(High Performance Processors and Systems)

Instruction Level Parallelism

Explicit register renaming

Politecnico di Milano

v1

Alessandro Verosimile <alessandro.verosimile@polimi.it>

Marco D. Santambrogio <marco.santambrogio@polimi.it>

1

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

9

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

10

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

?

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

12

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

15
15

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

?

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

18
18

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

?

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

Advanced Computer Architectures

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

?

22
22

Explicit Register Renaming

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```

?

Explicit Register Renaming

- Tomasulo provides Implicit Register Renaming
  - User registers renamed to reservation station tags

24
24

Explicit Register Renaming

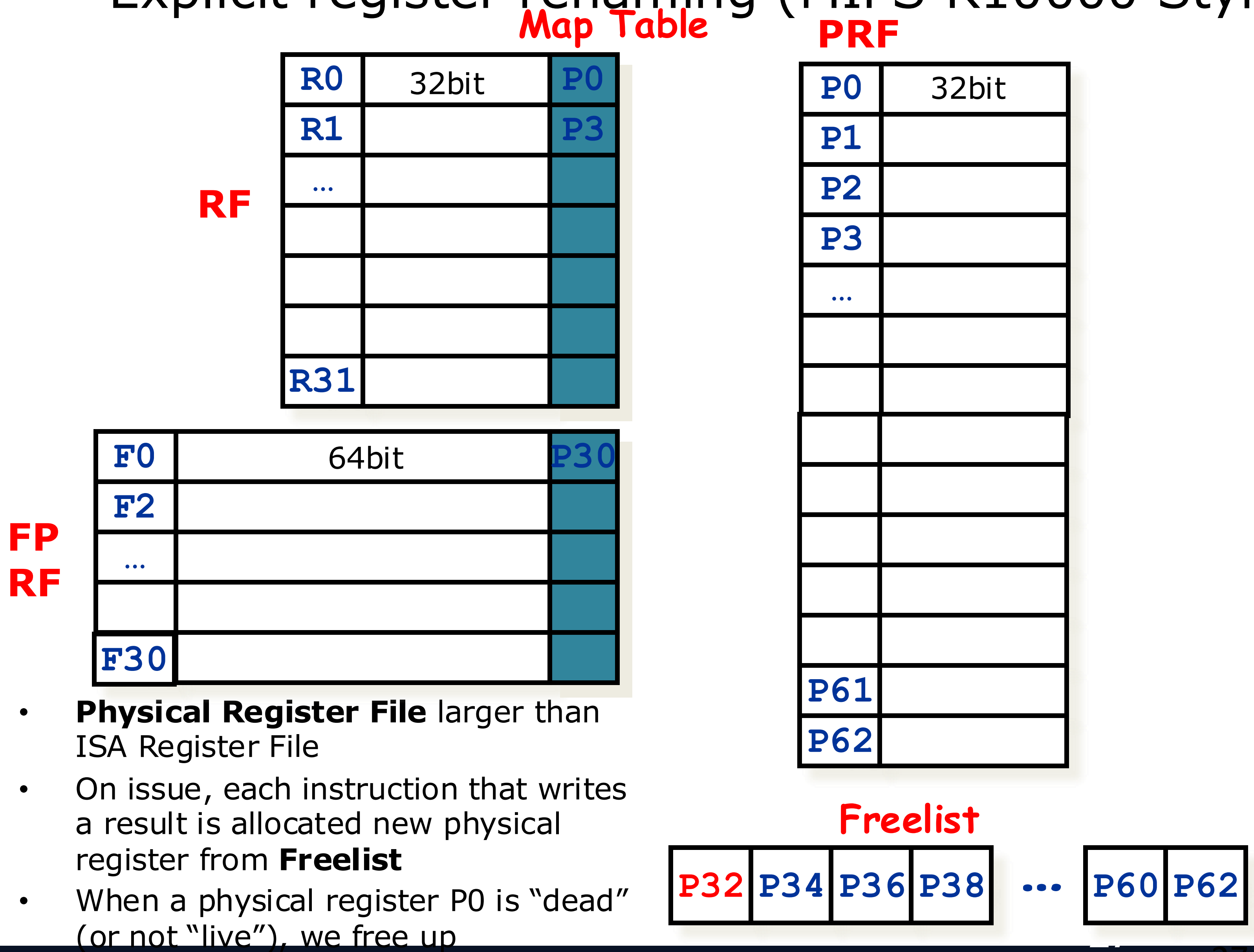
- Tomasulo provides Implicit Register Renaming
  - User registers renamed to reservation station tags
- Now we introduce Explicit Register Renaming:
  - Use physical register file that is larger than number of registers specified by the ISA
- Key insight: Allocate a new physical destination register for every instruction that writes
  - Very similar to a compiler transformation called Static Single Assignment (SSA) form — but in hardware!
  - Removes all chance of WAR or WAW hazards
  - Like Tomasulo, good for allowing full out-of-order completion
  - Like hardware-based dynamic compilation?

25
25

Explicit Register Renaming

```
ld x1, (x3)              ld P1, (Px)
addi x3, x1, #4          addi P2, P1, #4
sub x6, x7, x9           sub P3, Py, Pz
add x3, x3, x6           add P4, P2, P3
ld x6, (x1)              ld P5, (P1)
add x6, x6, x3           add P6, P5, P4
sd x6, (x1)              sd P6, (P1)
ld x6, (x11)             ld P7, (Pw)
```

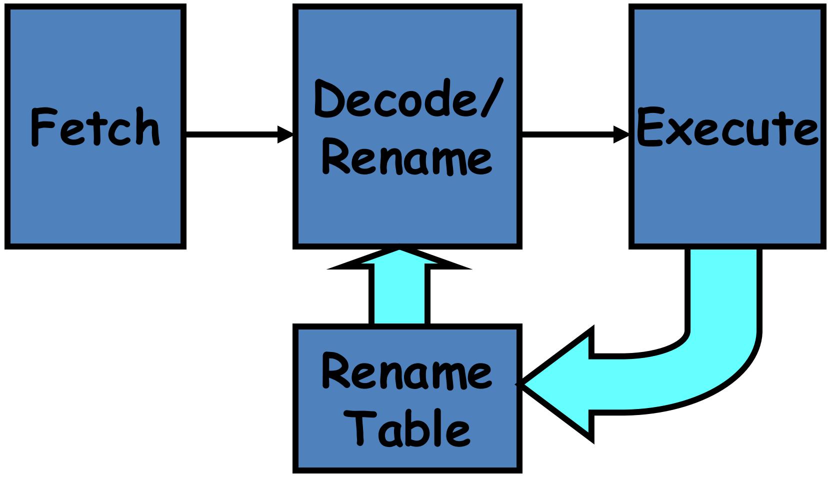
Explicit register renaming (MIPS R10000 Style)



- **Physical Register File** larger than ISA Register File
- On issue, each instruction that writes a result is allocated new physical register from **Freelist**
- When a physical register P0 is "dead" (or not "live"), we free up

