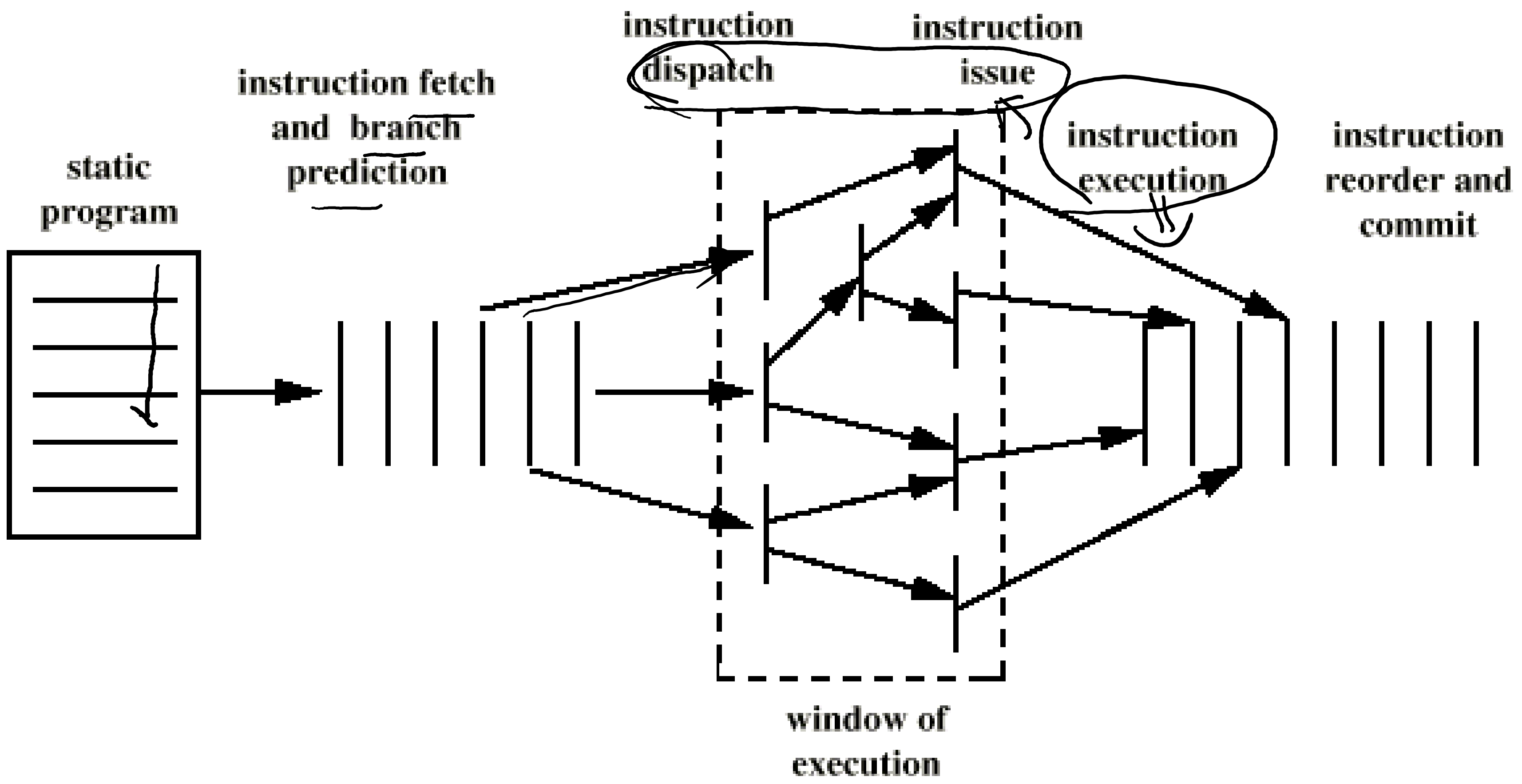
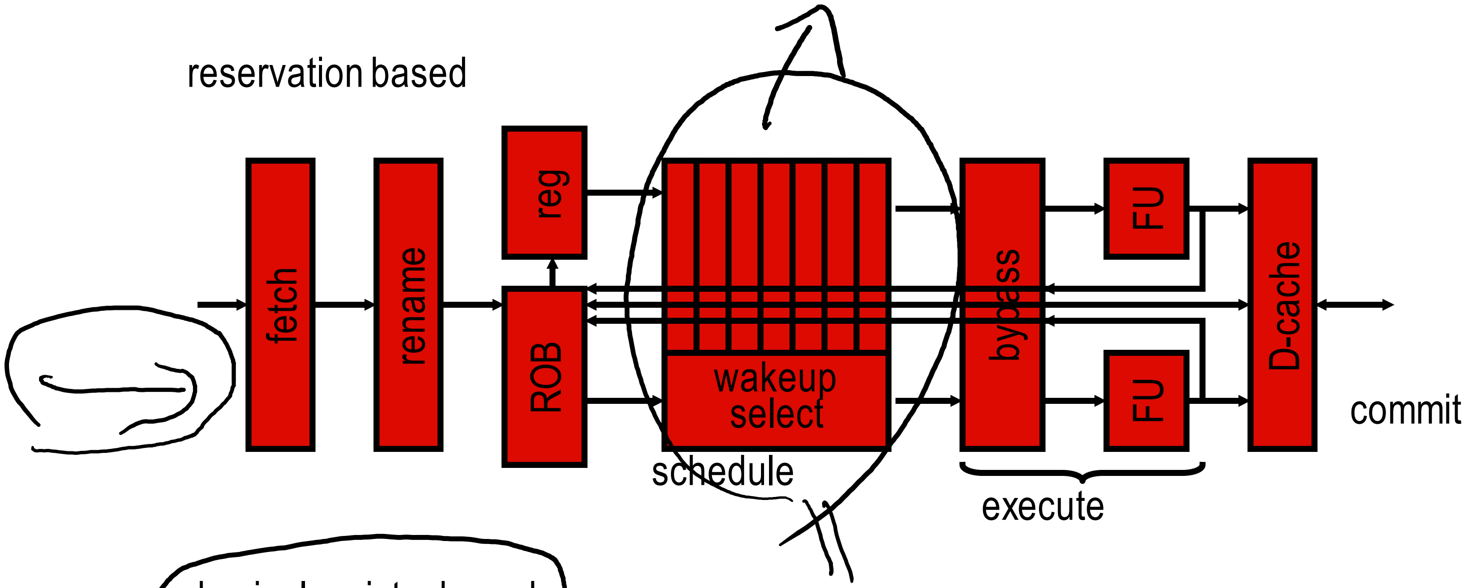
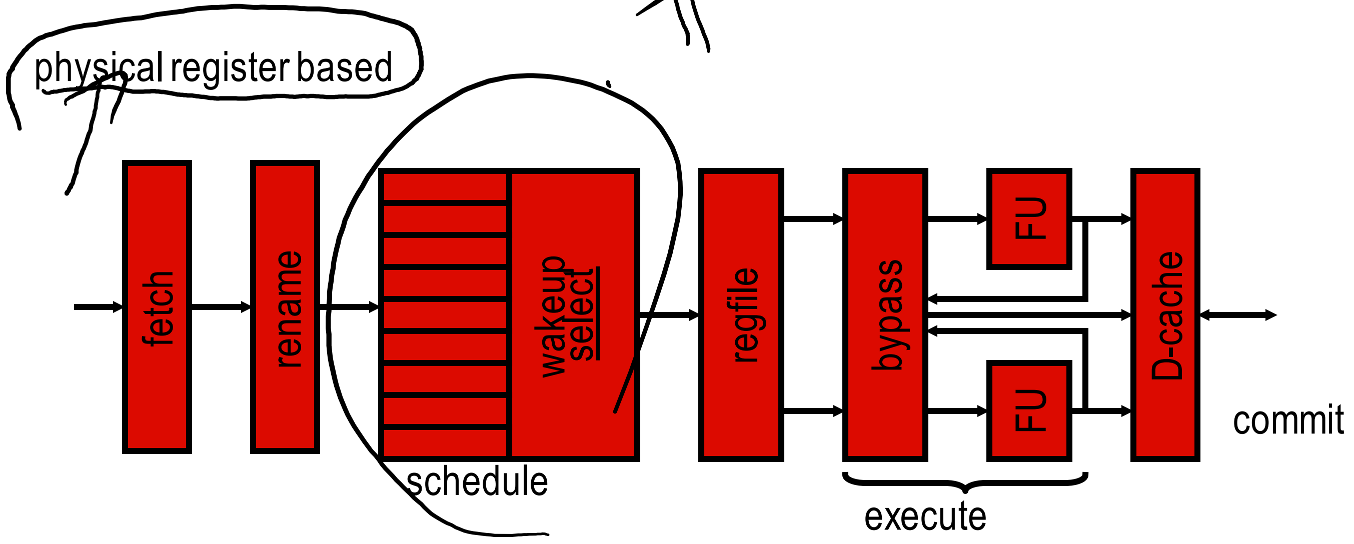
Lecture 13: Dynamic Scheduling w/ Renaming

