
  --Tomasulo Algorithm
- **Scoreboard + Register Renaming**

3.4， 3.5

Scoreboard vs. Tomasulo

- 特点
  - Multiple multiplier, etc. Funcs
  - Issue in order, Complete OOO
  - IF→ Issue, Ro
  - 4 stages pipeline
  - Scoreboare centralized control
- 缺点
  - Stall when WAW, WAR

- Fewer Func, unpipelined
- Issue in order, Complete OOO
- FP op. queue, Reservation station, LD/ST buffer, CDB
- Reg. Rename→No WAW, WAR
- Reduce structural hazard
- RAW detection decentralized—reservatio
- CDB→ forwarding path

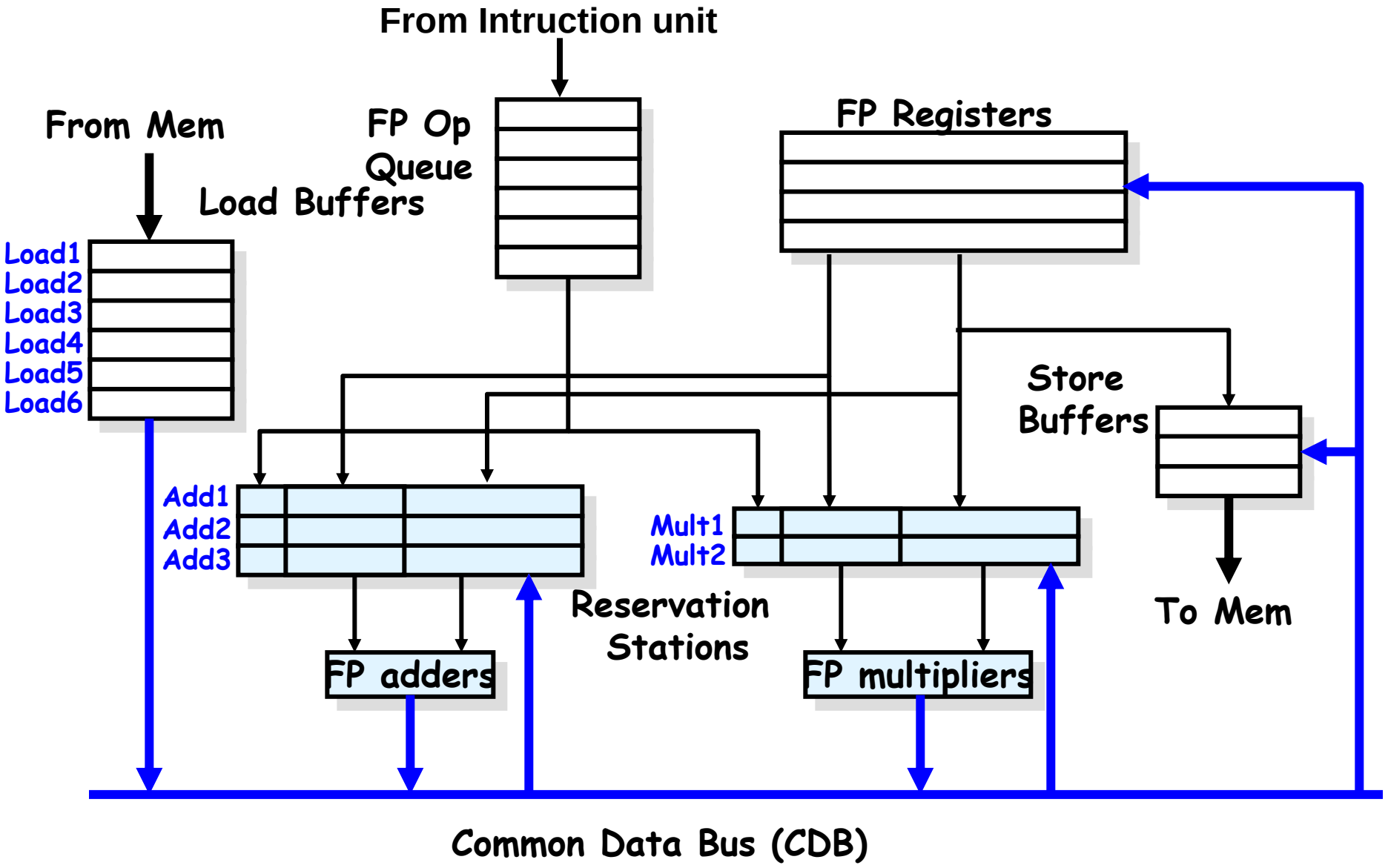
Dynamic Scheduling with Tomasulo's Algorithm

- For IBM 360/91 (before caches!)
- Goal: High Performance without special compilers
- Small number of floating point registers (4 in 360) prevented interesting compiler scheduling of operations
  - This led Tomasulo to try to figure out how to get more effective registers — renaming in hardware!
- Why Study 1966 Computer?
- The descendants of this have flourished!
  - Alpha 21264, HP 8000, MIPS 10000, Pentium III, PowerPC 604, …

Tomasulo Algorithm

- Control & buffers distributed with Function Units (FU)
  - FU buffers called "reservation stations"; have pending operands
- Registers in instructions replaced by values or pointers to reservation stations(RS); called register renaming ;
  - avoids WAR, WAW hazards
  - More reservation stations than registers, so can do optimizations compilers can't
- Results to FU from RS, not through registers, over Common Data Bus that broadcasts results to all FUs
- Load and Stores treated as FUs with RSs as well
- Integer instructions can go past branches, allowing FP ops beyond basic block in FP queue

Tomasulo Organization



Reservation Station Components

Reservation station:

- Op: Operation to perform in the unit
- Vj, Vk: Value of Source operands
  - Store buffers has V field, result to be stored
- Qj, Qk: Reservation stations producing source registers (value to be written)
  - Note: Qj,Qk=0 => ready
  - Store buffers only have Qi for RS producing result
- A: hold info. for memory address calculation
- Busy: Indicates reservation station or FU is busy

Register result status—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions that will write that register.

Three Stages of Tomasulo Algorithm

- Issue—get instruction from FP Op Queue

  If reservation station free (no structural hazard), control issues instr & sends operands (renames registers).