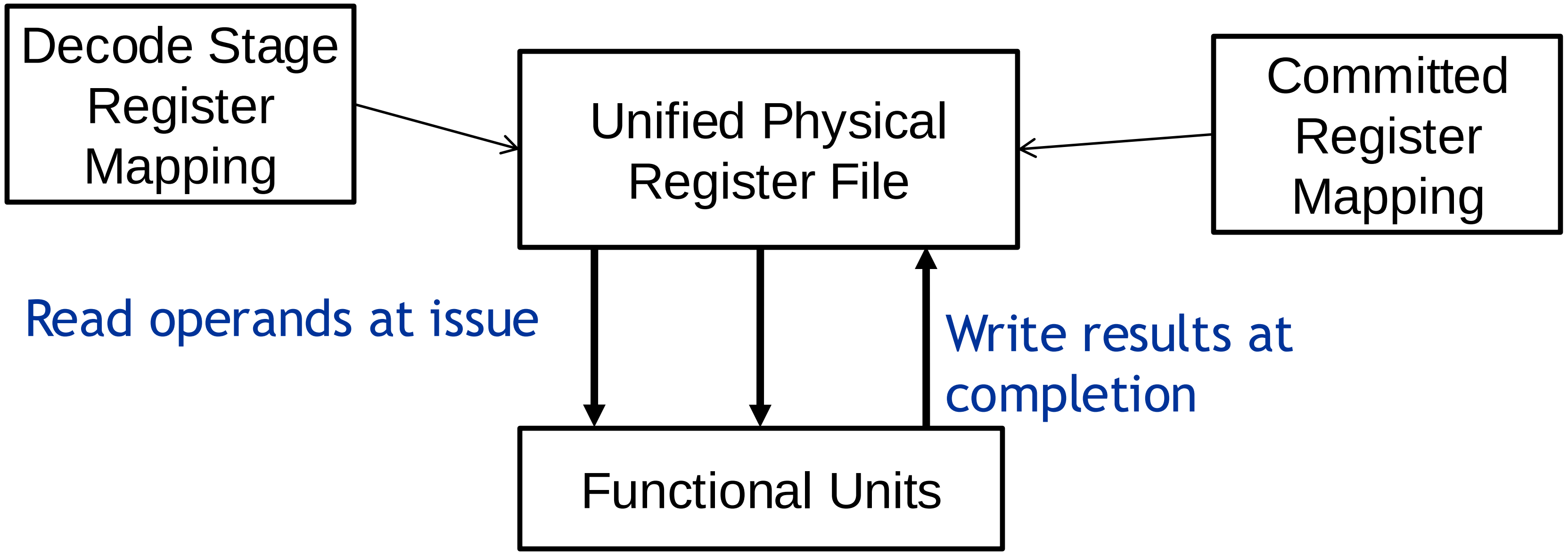
Explicit Register Renaming

- Mechanism? Keep a translation table:
  - ISA register ➔ physical register mapping
  - When register written, replace entry with new register from freelist
  - Physical register becomes free when not used by any active instructions



Unified Physical Register File

- Rename all architectural registers into a single physical register file during decode, no register values read
- Functional units read and write from single unified register file holding committed and temporary registers in execution
- Commit only updates mapping of architectural register to physical register, no data movement


Decode Stage Register Mapping
Unified Physical Register File
Committed Register Mapping
Read operands at issue
Write results at completion
Functional Units

HW register renaming

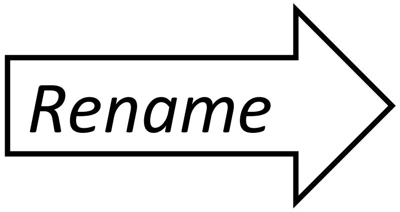
- Renaming map: simple data structure that supplies the physical register number of the register that currently corresponds to the requested architectural register
- Instruction commit: update permanently the renaming table to indicate that the physical register holding the destination value corresponds to the actual architectural register
- Use ROB to enforce in-order commit

30
30

Lifetime of Physical Registers

- Physical register file holds committed and speculative values
- Physical registers decoupled from ROB entries (no data in ROB)

```
ld x1, (x3)
addi x3, x1, #4
sub x6, x7, x9
add x3, x3, x6
ld x6, (x1)
add x6, x6, x3
sd x6, (x1)
ld x6, (x11)
```



```
ld P1, (Px)
addi P2, P1, #4
sub P3, Py, Pz
add P4, P2, P3
ld P5, (P1)
add P6, P5, P4
sd P6, (P1)
ld P7, (Pw)
```

- When can we reuse a physical register?

When next writer of same architectural register commits

Physical Register Management

Rename Table

x0	
x1	P8
x2	
x3	P7
x4	
x5	
x6	P5
x7	P6

Physical Regs

P0		
P1		
P2		
P3		
P4		
P5	<x6>	p
P6	<x7>	p
P7	<x3>	p
P8	<x1>	p
Pn		

Free List

P0
P1
P3
P2
P4

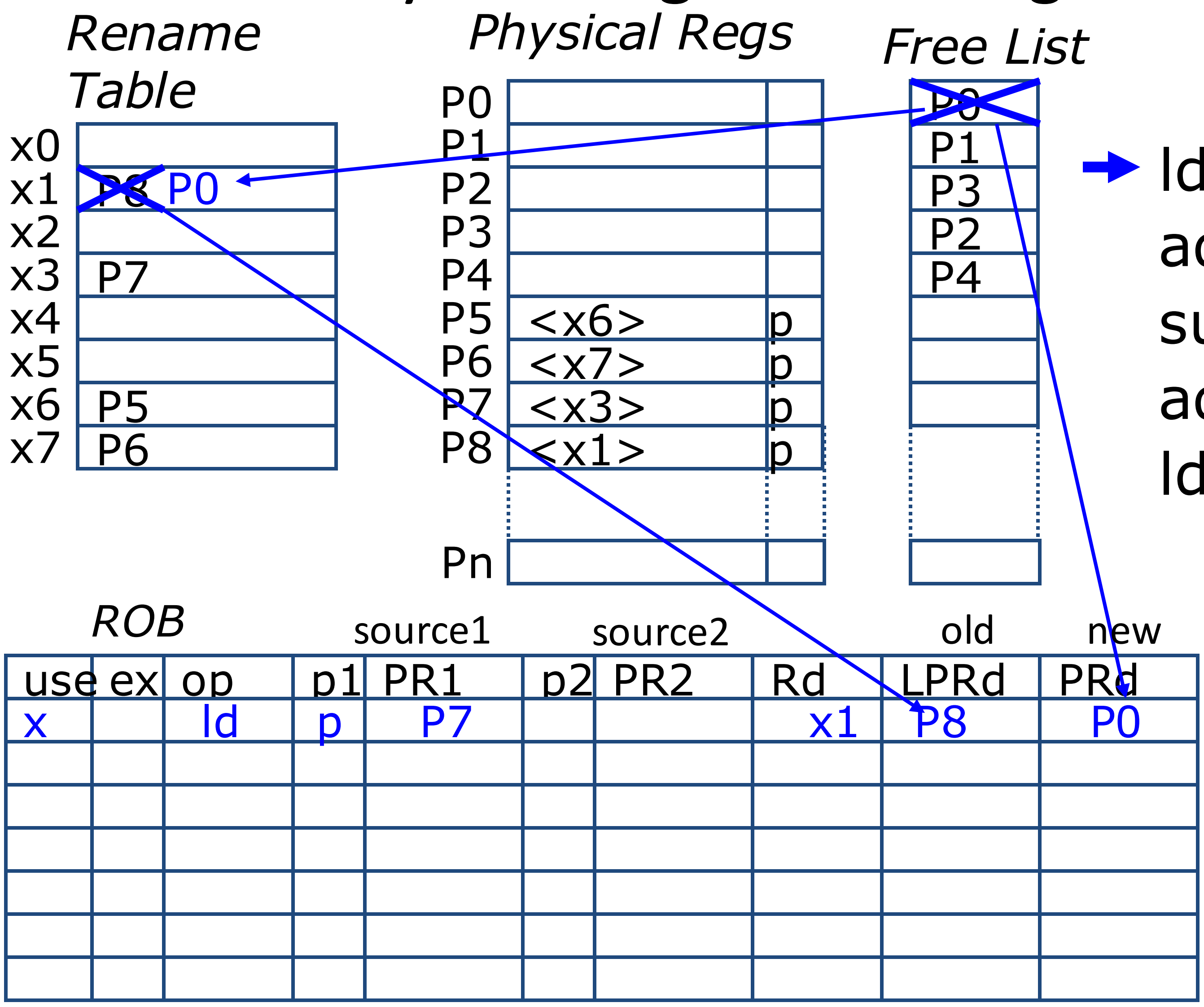
ld x1, 0(x3)
addi x3, x1, #4
sub x6, x7, x6
add x3, x3, x6
ld x6, 0(x1)

