Tomasulo's Algorithm

- Another dynamic scheduling technique
- Overcomes problems with scoreboards
  - Renaming of registers
  - Avoids WAW and WAR hazards
- Introduced with the IBM 360/91 for its floating-point unit
  - Desire to achieve high-performance for entire 360 family without recompiling code specifically for FP
  - To overcome long memory and floating-point delays
- Similar ideas used in the Alpha 21264, HP8000, MIPS 10000, Pentium II, Power PC, etc.

Tomasulo vs. Scoreboard

1. Control and buffers distributed with FUs vs. centralized in scoreboard (buffers called "reservation stations")
2. Registers in instructions replaced by pointers to reservation station buffer
3. HW renaming of registers to avoid WAR and WAW hazards
4. Common Data Bus broadcasts results to all FUs
5. Load and Stores treated as FUs
6. Memory disambiguation

Advantages

1. **Broadcast on CDB more efficient** - operands available without register file read
2. **Avoids WAR hazards** by reading the operands in the instruction-issue order, instead of stalling the WB stage. To accomplish this an instruction reads an available operand before waiting for the other.
3. **Avoids WAW hazards** by renaming the registers (using the *id* of a reservation station rather than the register *id*)

Avoiding WAR Hazards

WAR hazard if SUBD completes before ADDD:

```
DIVD    F0,F2,F4
ADDD    F10,F0,F8
SUBD    F8,F8,F14
SUBD    F2,F8,F14
```

Avoiding WAR Hazards

WAR hazard if SUBD completes before ADDD:

```
DIVD    F0,F2,F4
ADDD    F10,F0,F8
SUBD    F8,F8,F14
SUBD    F2,F8,F14
```

Avoiding WAR Hazards

WAR hazard if SUBD completes before ADDD:

```
DIVD    F0,F2,F4
ADDD    F10,F0,F8
SUBD    F8,F8,F14
SUBD    F2,F8,F14
```

Change name of F8 to t0 (at next set of F8):

```
DIVD    F0,F2,F4
ADDD    F10,F0,F8
SUBD    t0,F8,F14
SUBD    F2,t0,F14
...all further uses of F8 renamed to t0
```

Avoiding WAR Hazards

WAR hazard if SUBD completes before ADDD:

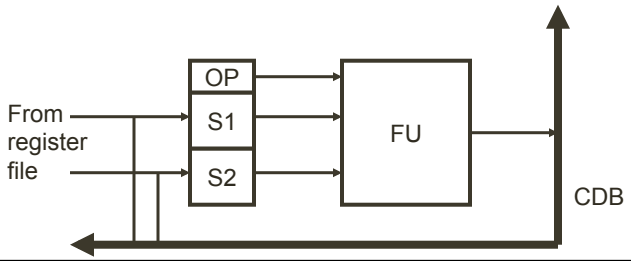
```
DIVD    F0,F2,F4
ADDD    F10,F0,F8
SUBD    F8,F8,F14
SUBD    F2,F8,F14
```

Change name of F8 to t0:

```
DIVD    F0,F2,F4
ADDD    F10,F0,F8
SUBD    t0,F8,F14
SUBD    F2,t0,F14
...all further uses of F8 renamed to t0
```

Reservation Stations

- **"Reservation stations"** associated with every FU.
  - Holds operation and source operands.
  - When sources aren't available yet, instruction can be sent to reservation station. When operands become available, the reservation station receives their values.
  - Avoids WAR and WAW hazards by renaming the registers to be ***pointers to reservation stations***.
  - Dynamically captures data flow graph



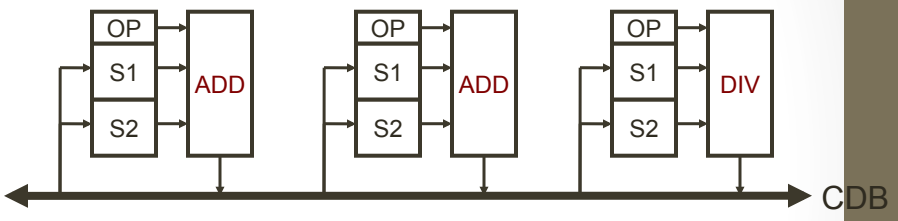
Reservation Stations

- The dynamic data flow graph is essentially built using the reservation stations.
- Example

```
DIVD      F4,F7,F8
SUBD      F0,F1,F2
ADDD      F3,F0,F4
```

