COMP4611: Design and Analysis of Computer Architectures

Pipelining
LLP, Dynamic Scheduling

Lin Gu

CSE, HKUST

1

Loop-Level Parallelism (LLP) Analysis

- Loop-Level Parallelism (LLP) analysis focuses on whether data accesses in later iterations of a loop are data dependent on data values produced in earlier iterations.

  e.g. in `for (i=1; i<=1000; i++)`
  `x[i] = x[i] + s;`

  The computation in each iteration is independent of the previous iterations and the loop is thus parallel. The use of x[i] twice is within a single iteration.
  ⇒Thus loop iterations are independent from each other
- Loop-carried Dependence: A data dependence between different loop iterations (data produced in earlier iteration used in a later one) – limits parallelism.
- Instruction level parallelism (ILP) analysis, on the other hand, is usually done when instructions are generated by the compiler.

2

LLP Analysis Example 1

- In the loop:

```
for (i=1; i<=100; i=i+1) {
    A[i+1] = A[i] + C[i];  /* S1 */
    B[i+1] = B[i] + A[i+1];} /* S2 */
}
```

(Where A, B, C are distinct non-overlapping arrays)

- **S2 uses the value A[i+1], computed by S1 in the same iteration. This data dependence is within the same iteration (not a loop-carried dependence).**
  ⇒ does not prevent loop iteration from being parallelized.
- **S1 uses a value computed by S1 in an earlier iteration, since iteration i computes A[i+1] read in iteration i+1 This is thus loop-carried dependence, and limits parallelism. The same applies for S2 for B[i] and B[i+1]**
  ⇒These two dependences are loop-carried spanning more than one iteration

3

LLP Analysis Example 2

- In the loop:

```
for (i=1; i<=100; i=i+1) {
    A[i] = A[i] + B[i];      /* S1 */
    B[i+1] = C[i] + D[i];    /* S2 */
}
```

- **S1 uses the value B[i] computed by S2 in the previous iteration (loop-carried dependence)**
- **This dependence is not circular:**
  - S1 depends on S2 but S2 does not depend on S1.
- **Can be made parallel by replacing the code with the following:**

```
A[1] = A[1] + B[1];                 Loop Start-up code
for (i=1; i<=99; i=i+1) {
    B[i+1] = C[i] + D[i];
    A[i+1] = A[i+1] + B[i+1];
}
B[101] = C[100] + D[100];

                                    Loop Completion code
```

4

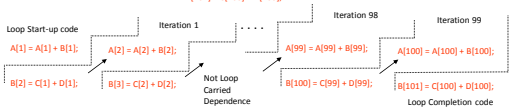
LLP Analysis Example 2

Original Loop:

```
for (i=1; i<=100; i=i+1) {
    A[i] = A[i] + B[i];      /* S1 */
    B[i+1] = C[i] + D[i];    /* S2 */
}
```

Modified Parallel Loop:

```
A[1] = A[1] + B[1];
for (i=1; i<=99; i=i+1) {
    B[i+1] = C[i] + D[i];
    A[i+1] = A[i+1] + B[i+1];
}
B[101] = C[100] + D[100];
```



5

Dynamic Scheduling Through Hardware Schemes

6

Static vs. Dynamic Scheduling

- Static Scheduling by compiler
  - Code scheduling for LD delay slots and branch delay slots
  - Avoiding data hazards
  - In-order instruction issue:
    - If an instruction is stalled, no later instructions can proceed.
    - Multiple copies of a unit may be idle – inefficiency
    - Static scheduling cannot help
- Dynamic Scheduling by Hardware
  - Allow *Out-of-order execution*, *Out-of-order "completion"*
  - Even though an instruction is stalled, later instructions, with no data dependencies with the instructions which are stalled and causing the stall, can proceed
  - Efficient utilization of functional unit with multiple units

7

Dynamic Pipeline Scheduling: The Concept

- Dynamic pipeline scheduling overcomes the limitations of in-order execution by allowing out-of-order instruction execution.
  - Works when dependencies are unknown at compile time
  - Simpler compiler
- Instructions are allowed to start executing out-of-order as soon as their operands are available.
- Example:

In the case of in-order execution
SUBD must wait for DIVD to complete
which stalled ADDD before starting execution
In out-of-order execution SUBD can start as soon
as the values of its operands F8, F14 are available.

```
DIVD  F0, F2, F4
ADDD  F10, F0, F8
SUBD  F12, F8, F14
```

This implies allowing out-of-order instruction "completion'.