- Execute—operate on operands (EX)

  When both operands ready then execute;
  if not ready, watch Common Data Bus for result

- Write result—finish execution (WB)

  Write on Common Data Bus to all awaiting units; mark reservation station available

Data path

- Normal data bus: data + destination (“go to” bus)
- Common data bus: data + source (“come from” bus)
  - 64 bits of data + 4 bits of Functional Unit source address
  - Write if matches expected Functional Unit (produces result)
  - Does the broadcast
- Example speed:
  3 clocks for Fl .pt. +,-; 10 for * ; 40 clks for /

Instruction state	Wait until	Action or bookkeeping
Issue FP Operation	Station r empty	`if (Register Stat[rs].Qi ≠0) {RS[r].Qj← RegisterStat[rs].Qi} else {RS[r].Vj← Regs[rs]; RS[r].Qj← 0}; if (RegisterStat[rt].Qi≠0) {RS[r].Qk← RegisterStat[rt]Q.i} else {RS[r].Vk← Regs[rt]; RS[r].Qk← 0}; RS[r].Busy← yes; RegisterStat[rd].Qi=r;`
Load or Store	Buffer r empty	`if (Register Stat[rs].Qi ≠0) {RS[r].Qj← RegisterStat[rs].Qi} else {RS[r].Vj← Regs[rs]; RS[r].Qj← 0}; RS[r].A← imm; RS[r].Busy← yes;`
Load only		`RegisterStat[rt].Qi=r;`
Store only		`if (Register Stat[rt].Qi ≠0) {RS[r].Qk← RegisterStat[rs].Qi} else {RS[r].Vk← Regs[rt]; RS[r].Qk← 0};`
Execute FP Operation	(RS[r].Qj=0) and (RS[r].Qk=0)	Compute result: operands are in Vj and Vk
Load/Store step 1	RS[r].Qj=0 & r is head of load/store queue	`RS[r].A←RS[r].Vj + RS[r].A;`
Load step 2	RS[r].A<>0	Read from Mem[RS[r].A]
Write result FP Operation or Load	Execution complete at r & CDB available	`∀x(if (RegisterStat[x].Qi=r) {Regs[x]← result; RegisterStat[x].Qi← 0}); ∀x(if (RS[x].Qj=r) {RS[x].Vj← result;RS[x].Qj ← 0}); ∀x(if (RS[x].Qk=r) {RS[x].Vk← result;RS[x].Qk← 0}); RS[r].Busy← no;`
Store	Execution complete at r & RS[r].Qk=0	`Mem[RS[r].A]←RS[r].Vk; RS[r].Busy← no;`

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 Start*	*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 Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2		
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 Start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	
LD	F2	45+	R3	2	3		
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 Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	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 Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	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*
3	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 Start	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6		
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
2	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 Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6		
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					
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			

Tomasulo Example Cycle 8

Instruction status:

Instruction		*j*	*k*	*Issue*	Exec start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	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*
0	Add1	Yes	SUBD	M(A1)	M(A2)		
	Add2	Yes	ADDD		M(A2)	Add1	
	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	Add1	Mult2			

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

Tomasulo Example Cycle 9

Instruction status:

Instruction		j	k	Issue	Exec start	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
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					
3	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 start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
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					
2	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			

Tomasulo Example Cycle 11

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
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					
1	Add2	Yes	ADDD	(M-M)	M(A2)		
	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			

Tomasulo Example Cycle 12

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	12	

	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					
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)		ADD2	(M-M)	Mult2			

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

Tomasulo Example Cycle 13

Instruction status:

Instruction		j	k	Issue	Exec Start	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	12	13

	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			

- **All simple operation are end here.**

Tomasulo Example Cycle 14

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	12	13

	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 Start	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	15	
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	12	13

	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 Start	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	15	16
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	6	10	12	13

	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**

Tomasulo Example Cycle 17

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	15	16
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5	17		
ADDD	F6	F8	F2	6	10	12	13

	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					
39	Mult2	Yes	DIVD	M*F4	M(A1)		

Register result status:

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

Tomasulo Example Cycle 55

Instruction status:

Instruction		*j*	*k*	*Issue*	*Exec Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	15	16
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5	17		
ADDD	F6	F8	F2	6	10	12	13

	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 Start	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	15	16
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5	17	56	
ADDD	F6	F8	F2	6	10	12	13

	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 Start	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	15	16
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5	17	56	57
ADDD	F6	F8	F2	6	10	12	13

	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	No					

Register result status:

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

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

Tomasulo's scheme offers **3** 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**

Tomasulo Drawbacks

- Complexity
  - delays of 360/91, MIPS 10000, Alpha 21264, IBM PPC 620 in CA:AQA 2/e, but not in silicon!
- Many associative stores (CDB) at high speed
- Performance limited by Common Data Bus
  - Each CDB must go to multiple functional units ⇒high capacitance, high wiring density
  - Number of functional units that can complete per cycle limited to one!
    - Multiple CDBs ⇒ more FU logic for parallel assoc stores
- Non-precise interrupts!
  - We will address this later

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 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.

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			

	Busy	*Addr*	*Fu*
Load1	No		
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*
0	**80**	*Fu*									

Tomasulo Loop Example

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

- Assume Multiply takes 4 clocks
- Assume first load takes 8 clocks (cache miss), second load takes 1 clock (hit)
- To be clear, will show clocks for SUBI, BNEZ
- Reality: integer instructions ahead

Loop Example Cycle 1

Instruction status:

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

	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		
1	SD	F4	0	R1			
2	LD	F0	0	R1			
2	MULTD	F4	F0	F2			
2	SD	F4	0	R1			

	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		
2	LD	F0	0	R1			
2	MULTD	F4	F0	F2			
2	SD	F4	0	R1			

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

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S1 Vk	S2 Qj	S2 Qk	RS Code
	Add1	No						LD F0 0 R1
	Add2	No						MULTD F4 F0 F2
	Add3	No						SD F4 0 R1
	Mult1	Yes	Multd		R(F2)	Load1		SUBI R1 R1 #8
	Mult2	No						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 “DataFlow” 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		
2	LD	F0	0	R1			
2	MULTD	F4	F0	F2			
2	SD	F4	0	R1			

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

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		
2	LD	F0	0	R1			
2	MULTD	F4	F0	F2			
2	SD	F4	0	R1			

	Busy	Addr	Fu
Load1	Yes	80	
Load2	No		
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	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

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		
2	MULTD	F4	F0	F2			
2	SD	F4	0	R1			

	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	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		
2	SD	F4	0	R1			

	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:

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*	*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	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?
- 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	RS 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?
- Note: Dispatching BNEZ

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*	*S1 Vk*	*S2 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*	*Vj*	*S1 Vk*	*S2 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						

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	S2 Vk	RS Qj	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	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	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?

