

# Lecture 6

## Instruction Level Parallelism (ILP)

(Ch.3)

**Cristinel Ababei**

Dept. of Electrical and Computer Engineering



MARQUETTE  
UNIVERSITY

**BE THE DIFFERENCE.**

Credits: Slides adapted from presentations of Sudeep Pasricha and others: Kubiawicz, Patterson, Mutlu, Elsevier

1

1

## Outline

- ILP, Loop Unrolling
- Static Branch Prediction
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- Tomasulo Algorithm
- Conclusion

2

2

## Instruction-Level Parallelism (ILP)

- **Instruction-Level Parallelism:** overlap the execution of instructions to improve performance
- Two approaches to exploit ILP that exists inside workloads:
  - 1) Rely on **software** technology to find parallelism, **statically** at compile-time (e.g., Itanium 2)
  - 2) Rely on **hardware** to help discover and exploit the parallelism **dynamically** (e.g., Pentium 4, AMD Opteron, IBM Power, Intel Core series, new ARM Cortex A9), and

3

3

## Instruction-Level Parallelism (ILP)

- For typical **Basic Block** (BB), existing ILP is quite small
  - BB: a straight-line code sequence with no branches in except to the entry and no branches out except at the exit
  - average dynamic branch frequency 15% to 25%  
=> 4 to 7 instructions execute between a pair of branches
  - Plus, instructions in BB likely to depend on each other
- To obtain substantial performance enhancements, we must exploit ILP across multiple basic blocks
- Simplest type of ILP: **loop-level parallelism** to exploit parallelism among iterations of a loop. e.g.,  
for (i=1; i<=1000; i=i+1)  
    x[i] = x[i] + y[i];

4

4

## Loop-Level Parallelism

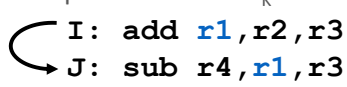
- Exploit loop-level parallelism by “**unrolling loop**” (which increases BB size) either by
  1. Static methods via loop unrolling by compiler
  2. Dynamic methods via branch prediction
- Determining **instruction dependence** is critical to Loop Level Parallelism
- If 2 instructions are
  - **Parallel**, they can execute simultaneously in a pipeline of arbitrary depth without causing any stalls (assuming no structural hazards)
  - **Dependent**, they are not parallel and must be executed in order, although they may often be partially overlapped

5

5

## Data Dependence and Hazards

- Instr<sub>j</sub> is **data dependent** (aka **true dependence**) on Instr<sub>i</sub>:
  1. Instr<sub>j</sub> tries to read operand before Instr<sub>i</sub> writes it
  2. or Instr<sub>j</sub> is data dependent on Instr<sub>k</sub> which is dependent on Instr<sub>i</sub>



```
I: add r1, r2, r3
J: sub r4, r1, r3
```
- If two instructions are data dependent, they cannot execute simultaneously or be completely overlapped
- Data dependence in instruction sequence  
⇒ data dependence in source code ⇒ effect of original data dependence must be preserved
- Data dependence causes a hazard in pipeline, called a **Read After Write (RAW) hazard**

6

6

# ILP and Data Dependences, Hazards

- HW/SW must preserve **program order**:  
ordered instructions would execute as if executed sequentially as determined by original source program
  - Dependences are a property of **programs**
- Presence of dependence indicates **potential** for a hazard, but actual hazard and length of any stall is property of the **pipeline**
- Importance of the data dependences
  - 1) indicates the possibility of a hazard
  - 2) determines order in which results must be calculated
  - 3) sets an upper bound on how much parallelism can possibly be exploited
- **Key Insight: exploit parallelism by preserving program order**


7

7

## Name Dependence #1: Anti-dependence

- **Name dependence**: when 2 instructions use same register or memory location, called a **name**, but no **flow of data** between the instructions associated with that name; **2 versions of name dependence**
- Instr<sub>j</sub> writes operand before Instr<sub>i</sub> reads it

```
    I: sub r4, r1, r3
    J: add r1, r2, r3
    K: mul r6, r1, r7
```



Called an “**anti-dependence**” by compiler writers.  
This results from reuse of the name “**r1**”

- If anti-dependence caused a hazard in the pipeline, called a **Write After Read (WAR) hazard**


8

8

## Name Dependence #2: Output dependence

- Instr<sub>j</sub> writes operand before Instr<sub>i</sub> writes it.

```
    I: sub r1, r4, r3
    J: add r1, r2, r3
    K: mul r6, r1, r7
```



- Called an “**output dependence**” by compiler writers  
This also results from the reuse of name “r1”
- If output dependence causes a hazard in the pipeline, called a **Write After Write (WAW) hazard**
- **Key Insight: Instructions involved in a name dependence can execute simultaneously if name used in instructions is changed so instructions do not conflict**
  - **Register renaming** resolves name dependence for regs
  - Either by compiler or by HW

9

9

## Control Dependences

- Every instruction is control dependent on some set of branches
- In general, these **control dependences must be preserved to preserve program order**

```
if p1 {
    S1;
};
if p2 {
    S2;
}
```

- S1 is **control dependent** on p1, and S2 is **control dependent** on p2 but not on p1.

10

10

## Control Dependence Ignored

- Key Insight: Control dependences need not be preserved
  - Willing to execute instructions that should not have been executed, thereby violating the control dependences, if can do so without affecting correctness of the program
- Instead, 2 properties critical to program correctness are
  - 1) Exception behavior
  - 2) Data flow

11

11

## Example of Software Techniques

- This code, adds a scalar to a vector:

```
for (i=1000; i>0; i=i-1)
    x[i] = x[i] + s;
```
- Assume following latencies for all examples
  - Ignore delayed branch in these examples

| <i>Instruction<br/>producing result</i> | <i>Instruction<br/>using result</i> | <i>Stall latency in<br/>clock cycles</i> |
|---|-------------------------------------|--|
| FP ALU op                               | Another FP ALU op                   | 3  |
| FP ALU op                               | Store double                        | 2  |
| Load double                             | FP ALU op                           | 1  |
| Load double                             | Store double                        | 0  |
| Integer op                              | Integer op                          | 0  |

12

12

## FP Loop: Where are the Hazards?

```

Loop: LD    F0,0(R1)    ;F0=vector element
      ADDD  F4,F0,F2    ;add scalar from F2
      SD    0(R1),F4    ;store result
      SUBI  R1,R1,8     ;decrement pointer 8B (DW)
      BNEZ  R1,Loop     ;branch R1!=zero
      NOP                      ;delayed branch slot
    
```

| <i>Instruction<br/>producing result</i> | <i>Instruction<br/>using result</i> | <i>Stall latency in<br/>clock cycles</i> |
|---|-------------------------------------|--|
| FP ALU op                               | Another FP ALU op                   | 3  |
| FP ALU op                               | Store double                        | 2  |
| Load double                             | FP ALU op                           | 1  |
| Load double                             | Store double                        | 0  |
| Integer op                              | Integer op                          | 0  |

Where are the stalls?

