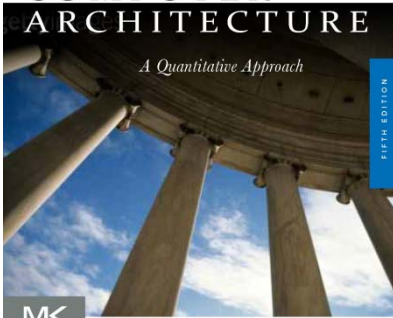


JOHN L. HENNESSY DAVID A. PATTERSON

**COMPUTER
ARCHITECTURE**

A Quantitative Approach

FIFTH EDITION




Chapter 3

Instruction-Level Parallelism and Its Exploitation

Static vs. Dynamic Scheduling

- Dynamic Scheduling
 - Fast
 - Requires complex hardware
 - More power consumption
 - May result in a slower clock
- Static Scheduling
 - Done in S/W (compiler)
 - Maybe not as fast
 - Simpler processor design (less complex)

Dynamic Scheduling

- In Simple pipelines, instructions are issued in order.
- If an instruction stalls, all instructions after it are stalled too (could be O.K. to execute).
- DIV.D F0,F2,F4
- ADD.D F10,F0,F8
- SUB.D F12,F8,F14 

Dynamic Scheduling

- Rearrange order of instructions to reduce stalls while maintaining data flow
- Advantages:
 - Compiler doesn't need to have knowledge of microarchitecture
 - Handles cases where dependencies are unknown at compile time
- Disadvantage:
 - Substantial increase in hardware complexity
 - Complicates exceptions

Dynamic Scheduling

- Rearrange order of instructions to reduce stalls while maintaining data flow
- Instructions are issued in program order
- But, the instruction begins execution as soon as its operand are ready
- Out of order execution → out of order completion
- DIV.D F0,F2,F4
- ADD.D F6,F0,F8 Antidependence
- SUB.D F8,F10,F14 Output Dependence
- MUL.D F6,F10,F8

Dynamic Scheduling

- To allow dynamic scheduling, split the ID stage in the simple MIPS pipeline into 2 stages
 - Issue: Decode and check for structural hazards
 - Read operand: wait for data hazard → read operand
- Instruction fetch stage before issue, and execution starts after read operand.
- Instructions pass through the issue stage in order, they can be delayed or pass each other at the read operand stage.

Dynamic Scheduling

- Major complication for exception handling.
- Must preserve the exception behavior as if the instructions are executed in the program order.
- May delay notification until the processor knows the instruction is the next one completed.
- Imprecise exception may occur
 - Later instructions (in program order) may have been completed already.
 - Earlier instructions may have not been completed

Register Renaming

- Example:

DIV.D F0,F2,F4

ADD.D F6,F0,F8

S.D F6,0(R1)

SUB.D F8,F10,F14

MUL.D F6,F10,F8

antidependence



antidependence

+ name dependence with F6

Register Renaming

- Example:

DIV.D F0,F2,F4
ADD.D F6,F0,F8
S.D F6,0(R1)
SUB.D F8,F10,F14
MUL.D F6,F10,F8

DIV.D F0,F2,F4
ADD.D S,F0,F8
S.D S,0(R1)
SUB.D T,F10,F14
MUL.D F6,F10,T

- Now only RAW hazards remain, which can be strictly ordered

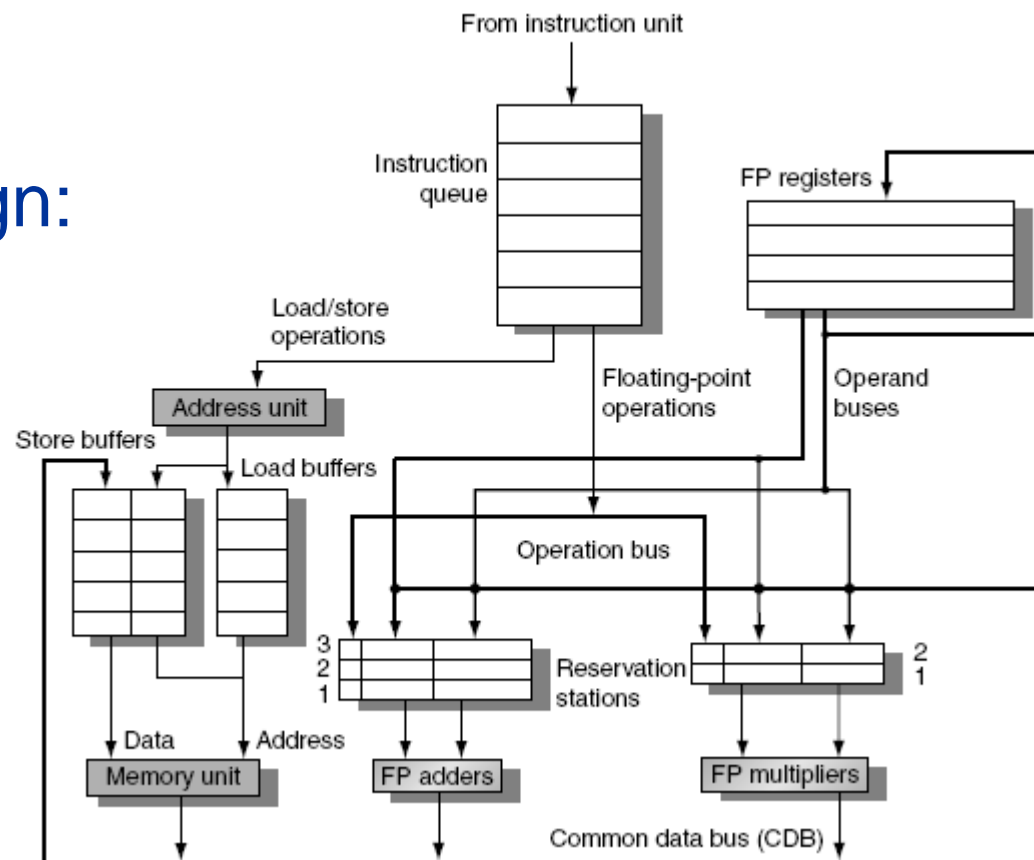
Register Renaming

- Register renaming is provided by reservation stations (RS)
 - Contains:
 - The instruction
 - Buffered operand values (when available)
 - Reservation station number of instruction providing the operand values
 - RS fetches and buffers an operand as soon as it becomes available (not necessarily involving register file)
 - Pending instructions designate the RS to which they will send their output
 - Result values broadcast on a result bus, called the common data bus (CDB)
 - Only the last output updates the register file
 - As instructions are issued, the register specifiers are renamed with the reservation station
 - May be more reservation stations than registers

Tomasulo's Algorithm

- Load and store buffers
 - Contain data and addresses, act like reservation stations

- Top-level design:



Tomasulo's Algorithm

- Three Steps:
 - Issue
 - Get next instruction from FIFO queue
 - If available RS, issue the instruction to the RS with operand values if available
 - If no RS is available, stall the instruction issue
 - Execute
 - When operand becomes available, store it in any reservation stations waiting for it
 - When all operands are ready, issue the instruction
 - Loads and store maintained in program order through effective address
 - No instruction allowed to initiate execution until all branches that proceed it in program order have completed
 - Write result
 - Write result on CDB into reservation stations and store buffers
 - (Stores must wait until address and value are received)

Tomasulo's Algorithm

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

V_j, V_k: Value of Source operands

- Store buffers has V field, result to be stored

Q_j, Q_k: Reservation stations producing source registers
(value to be written)

- Note: Q_j, Q_k=0 → ready
- Store buffers only have Q_j for RS producing result

A: Used to hold info for the load store (initially immediate, then effective address)

Busy: Indicates reservation station or FU is busy

Register result status— **Q_i** indicates which functional unit will write each register, 0 means no write to this register

Example

| Instruction | | Instruction status | | |
|-------------|-----------|--------------------|---------|--------------|
| | | Issue | Execute | Write Result |
| L.D | F6,32(R2) | √ | √ | √ |
| L.D | F2,44(R3) | √ | √ | |
| MUL.D | F0,F2,F4 | √ | | |
| SUB.D | F8,F2,F6 | √ | | |
| DIV.D | F10,F0,F6 | √ | | |
| ADD.D | F6,F8,F2 | √ | | |

| Reservation stations | | | | | | | |
|----------------------|------|------|----|---------------------|-------|-------|----------------|
| Name | Busy | Op | Vj | Vk | Qj | Qk | A |
| Load1 | No | | | | | | |
| Load2 | Yes | Load | | | | | 44 + Regs [R3] |
| Add1 | Yes | SUB | | Mem[32 + Regs [R2]] | Load2 | | |
| Add2 | Yes | ADD | | | Add1 | Load2 | |
| Add3 | No | | | | | | |
| Mult1 | Yes | MUL | | Regs [F4] | Load2 | | |
| Mult2 | Yes | DIV | | Mem[32 + Regs [R2]] | Mult1 | | |

| Register status | | | | | | | | | |
|-----------------|-------|-------|----|------|------|-------|-----|-----|-----|
| Field | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
| Qi | Mult1 | Load2 | | Add2 | Add1 | Mult2 | | | |

Dealing with WAR

| Instruction | | Instruction status | | | |
|-------------|-------------|--------------------|--|--|--|
| L.D | F6, 32(R2) | | | | |
| L.D | F2, 44(R3) | | | | |
| MUL.D | F0, F2, F4 | | | | |
| SUB.D | F8, F2, F6 | | | | |
| DIV.D | F10, F0, F6 | | | | |
| ADD.D | F6, F8, F2 | | | | |

| Name | Busy | Op | | |
|-------|------|------|--------------------|-------|
| Load1 | No | | | |
| Load2 | Yes | Load | | |
| Add1 | Yes | SUB | | |
| Add2 | Yes | ADD | | |
| Add3 | No | | | |
| Mult1 | Yes | MUL | Regs[F4] | Load2 |
| Mult2 | Yes | DIV | Mem[32 + Regs[R2]] | Mult1 |

| Register status | | | | | | | | | |
|-----------------|-------|-------|----|------|------|-------|-----|-----|-----|
| Field | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
| Qi | Mult1 | Load2 | | Add2 | Add1 | Mult2 | | | |

The processor issues both DIV and ADD although there is a WAR hazard.

If F6 is ready when DIV is issued, its value is read and stored in the RS (ADD may change it that is O.K.)