8

Dynamic Pipeline Scheduling

Dynamic instruction scheduling can be accomplished by:

- **Dividing the Instruction Decode ID stage into two stages:**
  - Issue: Decode instructions, check for structural hazards.
  - Read operands: Wait until data hazard conditions, if any, are resolved, then read operands when available.

  (All instructions pass through the issue stage in order but can be stalled or pass each other in the read operands stage).
- **In the instruction fetch stage IF, fetch an additional instruction every cycle into a latch or several instructions into an instruction queue.**
- **Increase the number of functional units to meet the demands of the additional instructions in their EX stage.**

9

Dynamic Pipeline Scheduling

- Two dynamic scheduling approaches exist:
  - **Dynamic scheduling with a Scoreboard used first in CDC6600**
  - **The Tomasulo approach pioneered by the IBM 360/91**
- Most of the modern microprocessors use similar techniques

10

Dynamic Scheduling With A Scoreboard

- The scoreboard is a hardware mechanism that maintains an execution rate of one instruction per cycle by executing an instruction as soon as its operands are available and no hazard conditions prevent it.
- It replaces ID, EX, WB with four stages: ID1, ID2, EX, WB
- Every instruction goes through the scoreboard where a record of data dependencies is constructed (corresponds to instruction issue).
- A system with a scoreboard is assumed to have several functional units with their status information reported to the scoreboard.

11

Dynamic Scheduling With A Scoreboard

- If the scoreboard determines that an instruction cannot execute immediately it executes another instruction and keeps monitoring hardware units' status and decide when the blocked instruction can proceed to execute
- The scoreboard also decides when an instruction can write its results to registers
- Hazard detection and resolution is centralized in the scoreboard

12

Scoreboard Implications

- Out-of-order execution ==> WAR, WAW hazards?

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

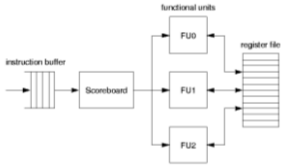
- If the pipeline executes SUBD before ADDD, it will yield incorrect execution
- A WAW hazard would occur. We must detect the hazard and stall until other completes.

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

13

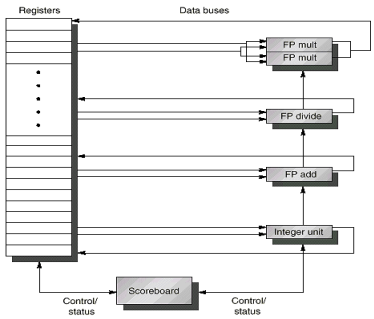
Scoreboard Specifics

- Several functional units
  - several floating-point units, integer units, and memory reference units
- Data dependencies (hazards) are detected when an instruction reaches the scoreboard
  - corresponding to instruction issue replacing part of the ID stage



- Scoreboard determines
  - when the instruction is ready for execution
  - based on when its operands and functional unit become available
  - where results are written

14



The basic structure of a MIPS processor with a scoreboard

15

Instruction Execution Stages with A Scoreboard

1 *Issue (ID1):* If a functional unit for the instruction is available, the scoreboard issues the instruction to the functional unit and updates its internal data structure; structural and WAW hazards are resolved here. (this replaces part of ID stage in the conventional MIPS pipeline).

2 *Read operands (ID2):* The scoreboard monitors the availability of the source operands. A source operand is available when no earlier active instruction writes it. When all source operands are available the scoreboard tells the functional unit to *read* all operands from the registers (no forwarding supported) and start execution (RAW hazards resolved here dynamically). This completes ID.

3 *Execution (EX):* The functional unit starts execution upon receiving operands. When the results are ready it notifies the scoreboard (replaces EX, MEM in MIPS).