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	16	
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*	*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*
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	16	
2	LD	F0	0	R1	6	10	11
2	MULTD	F4	F0	F2	7	15	16
2	SD	F4	0	R1	8	17	

	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						

Summary of Tomasulo Algorithm

- 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)
- Lasting Contributions
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- 360/91 descendants are Pentium III; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264

What about Precise Interrupts?

- Tomasulo had:

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

- Need to “fix” the out-of-order completion aspect so that we can find precise breakpoint in instruction stream.

  ➔ Speculation，Reorder buffer ! (later)

Scoreboard vs. Tomasulo

- 特点
  - Multiple multiplier, etc. Funcs
  - Issue in order, Complete OOO
  - IF→ Issue, Ro
  - 4 stages pipeline
  - Scoreboare centralized control
- 缺点
  - Stall when WAW, WAR

- 特点
  - Fewer Func, unpipelined
  - Issue in order, Complete OOO
  - FP op. queue, Reservation station, LD/ST buffer, CDB
  - 3 stages pipeline
  - Reg. Rename→No WAW, WAR
  - Reduce structural hazard
  - RAW detection decentralized—reservation
  - CDB→ forwarding path

•Can Scoreboard avoid WAW, WAR with Reg. Rename?

Scoreboard Pipeline stage description

- **Issue:** a instruction is issued when
  - *The functional unit is available and*
  - *No other active instruction has the* **same** *destination register.*
  - Avoid **strutural** hazard and **WAW** hazard
- **Read Operands** (RD)
  - *The read operation is delayed until both the operands are available.*
  - *This means that no previously issued but ncompleted instruction has the operand as its destination.*
  - This resolves **RAW** hazards **dynamically**
- **Execution (EX)**
  - *Notify the scoreboard when completed so the functional unit can be reused.*
- **Write result (WB)**
  - *The scoreboard checks for **WAR** hazards and stalls the completing instruction if necessary.*

The scoreboard algorithm

- Scoreboard-takes full responsibility for instruction issue and execution
  - Create the dependence records
  - Decide when to fetch the operand
  - Decide when to enter execution
  - Decide when the result can be written into the register file
- Three data structure
  - Instruction status:
    - which of the four steps the instruction is in
  - Functional unit status: buzy,op,Fi, Fj,Fk,Qj,Qk ,Rj,Rk
  - Register result status:
    - which functional unit will write that register

Explicit Register Renaming

- Make use of a *physical* register file that is larger than number of registers specified by ISA
- Key insight: Allocate a new physical destination register for every instruction that writes
  - Very similar to a compiler transformation called Static Single Assignment (SSA) form — but in hardware!
  - Removes all chance of WAR or WAW hazards
  - Like Tomasulo, good for allowing full out-of-order completion
  - Like hardware-based dynamic compilation?
- Mechanism? Keep a translation table:
  - ISA register ⇒ physical register mapping
  - When register written, replace entry with new register from freelist.
  - Physical register becomes free when not used by any active instructions

Advantages of Explicit Renaming

- Decouples *renaming* from *scheduling:*
  - Pipeline can be exactly like “standard” MIPS pipeline (perhaps with multiple operations issued per cycle)
  - Or, pipeline could be Tomasulo-like or a scoreboard, etc.
  - Standard forwarding or bypassing could be used
- Allows data to be fetched from single register file
  - No need to bypass values from reservation station or reorder buffer
  - This can be important for balancing pipeline

Adv. Explicit Ren…

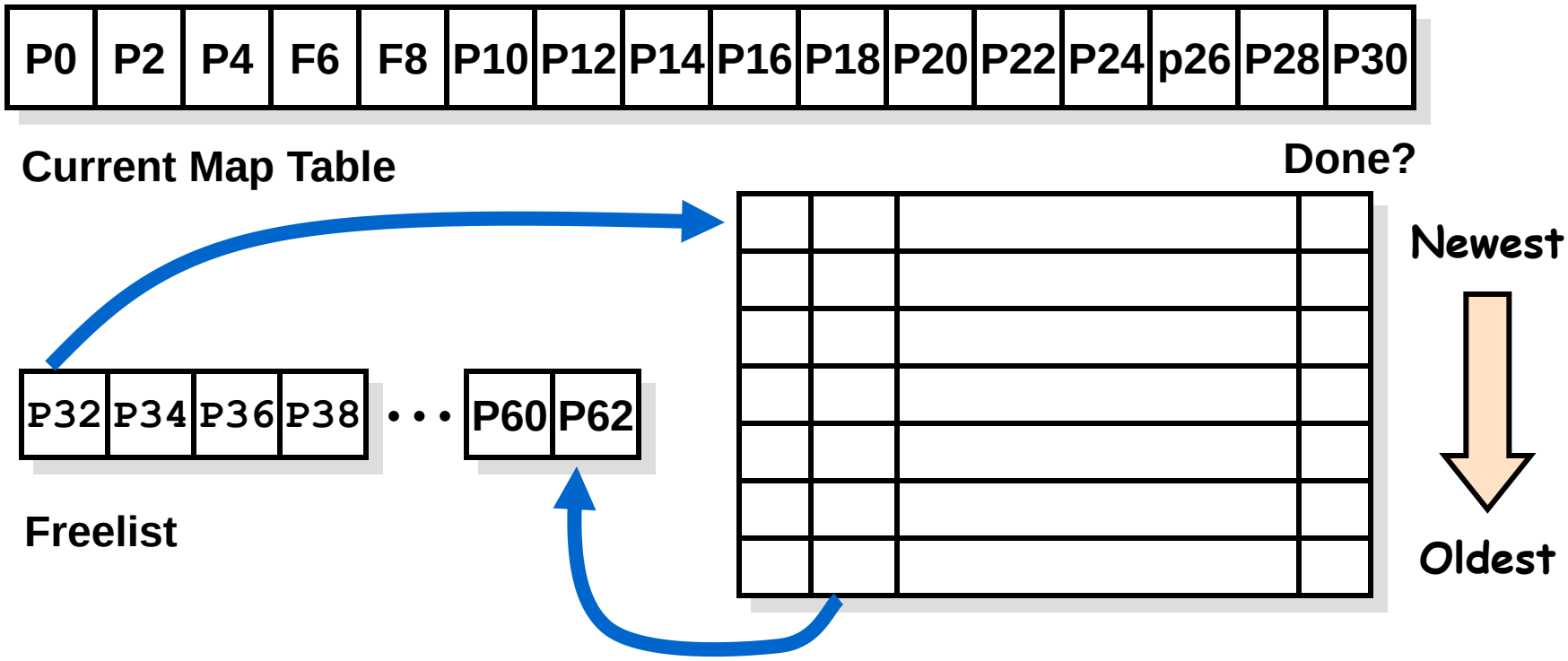
- Many processors use a variant of this technique:
  - R10000, Alpha 21264, HP PA8000