If not ready, RS will read it from the FU producing it, again ADD may change F6 since we will read it from the FU not F6

Instruction stream

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec</i> | <i>Write</i> | <i>Comp</i> | <i>Result</i> |
|-------------|----------|----------|--------------|-------------|--------------|-------------|---------------|
| 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 | <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 | | | | | |
| | Mult1 | No | | | | | |
| | Mult2 | No | | | | | |

FU count down

3 FP Adder R.S.
2 FP Mult R.S.

Register result status:

Clock

0

Clock cycle counter

| | <i>F0</i> | <i>F2</i> | <i>F4</i> | <i>F6</i> | <i>F8</i> | <i>F10</i> | <i>F12</i> | ... | <i>F30</i> |
|----|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| FU | | | | | | | | | |

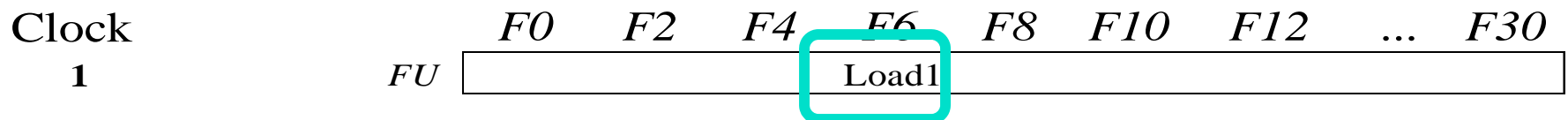
Instruction status:

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

Reservation Stations:

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

Register result status:



Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Comp</i> | <i>Write Result</i> |
|-------------|----------|----------|--------------|------------------|---------------------|
| LD | F6 | 34+ | R2 | 1 | |
| LD | F2 | 45+ | R3 | 2 | |
| MULTD | F0 | F2 | F4 | | |
| SUBD | F8 | F6 | F2 | | |
| DIVD | F10 | F0 | F6 | | |
| ADDD | F6 | F8 | F2 | | |

| | Busy | Address |
|-------|------|---------|
| Load1 | Yes | 34+R2 |
| Load2 | Yes | 45+R3 |
| Load3 | No | |

Reservation Stations:

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

Register result status:



Note: Can have multiple loads outstanding

Instruction status:

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

Reservation Stations:

| <i>Time</i> | <i>Name</i> | <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 | | | | | |
| Mult1 | | Yes | MULTD | | R(F4) | Load2 | |
| Mult2 | | No | | | | | |

Register result status:

| Clock | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... | F30 |
|-------|-------|-------|----|-------|----|-----|-----|-----|-----|
| 3 | Mult1 | Load2 | | Load1 | | | | | |

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

Instruction status:

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

Reservation Stations:

| Time | Name | Busy | Op | <i>S1 Vi</i> | <i>S2 Vk</i> | <i>RS Oi</i> | <i>RS Ok</i> |
|-------|------|-------|-------|--------------|--------------|--------------|--------------|
| 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 | Mult1 | Load2 | | M(A1) | Add1 | | | | |

- Load2 completing; what is waiting for Load2?

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Exec Write</i> | | | Busy | Address | |
|-------------|----------|----------|-------------------|-------------|---------------|------|---------|----|
| | | | <i>Issue</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 | | | | |
| DIVD | F10 | F0 | F6 | 5 | | | | |
| ADDD | F6 | F8 | F2 | | | | | |

Reservation Stations:

| Time | Name | Busy | Op | <i>S1</i> | <i>S2</i> | <i>RS</i> | <i>RS</i> |
|------|-------|------|-------|-----------|-----------|-----------|-----------|
| | | | | <i>Vj</i> | <i>Vk</i> | <i>Qj</i> | <i>Qk</i> |
| 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 | <i>F0</i> | <i>F2</i> | <i>F4</i> | <i>F6</i> | <i>F8</i> | <i>F10</i> | <i>F12</i> | ... | <i>F30</i> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 5 | FU | | | | | | | | |
| | Mult1 | M(A2) | | M(A1) | Add1 | Mult2 | | | |

- Timer starts down for Add1, Mult1

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Comp</i> | <i>Write Result</i> | Busy | Address | |
|-------------|----------|----------|--------------|------------------|---------------------|------|---------|----|
| 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 | | | | |
| DIVD | F10 | F0 | F6 | 5 | | | | |
| ADDD | F6 | F8 | F2 | 6 | | | | |

Reservation Stations:

| Time | Name | Busy | Op | <i>S1 Vj</i> | <i>S2 Vk</i> | <i>RS Qj</i> | <i>RS Qk</i> |
|------|-------|------|-------|--------------|--------------|--------------|--------------|
| 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?

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Comp</i> | <i>Write Result</i> | Busy | Address | |
|-------------|----------|----------|--------------|------------------|---------------------|------|---------|----|
| 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 | | | |
| DIVD | F10 | F0 | F6 | 5 | | | | |
| ADDD | F6 | F8 | F2 | 6 | | | | |

Reservation Stations:

| Time | Name | Busy | Op | <i>S1 Vj</i> | <i>S2 Vk</i> | <i>RS Qj</i> | <i>RS Qk</i> |
|------|-------|------|-------|--------------|--------------|--------------|--------------|
| 0 | Add1 | Yes | SUBD | M(A1) | M(A2) | | |
| | Add2 | Yes | ADDD | | M(A2) | Add1 | ← waiting |
| | 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?