4 *Write result (WB):* Once the scoreboard senses that a functional unit completed execution, it checks for WAR hazards and stalls the completing instruction if needed otherwise the write-back is completed.

16

Three Parts of the Scoreboard

1 Instruction status: Which of 4 steps the instruction is in.

2 Functional unit status: Indicates the state of the functional unit (FU). Nine fields for each functional unit:

- **Busy** — **Indicates whether the unit is busy or not**
- **Op** — **Operation to perform in the unit (e.g., ADD.D or SUB.D)**
- **Fi** — **Destination register**
- **Fj, Fk** — **Source-register numbers**
- **Qj, Qk** — **Functional units producing source registers Fj, Fk**
- **Rj, Rk** — **Flags indicating when Fj, Fk are ready (set to Yes after operand is available to read)**

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

17

A Scoreboard Example

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

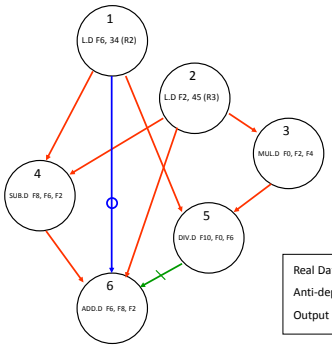
Functional Unit (FU)	# of FUs	EX Latency
Integer	1	0
Floating Point Multiply	2	10
Floating Point Add	1	2
Floating point Divide	1	40

All functional units are not pipelined

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

18

Dependency Graph For Example Code



Example Code

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

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

Output Dependence:
(1, 6)

Anti-dependence:
(5, 6)

Real Data Dependence (RAW)
Anti-dependence (WAR)
Output Dependence (WAW)

19

Scoreboard Example: Cycle 1

FP Latency: Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1			
L.D F2	45+	R3				
MUL.DF0	F2	F4				
SUB.DF8	F6	F2				
DIV.D F10	F0	F6				
ADD.DF6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

20

Scoreboard Example: Cycle 2

FP Latency: Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2		
L.D F2	45+	R3				
MUL.DF0	F2	F4				
SUB.DF8	F6	F2				
DIV.D F10	F0	F6				
ADD.DF6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

- Issue second L.D? No, stall on structural hazard

21

Scoreboard Example: Cycle 3

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	
L.D F2	45+	R3				
MUL.DF0	F2	F4	?			
SUB.DF8	F6	F2				
DIV.D F10	F0	F6				
ADD.DF6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
3 FU				Integer					

- Issue MUL.D? In-order issue !!!

22

Scoreboard Example: Cycle 4

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3				
MUL.DF0	F2	F4				
SUB.DF8	F6	F2				
DIV.D F10	F0	F6				
ADD.DF6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
4 FU				Integer					

23

Scoreboard Example: Cycle 5

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5			
MUL.D F0	F2	F4				
SUB.D F8	F6	F2				
DIV.D F10	F0	F6				
ADD.D F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
5 FU		Integer							

24

Scoreboard Example: Cycle 6

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6		
MUL.D F0	F2	F4	6			
SUB.D F8	F6	F2				
DIV.D F10	F0	F6				
ADD.D F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
6 FU	Mult1	Integer							

Scoreboard Example: Cycle 7

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	
MUL.D F0	F2	F4	6			
SUB.D F8	F6	F2	7			
DIV.D F10	F0	F6				
ADD.D F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	No								

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
7 FU	Mult1	Integer			Add				

- Read multiply operands?

Scoreboard Example: Cycle 8a (First half of cycle 8)

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	
MUL.D F0	F2	F4	6			
SUB.D F8	F6	F2	7			
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
8 FU	Mult1	Integer			Add	Divide			

Scoreboard Example: Cycle 8b (Second half of cycle 8)

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6			
SUB.D F8	F6	F2	7			
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
8 FU	Mult1				Add	Divide			

Scoreboard Example: Cycle 9

FP Latency: Add = 2 cycles, Multiply = 10, Divide = 40

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9		
SUB.D F8	F6	F2	7	9		
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2	?			

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
10	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
2	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
9 FU	Mult1				Add	Divide			

- Read operands for MUL.D & SUB.D? Issue ADD.D?

