

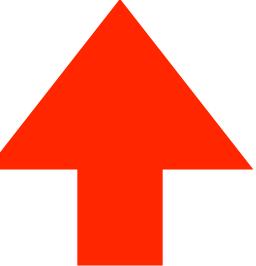
Dynamic Instruction Scheduling

(III)

Hung-Wei Tseng

What do you need to execution an instruction?

- Whenever the instruction is decoded — put decoded instruction somewhere
- Whenever the inputs are ready — **all data dependencies are resolved**
- Whenever the target functional unit is available



- This instruction has completed its own work in the current stage
- No other instruction is occupying the next stage
- The next stage has all its inputs ready

Tomasulo in motion

① ld X6, 0(X10) ② add X7, X6, X12 ③ sd X7, 0(X10) ④ addi X10, X10, 8 ⑤ bne X10, X5, LOOP

⑥ ld X6, 0(X10) ⑦ add X7, X6, X12 ⑧ sd X7, 0(X10) ⑨ addi X10, X10, 8 ⑩ bne X10, X5, LOOP

D AQ AR I MEM INT WB

D I I I I INT WB

D AQ AR I I I I MEM WB

D D I INT WB

D I I BR WB

D AQ AR I MEM WB

D D I I I I INT WB

D AQ AR I I I I MEM

D I I INT WB

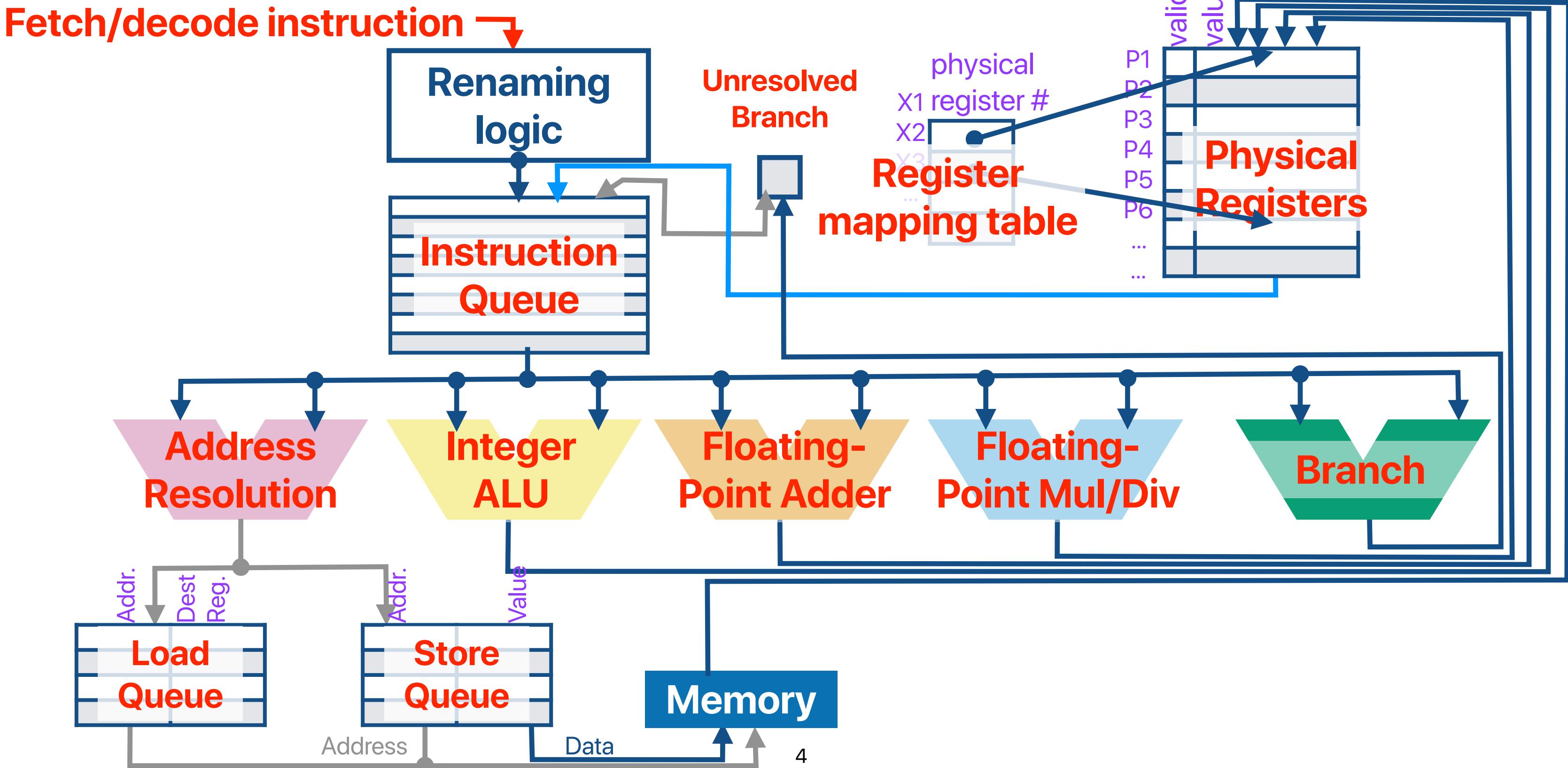
D I I I I I I I I

no reservation station for add!

Takes 13 cycles to issue all instructions

	INST	Vj	Vk	Vst	Qj	Qk	Qst	A	Inst #
LD1	ld	0	[X10]						1
LD2	ld	0	INT2						6
LD3									
ST1	sd	0	[X10]	INT1					3
ST2	sd	0	[X10]	INT2					8
ST3									
INT1	add	8	INT2						9
INT2	add		[X12]	[LD2]					7
MUL1									
MUL2									
BR	br		[X5]	INT1					10

Overview of a processor supporting register renaming

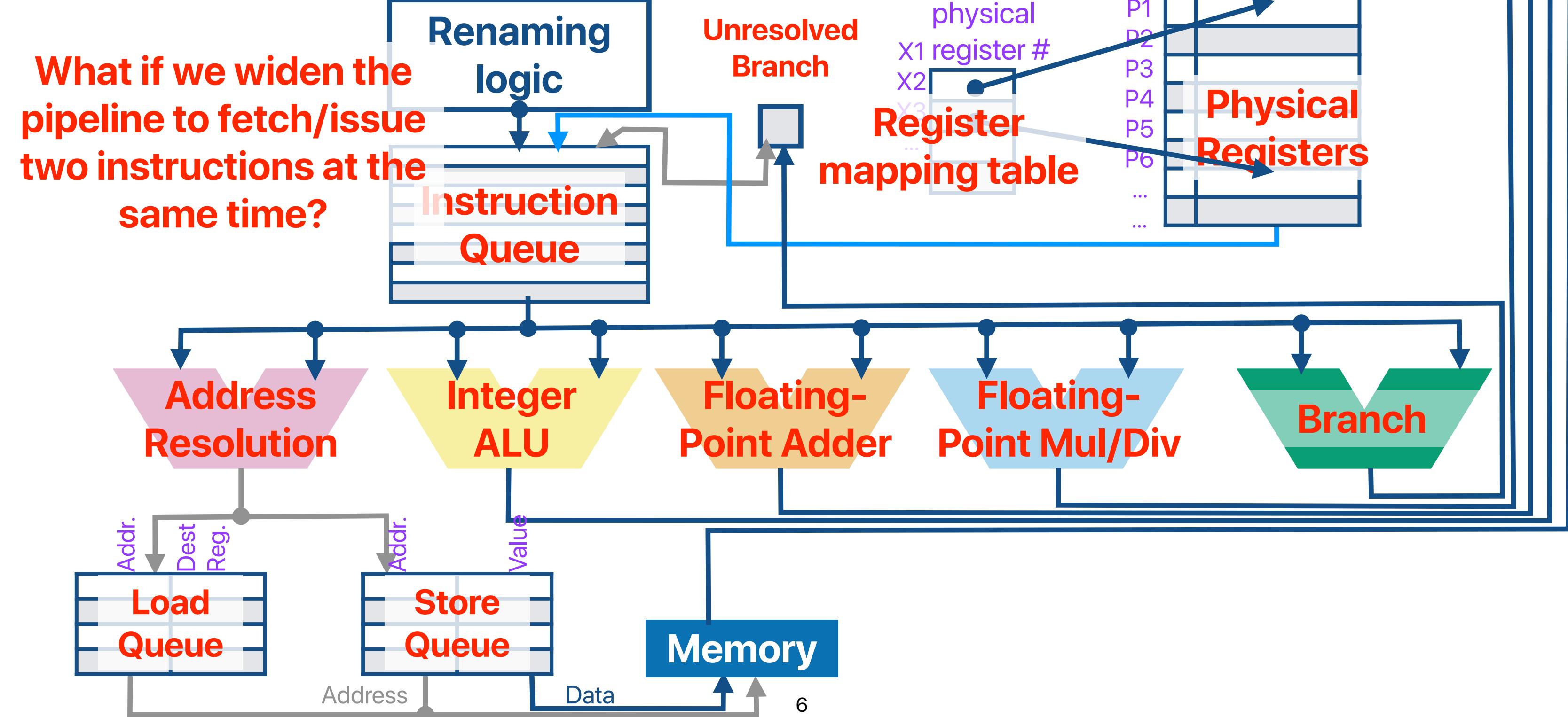


Register renaming in motion

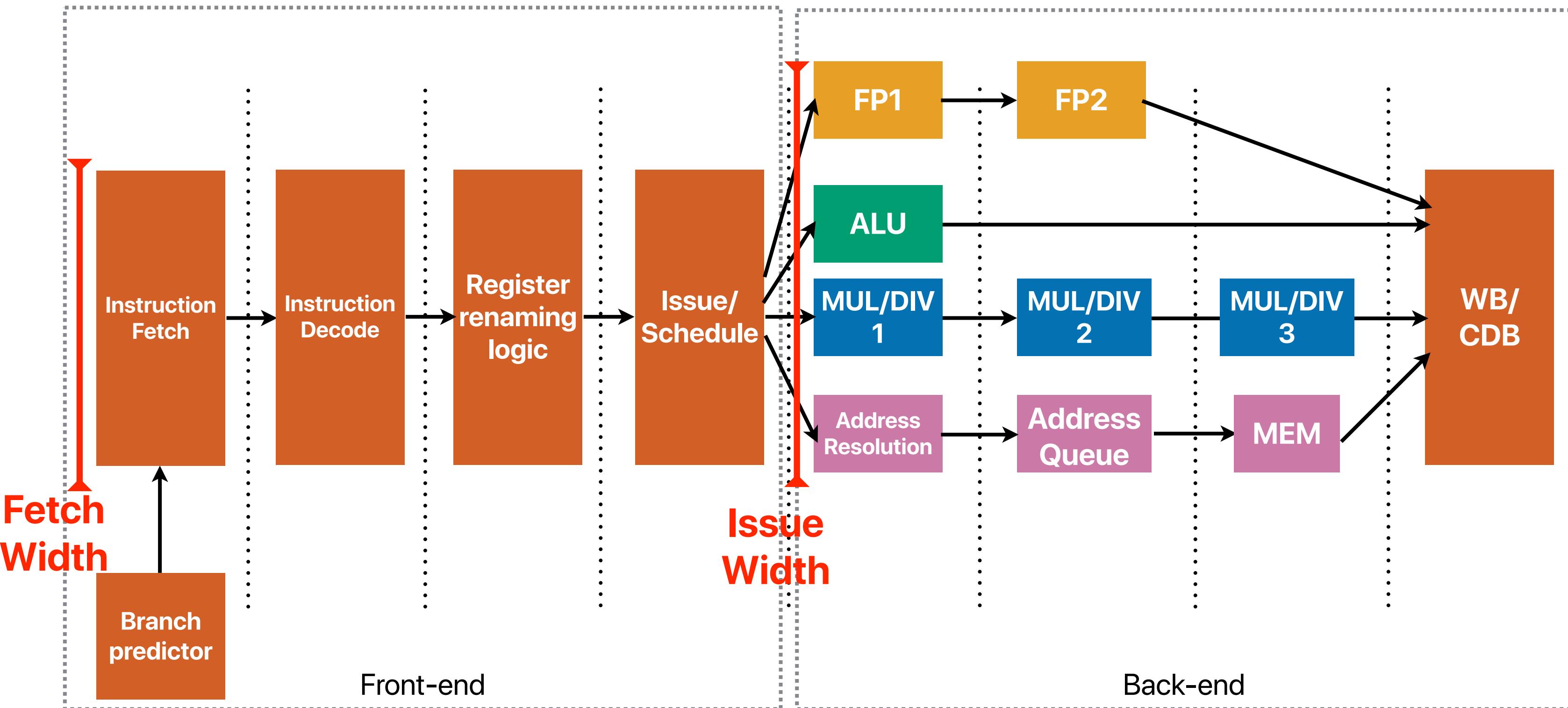
The diagram illustrates the execution of 10 instructions over 12 cycles through a 5-stage pipeline (R, I, AR, LSQ, MEM) and a Branch (BR) stage. The pipeline is color-coded: R (orange), I (yellow), AR (green), LSQ (blue), MEM (red), and BR (purple). The first instruction (ld X6,0(X10)) starts at cycle 1 and reaches the WB stage by cycle 5. Subsequent instructions (add, sd, addi, bne) are issued sequentially, with their execution paths overlapping. By cycle 12, all instructions have completed their execution stages.