- Another way to get precise interrupt points:
  - All that needs to be "undone" for precise break point is to undo the table mappings
  - Provides an interesting mix between reorder buffer and future file
    - Results are written immediately back to register file
    - Registers *names* are "freed" in program order (by ROB)

Explicit Renaming Support Includes:

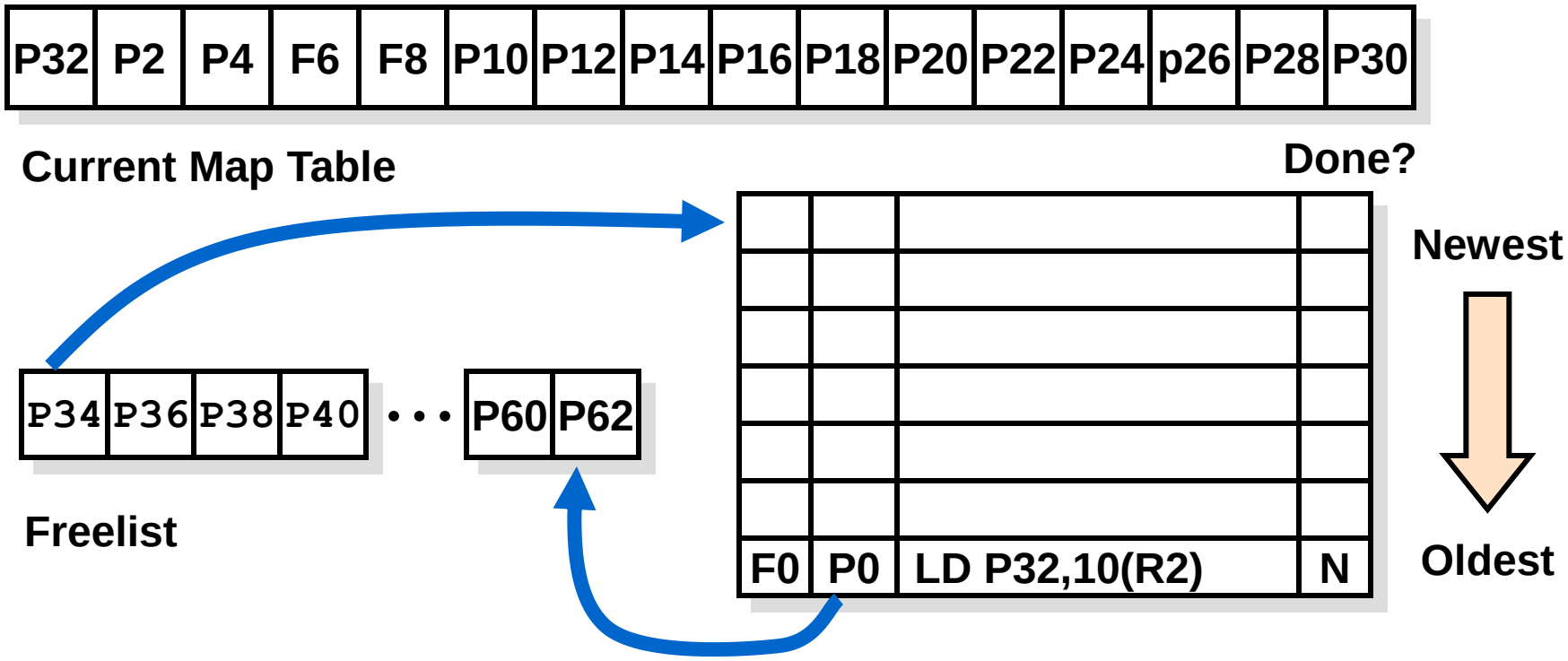
- Rapid access to a table of translations
- A physical register file that has more registers than specified by the ISA
- Ability to figure out which physical registers are free.
  - No free registers ⇒ stall on issue
- Thus, register renaming doesn't require reservation stations. However:
  - Many modern architectures use explicit register renaming + Tomasulo-like reservation stations to control execution.
- Two Questions:
  - How do we manage the "free list"?
  - How does Explicit Register Renaming mix with Precise Interupts?

Explicit register renaming: (R1000 Style)



