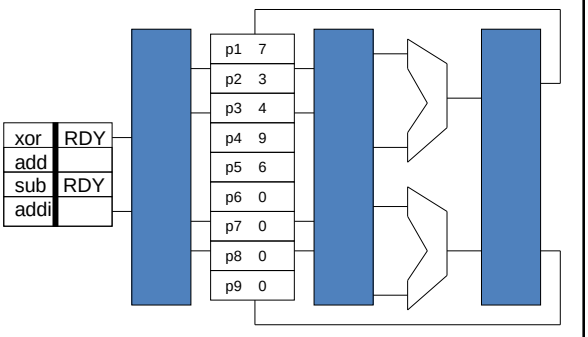
Register Read

- When do instructions read the register file?
- Option #1: after issue, right before execute
  - (Not done at decode)
  - Read **physical** register (renamed)
  - Or get value via bypassing (based on physical register name)
  - This is Pentium 4, MIPS R10k, Alpha 21264 style
- Physical register file may be large
  - Multi-cycle read
- Option #2: as part of dispatch, keep values in Issue Queue
  - Pentium Pro, Core 2, Core i7

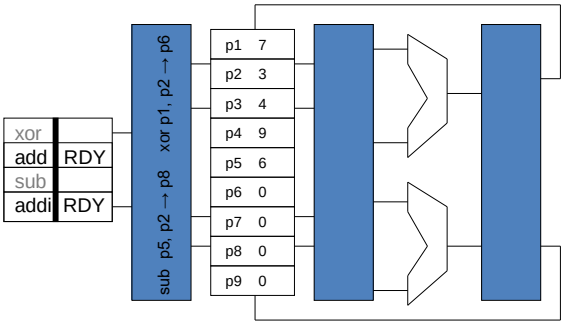
43

OOO execution (2-wide)



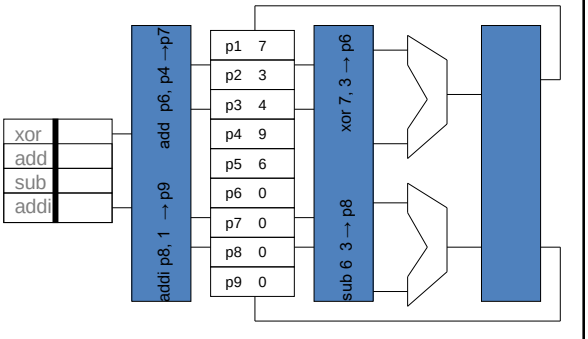
48

OOO execution (2-wide)



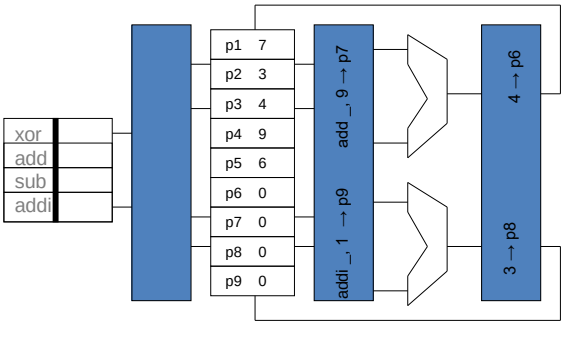
49

OOO execution (2-wide)



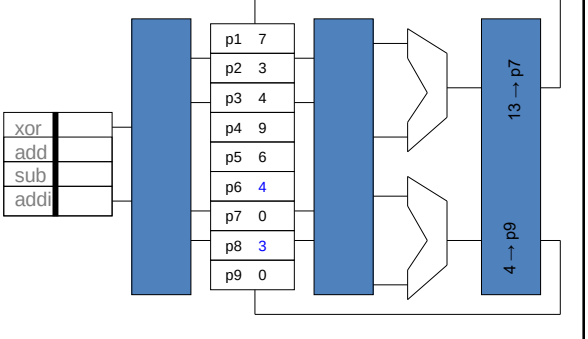
50

OOO execution (2-wide)



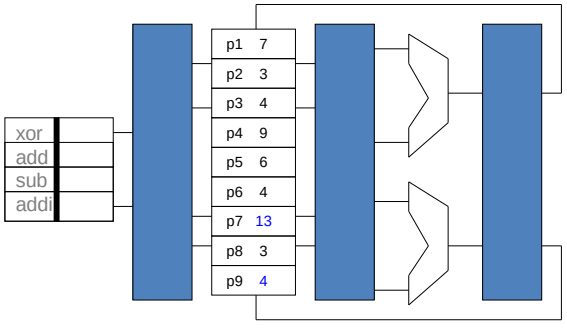
51

OOO execution (2-wide)



