COSC4201
Instruction Level Parallelism
Dynamic Scheduling

Prof. Mokhtar Aboelaze

Parts of these slides are taken from Notes by
Prof. David Patterson (UCB)

Outline

- Data dependence and hazards
- Exposing parallelism (loop unrolling and scheduling)
- Reducing branch costs (prediction)
- Dynamic scheduling
- Speculation
- Multiple issue and static scheduling
- Advanced techniques
- Example

Introduction

- Loads or a stores can safely be done in any order, provided they access different addresses.
- If a load and a store access the same address, then
  - Either load is before store in program order, interchanging them results in WAR hazard.
  - The store is before the load in program order, interchanging them result in a RAW Hazard
  - Interchanging 2 stores, result in a WAW hazard.
- To proceed with a load, processor must check whether any uncompleted store that precedes the load in program order share the same data memory address as the load.
- Similarly, a store must check loads and stores.
- A not very efficient way, is to guarantee that address calculation are done in program order.

Speculation

- In dynamic scheduling, we wait before executing an instruction after a branch until the branch is resolved (integer operations may go ahead beyond branches).
- 3 components of HW-based speculation:

1. Dynamic branch prediction to choose which instructions to execute
2. Speculation to allow execution of instructions before control dependences are resolved

   + ability to undo effects of incorrectly speculated sequence
3. Dynamic scheduling to deal with scheduling of different combinations of basic blocks

Speculation

- Must separate execution from allowing instruction to finish or "commit"
- This additional step called instruction commit
- When an instruction is no longer speculative, allow it to update the register file or memory
- Requires additional set of buffers to hold results of instructions that have finished execution but have not committed
- This reorder buffer (ROB) is also used to pass results among instructions that may be speculated

Speculation

- In Tomasulo's algorithm, once an instruction writes its result, any subsequently issued instructions will find result in the register file
- With speculation, the register file is not updated until the instruction commits
  - (we know definitively that the instruction should execute)
- Thus, the ROB supplies operands in interval between completion of instruction execution and instruction commit
  - ROB is a source of operands for instructions, just as reservation stations (RS) provide operands in Tomasulo's algorithm
  - ROB extends architecture registers like RS
- ROB holds the results between the operation associated with the instruction completes, and commit

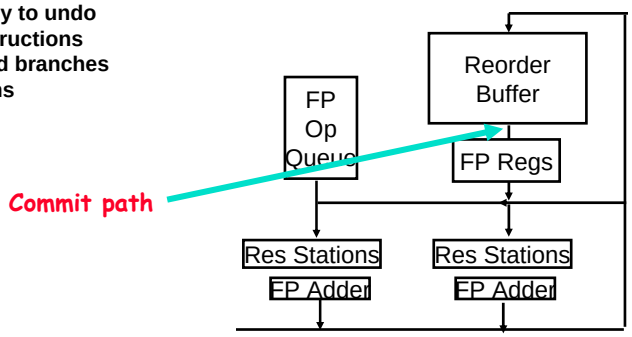
ROB

° Each entry in the ROB contains four fields:

1. Instruction type
   - a branch (has no destination result), a store (has a memory address destination), or a register operation (ALU operation or load, which has register destinations)
2. Destination
   - Register number (for loads and ALU operations) or memory address (for stores) where the instruction result should be written
3. Value
   - Value of instruction result until the instruction commits
4. Ready
   - Indicates that instruction has completed execution, and the value is ready

ROB

- Holds instructions in FIFO order, exactly as issued
- When instructions complete, results placed into ROB
  - Supplies operands to other instruction between execution complete & commit ⇒ more registers like RS
  - Tag results with ROB buffer number instead of reservation station
- Instructions commit ⇒values at head of ROB placed in registers
- As a result, easy to undo speculated instructions on mispredicted branches or on exceptions



Steps

1. **Issue—get instruction from FP Op Queue**

   If reservation station and reorder buffer slot free, issue instr & send operands & reorder buffer no. for destination (this stage sometimes called “dispatch”), OR stall

2. **Execution—operate on operands (EX)**

   When both operands ready then execute; if not ready, watch CDB for result; when both in reservation station, execute; checks RAW (sometimes called “issue”)

3. **Write result—finish execution (WB)**

   Write on Common Data Bus to all awaiting FUs (ROB tag) & reorder buffer; mark reservation station available.