View of Superscalar Execution



Generic Superscalar Processor Models



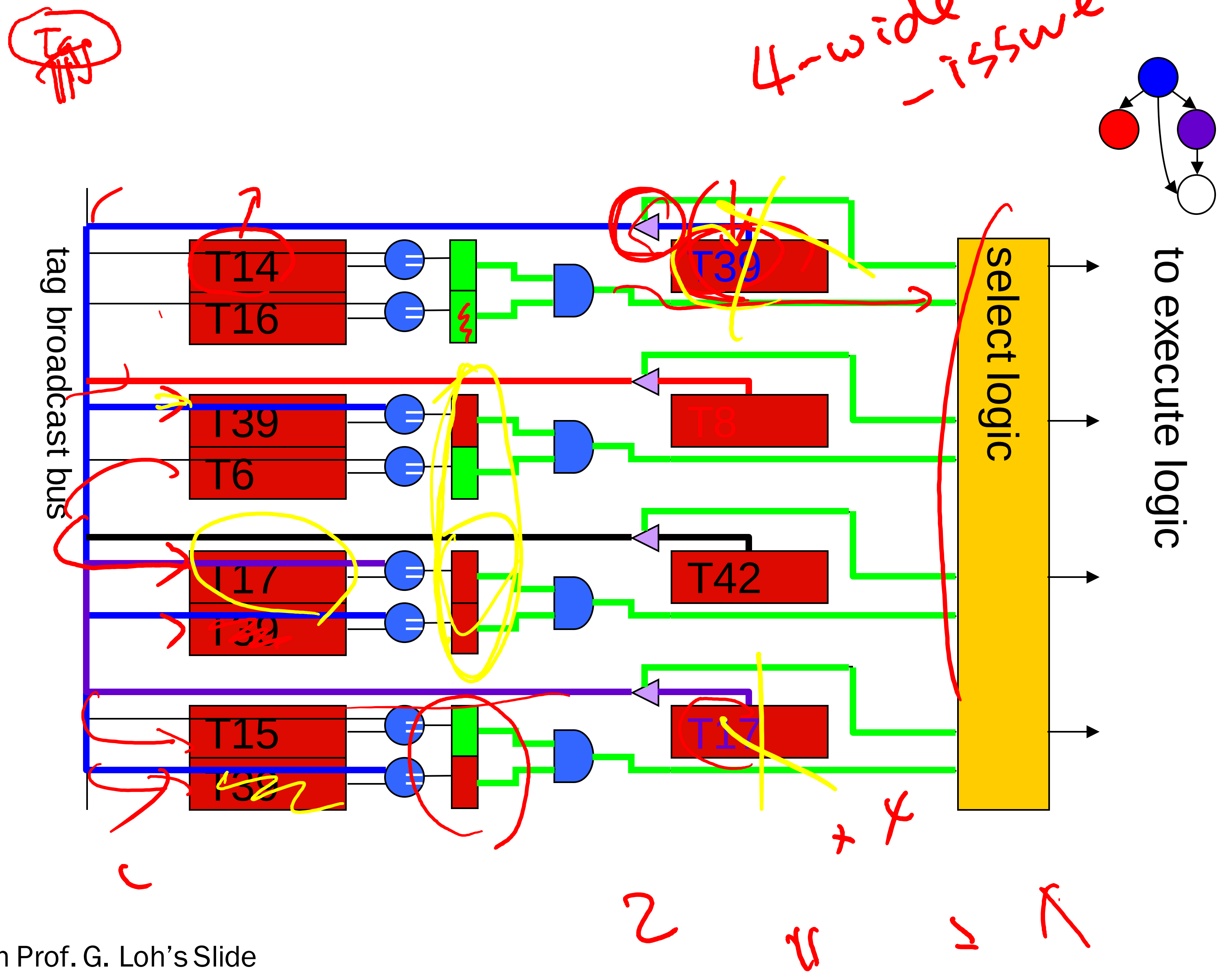


Instr Scheduling: Wakeup & Select

- wakeup logic
  - ✓ to notify the resolution of data dependency of input operands
  - ✓ wake up instructions w/ zero input dependency
- select logic
  - ✓ choose and fire ready instructions
  - ✓ deal with structure hazard
- wakeup-select is likely on the critical path
  - ✓ associative match

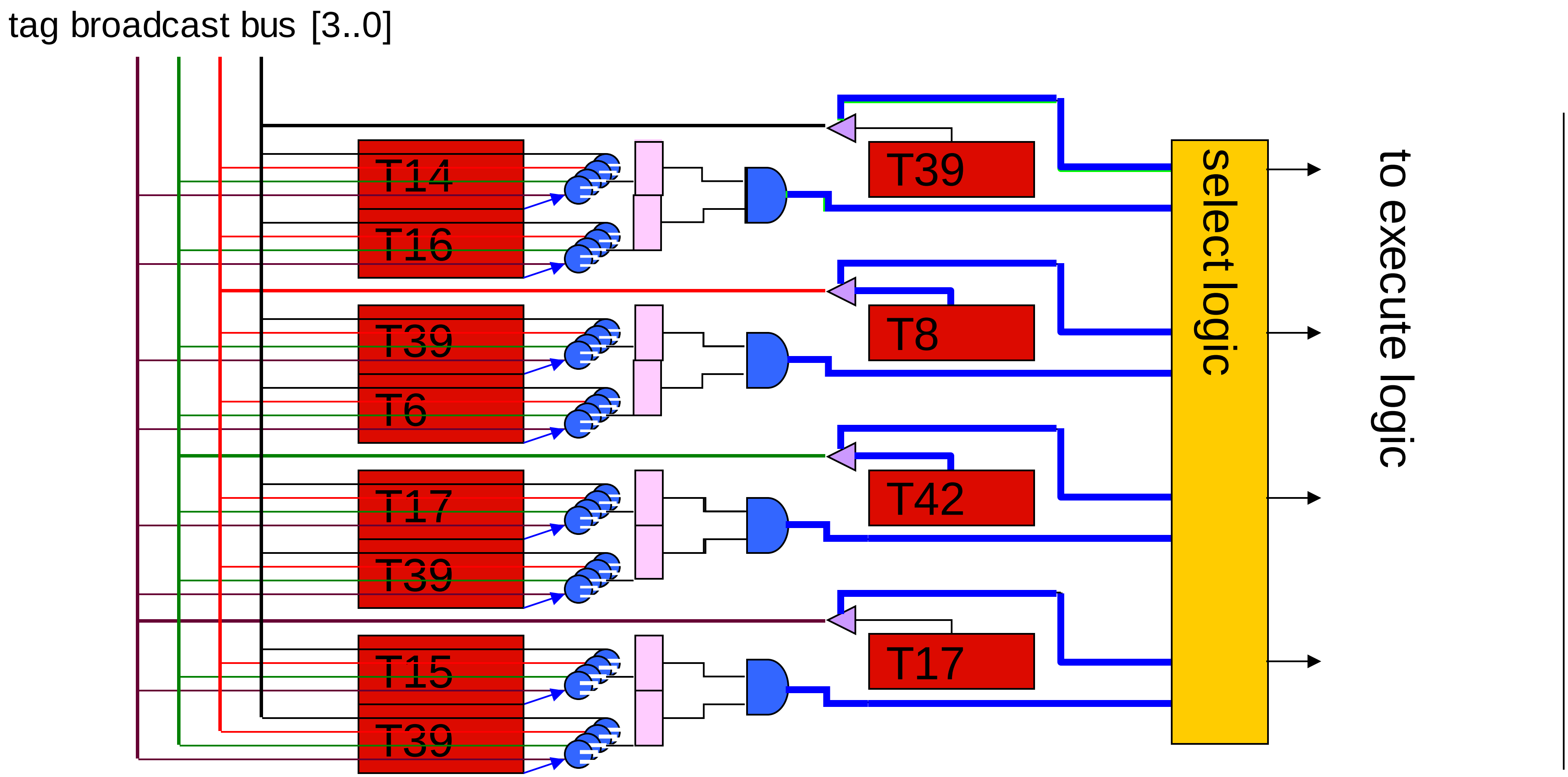
Scalar Scheduler (Issue Width = 1)

4-wide _issue mach.



