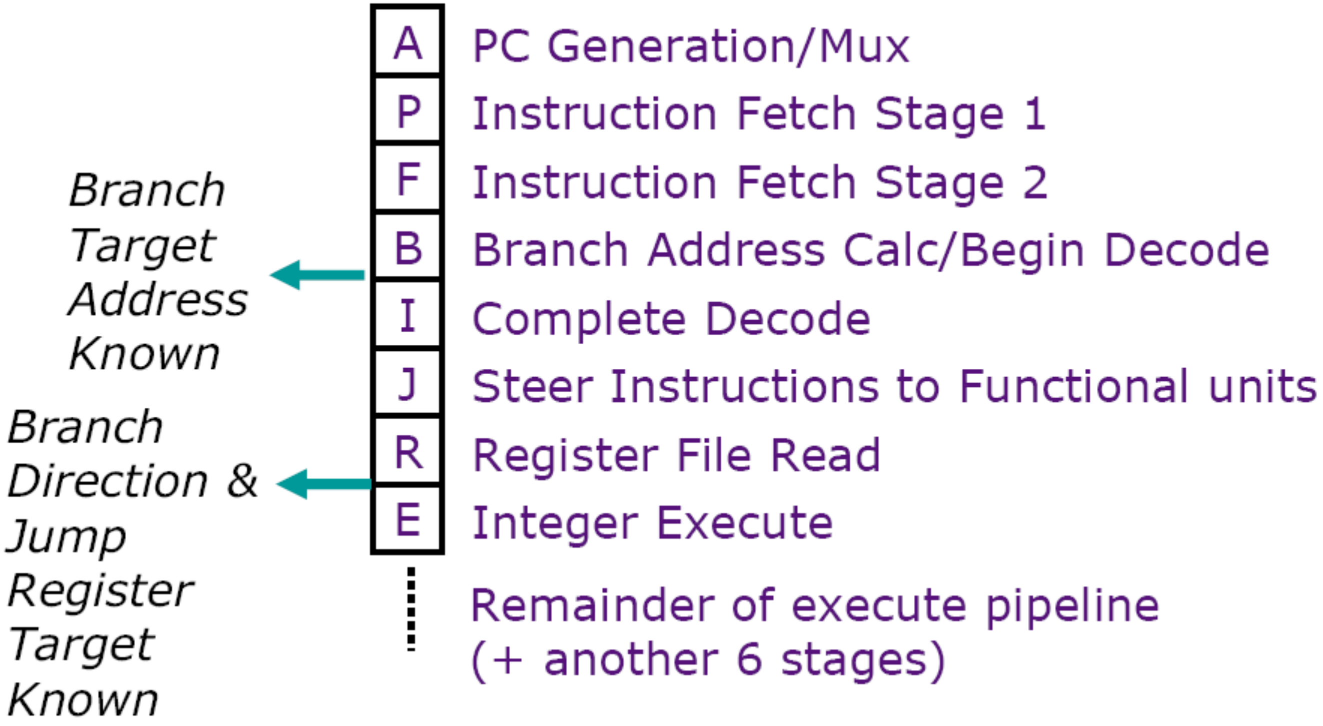
Selected Problems: Branch Prediction, OoO Processing

Suvinay Subramanian

(adapted from prior 6.823 offerings)

Problem 3.4: Branch Prediction

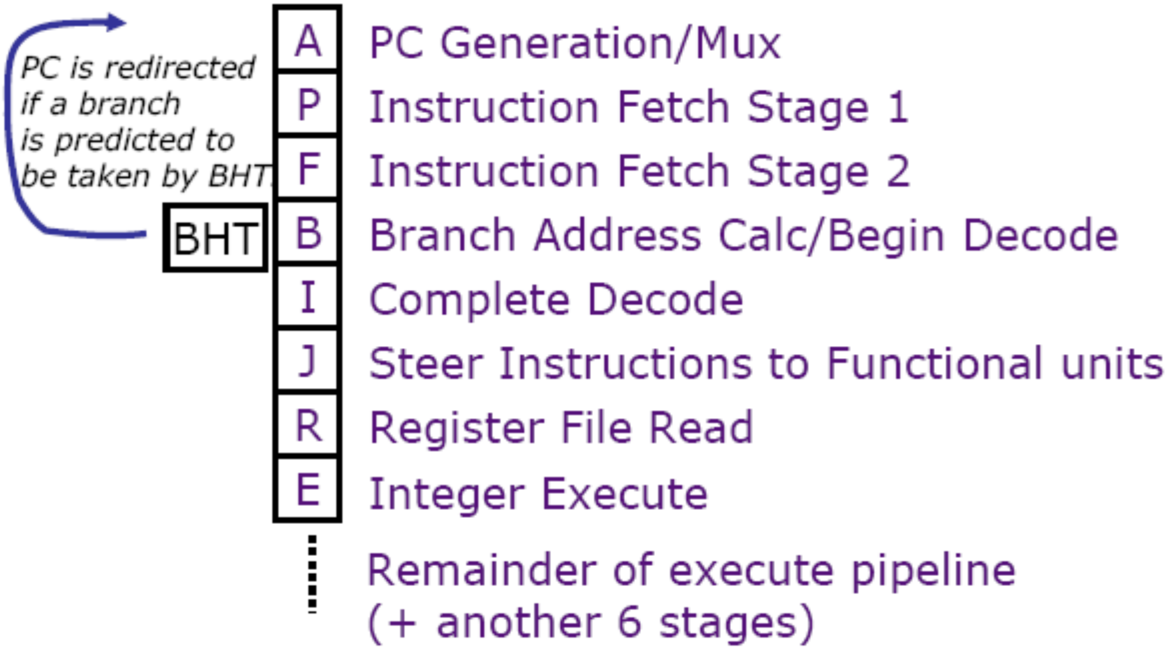
Branch Prediction



A PC Generation/Mux
P Instruction Fetch Stage 1
F Instruction Fetch Stage 2
B Branch Address Calc/Begin Decode
I Complete Decode
J Steer Instructions to Functional units
R Register File Read
E Integer Execute
Remainder of execute pipeline (+ another 6 stages)
Branch Target Address Known
Branch Direction & Jump Register Target Known

Instruction	Taken known? (At the end of)	Target known? (At the end of)
BEQZ/BNEZ	R	B
J	B (always taken)	B
JR	B (always taken)	R

Add Branch History Table (BHT)…



PC is redirected if a branch is predicted to be taken by BHT

BHT

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

J Steer Instructions to Functional units

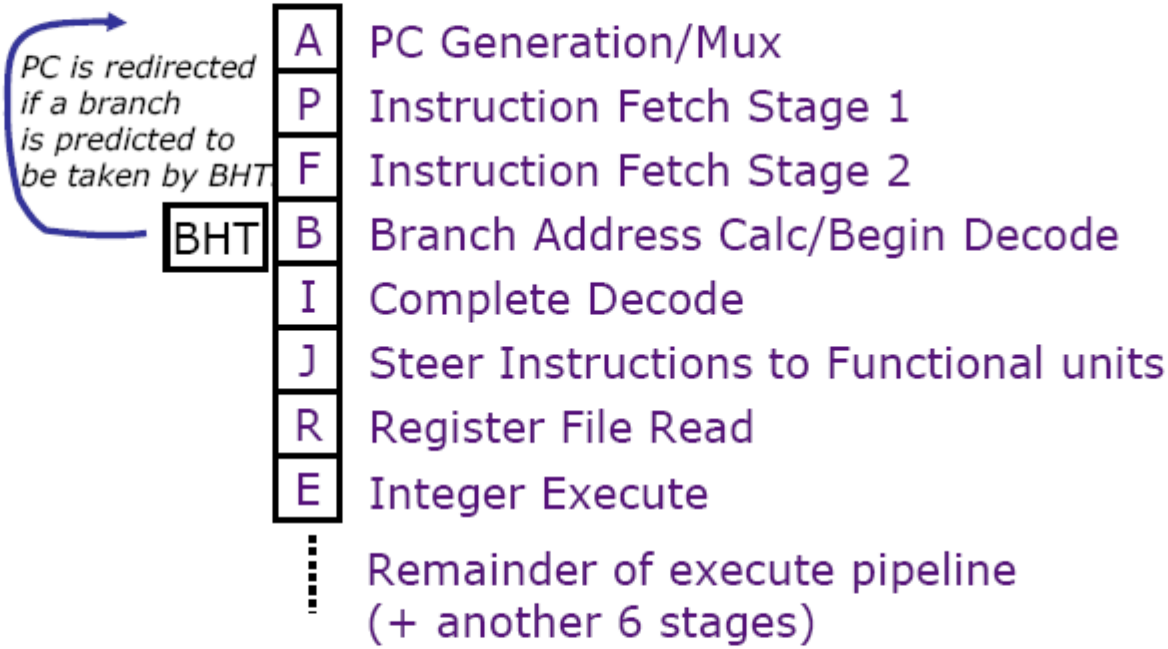
R Register File Read

E Integer Execute

Remainder of execute pipeline
(+ another 6 stages)

	Predicted Taken?	Actually Taken?	Pipeline bubbles
BEQZ/ BNEZ	Y	Y	
	Y	N	
	N	Y	
	N	N	
J	Always taken (No lookup)	Y	
JR	Always taken (No lookup)	Y	

