Lecture 3 Instruction level/thread level parallelism

Spring 2020

Why study this?

➤ Typical edge Al accelerator uses a pipelined processor to run the sequence control and the "non-accelerated functions" required by the model, or the part that is not implemented in the accelerator.

Amdahal's law

$$Execution \ time_{new} = Execution \ time_{old} \times \left((1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right)$$

Architecture of SIMT GPU rooted in conventional RISC pipelined processor

Parallelism from the view point of application and architecture

- Parallelism in applications
 - ▶ Data-Level Parallelism (DLP)
 - ► Task-Level Parallelism (TLP)
- Architectural parallelism
 - ► Instruction-Level Parallelism (ILP)
 - Vector architectures/Graphic Processor Units (GPUs)
 - ► Thread-Level Parallelism
 - ► Request-Level Parallelism

Current Trends in Architecture

- Cannot continue to leverage Instruction-Level parallelism (ILP)
 - ▶ ILP by branch prediction, pipelined operation, VLIW
 - ► Single processor performance improvement ended in 2003
 - ▶ Deep pipeline, OOO, superscalar
- ► New models for performance:
 - ▶ Data-level parallelism (DLP): SIMT, SIMD, Vector
 - ► Thread-level parallelism (TLP): MIMD, Multi-core, Simultaneous multithreading (Hyper threading)
 - ▶ Domain specific computing: DLA, NNP,....

Outline

- ILP
- Compiler techniques to increase ILP
- Loop Unrolling
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- (Start) Tomasulo Algorithm
- Conclusion

Instruction Level Parallelism

- ▶ Pipelining become universal technique in 1985
 - Overlaps execution of instructions
 - Exploits "Instruction Level Parallelism"
- ▶ Beyond this, there are two main approaches:
 - ► Hardware-based dynamic approaches
 - Used in server and desktop processors
 - Superscalar processor execution model
 - Superscalar 000 processor execution model
 - Compiler-based static approaches (VLIW)
 - ▶ Not as successful outside of scientific applications
 - Movidius Compute Stick USB Al accelerator

Recall from Pipelining Review

- ► Pipeline CPI = Ideal pipeline CPI + Structural Stalls
 - + Data Hazard Stalls + Control Stalls
 - ► Ideal pipeline CPI: measure of the maximum performance attainable by the implementation
 - <u>Structural hazards</u>: 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)

7

Instruction Level Parallelism

- Instruction-Level Parallelism (ILP): overlap the execution of instructions to improve performance
- 2 approaches to exploit ILP:
 - Rely on hardware to help discover and exploit the parallelism dynamically (e.g., Pentium 4, AMD Opteron, IBM Power, Intel Icore xxx,), and
 - 2) Rely on software technology to find parallelism, statically at compile-time (e.g., Itanium 2,)
- On this topic spend our time will.

Instruction-Level Parallelism (ILP)

- Basic Block (BB) 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: loop-level parallelism to exploit parallelism among iterations of a loop. E.g.,

for
$$(i=1; i <= 1000; i=i+1)$$

 $x[i] = x[i] + y[i];$

Loop-Level Parallelism

- Exploit loop-level parallelism to parallelism by "unrolling loop" either by
- 1. dynamic via branch prediction or (Most found in current processors)
- static via loop unrolling by compiler (used in CASLab GPU OpenCL Compiler)(Another way is vectors)
- Determining instruction dependence is critical to Loop Level Parallelism
- If 2 instructions are
 - <u>parallel</u>, 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

Data Dependence and Hazards

- Instr_J is data dependent (aka true dependence) on Instr_{I:} (aka: also known as)
 - 1. Instrutries to read operand before Instruwrites it

```
I: add r1,r2,r3
J: sub r4,r1,r3
```

- or Instr_J is data dependent on Instr_K which is dependent on Instr_I
- 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
- If data dependence caused a hazard in pipeline, called a Read After Write (RAW) hazard

ILP and Data Dependencies, Hazards

- HW/SW must preserve program order: instructions would execute in order 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 dependencies
 - 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
- ► HW/SW goal: exploit parallelism by preserving program order only where it affects the outcome of the program

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_J writes operand <u>before</u> Instr_I 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

Name Dependence #2: Output dependence

Instr_J writes operand <u>before</u> Instr_I 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 caused a hazard in the pipeline, called a Write After Write (WAW) hazard
- 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

Control Dependencies

Every instruction is control dependent on some set of branches, and, in general, these control dependencies 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.

Control Dependence Ignored

- Control dependence needs 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 and
 - 2) data flow

Exception Behavior

- Preserving exception behavior
 - ⇒ any changes in instruction execution order must not change how exceptions are raised in program
 - $(\Rightarrow$ no new exceptions)
- Example:

```
DADDU R2,R3,R4
BEQZ R2,L1
LW R1,0(R2)
```

- (Assume branches not delayed)
- Problem with moving LW before BEQZ?

Data Flow

- Data flow: actual flow of data values among instructions that produce results and those that consume them
 - branches make flow dynamic, determine which instruction is supplier of data
- Example:

```
DADDU R1,R2,R3
BEQZR4,L
DSUBU R1,R5,R6
L: ...
OR R7,R1,R8
```

OR depends on DADDU or DSUBU?
Must preserve data flow on execution

Outline

- **ILP**
- Compiler techniques to increase ILP
- Loop Unrolling
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- (Start) Tomasulo Algorithm
- Conclusion

Software Techniques - Example

► This code, add 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

Instruction producing result	Instruction using result	Latency in cycles i	stalls between n cycles
FP ALU op	Another FP ALU op	4	3
FP ALU op	Store double	3	2
Load double	FP ALU op	1	1
Load double	Store double	1	0
Integer op	Integer op	1	0

FP Loop: Where are the Hazards?