From Prof. G. Loh's Slide

Superscalar Scheduler (Width = 4)



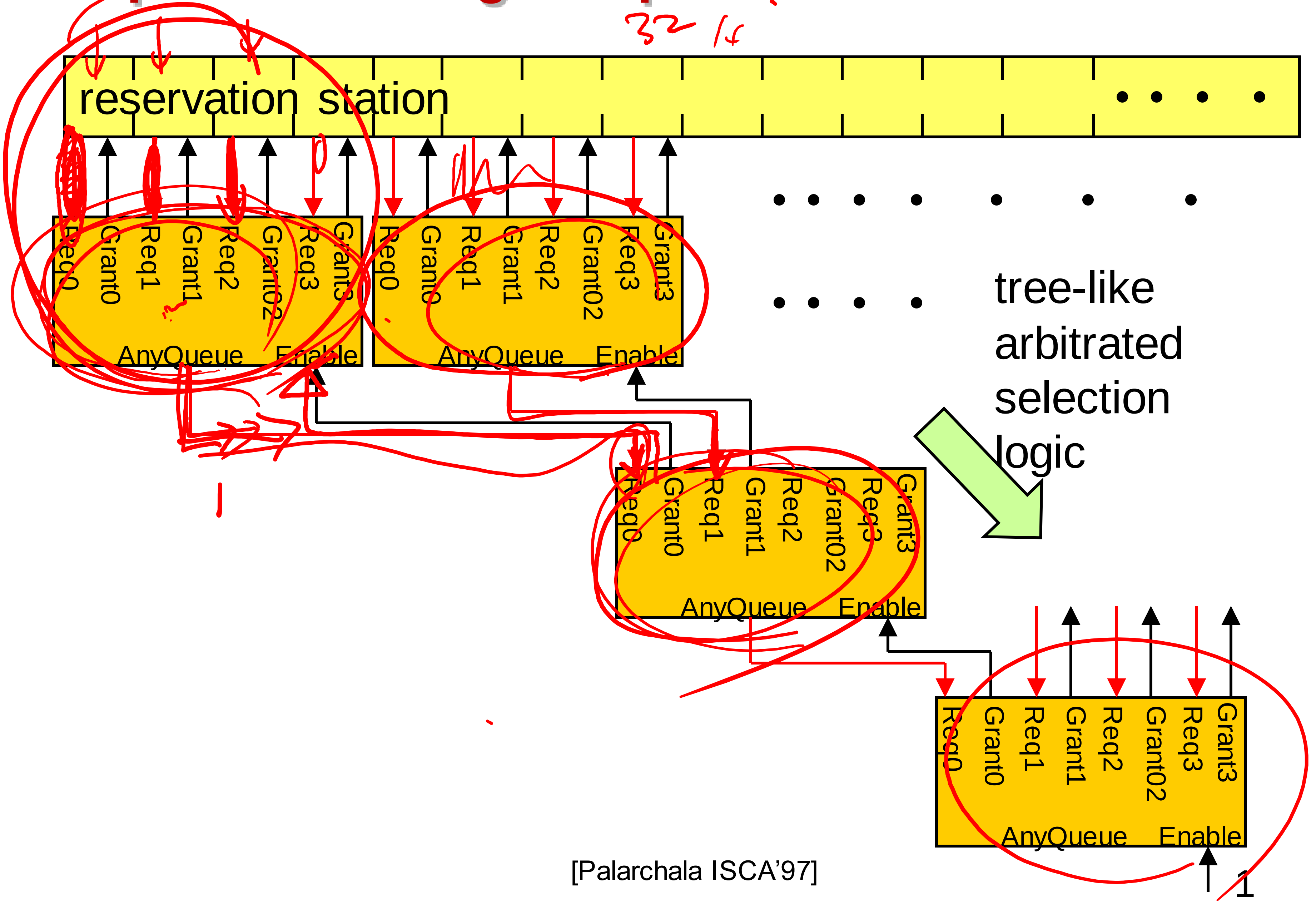
snapshot of RS (only 4 entries shown)

Adapted from Prof. G. Loh's Slide

Selection Logic

- select ready instructions to be issued
- goal
  - ✓ to reduce the height of DFG
- methods
  - ✓ location-based (e.g., leftmost ready first)
    - ○ allow simple, faster hardware
  - ✓ oldest ready first
    - ○ use location-based (in-order issue) with “compaction”
    - ○ slow and complex