Add Branch History Table (BHT)…


PC is redirected if a branch is predicted to be taken by BHT
BHT
A
P
F
B
I
J
R
E

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

J Steer Instructions to Functional units

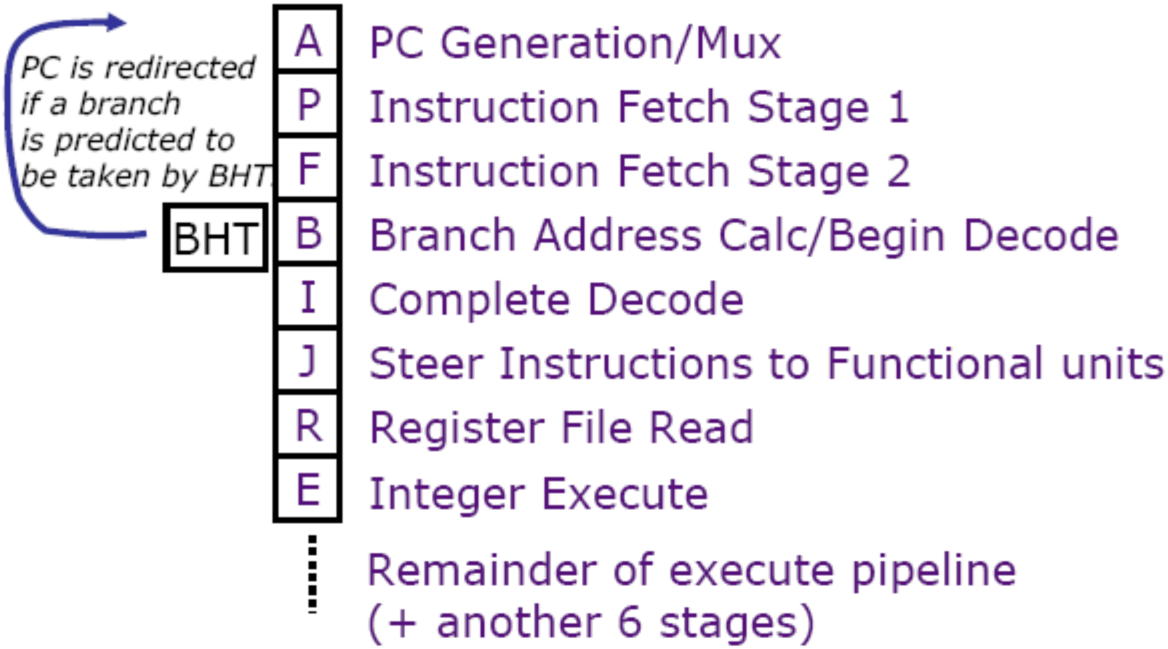
R Register File Read

E Integer Execute

Remainder of execute pipeline
(+ another 6 stages)

	Predicted Taken?	Actually Taken?	Pipeline bubbles
BEQZ/ BNEZ	Y	Y	3
	Y	N	
	N	Y	
	N	N	
J	Always taken (No lookup)	Y	
JR	Always taken (No lookup)	Y	

Add Branch History Table (BHT)…



PC is redirected if a branch is predicted to be taken by BHT
BHT
A
P
F
B
I
J
R
E

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

J Steer Instructions to Functional units

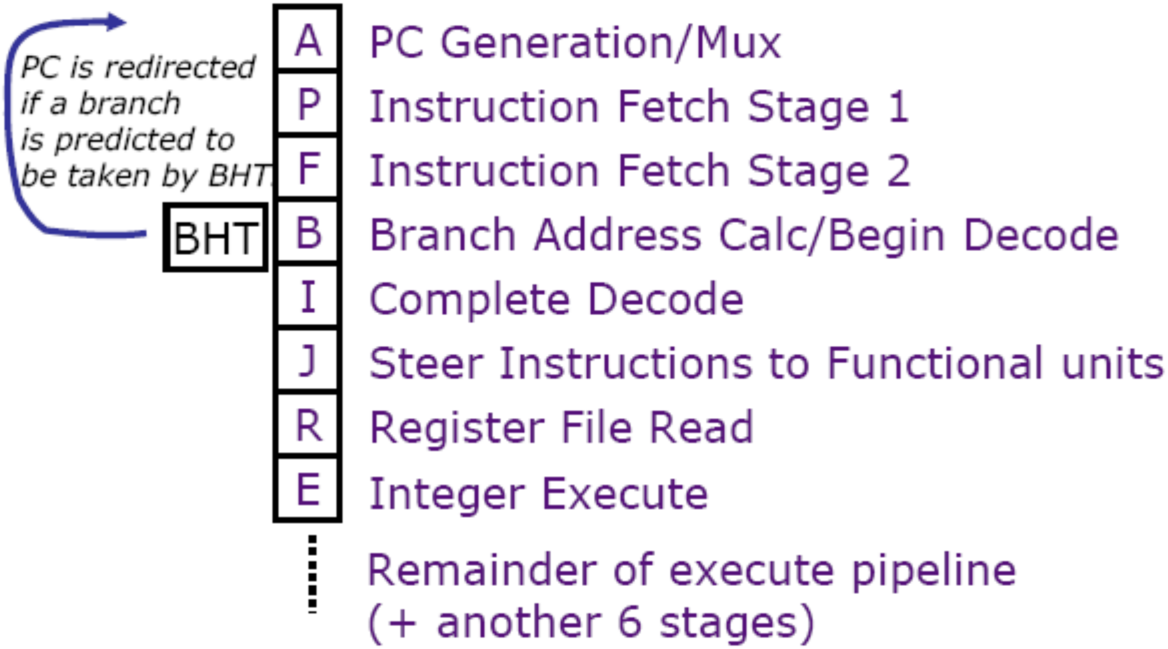
R Register File Read

E Integer Execute

Remainder of execute pipeline
(+ another 6 stages)

	Predicted Taken?	Actually Taken?	Pipeline bubbles
BEQZ/ BNEZ	Y	Y	3
	Y	N	6
	N	Y	
	N	N	
J	Always taken (No lookup)	Y	
JR	Always taken (No lookup)	Y	

Add Branch History Table (BHT)…



PC is redirected if a branch is predicted to be taken by BHT

BHT

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

J Steer Instructions to Functional units

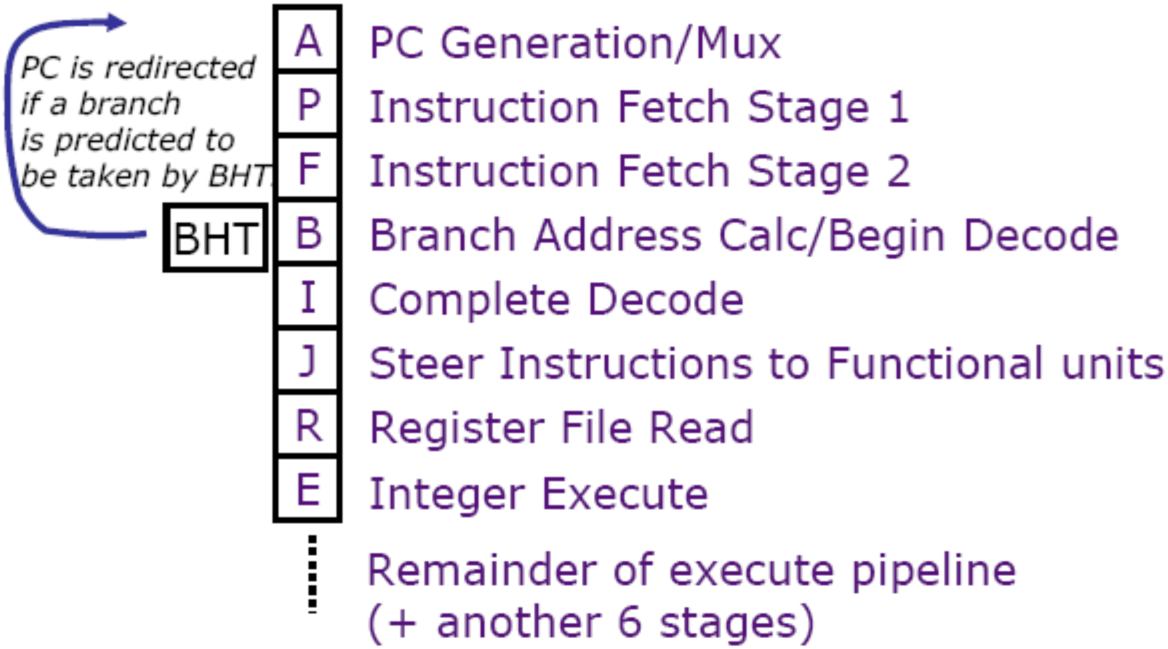
R Register File Read

E Integer Execute

Remainder of execute pipeline
(+ another 6 stages)

	Predicted Taken?	Actually Taken?	Pipeline bubbles
BEQZ/ BNEZ	Y	Y	3
	Y	N	6
	N	Y	6
	N	N	
J	Always taken (No lookup)	Y	
JR	Always taken (No lookup)	Y	

Add Branch History Table (BHT)…



PC is redirected if a branch is predicted to be taken by BHT

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

J Steer Instructions to Functional units

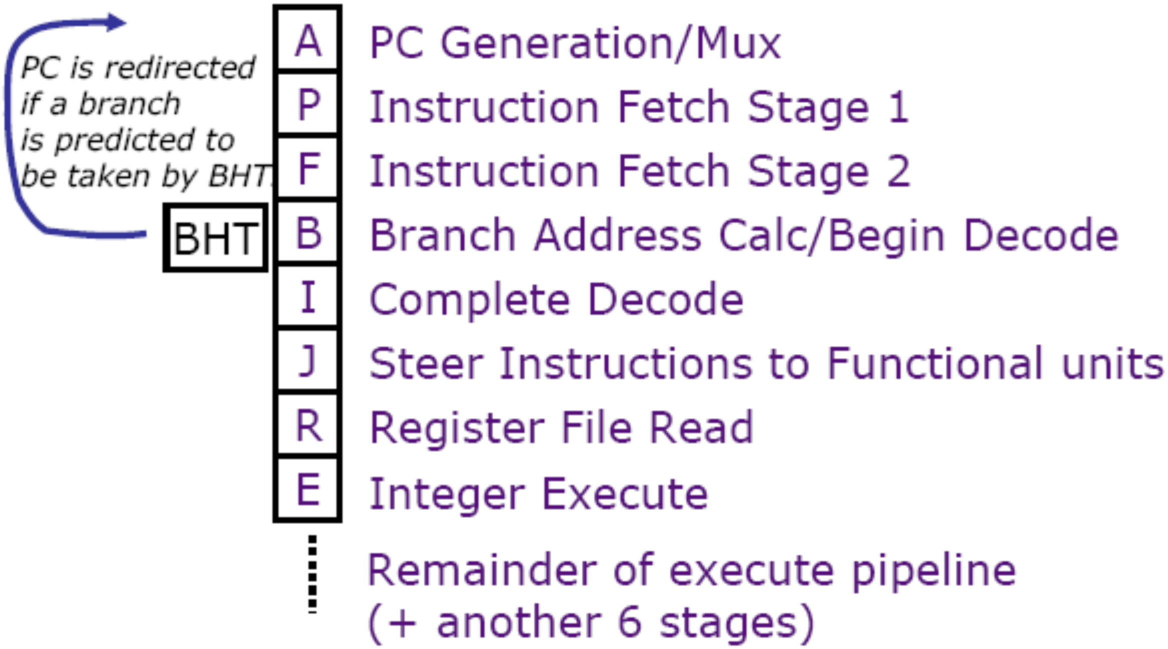
R Register File Read

E Integer Execute

Remainder of execute pipeline
(+ another 6 stages)