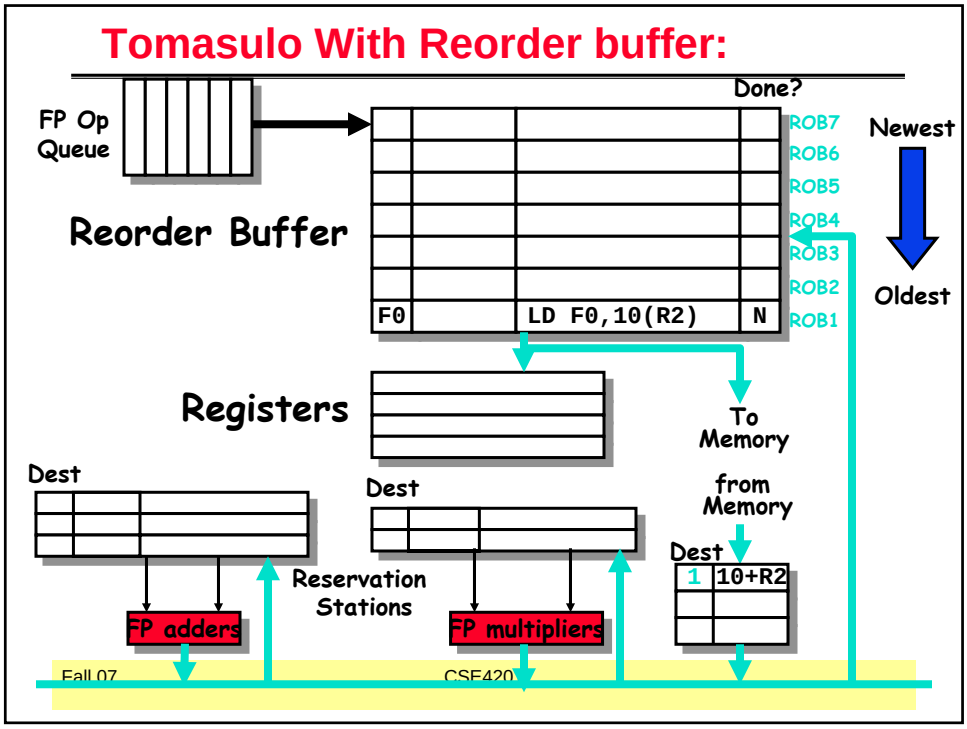
4. **Commit—update register with reorder result**

   When instr. at head of reorder buffer & result present, update register with result (or store to memory) and remove instr from reorder buffer. Mispredicted branch flushes reorder buffer (sometimes called “graduation”)