ROB

				source1		source2		old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd

(LPRd requires third read port on Rename Table for each instruction)

Physical Register Management



Rename Table

x0	
x1	~~P8~~ P0
x2	
x3	P7
x4	
x5	
x6	P5
x7	P6

Physical Regs

P0		
P1		
P2		
P3		
P4		
P5	<x6>	p
P6	<x7>	p
P7	<x3>	p
P8	<x1>	p
Pn		

Free List

~~P0~~
P1
P3
P2
P4

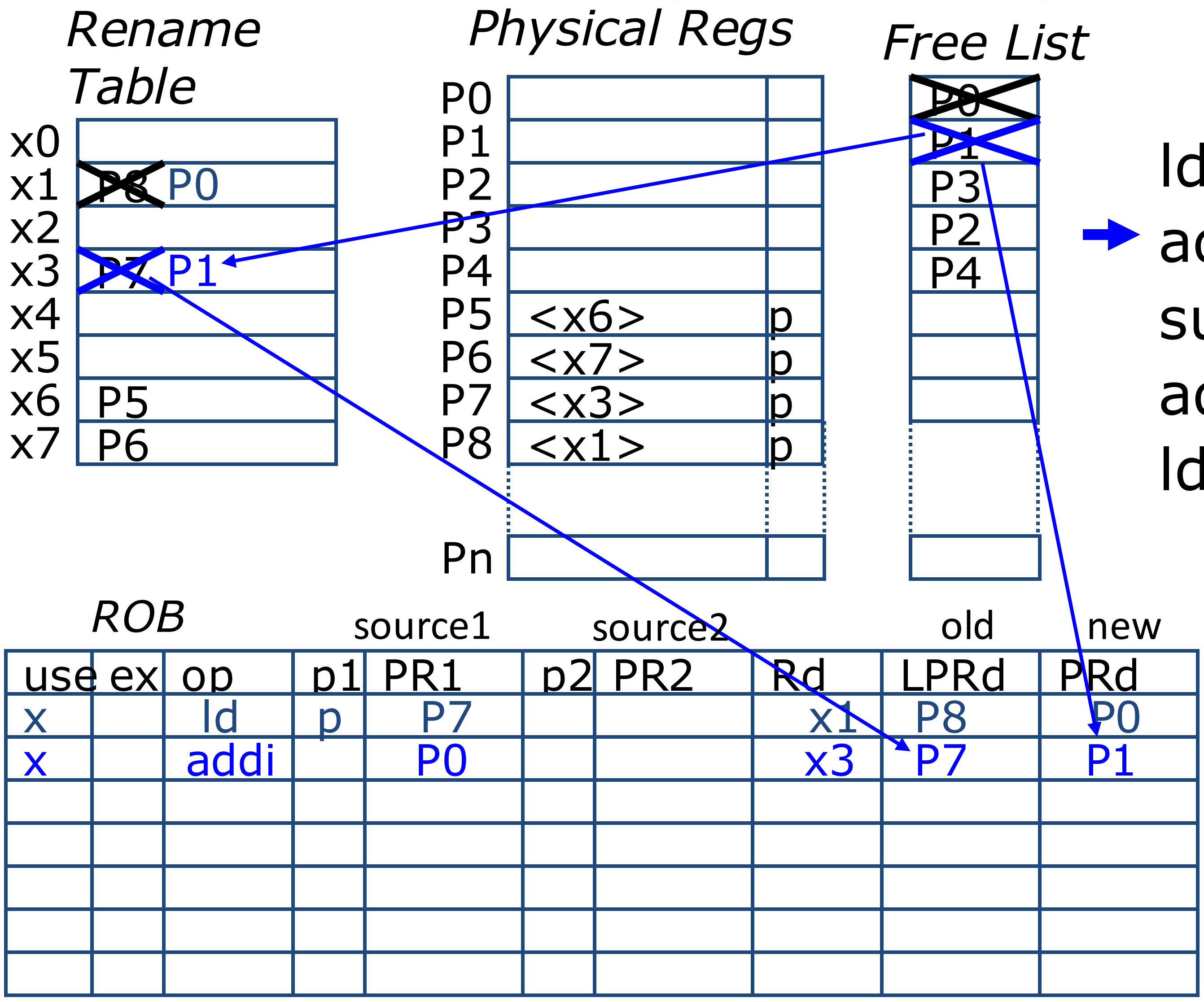
ROB

			source1		source2			old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		ld	p	P7			x1	P8	P0

→ ld x1, 0(x3)
addi x3, x1, #4
sub x6, x7, x6
add x3, x3, x6
ld x6, 0(x1)

Advanced Computer Architectures

Physical Register Management



Rename Table

x0	
x1	~~P8~~ P0
x2	
x3	~~P7~~ P1
x4	
x5	
x6	P5
x7	P6

Physical Regs

P0		
P1		
P2		
P3		
P4		
P5	<x6>	p
P6	<x7>	p
P7	<x3>	p
P8	<x1>	p
Pn		

Free List

~~P0~~
~~P1~~
P3
P2
P4

ROB

			source1		source2			old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		ld	p	P7			x1	P8	P0
x		addi		P0			x3	P7	P1

ld x1, 0(x3)