52

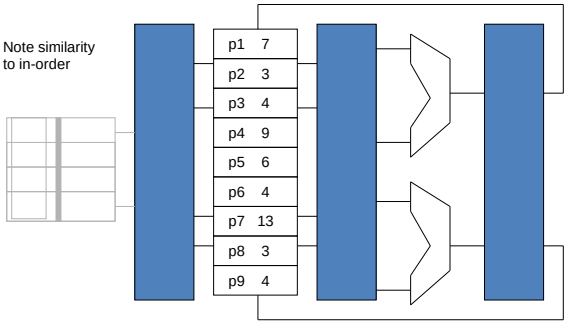
OOO execution (2-wide)



53

53

OOO execution (2-wide)



54

54

Multi-cycle operations

- Multi-cycle ops (load, fp, multiply, *etc.*)
  - Wakeup deferred a few cycles
    - Structural hazard?
- Cache misses?
  - Speculative wake-up (assume hit)
  - Cancel exec of dependents
  - Re-issue later
  - Details: complicated, not important

55

55

Re-order Buffer (ROB)

- All instructions in order
- Two purposes
  - Misprediction recovery
  - In-order commit
    - Maintain appearance of in-order execution
    - Freeing of physical registers

56

56

RENAMING REVISITED

57

57

Renaming revisited

- Overwritten register
  - Freed at commit
  - Restore in map table on recovery
    - Branch mis-prediction recovery
  - Also must be read at rename

58

58

Renaming example

Original insns

```
xor  r1,r2 → r3
add  r3,r4 → r4
sub  r5,r2 → r3
addi r3,1  → r1
```

r1	p1
r2	p2
r3	p3
r4	p4
r5	p5

Map table

p6
p7
p8
p9
p10

Free-list

59

59

Renaming example

Original insns	Renamed insns	Overwritten Reg
`xor r1,r2 → r3`	`xor p1, p2 →`	[p3]
`add r3,r4 → r4`		
`sub r5,r2 → r3`		
`addi r3,1 → r1`		

r1	p1
r2	p2
r3	**p3**
r4	p4
r5	p5

Map table

p6
p7
p8
p9
p10

Free-list

60

60

Renaming example

Original insns	Renamed insns	Overwritten Reg
`xor r1,r2 → r3`	`xor p1, p2 → p6`	[p3]
`add r3,r4 → r4`		
`sub r5,r2 → r3`		
`addi r3,1 → r1`		

r1	p1
r2	p2
r3	**p6**
r4	p4
r5	p5

Map table

p7
p8
p9
p10

Free-list

61

61

Renaming example

Original insns	Renamed insns	Overwritten Reg
`xor r1,r2 → r3`	`xor p1, p2 → p6`	[p3]
`add r3,r4 → r4`	`add p6, p4 →`	[p4]
`sub r5,r2 → r3`		
`addi r3,1 → r1`		

r1	p1
r2	p2
r3	p6
r4	**p4**
r5	p5

Map table

p7
p8
p9
p10

Free-list

62

62

Renaming example

Original insns	Renamed insns	Overwritten Reg
`xor r1,r2 → r3`	`xor p1, p2 → p6`	[p3]
`add r3,r4 → r4`	`add p6, p4 → p7`	[p4]
`sub r5,r2 → r3`		
`addi r3,1 → r1`		

r1	p1
r2	p2
r3	p6
r4	**p7**
r5	p5

Map table

p8
p9
p10

Free-list

63

63

Renaming example

Original insns	Renamed insns	Overwritten Reg
`xor r1,r2 → r3`	`xor p1, p2 → p6`	[p3]
`add r3,r4 → r4`	`add p6, p4 → p7`	[p4]
`sub r5,r2 → r3`	`sub p5, p2 →`	[p6]
`addi r3,1 → r1`		

r1	p1
r2	p2
r3	**p6**
r4	p7
r5	p5

Map table

p8
p9
p10

Free-list

64

64

Renaming example

Original insns	Renamed insns	Overwritten Reg
xor r1,r2 → r3	xor p1, p2 → p6	[p3]
add r3,r4 → r4	add p6, p4 → p7	[p4]
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1		

r1	p1
r2	p2
r3	**p8**
r4	p7
r5	p5

Map table

p9
p10

Free-list

65

65

Renaming example

Original insns	Renamed insns	Overwritten Reg
xor r1,r2 → r3	xor p1, p2 → p6	[p3]
add r3,r4 → r4	add p6, p4 → p7	[p4]
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1	addi p8, 1 →	[p1]

r1	**p1**
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p9
p10

Free-list

66

66

Renaming example

Original insns	Renamed insns	Overwritten Reg
xor r1,r2 → r3	xor p1, p2 → p6	[p3]
add r3,r4 → r4	add p6, p4 → p7	[p4]
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1	addi p8, 1 → p9	[p1]

r1	**p9**
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p10

Free-list

67

67

ROB

- ROB entry holds all info for recover/commit
  - Logical register names
  - Physical register names
  - Instruction types
- Dispatch: insert at tail
  - Full? Stall
- Commit: remove from head
  - Not completed? Stall