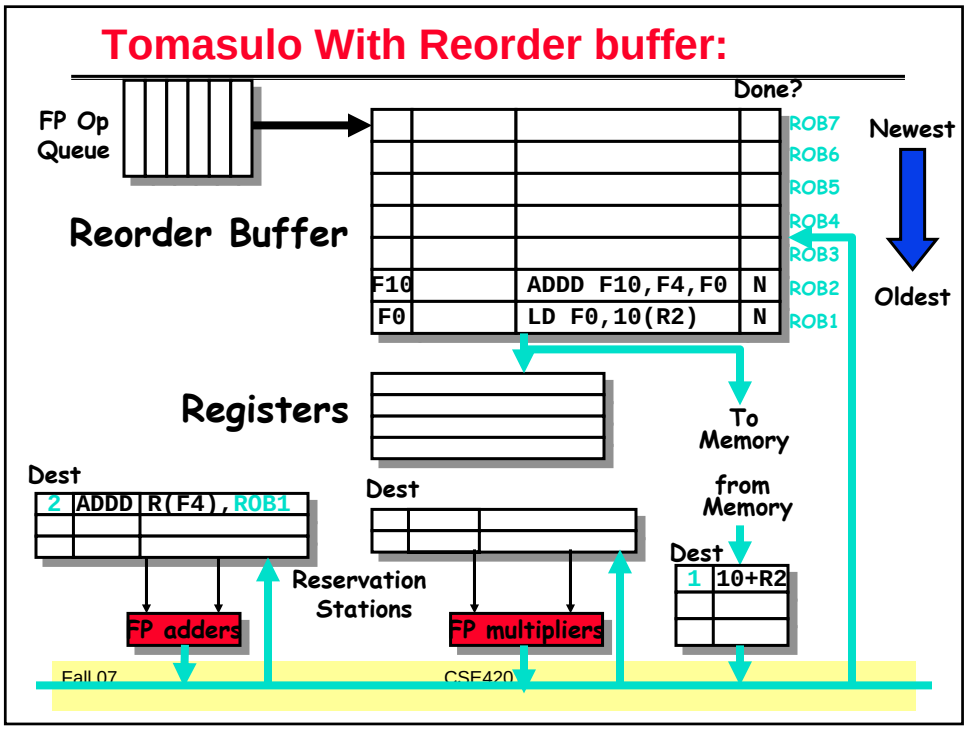
Example

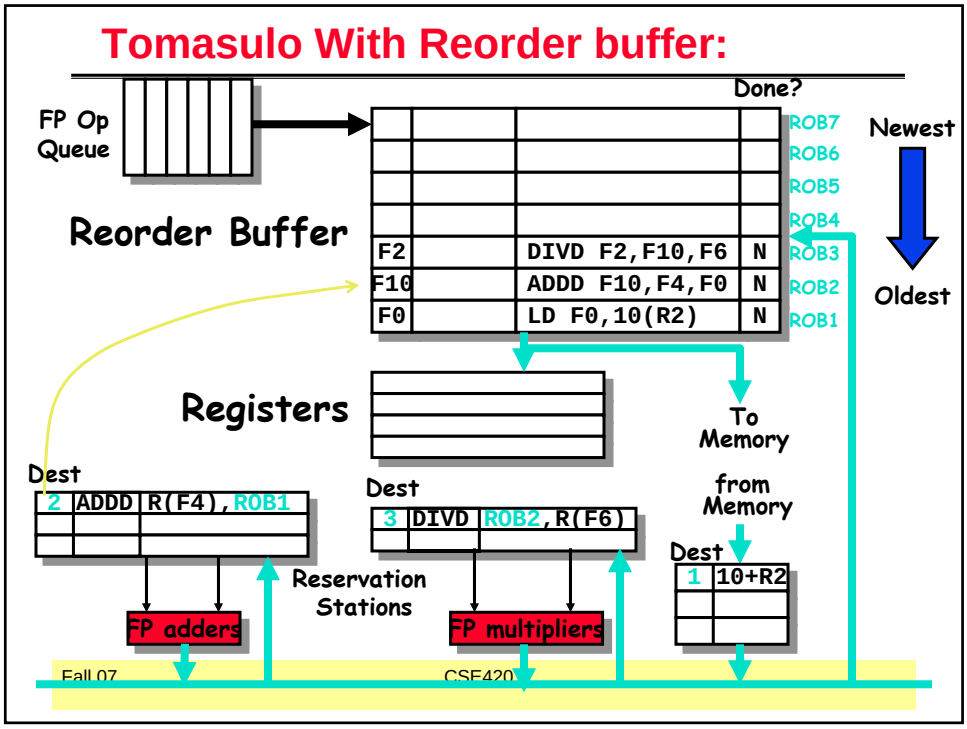
```
Loop  LD      F0,10(R2)
      ADDD    F10,F4,F0
      DIVD    F2,F10,F6
      DADD    R1,R1,-8
      BNE     R1,R2,Loop
```