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Exec Write</i> | | | Busy | Address | |
|-------------|----------|----------|-------------------|-------------|---------------|------|---------|----|
| | | | <i>Issue</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 | | | | |

Reservation Stations:

| <i>Time</i> | <i>Name</i> | <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 | | | | | |
| 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 | <i>F0</i> | <i>F2</i> | <i>F4</i> | <i>F6</i> | <i>F8</i> | <i>F10</i> | <i>F12</i> | ... | <i>F30</i> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 8 | Mult1 | M(A2) | | Add2 | (M-M) | Mult2 | | | |

Instruction status:

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

Reservation Stations:

| <i>Time</i> | <i>Name</i> | <i>Busy</i> | <i>Op</i> | <i>S1 Vj</i> | <i>S2 Vk</i> | <i>RS Qj</i> | <i>RS Qk</i> |
|-------------|-------------|-------------|-----------|--------------|--------------|--------------|--------------|
| | Add1 | No | | | | | |
| 1 | Add2 | Yes | ADDD | (M-M) | M(A2) | | |
| | Add3 | No | | | | | |
| 6 | Mult1 | Yes | MUL TD | 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> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 9 | Mult1 | M(A2) | | Add2 | (M-M) | Mult2 | | | |

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Comp</i> | <i>Write Result</i> | Load | Busy | Address |
|-------------|----------|----------|--------------|------------------|---------------------|------|-------|---------|
| 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 | | | |

Reservation Stations:

| <i>Time</i> | <i>Name</i> | <i>Busy</i> | <i>Op</i> | <i>S1 Vj</i> | <i>S2 Vk</i> | <i>RS Qj</i> | <i>RS Qk</i> |
|-------------|-------------|-------------|-----------|--------------|--------------|--------------|--------------|
| | 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 | <i>F0</i> | <i>F2</i> | <i>F4</i> | <i>F6</i> | <i>F8</i> | <i>F10</i> | <i>F12</i> | ... | <i>F30</i> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 10 | <i>FU</i> | Mult1 | M(A2) | | Add2 | (M-M) | Mult2 | | |

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

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Exec Write</i> | | | Busy | Address | |
|-------------|----------|----------|-------------------|-------------|---------------|------|---------|----|
| | | | <i>Issue</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:

| <i>Time</i> | <i>Name</i> | <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 | | | | | |
| 4 | 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> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 11 | Mult1 | M(A2) | | (M-M+M) | (M-M) | Mult2 | | | |

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

CSE4201

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Exec Write</i> | | | Busy | Address | |
|-------------|----------|----------|-------------------|-------------|---------------|------|---------|----|
| | | | <i>Issue</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:

| <i>Time</i> | <i>Name</i> | <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 | Mult1 | M(A2) | | (M-M+N) | (M-M) | Mult2 | | | |

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Exec Write</i> | | | Busy | Address | |
|-------------|----------|----------|-------------------|-------------|---------------|------|---------|----|
| | | | <i>Issue</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:

| <i>Time</i> | <i>Name</i> | <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 | Mult1 | M(A2) | | (M-M+M) | (M-M) | Mult2 | | | |

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Exec Write</i> | | | Busy | Address | |
|-------------|----------|----------|-------------------|-------------|---------------|------|---------|----|
| | | | <i>Issue</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:

| <i>Time</i> | <i>Name</i> | <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 | | | | | |
| 1 | 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> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 14 | Mult1 | M(A2) | | (M-M+M) | (M-M) | Mult2 | | | |

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Comp</i> | <i>Write Result</i> | Busy | Address | |
|-------------|----------|----------|--------------|------------------|---------------------|------|---------|----|
| LD | F6 | 34+ | R2 | 1 | 3 | 4 | Load1 | No |
| LD | F2 | 45+ | R3 | 2 | 4 | 5 | Load2 | No |
| MULTD | F0 | F2 | F4 | 3 | 15 | | 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 | Busy | Op | <i>S1</i> <i>Vj</i> | <i>S2</i> <i>Vk</i> | <i>RS</i> <i>Qj</i> | <i>RS</i> <i>Qk</i> |
|------|-------|------|-------|------------------------|------------------------|------------------------|------------------------|
| | 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+N | (M-M) | Mult2 | | | |

- Mult1 (MULTD) completing; who is waiting for it?**

Instruction status:

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

Reservation Stations:

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

Register result status:

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

- Just waiting for Mult2 (DIVD) to complete

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Exec Write</i> | | | Busy | Address | |
|-------------|----------|----------|-------------------|-------------|---------------|------|---------|----|
| | | | <i>Issue</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 | 15 | 16 | Load3 | No |
| SUBD | F8 | F6 | F2 | 4 | 7 | 8 | | |
| DIVD | F10 | F0 | F6 | 5 | | | | |
| ADDD | F6 | F8 | F2 | 6 | 10 | 11 | | |

Reservation Stations:

| <i>Time</i> | <i>Name</i> | <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 | | | | | |
| | Mult1 | No | | | | | |
| 1 | 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> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 55 | <i>FU</i> | M*F4 | M(A2) | | (M-M+N | (M-M) | Mult2 | | |

Instruction status:

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

Reservation Stations:

| <i>Time</i> | <i>Name</i> | <i>Busy</i> | <i>Op</i> | <i>S1 Vj</i> | <i>S2 Vk</i> | <i>RS Qj</i> | <i>RS Qk</i> |
|-------------|-------------|-------------|-----------|--------------|--------------|--------------|--------------|
| | Add1 | No | | | | | |
| | Add2 | No | | | | | |
| | Add3 | No | | | | | |
| | Mult1 | No | | | | | |
| 0 | 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> |
|-------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----|------------|
| 56 | M*F4 | M(A2) | | (M-M+N | (M-M) | Mult2 | | | |