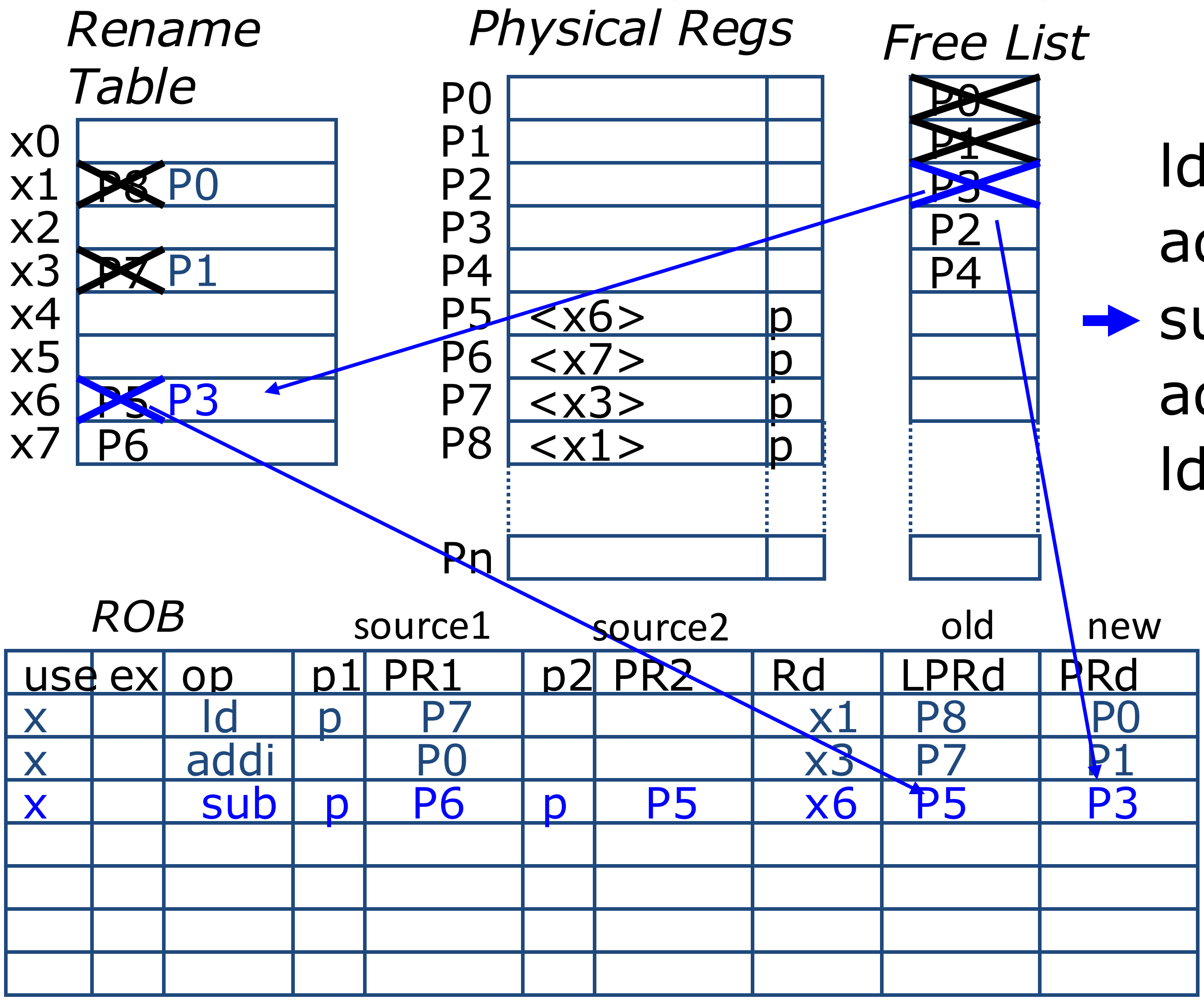
→ addi x3, x1, #4

sub x6, x7, x6

add x3, x3, x6

ld x6, 0(x1)

Physical Register Management

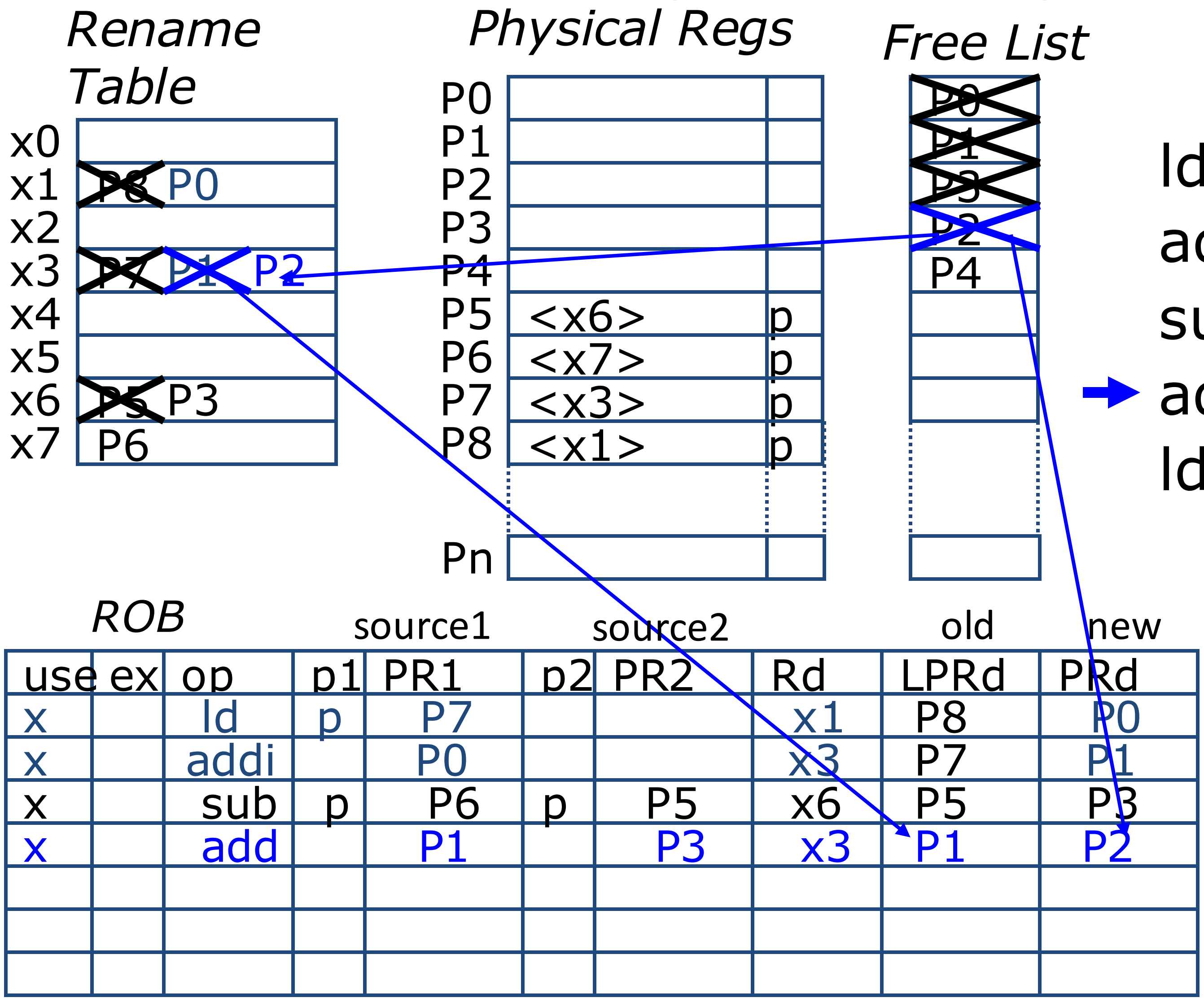


			source1		source2			old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		ld	p	P7			x1	P8	P0
x		addi		P0			x3	P7	P1
x		sub	p	P6	p	P5	x6	P5	P3