Overview of a processor supporting register renaming

Fetch/decode instruction →



Recap: Super Scalar Pipeline

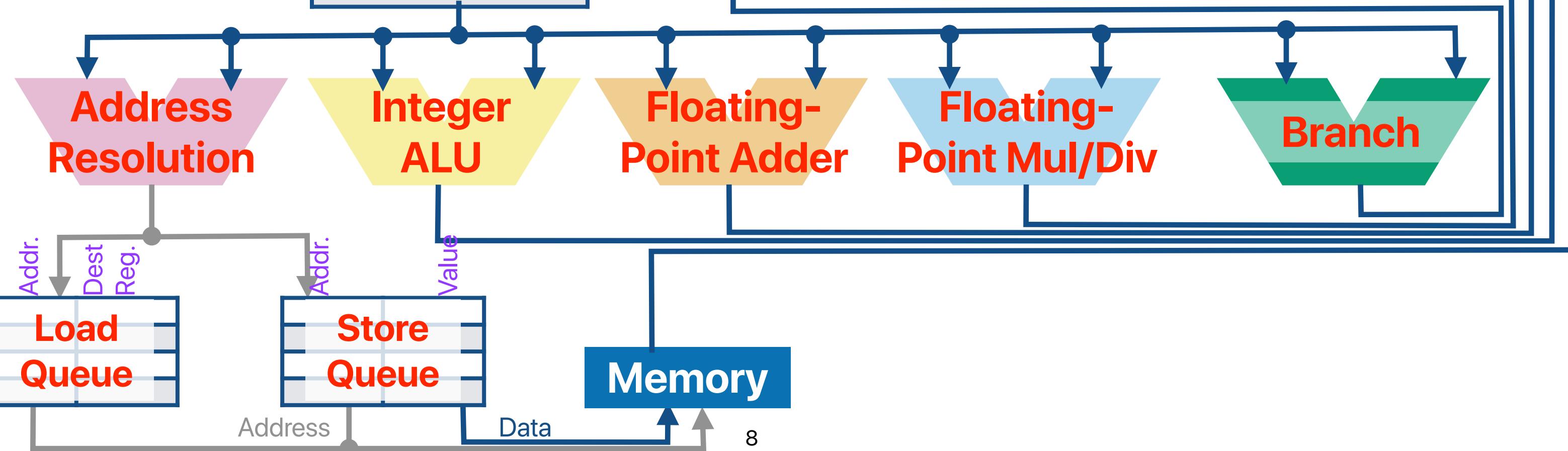
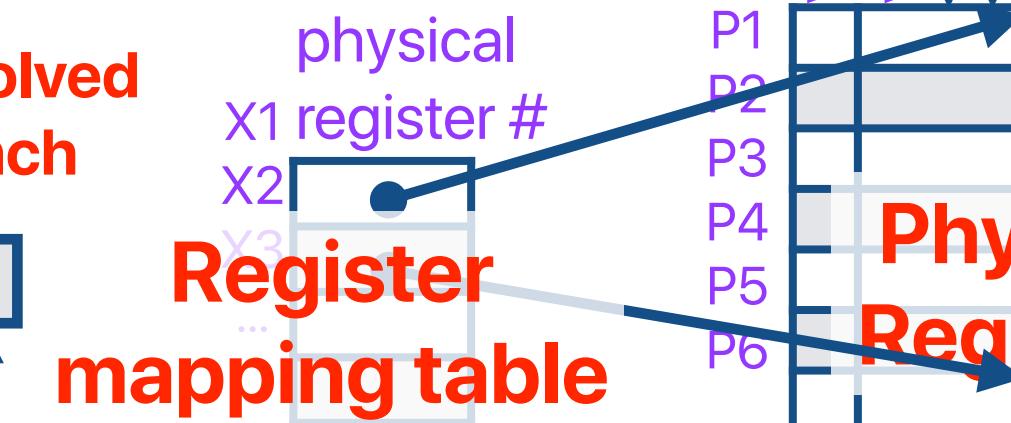


Overview of a processor supporting register renaming

Fetch/decode instruction →



Unresolved Branch



2-issue RR processor in motion

① ld X6, 0(X10) R
 ② add X7, X6, X12 R

③ sd X7, 0(X10)
 ④ addi X10, X10, 8
 ⑤ bne X10, X5, LOOP
 ⑥ ld X6, 0(X10)
 ⑦ add X7, X6, X12
 ⑧ sd X7, 0(X10)
 ⑨ addi X10, X10, 8
 ⑩ bne X10, X5, LOOP

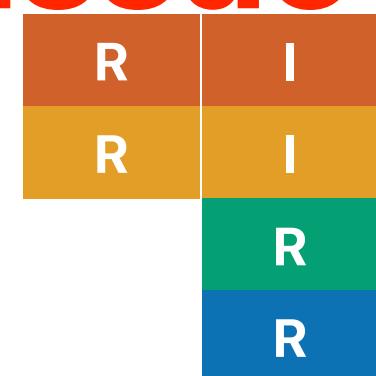
Renamed instruction		
1	ld P1, 0(X10)	
2	add P2, P1, X12	
3		
4		
5		
6		
7		
8		
9		
10		

Physical Register	
X5	
X6	P1
X7	P2
X10	
X12	

Valid Value In use			Valid Value In use		
P1	0	1	P6		
P2	0	1	P7		
P3			P8		
P4			P9		
P5			P10		

2-issue RR processor in motion

- ① ld X6, 0(X10)
- ② add X7, X6, X12
- ③ sd X7, 0(X10)
- ④ addi X10, X10, 8
- ⑤ bne X10, X5, LOOP
- ⑥ ld X6, 0(X10)
- ⑦ add X7, X6, X12
- ⑧ sd X7, 0(X10)
- ⑨ addi X10, X10, 8
- ⑩ bne X10, X5, LOOP



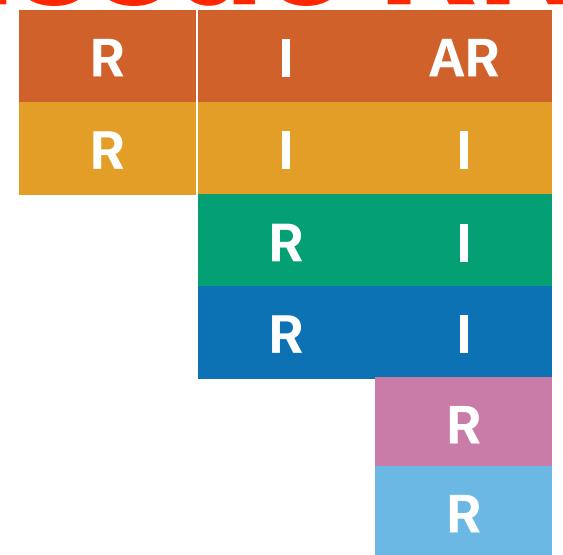
Renamed instruction		
1	ld P1, 0(X10)	
2	add P2, P1, X12	
3	sd P2, 0(X10)	
4	addi P3, X10, 8	
5		
6		
7		
8		
9		
10		

Physical Register	
X5	
X6	P1
X7	P2
X10	P3
X12	

Valid Value In use			Valid Value In use		
P1	0	1	P6		
P2	0	1	P7		
P3	0	1	P8		
P4			P9		
P5			P10		

2-issue RR processor in motion

- ① ld X6, 0(X10)
- ② add X7, X6, X12
- ③ sd X7, 0(X10)
- ④ addi X10, X10, 8
- ⑤ bne X10, X5, LOOP
- ⑥ ld X6, 0(X10)
- ⑦ add X7, X6, X12
- ⑧ sd X7, 0(X10)
- ⑨ addi X10, X10, 8
- ⑩ bne X10, X5, LOOP



Renamed instruction		
1	ld P1, 0(X10)	
2	add P2, P1, X12	
3	sd P2, 0(X10)	
4	addi P3, X10, 8	
5	bne P3, X5, LOOP	
6	ld P4, 0(P3)	
7		
8		
9		
10		

Physical Register	
X5	
X6	P1
X7	P2
X10	P3
X12	

Valid Value In use			Valid Value In use		
P1	0	1	P6		
P2	0	1	P7		
P3	0	1	P8		
P4	0	1	P9		
P5			P10		

2-issue RR processor in motion

①	ld	X6, 0(X10)	R	I	AR	AQ
②	add	X7, X6, X12	R	I	I	I
③	sd	X7, 0(X10)	R	I	I	I
④	addi	X10, X10, 8	R	I	INT	
⑤	bne	X10, X5, LOOP		R	I	
⑥	ld	X6, 0(X10)		R	I	
⑦	add	X7, X6, X12			R	
⑧	sd	X7, 0(X10)			R	
⑨	addi	X10, X10, 8				
⑩	bne	X10, X5, LOOP				

Renamed instruction	
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	
10	

Physical Register	
X5	
X6	P1
X7	P5
X10	P3
X12	

Valid Value In use			Valid Value In use		
P1	0	1	P6		
P2	0	1	P7		
P3	0	1	P8		
P4	0	1	P9		
P5	0	1	P10		

2-issue RR processor in motion

①	ld	X6, 0(X10)	R	I	AR	AQ	MEM
②	add	X7, X6, X12	R	I	I	I	I
③	sd	X7, 0(X10)	R	I	I	I	I
④	addi	X10, X10, 8	R	I	INT	WB	
⑤	bne	X10, X5, LOOP		R	I	I	
⑥	ld	X6, 0(X10)		R	I	I	
⑦	add	X7, X6, X12			R	I	
⑧	sd	X7, 0(X10)			R	I	
⑨	addi	X10, X10, 8				R	
⑩	bne	X10, X5, LOOP				R	

Renamed instruction	
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

Physical Register	
X5	
X6	P1
X7	P5
X10	P3
X12	

	Valid	Value	In use		Valid	Value	In use
P1	0		1	P6			
P2	0		1	P7			
P3	1		1	P8			
P4	0		1	P9			
P5	0		1	P10			