Instruction status:

| Instruction | <i>j</i> | <i>k</i> | <i>Issue</i> | <i>Exec Comp</i> | <i>Write Result</i> | Busy | Address | |
|-------------|----------|----------|--------------|------------------|---------------------|------|---------|----|
| LD | F6 | 34+ | R2 | 1 | 3 | 4 | Load1 | No |
| LD | F2 | 45+ | R3 | 2 | 4 | 5 | Load2 | No |
| MULTD | F0 | F2 | F4 | 3 | 15 | 16 | Load3 | No |
| 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 | <i>S1</i> <i>Vj</i> | <i>S2</i> <i>Vk</i> | <i>RS</i> <i>Qj</i> | <i>RS</i> <i>Qk</i> |
|-------|------|------|------|------------------------|------------------------|------------------------|------------------------|
| 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+N | (M-M) | Result | | | |

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

Tomasulo's Algorithm

- Load and stores could be done out of order provided they access different memory locations.
- If they access same location, must preserve order (WAR, RAW, or WAW).
- If address calculation is done in program order, load/store can check if any uncompleted load/store share the same address
- Either wait or forward if possible.

Hardware-Based Speculation

- Execute instructions along predicted execution paths but only commit the results if prediction was correct
- Instruction commit: allowing an instruction to update the register file when instruction is no longer speculative
- Need an additional piece of hardware to prevent any irrevocable action until an instruction commits
- Need to separate executing the instruction to pass data to other instructions from completing (performing operations that can not be undone)

Reorder Buffer

- Register values and memory values are not written until an instruction commits
- On misprediction:
 - Speculated entries in ROB are cleared
- Exceptions:
 - Not recognized until it is ready to commit

Reorder Buffer

- **Reorder buffer** – holds the result of instruction between completion and commit (and supply them to any instruction who needs them just like the RS in Tomasulo's)
- Four fields:
 - Instruction type: branch/store/register
 - Destination field: register number or memory address
 - Value field: output value
 - Ready field: completed execution?
- Modify reservation stations:
 - Operand source is now reorder buffer instead of functional unit (results are tagged with ROB entry #)

Reorder Buffer

- Register values and memory values are not written until an instruction commits
- On misprediction:
 - Speculated entries in ROB are cleared
- Exceptions:
 - Not recognized until it is ready to commit
- 4 stages
 - Issue
 - Execute
 - Write Result
 - Commit

Reorder Buffer

- Issue
 - If empty RS and ROB entry → Issue; else stall
 - Send operands to RS if available in registers or ROB
 - The number of the ROB entry allocated to instruction is sent to RS to tag the results with
 - If operands are not available yet, the ROB entry is sent to the RS to wait for results on the CDB