Scoreboard Example: Cycle 11

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9		
SUB.D F8	F6	F2	7	9	11	
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
8	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
0	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
11 FU	Mult1				Add	Divide			

Scoreboard Example: Cycle 12

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9		
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
7	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	No								
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
12 FU	Mult1					Divide			

- Read operands for DIV.D?

Scoreboard Example: Cycle 13

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9		
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2	13			

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
6	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
13 FU	Mult1					Add	Divide		

Scoreboard Example: Cycle 17

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9		
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2	13	14	16	?

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
2	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
17 FU	Mult1			Add		Divide			

- Write result of ADD.D? No, WAR hazard

Scoreboard Example: Cycle 20

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9	19	20
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8			
ADD.D F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
20 FU				Add		Divide			

Scoreboard Example: Cycle 21

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9	19	20
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8	21		
ADD.D F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
21 FU				Add		Divide			

Scoreboard Example: Cycle 22

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9	19	20
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8	21		
ADD.D F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
40	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
22 FU						Divide			

Scoreboard Example: Cycle 61

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9	19	20
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8	21	61	
ADD.D F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
61 FU						Divide			

37

Scoreboard Example: Cycle 62

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D F6	34+	R2	1	2	3	4
L.D F2	45+	R3	5	6	7	8
MUL.D F0	F2	F4	6	9	19	20
SUB.D F8	F6	F2	7	9	11	12
DIV.D F10	F0	F6	8	21	61	62
ADD.D F6	F8	F2	13	14	16	22

Instruction Block done

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	No								

Register result status

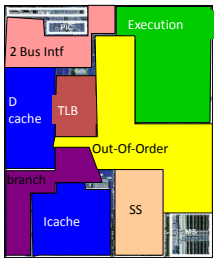
Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
62 FU									

- We have:
  - In-oder issue,
  - Out-of-order execute and commit

38

Where have all the transistors gone?

- Superscalar (multiple instructions per clock cycle)
- 3 levels of cache
- Branch prediction (predict outcome of decisions)
- Out-of-order execution (executing instructions in different order than programmer wrote them)



Intel Pentium III (10M transistors)

39

Pipelining
The Tomasulo's Algorithm

40

The Tomasulo's Algorithm

- From IBM 360/91
- Goal: High Performance using a limited number of registers without a special compiler
  - 4 double-precision FP registers on 360
  - Uses register renaming
- Why Study a 1966 Computer?
  - The descendants of this include: Alpha 21264, HP 8000, MIPS 10000, Pentium III, PowerPC 604, …

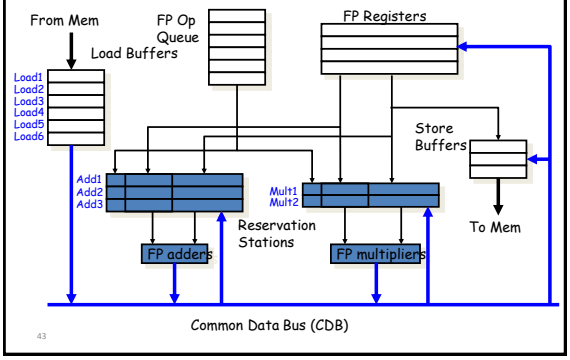
41

Tomasulo Algorithm

- Control & buffers are distributed with Function Units (FU)
  - FU buffers called "reservation stations (RS)"
  - Contain information about instructions, including operands
  - More reservation stations than registers, so can do optimizations compilers can't
- Registers in instructions replaced by values or pointers to reservation stations
  - form of register renaming
  - avoids WAR, WAW hazards
- Results to FU from RS, not through registers (equivalent of forwarding). A Common Data Bus (CDB) broadcasts results to all FUs (their RSes)
- Loads and Stores treated as FUs with RSes as well

42

Tomasulo Organization



43

Tomasulo Organization

Reservation Station Components

- Busy: Indicates reservation station or FU is busy
- Op: Operation to perform in the unit (e.g., + or –)
- Vj, Vk: Value of Source operands
- Qj, Qk: Reservation stations producing source registers (value to be written)
  - Note: Qj,Qk=0 => ready
- A: effective address