	Predicted Taken?	Actually Taken?	Pipeline bubbles
BEQZ/ BNEZ	Y	Y	3
	Y	N	6
	N	Y	6
	N	N	0
J	Always taken (No lookup)	Y	
JR	Always taken (No lookup)	Y	

Add Branch History Table (BHT)…



PC is redirected if a branch is predicted to be taken by BHT

BHT

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

J Steer Instructions to Functional units

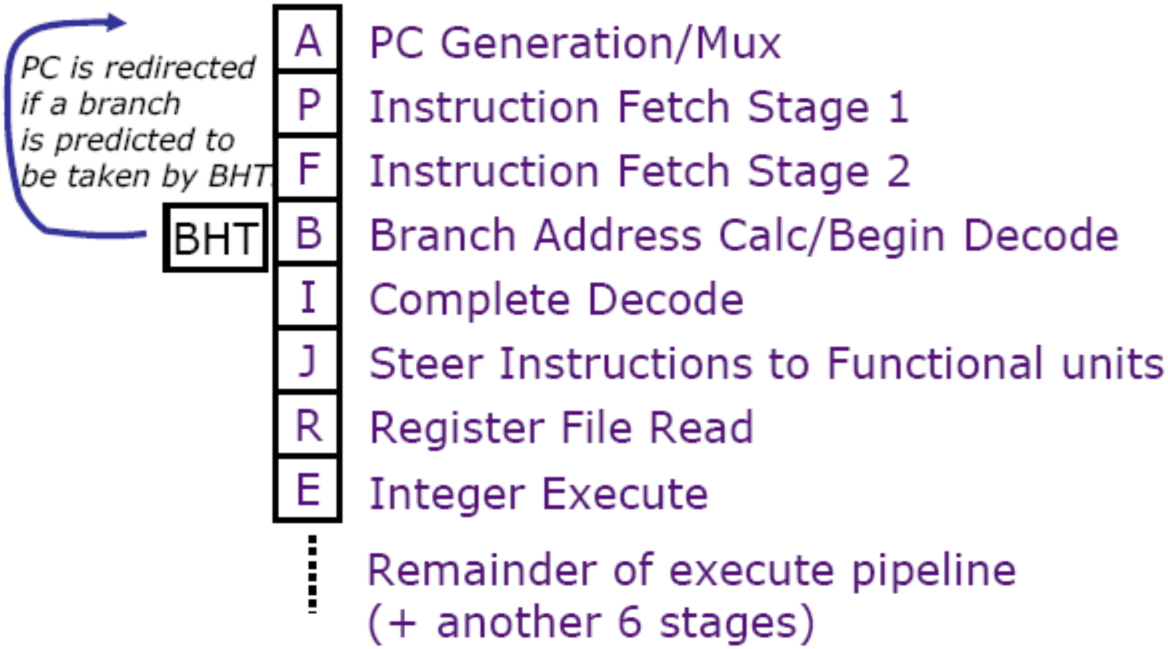
R Register File Read

E Integer Execute

Remainder of execute pipeline
(+ another 6 stages)

	Predicted Taken?	Actually Taken?	Pipeline bubbles
BEQZ/ BNEZ	Y	Y	3
	Y	N	6
	N	Y	6
	N	N	0
J	Always taken (No lookup)	Y	3
JR	Always taken (No lookup)	Y	

Add Branch History Table (BHT)…



PC is redirected if a branch is predicted to be taken by BHT

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

J Steer Instructions to Functional units

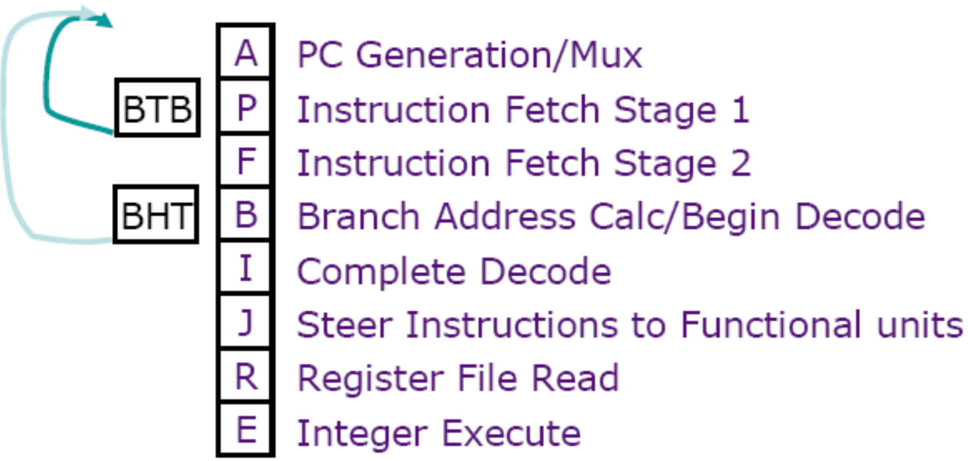
R Register File Read

E Integer Execute

Remainder of execute pipeline
(+ another 6 stages)

	Predicted Taken?	Actually Taken?	Pipeline bubbles
BEQZ/ BNEZ	Y	Y	3
	Y	N	6
	N	Y	6
	N	N	0
J	Always taken (No lookup)	Y	3
JR	Always taken (No lookup)	Y	6

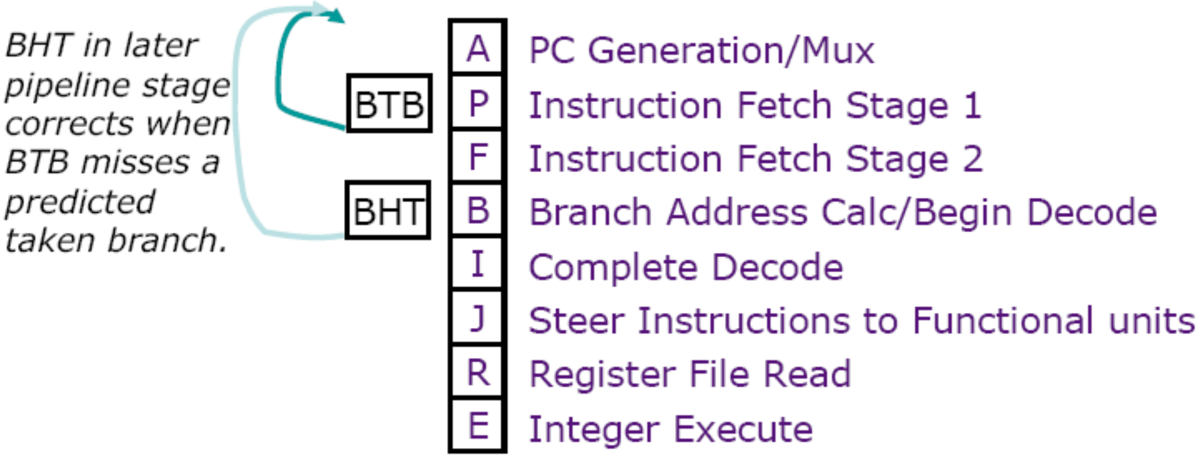
Add Branch Target Buffer (BTB)…


BTB
BHT
A
P
F
B
I
J
R
E
PC Generation/Mux
Instruction Fetch Stage 1
Instruction Fetch Stage 2
Branch Address Calc/Begin Decode
Complete Decode
Steer Instructions to Functional units
Register File Read
Integer Execute

BHT in later pipeline stage corrects when BTB misses a predicted taken branch.

	BTB Hit?	(BHT) Predicted Taken?	Actually Taken?	Pipeline bubbles
Conditional Branches	Y	Y	Y	
	Y	Y	N	
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	
	N	Y	N	
	N	N	Y	
	N	N	N	

Add Branch Target Buffer (BTB)…



BHT in later pipeline stage corrects when BTB misses a predicted taken branch.

BTB

BHT

A PC Generation/Mux

P Instruction Fetch Stage 1

F Instruction Fetch Stage 2

B Branch Address Calc/Begin Decode

I Complete Decode

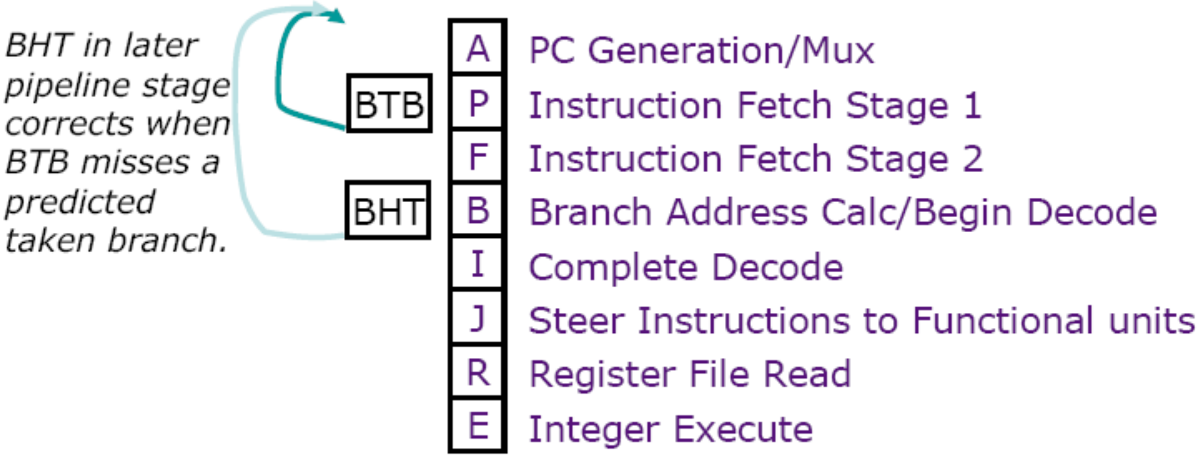
J Steer Instructions to Functional units

R Register File Read

E Integer Execute

	BTB Hit?	**(BHT) Predicted Taken?**	**Actually Taken?**	**Pipeline bubbles**
Conditional Branches	Y	Y	Y	1
	Y	Y	N	
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	
	N	Y	N	
	N	N	Y	
	N	N	N	