13

13

## FP Loop: Showing Stalls

```

1 Loop: LD    F0,0(R1)    ;F0=vector element
2      stall
3      ADDD  F4,F0,F2    ;add scalar in F2
4      stall
5      stall
6      SD    0(R1),F4    ;store result
7      SUBI  R1,R1,8     ;decrement pointer 8B (DW)
8      BNEZ  R1,Loop     ;branch R1!=zero
9      stall             ;delayed branch slot
    
```

| <i>Instruction<br/>producing result</i> | <i>Instruction<br/>using result</i> | <i>Stall latency in<br/>clock cycles</i> |
|---|-------------------------------------|--|
| FP ALU op                               | Another FP ALU op                   | 3  |
| FP ALU op                               | Store double                        | 2  |
| Load double                             | FP ALU op                           | 1  |
| Load double                             | Store double                        | 0  |
| Integer op                              | Integer op                          | 0  |

9 clocks: How can we minimize stalls?

14

14

# Revised FP Loop Minimizing Stalls

```

1 Loop:   LD      F0,0(R1)
2         stall
3         ADDD    F4,F0,F2
4         SUBI    R1,R1,8
5         BNEZ    R1,Loop      ;delayed branch
6         SD      8(R1),F4      ;altered when move past SUBI
    
```

Move SD after BNEZ and change address of SD

| Instruction<br>producing result | Instruction<br>using result | Stall latency in<br>clock cycles |
|---------------------------------|-----------------------------|----------------------------------|
| FP ALU op                       | Another FP ALU op           | 3                                |
| FP ALU op                       | Store double                | 2                                |
| Load double                     | FP ALU op                   | 1                                |
| Load double                     | Store double                | 0                                |
| Integer op                      | Integer op                  | 0                                |

6 clocks: Can we make this code any faster?

15

15

# Unroll Loop Four Times (straightforward way)

```

1 Loop:   LD      F0,0(R1)
2         ADDD    F4,F0,F2
3         SD      0(R1),F4
4         LD      F6,-8(R1)
5         ADDD    F8,F6,F2
6         SD      -8(R1),F8
7         LD      F10,-16(R1)
8         ADDD    F12,F10,F2
9         SD      -16(R1),F12
10        LD      F14,-24(R1)
11        ADDD    F16,F14,F2
12        SD      -24(R1),F16
13        SUBI    R1,R1,#32
14        BNEZ    R1,LOOP
15        NOP
    
```

;drop SUBI & BNEZ  
 ;drop SUBI & BNEZ  
 ;drop SUBI & BNEZ  
 ;alter to 4\*8

Rewrite  
loop to  
minimize  
stalls?

$15 + 4 \times (1+2) = 27$  clock cycles, or 6.8 per iteration

Assumes R1 is multiple of 4

16

16



## Unrolled Loop That Minimizes Stalls

```
1 Loop:  LD      F0, 0(R1)
2        LD      F6, -8(R1)
3        LD      F10, -16(R1)
4        LD      F14, -24(R1)
5        ADDDD   F4, F0, F2
6        ADDDD   F8, F6, F2
7        ADDDD   F12, F10, F2
8        ADDDD   F16, F14, F2
9        SD      0(R1), F4
10       SD      -8(R1), F8
11       SD      -16(R1), F12
12       SUBI    R1, R1, #32
13       BNEZ    R1, LOOP
14       SD      8(R1), F16 ; 8-32 = -24
```

14 clock cycles, or 3.5 per iteration

- What assumptions made when moved code?

- OK to move store past SUBI even though changes register
- OK to move loads before stores: get right data?

17

17

## Compiler Perspectives on Code Movement

- Compiler concerned about dependences in program
- Whether or not a HW hazard depends on pipeline
- Try to schedule instructions to avoid hazards that cause performance losses
- (True) Data dependences (RAW)
  - Instruction i produces a result used by instruction j, or
  - Instruction j is data dependent on instruction k, and instruction k is data dependent on instruction i.
- If dependent, can't execute in parallel
- Easy to determine for registers (fixed names)
- Hard for memory ("memory disambiguation" problem)

18

18

# Compiler Perspectives on Code Movement

- Another kind of dependence called **name dependence**: two instructions use same name (register or memory location) but don not exchange data
- **Antidependence** (WAR if a hazard for HW)
  - Instruction j writes a register or memory location that instruction i reads from and instruction i is executed first
- **Output dependence** (WAW if a hazard for HW)
  - Instruction i and instruction j write the same register or memory location; ordering between instructions must be preserved.

19

19

## Where are the name dependences?

```
1 Loop:  LD      F0,0(R1)
2        ADDD    F4,F0,F2
3        SD      0(R1),F4      ;drop SUBI & BNEZ
4        LD      F0,-8(R1)
5        ADDD    F4,F0,F2
6        SD      -8(R1),F4     ;drop SUBI & BNEZ
7        LD      F0,-16(R1)
8        ADDD    F4,F0,F2
9        SD      -16(R1),F4    ;drop SUBI & BNEZ
10       LD      F0,-24(R1)
11       ADDD    F4,F0,F2
12       SD      -24(R1),F4
13       SUBI    R1,R1,#32     ;alter to 4*8
14       BNEZ    R1,LOOP
15       NOP
```

**How can we remove them?**

20

20

## Where are the name dependences?

```
1 Loop:  LD      F0,0(R1)
2        ADDD    F4,F0,F2
3        SD      0(R1),F4          ;drop SUBI & BNEZ
4        LD      F6,-8(R1)
5        ADDD    F8,F6,F2
6        SD      -8(R1),F8        ;drop SUBI & BNEZ
7        LD      F10,-16(R1)
8        ADDD    F12,F10,F2
9        SD      -16(R1),F12      ;drop SUBI & BNEZ
10       LD      F14,-24(R1)
11       ADDD    F16,F14,F2
12       SD      -24(R1),F16
13       SUBI    R1,R1,#32        ;alter to 4*8
14       BNEZ    R1,LOOP
15       NOP
```

Called “register renaming”

21

21

## Compiler Perspectives

- Name Dependences are Hard to discover for Memory Accesses
  - Does  $100(R4) = 20(R6)$ ?
  - From different loop iterations, does  $20(R6) = 20(R6)$ ?
- Our example required compiler to know that if R1 does not change then:

$$0(R1) \neq -8(R1) \neq -16(R1) \neq -24(R1)$$

There were no dependences between some loads and stores so they could be moved by each other

22

22

## When is it Advantageous to Unroll Loop?

- Example: Where are data dependences?  
(A,B,C distinct & nonoverlapping)

```
for (i=0; i<100; i=i+1) {  
    A[i+1] = A[i] + C[i];    /* S1 */  
    B[i+1] = B[i] + A[i+1];  /* S2 */  
}
```

1. S2 uses the value, A[i+1], computed by S1 in the same iteration.
2. S1 uses a value computed by S1 in an earlier iteration, since iteration i computes A[i+1] which is read in iteration i+1. The same is true of S2 for B[i] and B[i+1].