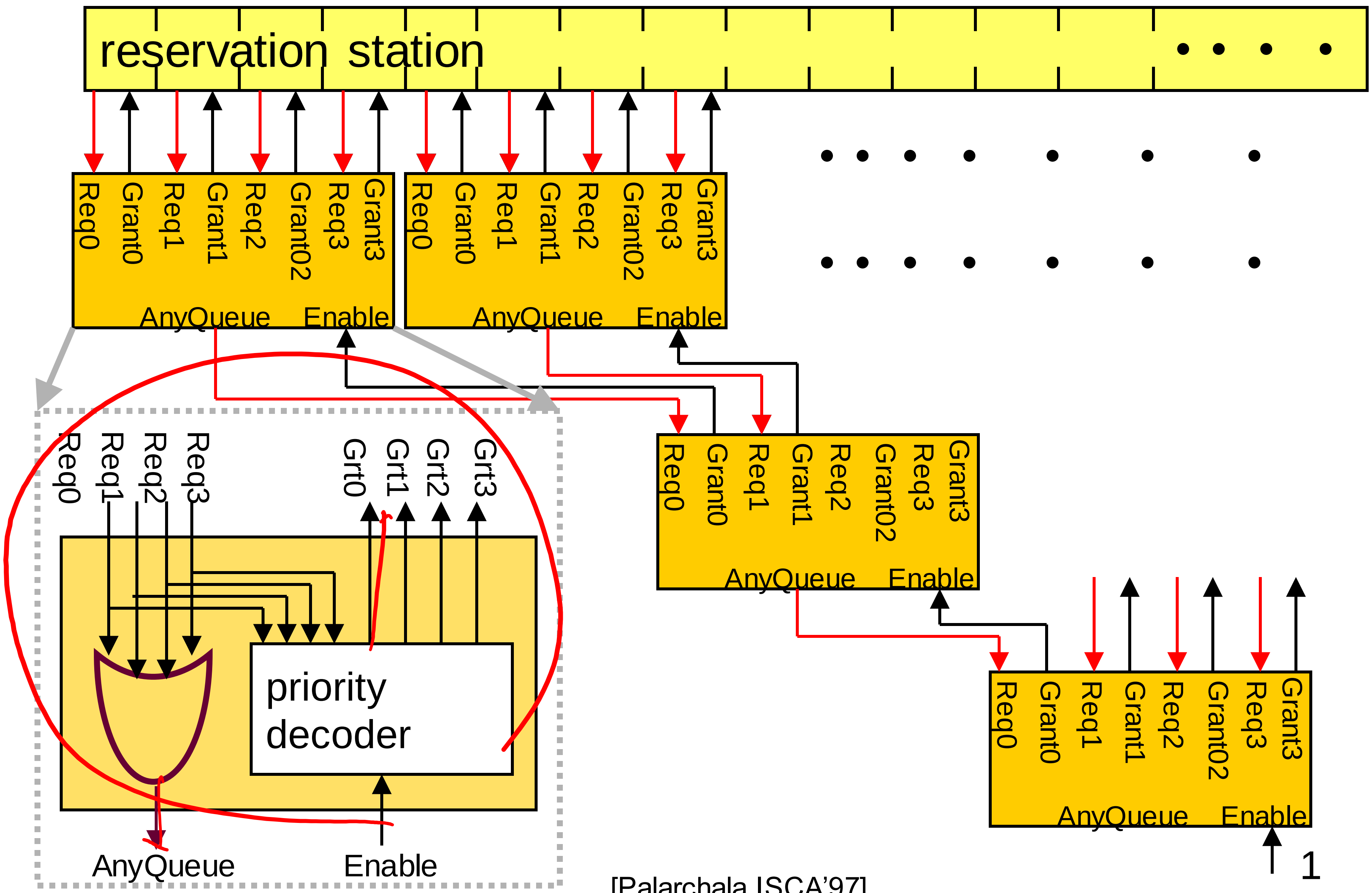
Simple Select Logic Implementation



tree-like arbitrated selection logic

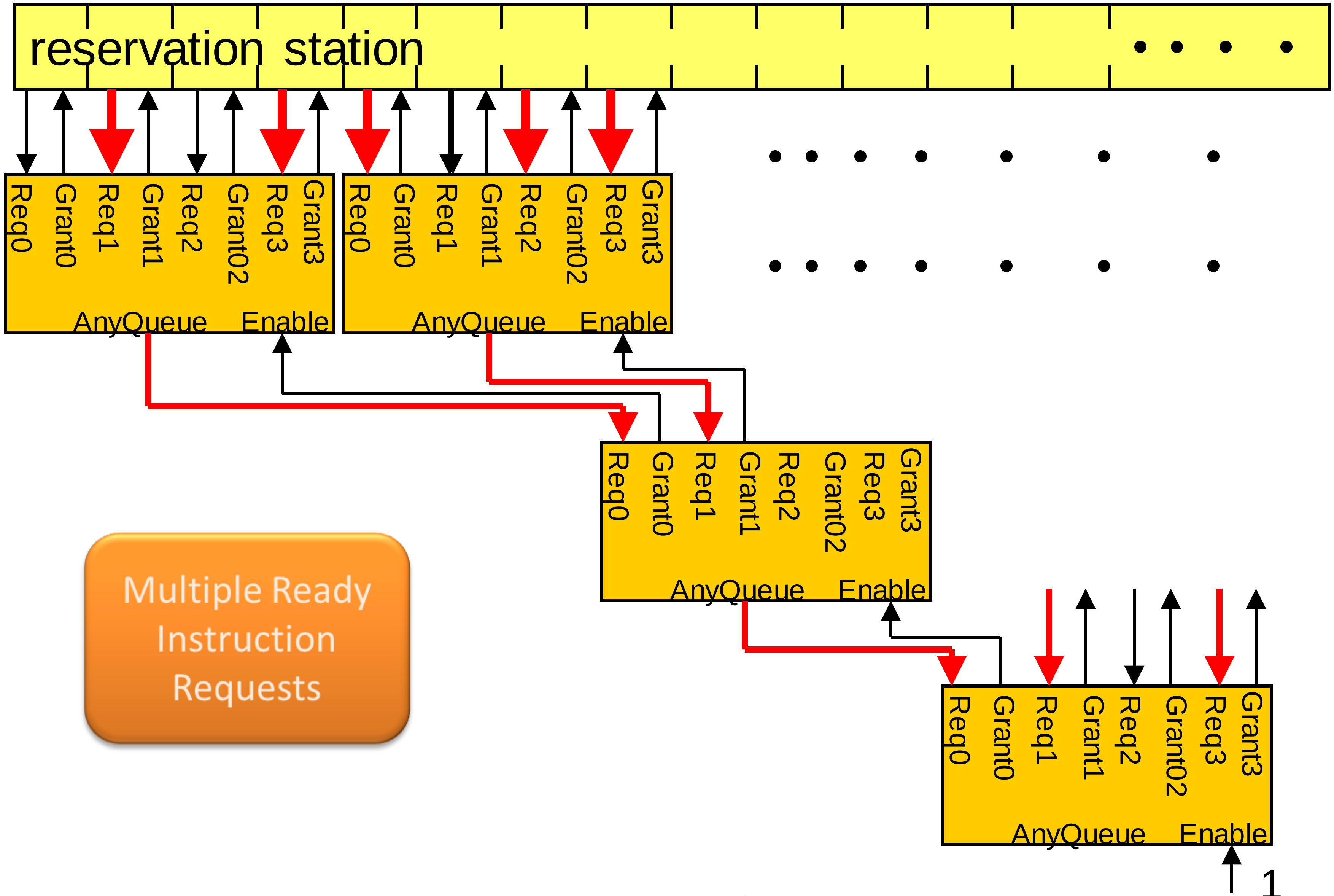
[Palarchala ISCA'97]

Simple Select Logic Implementation



[Palarchala ISCA'97]

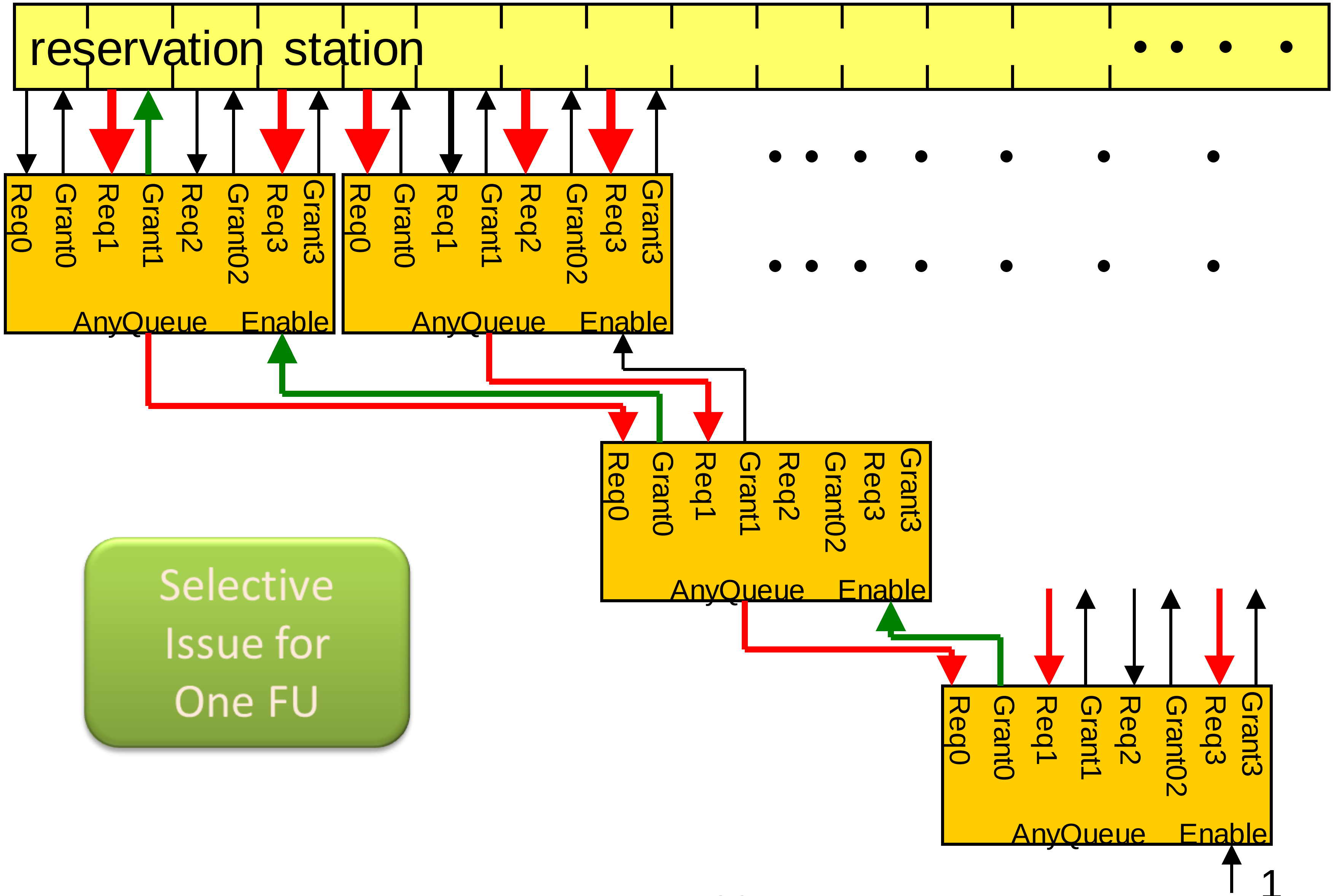
Simple Select Logic Implementation



Multiple Ready Instruction Requests

[Palarchala ISCA'97]