Add Branch Target Buffer (BTB)…

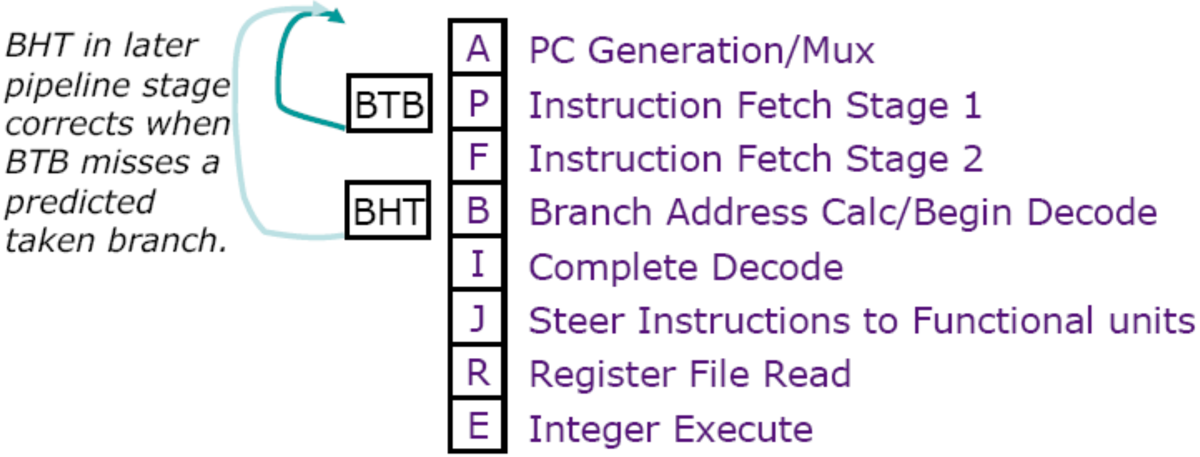


BHT in later pipeline stage corrects when BTB misses a predicted taken branch.

A	PC Generation/Mux
P	Instruction Fetch Stage 1
F	Instruction Fetch Stage 2
B	Branch Address Calc/Begin Decode
I	Complete Decode
J	Steer Instructions to Functional units
R	Register File Read
E	Integer Execute

	BTB Hit?	**(BHT) Predicted Taken?**	**Actually Taken?**	**Pipeline bubbles**
Conditional Branches	Y	Y	Y	1
	Y	Y	N	6
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	
	N	Y	N	
	N	N	Y	
	N	N	N	

Add Branch Target Buffer (BTB)…

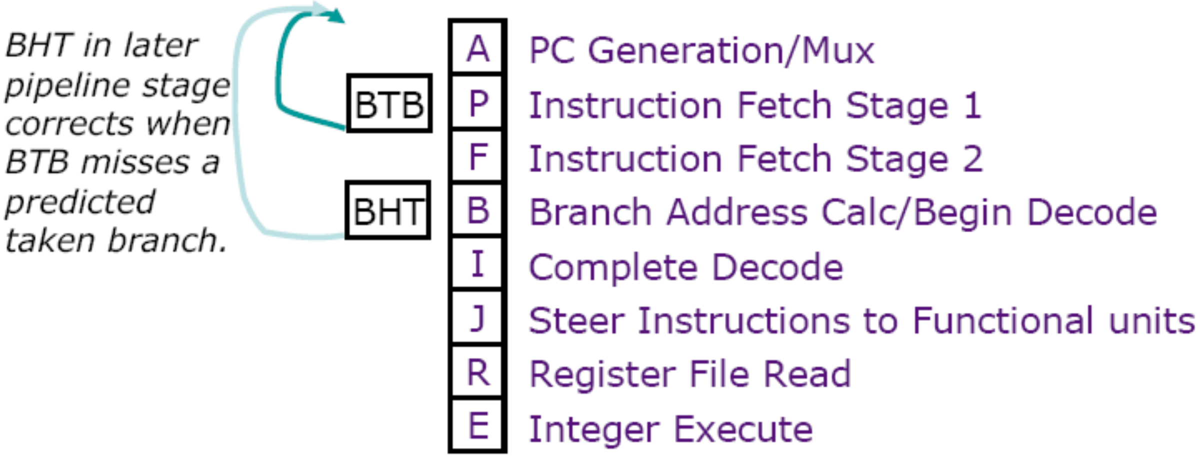


BHT in later pipeline stage corrects when BTB misses a predicted taken branch.

A PC Generation/Mux
P Instruction Fetch Stage 1
F Instruction Fetch Stage 2
B Branch Address Calc/Begin Decode
I Complete Decode
J Steer Instructions to Functional units
R Register File Read
E Integer Execute

	BTB Hit?	(BHT) Predicted Taken?	Actually Taken?	Pipeline bubbles
Conditional Branches	Y	Y	Y	1
	Y	Y	N	6
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	3
	N	Y	N	
	N	N	Y	
	N	N	N	

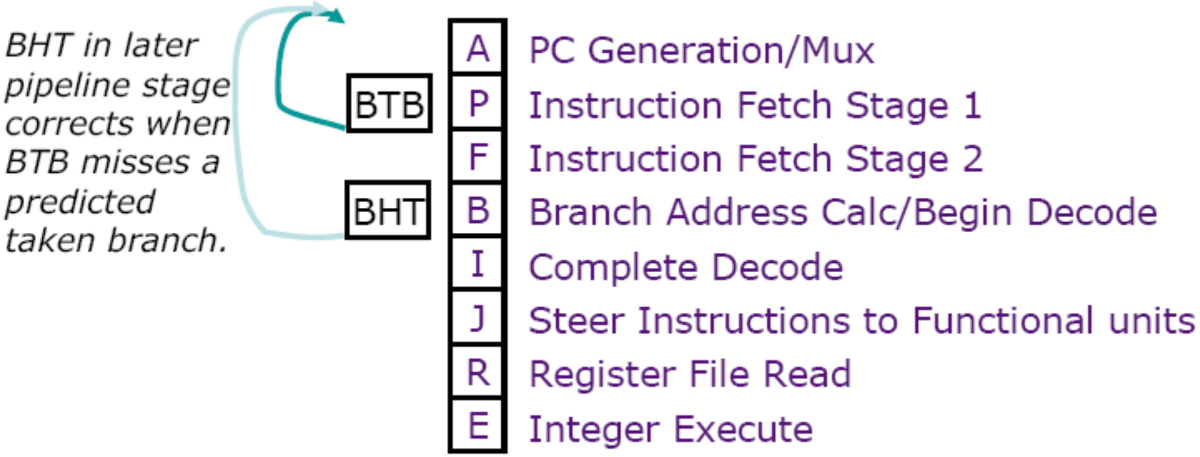
Add Branch Target Buffer (BTB)…


BHT in later pipeline stage corrects when BTB misses a predicted taken branch.
BTB
BHT
A
P
F
B
I
J
R
E

A PC Generation/Mux
P Instruction Fetch Stage 1
F Instruction Fetch Stage 2
B Branch Address Calc/Begin Decode
I Complete Decode
J Steer Instructions to Functional units
R Register File Read
E Integer Execute

	BTB Hit?	(BHT) Predicted Taken?	Actually Taken?	Pipeline bubbles
Conditional Branches	Y	Y	Y	1
	Y	Y	N	6
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	3
	N	Y	N	6
	N	N	Y	
	N	N	N	

Add Branch Target Buffer (BTB)...



BHT in later pipeline stage corrects when BTB misses a predicted taken branch.

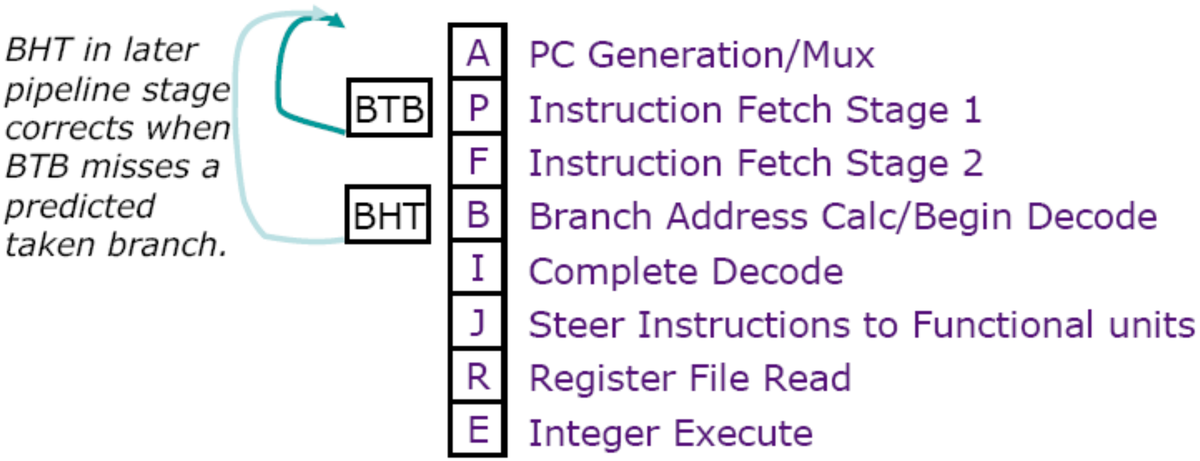
BTB

BHT

- A PC Generation/Mux
- P Instruction Fetch Stage 1
- F Instruction Fetch Stage 2
- B Branch Address Calc/Begin Decode
- I Complete Decode
- J Steer Instructions to Functional units
- R Register File Read
- E Integer Execute

	BTB Hit?	(BHT) Predicted Taken?	Actually Taken?	Pipeline bubbles
Conditional Branches	Y	Y	Y	1
	Y	Y	N	6
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	3
	N	Y	N	6
	N	N	Y	6
	N	N	N	

Add Branch Target Buffer (BTB)…

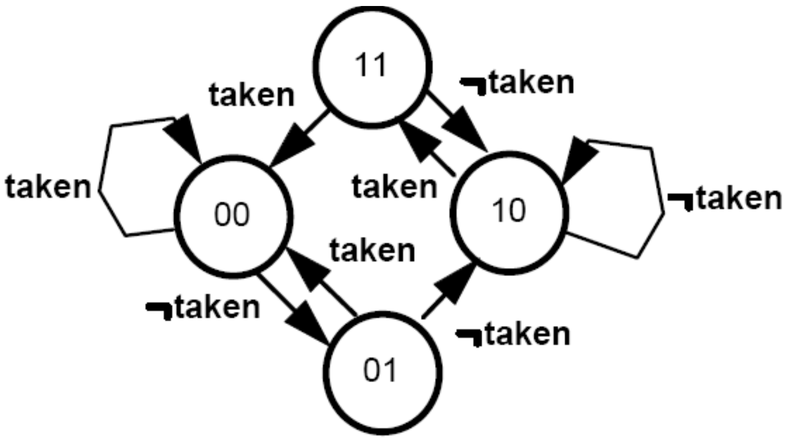


BHT in later pipeline stage corrects when BTB misses a predicted taken branch.