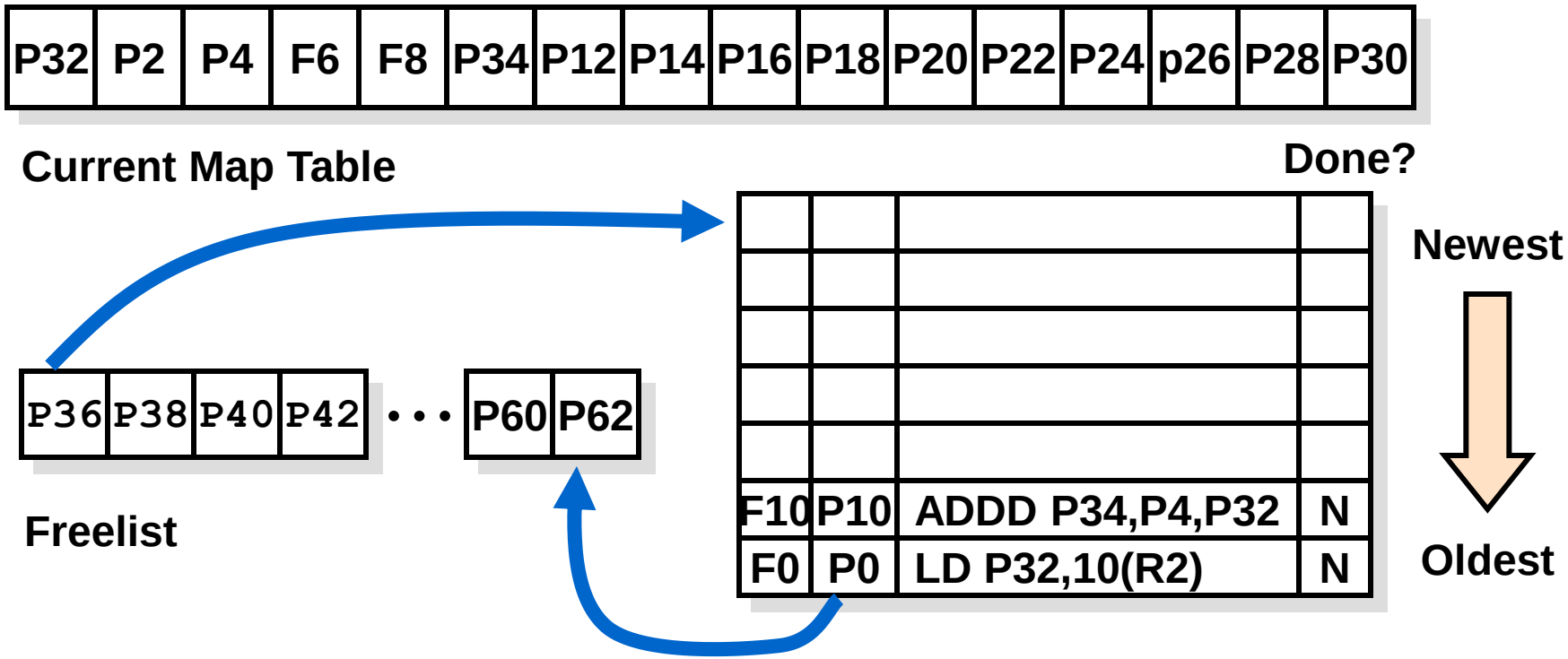
- Physical register file larger than ISA register file
- On issue, each instruction that modifies a register is allocated new physical register from freelist

Explicit register renaming: (R1000 Style)

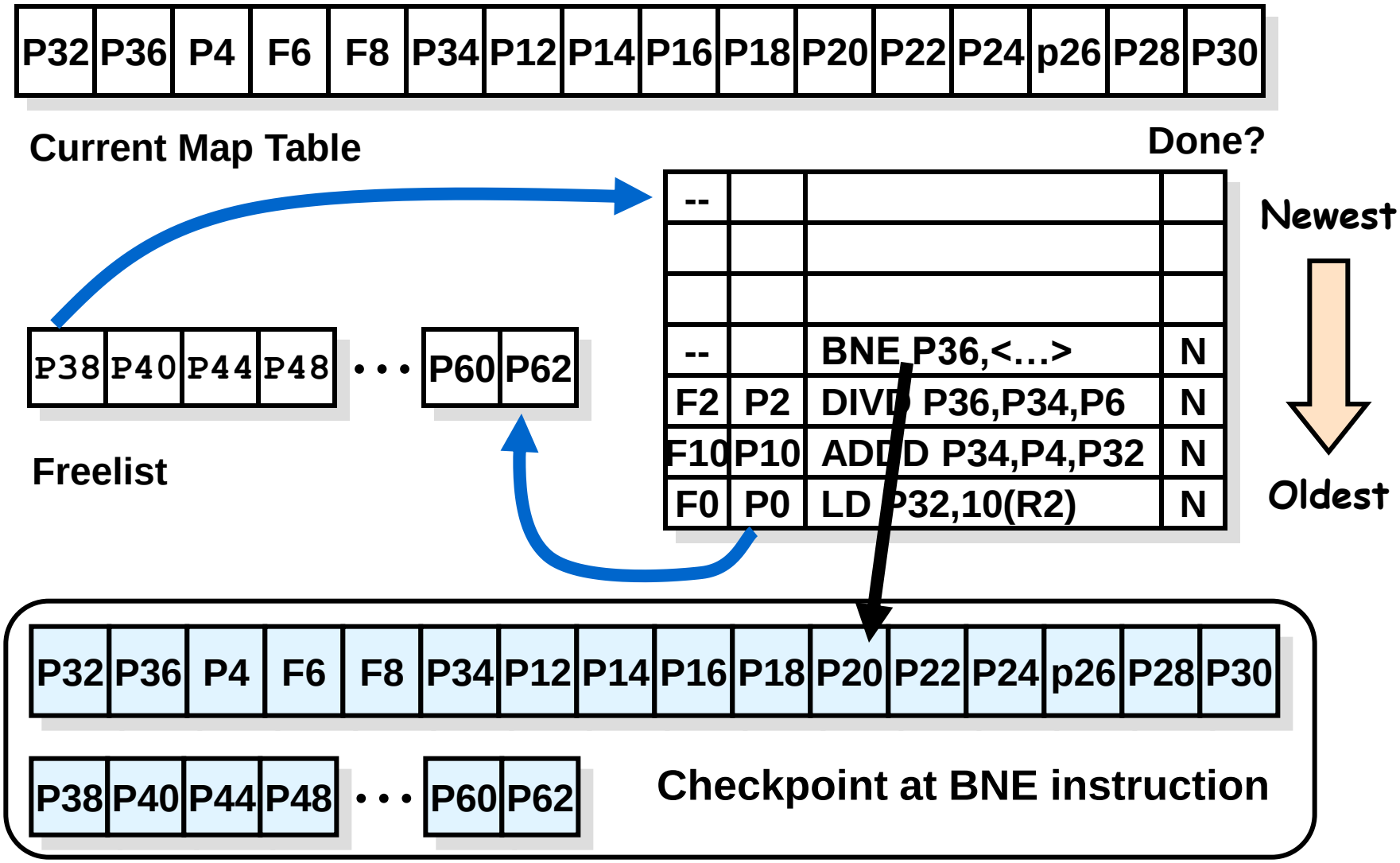


- Note that physical register P0 is “dead” (or not “live”) past the point of this load.
  - **When we go to commit the load, we free up**