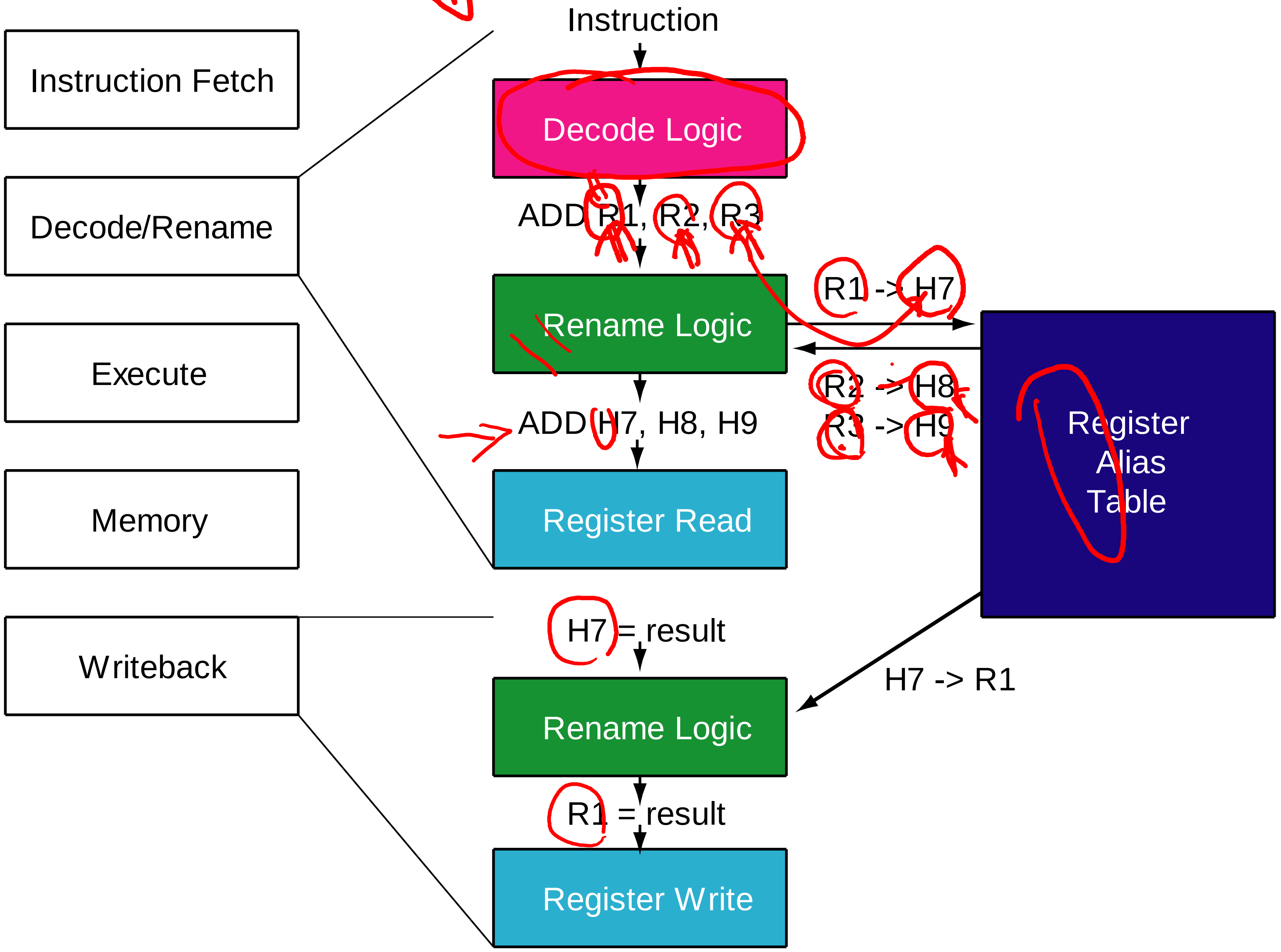
Simple Select Logic Implementation



Selective Issue for One FU

[Palarchala ISCA'97]

Register Renaming



Register Renaming

- ~~Alpha 21264+, MIPS R10K+~~, Pentium 4 use explicit register renaming
  - ✓ registers are not read until instruction dispatches (begins execution)
  - ✓ register renaming ensures no conflicts

PR13.

ADD R5 R1+[illegible] PR 37

```
DIV  R5, R4, R2
ADD  R7, R5, R1
SUB  R5, R3, R2
LW   R7, 1000(R5)
```

DIV PR37, PR45, PR2
ADD PR4, PR37, PR27
SUB PR42

RAT

R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	~~PR13~~ PR42
R6	PR20
R7	PR30

...

PR37

Register Renaming

- assume that PR37 is free and allocated to DIV’s destination register R5

```
DIV R5, R4, R2
ADD R7, R5, R1
SUB R5, R3, R2
LW  R7, 1000(R5)
```

```
DIV PR37, PR45, PR2
```

RAT

R1	PR23	
R2	PR2	
R3	PR17	
R4	PR45	
R5	~~PR13~~	PR37
R6	PR20	
R7	PR30	

…

Register Renaming

- assume that PR37 is free and allocated to DIV's destination register R5

```
DIV R5, R4, R2          DIV PR37, PR45, PR2
ADD R7, R5, R1
SUB R5, R3, R2
LW  R7, 1000(R5)
```

RAT

R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	PR37
R6	PR20
R7	PR30

...

Register Renaming

- assume that PR4 is free and allocated to ADD's destination register R7

```
DIV R5, R4, R2          DIV PR37, PR45, PR2
ADD R7, R5, R1          ADD PR4 , PR37, PR23
SUB R5, R3, R2
LW  R7, 1000(R5)
```

RAT

R1	PR23
R2	PR2
R3	PR17
R4	PR45
R5	PR37
R6	PR20
R7	~~PR30~~ PR4

...

Register Renaming

```
DIV R5, R4, R2
ADD R7, R5, R1
SUB R5, R3, R2
LW  R7, 1000(R5)
```

```
DIV PR37, PR45, PR2
ADD PR4 , PR37, PR23
SUB PR42, PR17, PR2
```

RAT

R1	PR23	
R2	PR2	
R3	PR17	
R4	PR45	
R5	~~PR37~~	PR42
R6	PR20	
R7	PR4	

…

Register Renaming

```
DIV R5, R4, R2
ADD R7, R5, R1
SUB R5, R3, R2
LW  R7, 1000(R5)
```

```
DIV PR37, PR45, PR2
ADD PR4 , PR37, PR23
SUB PR42, PR17, PR2
LW  PR19, 1000(PR42)
```

RAT

R1	PR23	
R2	PR2	
R3	PR17	
R4	PR45	
R5	PR42	
R6	PR20	
R7	~~PR4~~	PR19

...

Explicit Register Renaming

- make use of a physical register file that is larger than number of registers specified by ISA
- key insight: allocate a new physical destination register for every instruction that writes
  - ✓ removes all chance of WAR or WAW hazards
  - ✓ like Tomasulo, good for allowing full out-of-order completion
- mechanism? keep a translation table: RAT
  - ✓ ISA register $\Rightarrow$ physical register mapping
  - ✓ when register written, replace entry with new register from freelist
  - ✓ physical register becomes free when not used by any active instructions

c false dependency incurred by limited # of logical registers

Advantages of Explicit Renaming

- decouples renaming from scheduling:
  - ✓ pipeline can be exactly like "standard" DLX pipeline (perhaps with multiple operations issued per cycle)
  - ✓ or, pipeline could be Tomasulo-like or a scoreboard, etc.
  - ✓ standard forwarding or bypassing could be used
- allows data to be fetched from single register file
  - ✓ no need to bypass values from reorder buffer
  - ✓ this can be important for balancing pipeline
- many processors use a variant of this technique:
  - ✓ R10000, Alpha 21264, HP PA8000
- another way to get precise interrupt points:
  - ✓ all that needs to be "undone" for precise break point is to undo the table mappings
  - ✓ this provides an interesting mix between reorder buffer and future file
    - ○ results are written immediately back to register file
    - ○ registers names are "freed" in program order (by ROB)