This is a “**loop-carried dependence**”: between iterations

- For our prior example, each iteration was distinct
  - In this case, iterations can't be executed in parallel?
  - **Loop-carried dependences**: prevent instruction reordering after unrolling as we did in an earlier example

23

23

## Five (5) Loop Unrolling Decisions

- Requires understanding how one instruction depends on another and how the instructions can be changed or reordered given the dependences.
- Main steps:
  1. Determine loop unrolling useful by finding that loop iterations were independent
  2. Use different registers to avoid unnecessary constraints forced by using same registers for different computations
  3. Eliminate the extra test and branch instructions and adjust the loop termination and iteration code
  4. Determine that loads and stores in unrolled loop can be interchanged by observing that loads and stores from different iterations are independent
    - Transformation requires analyzing memory addresses and finding that they do not refer to the same address
  5. Schedule the code, preserving any dependences needed to yield the same result as the original code

24

24

## Three (3) Limits to Loop Unrolling

1. Decrease in amount of overhead amortized with each extra unrolling
    - Amdahl's Law
  2. Growth in code size
    - For larger loops, concern it increases the instruction cache miss rate
  3. Register pressure: potential shortfall in registers created by aggressive unrolling and scheduling
    - If not be possible to allocate all live values to registers, may lose some or all of its advantage
- Loop unrolling reduces impact of branches on pipeline!
  - **Another way is branch prediction** (see next slides)

25

25

## Outline

- ILP, Loop Unrolling
- **Static Branch Prediction**
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- Tomasulo Algorithm
- Conclusion

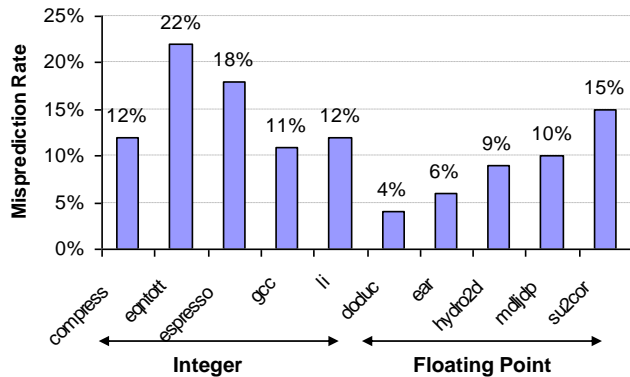
26

26

## Static Branch Prediction

- To reorder code around branches, we need to predict branch statically when compiling
- Simplest scheme is to **predict a branch as taken**
  - Average misprediction = untaken branch frequency = 34% SPEC

**More accurate scheme predicts branches using profile information collected from earlier runs, and modify prediction based on last run**



27

27

## Outline

- ILP, Loop Unrolling
- Static Branch Prediction
- **Dynamic Branch Prediction**
- Overcoming Data Hazards with Dynamic Scheduling
- Tomasulo Algorithm
- Conclusion

28

28

# Dynamic Branch Prediction

*"learning based on past behavior"*

## Temporal correlation

The way a branch resolves may be a good predictor of the way it will resolve at the next execution

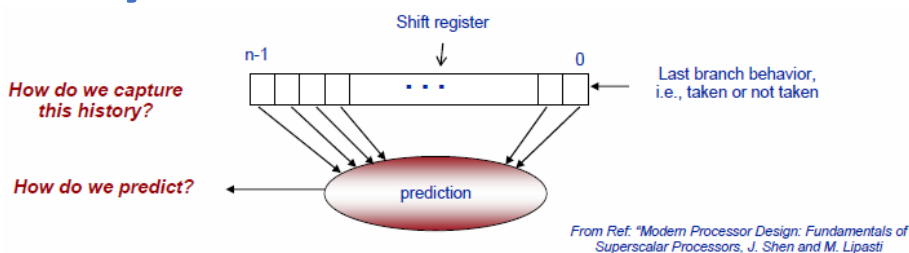
## Spatial correlation

Several branches may resolve in a highly correlated manner (*a preferred path of execution*)

29

29

# Dynamic Branch Prediction



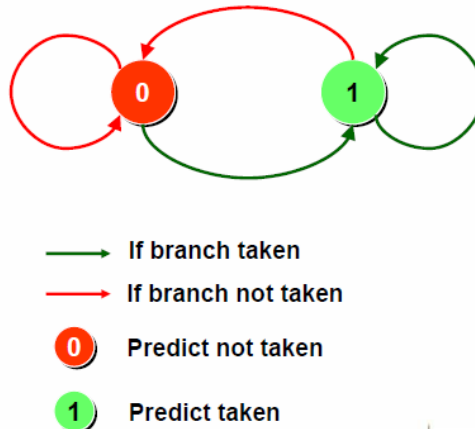
- Use past behavior to predict the future
- Local vs. global behaviors
  - Branches show surprisingly good correlation with one another and their history
    - They are not totally random events

30

30

# FSM of the Simplest Dynamic BP

- A 2-state machine
- Change mind fast



31

31

## Example using 1-bit branch history table

```

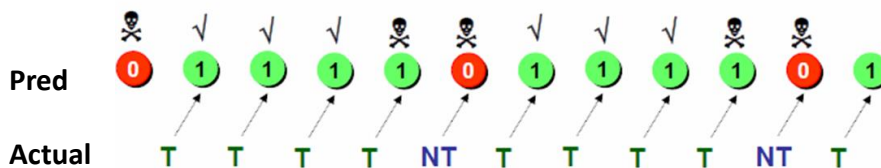
for (i=0; i<4; i++) {
    ....
}

```

```

addi r10, r0, 4
addi r1, r1, r0
L1:
... ..
addi r1, r1, 1
bne r1, r10, L1