ld x1, 0(x3)
addi x3, x1, #4
➡ sub x6, x7, x6
add x3, x3, x6
ld x6, 0(x1)

35

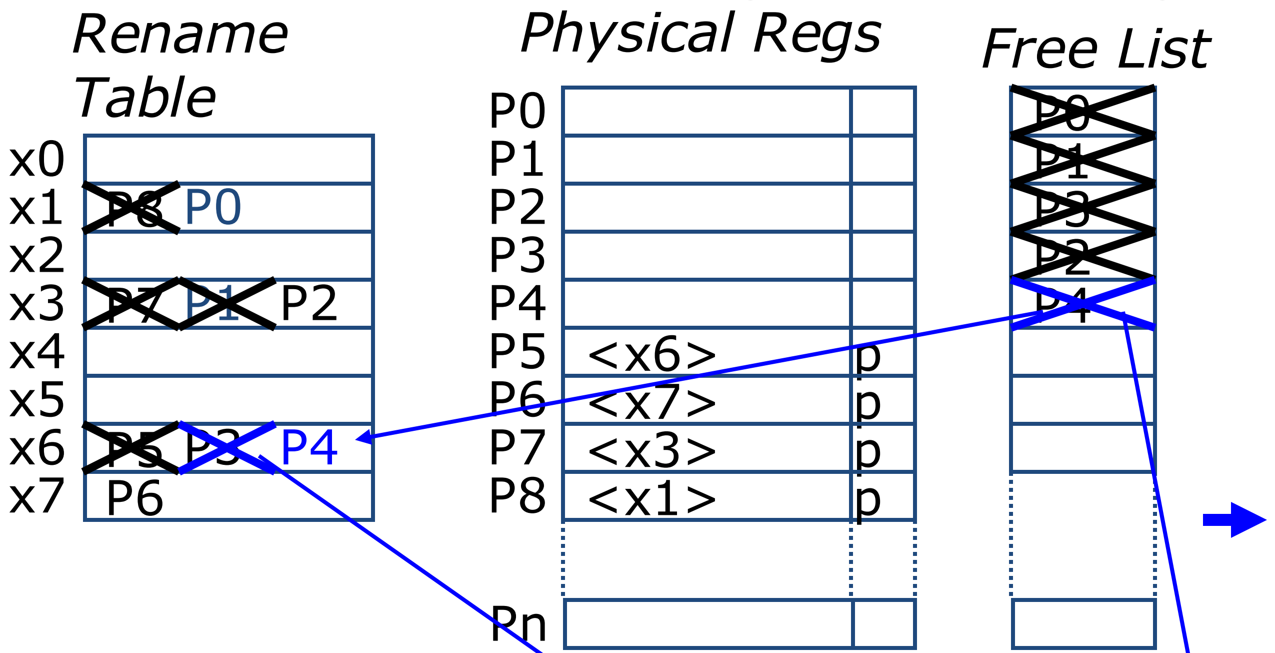
Physical Register Management



				source1		source2		old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		ld	p	P7			x1	P8	P0
x		addi		P0			x3	P7	P1
x		sub	p	P6	p	P5	x6	P5	P3
x		add		P1		P3	x3	P1	P2

ld x1, 0(x3)
addi x3, x1, #4
sub x6, x7, x6
→ add x3, x3, x6
ld x6, 0(x1)

Physical Register Management

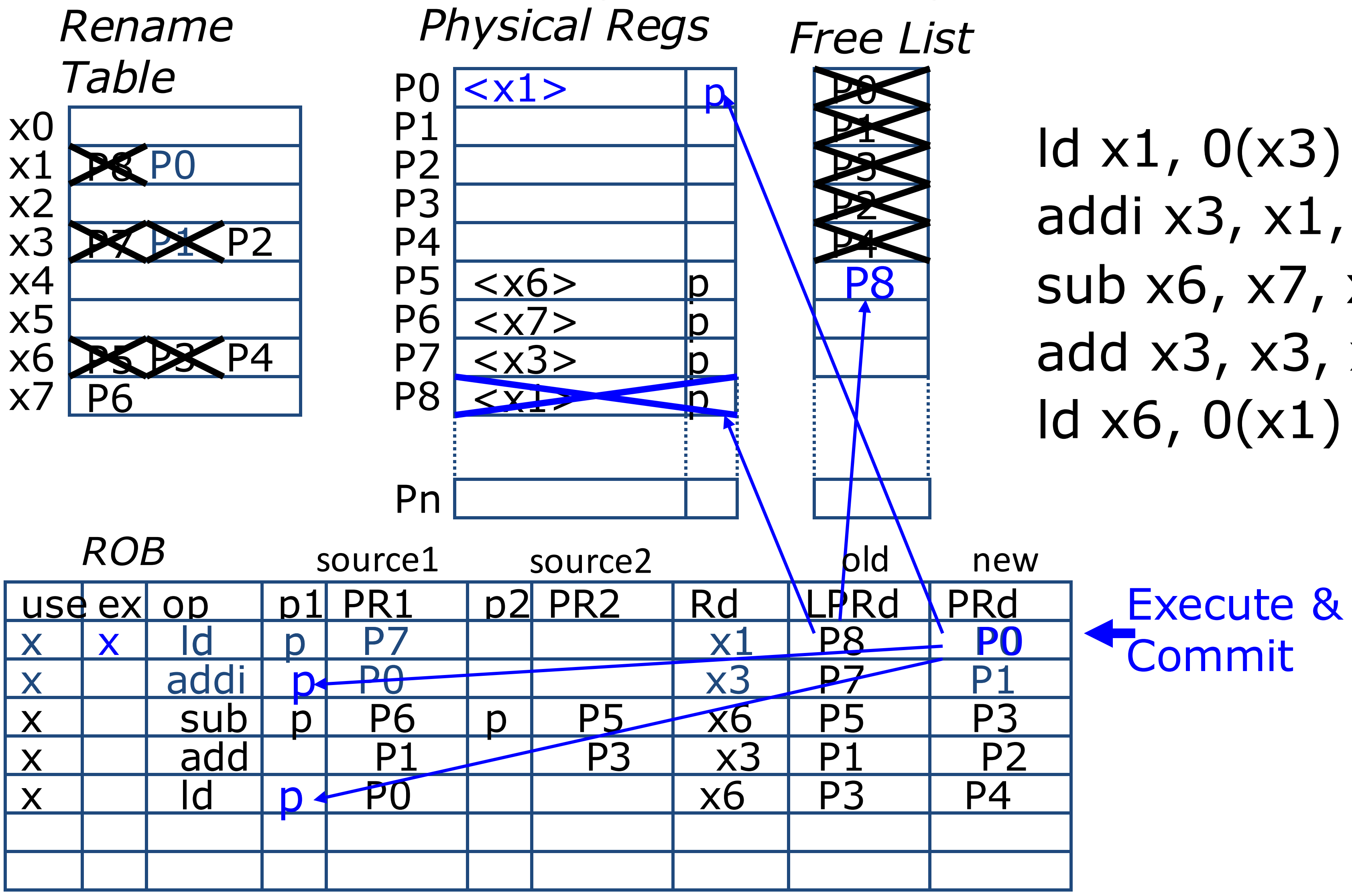


```
ld x1, 0(x3)
addi x3, x1, #4
sub x6, x7, x6
add x3, x3, x6
➡ ld x6, 0(x1)
```