68

68

Recovery

- Completely remove wrong path instructions
  - Flush from IQ
  - Remove from ROB
  - Restore map table to before misprediction
  - Free destination registers

69

69

Recovery example

Original insns	Renamed insns	Overwritten Reg
bnz r1 loop	bnz p1, loop	[ ]
xor r1, r2 → r3	xor p1, p2 → p6	[p3]
add r3, r4 → r4	add p6, p4 → p7	[p4]
sub r5, r2 → r3	sub p5, p2 → p8	[p6]
addi r3, 1 → r1	addi p8, 1 → p9	[p1]

r1	p9
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p10

Free-list

70

70

Recovery example

Original insns	Renamed insns	Overwritten Reg
bnz r1 loop	bnz p1, loop	[ ]
xor r1, r2 → r3	xor p1, p2 → p6	[p3]
add r3, r4 → r4	add p6, p4 → p7	[p4]
sub r5, r2 → r3	sub p5, p2 → p8	[p6]
addi r3, 1 → r1	addi p8, 1 → p9	[p1]

r1	**p1**
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p9
p10

Free-list

71

71

Recovery example

Original insns	Renamed insns	Overwritten Reg
bnz r1 loop	bnz p1, loop	[ ]
xor r1, r2 → r3	xor p1, p2 → p6	[p3]
add r3, r4 → r4	add p6, p4 → p7	[p4]
sub r5, r2 → r3	sub p5, p2 → p8	[p6]

r1	p1
r2	p2
r3	**p6**
r4	p7
r5	p5

Map table

p8
p9
p10

Free-list

72

72

Recovery example

Original insns	Renamed insns	Overwritten Reg
bnz r1 loop	bnz p1, loop	[ ]
xor r1, r2 → r3	xor p1, p2 → p6	[p3]
add r3, r4 → r4	add p6, p4 → p7	[p4]

r1	p1
r2	p2
r3	p6
r4	**p4**
r5	p5

Map table

p7
p8
p9
p10

Free-list

73

73

Recovery example

Original insns	Renamed insns	Overwritten Reg
bnz r1 loop	bnz p1, loop	[ ]
xor r1, r2 → r3	xor p1, p2 → p6	[p3]

r1	p1
r2	p2
r3	**p3**
r4	p4
r5	p5

Map table

p6
p7
p8
p9
p10

Free-list

74

74

Recovery example

Original insns	Renamed insns	Overwritten Reg
bnz r1 loop	bnz p1, loop	[ ]

r1	p1
r2	p2
r3	p6
r4	p4
r5	p5

Map table

p7
p8
p9
p10

Free-list

75

75

What about stores

- Stores: Write D$, not registers
  - Can we rename memory?
  - Recover in the cache?

No (at least not easily)

- Cache writes unrecoverable
- Stores: only when certain
  - Commit

76

76

Commit

Original insns	Renamed insns	Overwritten Reg
xor r1, r2 → r3	xor p1, p2 → p6	[p3]
add r3, r4 → r4	add p6, p4 → p7	[p4]
sub r5, r2 → r3	sub p5, p2 → p8	[p6]
addi r3, 1 → r1	addi p8, 1 → p9	[p1]

- At commit: instruction becomes architected state
- In-order
- Only when instructions are finished
- Free overwritten register (why?)

77

77

Freeing over-written register

Original insns	Renamed insns	Overwritten Reg
xor r1,r2 → r3	xor p1, p2 → p6	[p3]
add r3, r4 → r4	add p6, p4 → p7	[p4]
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1	addi p8, 1 → p9	[p1]

- Before xor: **r3→ p3**
- After xor: **r3→ p6**
  - Insns older than xor reads p3
  - Insns younger than xor read p6 (until next r3-writing instruction)
- At commit of xor, no older instructions exist
  - No one else needs p3 → free it!