```



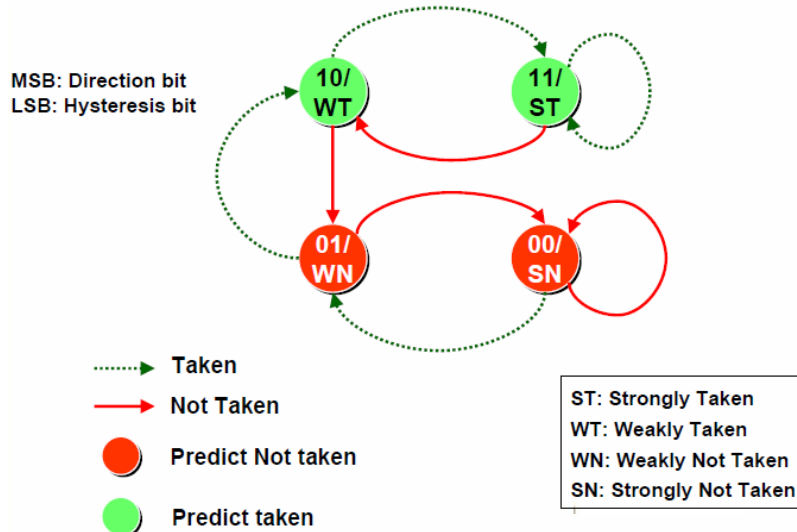
60% accuracy

32

32



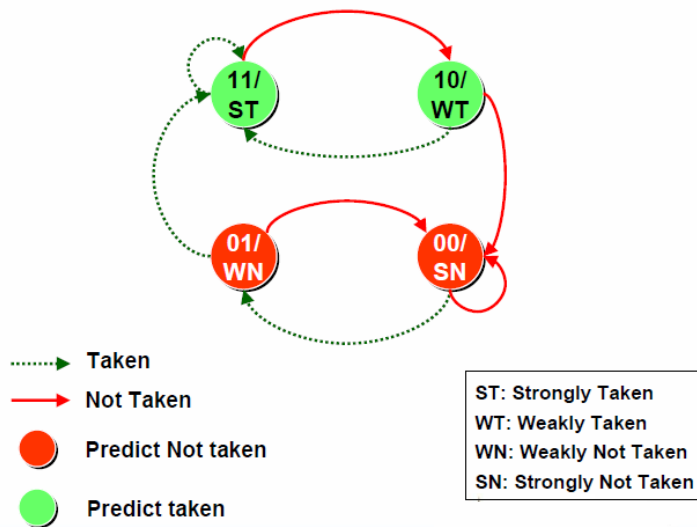
## 2-bit Saturating Up/Down Counter Predictor



33

33

## 2-bit Counter Predictor (Another Scheme)



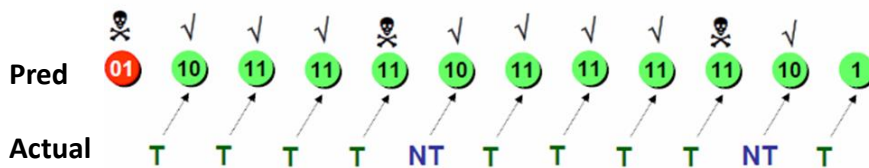
34

34

## Example using 2-bit up/down counter

```
for (i=0; i<4; i++) {
    ...
}
```

```
addi r10, r0, 4
addi r1, r1, r0
L1:
... ..
addi r1, r1, 1
bne r1, r10, L1
```



80% accuracy

35

35

## Branch History Table (BHT)

- BHT is a table of "Predictors"

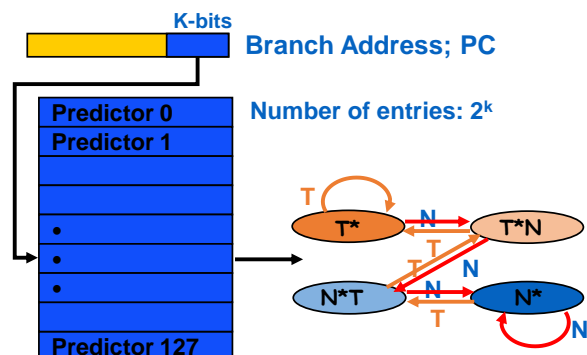
- 2-bit, saturating counters indexed by PC address of Branch

- In Fetch phase of branch:

- Predictor from BHT used to make prediction

- When branch completes:

- Update corresponding Predictor

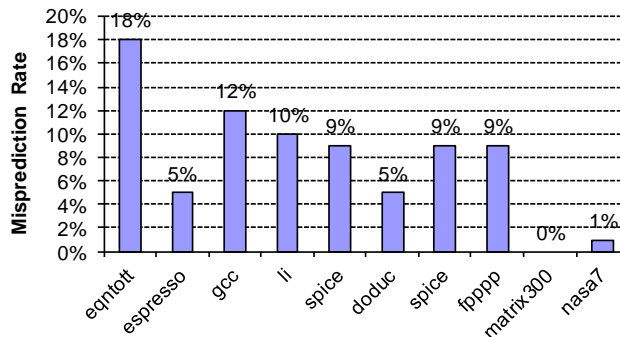


36

36

## BHT Accuracy

- Mispredict because either:
  - Wrong guess for that branch
  - Got branch history of wrong branch when index the table
- 4096 entry table:



37

37

## Advanced Techniques:

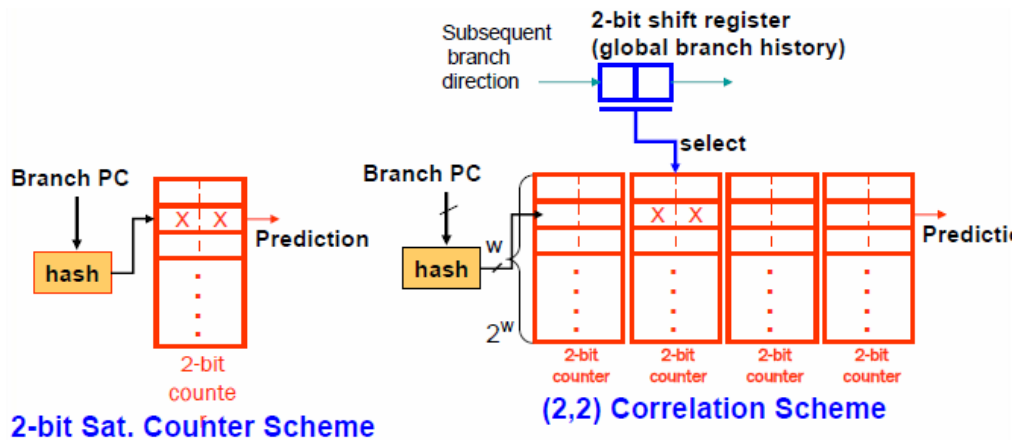
### 1. Correlated Branch Prediction (2-level Predictors)

- Hypothesis: recent branches are correlated; that is, behavior of recently executed branches affects prediction of current branch
- Keep track of the behavior of previous branches, and use that to predict the behavior of the current branch
- Idea:
  - Record  $m$  most recently executed branches (represents a path through the program) as taken or not taken
  - Use that pattern to select the proper  $n$ -bit branch history table
- In general,  $(m,n)$  predictor means record last  $m$  branches to select between  $2^m$  history tables, each with  $n$ -bit counters
  - Thus, old 2-bit BHT is a  $(0,2)$  predictor
- Global Branch History:  $m$ -bit shift register keeping T/NT status of last  $m$  branches

38

38

## Correlated Branch Predictor



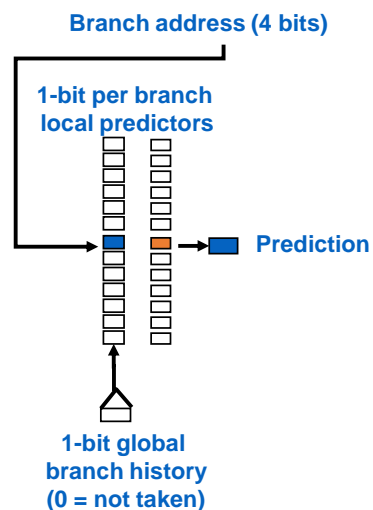
- (M,N) correlation scheme
  - M: shift register size (# bits)
  - N: N-bit counter

39

39

## Correlating Branches

- Again, the idea is: taken/not taken of recently executed branches is related to behavior of next branch (as well as the history of that branch behavior)
- Then, behavior of recent branches selects between, say, 2 predictions of next branch, updating just that prediction
- Example: **(1,1) predictor**
  - 1-bit global, 1-bit local