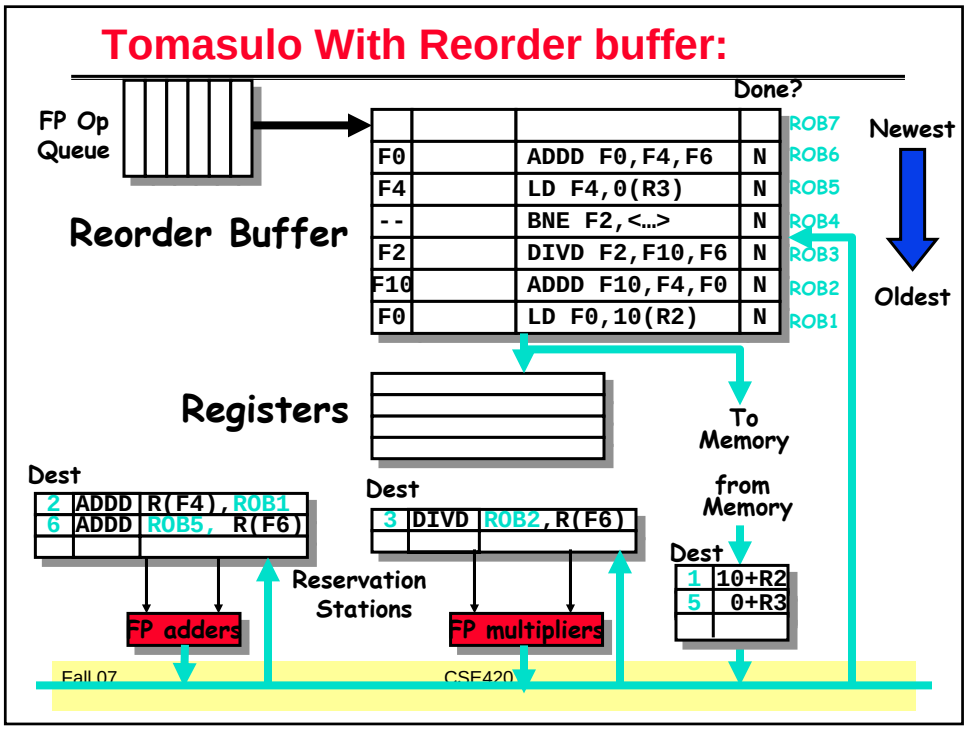
Tomasulo With Reorder buffer:

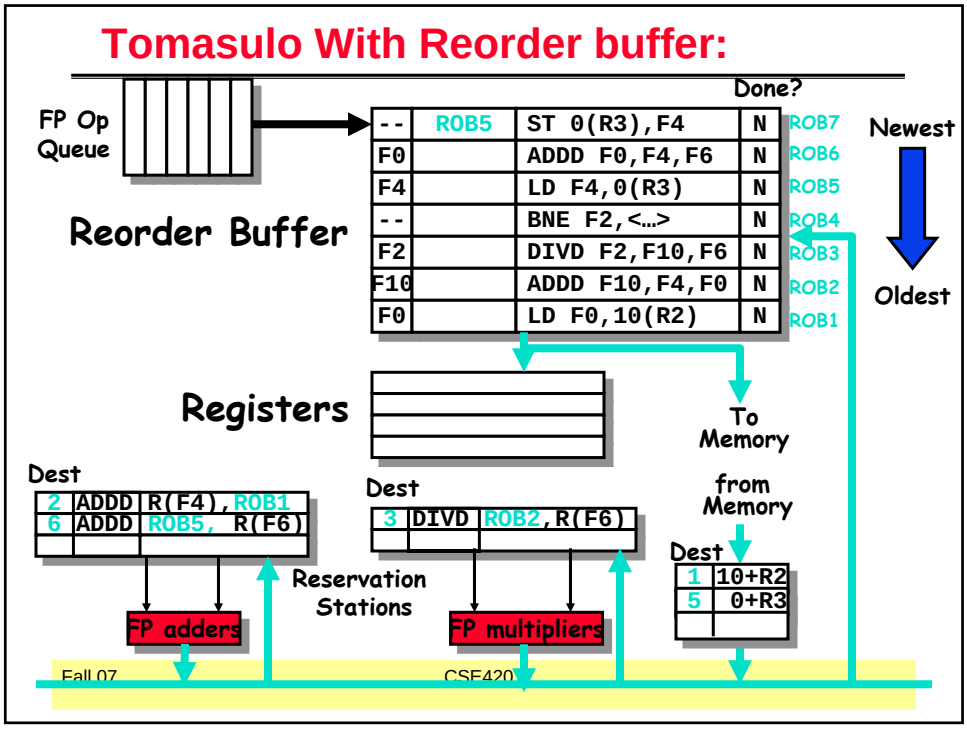


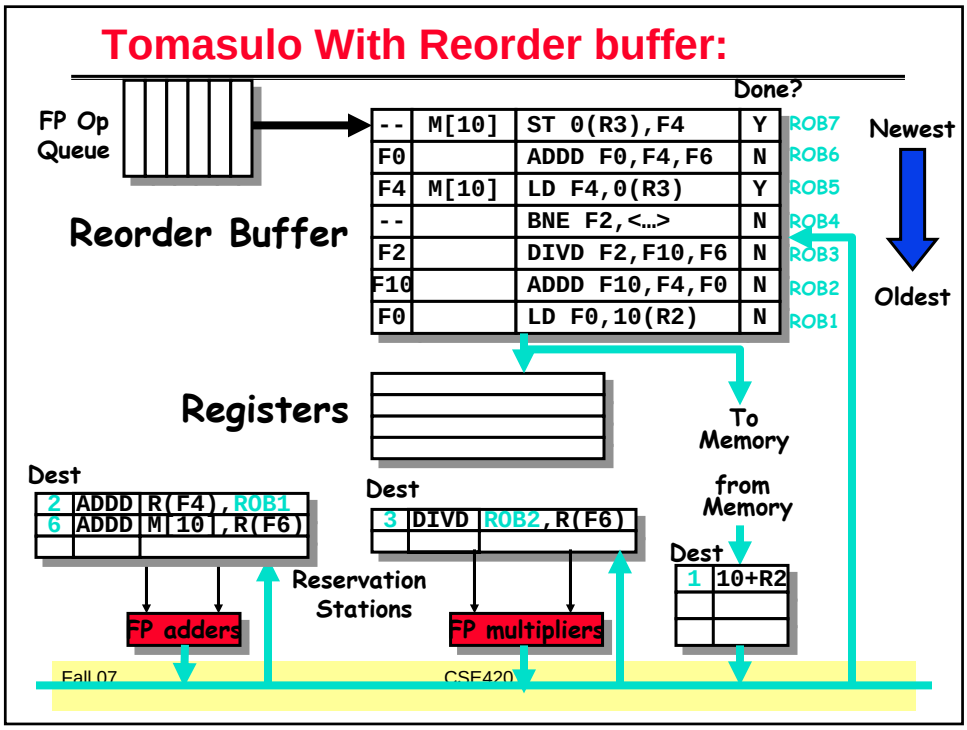
Tomasulo With Reorder buffer:

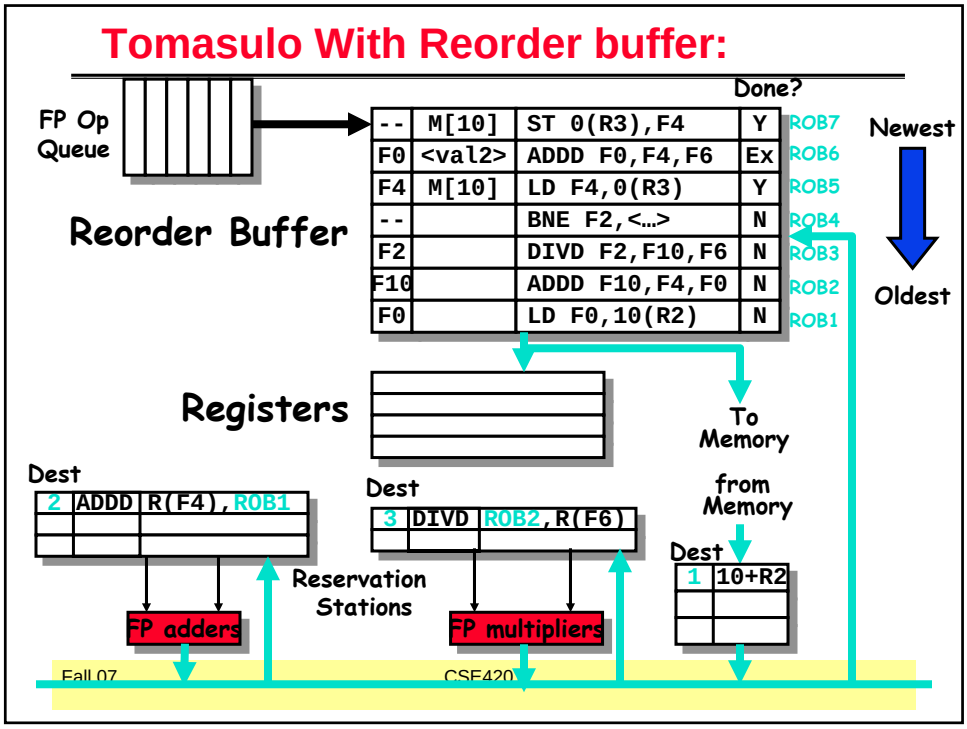


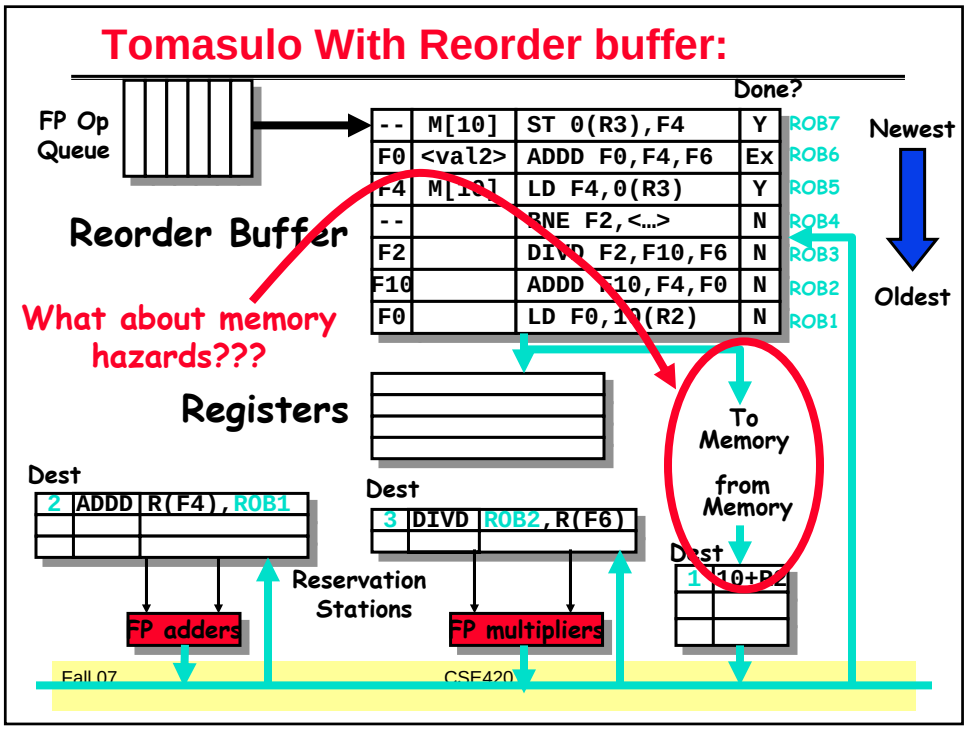

Tomasulo With Reorder buffer:
FP Op Queue
Reorder Buffer
Done?
Newest
Oldest
ROB7
ROB6
ROB5
ROB4
ROB3
ROB2
ROB1
F2 DIVD F2,F10,F6 N
F10 ADDD F10,F4,F0 N
F0 LD F0,10(R2) N
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
3 DIVD ROB2,R(F6)
1 10+R2
Reservation Stations
FP adders
FP multipliers
Fall 07
CSE420