44

Tomasulo Organization

- Register result status— Qi
  - Indicates which functional unit will write each register, if one exists. Blank when no pending instructions that will write that register
- Common data bus
  - Normal data bus: data + destination (“go to” bus)
  - CDB: 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

45

Three Stages of Tomasulo Algorithm

- 1. Issue—get instruction from FP Op Queue
  - If reservation station free (no structural hazard), control issues the instruction & sends operands (renames registers).
- 2. Execute—operate on operands (EX)
  - When both operands ready then execute; if not ready, watch Common Data Bus for result
- 3. Write result—finish execution (WB)
  - Write on Common Data Bus to all awaiting units; mark reservation station available

46

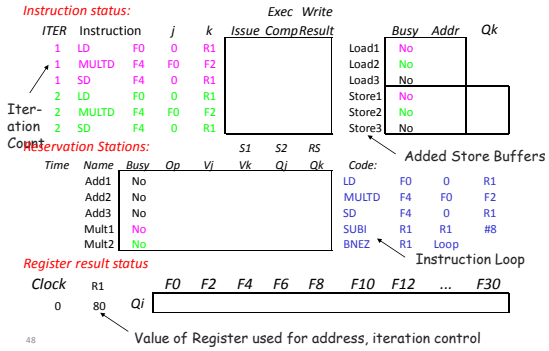
Tomasulo Loop Example

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

- This time assume multiply takes 4 clock cycles in the execution stage
- Assume 1st load takes 8 clock cycles (L1 cache miss) in the execution stage, 2nd load takes 1 extra cycle (hit)
- Assume store takes 3 cycles in the execution stage
- To be clear, will show clocks for SUBI, BNEZ
- Show about 2 iterations

47

Loop Example using simplified presentation for load/store components



48

Loop Example Cycle 1

Instruction status: *Exec Write*

ITER	Instruction	j	k	Issue	CompResult
1	LD F0 0 R1	1			

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

Reservation Stations:

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

Code:

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

Register result status

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

49

Loop Example Cycle 2

Instruction status: *Exec Write*

ITER	Instruction	j	k	Issue	CompResult
1	LD F0 0 R1	1			
1	MULTD F4 F0 F2	2			

	Busy	Addr	Qk
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	Qi	Load1		Mult1						

50

Loop Example Cycle 3

Instruction status: *Exec Write*

ITER	Instruction	j	k	Issue	CompResult
1	LD F0 0 R1	1			
1	MULTD F4 F0 F2	2			
1	SD F4 0 R1	3			

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

Reservation Stations:

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

Code:

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

Register result status

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

51

Loop Example Cycle 4

Instruction status: *Exec Write*

ITER	Instruction	j	k	Issue	CompResult
1	LD F0 0 R1	1			
1	MULTD F4 F0 F2	2			
1	SD F4 0 R1	3			

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

Reservation Stations:

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

Code:

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

Register result status

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

- Dispatching SUBI Instruction (not in FP queue)

52

Loop Example Cycle 5

Instruction status: *Exec Write*

ITER	Instruction	j	k	Issue	CompResult
1	LD F0 0 R1	1			
1	MULTD F4 F0 F2	2			
1	SD F4 0 R1	3			

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

Reservation Stations:

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

Code:

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

Register result status

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

- And, BNEZ instruction (not in FP queue)

53

Loop Example Cycle 6

Instruction status: *Exec Write*

ITER	Instruction	j	k	Issue	CompResult
1	LD F0 0 R1	1			
1	MULTD F4 F0 F2	2			
1	SD F4 0 R1	3			
2	LD F0 0 R1	6			

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

Reservation Stations:

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

Code:

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

Register result status

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

- Notice that F0 never sees Load from location 80

54

Loop Example Cycle 7

Instruction status: (Exec, Write)

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

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

Reservation Stations:

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

Code:

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

Register result status

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

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

55

Loop Example Cycle 8

Instruction status: (Exec, Write)

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

	Busy	Addr	Qk
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	Qi		Load2		Mult2				

56

Loop Example Cycle 9

Instruction status: (Exec, Write)