- First translate into MIPS code:
 - -To simplify, assume 8 is lowest address

```
Loop: L.D F0,0(R1); F0=vector element

ADD.D F4,F0,F2; add scalar from F2

S.D 0(R1),F4; store result

DADDUI R1,R1,-8; decrement pointer 8B (DW)

BNEZ R1,Loop; branch R1!=zero
```

FP Loop Showing Stalls

```
1 Loop: L.D F0,0(R1); F0=vector element
2
        stall
        ADD.D F4,F0,F2 ; add scalar in F2
        stall
5
        stall
6
        S.D
               0(R1),F4 ;store result
        DADDUI R1, R1, -8 ; decrement pointer 8B (DW)
        stall
                          ;assumes can't forward to branch
9
        BNEZ
               R1,Loop ;branch R1!=zero
Instruction
                 Instruction
                                     Latency in
producing result
                 using result
                                     clock cycles
FP ALU op
                 Another FP ALU op
FP ALU op
                 Store double
Load double
                 FP ALU op
```

9 clock cycles: Rewrite code to minimize stalls?

Revised FP Loop Minimizing Stalls

```
1 Loop: L.D F0,0(R1)
2     DADDUI R1,R1,-8
3     ADD.D F4,F0,F2
4     stall
5     stall
6     S.D 8(R1),F4 ;altered offset when move DSUBUI
7     BNEZ R1,Loop
```

Swap DADDUI and S.D by changing address of S.D

Instruction producing result	Instruction using result	Latency in clock cycles
FP ALU op	Another FP ALU op	3
FP ALU op	Store double	2
Load double	FP ALU op	1

7 clock cycles, but just 3 for execution (L.D, ADD.D,S.D), 4 for loop overhead; How make faster?

Unroll Loop Four Times (straightforward way)

```
1 cycle stall
                                            Rewrite loop to minimize
              F0,0(R1)
  Loop:L.D
                                              stalls?
                             2 cycles stall
             F4,F0,F2
       ADD.D
6
       S.D
              0(R1),F4
                            ;drop DSUBUI & BNEZ
             F6, -8(R1)
       L.D
9
       ADD.D F8, F6, F2
12
       S.D
             -8(R1), F8
                            ;drop DSUBUI & BNEZ
13
       L.D F10, -16(R1)
       ADD.D F12,F10,F2
15
       S.D
           -16(R1),F12
18
                            ;drop DSUBUI & BNEZ
19
       L.D F14, -24(R1)
21
       ADD.D F16,F14,F2
24
       S.D
              -24(R1), F16
25
       DADDUI R1, R1, #-32
                           ;alter to 4*8
26
       BNEZ
              R1,LOOP
```

2020/4/7

27 clock cycles, or 6.75 per iteration (Assumes R1 is multiple of 4)

Unrolled Loop Detail

- Do not usually know upper bound of loop
- Suppose it is n, and we would like to unroll the loop to make k copies of the body
- Instead of a single unrolled loop, we generate a pair of consecutive loops:
 - 1st executes (n mod k) times and has a body that is the original loop
 - ▶ 2nd is the unrolled body surrounded by an outer loop that iterates (n/k) times
- For large values of n, most of the execution time will be spent in the unrolled loop

Unrolled Loop That Minimizes Stalls

```
F0,0(R1)
1 Loop:L.D
2
           F6,-8(R1)
       L.D
       L.D
           F10,-16(R1)
4
      L.D F14,-24(R1)
5
       ADD.D F4,F0,F2
6
             F8, F6, F2
       ADD.D
       ADD.D F12,F10,F2
8
             F16,F14,F2
       ADD.D
9
       S.D 0(R1), F4
10
       S.D
           -8(R1),F8
11
       S.D
             -16(R1),F12
12
      DSUBUI R1,R1,#32
             8(R1), F16 ; 8-32 = -24
13
       S.D
       BNEZ
             R1,LOOP
14
```

14 clock cycles, or 3.5 per iteration

5 Loop Unrolling Decisions

- Requires understanding how one instruction depends on another and how the instructions can be changed or reordered given the dependences:
- 1. Determine loop unrolling useful by finding that loop iterations were independent (except for maintenance code)
- 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

3 Limits to Loop Unrolling

- Decrease in amount of overhead amortized with each extra unrolling
 - Amdahl's Law
- Growth in code size
 - For larger loops, concern it increases the instruction cache miss rate
- 3. <u>Register pressure</u>: 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

Dynamic Branch Prediction

- Why does prediction work?
 - Underlying algorithm has regularities
 - Data that are being operated on have regularities
 - Instruction sequence has redundancies that are artifacts of way that humans/compilers think about problems
- Is dynamic branch prediction better than static branch prediction?
 - Seems to be
 - There are a small number of important branches in programs which have dynamic behavior

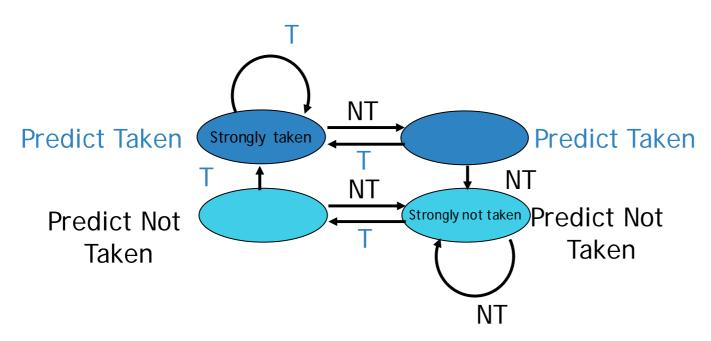
Dynamic Branch Prediction

- Performance = f(accuracy, cost of misprediction)
- Branch History Table: Lower bits of PC address index table of 1-bit values
 - Says whether or not branch taken last time
 - No address check
- Problem: in a loop, 1-bit BHT will cause two mispredictions):