Tomasulo With Reorder buffer:
FP Op Queue
Reorder Buffer
Done?
Newest
Oldest
ROB7
ROB6
ROB5
ROB4
ROB3
ROB2
ROB1
F0 ADDD F0,F4,F6 N
F4 LD F4,0(R3) N
-- BNE F2,<…> N
F2 DIVD F2,F10,F6 N
F10 ADDD F10,F4,F0 N
F0 LD F0,10(R2) N
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
6 ADDD ROB5, R(F6)
3 DIVD ROB2,R(F6)
1 10+R2
5 0+R3
Reservation Stations
FP adders
FP multipliers
Fall 07
CSE420


Tomasulo With Reorder buffer:
FP Op Queue
Reorder Buffer
Done?
-- ROB5 ST 0(R3),F4 N ROB7
F0 ADDD F0,F4,F6 N ROB6
F4 LD F4,0(R3) N ROB5
-- BNE F2,<...> N ROB4
F2 DIVD F2,F10,F6 N ROB3
F10 ADDD F10,F4,F0 N ROB2
F0 LD F0,10(R2) N ROB1
Newest
Oldest
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
6 ADDD ROB5, R(F6)
Dest
3 DIVD ROB2,R(F6)
Dest
1 10+R2
5 0+R3
Reservation Stations
FP adders
FP multipliers
Fall 07
CSE420


Tomasulo With Reorder buffer:
FP Op Queue
Reorder Buffer
Done?
-- M[10] ST 0(R3),F4 Y ROB7
F0 ADDD F0,F4,F6 N ROB6
F4 M[10] LD F4,0(R3) Y ROB5
-- BNE F2,<...> N ROB4
F2 DIVD F2,F10,F6 N ROB3
F10 ADDD F10,F4,F0 N ROB2
F0 LD F0,10(R2) N ROB1
Newest
Oldest
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
6 ADDD M[10],R(F6)
Dest
3 DIVD ROB2,R(F6)
Dest
1 10+R2
Reservation Stations
FP adders
FP multipliers
Fall 07
CSE420


Tomasulo With Reorder buffer:
FP Op Queue
Reorder Buffer
Done?
-- M[10] ST 0(R3),F4 Y ROB7
F0 <val2> ADDD F0,F4,F6 Ex ROB6
F4 M[10] LD F4,0(R3) Y ROB5
-- BNE F2,<...> N ROB4
F2 DIVD F2,F10,F6 N ROB3
F10 ADDD F10,F4,F0 N ROB2
F0 LD F0,10(R2) N ROB1
Newest
Oldest
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
3 DIVD ROB2,R(F6)
1 10+R2
Reservation Stations
FP adders
FP multipliers
Fall 07
CSE420


Tomasulo With Reorder buffer:
FP Op Queue
Reorder Buffer
Done?
-- M[10] ST 0(R3),F4 Y ROB7
F0 <val2> ADDD F0,F4,F6 Ex ROB6
F4 M[10] LD F4,0(R3) Y ROB5
-- BNE F2,<...> N ROB4
F2 DIVD F2,F10,F6 N ROB3
F10 ADDD F10,F4,F0 N ROB2
F0 LD F0,10(R2) N ROB1
Newest
Oldest
What about memory hazards???
Registers
To Memory
from Memory
Dest
2 ADDD R(F4),ROB1
3 DIVD ROB2,R(F6)
Reservation Stations
FP adders
FP multipliers
Fall 07
CSE420