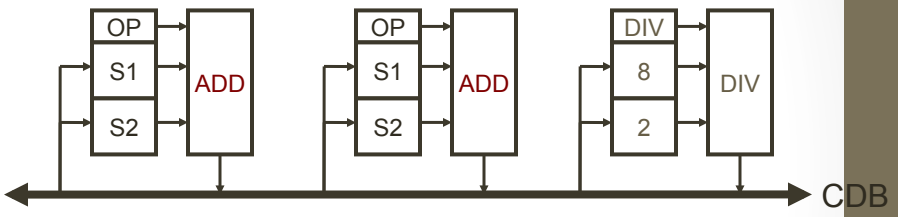
2 FP adders and 1 FP divider

Data Flow Example

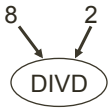



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

Data Flow Example

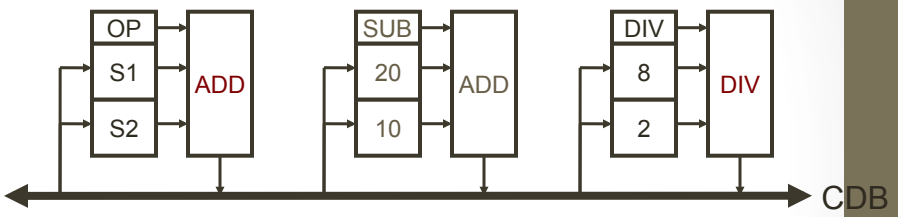



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

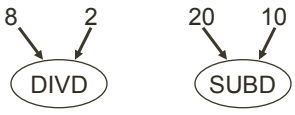


DIVD issues: Rename F4, Read F7 & F8

Data Flow Example

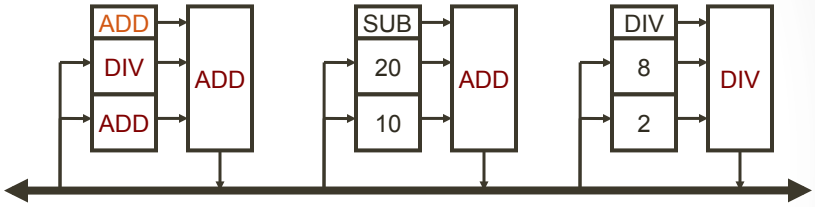



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

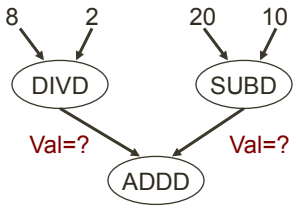


SUBD issues: Rename F0, Read F1 and F2

Data Flow Example

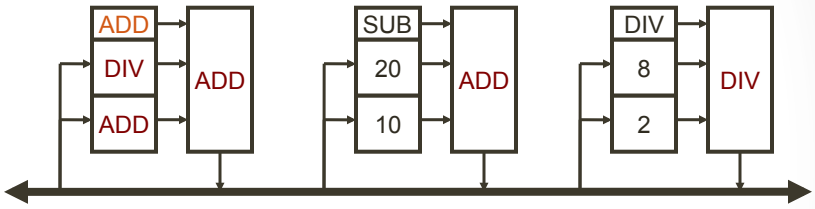



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

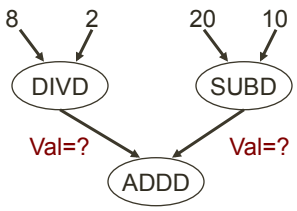


ADDD issues: Input values not available

Data Flow Example

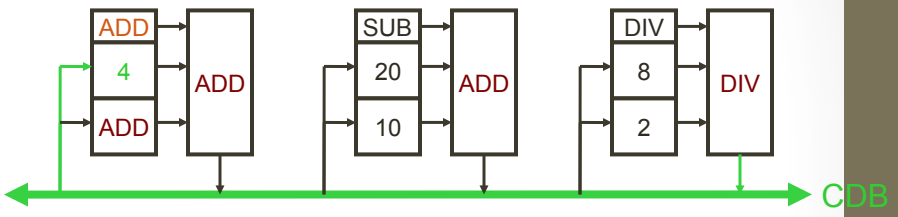



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

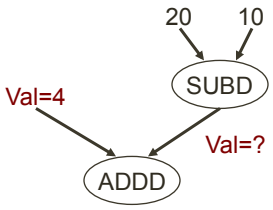


DIVD finishes

Data Flow Example