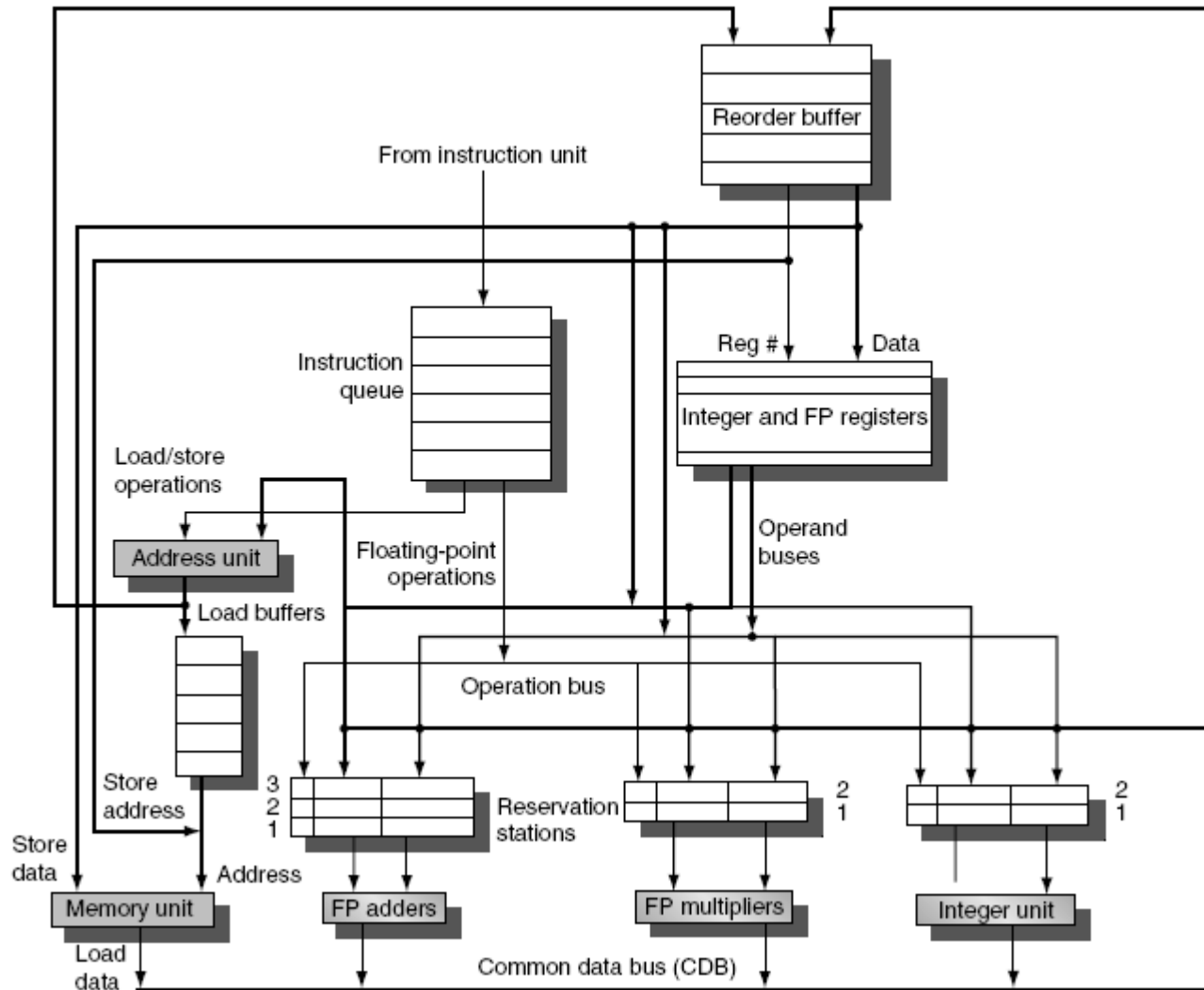
Reorder Buffer

- Execute
 - If one or more operands are not available, monitor the CDB.
 - When the result is broadcast on the CDB (we know that from the ROB entry tag) copy it
 - When all operands are ready, start execution
- Write Result
 - When execution is completed, broadcast the result on the CDB tagged with ROB entry #
 - Results are copied to ROB entry and all waiting RS
- Execute out of order, commit in order.

Reorder buffer

- When an instruction reaches the head of the ROB and the result is ready in the buffer,
 - If ALU op write it to the register file and remove instruction from ROB
 - If the instruction is a store, write it to the memory and remove the instruction from the ROB
 - If the instruction is a branch, if prediction is correct, remove it from the ROB. If misprediction flush the ROB and start from the correct successor.

Overview of Design



Multiple Issue and Static Scheduling

- To achieve $CPI < 1$, need to complete multiple instructions per clock
- Solutions:
 - Statically scheduled superscalar processors
 - VLIW (very long instruction word) processors
 - dynamically scheduled superscalar processors

Multiple Issue

| Common name | Issue structure | Hazard detection | Scheduling | Distinguishing characteristic | Examples |
|---------------------------|------------------|--------------------|--------------------------|---|---|
| Superscalar (static) | Dynamic | Hardware | Static | In-order execution | Mostly in the embedded space: MIPS and ARM, including the ARM Cortex A8 |
| Superscalar (dynamic) | Dynamic | Hardware | Dynamic | Some out-of-order execution, but no speculation | None at the present |
| Superscalar (speculative) | Dynamic | Hardware | Dynamic with speculation | Out-of-order execution with speculation | Intel Core i3, i5, i7; AMD Phenom; IBM Power 7 |
| VLIW/LIW | Static | Primarily software | Static | All hazards determined and indicated by compiler (often implicitly) | Most examples are in signal processing, such as the TI C6x |
| EPIC | Primarily static | Primarily software | Mostly static | All hazards determined and indicated explicitly by the compiler | Itanium |

VLIW Processors

- Package multiple operations into one instruction
- Example VLIW processor:
 - One integer instruction (or branch)
 - Two independent floating-point operations
 - Two independent memory references
- Must be enough parallelism in code to fill the available slots

VLIW Processors

- Disadvantages:
 - Statically finding parallelism
 - Code size
 - No hazard detection hardware
 - Binary code compatibility

VLIW Processors

- Package multiple operations into one instruction
- Example VLIW processor:
 - One integer instruction (or branch)
 - Two independent floating-point operations
 - Two independent memory references
- Must be enough parallelism in the code to fill the available slots

VLIW Processors

- Disadvantages:
 - Statically finding parallelism
 - Code size
 - No hazard detection hardware
 - Binary code compatibility

VLIW Example

| Source instruction | Instruction using result | Latency |
|--------------------|--------------------------|---------|
| FP ALU OP | FP ALU OP | 3 |
| FP ALU OP | Store double | 2 |
| Load double | FP ALU OP | 1 |
| Load Double | Store double | 0 |

```

Loop: L.D      F0, 0(R1)
      ADD.D    F4, F0, F2
      S.D     0(R1), F4
      DADDUI  R1, R1, #-8
      BNE R   1, R2, Loop
  
```

```

For (l=1000;l>0;l++)
    x[l]=x[l]+s;
  
```

VLIW Example

- Assume that we can schedule 2 memory operations, 2 FP operations, and one integer or branch

| <i>Memory reference 1</i> | <i>Memory reference 2</i> | <i>FP operation 1</i> | <i>FP op. 2</i> | <i>Int. op/branch</i> | <i>Clock</i> |
|---------------------------|---------------------------|-----------------------|-----------------|-----------------------|--------------|
| LD F0,0(R1) | LD F6,-8(R1) | | | | 1 |
| LD F10,-16(R1) | LD F14,-24(R1) | | | | 2 |
| LD F18,-32(R1) | LD F22,-40(R1) | ADDD F4,F0,F2 | ADDD F8,F6,F2 | | 3 |
| LD F26,-48(R1) | | ADDD F12,F10,F2 | ADDD F16,F14,F2 | | 4 |
| | | ADDD F20,F18,F2 | ADDD F24,F22,F2 | | 5 |
| SD 0(R1),F4 | SD -8(R1),F8 | ADDD F28,F26,F2 | | | 6 |
| SD -16(R1),F12 | SD -24(R1),F16 | | | DADD R1,R1,#-56 | 7 |
| SD 24(R1),F20 | SD 16(R1),F24 | | | | 8 |
| SD 8(R1),F28 | | | | BNEZ R1,LOOP | 9 |