A	PC Generation/Mux
P	Instruction Fetch Stage 1
F	Instruction Fetch Stage 2
B	Branch Address Calc/Begin Decode
I	Complete Decode
J	Steer Instructions to Functional units
R	Register File Read
E	Integer Execute

	BTB Hit?	(BHT) Predicted Taken?	Actually Taken?	Pipeline bubbles
Conditional Branches	Y	Y	Y	1
	Y	Y	N	6
	Y	N	Y	Cannot occur
	Y	N	N	Cannot occur
	N	Y	Y	3
	N	Y	N	6
	N	N	Y	6
	N	N	N	0

```
ADDRESS                INSTRUCTION
 0x1000          BR1:  BEQZ R5, NEXT      ; always taken
 0x1004                ADDI R4, R4, #4
 0x1008                MULT R3, R5, R3
 0x100C                ST   R3, 0(R4)
 0x1010                SUBI R5, R5, #1
 0x1014         NEXT:  ADDI R1, R1, #1
 0x1018                SLTI R2, R1, 100   ; repeat 100 times
 0x101C          BR2:  BNEZ R2, BR1
 0x1020                NOP
 0x1024                NOP
 0x1028                NOP
```



1X : Guess Not Taken

0X : Guess Taken

Initial Snapshot

(Valid) V	Entry PC	Predicted Target PC
1	0x101C	0x1000

BTB

	. . .	
BR1	1	1
	. . .	
BR2	0	0
	. . .	

BHT

```
ADDRESS              INSTRUCTION
 0x1000       BR1:   BEQZ R5, NEXT       ; always taken
 0x1004              ADDI R4, R4, #4
 0x1008              MULT R3, R5, R3
 0x100C              ST   R3, 0(R4)
 0x1010              SUBI R5, R5, #1
 0x1014      NEXT:   ADDI R1, R1, #1
 0x1018              SLTI R2, R1, 100    ; repeat 100 times
 0x101C       BR2:   BNEZ R2, BR1
 0x1020              NOP
 0x1024              NOP
 0x1028              NOP
```

BHT

	. . .	
BR1	1	1
	. . .	
BR2	0	0
	. . .	

BTB

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1																							
0x1018	SLTI R2, R1, 100																							
0x101C	BNEZ R2, LOOP																							
0x1000	BEQZ R5, NEXT																							
0x1014	ADDI R1, R1, #1																							

```
ADDRESS                   INSTRUCTION
 0x1000         BR1:   BEQZ R5, NEXT       ; always taken
 0x1004                ADDI R4, R4, #4
 0x1008                MULT R3, R5, R3
 0x100C                ST   R3, 0(R4)
 0x1010                SUBI R5, R5, #1
 0x1014        NEXT:   ADDI R1, R1, #1
 0x1018                SLTI R2, R1, 100    ; repeat 100 times
 0x101C         BR2:   BNEZ R2, BR1
 0x1020                NOP
 0x1024                NOP
 0x1028                NOP
```

	...	
BR1	0	0
	...	
BR2	0	0
	...	

BHT

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000
1	0x1000	0x1014

BTB

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1																							
0x1018	SLTI R2, R1, 100																							
0x101C	BNEZ R2, LOOP																							
0x1000	BEQZ R5, NEXT																							
0x1014	ADDI R1, R1, #1																							

```
ADDRESS                 INSTRUCTION
 0x1000         BR1:    BEQZ R5, NEXT       ; always taken
 0x1004                 ADDI R4, R4, #4
 0x1008                 MULT R3, R5, R3
 0x100C                 ST   R3, 0(R4)
 0x1010                 SUBI R5, R5, #1
 0x1014        NEXT:    ADDI R1, R1, #1
 0x1018                 SLTI R2, R1, 100    ; repeat 100 times
 0x101C         BR2:    BNEZ R2, BR1
 0x1020                 NOP
 0x1024                 NOP
 0x1028                 NOP
```

	...	
BR1	0	0
	...	
BR2	0	0
	...	

BHT

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000
1	0x1000	0x1014

BTB

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1								A	P	F	B	I	J	R	E								
0x1018	SLTI R2, R1, 100																							
0x101C	BNEZ R2, LOOP																							
0x1000	BEQZ R5, NEXT																							
0x1014	ADDI R1, R1, #1																							

```
ADDRESS                  INSTRUCTION
0x1000          BR1:     BEQZ R5, NEXT        ; always taken
0x1004                   ADDI R4, R4, #4
0x1008                   MULT R3, R5, R3
0x100C                   ST   R3, 0(R4)
0x1010                   SUBI R5, R5, #1
0x1014         NEXT:     ADDI R1, R1, #1
0x1018                   SLTI R2, R1, 100     ; repeat 100 times
0x101C          BR2:     BNEZ R2, BR1
0x1020                   NOP
0x1024                   NOP
0x1028                   NOP
```

	...	
BR1	0	0
	...	
BR2	0	0
	...	

BHT

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000
1	0x1000	0x1014

BTB

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1								A	P	F	B	I	J	R	E								
0x1018	SLTI R2, R1, 100									A	P	F	B	I	J	R	E							
0x101C	BNEZ R2, LOOP																							
0x1000	BEQZ R5, NEXT																							
0x1014	ADDI R1, R1, #1																							

```
ADDRESS             INSTRUCTION
0x1000      BR1:    BEQZ R5, NEXT        ; always taken
0x1004              ADDI R4, R4, #4
0x1008              MULT R3, R5, R3
0x100C              ST   R3, 0(R4)
0x1010              SUBI R5, R5, #1
0x1014     NEXT:    ADDI R1, R1, #1
0x1018              SLTI R2, R1, 100     ; repeat 100 times
0x101C      BR2:    BNEZ R2, BR1
0x1020              NOP
0x1024              NOP
0x1028              NOP
```

	...	
BR1	0	0
	...	
BR2	0	0
	...	

BHT

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000
1	0x1000	0x1014

BTB

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1								A	P	F	B	I	J	R	E								
0x1018	SLTI R2, R1, 100									A	P	F	B	I	J	R	E							
0x101C	BNEZ R2, LOOP										A	P	F	B	I	J	R	E						
0x1000	BEQZ R5, NEXT																							
0x1014	ADDI R1, R1, #1																							

```
ADDRESS                 INSTRUCTION
 0x1000        BR1:   BEQZ R5, NEXT        ; always taken
 0x1004               ADDI R4, R4, #4
 0x1008               MULT R3, R5, R3
 0x100C               ST   R3, 0(R4)
 0x1010               SUBI R5, R5, #1
 0x1014       NEXT:   ADDI R1, R1, #1
 0x1018               SLTI R2, R1, 100     ; repeat 100 times
 0x101C        BR2:   BNEZ R2, BR1
 0x1020               NOP
 0x1024               NOP
 0x1028               NOP
```

	...	
BR1	0	0
	...	
BR2	0	0
	...	

BHT

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000
1	0x1000	0x1014

BTB

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1								A	P	F	B	I	J	R	E								
0x1018	SLTI R2, R1, 100									A	P	F	B	I	J	R	E							
0x101C	BNEZ R2, LOOP										A	P	F	B	I	J	R	E						
0x1000	BEQZ R5, NEXT																							
0x1014	ADDI R1, R1, #1																							

```
ADDRESS                 INSTRUCTION
0x1000          BR1:    BEQZ R5, NEXT        ; always taken
0x1004                  ADDI R4, R4, #4
0x1008                  MULT R3, R5, R3
0x100C                  ST   R3, 0(R4)
0x1010                  SUBI R5, R5, #1
0x1014          NEXT:   ADDI R1, R1, #1
0x1018                  SLTI R2, R1, 100     ; repeat 100 times
0x101C          BR2:    BNEZ R2, BR1
0x1020                  NOP
0x1024                  NOP
0x1028                  NOP
```

BHT

	...	
BR1	0	0
	...	
BR2	0	0
	...	

BTB

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000
1	0x1000	0x1014

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1								A	P	F	B	I	J	R	E								
0x1018	SLTI R2, R1, 100									A	P	F	B	I	J	R	E							
0x101C	BNEZ R2, LOOP										A	P	F	B	I	J	R	E						
0x1000	BEQZ R5, NEXT												A	P	F	B	I	J	R	E				
0x1014	ADDI R1, R1, #1																							

```
ADDRESS                 INSTRUCTION
 0x1000        BR1:   BEQZ R5, NEXT        ; always taken
 0x1004               ADDI R4, R4, #4
 0x1008               MULT R3, R5, R3
 0x100C               ST   R3, 0(R4)
 0x1010               SUBI R5, R5, #1
 0x1014       NEXT:   ADDI R1, R1, #1
 0x1018               SLTI R2, R1, 100     ; repeat 100 times
 0x101C        BR2:   BNEZ R2, BR1
 0x1020               NOP
 0x1024               NOP
 0x1028               NOP
```

	...	
BR1	0	0
	...	
BR2	0	0
	...	