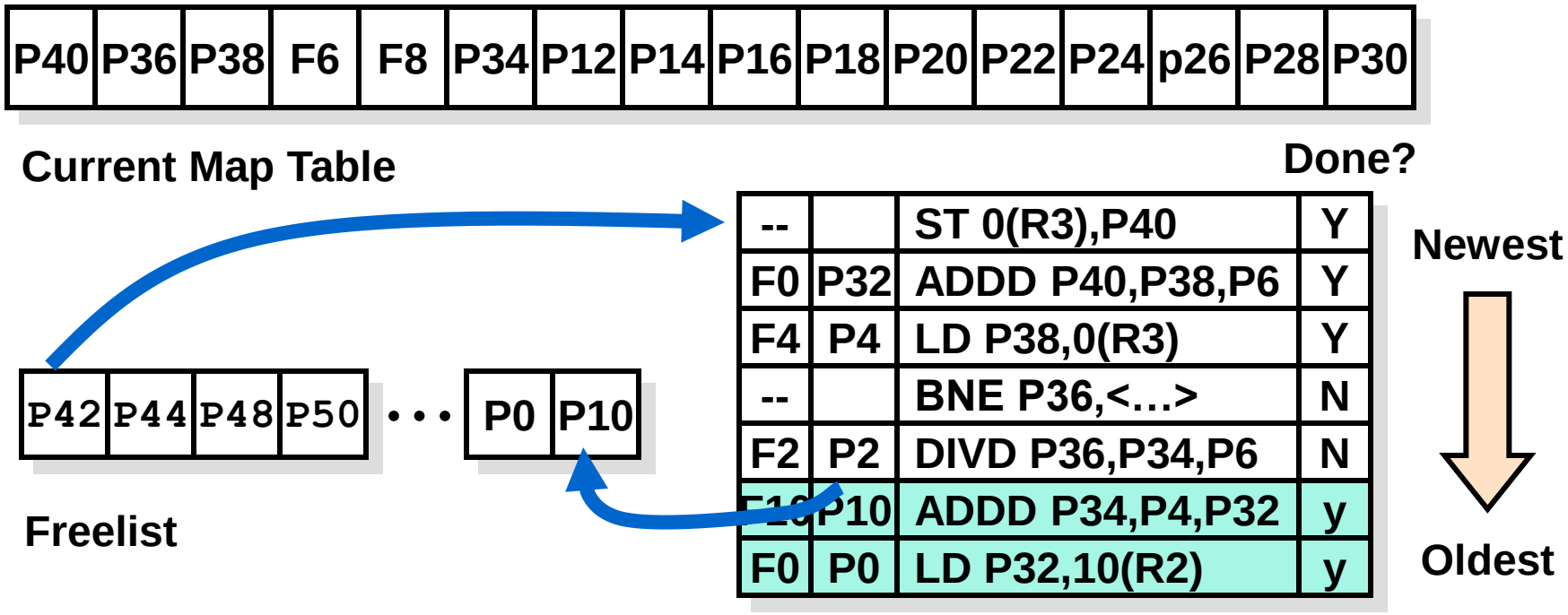
Explicit register renaming:
(R1000 Style)

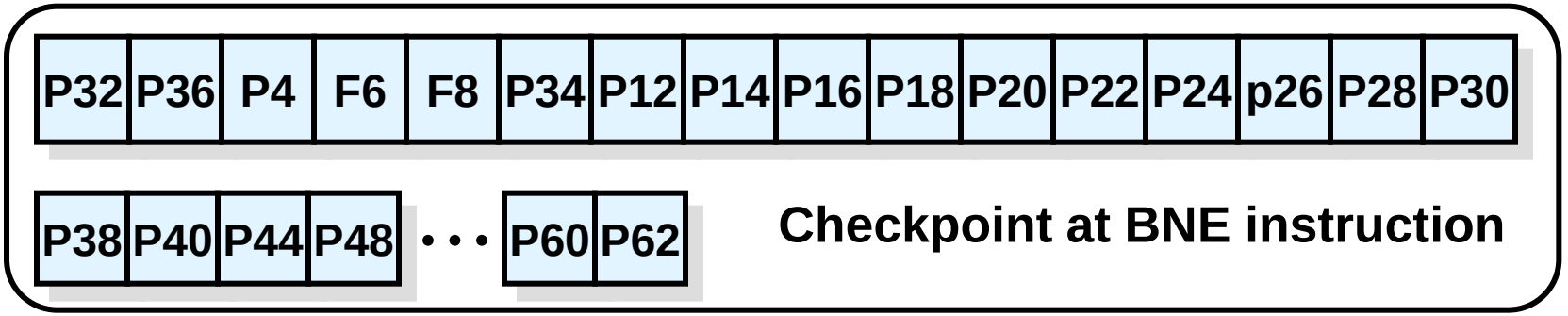


Explicit register renaming:
(R1000 Style)