Dynamic Scheduling, Multiple Issue, and Speculation

- Modern microarchitectures:
 - Dynamic scheduling + multiple issue + speculation
- Two approaches:
 - Assign reservation stations and update pipeline control table in half clock cycles
 - Only supports 2 instructions/clock
 - Design logic to handle any possible dependencies between the instructions
 - Hybrid approaches
- Issue logic can become bottleneck

Multiple Issue

- Limit the number of instructions of a given class that can be issued in a “bundle”
 - I.e. on FP, one integer, one load, one store
- Examine all the dependencies among the instructions in the bundle
- If dependencies exist in bundle, encode them in reservation stations
- Also need multiple completion/commit

Example

```
Loop: LD R2,0(R1)      ;R2=array element
      DADDIU R2,R2,#1  ;increment R2
      SD R2,0(R1)     ;store result
      DADDIU R1,R1,#8  ;increment pointer
      BNE R2,R3,LOOP  ;branch if not last element
```


Example (No Speculation)

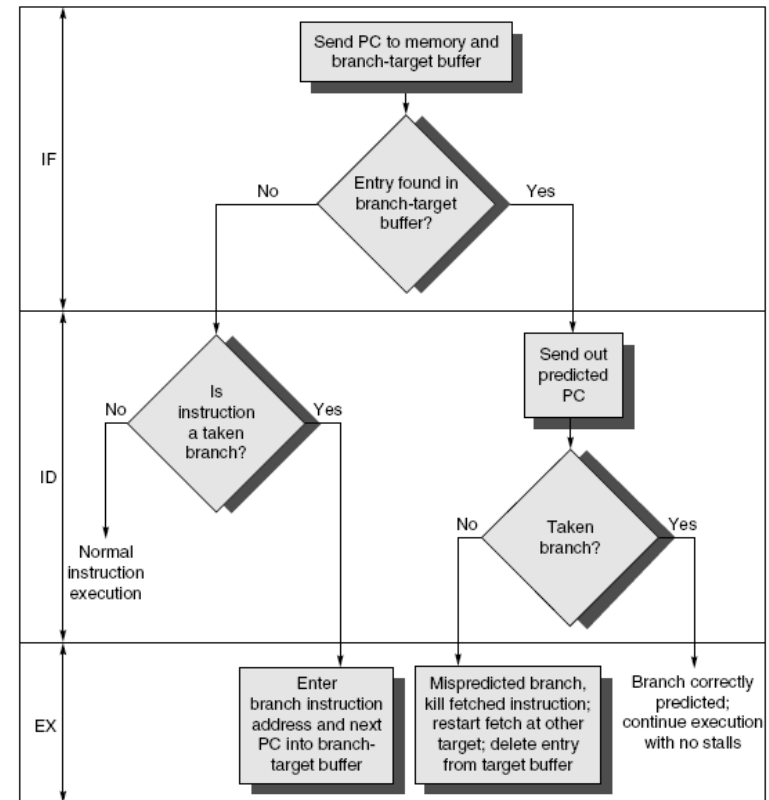
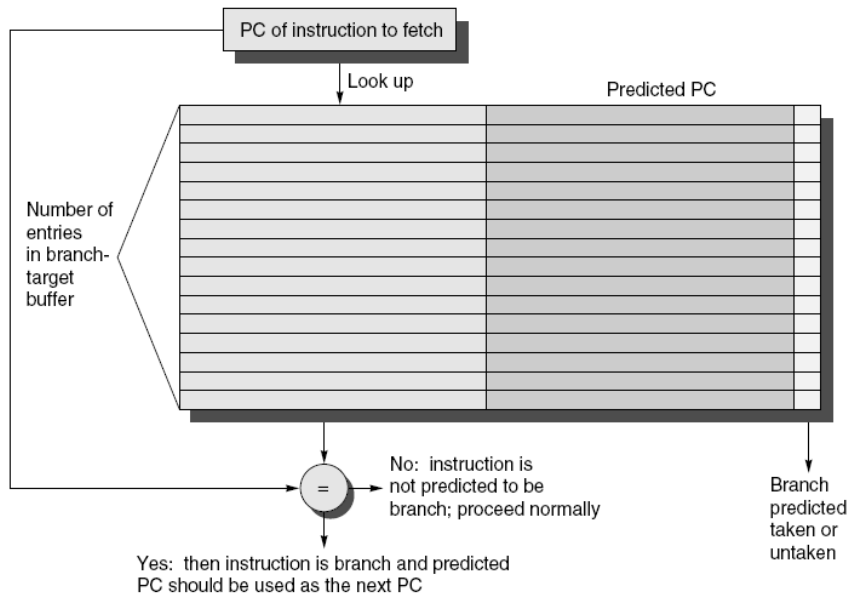
| Iteration number | Instructions | Issues at clock cycle number | Executes at clock cycle number | Memory access at clock cycle number | Write CDB at clock cycle number | Comment |
|------------------|-----------------|------------------------------|--------------------------------|-------------------------------------|---------------------------------|------------------|
| 1 | LD R2,0(R1) | 1 | 2 | 3 | 4 | First issue |
| 1 | DADDIU R2,R2,#1 | 1 | 5 | | 6 | Wait for LW |
| 1 | SD R2,0(R1) | 2 | 3 | 7 | | Wait for DADDIU |
| 1 | DADDIU R1,R1,#8 | 2 | 3 | | 4 | Execute directly |
| 1 | BNE R2,R3,LOOP | 3 | 7 | | | Wait for DADDIU |
| 2 | LD R2,0(R1) | 4 | 8 | 9 | 10 | Wait for BNE |
| 2 | DADDIU R2,R2,#1 | 4 | 11 | | 12 | Wait for LW |
| 2 | SD R2,0(R1) | 5 | 9 | 13 | | Wait for DADDIU |
| 2 | DADDIU R1,R1,#8 | 5 | 8 | | 9 | Wait for BNE |
| 2 | BNE R2,R3,LOOP | 6 | 13 | | | Wait for DADDIU |
| 3 | LD R2,0(R1) | 7 | 14 | 15 | 16 | Wait for BNE |
| 3 | DADDIU R2,R2,#1 | 7 | 17 | | 18 | Wait for LW |
| 3 | SD R2,0(R1) | 8 | 15 | 19 | | Wait for DADDIU |
| 3 | DADDIU R1,R1,#8 | 8 | 14 | | 15 | Wait for BNE |
| 3 | BNE R2,R3,LOOP | 9 | 19 | | | Wait for DADDIU |