ROB

			source1		source2			old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		ld	p	P7			x1	P8	P0
x		addi		P0			x3	P7	P1
x		sub	p	P6	p	P5	x6	P5	P3
x		add		P1		P3	x3	P1	P2
x		ld		P0			x6	P3	P4

Physical Register Management



Rename Table

x0	
x1	~~P8~~ P0
x2	
x3	~~P7~~ ~~P1~~ P2
x4	
x5	
x6	~~P5~~ ~~P3~~ P4
x7	P6

Physical Regs

P0	<x1>	p
P1		
P2		
P3		
P4		
P5	<x6>	p
P6	<x7>	p
P7	<x3>	p
P8	~~<x1>~~	~~p~~
Pn		

Free List

~~P0~~
~~P1~~
~~P3~~
~~P2~~
~~P4~~
P8

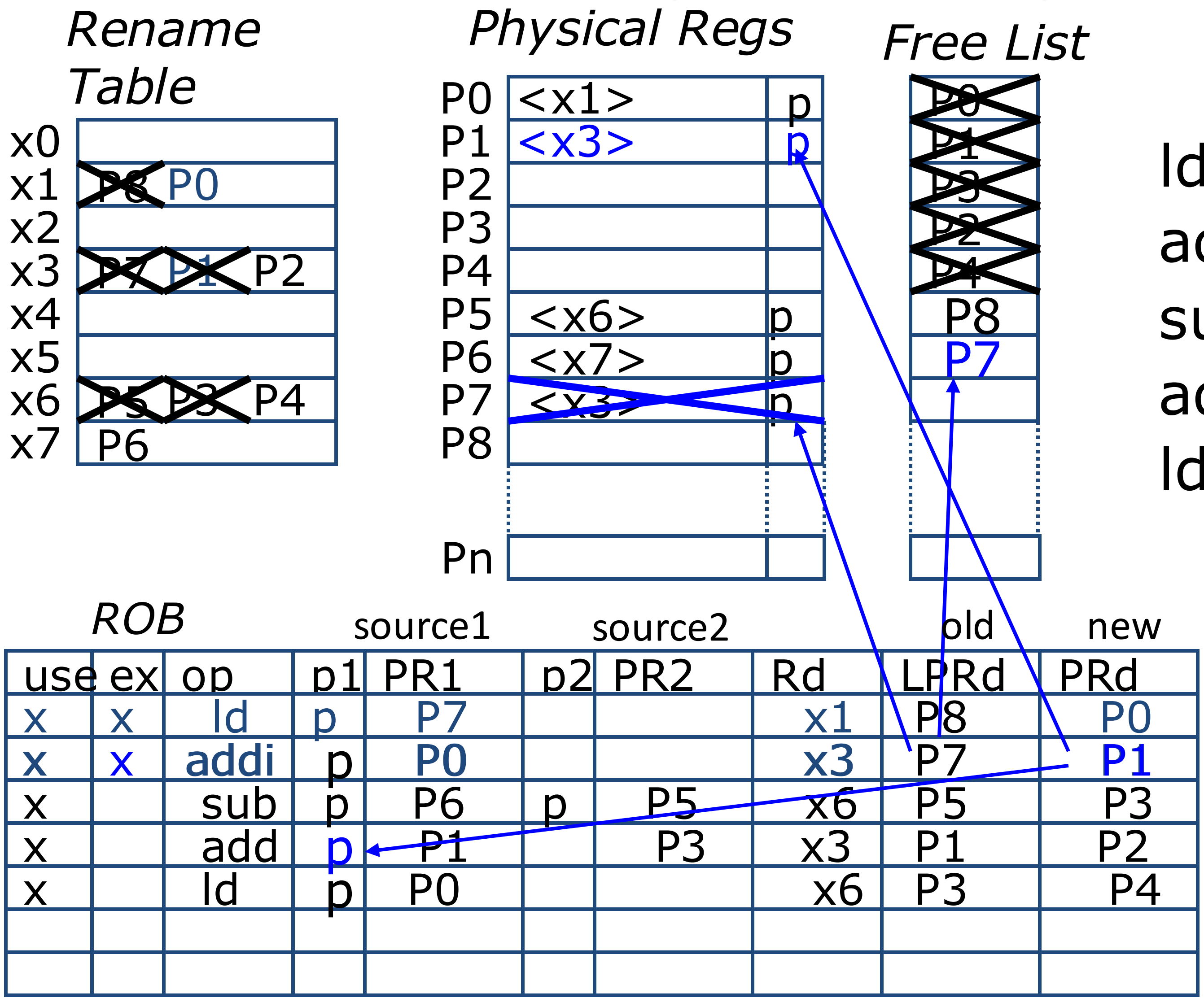
ld x1, 0(x3)
addi x3, x1, #4
sub x6, x7, x6
add x3, x3, x6
ld x6, 0(x1)

ROB

			source1		source2			old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x	x	ld	p	P7			x1	P8	P0
x		addi	p	P0			x3	P7	P1
x		sub	p	P6	p	P5	x6	P5	P3
x		add		P1		P3	x3	P1	P2
x		ld	p	P0			x6	P3	P4

Execute & Commit

Physical Register Management



Rename Table

x0	
x1	~~P8~~ P0
x2	
x3	~~P7~~ ~~P1~~ P2
x4	
x5	
x6	~~P5~~ ~~P3~~ P4
x7	P6

Physical Regs

P0	<x1>	p
P1	<x3>	p
P2		
P3		
P4		
P5	<x6>	p
P6	<x7>	p
P7	~~<x3>~~	~~p~~
P8		
Pn		

Free List

~~P0~~
~~P1~~
~~P3~~
~~P2~~
~~P4~~
P8
P7

ld x1, 0(x3)
addi x3, x1, #4
sub x6, x7, x6
add x3, x3, x6
ld x6, 0(x1)