BHT

(Valid) V	Entry PC	Predicted Target PC
1	0x101c	0x1000
1	0x1000	0x1014

BTB

TIME →

Address	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0x1000	BEQZ R5, NEXT	A	P	F	B	I	J	R	E															
0x1014	ADDI R1, R1, #1								A	P	F	B	I	J	R	E								
0x1018	SLTI R2, R1, 100									A	P	F	B	I	J	R	E							
0x101C	BNEZ R2, LOOP										A	P	F	B	I	J	R	E						
0x1000	BEQZ R5, NEXT												A	P	F	B	I	J	R	E				
0x1014	ADDI R1, R1, #1														A	P	F	B	I	J	R	E		

Problem 3.6: Out-of-order execution with Physical register file

Out-of-order execution with Physical register file

```
loop:   lw     r1, 0 (r2)
        addi   r2, r2, 4
        beqz   r1, skip
        addi   r3, r3, 1
skip:   bne    r2, r4, loop
```

Rename Table

R1	~~P1~~	P4	
R2	~~P2~~	P5	
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4		
P5		
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
P7
P8
P9
⋮

Reorder Buffer (ROB)

	use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
next to commit →	x		lw	p	P2			r1	P1	P4
	x		addi	p	P2			r2	P2	P5
	x		beqz		P4					
	x		addi	p	P3			r3	P3	P6
	x		bne		P5	p	P0			
next available →										

Out-of-order execution with Physical register file

1. The first three instructions from the next loop iteration are issued into the ROB.

Rename Table			
R1	~~P1~~	P4	
R2	~~P2~~	P5	
R3	~~P3~~	P6	
R4	P0		

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```

Physical Regs		
P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4		
P5		
P6		
P7		
P8		
P9		

Free List
~~P4~~
~~P5~~
~~P6~~
P7
P8
P9
⋮

Reorder Buffer (ROB)

next to commit →

next available →

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		lw	p	P2			r1	P1	P4
x		addi	p	P2			r2	P2	P5
x		beqz		P4					
x		addi	p	P3			r3	P3	P6
x		bne		P5	p	P0			

Out-of-order execution with Physical register file

1. The first three instructions from the next loop iteration are issued into the ROB.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	P5	
R3	~~P3~~	P6	
R4	P0		

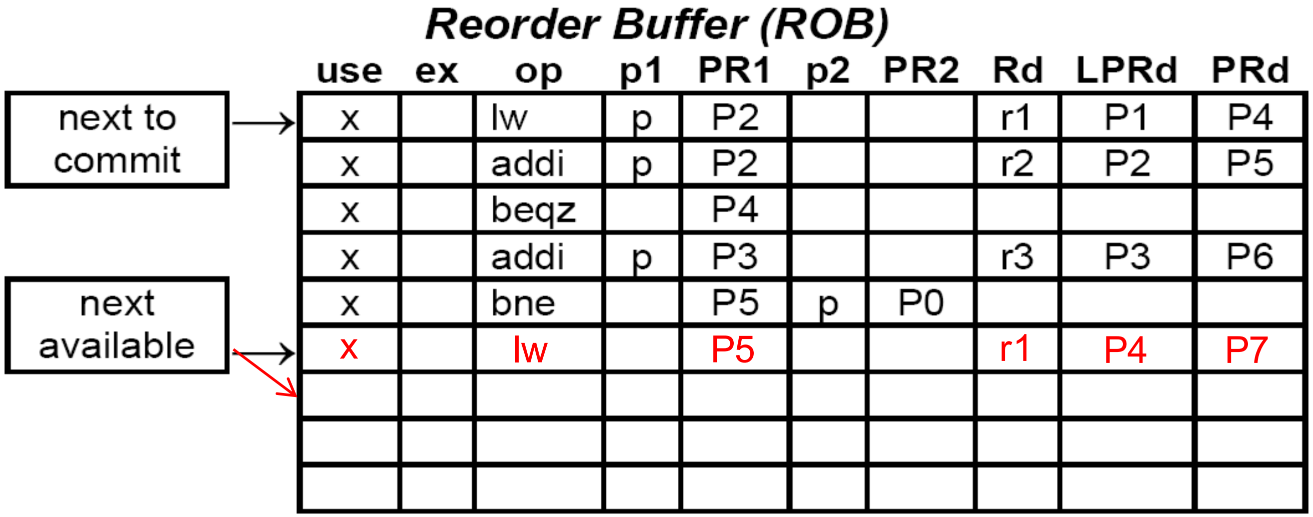
Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4		
P5		
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
P8
P9
⋮

```
loop:   lw      r1, 0 (r2)
        addi    r2, r2, 4
        beqz    r1, skip
        addi    r3, r3, 1
skip:   bne     r2, r4, loop
```



Reorder Buffer (ROB)

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		lw	p	P2			r1	P1	P4
x		addi	p	P2			r2	P2	P5
x		beqz		P4					
x		addi	p	P3			r3	P3	P6
x		bne		P5	p	P0			
x		lw		P5			r1	P4	P7

Out-of-order execution with Physical register file

1. The first three instructions from the next loop iteration are issued into the ROB.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4		
P5		
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```

Reorder Buffer (ROB)

next to commit →

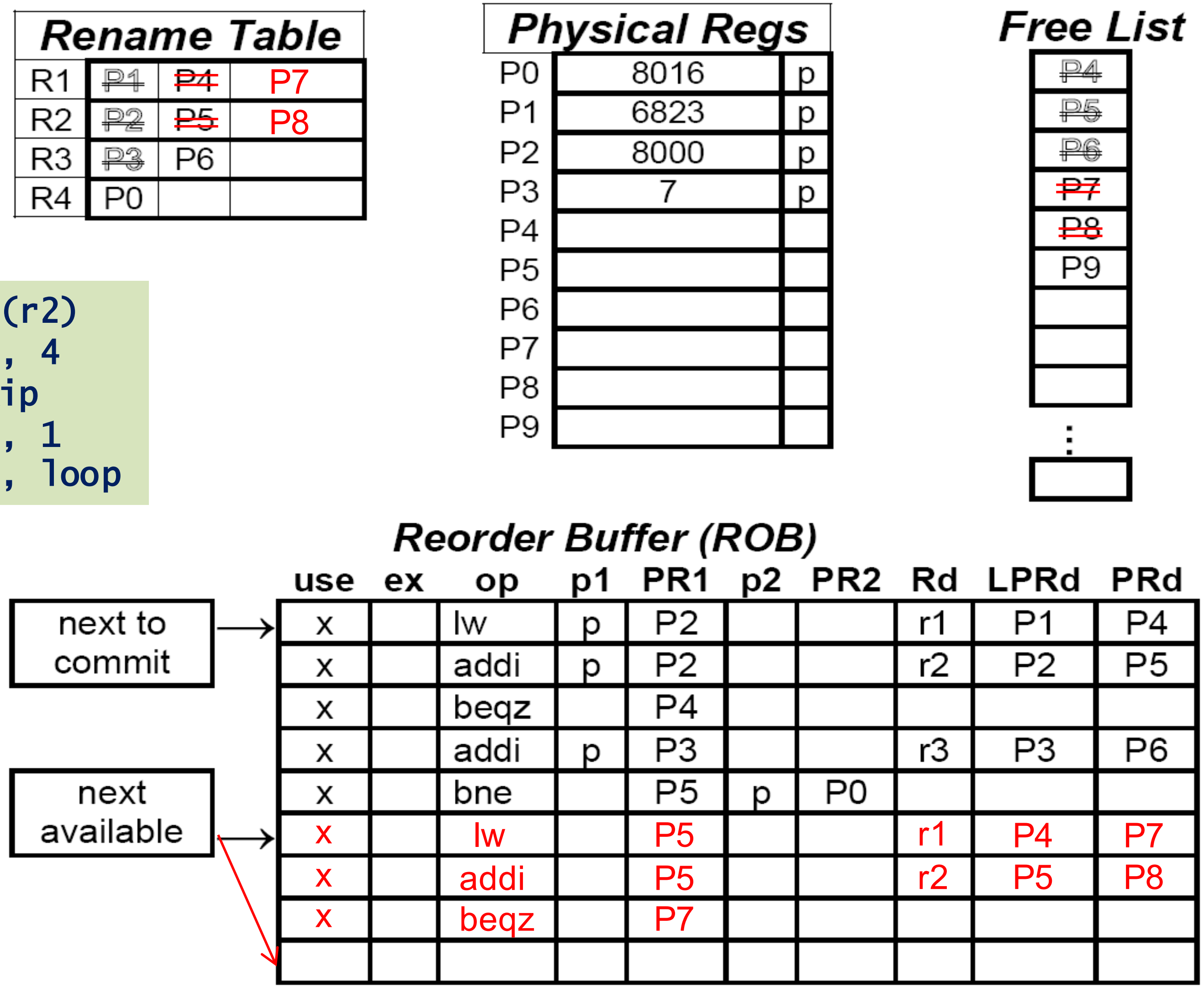
next available →

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		lw	p	P2			r1	P1	P4
x		addi	p	P2			r2	P2	P5
x		beqz		P4					
x		addi	p	P3			r3	P3	P6
x		bne		P5	p	P0			
x		lw		P5			r1	P4	P7
x		addi		P5			r2	P5	P8

Out-of-order execution with Physical register file

1. The first three instructions from the next loop iteration are issued into the ROB.

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```



Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4		
P5		
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

Reorder Buffer (ROB)

next to commit

next available

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		lw	p	P2			r1	P1	P4
x		addi	p	P2			r2	P2	P5
x		beqz		P4					
x		addi	p	P3			r3	P3	P6
x		bne		P5	p	P0			
x		lw		P5			r1	P4	P7
x		addi		P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

2. All instructions which are ready execute.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4		
P5		
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

```
loop:   lw      r1, 0 (r2)
        addi    r2, r2, 4
        beqz    r1, skip
        addi    r3, r3, 1
skip:   bne     r2, r4, loop
```

Reorder Buffer (ROB)

next to commit →

next available →

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		lw	p	P2			r1	P1	P4
x		addi	p	P2			r2	P2	P5
x		beqz		P4					
x		addi	p	P3			r3	P3	P6
x		bne		P5	p	P0			
x		lw		P5			r1	P4	P7
x		addi		P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

2. All instructions which are ready execute.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4		
P5		
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

```
loop:   lw     r1, 0 (r2)
        addi   r2, r2, 4
        beqz   r1, skip
        addi   r3, r3, 1
skip:   bne    r2, r4, loop
```

Reorder Buffer (ROB)

next to commit

next available

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x	x	lw	p	P2			r1	P1	P4
x	x	addi	p	P2			r2	P2	P5
x		beqz		P4					
x	x	addi	p	P3			r3	P3	P6
x		bne		P5	p	P0			
x		lw		P5			r1	P4	P7
x		addi		P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

2. All instructions which are ready execute.

```
loop:   lw      r1, 0 (r2)
        addi    r2, r2, 4
        beqz    r1, skip
        addi    r3, r3, 1
skip:   bne     r2, r4, loop
```

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4	0	p
P5	8004	p
P6	8	p
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

Reorder Buffer (ROB)

next to commit →

next available →

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x	x	lw	p	P2			r1	P1	P4
x	x	addi	p	P2			r2	P2	P5
x		beqz		P4					
x	x	addi	p	P3			r3	P3	P6
x		bne		P5	p	P0			
x		lw		P5			r1	P4	P7
x		addi		P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

2. All instructions which are ready execute.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4	0	p
P5	8004	p
P6	8	p
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

```
loop:   lw      r1, 0 (r2)
        addi    r2, r2, 4
        beqz    r1, skip
        addi    r3, r3, 1
skip:   bne     r2, r4, loop
```