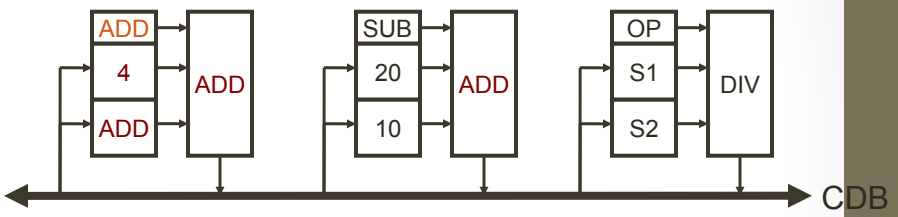



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

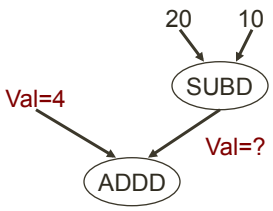


DIVD writes result to CDB, ADDD sees value

Data Flow Example

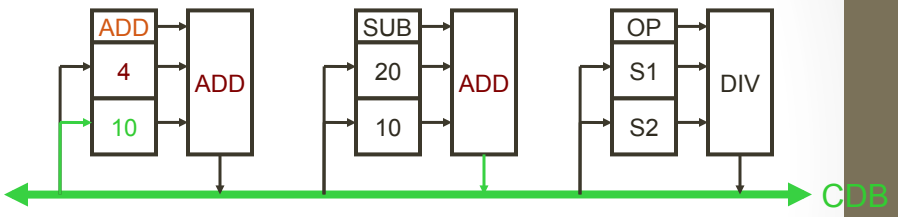



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

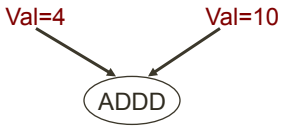


SUBD finishes

Data Flow Example

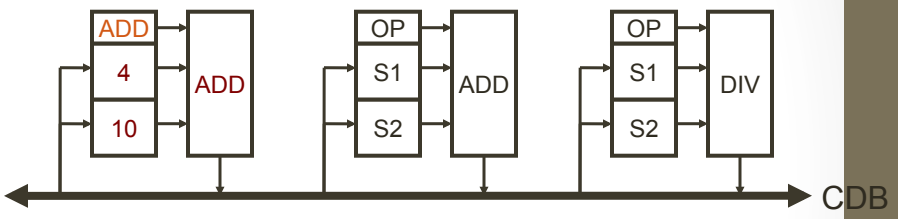



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

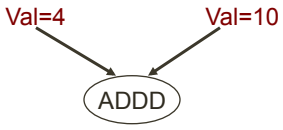


SUBD writes result to CDB, ADDD see results

Data Flow Example

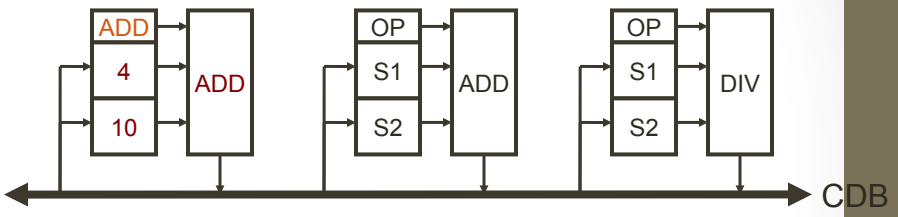



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

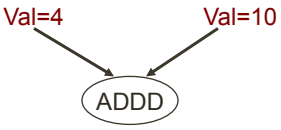


ADDD executes

Data Flow Example

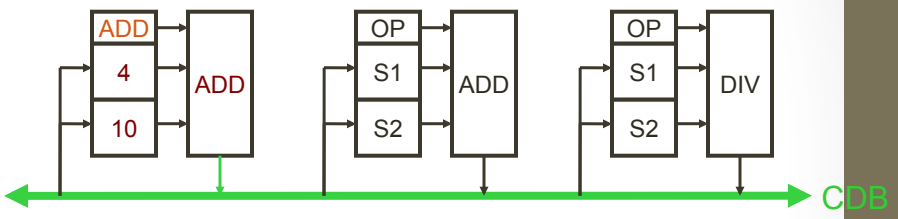



```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```



ADDD finishes

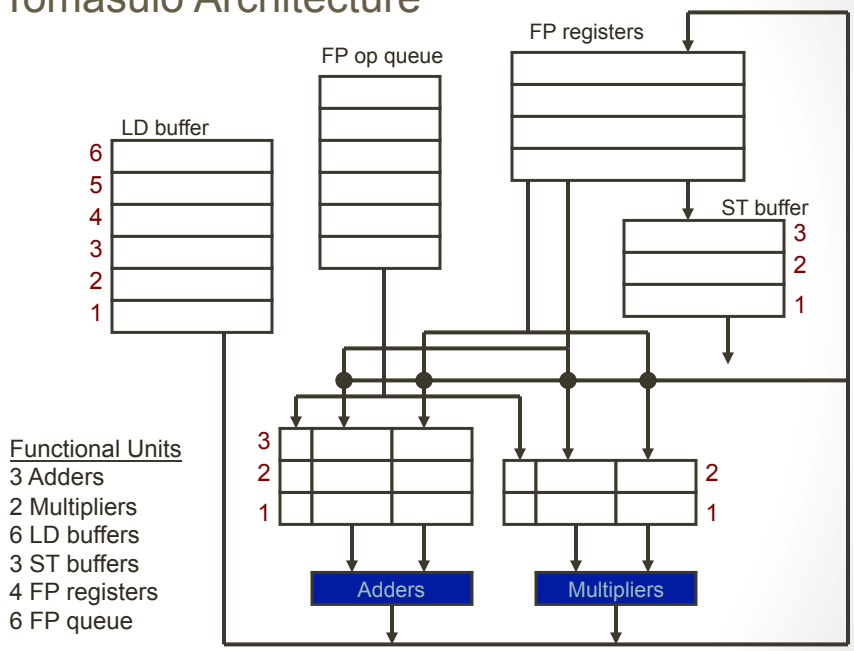
Data Flow Example




```
DIVD   F4,F7,F8
SUBD   F0,F1,F2
ADDD   F3,F0,F4

F1=20  F2=10
F7=8   F8=2
```

ADDD writes result to CDB

Tomasulo Architecture



Tomasulo Data Structures

- Reservation Station Components

Field	Description
Op	Which operation to perform (e.g., +, -)
Qj, Qk	Reservation stations producing source registers
Vj,Vk	Value of source registers
Rj, Rk	Flags indicating when Vj,Vk are available
Busy	Indicates reservation station is occupied

- Register Result Status
  - Entry per register
  - Indicates which Res Station will write a particular register; blank if no Res Station will write the register

Stages of Control

Issue (I) - Get instruction from FP op queue

- If reservation station free, issue instruction and send available operands. Rename registers.

Execute (EX) - Operate on operands

- When both operands ready, execute. If both aren't ready, watch CDB for corresponding operand.

Write result (WR) - Finish execution

- Write on CDB to all waiting FUs and register file, mark reservation station available.

Reservation Station Example

Code Sequence