2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB
①	ld	X6, 0(X10)	R	I				
②	add	X7, X6, X12	R	I	I	I	I	I
③	sd	X7, 0(X10)	R	I	I	I	I	I
④	addi	X10, X10, 8	R	I	INT	WB		
⑤	bne	X10, X5, LOOP		R	I	I		BR
⑥	ld	X6, 0(X10)		R	I	I		AR
⑦	add	X7, X6, X12		R	I	I		
⑧	sd	X7, 0(X10)		R	I	I		
⑨	addi	X10, X10, 8		R		I		
⑩	bne	X10, X5, LOOP		R		I		

	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6			
P2	0		1	P7			
P3	1		1	P8			
P4	0		1	P9			
P5	0		1	P10			

2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB	
①	ld	X6, 0(X10)	R	I					
②	add	X7, X6, X12	R	I	I	I	I	I	INT
③	sd	X7, 0(X10)	R	I	I	I	I	I	
④	addi	X10, X10, 8	R	I	INT		WB		
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	
⑦	add	X7, X6, X12		R	I	I	I	I	
⑧	sd	X7, 0(X10)		R	I	I	I	I	
⑨	addi	X10, X10, 8		R	I	I			
⑩	bne	X10, X5, LOOP		R	I	I			

Renamed instruction		Physical Register		Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	0	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	0	1	P9		
5	bne P3, X5, LOOP	X12		P5	0	1	P10		
6	ld P4, 0(P3)								
7	add P5, P1, X12								
8	sd P5, 0(P3)								
9	addi P6, P3, 8								
10	bne P6, 0(X10)								

2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB	
①	ld	X6, 0(X10)	R	I					
②	add	X7, X6, X12	R	I	I	I	I	I	INT WB
③	sd	X7, 0(X10)	R	I	I	I	I	I	I
④	addi	X10, X10, 8	R	I	INT	WB			
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	MEM
⑦	add	X7, X6, X12		R	I	I	I	I	I
⑧	sd	X7, 0(X10)		R	I	I	I	I	I
⑨	addi	X10, X10, 8		R	I	I	I	INT	
⑩	bne	X10, X5, LOOP		R	I	I	I		

Renamed instruction	Physical Register	Valid			Valid		
		Value	In use		Value	In use	
1 ld P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	0	1	P9
5 bne P3, X5, LOOP	X12			P5	0	1	P10
6 ld P4, 0(P3)							
7 add P5, P1, X12							
8 sd P5, 0(P3)							
9 addi P6, P3, 8							
10 bne P6, 0(X10)							

2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB	
①	ld	X6, 0(X10)	R	I	AR	AQ	MEM	WB	
②	add	X7, X6, X12	R	I	I	I	I	I	INT WB
③	sd	X7, 0(X10)	R	I	I	I	I	I	AR
④	addi	X10, X10, 8	R	I	INT	WB			
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	MEM WB
⑦	add	X7, X6, X12		R	I	I	I	I	I
⑧	sd	X7, 0(X10)		R	I	I	I	I	I
⑨	addi	X10, X10, 8		R	I	I	I	INT	WB
⑩	bne	X10, X5, LOOP		R	I	I	I	I	

	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	Physical Register	Valid Value In use			Valid Value In use		
		Valid	Value	In use	Valid	Value	In use
	X5				P1	1	1
	X6		P1		P2	1	1
	X7		P5		P3	1	1
	X10		P3		P4	1	1
	X12				P5	0	1

2-issue RR processor in motion

①	ld	X6, 0(X10)	R	I	AR	AQ	MEM	WB			
②	add	X7, X6, X12	R	I	I	I	I	I	INT	WB	
③	sd	X7, 0(X10)	R	I	I	I	I	I	I	AR	AQ
④	addi	X10, X10, 8	R	I	INT	WB					
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB			
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	MEM	WB	
⑦	add	X7, X6, X12		R	I	I	I	I	I	INT	
⑧	sd	X7, 0(X10)		R	I	I	I	I	I	I	
⑨	addi	X10, X10, 8		R	I	I	INT	WB			
⑩	bne	X10, X5, LOOP		R	I	I	I	I	I	BR	

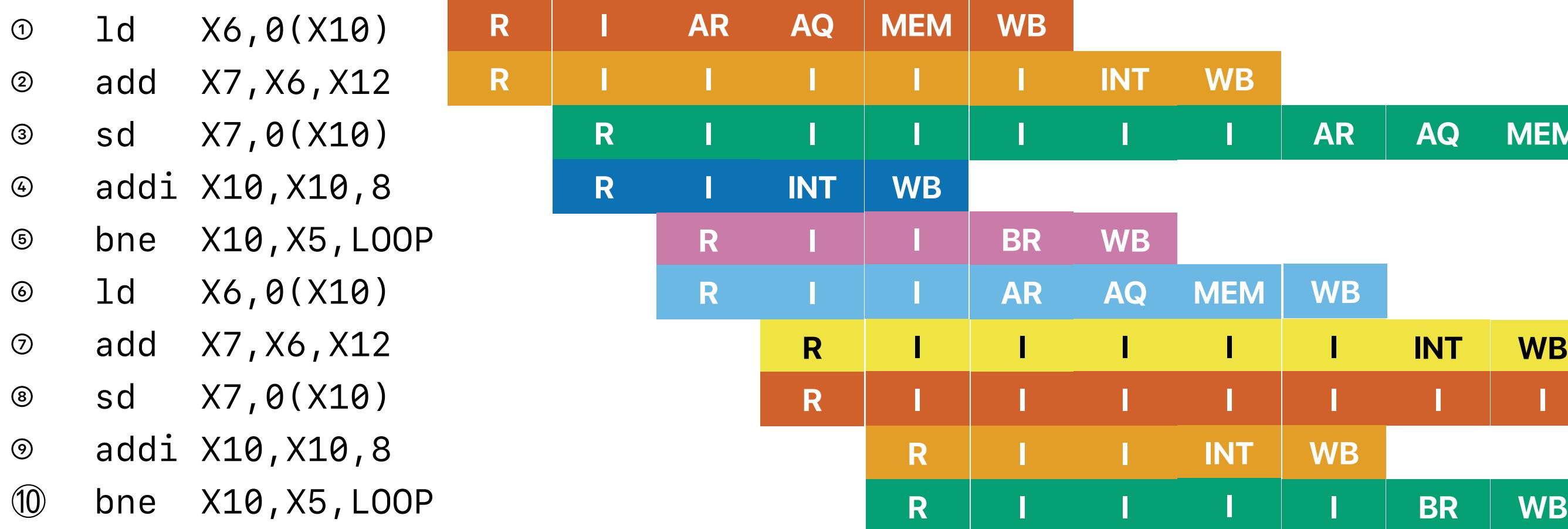
Renamed instruction	Physical Register	Valid			Valid		
		Value	In use		Value	In use	
1 ld P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	1	1	P9
5 bne P3, X5, LOOP	X12			P5	0	1	P10
6 ld P4, 0(P3)							
7 add P5, P1, X12							
8 sd P5, 0(P3)							
9 addi P6, P3, 8							
10 bne P6, 0(X10)							

2-issue RR processor in motion

①	ld	X6, 0(X10)	R	I	AR	AQ	MEM	WB			
②	add	X7, X6, X12	R	I	I	I	I	I	INT	WB	
③	sd	X7, 0(X10)	R	I	I	I	I	I	I	AR	AQ
④	addi	X10, X10, 8	R	I	INT	WB					
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB			
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	MEM	WB	
⑦	add	X7, X6, X12		R	I	I	I	I	I	INT	
⑧	sd	X7, 0(X10)		R	I	I	I	I	I	I	
⑨	addi	X10, X10, 8		R	I	I	INT	WB			
⑩	bne	X10, X5, LOOP		R	I	I	I	I	I	BR	

Renamed instruction	Physical Register	Valid			Valid		
		Value	In use		Value	In use	
1 ld P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	1	1	P9
5 bne P3, X5, LOOP	X12			P5	0	1	P10
6 ld P4, 0(P3)							
7 add P5, P1, X12							
8 sd P5, 0(P3)							
9 addi P6, P3, 8							
10 bne P6, 0(X10)							

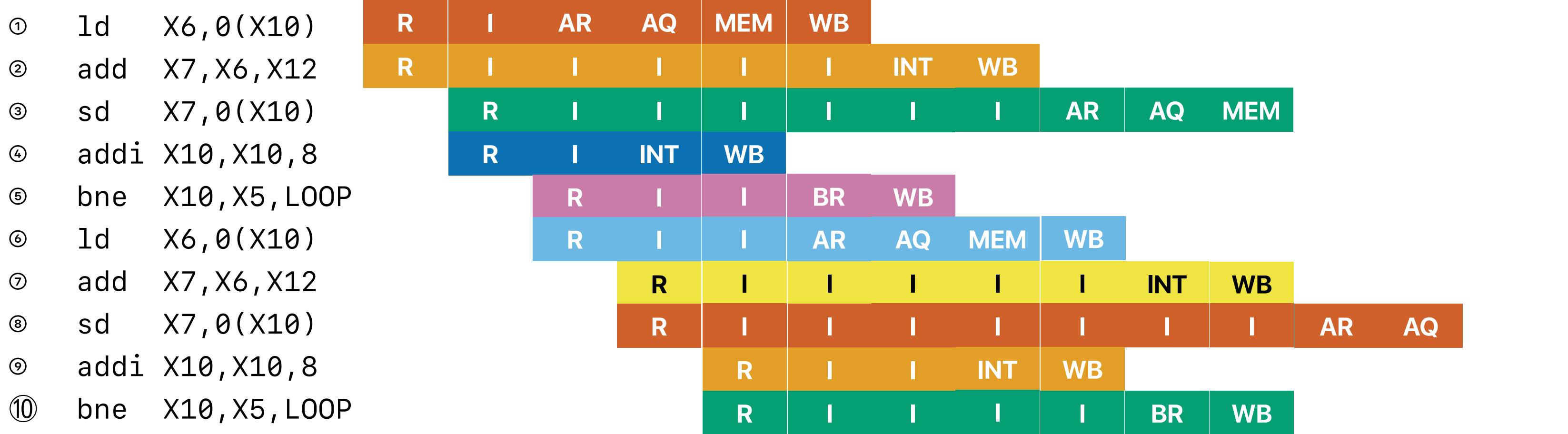
2-issue RR processor in motion



Renamed instruction	Physical Register	Valid Value In use			Valid Value In use		
		P1	P2	P3	P4	P5	P6
1 ld P1, 0(X10)	X5						
2 add P2, P1, X12	X6	P1					
3 sd P2, 0(X10)	X7	P5					
4 addi P3, X10, 8	X10	P3					
5 bne P3, X5, LOOP							
6 ld P4, 0(P3)							
7 add P5, P1, X12							
8 sd P5, 0(P3)							
9 addi P6, P3, 8							
10 bne P6, 0(X10)							