78

78

Commit Example

Original insns	Renamed insns	Overwritten Reg
xor r1,r2 → r3	xor p1, p2 → p6	[p3]
add r3,r4 → r4	add p6, p4 → p7	[p4]
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1	addi p8, 1 → p9	[p1]

r1	p9
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p10

Free-list

79

79

Commit Example

Original insns	Renamed insns	Overwritten Reg
xor r1,r2 → r3	xor p1, p2 → p6	[p3]
add r3,r4 → r4	add p6, p4 → p7	[p4]
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1	addi p8, 1 → p9	[p1]

r1	p9
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p10
p3

Free-list

80

80

Commit Example

Original insns	Renamed insns	Overwritten Reg
add r3,r4 → r4	add p6, p4 → p7	[p4]
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1	addi p8, 1 → p9	[p1]

r1	p9
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p10
p3
p4

Free-list

81

81

Commit Example

Original insns	Renamed insns	Overwritten Reg
sub r5,r2 → r3	sub p5, p2 → p8	[p6]
addi r3,1 → r1	addi p8, 1 → p9	[p1]

r1	p9
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p10
p3
p4
p6

Free-list

82

82

Commit Example

Original insns	Renamed insns	Overwritten Reg
`addi r3,1 → r1`	`addi p8, 1 → p9`	`[p1]`

r1	p9
r2	p2
r3	p8
r4	p7
r5	p5

Map table

p10
p3
p4
p6
p1

Free-list

83

83

Out of order pipeline diagrams

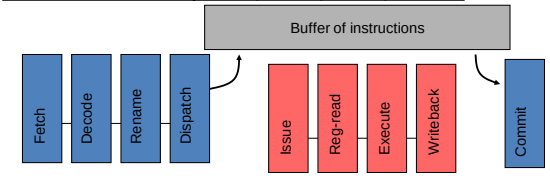
- Standard style: large and cumbersome
- Change layout slightly
  - Columns = stages (dispatch, issue, *etc.*)
  - Rows = instructions
  - Content of boxes = cycles
- For our purposes: issue/exec = 1 cycle
  - Ignore preg read latency, *etc.*
  - Load-use, mul, div, and FP longer

84

84

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`				
`add p2, p3 → p4`				
`xor p4, p5 → p6`				
`ld [p7] → p8`				



85

85

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`				
`add p2, p3 → p4`				
`xor p4, p5 → p6`				
`ld [p7] → p8`				

2-wide
Infinite ROB, IQ, Pregs
Loads: 3 cycles

86

86

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1			
`add p2, p3 → p4`	1			
`xor p4, p5 → p6`				
`ld [p7] → p8`				

Cycle 1:
- Dispatch 1st ld and add

87

87

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1	2	5	
`add p2, p3 → p4`	1			
`xor p4, p5 → p6`	2			
`ld [p7] → p8`	2			

Cycle 2:
- Dispatch xor and 2nd ld
- 1st Ld issues -- also note WB cycle while you do this
  (Note: don't issue if WB ports full)

88

88

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1	2	5	
`add p2, p3 → p4`	1			
`xor p4, p5 → p6`	2			
`ld [p7] → p8`	2	3	6	

Cycle 3:
- add and xor are not ready
- 2nd load is → issue it

89

89

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1	2	5	
`add p2, p3 → p4`	1	5	6	
`xor p4, p5 → p6`	2			
`ld [p7] → p8`	2	3	6	

Cycle 4:
- nothing

Cycle 5:
- add can issue

90

90

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1	2	5	6
`add p2, p3 → p4`	1	5	6	
`xor p4, p5 → p6`	2	6	7	
`ld [p7] → p8`	2	3	6	

Cycle 6:
- 1st load can commit (oldest instruction & finished)
- xor can issue

91

91

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1	2	5	6
`add p2, p3 → p4`	1	5	6	7
`xor p4, p5 → p6`	2	6	7	
`ld [p7] → p8`	2	3	6	

Cycle 7:
- add can commit (oldest instruction & finished)

92

92

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1	2	5	6
`add p2, p3 → p4`	1	5	6	7
`xor p4, p5 → p6`	2	6	7	8
`ld [p7] → p8`	2	3	6	8

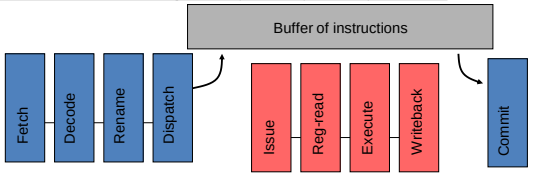
Cycle 8:
- xor and ld can commit (2-wide: can do both at once)

93

93

Out of order pipeline diagrams

Instruction	Disp	Issue	WB	Commit
`ld [p1] → p2`	1	2	5	6
`add p2, p3 → p4`	1	5	6	7
`xor p4, p5 → p6`	2	6	7	8
`ld [p7] → p8`	2	3	6	8



94

94