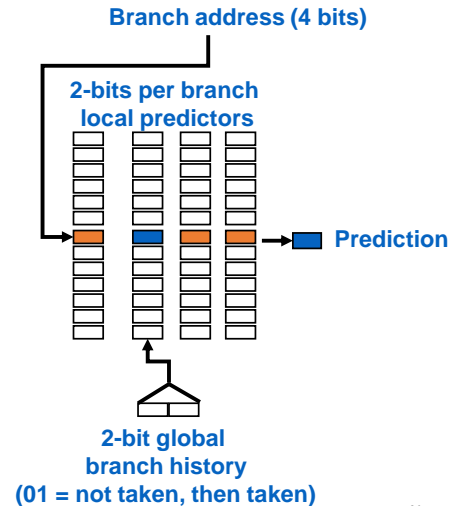


40

40

## Correlating Branches

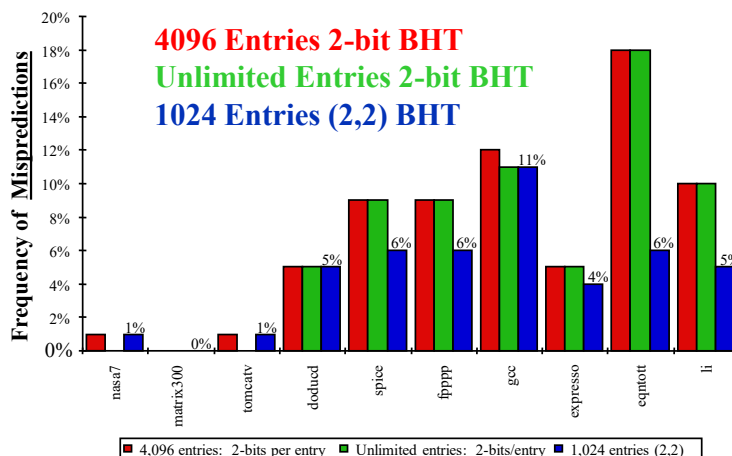
- General form: **(m, n) predictor**
- m bits for global history, n bits for local history
- Records correlation between m+1 branches
- Simple implementation: global history can be stored in a shift register
- Example: **(2,2) predictor**
  - 2-bit global, 2-bit local



41

41

## Accuracy of Different Schemes



42

42

## Advanced Techniques:

### 2. Tournament Predictors

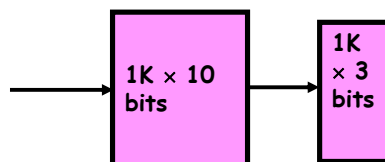
- Motivation for correlating branch predictors:
  - 2-bit local predictor failed on important branches
  - By **adding global information**, performance improved
- Tournament predictors: use two predictors, 1 based on global information and 1 based on local information, and combine with a selector
- Hopes to select right predictor for right branch (or right context of branch)

43

43

### Tournament Predictor in Alpha 21264

- **Local predictor:** consists of a 2-level predictor:
  - **Top level:** A local history table consisting of 1024 10-bit entries; each 10-bit entry corresponds to the most recent 10 branch outcomes for the entry. 10-bit history allows patterns of 10 branches to be discovered and predicted
  - **Next level:** Selected entry from the local history table is used to index a table of 1K entries consisting a 3-bit saturating counters, which provide the local prediction
- Total size:  $4K \times 2 + 4K \times 2 + 1K \times 10 + 1K \times 3 = 29K \text{ bits!}$   
(~180K transistors)

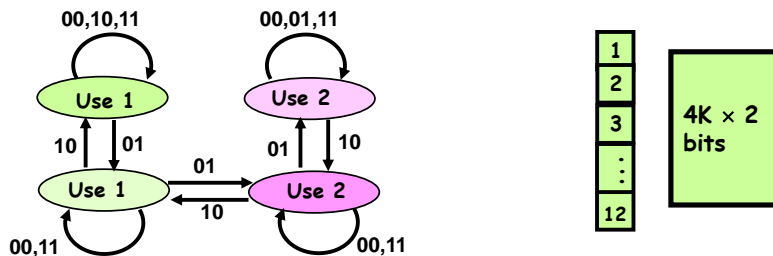


44

44

## Tournament Predictor in Alpha 21264

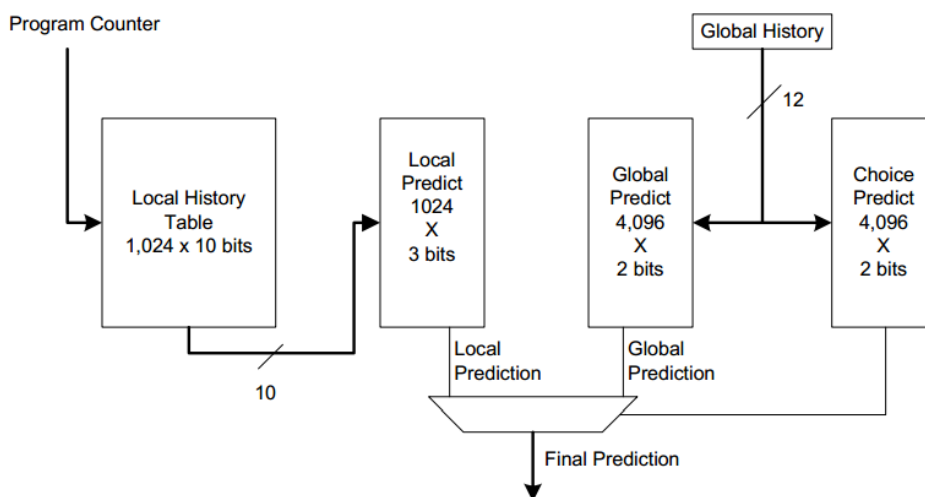
- 4K 2-bit counters to choose from among a global predictor and a local predictor
- **Global predictor:** also has 4K entries and is indexed by the history of the last 12 branches; each entry in the global predictor is a standard 2-bit predictor
  - 12-bit pattern:
    - » ith bit is 0 => ith prior branch not taken;
    - » ith bit is 1 => ith prior branch taken;