Dynamic Branch Prediction

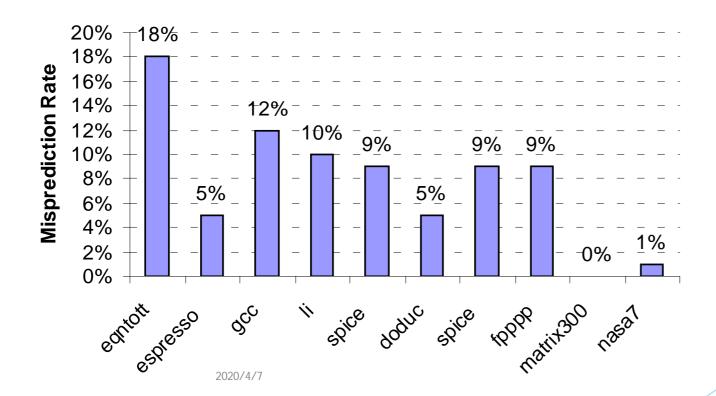
- Solution: 2-bit scheme where change prediction only if gets misprediction *twice*
- Adds hysteresis to decision making process

31



BHT Accuracy

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



Correlated Branch Prediction

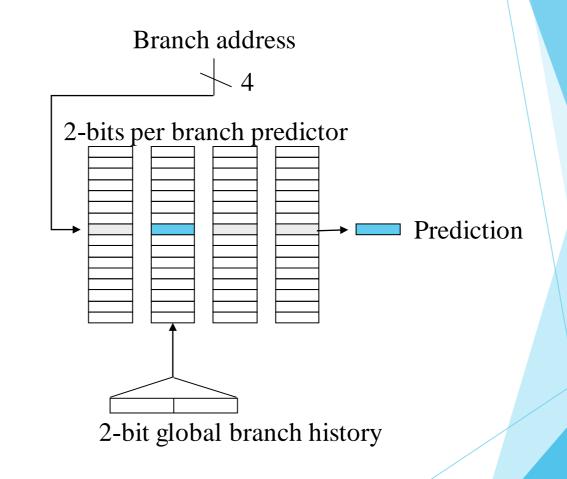
- Idea: record m most recently executed branches as taken or not taken, and use that pattern to select the proper n-bit branch history table
- In general, (m,n) predictor means record last m branches to select between 2^m history tables, each with n-bit counters
 - ▶ Thus, old 2-bit BHT is a (0,2) predictor
- Global Branch History: m-bit shift register keeping T/NT status of last m branches.

33

Correlating Branches

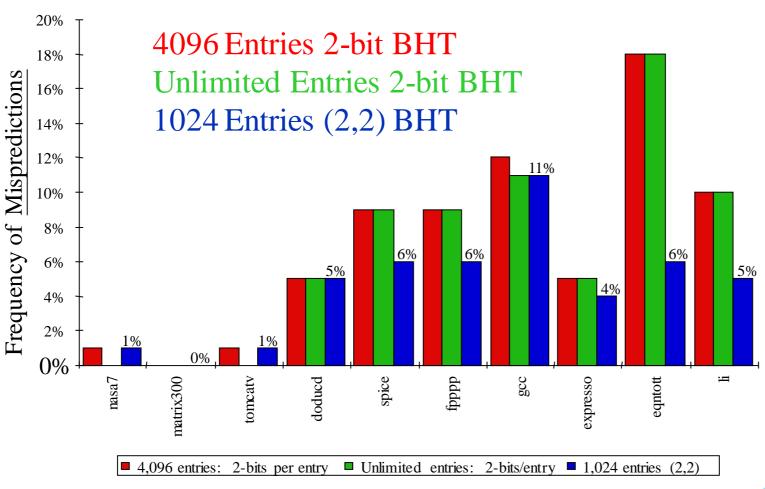
(2,2) predictor

 Behavior of recent branches selects between four predictions of next branch, updating just that prediction



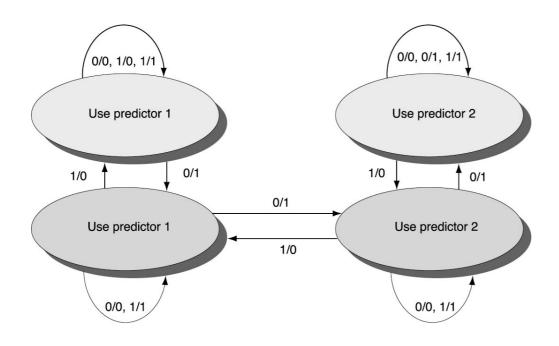
34

Accuracy of Different Schemes



Tournament Predictors

- Multilevel branch predictor
- Use n-bit saturating counter to choose between predictors
- Usual choice between global and local predictors

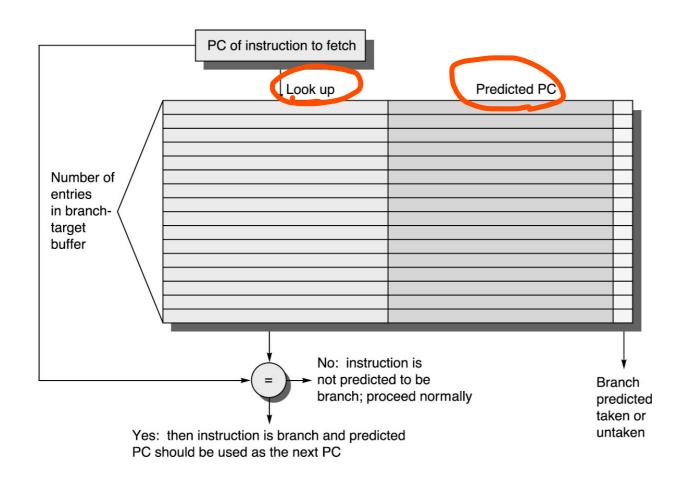


0/0 both are wrong1/0 P1 correct, P2 wrong1/1 both are correct0/1 P1 wrong, P2 correct