ROB

			source1		source2			old	new
use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x	x	ld	p	P7			x1	P8	P0
x	x	addi	p	P0			x3	P7	P1
x		sub	p	P6	p	P5	x6	P5	P3
x		add	p	P1		P3	x3	P1	P2
x		ld	p	P0			x6	P3	P4

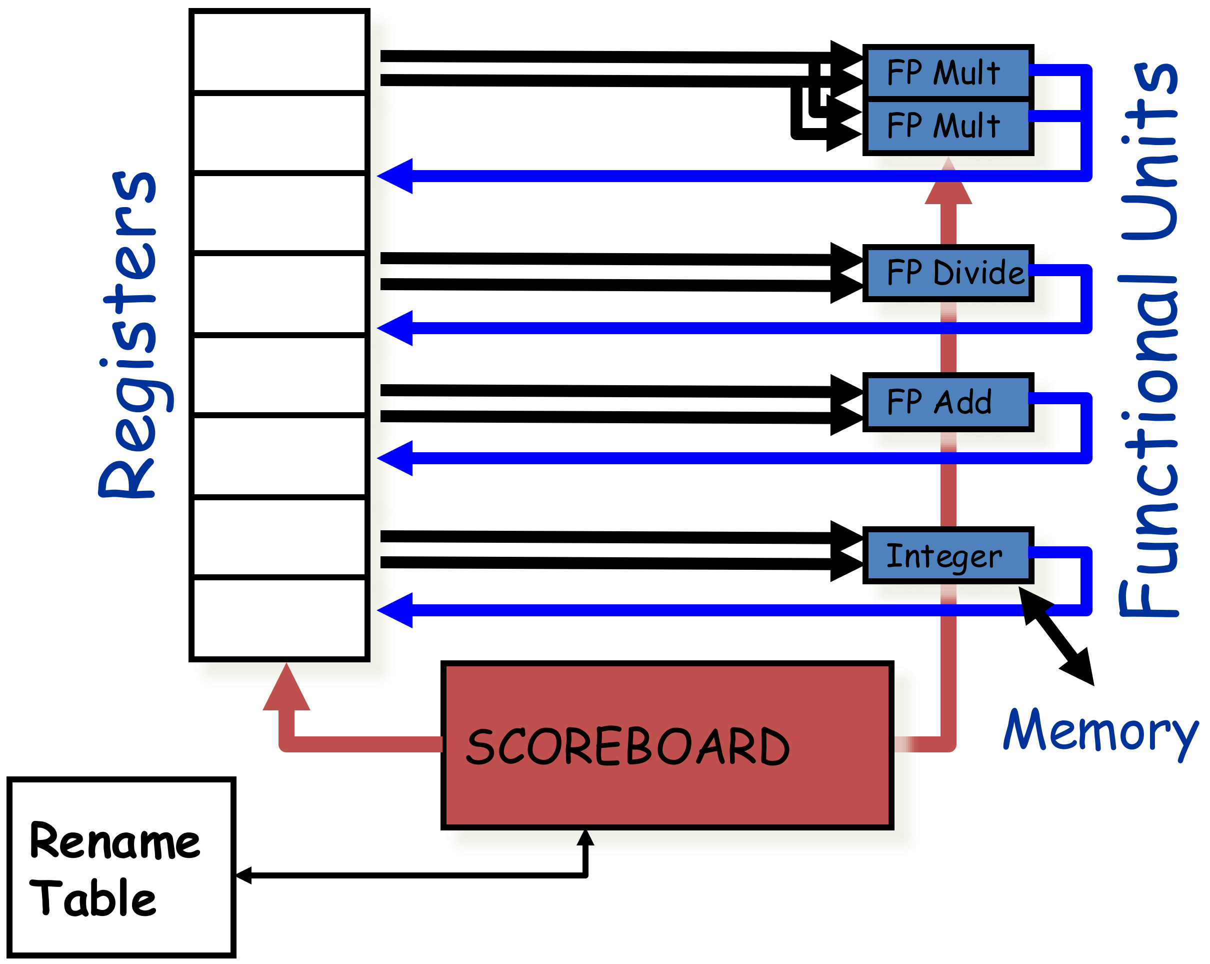
Execute & Commit

Explicit Register Renaming

- Tomasulo provides Implicit Register Renaming
  - User registers renamed to reservation station tags
- Explicit Register Renaming:
  - Use physical register file that is larger than number of registers specified by ISA
- Keep a translation table:
  - ISA register => physical register mapping
  - When register is written, replace table entry with new register from freelist.
  - Physical register becomes free when not being used by any instructions in progress.
- Pipeline can be exactly like "standard" DLX pipeline
  - IF, ID, EX, etc….
- Advantages:
  - Removes all WAR and WAW hazards
  - Like Tomasulo, good for allowing full out-of-order completion
  - Allows data to be fetched from a single register file
  - Makes speculative execution/precise interrupts easier:
    - All that needs to be "undone" for precise break point is to undo the table mappings

40
40

Question: Can we use explicit register renaming with scoreboard?



Stages of Scoreboard Control With Explicit Renaming

- Issue—decode instructions & check for structural hazards & allocate new physical register for result
  - Instructions issued in program order (for hazard checking)
  - Don't issue if no free physical registers
  - Don't issue if structural hazard

42

Stages of Scoreboard Control With Explicit Renaming

- Issue—decode instructions & check for structural hazards & allocate new physical register for result
  - Instructions issued in program order (for hazard checking)
  - Don't issue if no free physical registers
  - Don't issue if structural hazard
- Read operands—wait until no hazards, read operands
  - All real dependencies (RAW hazards) resolved in this stage, since we wait for instructions to write back data.

Stages of Scoreboard Control With Explicit Renaming

- Issue—decode instructions & check for structural hazards & allocate new physical register for result
  - Instructions issued in program order (for hazard checking)
  - Don't issue if no free physical registers
  - Don't issue if structural hazard
- Read operands—wait until no hazards, read operands
  - All real dependencies (RAW hazards) resolved in this stage, since we wait for instructions to write back data.
- Execution—operate on operands
  - The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard

Stages of Scoreboard Control With Explicit Renaming

- Issue—decode instructions & check for structural hazards & allocate new physical register for result
  - Instructions issued in program order (for hazard checking)
  - Don't issue if no free physical registers
  - Don't issue if structural hazard
- Read operands—wait until no hazards, read operands
  - All real dependencies (RAW hazards) resolved in this stage, since we wait for instructions to write back data.
- Execution—operate on operands
  - The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard
- Write result —finish execution

Stages of Scoreboard Control With Explicit Renaming

- Issue—decode instructions & check for structural hazards & allocate new physical register for result
  - Instructions issued in program order (for hazard checking)
  - Don’t issue if no free physical registers
  - Don’t issue if structural hazard
- Read operands—wait until no hazards, read operands
  - All real dependencies (RAW hazards) resolved in this stage, since we wait for instructions to write back data.
- Execution—operate on operands
  - The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard
- Write result —finish execution
- Note: No checks for WAR or WAW hazards!