45

45

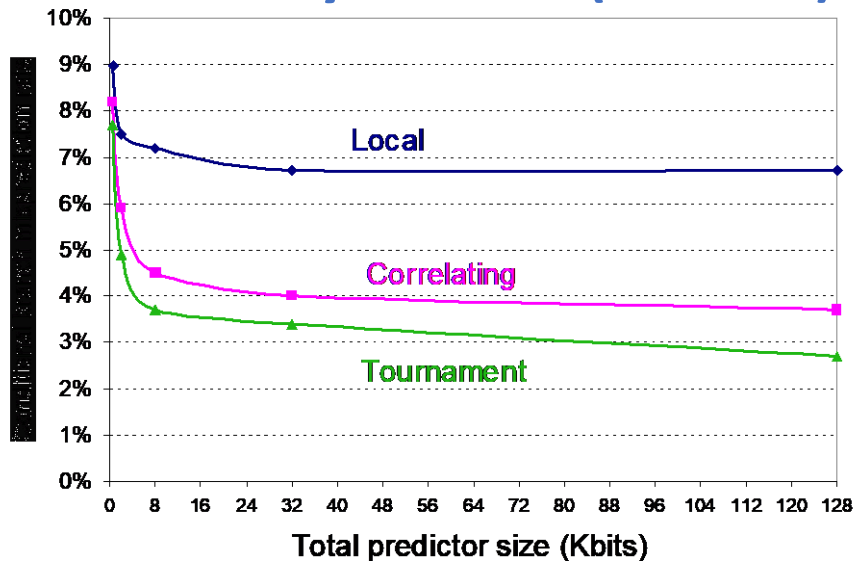
## Tournament Predictor in Alpha 21264



46

46

## Accuracy vs. Size (SPEC89)



47

47

## Dynamic Branch Prediction Summary

- Prediction becoming important part of execution
- Branch History Table: 2 bits for loop accuracy
- Correlation: recently executed branches correlated with next branch
  - Either different branches,
  - Or different executions of same branches
- Tournament predictors take insight to next level, by using multiple predictors
  - Usually one based on global information and one based on local information, and combining them with a selector
  - Tournament predictors using  $\approx 30K$  bits were in processors like the Power5 and Pentium 4 (circa 2006)

48

48



# Outline

- ILP, Loop Unrolling
- Static Branch Prediction
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- Tomasulo Algorithm
- Conclusion

49

49

## Dynamic Scheduling

- Dynamic scheduling - Hardware rearranges the instruction execution to reduce stalls while maintaining data flow and exception behavior
- It handles cases when dependences are unknown at compile time
- It allows processor to tolerate unpredictable delays such as cache misses, by executing other code while waiting for the miss to resolve
- It allows code that is compiled for one pipeline to run efficiently on a different pipeline
- It simplifies the compiler
- Hardware speculation - Technique with significant performance advantages, builds on dynamic scheduling 50

50

## HW Schemes: Instruction Parallelism

- Key idea: Allow instructions behind stall to proceed

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

- Enables **out-of-order execution** and allows **out-of-order completion** (e.g., SUBD)
- In a dynamically scheduled pipeline, all instructions still pass through issue stage in order (**in-order issue**)
- Will distinguish when an instruction begins execution and when it completes execution; between 2 times, the instruction is in execution

51

51

## Dynamic Scheduling – Splitting the ID Stage

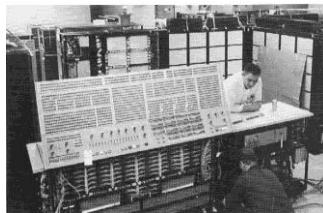
- Simple pipeline had 1 stage to check both structural and data hazards: **Instruction Decode (ID)**, also called **Instruction Issue**
- To allow out-of-order execution, split the ID pipe stage of simple 5-stage pipeline into 2 stages:
  - **Issue** - Decode instructions, check for structural hazards
  - **Read operands** - Wait until no data hazards, then read operands

52

52

## A Dynamic Algorithm: Tomasulo's Algorithm

- For IBM 360/91 (before caches!)
  - Long memory latency



- 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, Pentium 4, AMD Opteron, Power 5, Intel Core i3/i5/i7, ...

53

53

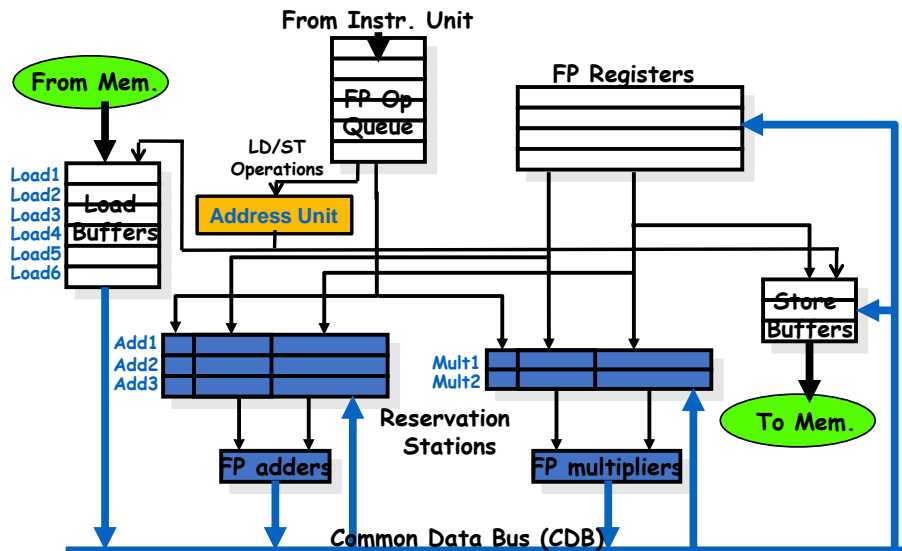
## Tomasulo Algorithm: Key Ideas

- Registers in instructions replaced by values or pointers to **reservation stations (RS)**; called **register renaming**;
  - Register renaming in hardware: **avoids WAR, WAW hazards**
  - More reservation stations than registers, so, can do optimizations compilers can not
- Tracks when operands for instructions are available to **minimize RAW hazards**
- Control & buffers **distributed with Function Units (FU)**
  - FU buffers called **reservation stations**; have instructions that have been issued but pending operands or wait for available FU
- Results to FU from RS, **not through registers**, over **Common Data Bus** that broadcasts results to all FUs
  - Avoids RAW hazards by executing an instruction only when its operands are available
- Load and Stores treated as FUs with RSs as well

54

54

## Structure of Tomasulo-based Processor



55

55

## Reservation Station Components

**Op:** Operation to perform in the unit (e.g., + or -)

$V_j, V_k$ : Value of Source operands

- Store buffer has V field, result to be stored

**Qj, Qk:** Reservation stations producing source registers (value to be written)

- Note:  $Q_j, Q_k = 0 \Rightarrow$  ready
- Store buffers only have  $Q_i$  for RS producing result

**Busy:** Indicates reservation station or FU is busy

**A(ddress):** holds information for the memory address calculation for a load or store

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

56

56

## Basic Functions of Some Elements

- **Load buffers** have 3 functions:
  1. hold components of effective address until it is computed
  2. track outstanding loads that are waiting on the memory
  3. hold the results of the completed loads that are waiting for the CDB
- **Store buffers** have 3 functions:
  1. hold components of effective address until it is computed
  2. hold destination memory addresses of outstanding stores that are waiting for the data value to store
  3. hold address and value to store until memory unit is available
- All results from FP units or the load unit are put on CDB, which goes to:
  - the FP registers
  - the reservation stations, and
  - the store buffers

57

57

## Three Stages of Tomasulo Algorithm

- 1.Issue**—get instruction from FP Op Queue
    - If reservation station free (no structural hazard), control issues instr & sends operands (renames registers)
    - Stall issue if structural hazard
  - 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
- 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