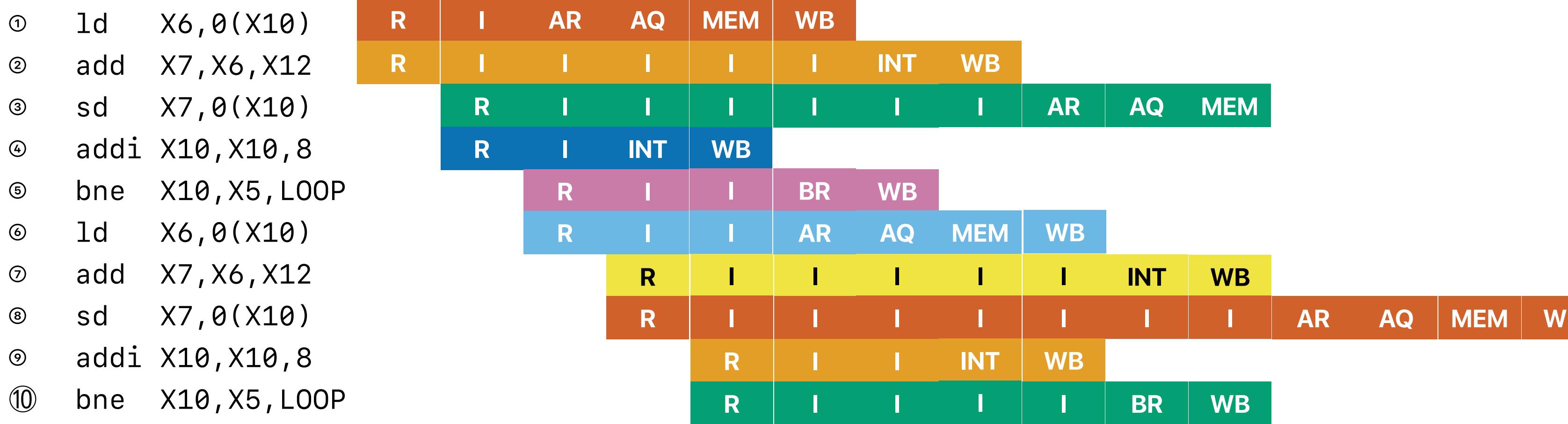
2-issue RR processor in motion

2-issue RR processor in motion



Renamed instruction	Physical Register	Valid			Valid		
		Value	In use		Value	In use	
1 1d P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	1	1	P9
5 bne P3, X5, LOOP	X12			P5	1	1	P10
6 1d P4, 0(P3)							
7 add P5, P1, X12							
8 sd P5, 0(P3)							
9 addi P6, P3, 8							
10 bne P6, 0(X10)							

2-issue RR processor in motion



Renamed instruction		Physical Register		Valid Value In use			Valid Value In use		
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	1	1	P9		
5	bne P3, X5, LOOP	X12		P5	1	1	P10		
6	ld P4, 0(P3)								
7	add P5, P1, X12								
8	sd P5, 0(P3)								
9	addi P6, P3, 8								
10	bne P6, 0(X10)								

2-issue RR processor in motion

Revisit the swap

- For the following RISC-V implementation of the swap function using XOR, how many cycles it takes the processor to issue all instructions? Assume the current PC is already at the first instruction. This processor fetches 2 instruction per cycle, with exactly the same register renaming hardware and pipeline as we showed previously.

① 1d X6, 0(X10)
② 1d X7, 0(X11)
③ add X8, X6, X0
④ add X6, X7, X0
⑤ add X7, X8, X0
⑥ sd X6, 0(X10)
⑦ sd X7, 0(X11)

- A. 5
- B. 7
- C. 9
- D. 11
- E. 13

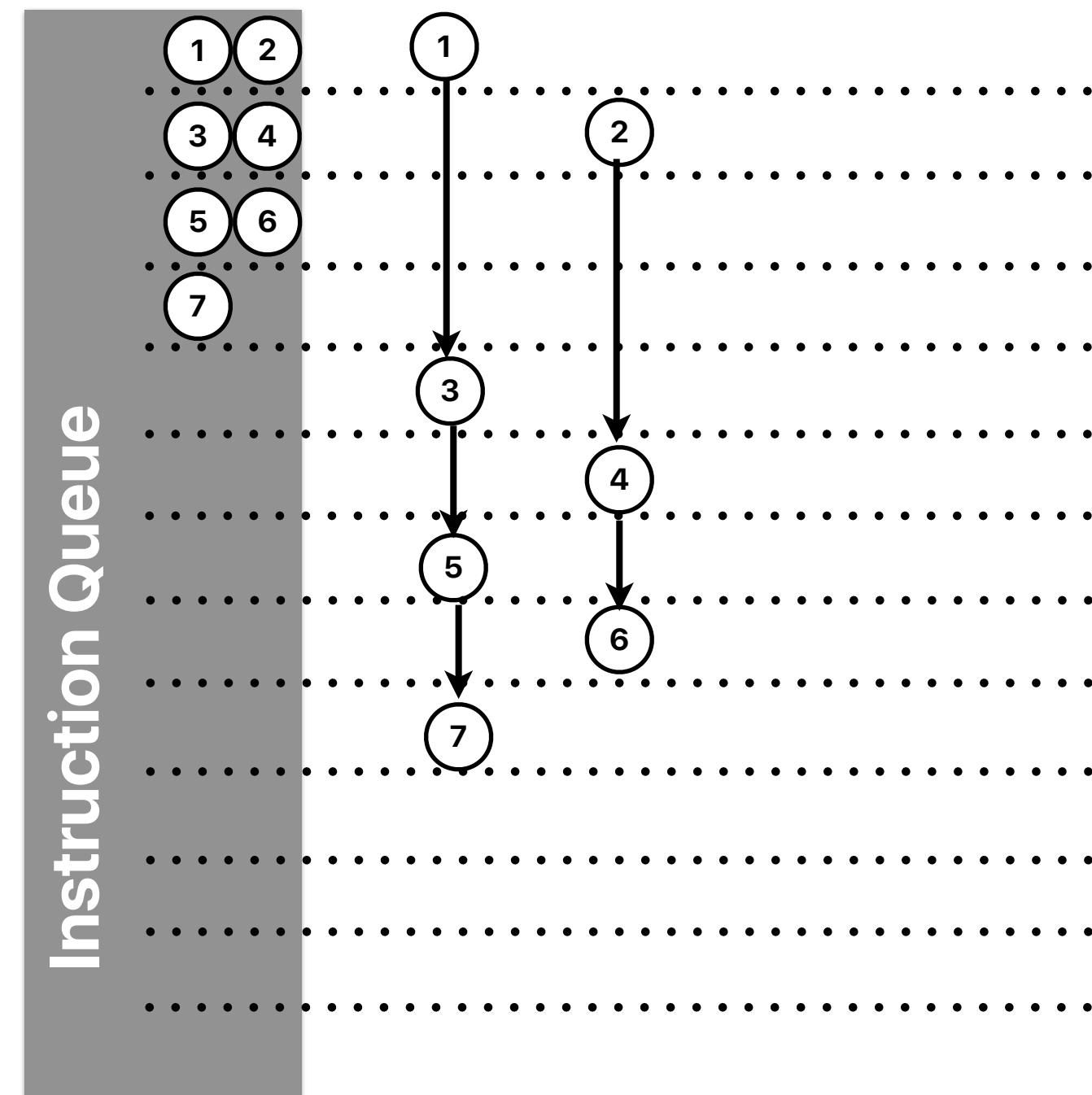


Revisit the swap

- For the following RISC-V implementation of the swap function using XOR, how many cycles it takes the processor to issue all instructions? Assume the current PC is already at the first instruction. This processor fetches 2 instruction per cycle, with exactly the same register renaming hardware and pipeline as we showed previously.

①	ld	X6, 0(X10)
②	ld	X7, 0(X11)
③	add	X8, X6, X0
④	add	X6, X7, X0
⑤	add	X7, X8, X0
⑥	sd	X6, 0(X10)
⑦	sd	X7, 0(X11)

- A. 5
- B. 7
- C. 9
- D. 11
- E. 13

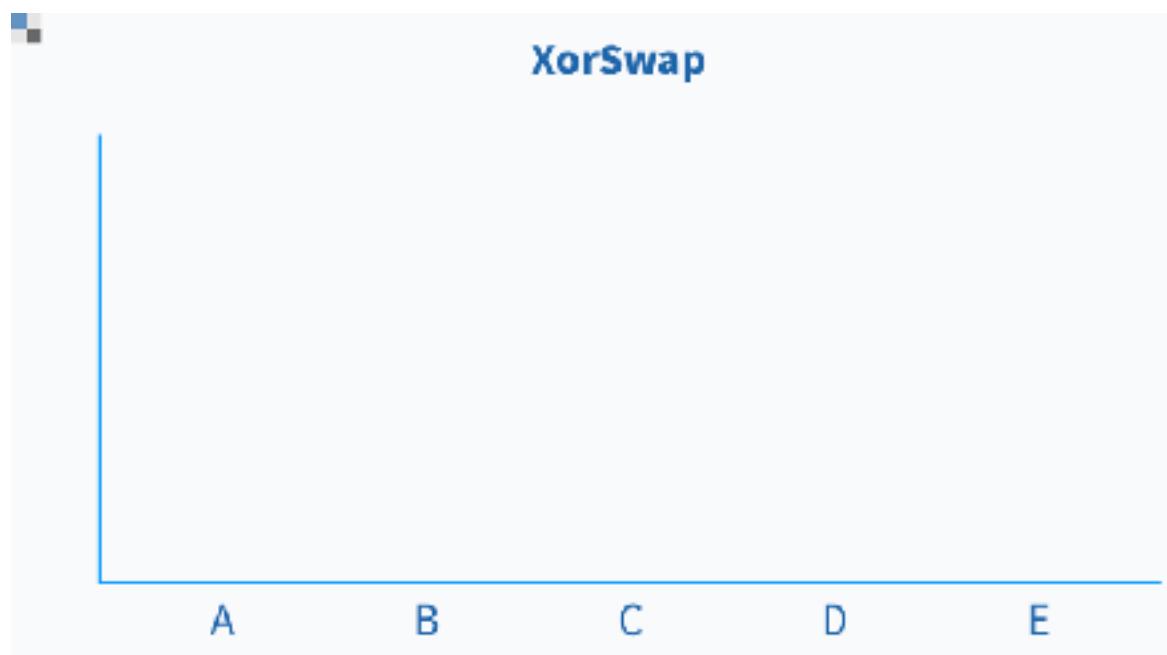


Revisit the XOR swap

- For the following RISC-V implementation of the swap function using XOR, how many cycles it takes the processor to issue all instructions? Assume the current PC is already at the first instruction. This processor fetches 2 instruction per cycle, with exactly the same register renaming hardware and pipeline as we showed previously.

```
① ld      X6, 0(X10)
② ld      X7, 0(X11)
③ xor    X6, X6, X7
④ xor    X7, X7, X6
⑤ xor    X6, X6, X7
⑥ sd      X6, 0(X10)
⑦ sd      X7, 0(X11)
```