ITER	Instruction	j	k	Issue	Comp	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	Qk
Load1	Yes	80	
Load2	Yes	72	
Load3	No		
Store1	Yes	80	Mult1
Store2	Yes	72	Mult2
Store3	No		

Reservation Stations:

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

Code:

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

Register result status

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

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

57

Loop Example Cycle 10

Instruction status: (Exec, Write)

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

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

Reservation Stations:

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

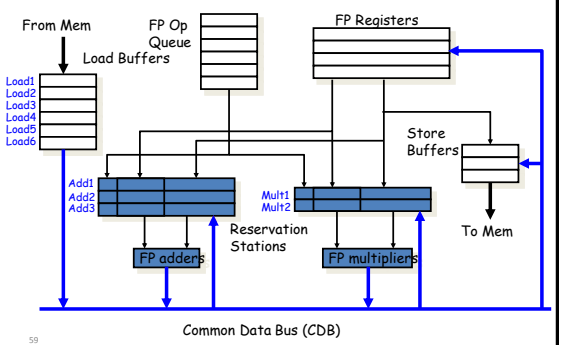
Register result status

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

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

58

Tomasulo Organization



59

Loop Example Cycle 11

Instruction status: (Exec, Write)

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

	Busy	Addr	Qk
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	Qi		Load3		Mult2				

- Next load in sequence

60

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	Qk
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					
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	Qi	Load3		Mult2					

- Why not issue third multiply?

61

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	Qk
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	Qi	Load3		Mult2					

- Why not issue third store?

62

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	Qk
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					
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	Qi	Load3		Mult2					

- Mult1 completing. Who is waiting?

63

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

Reservation Stations:

Time	Name	Busy	Op	Vj	S1 Vk	S2 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	Qi	Load3		Mult2					

- Mult2 completing. Who is waiting?

64

Loop Example Cycle 16

Instruction status:

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

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

Reservation Stations:

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

Code:

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

Register result status

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

65

Loop Example Cycle 17

Instruction status:

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

	Busy	Addr	Qk
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	Qi	Load3		Mult1					

66

Loop Example Cycle 18

Instruction status:

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

	Busy	Addr	Qk
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	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Multd		R(F2)	Load3	
	Mult2	No					

Code:

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

Register result status

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

67

Loop Example Cycle 19

Instruction status:

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

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

Reservation Stations:

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

Code:

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

Register result status

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

68

Loop Example Cycle 20

Instruction status:

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

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

Reservation Stations:

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

Code:

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

Register result status

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

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

69

Why can Tomasulo overlap iterations of loops?

- Register renaming
  - Multiple iterations use different physical destinations for registers (dynamic loop unrolling).
- Reservation stations
  - Buffer old values of registers - avoiding the WAR stall that we saw in the scoreboard.
- Other perspective: Tomasulo builds data flow dependency graph on the fly.

70

Tomasulo's scheme offers 2 major advantages

(1)the distribution of the hazard detection logic

- Distributed reservation stations and the CDB
- If multiple instructions waiting on single result, the 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.

(2) the elimination of stalls for WAW and WAR hazards

71

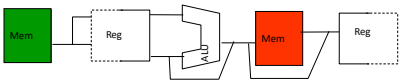
Recall from Pipelining

- Pipeline CPI = Ideal pipeline CPI + Structural Stalls + Data Hazard Stalls + Control Stalls
  - Ideal pipeline CPI: measure of the maximum performance attainable by the implementation
  - Structural hazards: HW cannot support this combination of instructions
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps)

72

Techniques to Reduce Stalls and Increase ILP

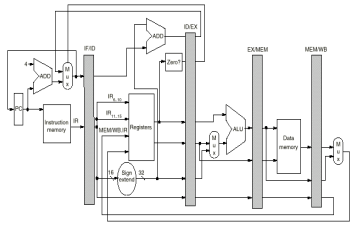
- **Hardware** Schemes to Reduce:
  - ❑ Structural hazards
    - ✓ Memory: Separate instruction and data memory
    - ✓ Registers: Write 1st half of cycle and read 2nd half of cycle