58

58

# Tomasulo Example

Instruction stream

Instruction status:

| Instruction | j   | k   | Issue | Comp | 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<br>Vj | S2<br>Vk | RS<br>Qj | RS<br>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     |    |    |    |    |    |     |     |     |     |

Clock cycle  
counter

59

59

## Tomasulo Example Cycle 1

Instruction status:

| Instruction | j   | k   | Issue | Comp | 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<br>Vj | S2<br>Vk | RS<br>Qj | RS<br>Qk |
|------|-------|------|----|----------|----------|----------|----------|
|      | Add1  | No   |    |          |          |          |          |
|      | Add2  | No   |    |          |          |          |          |
|      | Add3  | No   |    |          |          |          |          |
|      | Mult1 | No   |    |          |          |          |          |
|      | Mult2 | No   |    |          |          |          |          |

Register result status:

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

60

60

## Tomasulo Example Cycle 2

**Instruction status:**

| Instruction | j   | k   | Exec Write |      |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|------------|------|--------|-------|-------|-------|------|---------|
|             |     |     | Issue      | Comp | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2         | 1    |        |       |       |       | Yes  | 34+R2   |
| LD          | F2  | 45+ | R3         | 2    |        |       |       |       | Yes  | 45+R3   |
| MULTD       | F0  | F2  | F4         |      |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2         |      |        |       |       |       | No   |         |
| DIVD        | F10 | F0  | F6         |      |        |       |       |       | No   |         |
| ADDD        | F6  | F8  | F2         |      |        |       |       |       | No   |         |

**Reservation Stations:**

| Time | Name  | Busy | Op | S1<br>Vj | S2<br>Vk | RS<br>Qj | RS<br>Qk |
|------|-------|------|----|----------|----------|----------|----------|
|      |       |      |    |          |          |          |          |
|      | Add1  | No   |    |          |          |          |          |
|      | Add2  | No   |    |          |          |          |          |
|      | Add3  | No   |    |          |          |          |          |
|      | Mult1 | No   |    |          |          |          |          |
|      | Mult2 | No   |    |          |          |          |          |

**Register result status:**

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

**Note: Can have multiple loads outstanding**

61

61

## Tomasulo Example Cycle 3

**Instruction status:**

| Instruction | j   | k   | Exec Write |      |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|------------|------|--------|-------|-------|-------|------|---------|
|             |     |     | Issue      | Comp | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2         | 1    | 3      |       |       |       | Yes  | 34+R2   |
| LD          | F2  | 45+ | R3         | 2    |        |       |       |       | Yes  | 45+R3   |
| MULTD       | F0  | F2  | F4         | 3    |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2         |      |        |       |       |       | No   |         |
| DIVD        | F10 | F0  | F6         |      |        |       |       |       | No   |         |
| ADDD        | F6  | F8  | F2         |      |        |       |       |       | No   |         |

**Reservation Stations:**

| Time | Name  | Busy | Op    | S1<br>Vj | S2<br>Vk | RS<br>Qj | RS<br>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 |  |  |  |  |  |  |  |  |  |
|-------|--|--------------------------------|--|--|--|--|--|--|--|--|--|
|       |  | FU Mult1 Load2 Load1           |  |  |  |  |  |  |  |  |  |
| 3     |  |                                |  |  |  |  |  |  |  |  |  |

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

62

62

## Tomasulo Example Cycle 4

Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      |       |       |       | Yes  | 45+R3   |
| MULTD       | F0  | F2  | F4    | 3          |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          |        |       |       |       | No   |         |
| DIVD        | F10 | F0  | F6    |            |        |       |       |       | No   |         |
| ADDD        | F6  | F8  | F2    |            |        |       |       |       | No   |         |

Reservation Stations:

| Time | Name  | Busy | Op    | S1    |       | S2 |    | RS |  | RS    |
|------|-------|------|-------|-------|-------|----|----|----|--|-------|
|      |       |      |       | Vj    | Vk    | Qj | 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?

63

## Tomasulo Example Cycle 5

Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          |        |       |       |       | No   |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       | No   |         |
| ADDD        | F6  | F8  | F2    |            |        |       |       |       | No   |         |

Reservation Stations:

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

Register result status:

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

Timer starts down for Add1, Mult1

64



## Tomasulo Example Cycle 6

Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          |        |       |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          |        |       |       |       |      |         |

Reservation Stations:

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

Register result status:

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

Issue ADDD here despite name dependency on F6?

65

65

## Tomasulo Example Cycle 7

Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      |       |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          |        |       |       |       |      |         |

Reservation Stations:

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

Register result status:

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

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

66

66

## Tomasulo Example Cycle 8

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8     |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          |        |       |       |       |      |         |

### Reservation Stations:

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

### Register result status:

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

67

67

## Tomasulo Example Cycle 9

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8     |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          |        |       |       |       |      |         |

### Reservation Stations:

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

### Register result status:

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

68

68

## Tomasulo Example Cycle 10

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Busy | Address                 |
|-------------|-----|-----|-------|------------|--------|------|-------------------------|
|             |     |     |       | Comp       | Result |      |                         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4    | Load1<br>Load2<br>Load3 |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5    |                         |
| MULTD       | F0  | F2  | F4    | 3          |        |      |                         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8    |                         |
| DIVD        | F10 | F0  | F6    | 5          |        |      |                         |
| ADDD        | F6  | F8  | F2    | 6          | 10     |      |                         |

### Reservation Stations:

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

### Register result status:

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

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

69

69

## Tomasulo Example Cycle 11

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Busy | Address                 |
|-------------|-----|-----|-------|------------|--------|------|-------------------------|
|             |     |     |       | Comp       | Result |      |                         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4    | Load1<br>Load2<br>Load3 |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5    |                         |
| MULTD       | F0  | F2  | F4    | 3          |        |      |                         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8    |                         |
| DIVD        | F10 | F0  | F6    | 5          |        |      |                         |
| ADDD        | F6  | F8  | F2    | 6          | 10     | 11   |                         |

### Reservation Stations:

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

### Register result status:

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

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

70

70

## Tomasulo Example Cycle 12

### Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Write</i> |               |    | <i>Busy</i> | <i>Address</i> |
|-------------|----------|----------|--------------|-------------------|---------------|----|-------------|----------------|
|             |          |          |              | <i>Comp</i>       | <i>Result</i> |    |             |                |
| LD          | F6       | 34+      | R2           | 1                 | 3             | 4  | Load1       | No             |
| LD          | F2       | 45+      | R3           | 2                 | 4             | 5  | Load2       | No             |
| MULTD       | F0       | F2       | F4           | 3                 |               |    | Load3       | No             |
| SUBD        | F8       | F6       | F2           | 4                 | 7             | 8  |             |                |
| DIVD        | F10      | F0       | F6           | 5                 |               |    |             |                |
| ADDD        | F6       | F8       | F2           | 6                 | 10            | 11 |             |                |

