Conceptual execution on a processor which exploits ILP

- Instruction fetch and branch prediction
  - Corresponds to IF in simple pipeline
  - Complicated by multiple issue and the potential need to fetch more than one basic block at a time (see later)
- Instruction decode, dependence check, dispatch, issue
  - Corresponds (many variations) to ID
  - Although instructions are issued (i.e., assigned to functional units), they might not execute right away (cf. reservation stations)
- Instruction execution
  - Corresponds to EX and/or MEM (with various latencies)
- Instruction commit
  - Corresponds to WB (see later) but more complex because of speculation and out-of-order completion

Register renaming - Generalities

- Use a *physical* register file (or other storage form, e.g., reservation station or reorder buffer) larger than the ISA *logical* one
- When instruction is decoded
  - Give a new name to result register from free list. The register is *renamed*
  - Give source operands their physical names (from mapping table)

Register renaming -- Requirements

- Keep a mapping table logical - physical correspondence
  - Because of branch prediction, need to save the mapping table when predicting branch
- Keep a free list of empty physical registers
- Note that several physical registers can be mapped to the same logical register (corresponding to instructions at different times; avoids WAW hazards)

Register renaming – Scheme 1: File of physical registers

- Extra set of registers
- At decode:
  - Rename the result register (get from free list; update mapping table). If none available, we have a structural hazard
- When a physical register has been read for the last time, return it to the free list
  - Have a counter associated with each physical register (+ when a source logical register is renamed to physical register; - when instruction uses physical register as operand; release when counter is 0)
  - Simpler to wait till logical register has been assigned a new name by a later instruction and that later instruction has been *committed*

Scheme 1: Example

Before: add r3,r3,4
add r4,r7,r3
add r3, r2, r7

after add r37,r3,4
add r38,r7,r37
add r39,r2,r7

Free list r37,r38,r39 ….
r2, r3, r4, r7 not renamed yet

At this point r3 is remapped from r37 to r39
When r39 commits, r37 will be returned to the free list

Register renaming – Scheme 2; Reorder buffer

- Use of a reorder buffer
  - Reorder buffer = circular queue with head and tail pointers
- At issue (renaming time), an instruction is assigned an entry at the tail of the reorder buffer which becomes the name of (or a pointer to) the result register
- At end of functional-unit computation, value is put in the instruction reorder buffer's position
- When the instruction reaches the head of the buffer, its value is stored in the logical or physical (other reorder buffer entry) register.
- Still need of a mapping table

Scheme 2: Example

Before: add r3,r3,4 after add rob6,r3,4
add r4,r7,r3 add rob7,r7,rob6
add r3, r2, r7 add rob8,r2,r7

Assume reorder buffer is initially at position 6 and has more than 8 slots

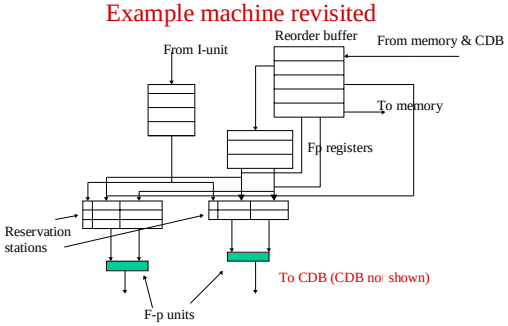
Data dependencies with register renaming

- Register renaming does not get rid of RAW dependencies
  - Still need for forwarding or for indicating whether a register has received its value
- Register renaming gets rid of WAW and WAR dependencies
- The reorder buffer, as its name implies, can be used for in-order completion

More on reorder buffer

- Tomasulo's scheme can be extended with the possibility of completing instructions in order
- Reorder buffer entry contains (this is not the only possible solution)
  - Type of instruction (branch, store, ALU, or load)
  - Destination (none, memory address, register)
  - Value and its presence/absence
- Replaces load/store buffers
- Reservation station tags and "true" register tags are now ids of entries in the reorder buffer

Example machine revisited



Need for 4 stages

- In Tomasulo's solution 3 stages: issue, execute, write
- Now 4 stages: issue, execute, write, commit
- Dispatch and Issue
  - Check for structural hazards (reservation stations busy, reorder buffer full). If one exists, stall the instruction and those following
  - If dispatch possible, send values to reservation station if the values are available in either the registers or the reorder buffer. Otherwise send tag.
  - Allocate an entry in the reorder buffer and send its number to the reservation station
  - When both operands are ready, issue to functional unit

Need for 4 stages (c'ed)

- Execute
- Write
  - Broadcast on common data bus the value and the tag (reorder buffer number). Reservation stations, if any match the tag, and reorder buffer (always) grab the value.
- Commit
  - When instr. at head of the reorder buffer has its result in the buffer it stores it in the real register (for ALU) or memory (for store). The reorder buffer entry (and/or physical register) is freed.

Reorder buffer

Entry #	Instruction	Issue	Execute	Write result	Commit
1	Load F6, 34(r2)	yes	yes	yes	yes
2	Load F2, 45(r3)	yes	yes		
3	Mul F0, F2, F4	yes			
4	Sub F8, F6, F2	yes			
5	Div F10, F0, F6	yes			
6	Add F6,F8,F2	yes			

Reservation Stations

Initial

Name	Busy	Fm	Vj	Vk	Qj	Qk
Add 1	yes	Sub	(#1)			(#2)
Add2	yes	Add			(#4)	(#2)
Add3	no					
Mul1	yes	Mul		(F4)	(#2)	
Mul2	yes	Div		(#1)	(#3)	

Register status

F0 (#3)	F2 (#2)	F4 ( )	F6(#6)	F8 (#4)	F10 (#5)	F12...