46
46

Scoreboard Example

Instruction status:

Instruction		j	k	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2				
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	No								
	Add	No								
	Divide	No								

Register Rename and Result

Clock	*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
FU	P0	P2	P4	P6	P8	P10	P12		P30

Initialized Rename Table – registers from P32 in the free list

47

Renamed Scoreboard 1

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1			
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	Yes	Load	P32		R2				Yes
	Int2	No								
	Mult1	No								
	Add	No								
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
1	*FU*	P0	P2	P4	P32	P8	P10	P12		P30

Each instruction allocates free register

48

Renamed Scoreboard 2

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2		
LD	F2	45+	R3	2			
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	Yes	Load	P32		R2				Yes
	Int2	Yes	Load	P34		R3				Yes
	Mult1	No								
	Add	No								
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
2	*FU*	P0	P34	P4	P32	P8	P10	P12		P30

49

Renamed Scoreboard 3

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	
LD	F2	45+	R3	2	3		
MULTD	F0	F2	F4	3			
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	Yes	Load	P32		R2				Yes
	Int2	Yes	Load	P34		R3				Yes
	Mult1	Yes	Multd	P36	P34	P4	Int2		No	Yes
	Add	No								
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
3	*FU*	P36	P34	P4	P32	P8	P10	P12		P30

50

Renamed Scoreboard 4

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	
MULTD	F0	F2	F4	3			
SUBD	F8	F6	F2	4			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	Yes	Load	P34		R3				Yes
	Mult1	Yes	Multd	P36	P34	P4	Int2		No	Yes
	Add	Yes	Sub	P38	P32	P34		Int2	Yes	No
	Divide	No								

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
4	*FU*	P36	P34	P4	P32	P38	P10	P12		P30

Renamed Scoreboard 5

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3			
SUBD	F8	F6	F2	4			
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
	Add	Yes	Sub	P38	P32	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
5	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

Renamed Scoreboard 6

Instruction status:

Instruction		j	k	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6		
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
10	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
2	Add	Yes	Sub	P38	P32	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
6	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

Renamed Scoreboard 7

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6		
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
9	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
1	Add	Yes	Sub	P38	P32	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
7	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

54

Renamed Scoreboard 8

Instruction status:

Instruction		j	k	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
8	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
0	Add	Yes	Sub	P38	P32	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
8	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

Renamed Scoreboard 9

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2				

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Int1	No								
	Int2	No								
7	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
9	*FU*	P36	P34	P4	P32	P38	P40	P12		P30

56

Renamed Scoreboard 10

Instruction status:

Instruction		j	k	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10			

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
6	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
10	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 10

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10			

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Int1	No								
	Int2	No								
6	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

WAR Hazard gone!

Register Rename and Result

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
10	FU	P36	P34	P4	P42	P38	P40	P12		P30

Notice that P32 not listed in Rename Table
Still live. Must not be reallocated by accident

Renamed Scoreboard 11

Instruction status:

Instruction		j	k	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10	11		

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
5	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
2	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
11	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

59

Renamed Scoreboard 12

Instruction status:

Instruction		j	k	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10	11		

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
4	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
1	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
12	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 13

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10	11	13	

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Int1	No								
	Int2	No								
3	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
0	Add	Yes	Addd	P42	P38	P34			Yes	Yes
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
13	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 14

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10	11	13	14

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
2	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
14	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

62

Renamed Scoreboard 15

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6		
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10	11	13	14

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
1	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
15	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Renamed Scoreboard 16

Instruction status:

Instruction		j	k	Issue	Read Oper	Exec Comp	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	16	
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10	11	13	14

Functional unit status:

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Int1	No								
	Int2	No								
0	Mult1	Yes	Multd	P36	P34	P4			Yes	Yes
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		No	Yes

Register Rename and Result

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
16	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

64

Renamed Scoreboard 17

Instruction status:

Instruction		j	k	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	16	17
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5			
ADDD	F6	F8	F2	10	11	13	14

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	No								
	Add	No								
	Divide	Yes	Divd	P40	P36	P32	Mult1		Yes	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
17	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

65

Renamed Scoreboard 18

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	16	17
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5	18		
ADDD	F6	F8	F2	10	11	13	14

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU Qk*	*Fj? Rj*	*Fk? Rk*
	Int1	No								
	Int2	No								
	Mult1	No								
	Add	No								
40	Divide	Yes	Divd	P40	P36	P32	Mult1		Yes	Yes

Register Rename and Result

Clock		*F0*	*F2*	*F4*	*F6*	*F8*	*F10*	*F12*	...	*F30*
18	*FU*	P36	P34	P4	P42	P38	P40	P12		P30

Compare to *Previous* Architectures

Instruction status:

					Read	*Exec*	*Write*		*Exec*	*Write*
Instruction		*j*	*k*	*Issue*	*Oper*	*Comp*	*Result*	*Issue*	*Comp*	*Result*
LD	F6	34+	R2	1	2	3	4	1	3	4
LD	F2	45+	R3	5	6	7	8	2	4	5
MULTD	F0	F2	F4	6	9	19	20	3	15	16
SUBD	F8	F6	F2	7	9	11	12	4	7	8
DIVD	F10	F0	F6	8	21	61	62	5	56	57
ADDD	F6	F8	F2	13	14	16	22	6	10	11

Compare to *Previous* Architectures

Instruction status:

					Read	*Exec*	*Write*
Instruction		*j*	*k*	*Issue*	*Oper*	*Comp*	*Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

	Exec	*Write*
Issue	*Comp*	*Result*
1	3	4
2	4	5
3	15	16
4	7	8
5	56	57
6	10	11

Instruction status:

					Read	*Exec*	*Write*
Instruction		*j*	*k*	*Issue*	*Oper*	*Comp*	*Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	2	3	4	5
MULTD	F0	F2	F4	3	6	16	17
SUBD	F8	F6	F2	4	6	8	9
DIVD	F10	F0	F6	5	18	58	59
ADDD	F6	F8	F2	10	11	13	14

Register renaming vs. ROB

- Instruction commit simpler than with ROB;
- Deallocating registers more complex;
- Dynamic mapping of architectural to physical registers complicates design and debugging;
- Used in PowerPC603/604, Pentium II-III-4, MIPS 10000/12000, Alpha 21264; Sandy-Bridge
  - 20 to 80 registers are added.

69
69

Summary

- Explicit Renaming: more physical registers than needed by ISA.
  - Rename table: tracks current association between architectural registers and physical registers
  - Uses a translation table to perform compiler-like transformation on the fly
- With Explicit Renaming:
  - All registers concentrated in single register file
  - Can utilize bypass network that looks more like 5-stage pipeline
  - Introduces a register-allocation problem
    - Need to handle branch misprediction and precise exceptions differently, but ultimately makes things simpler
- For precise exceptions and branch prediction:
  - Clearly need something like reorder buffer

70

What about Multiple Issue?

Advanced Computer Architectures

(High Performance Processors and Systems)

Instruction Level Parallelism

Explicit register renaming

Politecnico di Milano

v1