- A. 4
- B. 6
- C. 8
- D. 10
- E. 12

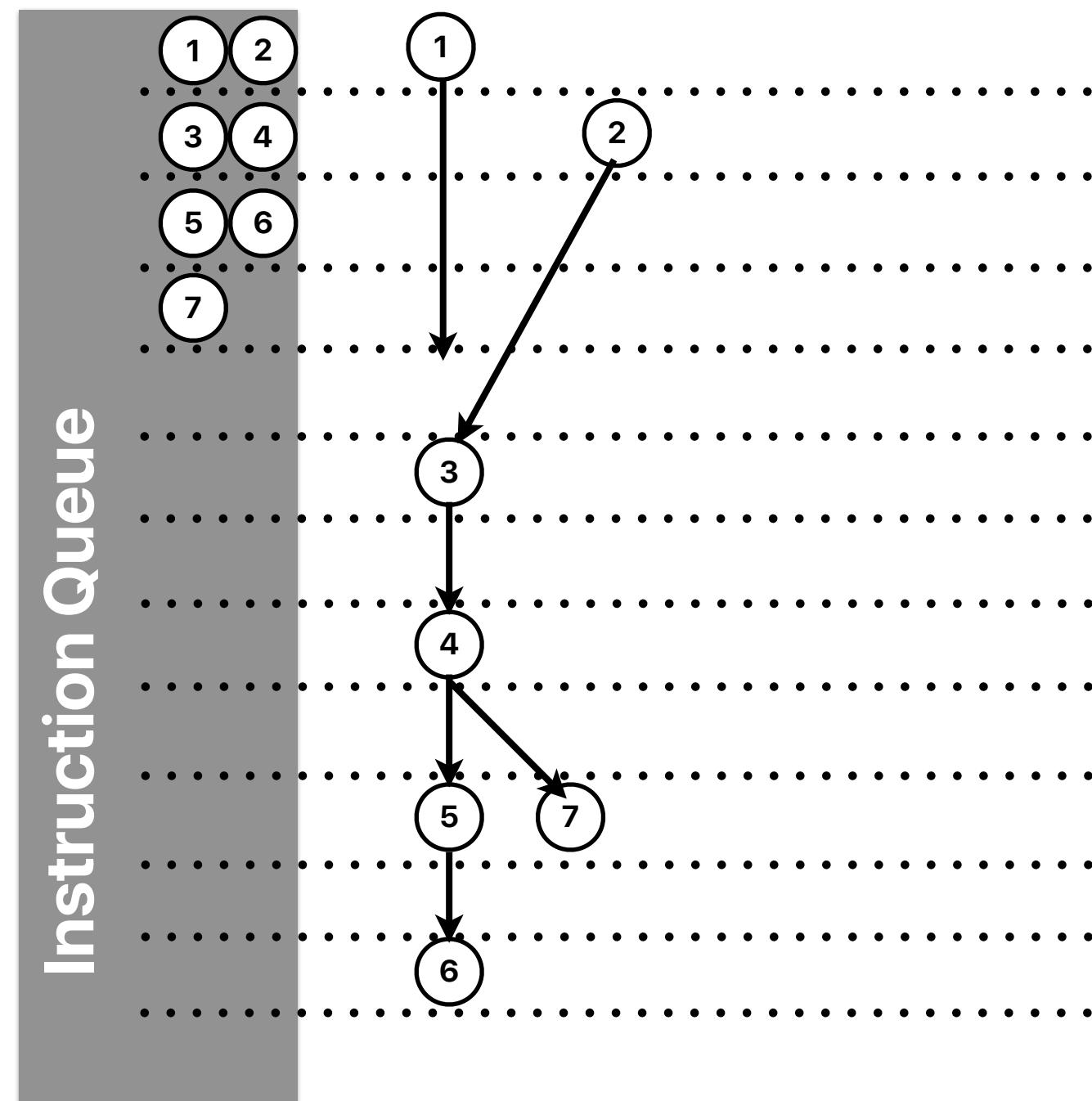


Revisit the XOR swap

- For the following RISC-V implementation of the swap function using XOR, how many cycles it takes the processor to issue all instructions? Assume the current PC is already at the first instruction. This processor fetches 2 instruction per cycle, with exactly the same register renaming hardware and pipeline as we showed previously.

①	ld	X6, 0(X10)
②	ld	X7, 0(X11)
③	xor	X6, X6, X7
④	xor	X7, X7, X6
⑤	xor	X6, X6, X7
⑥	sd	X6, 0(X10)
⑦	sd	X7, 0(X11)

- A. 4
- B. 6
- C. 8
- D. 10
- E. 12



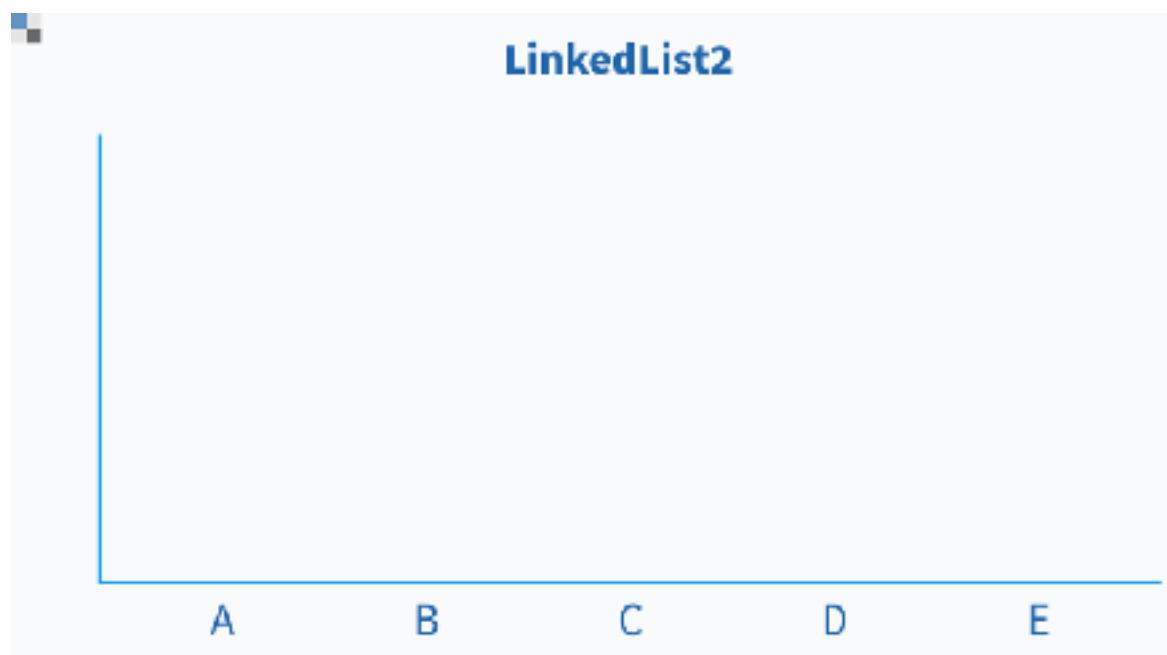
What about “linked list”

- For the following C code and its translation in RISC-V, how many cycles it takes the processor to issue all instructions? Assume the current PC is already at the first instruction and this linked list has only three nodes. This processor can fetch 2 instruction per cycle, with exactly the same register renaming hardware and pipeline as we showed previously.

```
do {  
    number_of_nodes++;  
    current = current->next;  
} while ( current != NULL )
```

```
LOOP:  ld    X10, 8(X10)  
        addi  X7,  X7,  1  
        bne   X10, X0,  LOOP
```

- A. 9
- B. 10
- C. 11
- D. 12
- E. 13



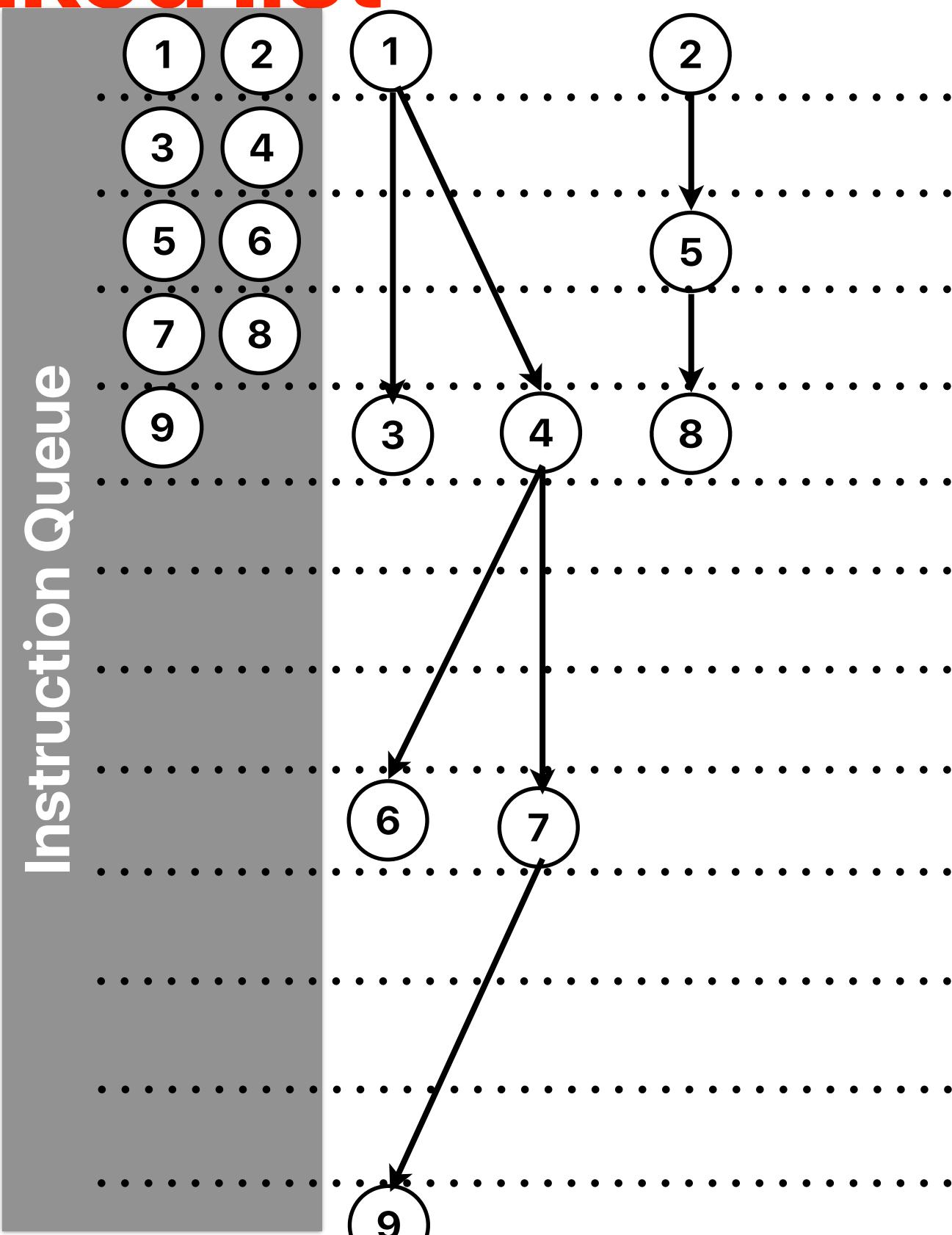
What about “linked list”

Static instructions

```
LOOP: ld X10, 8(X10)
      addi X7, X7, 1
      bne X10, X0, LOOP
```

Dynamic instructions

- ① ld X10, 8(X10)
- ② addi X7, X7, 1
- ③ bne X10, X0, LOOP
- ④ ld X10, 8(X10)
- ⑤ addi X7, X7, 1
- ⑥ bne X10, X0, LOOP
- ⑦ ld X10, 8(X10)
- ⑧ addi X7, X7, 1
- ⑨ bne X10, X0, LOOP



What about “linked list”

- For the following C code and its translation in RISC-V, how many cycles it takes the processor to issue all instructions? Assume the current PC is already at the first instruction and this linked list has only three nodes. This processor can fetch 2 instruction per cycle, with exactly the same register renaming hardware and pipeline as we showed previously.

```
do {  
    number_of_nodes++;  
    current = current->next;  
} while ( current != NULL )
```

```
LOOP:  ld    X10, 8(X10)  
        addi  X7,  X7,  1  
        bne   X10, X0,  LOOP
```