Reorder Buffer (ROB)

next to commit →

next available →

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x	x	lw	p	P2			r1	P1	P4
x	x	addi	p	P2			r2	P2	P5
x		beqz	p	P4					
x	x	addi	p	P3			r3	P3	P6
x		bne	p	P5	p	P0			
x		lw	p	P5			r1	P4	P7
x		addi	p	P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

3. As many instructions as possible commit.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4	0	p
P5	8004	p
P6	8	p
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```

Reorder Buffer (ROB)

next to commit

next available

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x	x	lw	p	P2			r1	P1	P4
x	x	addi	p	P2			r2	P2	P5
x		beqz	p	P4					
x	x	addi	p	P3			r3	P3	P6
x		bne	p	P5	p	P0			
x		lw	p	P5			r1	P4	P7
x		addi	p	P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

3. As many instructions as possible commit.

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	6823	p
P2	8000	p
P3	7	p
P4	0	p
P5	8004	p
P6	8	p
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
⋮

Reorder Buffer (ROB)

next to commit

next available

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
~~x~~	x	lw	p	P2			r1	P1	P4
x	x	addi	p	P2			r2	P2	P5
x		beqz	p	P4					
x	x	addi	p	P3			r3	P3	P6
x		bne	p	P5	p	P0			
x		lw	p	P5			r1	P4	P7
x		addi	p	P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

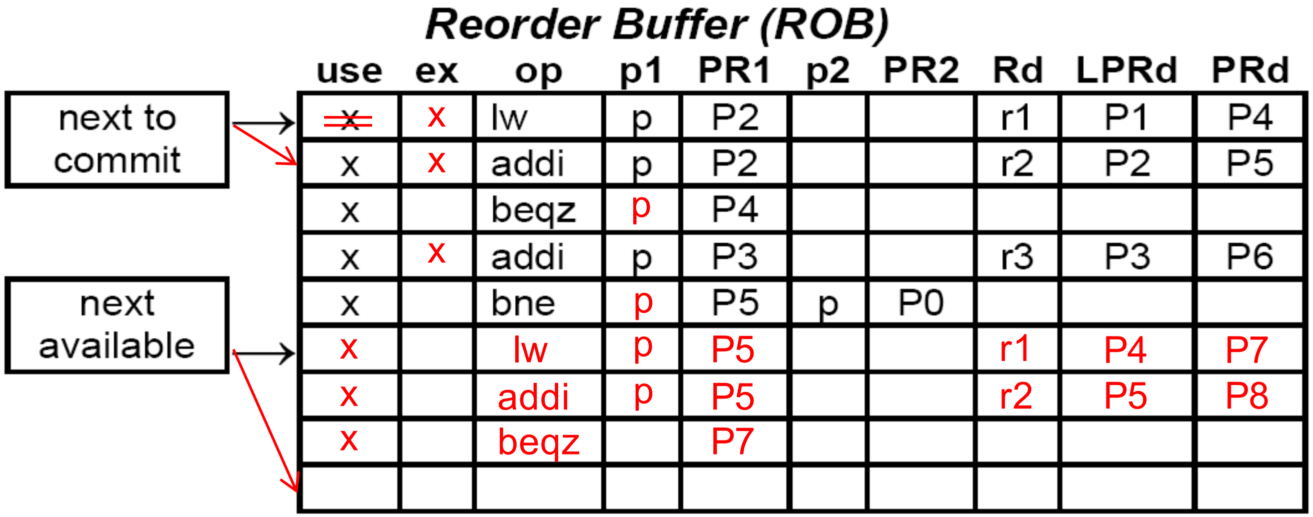
3. As many instructions as possible commit.

Rename Table			
R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs		
P0	8016	p
P1	~~6823~~	~~p~~
P2	8000	p
P3	7	p
P4	0	p
P5	8004	p
P6	8	p
P7		
P8		
P9		

Free List
~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
P1
⋮

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```



Reorder Buffer (ROB)

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
~~x~~	x	lw	p	P2			r1	P1	P4
x	x	addi	p	P2			r2	P2	P5
x		beqz	p	P4					
x	x	addi	p	P3			r3	P3	P6
x		bne	p	P5	p	P0			
x		lw	p	P5			r1	P4	P7
x		addi	p	P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

3. As many instructions as possible commit.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

P0	8016	p
P1	~~6823~~	~~p~~
P2	~~8000~~	~~p~~
P3	7	p
P4	0	p
P5	8004	p
P6	8	p
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
P1
P2
⋮

```
loop:   lw      r1, 0 (r2)
        addi    r2, r2, 4
        beqz    r1, skip
        addi    r3, r3, 1
skip:   bne     r2, r4, loop
```

Reorder Buffer (ROB)

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
~~x~~	x	lw	p	P2			r1	P1	P4
~~x~~	x	addi	p	P2			r2	P2	P5
x		beqz	p	P4					
x	x	addi	p	P3			r3	P3	P6
x		bne	p	P5	p	P0			
x		lw	p	P5			r1	P4	P7
x		addi	p	P5			r2	P5	P8
x		beqz		P7					

next to commit

next available

Out-of-order execution with Physical register file

4. The processor detects that the beqz instruction has been mispredicted, and thus, recovery action is taken.

Rename Table

R1	~~P1~~	~~P4~~	P7
R2	~~P2~~	~~P5~~	P8
R3	~~P3~~	P6	
R4	P0		

Physical Regs

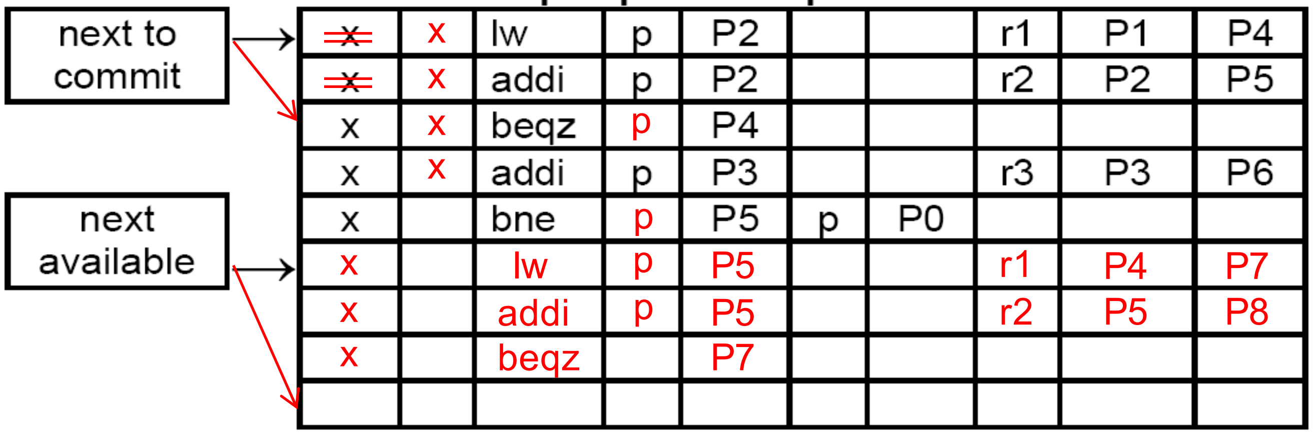
P0	8016	p
P1	~~6823~~	~~p~~
P2	~~8000~~	~~p~~
P3	7	p
P4	0	p
P5	8004	p
P6	8	p
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
~~P7~~
~~P8~~
P9
P1
P2
⋮

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```

Reorder Buffer (ROB)



use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
~~x~~	x	lw	p	P2			r1	P1	P4
~~x~~	x	addi	p	P2			r2	P2	P5
x	x	beqz	p	P4					
x	x	addi	p	P3			r3	P3	P6
x		bne	p	P5	p	P0			
x		lw	p	P5			r1	P4	P7
x		addi	p	P5			r2	P5	P8
x		beqz		P7					

Out-of-order execution with Physical register file

4. The processor detects that the beqz instruction has been mispredicted, and thus, recovery action is taken.

Rename Table

R1	~~P1~~	~~P4~~	P4
R2	~~P2~~	~~P5~~	P5
R3	~~P3~~	P6	P3
R4	P0		

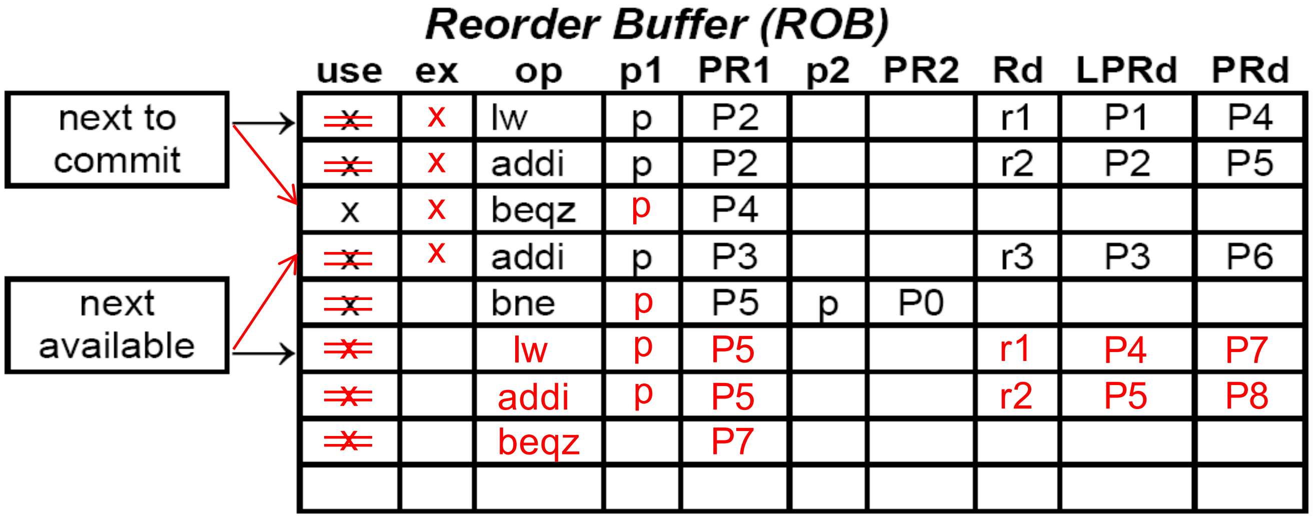
Physical Regs

P0	8016	p
P1	~~6823~~	~~p~~
P2	~~8000~~	~~p~~
P3	7	p
P4	0	p
P5	8004	p
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
P7
P8
P9
P1
P2
P6
⋮