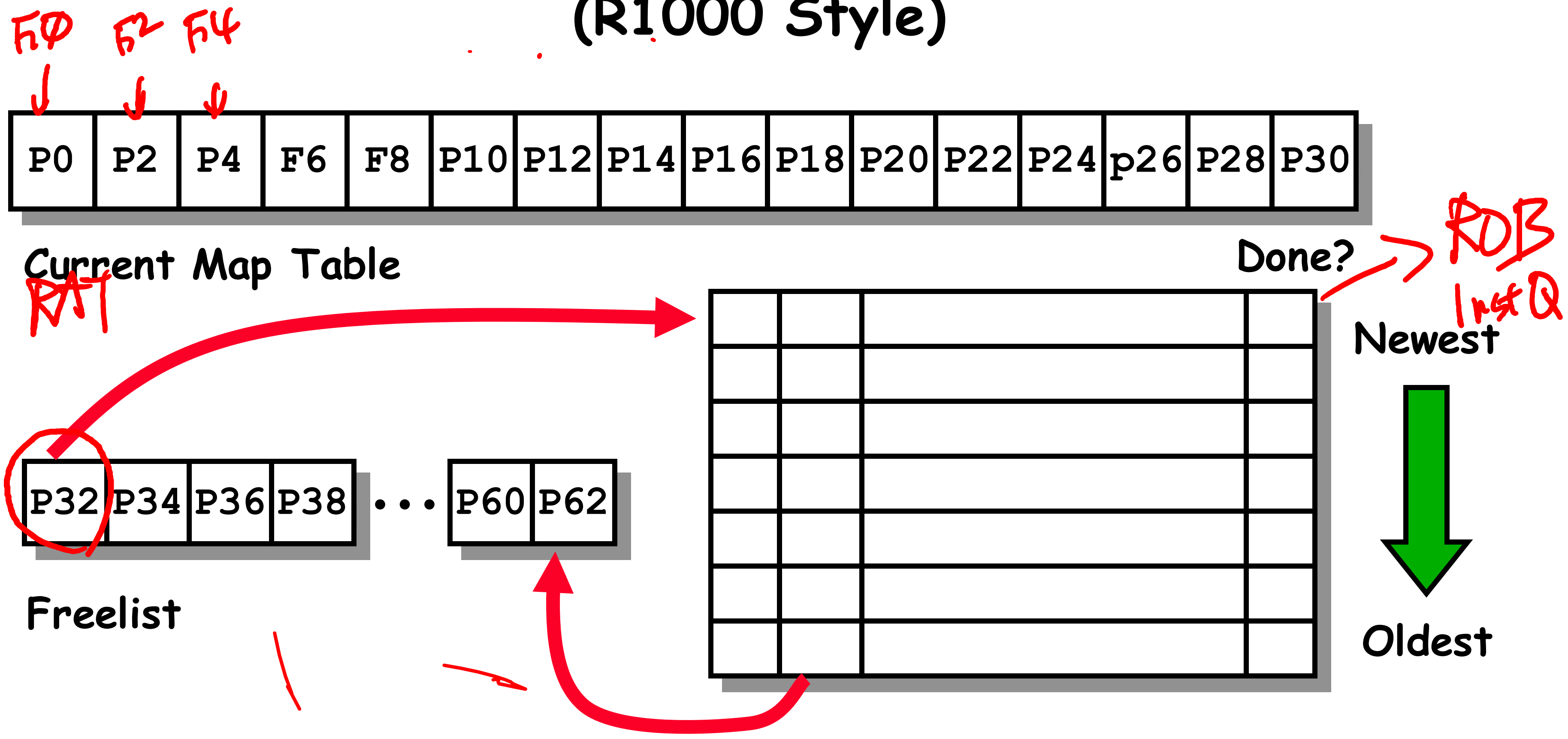
Explicit register renaming:
(R1000 Style)

```
        LD          F0    10    R2
        ADDD        F10   F4    F0
        DIVD        F2    F10   F6
        BNEZ        F2    Exit

        LD          F4    0     R3
        ADDD        F0    F4    F9
        SD          F4    0     R3
        …
Exit:
```

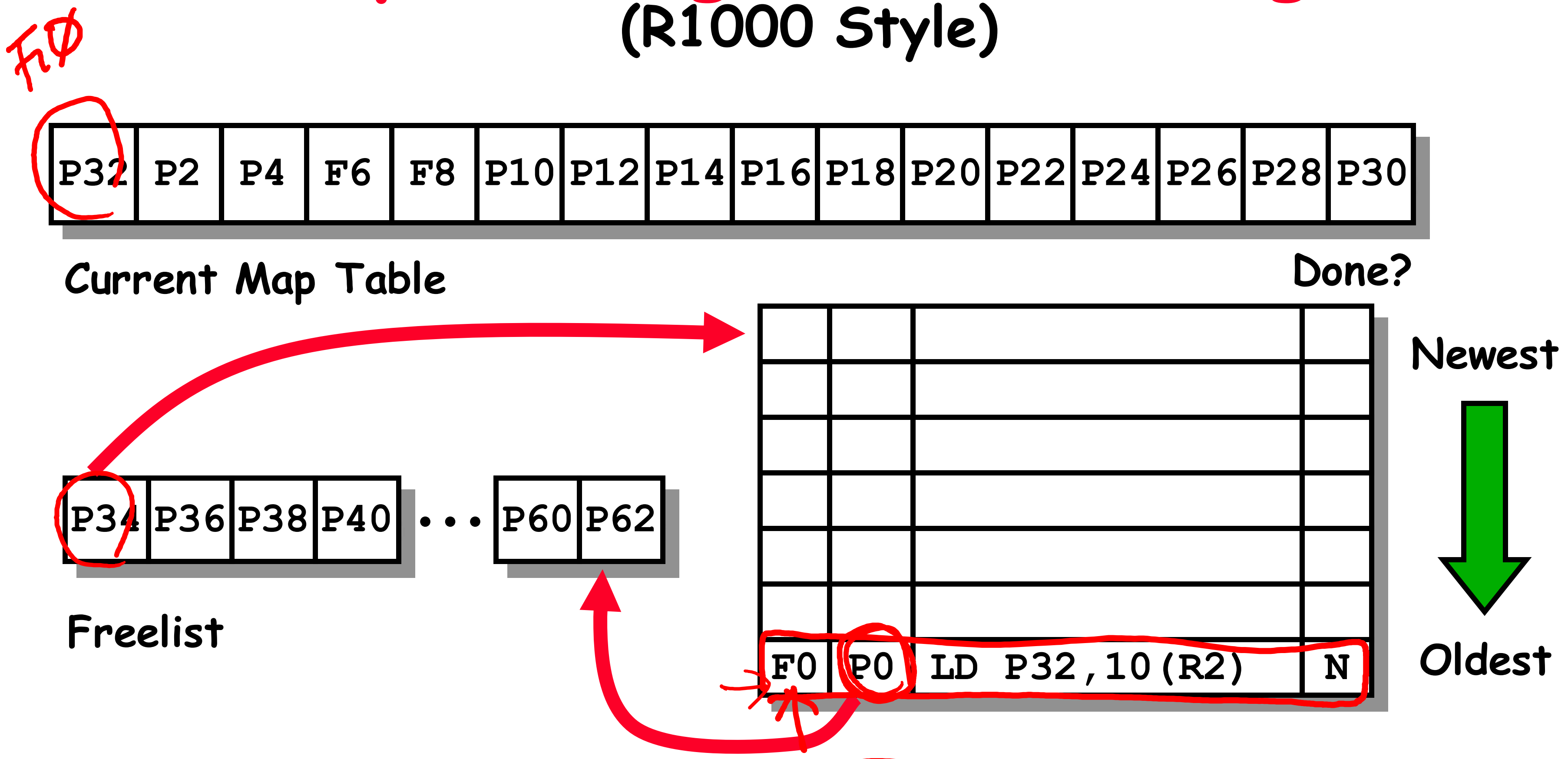
The same code example as Lecture 12

Explicit register renaming:
(R1000 Style)