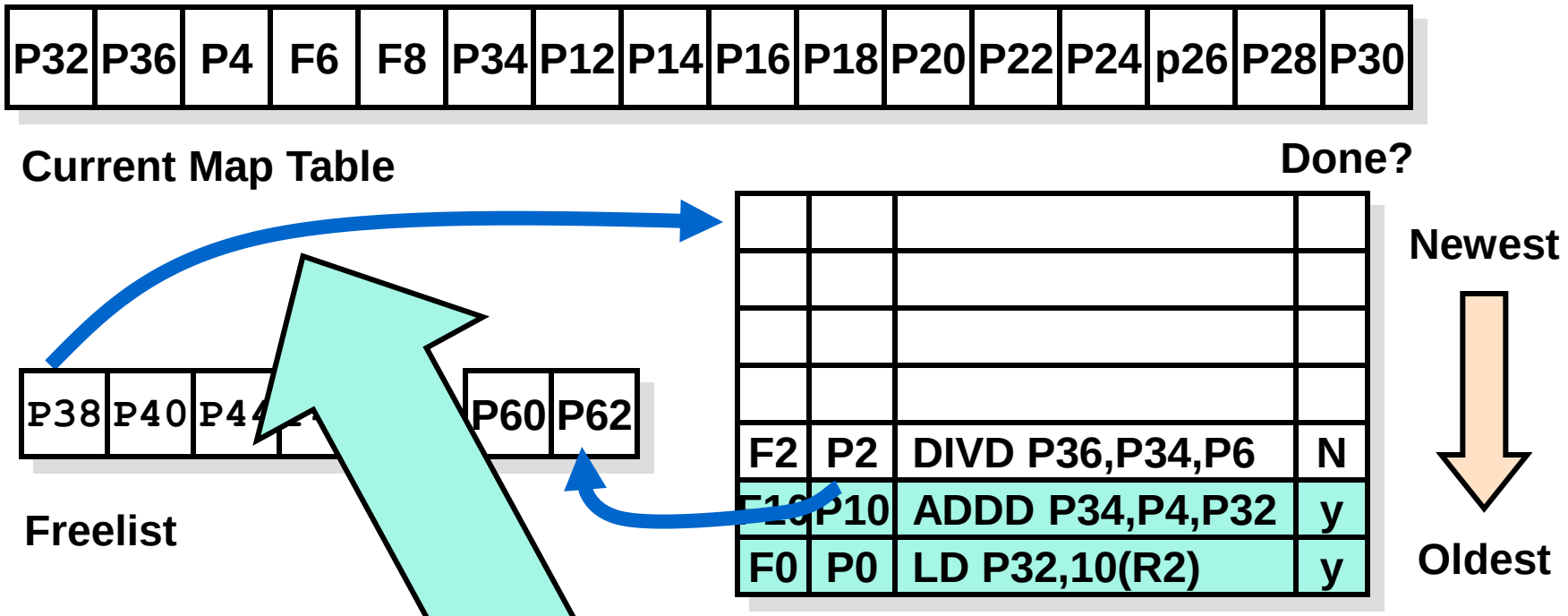


Explicit register renaming:
(R1000 Style)

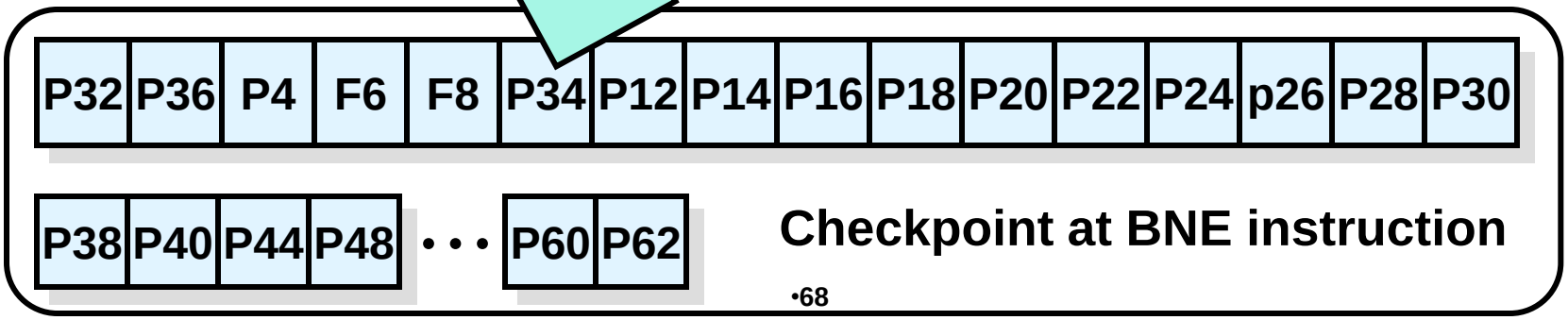




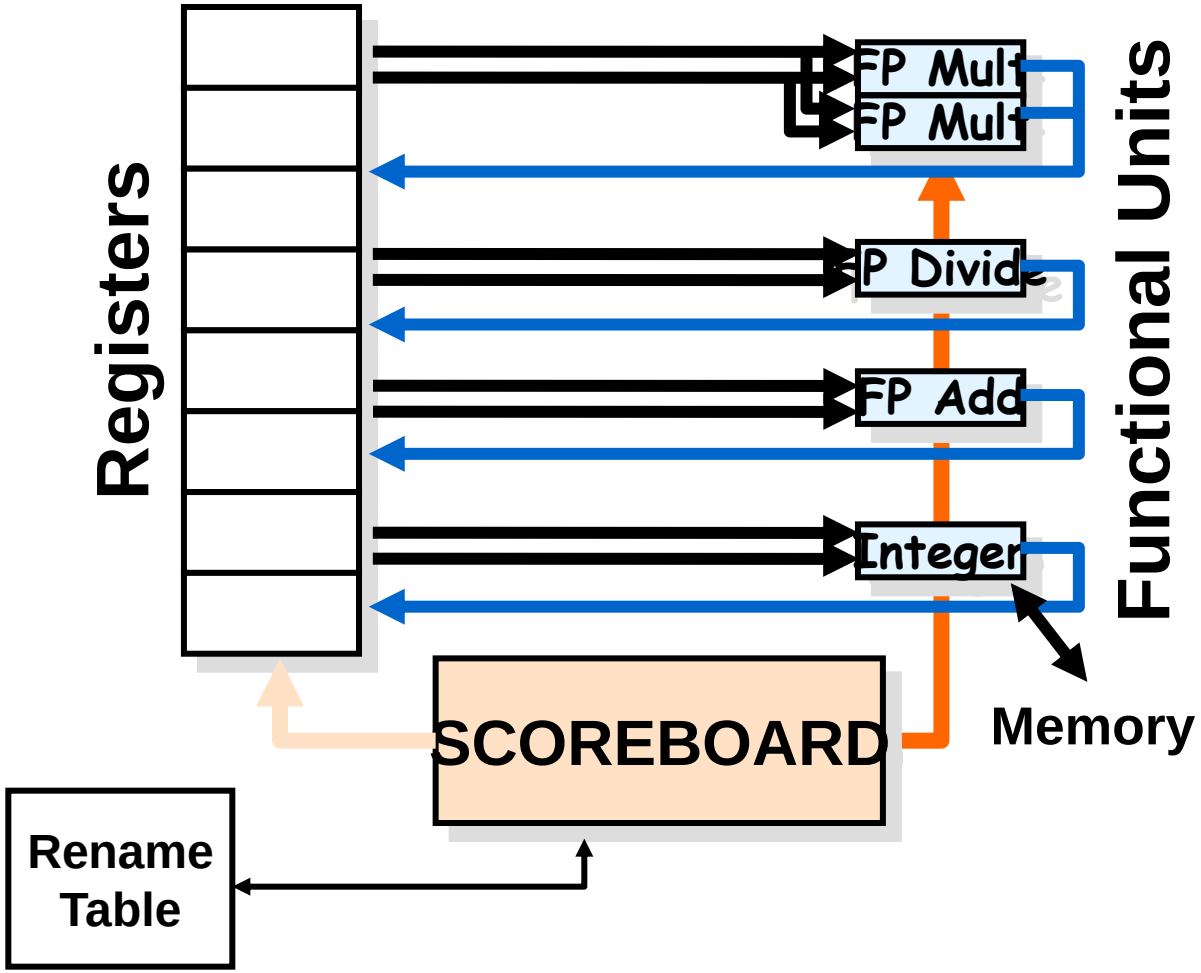
Explicit register renaming:
(R1000 Style)



Speculation error fixed by restoring map table and freelist



Can we use explicit register renaming with scoreboard?