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```



Reorder Buffer (ROB)

next to commit

next available

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
~~x~~	x	lw	p	P2			r1	P1	P4
~~x~~	x	addi	p	P2			r2	P2	P5
x	x	beqz	p	P4					
~~x~~	x	addi	p	P3			r3	P3	P6
~~x~~		bne	p	P5	p	P0			
~~x~~		lw	p	P5			r1	P4	P7
~~x~~		addi	p	P5			r2	P5	P8
~~x~~		beqz		P7					

Out-of-order execution with Physical register file

5. The beqz instruction commits.

Rename Table

R1	~~P1~~	~~P4~~	P4
R2	~~P2~~	~~P5~~	P5
R3	~~P3~~	P6	P3
R4	P0		

Physical Regs

P0	8016	p
P1	~~6823~~	~~p~~
P2	~~8000~~	~~p~~
P3	7	p
P4	0	p
P5	8004	p
P6		
P7		
P8		
P9		

Free List

~~P4~~
~~P5~~
~~P6~~
P7
P8
P9
P1
P2
P6
⋮

```
loop:   lw      r1, 0 (r2)
        addi    r2, r2, 4
        beqz    r1, skip
        addi    r3, r3, 1
skip:   bne     r2, r4, loop
```

Reorder Buffer (ROB)

next to commit

next available

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
~~x~~	x	lw	p	P2			r1	P1	P4
~~x~~	x	addi	p	P2			r2	P2	P5
x	x	beqz	p	P4					
~~x~~	x	addi	p	P3			r3	P3	P6
~~x~~		bne	p	P5	p	P0			
~~x~~		lw	p	P5			r1	P4	P7
~~x~~		addi	p	P5			r2	P5	P8
~~x~~		beqz		P7					

Out-of-order execution with Physical register file

5. The beqz instruction commits.

Rename Table

R1	P4		
R2	P5		
R3	P3		
R4	P0		

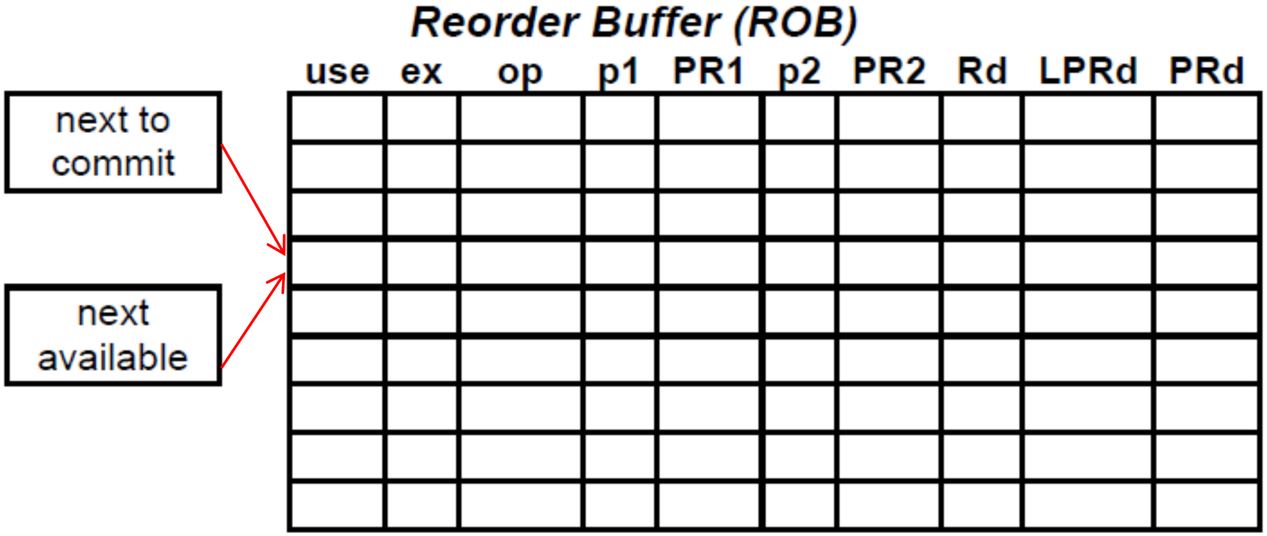
Physical Regs

P0	8016	p
P1		
P2		
P3	7	p
P4	0	p
P5	8004	p
P6		
P7		
P8		
P9		

Free List

P9
P1
P2
P6
P7
P8
⋮

```
loop:  lw    r1, 0 (r2)
       addi  r2, r2, 4
       beqz  r1, skip
       addi  r3, r3, 1
skip:  bne   r2, r4, loop
```



Reorder Buffer (ROB)

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd

Out-of-order execution with Physical register file

6. The correct next instruction is fetched and is written into the ROB.

Rename Table

R1	P4		
R2	P5		
R3	P3		
R4	P0		

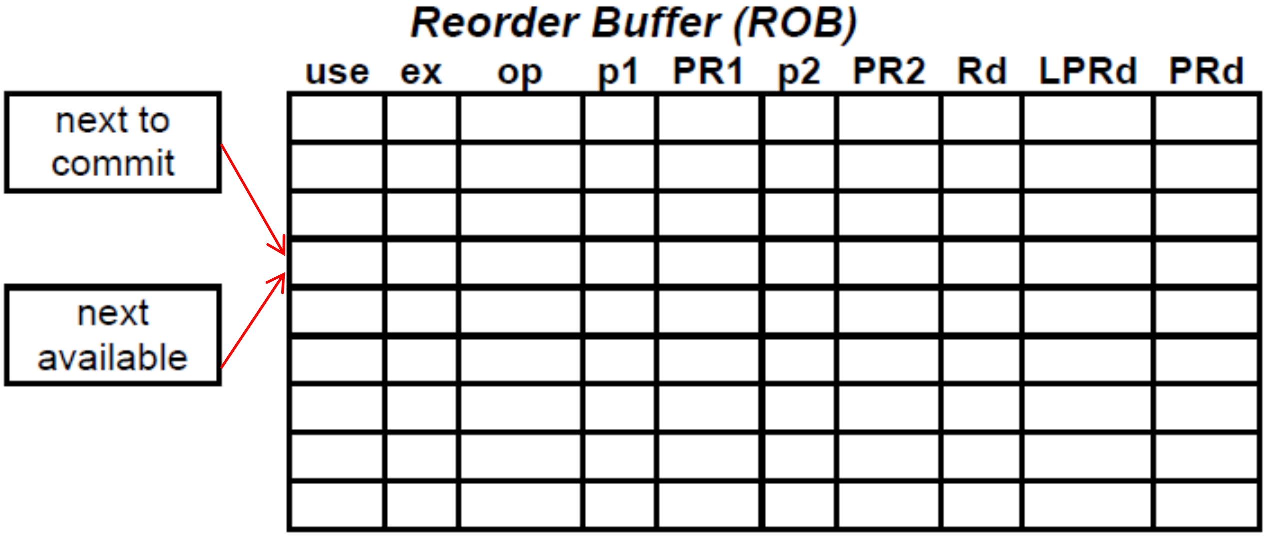
Physical Regs

P0	8016	p
P1		
P2		
P3	7	p
P4	0	p
P5	8004	p
P6		
P7		
P8		
P9		

Free List

P9
P1
P2
P6
P7
P8
⋮

```
loop:   lw     r1, 0 (r2)
        addi   r2, r2, 4
        beqz   r1, skip
        addi   r3, r3, 1
skip:   bne    r2, r4, loop
```



Reorder Buffer (ROB)

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd

Out-of-order execution with Physical register file

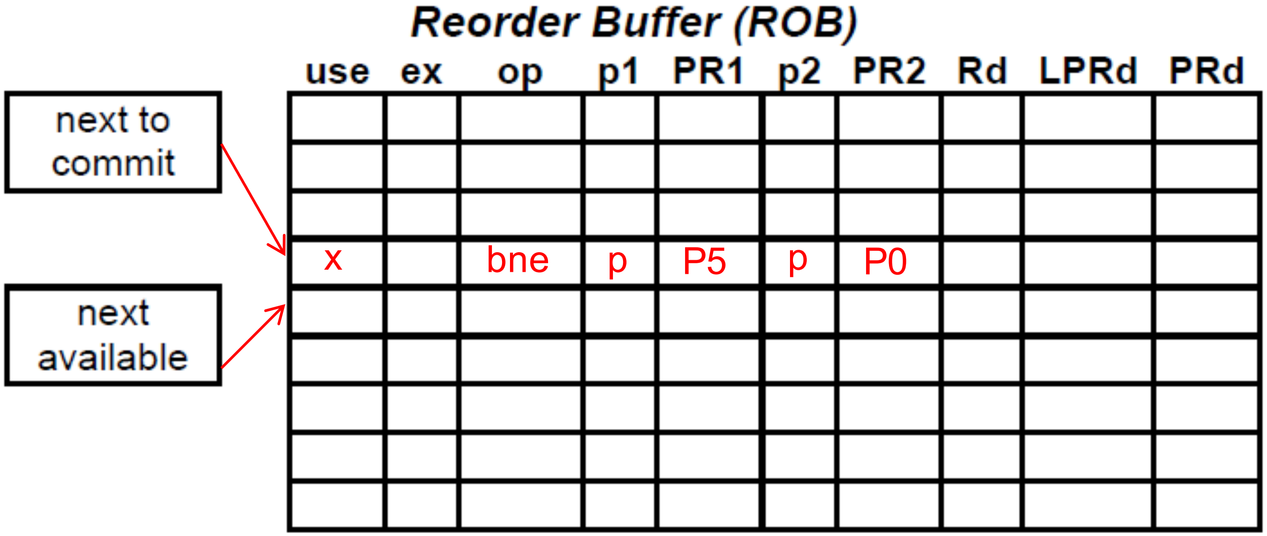
6. The correct next instruction is fetched and is written into the ROB.

Rename Table			
R1	P4		
R2	P5		
R3	P3		
R4	P0		

Physical Regs		
P0	8016	p
P1		
P2		
P3	7	p
P4	0	p
P5	8004	p
P6		
P7		
P8		
P9		

Free List
P9
P1
P2
P6
P7
P8
⋮

```
loop:   lw     r1, 0 (r2)
        addi   r2, r2, 4
        beqz   r1, skip
        addi   r3, r3, 1
skip:   bne    r2, r4, loop
```



Reorder Buffer (ROB)

use	ex	op	p1	PR1	p2	PR2	Rd	LPRd	PRd
x		bne	p	P5	p	P0			

That’s all for today…