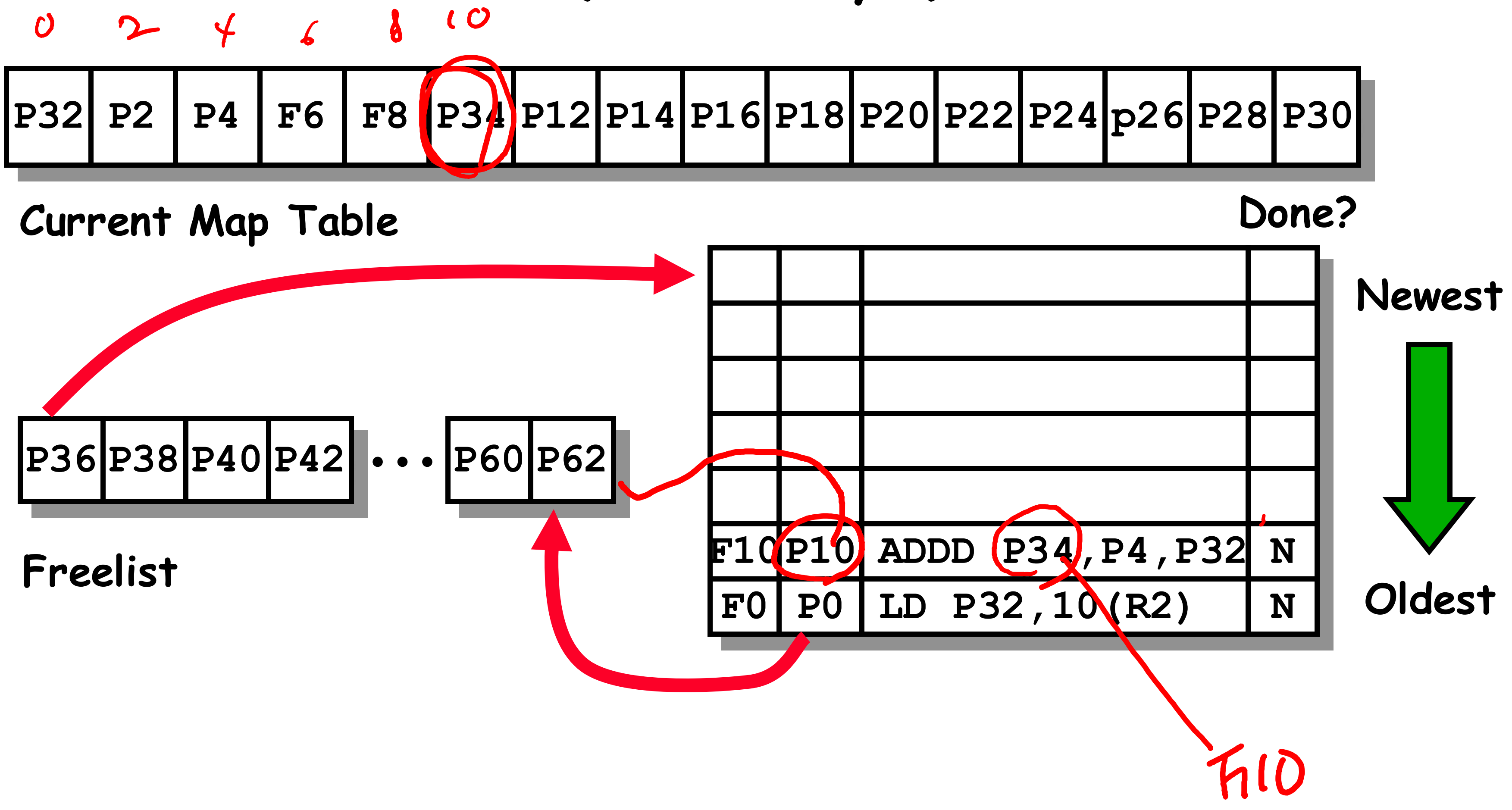
- Physical register file larger than ISA register file
- On issue, each instruction that modifies a register is allocated new physical register from freelist

Explicit register renaming:
(R1000 Style)

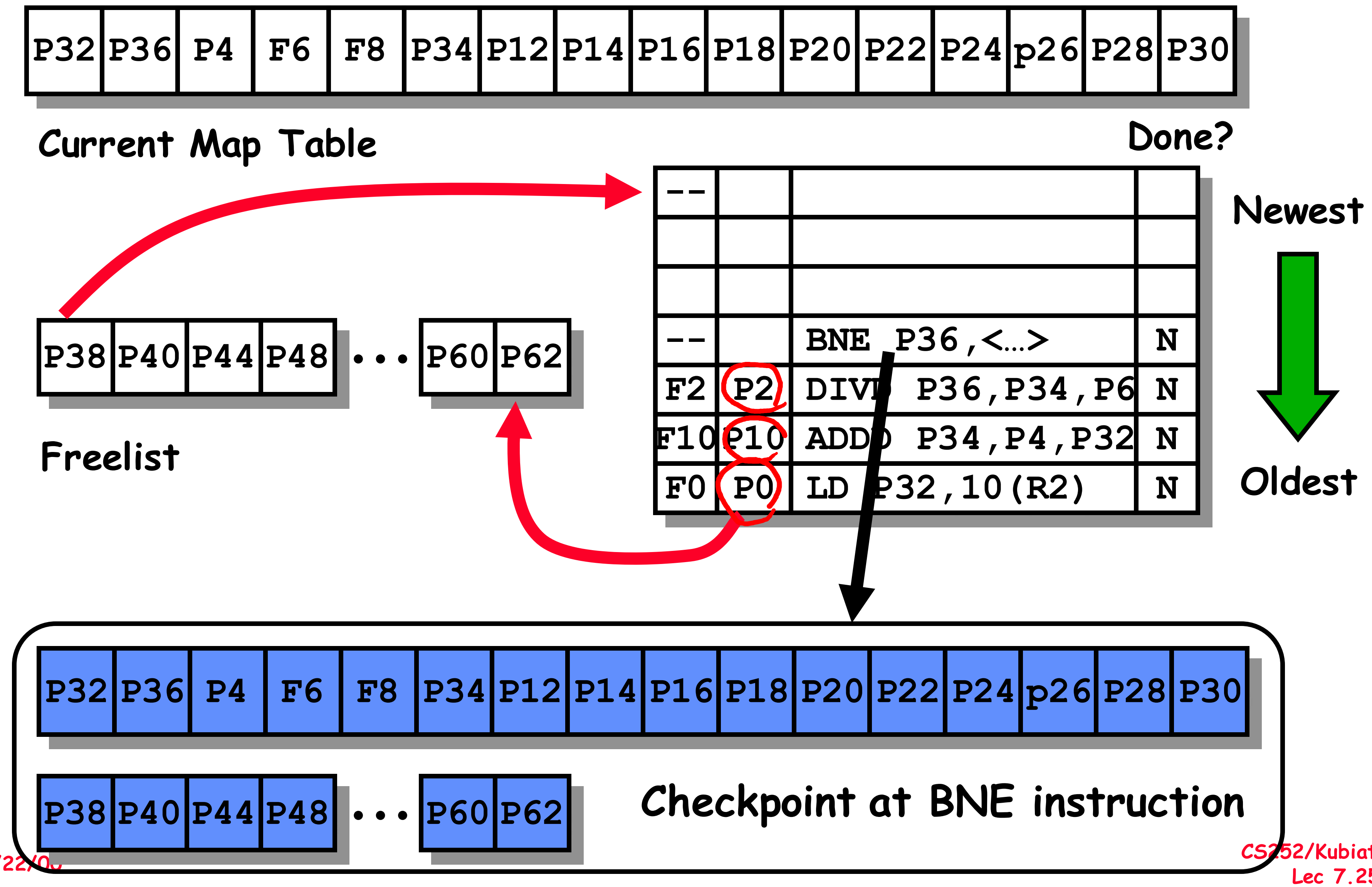


- Note that physical register P0 is "dead" (or not "live") past the point of this load.
  - When we go to commit the load, we free up

Explicit register renaming:
(R1000 Style)