Four Stages of Scoreboard Control With Explicit Renaming

- Issue—decode instructions & check for structural hazards & allocate new physical register for result
  - Instructions issued in program order (for hazard checking)
  - Don't issue if no free physical registers
  - Don't issue if structural hazard
- Read operands—wait until no hazards, read operands
  - All real dependencies (RAW hazards) resolved in this stage, since we wait for instructions to write back data.
- Execution—operate on operands
  - The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard
- Write result—finish execution
- Note: No checks for WAR or WAW hazards!

Scoreboard With Explicit Renaming

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*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				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	No								
	Add	No								
	Divide	No								

Register Rename and Result

Clock	*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
FU	P0	P2	P4	P6	P8	P10	P12		P30

- **Initialized Rename Table**

- **Each instruction allocates free register**
- **Similar to single-assignment compiler transformation**

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	Exec Start	*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					

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	Yes	Load	P32		R2				Yes
	Int2	No								
	Mult1	No								
	Add	No								
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
1	*FU*	P0	P2	P4	P32	P8	P10	P12		P30

Renamed Scoreboard 2

Instruction status:

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

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	Yes	Load	P32		R2				No
	Int2	Yes	Load	P34		R3				Yes
	Mult1	No								
	Add	No								
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
2	*FU*	P0	P34	P4	P32	P8	P10	P12		P30

Renamed Scoreboard 3

Instruction status:

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

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	Yes	Load	P32		R2				No
	Int2	Yes	Load	P34		R3				No
	Mult1	Yes	Multd	P36	P34	P4	Int2		No	Yes
	Add	No								
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
3	*FU*	P36	P34	P4	P32	P8	P10	P12		P30

- **Next step Int1 will write result, where need the value?**

Instruction status:

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

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	Yes	Load	P32		R2				No
	Int2	Yes	Load	P34		R3				No
	Mult1	Yes	Multd	P36	P34	P4	Int2		No	Yes
	Add	Yes	Sub	P38	P32	P34	Int1	Int2	No	No
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
4	*FU*	P36	P34	P4	P32	P38	P10	P12		P30

- **Next step Int2 will write result, where need the value?**

Instruction status:

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

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	Yes	Load	P34		R3				No
	Mult1	Yes	Multd	P36	P34	P4	Int2		No	Yes
	Add	Yes	Sub	P38	P32v	P34		Int2	Yes	No
	Divide	Yes	Divd	P40	P36	P32v	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
5	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

- **Why ADDD not issue ? Structure hazard ! Adder is occupied by with SUBD.**

Instruction status:

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

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	Yes	Multd	P36	P34v	P4			Yes	Yes
	Add	Yes	Sub	P38	P32v	P34v			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32v	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
6	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

Renamed Scoreboard 7

Instruction status:

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

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	Yes	Multd	P36	P34v	P4			No	No
	Add	Yes	Sub	P38	P32v	P34v			No	No
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
7	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

• Next step Adder will complete execution

Renamed Scoreboard 8

Instruction status:

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

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
9	Mult1	Yes	Multd	P36	P34v	P4			No	No
1	Add	Yes	Sub	P38	P32v	P34v			No	No
	Divide	Yes	Divd	P40	P36	P32v	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
8	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

Renamed Scoreboard 9

- **Next step Adder will write result, where need the value?**

				2	3	4	5	6
LD	F2	45+	R3					
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2					

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
8	Mult1	Yes	Multd	P36	P34v	P4			No	No
0	Add	Yes	Sub	P38	P32v	P34v			No	No
	Divide	Yes	Divd	P40	P36	P32v	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
9	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

Adder is cleared, so ADDD can be issued next cycle.

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*		*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2					

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
7	Mult1	Yes	Multd	P36	P34v	P4			No	No
	Add	No								
	Divide	Yes	Divd	P40	P36	P32v	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
10	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

Notice that P32 not listed in Rename Table – Still live. Must not be reallocated by accident

Instruction status:

Instruction		j	k	Issue	Read Oper		Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11				

WAR dependence

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Int1	No								
	Int2	No								
6	Mult1	Yes	Multd	P36	P34	P4			No	No
	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

WAR Hazard gone!

Register Renaming and Result

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
11	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 12

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	Exec start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11	12			

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
5	Mult1	Yes	Multd	P36	P34	P4			No	No
2	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
12	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 13

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	Exec start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11	12	13		

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
4	Mult1	Yes	Multd	P36	P34	P4			No	No
1	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
13	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 14

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	Exec start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11	12	13	14	

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
3	Mult1	Yes	Multd	P36	P34	P4			No	No
0	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
14	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 15

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	Exec start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11	12	13	14	15

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
2	Mult1	Yes	Multd	P36	P34	P4			No	No
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
15	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 16

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	Exec start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8		
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11	12	13	14	15

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
1	Mult1	Yes	Multd	P36	P34	P4			No	No
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
16	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 17

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8	17	
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11	12	13	14	15

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
0	Mult1	Yes	Multd	P36	P34	P4			No	No
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	*...*	*F30*
17	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 18

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Start*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8	17	18
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5				
ADDD	F6	F8	F2	11	12	13	14	15

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	No								
	Add	No								
	Divide	Yes	Divd	P40	P36v	P32	~~Mult1~~		**Yes**	**Yes**

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
18	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 19

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	Exec start	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4	5
LD	F2	45+	R3	2	3	4	5	6
MULTD	F0	F2	F4	3	7	8	17	18
SUBD	F8	F6	F2	4	7	8	9	10
DIVD	F10	F0	F6	5	19			
ADDD	F6	F8	F2	11	12	13	14	15

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	No								
	Add	No								
40	Divide	Yes	Divd	P40	P36	P32	Mult1		**NO**	**NO**

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
19	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Summary #2

- Explicit Renaming: more physical registers than needed by ISA.
  - Separates *renaming* from *scheduling*
    - Opens up lots of options for resolving RAW hazards
  - Rename table: tracks current association between architectural registers and physical registers
  - Potentially complicated rename table management