- A. 9
- B. 10
- C. 11
- D. 12
- E. 13

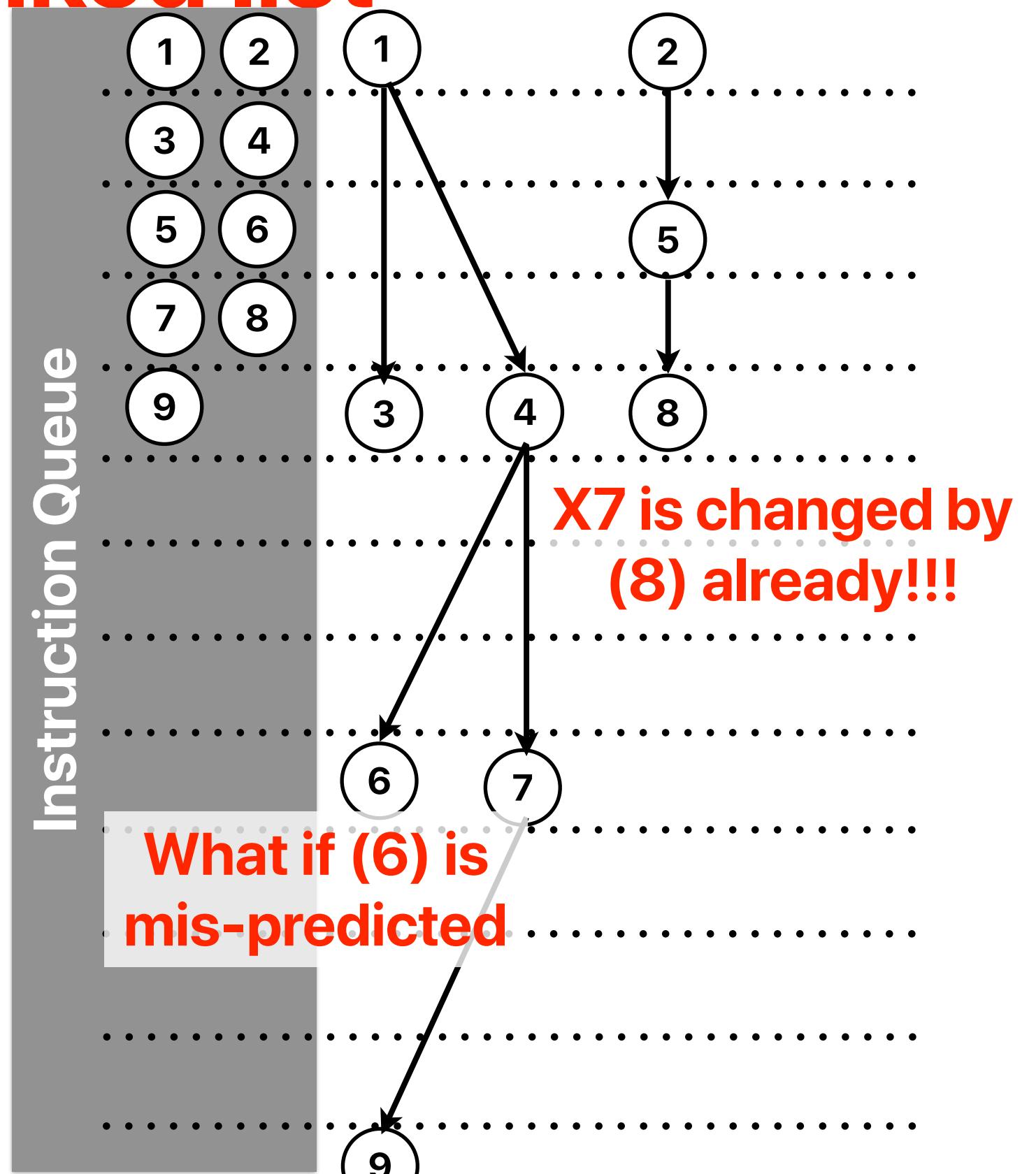
What about “linked list”

Static instructions

```
LOOP: ld    X10, 8(X10)  
      addi  X7, X7, 1  
      bne   X10, X0, LOOP
```

Dynamic instructions

- ① ld X10, 8(X10)
- ② addi X7, X7, 1
- ③ bne X10, X0, LOOP
- ④ ld X10, 8(X10)
- ⑤ addi X7, X7, 1
- ⑥ bne X10, X0, LOOP
- ⑦ ld X10, 8(X10)
- ⑧ addi X7, X7, 1
- ⑨ bne X10, X0, LOOP



In which pipeline stage can we change the next PC?

- How many of the following pipeline stages can we have exceptions?

- ① IF
- ② ID
- ③ EXE
- ④ MEM
- ⑤ WB

A. 1

B. 2

C. 3

D. 4

E. 5



In which pipeline stage can we change the next PC?

- How many of the following pipeline stages can we have exceptions?
 - ① IF — **page fault, illegal address** — jump to **page fault handler**
 - ② ID — **unknown instruction** — jump to **some exception handler**
 - ③ EXE — **divide by zero, overflow, underflow** — jump to **exception handler**
 - ④ MEM — **page fault, illegal address** — jump to **page fault handler**
 - ⑤ WB

A. 1

B. 2

C. 3

D. 4

E. 5

What about “linked list”

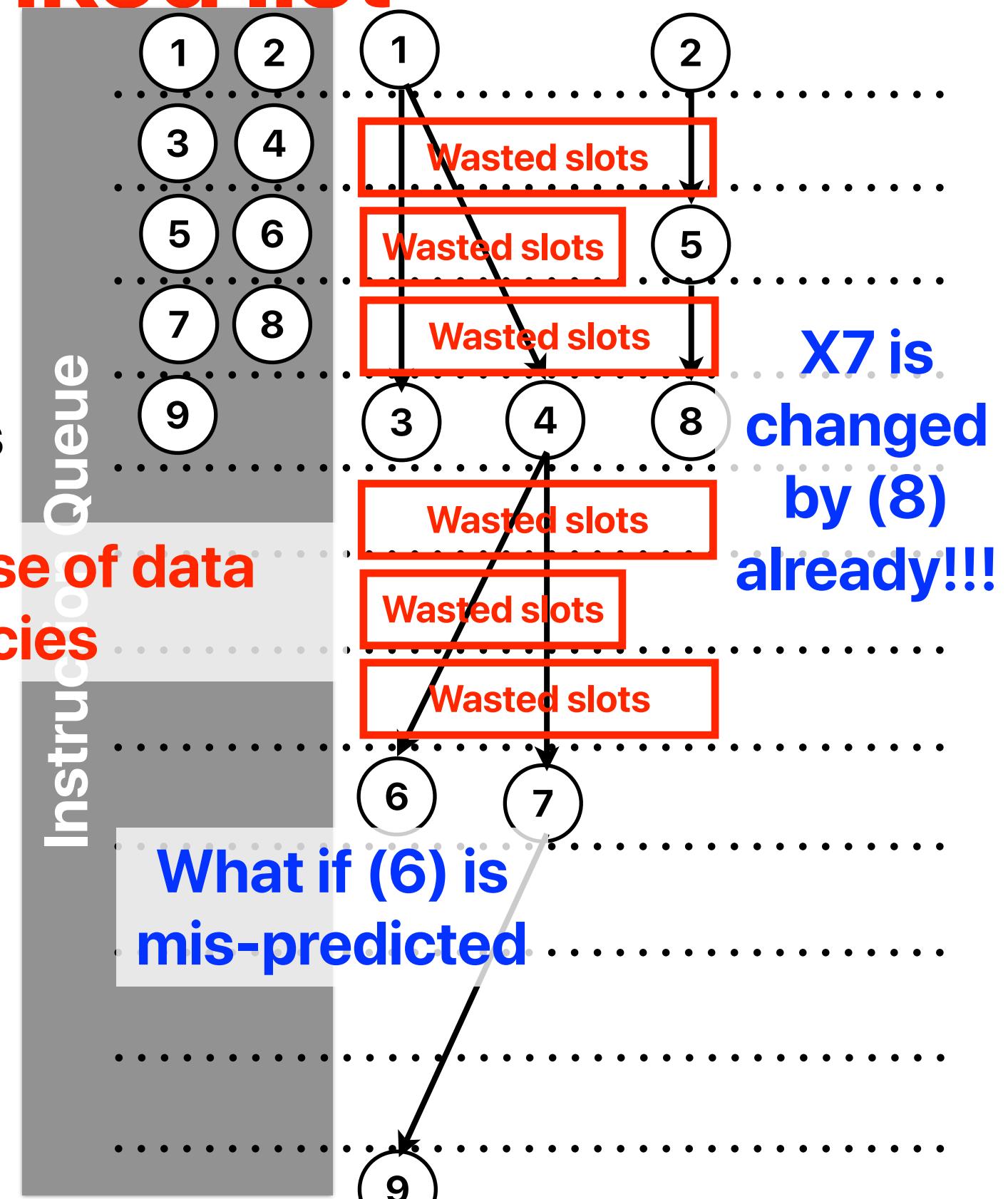
Static instructions

```
LOOP: ld X10, 8(X10)  
      addi X7, X7, 1  
      bne X10, X0, LOOP
```

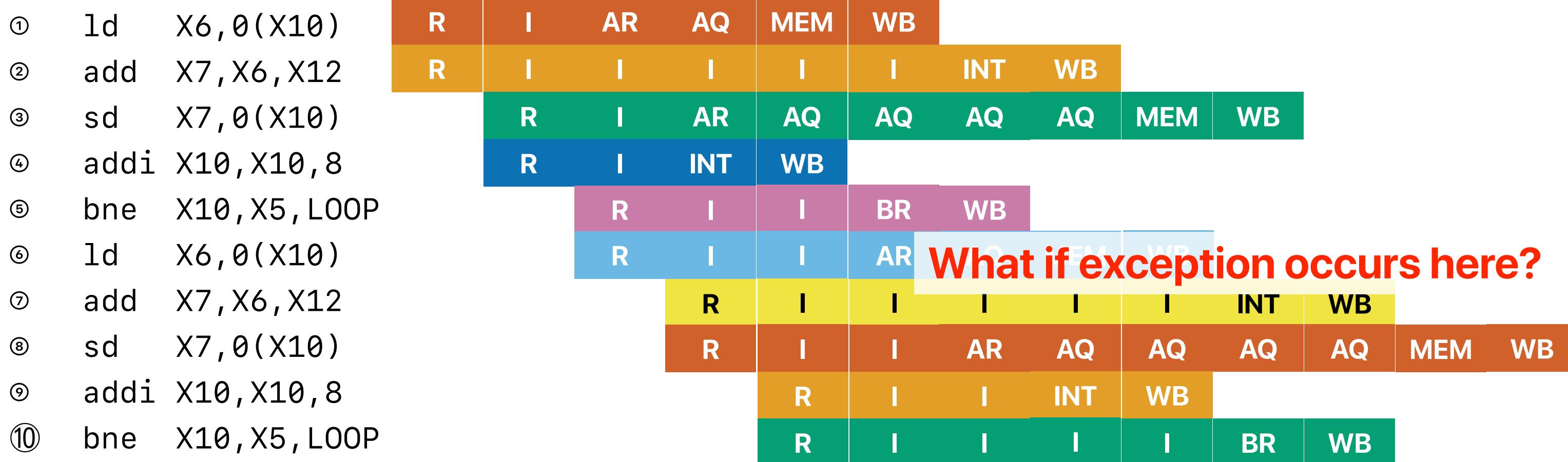
Dynamic instructions

```
① ld X10, 8(X10)  
② addi X7, X7, 1  
③ bne X10, X0, LOOP  
④ ld X10, 8(X10)  
⑤ addi X7, X7, 1  
⑥ bne X10, X0, LOOP  
⑦ ld X10, 8(X10)  
⑧ addi X7, X7, 1  
⑨ bne X10, X0, LOOP
```

ILP is low because of data dependencies



2-issue RR processor in motion



Renamed instruction		Physical Register		Valid Value In use			Valid Value In use		
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	1	1	P9		
5	bne P3, X5, LOOP	X12		P5	1	1	P10		
6	ld P4, 0(P3)								
7	add P5, P1, X12								
8	sd P5, 0(P3)								
9	addi P6, P3, 8								
10	bne P6, 0(X10)								