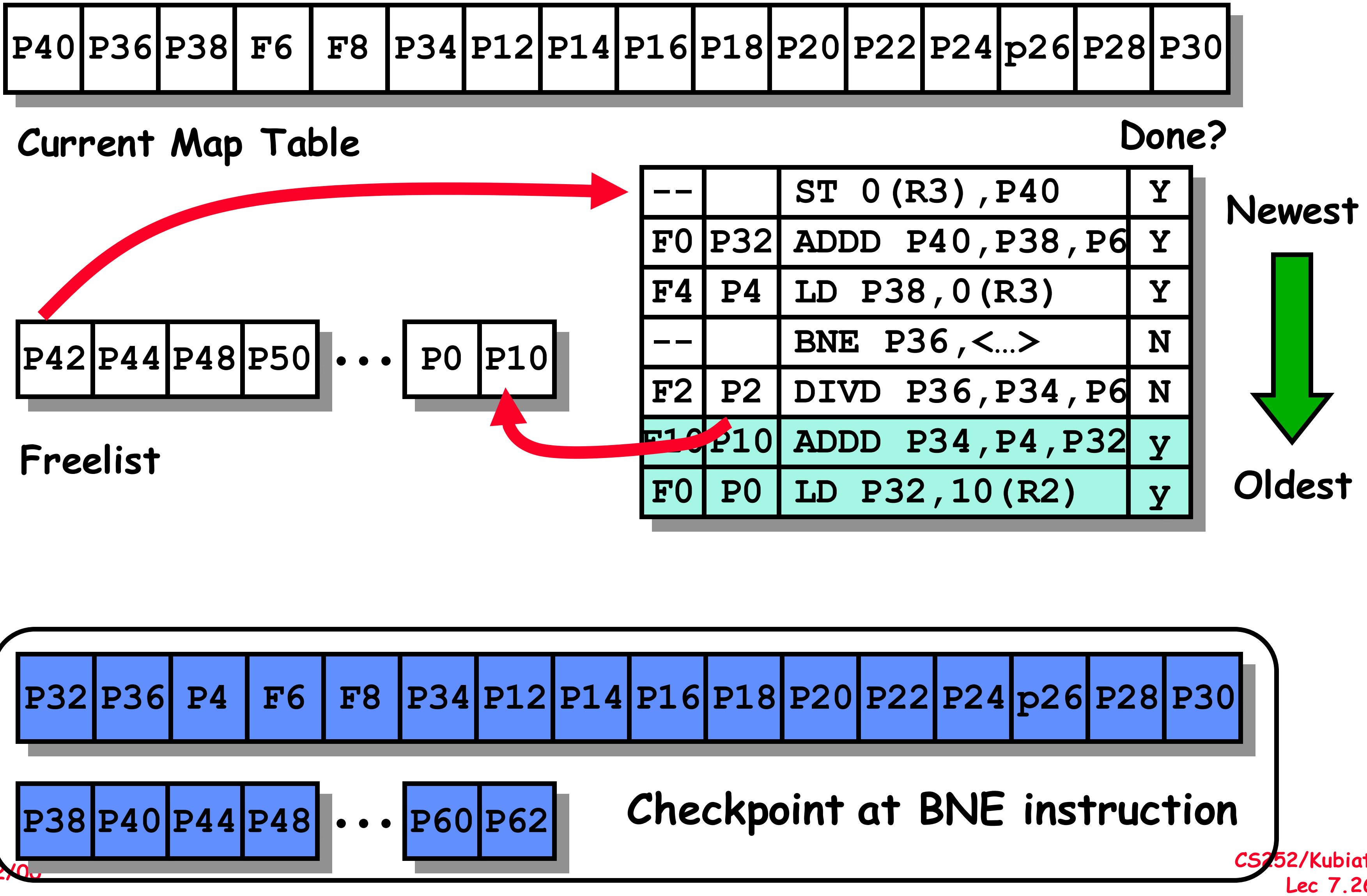


Explicit register renaming:
(R1000 Style)



P32 P36 P4 F6 F8 P34 P12 P14 P16 P18 P20 P22 P24 p26 P28 P30
Current Map Table
Done?
--
--
BNE P36,<…>
N
F2 P2 DIVD P36,P34,P6
N
F10 P10 ADDD P34,P4,P32
N
F0 P0 LD P32,10(R2)
N
Newest
Oldest
P38 P40 P44 P48 • • • P60 P62
Freelist
P32 P36 P4 F6 F8 P34 P12 P14 P16 P18 P20 P22 P24 p26 P28 P30
P38 P40 P44 P48 • • • P60 P62
Checkpoint at BNE instruction

Explicit register renaming:
(R1000 Style)



P40	P36	P38	F6	F8	P34	P12	P14	P16	P18	P20	P22	P24	p26	P28	P30

Current Map Table

			Done?
--		ST 0(R3),P40	Y
F0	P32	ADDD P40,P38,P6	Y
F4	P4	LD P38,0(R3)	Y
--		BNE P36,<…>	N
F2	P2	DIVD P36,P34,P6	N
F10	P10	ADDD P34,P4,P32	y
F0	P0	LD P32,10(R2)	y

P42	P44	P48	P50	• • •	P0	P10	

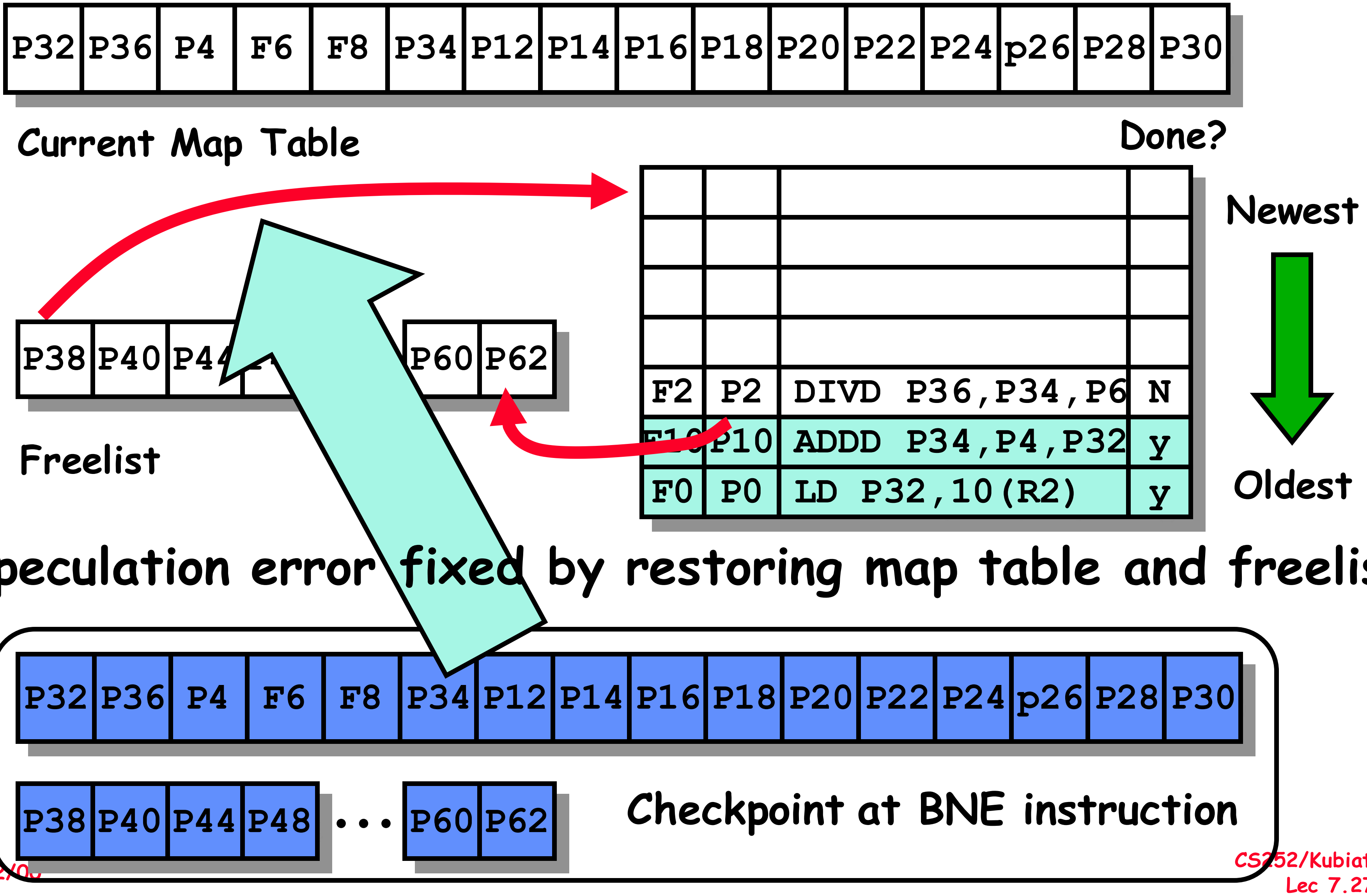
Freelist

P32	P36	P4	F6	F8	P34	P12	P14	P16	P18	P20	P22	P24	p26	P28	P30

P38	P40	P44	P48	• • •	P60	P62

Checkpoint at BNE instruction

Explicit register renaming:
(R1000 Style)


P32 P36 P4 F6 F8 P34 P12 P14 P16 P18 P20 P22 P24 p26 P28 P30
Current Map Table
Done?
Newest
Oldest
P38 P40 P44 P60 P62
Freelist
F2 P2 DIVD P36,P34,P6 N
F10 P10 ADDD P34,P4,P32 y
F0 P0 LD P32,10(R2) y
P32 P36 P4 F6 F8 P34 P12 P14 P16 P18 P20 P22 P24 p26 P28 P30
P38 P40 P44 P48 ... P60 P62
Checkpoint at BNE instruction

Speculation error fixed by restoring map table and freelist

Announcement

- next two lectures:
  - ✓ guest lecture – floating-point arithmetic (3/27)
  - ✓ tentatively no lecture (3/29)
- MP assignment
  - ✓ MP3 checkpoint 2 due on 3/18 5pm