73

Techniques to Reduce Stalls

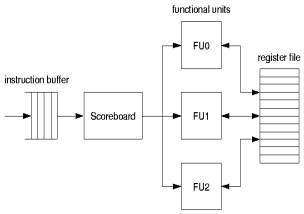
- **Hardware** Schemes to Reduce:
  - ❑ Data Hazards
    - ✓ Forwarding
  - ❑ Control Hazards
    - ✓ Moving the branch resolution earlier in the pipeline



74

Techniques to Reduce Stalls and Increase ILP

- **Hardware** Schemes to increase ILP:
  - ❑ Scoreboarding
    - ✓ Allows out-of-order execution of instructions



75

Techniques to Reduce Stalls and Increase ILP

- **Hardware** Schemes to increase ILP:
  - ❑ Scoreboarding
    - ✓ Allows out-of-order execution of instructions

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
L.D	F6	34+	R2	1	2	3	4
L.D	F2	45+	R3	5	6	7	8
MUL.D	F0	F2	F4	6	9	19	20
SUB.D	F8	F6	F2	7	9	11	12
DIV.D	F10	F0	F6	8	21	61	62
ADD.D	F6	F8	F2	13	14	16	22

- We have:
  - In-oder issue,
  - Out-of-order execute and "completion"

76

Techniques to Reduce Stalls and Increase ILP

- **Hardware** Schemes to reduce stalls
  - ❑ The Tomasulo's Algorithm
    - ✓ Similar to scoreboarding but more advanced (e.g., register renaming)
  - ❑ Control Hazards
    - ✓ Dynamic branch prediction (using buffer lookup schemes)

77

Techniques to Reduce Stalls and Increase ILP

- **Software** Schemes to Reduce:
  - ❑ Data Hazards
    - ✓ Compiler Scheduling: reduce load stalls

Original code with stalls:

```
        LD    Rb,b
Stall → LD    Rc,c
        DADD  Ra,Rb,Rc
        SD    Ra,a
        LD    Re,e
Stall → LD    Rf,f
        DSUB  Rd,Re,Rf
        SD    Rd,d
```

Scheduled code with no stalls:

```
LD    Rb,b
LD    Rc,c
LD    Re,e
DADD  Ra,Rb,Rc
LD    Rf,f
SD    Ra,a
DSUB  Rd,Re,Rf
SD    Rd,d
```

78

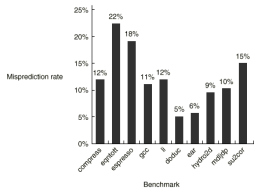
Techniques to Reduce Stalls and Increase ILP

- **Software** Schemes to Reduce:
  - ❑ Data Hazards
    - ✓ Compiler Scheduling: register renaming to eliminate WAW and WAR hazards

MUL.D	F0, F1, F2	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB	
ADD.D	F0, F3, F4		IF	ID	A1	A2	A3	A4	MEM	WB			

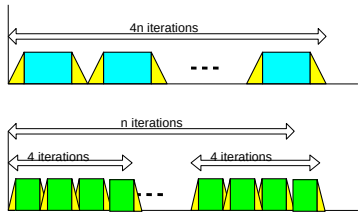
Techniques to Reduce Stalls and Increase ILP

- **Software** Schemes to Reduce:
  - ❑ Control Hazards
    - ✓ Branch prediction
      - ✓ Example : choosing backward branches **(loop)** as taken and forward branches **(if)** as not taken
      - ✓ Tracing Program behaviour



Techniques to Reduce Stalls and Increase ILP

- **Software** Schemes to Reduce:
  - ❑ Control Hazards
    - ✓ Loop unrolling



Techniques to Reduce Stalls and Increase ILP

- **Software** Schemes to Reduce:
  - ❑ Control Hazards
    - ✓ Increase loop-level parallelism

```
for (i=1; i<=100; i=i+1) {
        A[i] = A[i] + B[i];      /* S1 */
        B[i+1] = C[i] + D[i];    /* S2 */
    }
```

– Can be made parallel by replacing the code with the following:

```
A[1] = A[1] + B[1];
for (i=1; i<=99; i=i+1) {
    B[i+1] = C[i] + D[i];
    A[i+1] = A[i+1] + B[i+1];
}
B[101] = C[100] + D[100];
```