Branch Target Buffers (BTB)

- Branch target calculation is costly and stalls the instruction fetch.
- BTB stores PCs the same way as caches
- The PC of a branch is sent to the BTB
- When a match is found the corresponding Predicted PC is returned
- If the branch was predicted taken, instruction fetch continues at the returned predicted PC

Branch Target Buffers



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 (GA)
 - Or different executions of same branches (PA), Per-Address
- 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
- Branch Target Buffer: include branch address & prediction

39

Outline

- **ILP**
- Compiler techniques to increase ILP
- Loop Unrolling
- Static Branch Prediction
- Dynamic Branch Prediction
- Overcoming Data Hazards with Dynamic Scheduling
- (Start) Tomasulo Algorithm
- Conclusion

Advantages of Dynamic Scheduling

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

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*
- Note: Dynamic execution creates WAR and WAW hazards and makes exceptions harder

Dynamic Scheduling Step 1

- Simple pipeline had 1 stage to check both structural and data hazards: Instruction Decode (ID), also called Instruction Issue
- 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

43

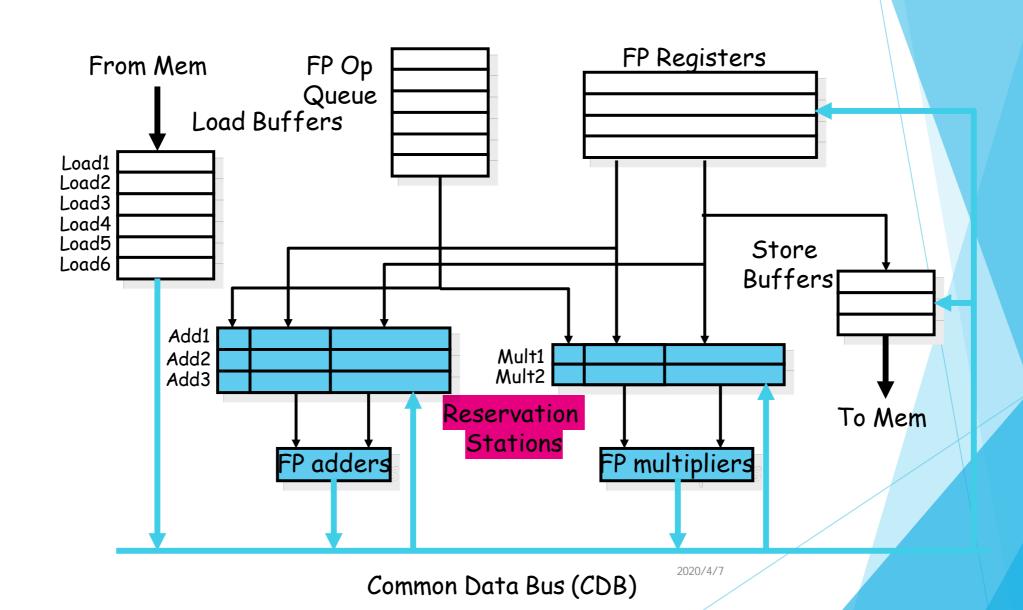
A Dynamic Algorithm: Tomasulo's

- 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, ... 2020's

Tomasulo Algorithm

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

Tomasulo Organization



Reservation Station Components

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

Vj, Vk: Value of Source operands

Store buffers has V field, result to be stored

Oj, Ok: Reservation stations producing source registers (value to be written)

Note: Qj,Qk=0 => ready

Store buffers only have Qj for RS producing result

Busy: Indicates reservation station or FU is busy

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

Three Stages of Tomasulo Algorithm

1. Issue—get instruction from FP Op Queue

If reservation station free (no structural hazard), control issues instr & 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

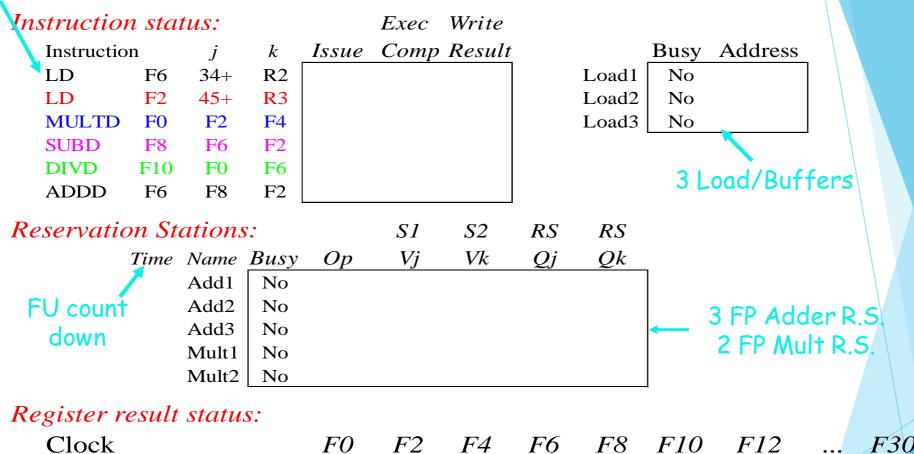
3. Write result—finish execution (WB)

