CSE 490/590 Computer Architecture

ILP I

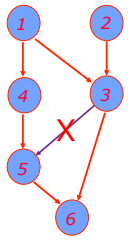
Steve Ko
Computer Sciences and Engineering
University at Buffalo

Last time…

- Scoreboard
  - Data structure that keeps track of dependencies among instructions
- In-order limitations
  - Out-of-order alone cannot solve
- Register renaming
  - Overcoming the restriction caused by the # of registers

Instruction-level Parallelism via R*enaming*

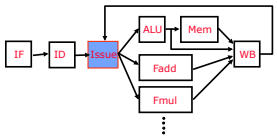
					latency
1	LD	F2,	34(R2)		1
2	LD	F4,	45(R3)		long
3	MULTD	F6,	F4,	F2	3
4	SUBD	F8,	F2,	F2	1
5	DIVD	F4',	F2,	F8	4
6	ADDD	F10,	F6,	F4'	1



In-order: 1 (2,1) . . . . . 2 3 4 4 3 5 . . . 5 6 6
Out-of-order: 1 (2,1) 4 4 5 . . . 2 (3,5) 3 6 6

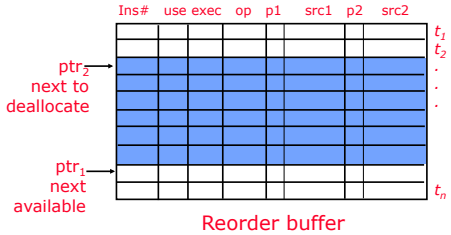
Any antidependence can be eliminated by renaming.
(renaming ⇒ additional storage)
Can it be done in hardware? yes!

Register Renaming



- Decode does register renaming and adds instructions to the issue stage reorder buffer (ROB)
  ⇒ renaming makes WAR or WAW hazards impossible
- Any instruction in ROB whose RAW hazards have been satisfied can be dispatched.
  ⇒ Out-of-order or dataflow execution

Dataflow Execution



Instruction slot is candidate for execution when:
- It holds a valid instruction ("use" bit is set)
- It has not already started execution ("exec" bit is clear)
- Both operands are available (p1 and p2 are set)

Renaming & Out-of-order Issue

An example

Renaming table

	p	data
F1		
F2	1	v1
F3		
F4	1	t5
F5		
F6	1	t3
F7		
F8	1	t4

Reorder buffer

Ins#	use	exec	op	p1	src1	p2	src2
1	0	0	LD				
2	0	0	LD				
3	1	0	MUL	0	v2	1	v1
4	0	0	SUB	1	v1	1	v1
5	1	0	DIV	1	v1	0	t4

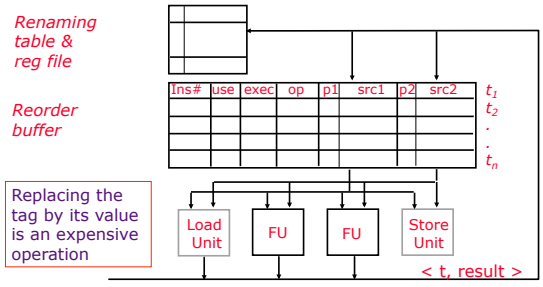
v1

data / t_i

1	LD	F2,	34(R2)	
2	LD	F4,	45(R3)	
3	MULTD	F6,	F4,	F2
4	SUBD	F8,	F2,	F2
5	DIVD	F4,	F2,	F8
6	ADDD	F10,	F6,	F4

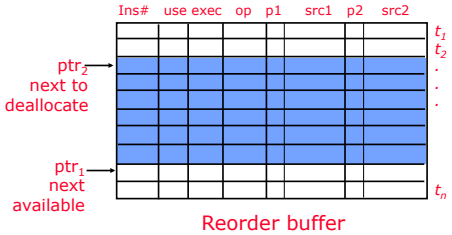
- *When are tags in sources replaced by data?*
  Whenever an FU produces data
- *When can a name be reused?*
  Whenever an instruction completes

Data-Driven Execution



- Instruction template (i.e., tag t) is allocated by the Decode stage, which also associates tag with register in regfile
- When an instruction completes, its tag is deallocated