Speculative Execution

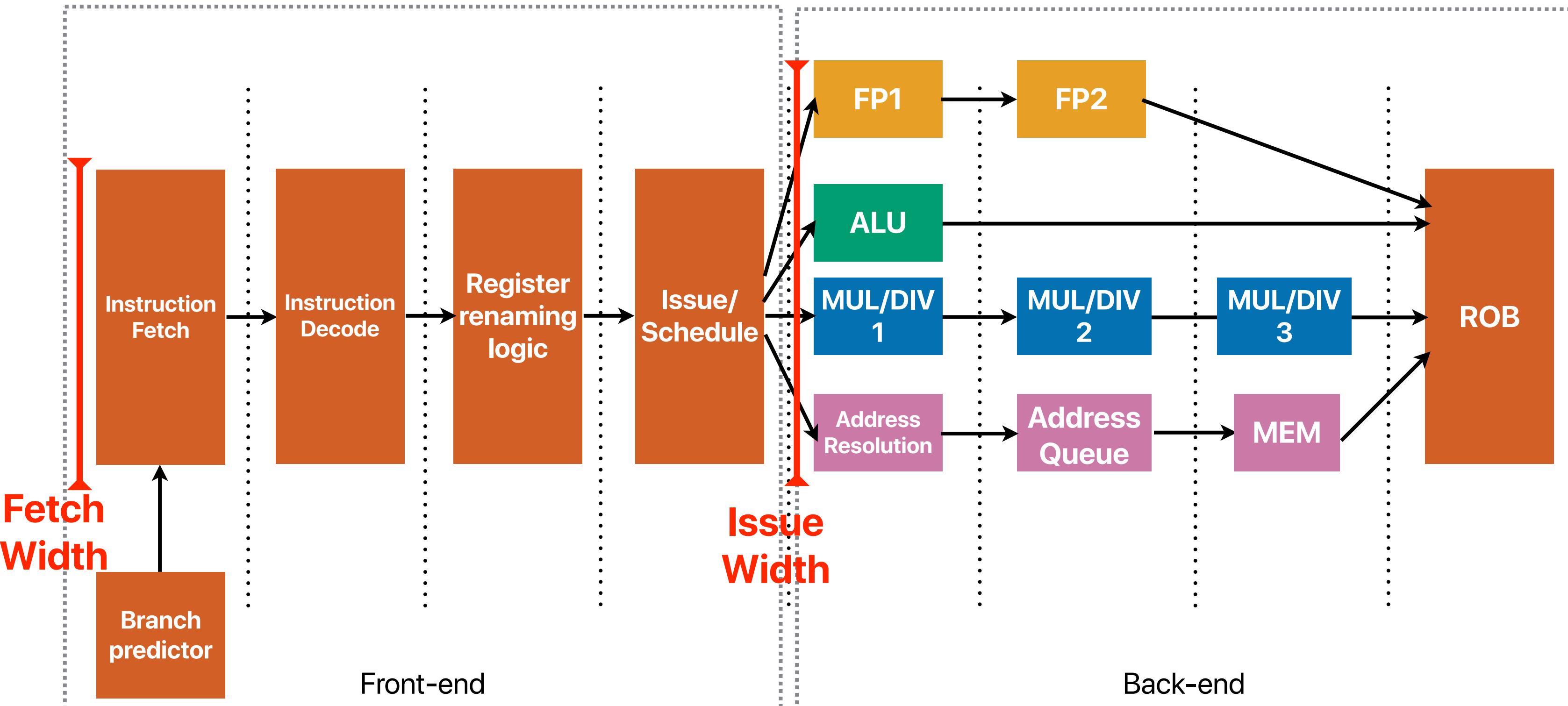
- Any execution of an instruction before a prior instruction finishes is considered as **speculative execution**
- Because it's speculative, we need to preserve the capability to restore to the states before it's executed
 - Branch mis-prediction
 - Exceptions

Reorder Buffer (ROB)

Reorder buffer/Commit stage

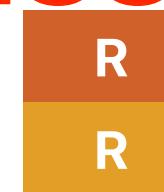
- Reorder buffer — a buffer keep track of the program order of instructions
 - Can be combined with IQ or physical registers — make either as a circular queue
- Commit stage — should the outcome of an instruction be realized
 - An instruction can only leave the pipeline if all it's previous are committed
 - If any prior instruction failed to commit, the instruction should yield it's ROB entry, restore all it's architectural changes

Pipeline SuperScalar/OoO/ROB



2-issue RR processor in motion

- ① ld X6, 0(X10)
- ② add X7, X6, X12
- ③ sd X7, 0(X10)
- ④ addi X10, X10, 8
- ⑤ bne X10, X5, LOOP
- ⑥ ld X6, 0(X10)
- ⑦ add X7, X6, X12
- ⑧ sd X7, 0(X10)
- ⑨ addi X10, X10, 8
- ⑩ bne X10, X5, LOOP

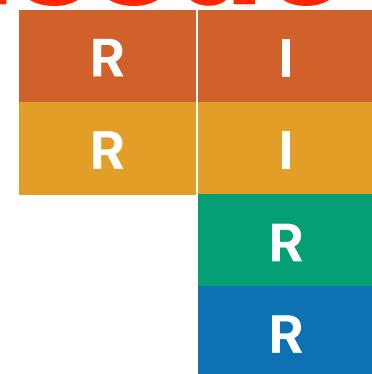


Renamed instruction			Physical Register			Valid Value In use			Valid Value In use		
			X5			P1	0	1	P6		
1	ld P1, 0(X10)		X6	P1		P2	0	1	P7		
2	add P2, P1, X12		X7	P2		P3			P8		
3			X10			P4			P9		
4			X12			P5			P10		
5											
6											
7											
8											
9											
10											

head
tail

2-issue RR processor in motion

- ① ld X6, 0(X10)
- ② add X7, X6, X12
- ③ sd X7, 0(X10)
- ④ addi X10, X10, 8
- ⑤ bne X10, X5, LOOP
- ⑥ ld X6, 0(X10)
- ⑦ add X7, X6, X12
- ⑧ sd X7, 0(X10)
- ⑨ addi X10, X10, 8
- ⑩ bne X10, X5, LOOP

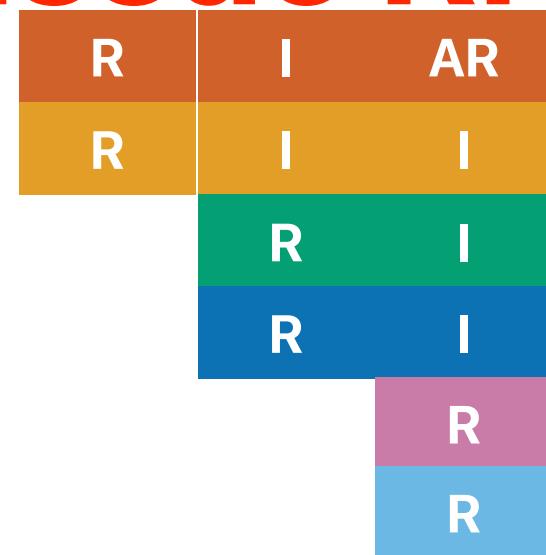


Renamed instruction ← head ← tail

	Renamed instruction	Physical Register	Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5	P1	0	1	P6		
2	add P2, P1, X12	X6	P2	0	1	P7		
3	sd P2, 0(X10)	X7	P3	0	1	P8		
4	addi P3, X10, 8	X10	P4			P9		
5		X12	P5			P10		
6								
7								
8								
9								
10								

2-issue RR processor in motion

- ① ld X6, 0(X10)
- ② add X7, X6, X12
- ③ sd X7, 0(X10)
- ④ addi X10, X10, 8
- ⑤ bne X10, X5, LOOP
- ⑥ ld X6, 0(X10)
- ⑦ add X7, X6, X12
- ⑧ sd X7, 0(X10)
- ⑨ addi X10, X10, 8
- ⑩ bne X10, X5, LOOP



	Renamed instruction	Physical Register						
			Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5	P1	0	1	P6		
2	add P2, P1, X12	X6	P1	0	1	P7		
3	sd P2, 0(X10)	X7	P2	0	1	P8		
4	addi P3, X10, 8	X10	P3	0	1	P9		
5	bne P3, X5, LOOP	X12				P10		
6	ld P4, 0(P3)							
7								
8								
9								
10								

head

tail

2-issue RR processor in motion

			R	I	AR	AQ
①	ld	X6, 0(X10)	R	I	AR	AQ
②	add	X7, X6, X12	R	I	I	I
③	sd	X7, 0(X10)	R	I	I	I
④	addi	X10, X10, 8	R	I	INT	
⑤	bne	X10, X5, LOOP		R	I	
⑥	ld	X6, 0(X10)		R	I	
⑦	add	X7, X6, X12			R	
⑧	sd	X7, 0(X10)			R	
⑨	addi	X10, X10, 8				
⑩	bne	X10, X5, LOOP				

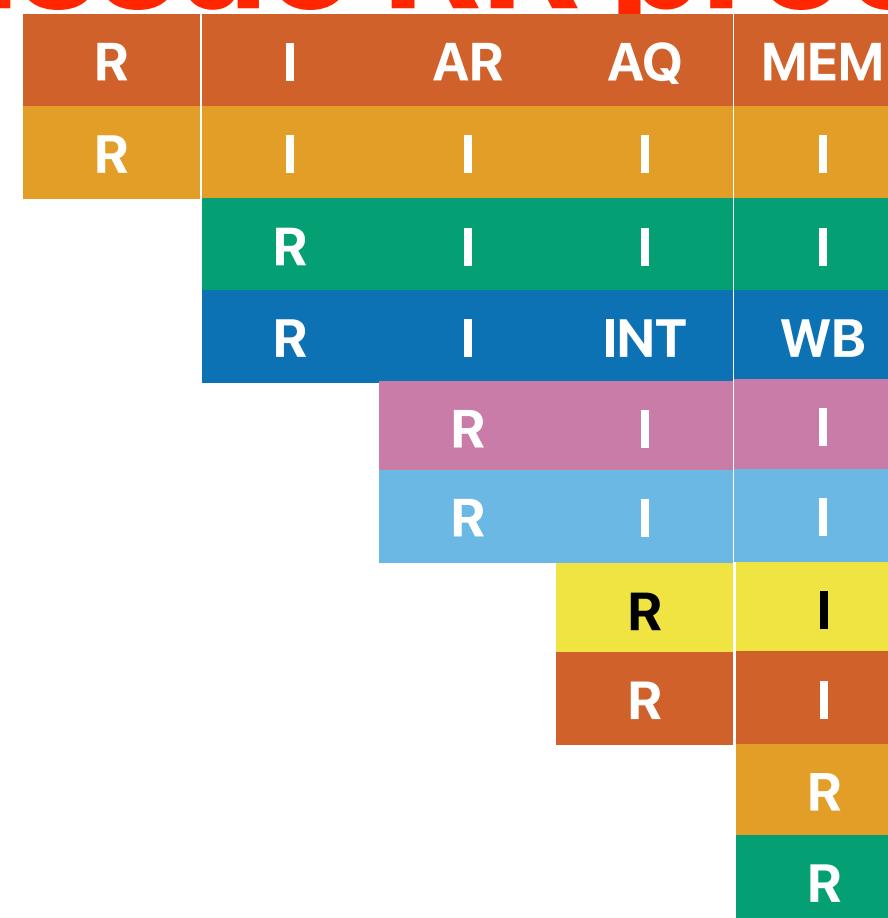
Renamed instruction			Physical Register			Valid Value In use			Valid Value In use		
1	ld	P1, 0(X10)	X5			P1	0	1	P6		
2	add	P2, P1, X12	X6	P1		P2	0	1	P7		
3	sd	P2, 0(X10)	X7	P5		P3	0	1	P8		
4	addi	P3, X10, 8	X10	P3		P4	0	1	P9		
5	bne	P3, X5, LOOP	X12			P5	0	1	P10		
6	ld	P4, 0(P3)									
7	add	P5, P1, X12									
8	sd	P5, 0(P3)									
9											
10											

← head

← tail

2-issue RR processor in motion

- ① ld X6, 0(X10)
- ② add X7, X6, X12
- ③ sd X7, 0(X10)
- ④ addi X10, X10, 8
- ⑤ bne X10, X5, LOOP
- ⑥ ld X6, 0(X10)
- ⑦ add X7, X6, X12
- ⑧ sd X7, 0(X10)
- ⑨ addi X10, X10, 8
- ⑩ bne X10, X5, LOOP