```
LD     F6, 34(R2)
LD     F2, 45(R3)
MULT   F0, F2, F4
SUBD   F8, F6, F2
DIVD   F10, F0, F6
ADDD   F6, F8, F2
```

Functional Units

3 FP adders, 2 FP multipliers, 3 load buffers

Pipeline Latencies

LD	1 cycle
MULT	10 cycles
SUBD, ADDD	2 cycles
DIVD	40 cycles

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2			
LD	F2	45+	R3			
MULT	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
L1	No	
L2	No	
L3	No	

Reservation Stations

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

Register result status

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

CLOCK 0

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULT	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
L1	Yes	34+R2
L2	No	
L3	No	

Reservation Stations

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				L1					

CLOCK 1

LD issues, available load buffer 1 occupied

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1		
LD	F2	45+	R3	2		
MULT	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
L1	Yes	34+R2
L2	Yes	45+R3
L3	No	

Reservation Stations

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU		L2		L1					

CLOCK 2

LD2 issues, occupies load buffer 2

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
L1	Yes	34+R2
L2	Yes	45+R3
L3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	L2	
	Mult2	No					

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	L2		L1					

CLOCK 3

LD1 finishes, MULT issues

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2		
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

	Busy	Address
L1	No	
L2	Yes	45+R3
L3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	Yes	SUBD	MA			L2
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		R(F4)	L2	
	Mult2	No					

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	L2		MA	A1				

CLOCK 4

MA is MEM[R2+34]
LD1 writes result to CDB, SUBD issues and gets operand1

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2			

	Busy	Address
L1	No	
L2	Yes	45+R3
L3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
0	Add1	Yes	SUBD	MA			L2
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD		R(F4)	L2	
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	L2		MA	A1	M2			

CLOCK 5

MA is MEM[R2+34]
LD2 finishes, SUBD blocked for LD2, DIVD issues

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
2	Add1	Yes	SUBD	MA	MB		
0	Add2	Yes	ADDD		MB	A1	
	Add3	No					
10	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		A2	A1	M2			

CLOCK 6

MA is MEM[R2+34], MB is MEM[R3+45]
LD2 writes result to CDB, SUBD acquires operand 2,
ADDD issues and blocks on SUBD
ADDD and DIVD have WAR hazard - renamed to RS

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
1	Add1	Yes	SUBD	MA	MB		
0	Add2	Yes	ADDD		MB	A1	
	Add3	No					
9	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		A2	A1	M2			

CLOCK 7

MA is MEM[R2+34], MB is MEM[R3+45]
Execution cycle

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
0	Add1	Yes	SUBD	MA	MB		
0	Add2	Yes	ADDD		MB	A1	
	Add3	No					
8	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		A2	A1	M2			

CLOCK 8

MA is MEM[R2+34], MB is MEM[R3+45]
SUBD finishes execution

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
0	Add2	Yes	ADDD	MA-MB	MB		
	Add3	No					
7	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		A2	MC	M2			

CLOCK 9

MA is MEM[R2+34], MB is MEM[R3+45], MC=MA-MB
SUBD writes result to CDB, ADDD gets both operands

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
2	Add2	Yes	ADDD	MA-MB	MB		
	Add3	No					
6	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		A2	MC	M2			

CLOCK 10

MA is MEM[R2+34], MB is MEM[R3+45], MC=MA-MB
ADDD can begin execution

Instruction status

Op		j	k	Issue	Execution	Write
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

	Busy	Address
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
1	Add2	Yes	ADDD	MA-MB	MB		
	Add3	No					
5	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		A2	MC	M2			

CLOCK 11

MA is MEM[R2+34], MB is MEM[R3+45], MC=MA-MB
Execution cycle

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
0	Add2	Yes	ADDD	MA-MB	MB		
	Add3	No					
4	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		A2	MC	M2			

CLOCK 12

MA is MEM[R2+34], MB is MEM[R3+45], MC=MA-MB
ADDD finishes execution

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		MD	MC	M2			

CLOCK 13

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8
ADDD writes result to CDB

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		MD	MC	M2			