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

- Normal data bus: data + destination ("go to" bus)
- Common data bus: data + source ("come from" bus)
 - ▶ 64 bits of data + 4 bits of Functional Unit source address
 - Write if matches expected Functional Unit (produces result)
 - Does the broadcast
- Example speed: 3 clocks for Fl .pt. +,-; 11 for *; 41 clks for /

Tomasulo Example

Instruction stream



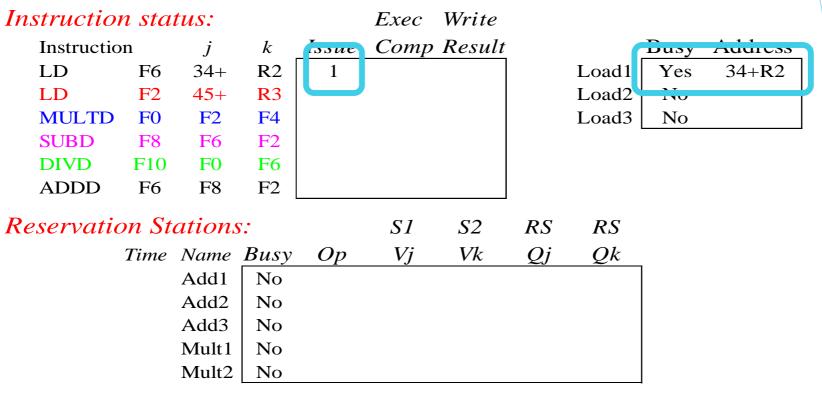
Clock

FU

F2 FO*F4* F8

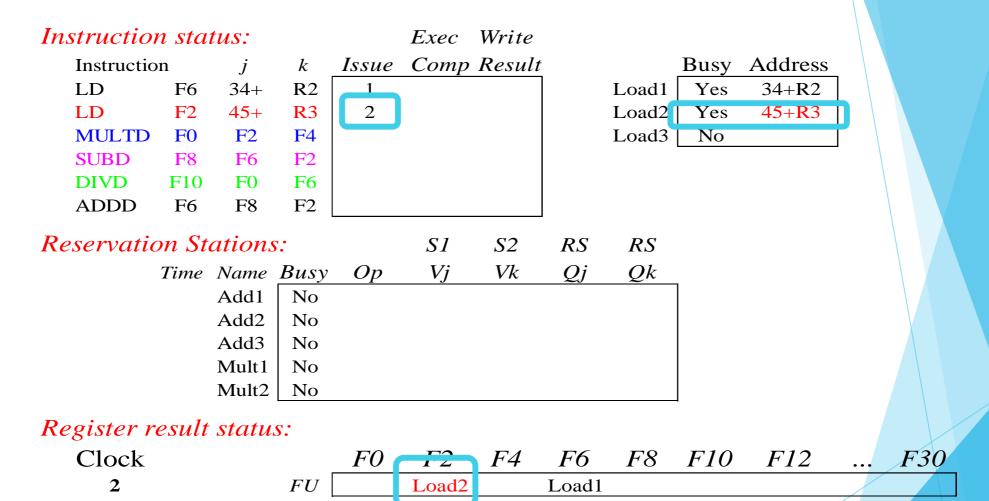
F10*F12* F30

Clock cycle counter

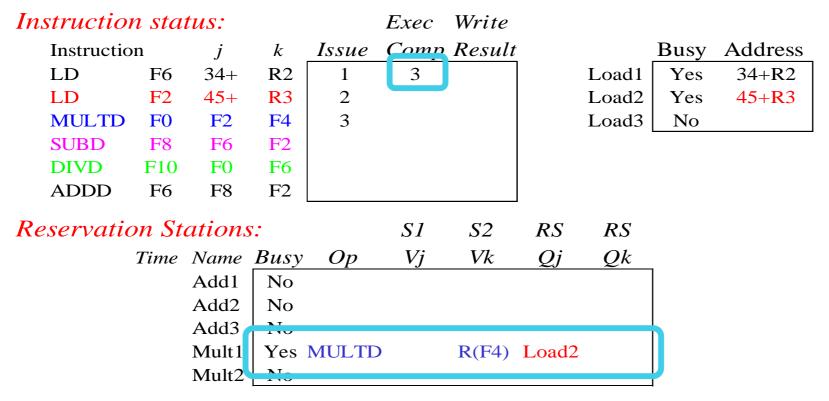


Register result status:



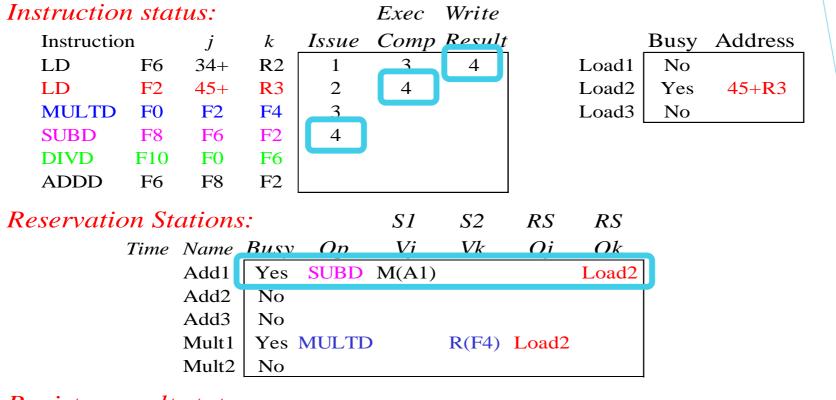


Note: Can have multiple loads outstanding



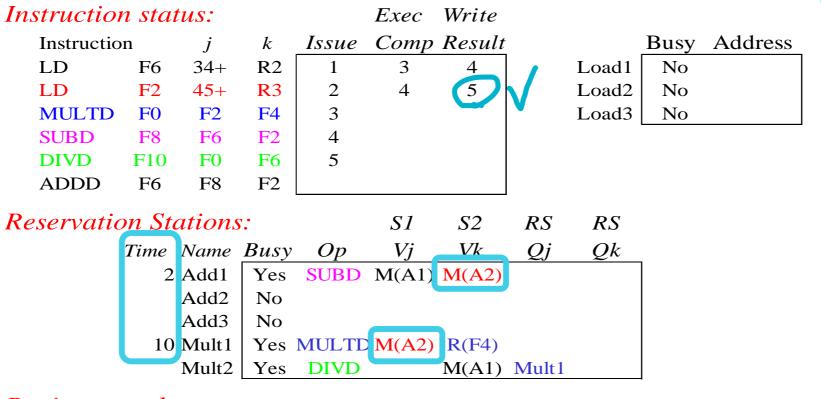
Register result status:

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



Register result status:

Load2 completing; what is waiting for Load2?



Register result status:

Timer starts down for Add1, Mult1

```
Instruction status:
                                 Exec Write
                                 Comp Result
                                                         Busy Address
   Instruction
                          Issue
                       k
   LD
                34 +
                      R2
            F6
                                   3
                                                   Load1
                                                          No
                45+
   LD
                      R3
            F2
                                         5
                                                   Load2
                                                          No
   MULTD
           FO
                 F2
                      F4
                                                   Load3
                                                          No
   SUBD
                 F6
                      F2
           F8
   DIVD
                 F0
           F10
                      F6
   ADDD
           F6
                 F8
                      F2
Reservation Stations:
                                  S1
                                        S2
                                              RS
                                                    RS
          Time Name Busy
                                  Vi
                                        Vk
                                              Q_{i}
                                                    Qk
                           Op
              1 Add1
                     Yes SURD M(A1) M(A2)
               Add2
                      Yes ADDD
                                       M(A2) Add1
               Add3
                      No
                      Yes MULTD M(A2) R(F4)
              9 Mult1
               Mult2
                      Yes DIVD
                                       M(A1) Mult1
Register result status:
                                                    F8
                                                         F10
                                                                            F30
   Clock
                            F0
                                  F2
                                        F4
                                              F6
                                                                F12
```

Issue ADDD here despite name dependency on F6?

M(A2)

Add2

Add1

Mult2

Mult1

6

```
Instruction status:
                                 Exec Write
                                 Comp Result
                                                          Busy Address
   Instruction
                           Issue
                       k
   LD
                 34 +
                      R2
            F6
                                    3
                                          4
                                                    Load1
                                                            No
   LD
                 45+
                       R3
            F2
                                          5
                                                    Load2
                                                            No
   MULTD
            FO
                 F2
                      F4
                                                    Load3
                                                           No
   SUBD
                 F6
                      F2
            F8
                             4
   DIVD
                 F0
           F10
                      F6
   ADDD
            F6
                 F8
                      F2
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                     RS
                                   Vi
                                         Vk
                                               Q_{i}
                                                     Ok
           Time Name Busy
                            Op
              0 Add1
                      Yes SUBD M(A1) M(A2)
                Add2
                      Yes ADDD
                                       M(A2) Add1
                Add3
                      No
                      Yes MULTD M(A2) R(F4)
              8 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

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

```
Instruction status:
                                 Exec Write
                                 Comp Result
                                                          Busy Address
   Instruction
                       k
                           Issue
   LD
                 34 +
                       R2
                                    3
                                                            No
            F6
                                          4
                                                    Load1
                 45+
   LD
                       R3
                                    4
                                                    Load2
            F2
                                          5
                                                            No
   MULTD
            FO
                 F2
                       F4
                                                    Load3
                                                            No
   SUBD
                                    7
                 F6
                      F2
                             4
                                          8
            F8
   DIVD
                 F0
                       F6
           F10
   ADDD
            F6
                 F8
                       F2
                             6
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                     RS
                                   V_{j}
                                         Vk
           Time Name Busy
                                               Qj
                                                     Qk
                            Op
                Add1
                      No
                      Yes ADDD (M-M) M(A2)
              2 Add2
                Add3
                      No
                      Yes MULTD M(A2) R(F4)
              7 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

57

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

8 FU Mult1 M(A2) Add2 (M-M) Mult2 ... F30

```
Instruction status:
                                 Exec Write
                                 Comp Result
                                                          Busy Address
   Instruction
                       k
                           Issue
   LD
                 34 +
                       R2
                                    3
                                                            No
            F6
                                          4
                                                    Load1
                 45+
   LD
                       R3
                                    4
                                                    Load2
            F2
                                          5
                                                            No
   MULTD
            FO
                 F2
                       F4
                                                    Load3
                                                            No
   SUBD
                                    7
                 F6
                      F2
                             4
                                          8
            F8
   DIVD
                 F0
                       F6
           F10
   ADDD
           F6
                 F8
                       F2
                             6
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                     RS
                                   V_{j}
                                         Vk
           Time Name Busy
                                               Qj
                                                     Qk
                            Op
                Add1
                      No
                      Yes ADDD (M-M) M(A2)
              1 Add2
                Add3
                      No
                      Yes MULTD M(A2) R(F4)
              6 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

58