Simplifying Allocation/Deallocation



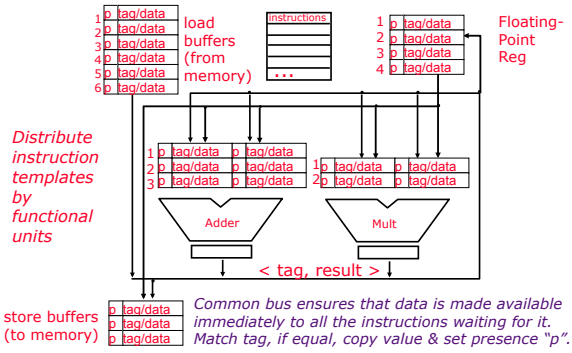
Reorder buffer

Instruction buffer is managed circularly

- "exec" bit is set when instruction begins execution
- When an instruction completes its "use" bit is marked free
- ptr_2 is incremented only if the "use" bit is marked free

IBM 360/91 Floating-Point Unit

R. M. Tomasulo, 1967



Common bus ensures that data is made available immediately to all the instructions waiting for it. Match tag, if equal, copy value & set presence "p".

Effectiveness?

Renaming and Out-of-order execution was first implemented in 1969 in IBM 360/91 but did not show up in the subsequent models until mid-Nineties.

Why ?

Reasons

1. Effective on a very small class of programs
2. Memory latency a much bigger problem
3. Exceptions not precise!

One more problem needed to be solved

Control transfers

CSE 490/590 Administrivia

- No office hours this week
  - Appointment via email if needed
  - Project-related questions → fastest: Safwan or Jangyoung
- Guest Lecture by Prof. Kris Schindler on Wed
- Guest lecture by Prof. Tevfik Kosar on Fri

Precise Interrupts

It must appear as if an interrupt is taken between two instructions (say I_i and I_{i+1})

- the effect of all instructions up to and including I_i is totally complete
- no effect of any instruction after I_i has taken place

The interrupt handler either aborts the program or restarts it at I_{i+1} .

Effect on Interrupts

Out-of-order Completion

I_1	DIVD	f6,	f6,	f4
I_2	LD	f2,	45(r3)	
I_3	MULTD	f0,	f2,	f4
I_4	DIVD	f8,	f6,	f2
I_5	SUBD	f10,	f0,	f6
I_6	ADDD	f6,	f8,	f2

out-of-order comp 1 2 2 3 1 4 3 5 5 4 6 6

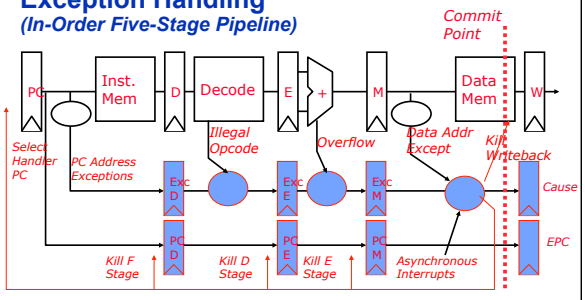
restore f2 *restore f10*

Consider interrupts

Precise interrupts are difficult to implement at high speed
- want to start execution of later instructions before exception checks finished on earlier instructions

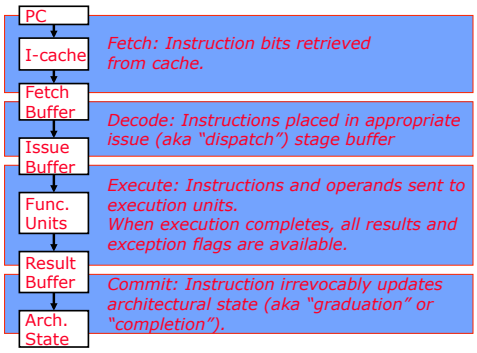
Exception Handling

(In-Order Five-Stage Pipeline)



- Hold exception flags in pipeline until commit point (M stage)
- Exceptions in earlier pipe stages override later exceptions
- Inject external interrupts at commit point (override others)
- If exception at commit: update Cause and EPC registers, kill all stages, inject handler PC into fetch stage

Phases of Instruction Execution



Acknowledgements

- These slides heavily contain material developed and copyright by
  - Krste Asanovic (MIT/UCB)
  - David Patterson (UCB)
- And also by:
  - Arvind (MIT)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
- MIT material derived from course 6.823
- UCB material derived from course CS252