CLOCK 14

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8
Execution cycle

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3		
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		MD	MC	M2			

CLOCK 15

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8
Execution cycle

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3	16	
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	MB	R(F4)		
0	Mult2	Yes	DIVD		MA	M1	

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	M1	MB		MD	MC	M2			

CLOCK 16

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8
MULTD finishes execution

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	ME	MA		

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	ME	MB		MD	MC	M2			

CLOCK 17

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8,
ME=MB*F4
MULTD writes result to CDB

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	DIVD	ME	MA		

Register result status

	F0	F2	F4	F6	F8	F10	F12 ...	F30
FU	ME	MB		MD	MC	M2		

CLOCK 18

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8, ME=MB*F4
DIVD has operands and can begin executing

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
1	Mult2	Yes	DIVD	ME	MA		

Register result status

	F0	F2	F4	F6	F8	F10	F12 ...	F30
FU	ME	MB		MD	MC	M2		

CLOCK 57

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8, ME=MB*F4
DIVD executing

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5	58	
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	DIVD	ME	MA		

Register result status

	F0	F2	F4	F6	F8	F10	F12 ...	F30
FU	ME	MB		MD	MC	M2		

CLOCK 58

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8, ME=MB*F4
DIVD finishes execution

Instruction status

Op		j	k	*Issue*	*Execution*	*Write*
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	5	6
MULT	F0	F2	F4	3	16	17
SUBD	F8	F6	F2	4	8	9
DIVD	F10	F0	F6	5	58	
ADDD	F6	F8	F2	6	12	13

	Busy	*Address*
L1	No	
L2	No	
L3	No	

Reservation Stations

Time	*Name*	*Busy*	*Op*	*S1 Vj*	*S2 Vk*	*RS for j Qj*	*RS for k Qk*
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	No					

Register result status

	F0	F2	F4	F6	F8	F10	F12 ...	F30
FU	ME	MB		MD	MC	MF		

CLOCK 59

MA=MEM[R2+34], MB=MEM[R3+45], MC=MA-MB, MD=F6-F8, ME=MB*F4, MF=ME/MA
DIVD writes result to CDB

Memory Disambiguation

- Store buffer holds target addresses
- When issuing a load, check:

  Addresses don't match - let load progress and it can complete before store

  Addresses match and value available - read from store buffer (or register file)

  Addresses match and value unavailable - wait until store buffer slot receives value (but we can issue and mark the operand as coming from the store buffer slot)

Loops

- Predict branches taken - multiple iterations of the loop are active at once
- Loop is effectively unrolled by the hardware

- For 360, only 4 FP registers, so unrolling by the compiler has limited benefits due to name dependencies.
- But with reservation stations, we effectively have more registers than what's "architected" and the hardware can rename as needed.

Loop Example

- Code sequence

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

- Assumptions
  - Multiply takes 4 cycles
  - Cache misses
  - Predict branches taken

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1			
MULTD	F4	F0	F2	1			
SD	F4	0	R1	1			
LD	F0	0+	R1	2			
MULTD	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	No		
L2	No		
L3	No		
S1	No		
S2	No		
S3	No		

Reservation Stations

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

Code:

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=80	FU								

CLOCK 0

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1			
SD	F4	0	R1	1			
LD	F0	0+	R1	2			
MULTD	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	Yes	80	
L2	No		
L3	No		
S1	No		
S2	No		
S3	No		

Reservation Stations

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

Code:

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=80 FU	L1								

CLOCK 1

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1			
LD	F0	0+	R1	2			
MULTD	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	Yes	80	
L2	No		
L3	No		
S1	No		
S2	No		
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MUL		F2	L1	
	Mult2	No					

Code:

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=80 FU	L1		M1						

CLOCK 2

MULTD from iter 1 issues

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2			
MULTD	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	Yes	80	
L2	No		
L3	No		