```
Instruction status:
                                  Exec Write
                                 Comp Result
                                                           Busy Address
   Instruction
                           Issue
                       k
   LD
                 34 +
                       R2
            F6
                                    3
                                                    Load1
                                                            No
                 45+
   LD
                       R3
            F2
                                          5
                                                    Load2
                                                            No
   MULTD
            FO
                 F2
                       F4
                                                    Load3
                                                            No
                                    7
   SUBD
                 F6
                       F2
                                          8
            F8
   DIVD
                 F0
           F10
                       F6
   ADDD
            F6
                 F8
                       F2
                                    10
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                      RS
                                   Vi
                                         Vk
                                                Q_{i}
                                                      Qk
           Time Name Busy
                            Op
                Add1
                       No
                      Yes ADDD (M-M) M(A2)
              0 Add2
                Add3
                       No
                       Yes MULTD M(A2) R(F4)
              5 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

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

```
Instruction status:
                                        Write
                                  Exec
                                  Comp Result
                                                            Busy Address
   Instruction
                            Issue
                        k
   LD
                 34 +
                       R2
                                                             No
            F6
                                     3
                                                     Load1
                 45+
   LD
                       R3
            F2
                                           5
                                                     Load2
                                                             No
   MULTD
            FO
                  F2
                       F4
                                                     Load3
                                                             No
                                    7
   SUBD
                  F6
                       F2
                                           8
            F8
   DIVD
                  F0
            F10
                       F6
   ADDD
            F6
                  F8
                       F2
                                    10
                                          11
Reservation Stations:
                                    S1
                                          S2
                                                RS
                                                      RS
                                    V_{j}
                                          Vk
                                                Qj
                                                      Qk
           Time Name Busy
                             Op
                Add1
                       No
                Add2
                       No
                Add3
                       No
                       Yes MULTD M(A2) R(F4)
               4 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

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

```
Instruction status:
                                  Exec Write
                                  Comp Result
                                                           Busy Address
   Instruction
                       k
                           Issue
   LD
                 34 +
                       R2
                                    3
                                                            No
            F6
                                          4
                                                    Load1
                 45+
   LD
                       R3
                                    4
                                                    Load2
            F2
                                          5
                                                            No
   MULTD
            FO
                 F2
                       F4
                                                    Load3
                                                            No
   SUBD
                                    7
                 F6
                       F2
                             4
                                          8
            F8
   DIVD
                 F0
                       F6
           F10
   ADDD
            F6
                 F8
                       F2
                                    10
                                          11
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                      RS
                                    V_{j}
                                          Vk
           Time Name Busy
                                                Qj
                                                      Qk
                            Op
                Add1
                       No
                Add2
                       No
                Add3
                       No
                       Yes MULTD M(A2) R(F4)
              3 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

61

Clock

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

FU Mult1 M(A2) (M-M+M(M-M) Mult2

```
Instruction status:
                                  Exec Write
                                  Comp Result
                                                           Busy Address
   Instruction
                       k
                           Issue
   LD
                 34 +
                       R2
                                    3
                                                            No
            F6
                                          4
                                                    Load1
                 45+
   LD
                       R3
                                    4
                                                    Load2
            F2
                                          5
                                                            No
   MULTD
            FO
                 F2
                       F4
                                                    Load3
                                                            No
   SUBD
                                    7
            F8
                 F6
                       F2
                             4
                                          8
   DIVD
                 F0
                       F6
           F10
   ADDD
            F6
                 F8
                       F2
                                    10
                                          11
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                      RS
                                    V_{j}
                                          Vk
           Time Name Busy
                                                Qj
                                                      Qk
                            Op
                Add1
                       No
                Add2
                       No
                Add3
                       No
                       Yes MULTD M(A2) R(F4)
              2 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

```
Instruction status:
                                  Exec Write
                                  Comp Result
                                                           Busy Address
   Instruction
                       k
                           Issue
   LD
                 34 +
                       R2
                                    3
                                                            No
            F6
                                          4
                                                    Load1
                 45+
   LD
                       R3
                                    4
                                                    Load2
            F2
                                          5
                                                            No
   MULTD
            FO
                 F2
                       F4
                                                    Load3
                                                            No
   SUBD
                                    7
                 F6
                       F2
                             4
                                          8
            F8
   DIVD
                 F0
                       F6
           F10
   ADDD
            F6
                 F8
                       F2
                                    10
                                          11
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                      RS
                                    V_{j}
                                          Vk
           Time Name Busy
                                                Qj
                                                      Qk
                            Op
                Add1
                       No
                Add2
                       No
                Add3
                       No
              1 Mult1
                       Yes MULTD M(A2) R(F4)
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

63

Clock

14

F0
F2
F4
F6
F8
F10
F12
...
F30

W-M+N (M-M)
Mult2

```
Instruction status:
                                  Exec Write
                                  Comp Result
                                                           Busy Address
   Instruction
                            Issue
                        k
   LD
                 34 +
                       R2
            F6
                                    3
                                           4
                                                     Load1
                                                             No
                 45+
   LD
                       R3
            F2
                                           5
                                                     Load2
                                                             No
   MULTD
            FO
                  F2
                       F4
                                    15
                                                     Load3
                                                             No
   SUBD
                  F6
                       F2
                                           8
            F8
   DIVD
                  F0
            F10
                       F6
   ADDD
            F6
                  F8
                       F2
                                    10
                                          11
Reservation Stations:
                                   S1
                                          S2
                                                RS
                                                      RS
                                    V_{i}
                                          Vk
                                                Qj
                                                      Qk
           Time Name Busy
                             Op
                Add1
                       No
                Add2
                       No
                Add3
                       No
                       Yes MULTD M(A2) R(F4)
              0 Mult1
                Mult2
                      Yes DIVD
                                        M(A1) Mult1
```

Register result status:

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

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

```
Instruction status:
                                       Write
                                 Exec
                                 Comp Result
                                                          Busy Address
   Instruction
                           Issue
                       k
   LD
                 34 +
                      R2
                                                           No
            F6
                                    3
                                          4
                                                    Load1
                 45+
   LD
                       R3
            F2
                                                    Load2
                                                           No
   MULTD
            FO
                 F2
                       F4
                                   15
                                                    Load3
                                                           No
                                         16
   SUBD
                 F6
                      F2
            F8
   DIVD
                 F0
           F10
                       F6
   ADDD
            F6
                 F8
                       F2
                                   10
                                         11
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                     RS
                                   V_{j}
                                         Vk
                                               Qj
                                                     Qk
           Time Name Busy
                            Op
                Add1
                      No
                Add2
                      No
                Add3
                      No
                Mult1
                      No
                           DIVD M*F4 M(A1)
             40 Mult2
                      Yes
Register result status:
                                                     F8
                                                          F10
                                                                              F30
   Clock
                            F0
                                  F2
                                         F4
                                               F6
                                                                  F12
```

Just waiting for Mult2 (DIVD) to complete.

M(A2)

(M-M+N.(M-M))

Mult2

16

Faster than light computation (skip a couple of cycles)

```
Instruction status:
                                 Exec Write
                                 Comp Result
                                                           Busy Address
   Instruction
                       k
                           Issue
   LD
                 34 +
                       R2
                                    3
                                                            No
            F6
                                          4
                                                    Load1
                 45+
   LD
                       R3
                                                    Load2
            F2
                                          5
                                                            No
   MULTD
            FO
                 F2
                       F4
                                   15
                                                    Load3
                                                            No
                                          16
   SUBD
            F8
                 F6
                       F2
                                          8
   DIVD
                 F0
                       F6
           F10
   ADDD
            F6
                 F8
                       F2
                                   10
                                          11
Reservation Stations:
                                   S1
                                         S2
                                               RS
                                                     RS
                                   V_{j}
                                         Vk
           Time Name Busy
                                                Qj
                                                      Qk
                            Op
                Add1
                       No
                Add2
                       No
                Add3
                       No
                Mult1
                       No
              1 Mult2
                      Yes
                           DIVD
                                 M*F4 M(A1)
```

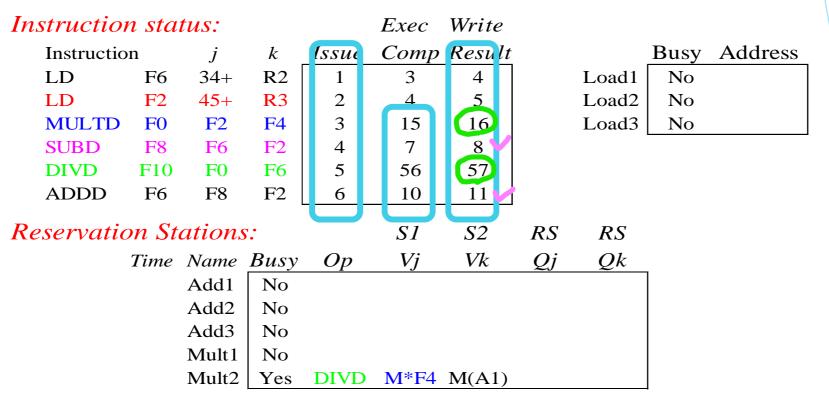
Register result status:

67

```
Instruction status:
                                  Exec Write
                                  Comp Result
                                                           Busy Address
   Instruction
                            Issue
                        k
   LD
                 34 +
                       R2
            F6
                                     3
                                           4
                                                     Load1
                                                             No
                 45+
   LD
                       R3
            F2
                                                     Load2
                                                             No
   MULTD
            FO
                  F2
                       F4
                                    15
                                                     Load3
                                          16
                                                             No
   SUBD
                  F6
                       F2
            F8
   DIVD
                                    56
            F10
                  FO
                       F6
   ADDD
            F6
                  F8
                       F2
                                    10
                                          11
Reservation Stations:
                                   S1
                                          S2
                                                RS
                                                      RS
                                    V_{i}
                                          Vk
                                                Qj
                                                      Qk
           Time Name Busy
                             Op
                Add1
                       No
                Add2
                       No
                Add3
                       No
                Mult1
                       No
              0 Mult2
                      Yes
                           DIVD
                                  M*F4 M(A1)
```

Register result status:

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



Register result status:

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

Why can Tomasulo overlap iterations of loops?

- Register renaming
 - Multiple iterations use different physical destinations for registers (dynamic loop unrolling).
- Reservation stations
 - Permit instruction issue to advance past integer control flow operations
 - Also buffer old values of registers totally avoiding the WAR stall
- Other perspective: Tomasulo building data flow dependency graph on the fly

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

Tomasulo Drawbacks

- Complexity
 - delays of 360/91, MIPS 10000, Alpha 21264!
- Many associative stores (CDB) at high speed
- Performance limited by Common Data Bus
 - ► Each CDB must go to multiple functional units ⇒high capacitance, high wiring density
 - Number of functional units that can complete per cycle limited to one!
 - ► Multiple CDBs ⇒ more FU logic for parallel assoc stores
- Non-precise interrupts!
 - ▶ We will address this later

And In Conclusion ... #1

- ► Leverage Implicit Parallelism for Performance: Instruction Level Parallelism
- Loop unrolling by compiler to increase ILP
- Branch prediction to increase ILP
- Dynamic HW exploiting ILP
 - Works when can't know dependence at compile time
 - Can hide L1 cache misses
 - Code for one machine runs well on another

73

And In Conclusion ... #2

- Reservations stations: renaming to larger set of registers + buffering source operands
 - Prevents registers as bottleneck
 - Avoids WAR, WAW hazards
 - Allows loop unrolling in HW
- 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, ...

74