Reorder buffer

Entry #	Instruction	Issue	Execute	Write result	Commit
1	Load F6, 34(r2)	yes	yes	yes	yes
2	Load F2, 45(r3)	yes	yes	yes	yes
3	Mul F0, F2, F4	yes	yes		
4	Sub F8, F6, F2	yes	yes		
5	Div F10, F0, F6	yes			
6	Add F6,F8,F2	yes			

Cycle after 2nd load has committed

Reservation Stations

Name	Busy	Fm	Vj	Vk	Qj	Qk
Add 1	no					
Add2	yes	Add		(#2)	(#4)	
Add3	no					
Mul1	yes	Mul	(#2)	(F4)		
Mul2	yes	Div		(#1)	(#3)	

Register status

F0 (#3)	F2( )	F4 ( )	F6(#6)	F8 (#4)	F10 (#5)	F12...

Reorder buffer

Entry #	Instruction	Issue	Execute	Write result	Commit
1	Load F6, 34(r2)	yes	yes	yes	yes
2	Load F2, 45(r3)	yes	yes	yes	yes
3	Mul F0, F2, F4	yes	yes		
4	Sub F8, F6, F2	yes	yes	yes	
5	Div F10, F0, F6	yes			
6	Add F6,F8,F2	yes	yes		

Cycle after sub has written its result in reorder buffer but can't commit yet

Reservation Stations

Name	Busy	Fm	Vj	Vk	Qj	Qk
Add 1	no					
Add2	yes	Add	(#2)	(#4)		
Add3	no					
Mul1	yes	Mul	(#2)	(F4)		
Mul2	yes	Div		(#1)	(#3)	

Still waiting for #3 to commit

Register status

F0 (#3)	F2( )	F4 ( )	F6(#6)	F8 (#4)	F10 (#5)	F12...

Reorder buffer

Entry #	Instruction	Issue	Execute	Write result	Commit
1	Load F6, 34(r2)	yes	yes	yes	yes
2	Load F2, 45(r3)	yes	yes	yes	yes
3	Mul F0, F2, F4	yes	yes		
4	Sub F8, F6, F2	yes	yes	yes	
5	Div F10, F0, F6	yes			
6	Add F6,F8,F2	yes	yes	yes	

Cycle after add has written its result in reorder buffer but cannot commit

Reservation Stations

Name	Busy	Fm	Vj	Vk	Qj	Qk
Add 1	no					
Add2	no					
Add3	no					
Mul1	yes	Mul	(#2)	(F4)		
Mul2	yes	Div		(#1)	(#3)	

Still waiting for #3 to commit

Register status

F0 (#3)	F2( )	F4 ( )	F6(#6)	F8 (#4)	F10 (#5)	F12...

Still waiting for #3,#4, #5 to commit

Reorder buffer

Entry #	Instruction	Issue	Execute	Write result	Commit
1	Load F6, 34(r2)	yes	yes	yes	yes
2	Load F2, 45(r3)	yes	yes	yes	yes
3	Mul F0, F2, F4	yes	yes	yes	yes
4	Sub F8, F6, F2	yes	yes	yes	
5	Div F10, F0, F6	yes	yes		
6	Add F6,F8,F2	yes	yes		

Cycle after mul has written its result and committed

Reservation Stations

Name	Busy	Fm	Vj	Vk	Qj	Qk
Add 1	no					
Add2	no					
Add3	no					
Mul1	no					
Mul2	yes	Div	(#3)	(#1)		

Still waiting for #3 to commit

Register status

F0 ()	F2( )	F4 ( )	F6(#6)	F8 (#4)	F10 (#5)	F12...

Still waiting for #3,#4, #5 to commit

Reorder buffer

Entry #	Instruction	Issue	Execute	Write result	Commit
1	Load F6, 34(r2)	yes	yes	yes	yes
2	Load F2, 45(r3)	yes	yes	yes	yes
3	Mul F0, F2, F4	yes	yes	yes	yes
4	Sub F8, F6, F2	yes	yes	yes	yes
5	Div F10, F0, F6	yes	yes		
6	Add F6,F8,F2	yes	yes	yes	

Now #3 can commit

Reservation Stations

Name	Busy	Fm	Vj	Vk	Qj	Qk
Add 1	no					
Add2	no					
Add3	no					
Mul1	no					
Mul2	yes	Div	(#3)	(#1)		

Register status

F0 ()	F2( )	F4 ( )	F6(#6)	F8 ()	F10 (#5)	F12...

Still waiting for #4, #5 to commit

Reorder buffer

Entry #	Instruction	Issue	Execute	Write result	Commit
1	Load F6, 34(r2)	yes	yes	yes	yes
2	Load F2, 45(r3)	yes	yes	yes	yes
3	Mul F0, F2, F4	yes	yes	yes	yes
4	Sub F8, F6, F2	yes	yes	yes	yes
5	Div F10, F0, F6	yes	yes		
6	Add F6,F8,F2	yes	yes	yes	

Reservation Stations

Name	Busy	Fm	Vj	Vk	Qj	Qk
Add 1	no					
Add2	no					
Add3	no					
Mul1	no					
Mul2	yes	Div	(#3)	(#1)		

The next "interesting event is completion of div; then commit of #5, then commit of #6

Register status

F0 () F2( ) F4 ( ) F6(#6) F8 () F10 (#5) F12...

Still waiting for #4, #5 to commit