S1	Yes	80	M1
S2	No		
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MUL		F2	L1	
	Mult2	No					

Code:

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

Register result status

		F0	F2	F4	F6	F8	F10	F12	...	F30
R1=80	FU	L1		M1						

CLOCK 3

SD from iter 1issues, occupies store buffer slot

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2			
MULTD	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	Yes	80	
L2	No		
L3	No		
S1	Yes	80	M1
S2	No		
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MUL		F2	L1	
	Mult2	No					

Code:

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

Register result status

		F0	F2	F4	F6	F8	F10	F12	...	F30
R1=72	FU	L1		M1						

CLOCK 4

SUBI decrements R1 (value=72)

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2			
MULTD	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	Yes	80	
L2	No		
L3	No		
S1	Yes	80	M1
S2	No		
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MUL		F2	L1	
	Mult2	No					

Code:

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=72 FU	L1		M1						

CLOCK 5

BNEZ predicted not taken

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6		
MULTD	F4	F0	F2	2			
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	Yes	80	
L2	Yes	72	
L3	No		
S1	Yes	80	M1
S2	No		
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MUL		F2	L1	
	Mult2	No					

Code:

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=72 FU	L2		M1						

CLOCK 6

LD from iter 2 issues and occupies load buffer slot 2

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6		
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2			

	Busy	Address	Qi
L1	Yes	80	
L2	Yes	72	
L3	No		
S1	Yes	80	M1
S2	No		
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L1	
	Mult2	Yes	MULTD		F2	L2	

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

Register result status

		F0	F2	F4	F6	F8	F10	F12	...	F30
R1=72	FU	L2		M1						

CLOCK 7

MULTD from iter 2 issues

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1		
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6		
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	Yes	80	
L2	Yes	72	
L3	No		
S1	Yes	80	M1
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L1	
	Mult2	Yes	MULTD		F2	L2	

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

Register result status

		F0	F2	F4	F6	F8	F10	F12	...	F30
R1=72	FU	L2		M2						

CLOCK 8

SD from iter 2 issues

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6		
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	Yes	80	
L2	Yes	72	
L3	No		
S1	Yes	80	M1
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L1	
	Mult2	Yes	MULTD		F2	L2	

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=64 FU	L2		M2						

CLOCK 9

SUBI decrements R1
LD from iter 1 finishes

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	Yes	72	
L3	No		
S1	Yes	80	M1
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	MULTD	MA	F2		
0	Mult2	Yes	MULTD		F2	L2	

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=64 FU	L2		M2						

CLOCK 10

MA=MEM[80]
BNEZ predicted taken, LD from iter 1 writes result to CDB, M1 receives operand 1, LD from iter 2 finishes

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	M1
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULTD	MA	F2		
4	Mult2	Yes	MULTD	MB	F2		

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=64 FU			M2						

CLOCK 11

MA=MEM[80],MB=MEM[72]
LD from iter 2 finishes, M2 receives operands, ST from iter 3 issues and occupies a load buffer

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	M1
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
2	Mult1	Yes	MULTD	MA	F2		
3	Mult2	Yes	MULTD	MB	F2		

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

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=64 FU			M2						

CLOCK 12

MA=MEM[80],MB=MEM[72]
MULTD from iters 1 and 2 execute

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2		
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	M1
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
1	Mult1	Yes	MULTD	MA	F2		
2	Mult2	Yes	MULTD	MB	F2		

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

Register result status

	FU	F0	F2	F4	F6	F8	F10	F12 ...	F30
R1=64				M2					

CLOCK 13

MA=MEM[80],MB=MEM[72]
MULTD from iters 1 and 2 execute

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7		
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	M1
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
0	Mult1	Yes	MULTD	MA	F2		
1	Mult2	Yes	MULTD	MB	F2		