Example

| Iteration number | Instructions | Issues at clock number | Executes at clock number | Read access at clock number | Write CDB at clock number | Commits at clock number | Comment |
|------------------|-----------------|------------------------|--------------------------|-----------------------------|---------------------------|-------------------------|-------------------|
| 1 | LD R2,0(R1) | 1 | 2 | 3 | 4 | 5 | First issue |
| 1 | DADDIU R2,R2,#1 | 1 | 5 | | 6 | 7 | Wait for LW |
| 1 | SD R2,0(R1) | 2 | 3 | | | 7 | Wait for DADDIU |
| 1 | DADDIU R1,R1,#8 | 2 | 3 | | 4 | 8 | Commit in order |
| 1 | BNE R2,R3,LOOP | 3 | 7 | | | 8 | Wait for DADDIU |
| 2 | LD R2,0(R1) | 4 | 5 | 6 | 7 | 9 | No execute delay |
| 2 | DADDIU R2,R2,#1 | 4 | 8 | | 9 | 10 | Wait for LW |
| 2 | SD R2,0(R1) | 5 | 6 | | | 10 | Wait for DADDIU |
| 2 | DADDIU R1,R1,#8 | 5 | 6 | | 7 | 11 | Commit in order |
| 2 | BNE R2,R3,LOOP | 6 | 10 | | | 11 | Wait for DADDIU |
| 3 | LD R2,0(R1) | 7 | 8 | 9 | 10 | 12 | Earliest possible |
| 3 | DADDIU R2,R2,#1 | 7 | 11 | | 12 | 13 | Wait for LW |
| 3 | SD R2,0(R1) | 8 | 9 | | | 13 | Wait for DADDIU |
| 3 | DADDIU R1,R1,#8 | 8 | 9 | | 10 | 14 | Executes earlier |
| 3 | BNE R2,R3,LOOP | 9 | 13 | | | 14 | Wait for DADDIU |

Branch-Target Buffer

- Need high instruction bandwidth!
 - Branch-Target buffers
 - Next PC prediction buffer, indexed by current PC



Branch Folding

- Optimization:
 - Larger branch-target buffer
 - Add target instruction into buffer to deal with longer decoding time required by larger buffer
 - “Branch folding”

Return Address Predictor

- Most unconditional branches come from function returns
- The same procedure can be called from multiple sites
 - Causes the buffer to potentially forget about the return address from previous calls
- Create return address buffer organized as a stack

Integrated Instruction Fetch Unit

- Design monolithic unit that performs:
 - Branch prediction
 - Instruction prefetch
 - Fetch ahead
 - Instruction memory access and buffering
 - Deal with crossing cache lines

Register Renaming

- Register renaming vs. reorder buffers
 - Instead of virtual registers from reservation stations and reorder buffer, create a single register pool
 - Contains visible registers and virtual registers
 - Use hardware-based map to rename registers during issue
 - WAW and WAR hazards are avoided
 - Speculation recovery occurs by copying during commit
 - Still need a ROB-like queue to update table in order
 - Simplifies commit:
 - Record that mapping between architectural register and physical register is no longer speculative
 - Free up physical register used to hold older value
 - In other words: SWAP physical registers on commit
 - Physical register de-allocation is more difficult

Integrated Issue and Renaming

- Combining instruction issue with register renaming:
 - Issue logic pre-reserves enough physical registers for the bundle (fixed number?)
 - Issue logic finds dependencies within bundle, maps registers as necessary
 - Issue logic finds dependencies between current bundle and already in-flight bundles, maps registers as necessary

How Much?

- How much to speculate
 - Mis-speculation degrades performance and power relative to no speculation
 - May cause additional misses (cache, TLB)
 - Prevent speculative code from causing higher costing misses (e.g. L2)
- Speculating through multiple branches
 - Complicates speculation recovery
 - No processor can resolve multiple branches per cycle

Energy Efficiency

- Speculation and energy efficiency
 - Note: speculation is only energy efficient when it significantly improves performance
- Value prediction
 - Uses:
 - Loads that load from a constant pool
 - Instruction that produces a value from a small set of values
 - Not been incorporated into modern processors
 - Similar idea--*address aliasing prediction*--is used on some processors