### Reservation Stations:

| Time | Name  | <i>Busy</i> | <i>Op</i> | <i>S1</i> | <i>S2</i> | <i>RS</i> | <i>RS</i> |
|------|-------|-------------|-----------|-----------|-----------|-----------|-----------|
|      |       |             |           | <i>Vj</i> | <i>Vk</i> | <i>Qj</i> | <i>Qk</i> |
|      | Add1  | No          |           |           |           |           |           |
|      | Add2  | No          |           |           |           |           |           |
|      | Add3  | No          |           |           |           |           |           |
| 3    | Mult1 | Yes         | MULTD     | M(A2)     | R(F4)     |           |           |
|      | Mult2 | Yes         | DIVD      |           | M(A1)     | Mult1     |           |

### Register result status:

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

71

71

## Tomasulo Example Cycle 13

### Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Write</i> |               |    | <i>Busy</i> | <i>Address</i> |
|-------------|----------|----------|--------------|-------------------|---------------|----|-------------|----------------|
|             |          |          |              | <i>Comp</i>       | <i>Result</i> |    |             |                |
| LD          | F6       | 34+      | R2           | 1                 | 3             | 4  | Load1       | No             |
| LD          | F2       | 45+      | R3           | 2                 | 4             | 5  | Load2       | No             |
| MULTD       | F0       | F2       | F4           | 3                 |               |    | Load3       | No             |
| SUBD        | F8       | F6       | F2           | 4                 | 7             | 8  |             |                |
| DIVD        | F10      | F0       | F6           | 5                 |               |    |             |                |
| ADDD        | F6       | F8       | F2           | 6                 | 10            | 11 |             |                |

### Reservation Stations:

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

### Register result status:

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

72

72

## Tomasulo Example Cycle 14

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          |        |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8     |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          | 10     | 11    |       |       |      |         |

### Reservation Stations:

| Time | Name  | Busy | Op    | S1    |       | S2    |    | RS |  |
|------|-------|------|-------|-------|-------|-------|----|----|--|
|      |       |      |       | Vj    | Vk    | Qj    | 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    | Mult1 | M(A2) |    | (M-M+M (M-M)) | Mult2 |     |     |     |     |

73

73

## Tomasulo Example Cycle 15

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          | 15     |       |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8     |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          | 10     | 11    |       |       |      |         |

### Reservation Stations:

| Time | Name  | Busy | Op    | S1    |       | S2    |    | RS |  |
|------|-------|------|-------|-------|-------|-------|----|----|--|
|      |       |      |       | Vj    | Vk    | Qj    | 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    | Mult1 | M(A2) |    | (M-M+M (M-M)) | Mult2 |     |     |     |     |

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

74

74

## Tomasulo Example Cycle 16

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          | 15     | 16    |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8     |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          | 10     | 11    |       |       |      |         |

### Reservation Stations:

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

### Register result status:

|       |           |           |           |           |           |           |            |            |     |            |
|-------|-----------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| Clock |           | <i>F0</i> | <i>F2</i> | <i>F4</i> | <i>F6</i> | <i>F8</i> | <i>F10</i> | <i>F12</i> | ... | <i>F30</i> |
| 16    | <i>FU</i> | M*F4      | M(A2)     |           | (M-M+N    | (M-M)     | Mult2      |            |     |            |

Just waiting for Mult2 (DIVD) to complete

75

75

Faster than light computation  
(skip a couple of cycles...)

76

76

## Tomasulo Example Cycle 55

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          | 15     | 16    |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8     |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          |        |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          | 10     | 11    |       |       |      |         |

### Reservation Stations:

| Time | Name  | Busy | Op   | S1   | S2    | RS | RS |
|------|-------|------|------|------|-------|----|----|
|      |       |      |      | Vj   | Vk    | Qj | 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    | M*F4 | M(A2) |    | (M-M+M) | (M-M) | Mult2 |     |     |     |

77

77

## Tomasulo Example Cycle 56

### Instruction status:

| Instruction | j   | k   | Issue | Exec Write |        | Load1 | Load2 | Load3 | Busy | Address |
|-------------|-----|-----|-------|------------|--------|-------|-------|-------|------|---------|
|             |     |     |       | Comp       | Result |       |       |       |      |         |
| LD          | F6  | 34+ | R2    | 1          | 3      | 4     |       |       | No   |         |
| LD          | F2  | 45+ | R3    | 2          | 4      | 5     |       |       | No   |         |
| MULTD       | F0  | F2  | F4    | 3          | 15     | 16    |       |       | No   |         |
| SUBD        | F8  | F6  | F2    | 4          | 7      | 8     |       |       |      |         |
| DIVD        | F10 | F0  | F6    | 5          | 56     |       |       |       |      |         |
| ADDD        | F6  | F8  | F2    | 6          | 10     | 11    |       |       |      |         |

### Reservation Stations:

| Time | Name  | Busy | Op   | S1   | S2    | RS | RS |
|------|-------|------|------|------|-------|----|----|
|      |       |      |      | Vj   | Vk    | Qj | 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    | M*F4 | M(A2) |    | (M-M+M) | (M-M) | Mult2 |     |     |     |

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

78

78

## Tomasulo Example Cycle 57

### Instruction status:

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

### Reservation Stations:

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

### Register result status:

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

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

79

79

## Why can Tomasulo Overlap Iterations of Loops?

### •Register renaming

- Provided by reservation stations (there should be many of them)
- Multiple iterations use different physical destinations for registers (dynamic loop unrolling)
- Name dependences are eliminated by register renaming

### •Reservation stations

- Permit instruction issue to advance past integer control flow operations (provided branches predicted accurately)
- Also buffer old values of registers - totally avoiding the WAR stall

### •Other perspective: Tomasulo building data flow dependency graph on the fly!

80

80



## Tomasulo's Scheme offers 2 Major Advantages

### 1. 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

### 2. Elimination of stalls for WAW and WAR hazards

81

81

## Tomasulo Drawbacks

### ●Complexity

- Design delays of 360/91, MIPS 10000, Alpha 21264, IBM PPC 620

### ●Performance limited by Common Data Bus

- Each CDB must go to multiple functional units
- $\Rightarrow$  high capacitance (think long delays), high wiring density
- Number of functional units that can complete per cycle limited to one!
  - Multiple CDBs  $\Rightarrow$  more FU logic for parallel stores

82

82

## Conclusion

- Leverage Implicit Parallelism for Performance
  - Instruction Level Parallelism (ILP)
- Loop unrolling by compiler to increase amount of ILP
- Branch prediction to increase amount of ILP
- HW Dynamic Scheduling
  - Works when cannot know dependence at compile time
  - Can hide L1 cache misses
  - Code for one machine runs well on another

83

83

## Conclusion

- Reservations stations: renaming to larger set of registers + buffering source operands
  - Prevents registers as bottleneck
  - Avoids WAR, WAW hazards
  - Allows loop unrolling in HW (provided branches predicted accurately)
- Not limited to basic blocks  
(integer units gets ahead, beyond branches)
- Helps cache misses as well
- Lasting Contributions
  - Dynamic scheduling
  - Register renaming
  - Load/store disambiguation
- 360/91 descendants are Intel Pentium 4, IBM Power 5, AMD Athlon/Opteron, Intel i3/i5/i7, ...

84

84