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

Register result status

	FU	F0	F2	F4	F6	F8	F10	F12 ...	F30
R1=64				M2					

CLOCK 14

MA=MEM[80],MB=MEM[72]
MULTD from iter 1 finishes

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	15
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7	15	
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	MA*F2
S2	Yes	72	M2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
0	Mult2	Yes	MULTD	MB	F2		

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

Register result status

	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=64				M2						

CLOCK 15

MA=MEM[80],MB=MEM[72]
MULTD from iter 1 writes to CDB, S1 receives operands, MULTD from iter 2 finishes

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	15
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7	15	16
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	MA*F2
S2	Yes	72	MB*F2
S3	No		

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L3	
	Mult2	No					

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

Register result status

	FU	F0	F2	F4	F6	F8	F10	F12	...	F30
R1=64				M1						

CLOCK 16

MA=MEM[80],MB=MEM[72]
MULTD from iter 2 writes result to CDB, S2 receives result, MULTD from iter 3 issues

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	15
SD	F4	0	R1	1	3		
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7	15	16
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	MA*F2
S2	Yes	72	MB*F2
S3	Yes	64	M1

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L3	
	Mult2	No					

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

Register result status

		F0	F2	F4	F6	F8	F10	F12 ...	F30
R1=64	FU			M1					

CLOCK 17

MA=MEM[80],MB=MEM[72]
SD from iter 3 issues

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	15
SD	F4	0	R1	1	3	18	
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7	15	16
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	Yes	80	MA*F2
S2	Yes	72	MB*F2
S3	Yes	64	M1

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L3	
	Mult2	No					

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

Register result status

		F0	F2	F4	F6	F8	F10	F12 ...	F30
R1=56	FU			M1					

CLOCK 18

MA=MEM[80],MB=MEM[72]
SD from iter 1 finishes, SUBI decrements R1

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	15
SD	F4	0	R1	1	3	18	19
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7	15	16
SD	F4	0	R1	2	8		

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	No		
S2	Yes	72	MB*F2
S3	Yes	64	M1

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L3	
	Mult2	No					

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

Register result status

		F0	F2	F4	F6	F8	F10	F12	...	F30
R1=56	FU			M1						

CLOCK 19

MA=MEM[80],MB=MEM[72]
SD from iter 1 writes result to CDB

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	15
SD	F4	0	R1	1	3	18	19
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7	15	16
SD	F4	0	R1	2	8	20	

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	No		
S2	Yes	72	MB*F2
S3	Yes	64	M1

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L3	
	Mult2	No					

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

Register result status

		F0	F2	F4	F6	F8	F10	F12	...	F30
R1=56	FU			M1						

CLOCK 20

MA=MEM[80],MB=MEM[72]
SD from iter 2 finishes

Instruction status

Op		j	k	Itr	I	EX	W
LD	F0	0+	R1	1	1	9	10
MULTD	F4	F0	F2	1	2	14	15
SD	F4	0	R1	1	3	18	19
LD	F0	0+	R1	2	6	10	11
MULTD	F4	F0	F2	2	7	15	16
SD	F4	0	R1	2	8	20	21

	Busy	Address	Qi
L1	No		
L2	No		
L3	Yes	64	
S1	No		
S2	No		
S3	Yes	64	M1

Reservation Stations

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULTD		F2	L3	
	Mult2	No					

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

Register result status

	F0	F2	F4	F6	F8	F10	F12 ...	F30
R1=56 FU			M1					

CLOCK 21

MA=MEM[80],MB=MEM[72]
SD from iter 2 writes result to CDB

Disadvantages

- Good branch prediction to keep the pipeline full
  - Requires accurate branch prediction
- Single CDB - multiple FUs completing at the same time must arbitrate
  - May want multiple busses (expensive)
- Associative memories at reservation stations
  - Difficult to run such memories at high speeds

Tomasulo Summary

- Prevents bottleneck on registers
- Avoids WAR and WAW hazards
- Allows limited loop unrolling in HW
- With branch prediction, you can go beyond a basic block
- Contributions to modern processors
  - Dynamic scheduling
  - Register renaming
  - Load/store buffer memory disambiguation