Renamed instruction		
1	ld P1, 0(X10)	
2	add P2, P1, X12	
3	sd P2, 0(X10)	
4	addi P3, X10, 8	
5	bne P3, X5, LOOP	
6	ld P4, 0(P3)	
7	add P5, P1, X12	
8	sd P5, 0(P3)	
9	addi P6, P3, 8	
10	bne P6, 0(X10)	

head

Physical Register	
X5	
X6	P1
X7	P5
X10	P3
X12	

	Valid	Value	In use		Valid	Value	In use
P1	0		1	P6			
P2	0		1	P7			
P3	1		1	P8			
P4	0		1	P9			
P5	0		1	P10			

tail

2-issue RR processor in motion

		R	I	AR	AQ	MEM	WB
①	ld X6, 0(X10)	R	I	AR	AQ	MEM	WB
②	add X7, X6, X12	R	I	I	I	I	I
③	sd X7, 0(X10)	R	I	I	I	I	I
④	addi X10, X10, 8	R	I	INT	WB	C	
⑤	bne X10, X5, LOOP		R	I	I	BR	
⑥	ld X6, 0(X10)		R	I	I	AR	
⑦	add X7, X6, X12		R	I	I		
⑧	sd X7, 0(X10)		R	I	I		
⑨	addi X10, X10, 8		R	I	I		
⑩	bne X10, X5, LOOP		R	I			

	Renamed instruction	Physical Register
1	ld P1, 0(X10)	X5
2	add P2, P1, X12	X6
3	sd P2, 0(X10)	X7
4	addi P3, X10, 8	X10
5	bne P3, X5, LOOP	X12
6	ld P4, 0(P3)	
7	add P5, P1, X12	
8	sd P5, 0(P3)	
9	addi P6, P3, 8	
10	bne P6, 0(X10)	

head

tail

	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6			
P2	0		1	P7			
P3	1		1	P8			
P4	0		1	P9			
P5	0		1	P10			

2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB	C
①	ld	X6, 0(X10)	R	I					
②	add	X7, X6, X12	R	I	I	I	I	I	INT
③	sd	X7, 0(X10)	R	I	I	I	I	I	I
④	addi	X10, X10, 8	R	I	INT	WB	C	C	
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	
⑦	add	X7, X6, X12		R	I	I	I	I	
⑧	sd	X7, 0(X10)		R	I	I	I	I	
⑨	addi	X10, X10, 8		R	I	I	I	I	
⑩	bne	X10, X5, LOOP		R	I	I			

Renamed instruction		Physical Register		Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	0	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	0	1	P9		
5	bne P3, X5, LOOP	X12		P5	0	1	P10		
6	ld P4, 0(P3)								
7	add P5, P1, X12								
8	sd P5, 0(P3)								
9	addi P6, P3, 8								
10	bne P6, 0(X10)								

← head

← tail

2-issue RR processor in motion

		R	I	AR	AQ	MEM	WB	C
①	ld X6, 0(X10)	R	I	AR	AQ	MEM	WB	C
②	add X7, X6, X12	R	I	I	I	I	INT	WB
③	sd X7, 0(X10)	R	I	I	I	I	I	I
④	addi X10, X10, 8	R	I	INT	WB	C	C	C
⑤	bne X10, X5, LOOP		R	I	I	BR	WB	C
⑥	ld X6, 0(X10)		R	I	I	AR	AQ	MEM
⑦	add X7, X6, X12		R	I	I	I	I	I
⑧	sd X7, 0(X10)		R	I	I	I	I	I
⑨	addi X10, X10, 8		R	I	I	I	INT	
⑩	bne X10, X5, LOOP		R	I	I	I		

Renamed instruction		Physical Register		Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	0	1	P9		
5	bne P3, X5, LOOP	X12		P5	0	1	P10		
6	ld P4, 0(P3)								
7	add P5, P1, X12								
8	sd P5, 0(P3)								
9	addi P6, P3, 8								
10	bne P6, 0(X10)								

← head

← tail

2-issue RR processor in motion

		R	I	AR	AQ	MEM	WB	C	
①	ld X6, 0(X10)	R	I	AR	AQ	MEM	WB	C	
②	add X7, X6, X12	R	I	I	I	I	I	INT	WB
③	sd X7, 0(X10)	R	I	I	I	I	I	I	AR
④	addi X10, X10, 8	R	I	INT	WB	C	C	C	C
⑤	bne X10, X5, LOOP		R	I	I	BR	WB	C	C
⑥	ld X6, 0(X10)		R	I	I	AR	AQ	MEM	WB
⑦	add X7, X6, X12		R	I	I	I	I	I	I
⑧	sd X7, 0(X10)		R	I	I	I	I	I	I
⑨	addi X10, X10, 8		R	I	I	I	INT	WB	
⑩	bne X10, X5, LOOP		R	I	I	I	I	I	

Renamed instruction		Physical Register		Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	1	1	P9		
5	bne P3, X5, LOOP	X12		P5	0	1	P10		

6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

head

tail

2-issue RR processor in motion

		R	I	AR	AQ	MEM	WB	C	
①	ld X6, 0(X10)	R	I	AR	AQ	MEM	WB	C	
②	add X7, X6, X12	R	I	I	I	I	INT	WB	C
③	sd X7, 0(X10)	R	I	I	I	I	I	AR	AQ
④	addi X10, X10, 8	R	I	INT	WB	C	C	C	C
⑤	bne X10, X5, LOOP		R	I	I	BR	WB	C	C
⑥	ld X6, 0(X10)		R	I	I	AR	AQ	MEM	WB
⑦	add X7, X6, X12			R	I	I	I	I	INT
⑧	sd X7, 0(X10)			R	I	I	I	I	I
⑨	addi X10, X10, 8			R	I	I	INT	WB	C
⑩	bne X10, X5, LOOP			R	I	I	I	I	BR

Renamed instruction		Physical Register		Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	1	1	P9		
5	bne P3, X5, LOOP	X12		P5	0	1	P10		
6	ld P4, 0(P3)								
7	add P5, P1, X12								
8	sd P5, 0(P3)								
9	addi P6, P3, 8								
10	bne P6, 0(X10)								

← head

← tail

2-issue RR processor in motion

		R	I	AR	AQ	MEM	WB	C			
①	ld	X6, 0(X10)	R								
②	add	X7, X6, X12	R	I	I	I	I	INT	WB	C	
③	sd	X7, 0(X10)	R	I	I	I	I	I	AR	AQ	MEM
④	addi	X10, X10, 8	R	I	INT	WB	C	C	C	C	C
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	C	C	C
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	MEM	WB	C
⑦	add	X7, X6, X12			R	I	I	I	I	INT	WB
⑧	sd	X7, 0(X10)			R	I	I	I	I	I	I
⑨	addi	X10, X10, 8			R	I	I	INT	WB	C	C
⑩	bne	X10, X5, LOOP			R	I	I	I	I	BR	WB

Renamed instruction	Physical Register	Valid			Valid	Value	In use
		Value	In use	Valid			
1 ld P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	1	1	P9
5 bne P3, X5, LOOP	X12			P5	1	1	P10
6 ld P4, 0(P3)							
7 add P5, P1, X12							
8 sd P5, 0(P3)							
9 addi P6, P3, 8							
10 bne P6, 0(X10)							

← head

← tail

2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB	C				
①	ld	X6, 0(X10)	R	I	AR	AQ	MEM	WB	C				
②	add	X7, X6, X12	R	I	I	I	I	I	INT	WB	C		
③	sd	X7, 0(X10)	R	I	I	I	I	I	I	AR	AQ	MEM	C
④	addi	X10, X10, 8	R	I	INT	WB	C	C	C	C	C	C	C
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	C	C	C	C	C
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	MEM	WB	C	C	C
⑦	add	X7, X6, X12		R	I	I	I	I	I	INT	WB	C	
⑧	sd	X7, 0(X10)		R	I	I	I	I	I	I	I	AR	
⑨	addi	X10, X10, 8		R	I	I	INT	WB	C	C	C	C	
⑩	bne	X10, X5, LOOP		R	I	I	I	I	I	BR	WB	C	

Renamed instruction		Physical Register		Valid Value In use			Valid Value In use		
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	1	1	P9		
5	bne P3, X5, LOOP	X12		P5	1	1	P10		

head ← 8
tail ← 10

2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB	C					
①	ld	X6, 0(X10)	R	I	AR	AQ	MEM	WB	C					
②	add	X7, X6, X12	R	I	I	I	I	I	INT	WB	C			
③	sd	X7, 0(X10)	R	I	I	I	I	I	I	AR	AQ	MEM	C	
④	addi	X10, X10, 8	R	I	INT	WB	C	C	C	C	C	C	C	C
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	C	C	C	C	C	C
⑥	ld	X6, 0(X10)		R	I	I	AR	AQ	MEM	WB	C	C	C	C
⑦	add	X7, X6, X12		R	I	I	I	I	I	INT	WB	C		
⑧	sd	X7, 0(X10)		R	I	I	I	I	I	I	I	AR	AQ	
⑨	addi	X10, X10, 8		R	I	I	INT	WB	C	C	C	C	C	
⑩	bne	X10, X5, LOOP		R	I	I	I	I	I	BR	WB	C	C	

Renamed instruction		Physical Register		Valid Value In use			Valid Value In use		
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	1	1	P9		
5	bne P3, X5, LOOP	X12		P5	1	1	P10		

head ← tail

2-issue RR processor in motion

①	ld	X6, 0(X10)	R	I	AR	AQ	MEM	WB	C		
②	add	X7, X6, X12	R	I	I	I	I	I	INT	WB	C
③	sd	X7, 0(X10)	R	I	I	I	I	I	I	AR	AQ MEM C
④	addi	X10, X10, 8	R	I	INT	WB	C	C	C	C	C
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	C	C	C
⑥	ld	X6, 0(X10)	R	I	I	AR	AQ	MEM	WB	C	C
⑦	add	X7, X6, X12		R	I	I	I	I	I	INT	WB
⑧	sd	X7, 0(X10)	R	I	I	I	I	I	I	I	AR AQ MEM
⑨	addi	X10, X10, 8	R	I	I	INT	WB	C	C	C	C
⑩	bne	X10, X5, LOOP	R	I	I	I	I	I	BR	WB	C

Renamed instruction			Physical Register			Valid Value In use			Valid Value In use								
			X5	X6	X7	X10	X12	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	ld P1, 0(x10)																
2	add P2, P1, x12		X5														
3	sd P2, 0(x10)			X6	P1												
4	addi P3, x10, 8				X7	P5											
5	bne P3, x5, LOOP					X10	P3										
6	ld P4, 0(p3)					X12											
7	add P5, P1, x12																
8	sd P5, 0(P3)																
9	addi P6, P3, 8																
10	bne P6, 0(x10)																

head ← tail

2-issue RR processor in motion

Renamed instruction			Physical Register			Valid Value In use			Valid Value In use		
			X5			P1	1	1	P6		
1	ld	P1, 0(x10)	X6	P1		P2	1	1	P7		
2	add	P2, P1, X12	X7	P5		P3	1	1	P8		
3	sd	P2, 0(x10)	X10	P3		P4	1	1	P9		
4	addi	P3, X10, 8	X12			P5	1	1	P10		
5	bne	P3, X5, LOOP									
6	ld	P4, 0(P3)									
7	add	P5, P1, X12									
8	sd	P5, 0(P3)									
9	addi	P6, P3, 8									
10	bne	P6, 0(x10)									

How good is SS/OoO/ROB with this code?

- Consider the following dynamic instructions

- ① 1d X1, 0(X10)
- ② addi X10, X10, 8
- ③ add X20, X20, X1
- ④ bne X10, X2, LOOP
- ⑤ 1d X1, 0(X10)
- ⑥ addi X10, X10, 8
- ⑦ add X20, X20, X1
- ⑧ bne X10, X2, LOOP

Assume a superscalar processor with issue width as 2 & unlimited physical registers that can fetch up to 2 instructions per cycle, 3 cycles to execute a memory instruction how many cycles it takes to issue all instructions?

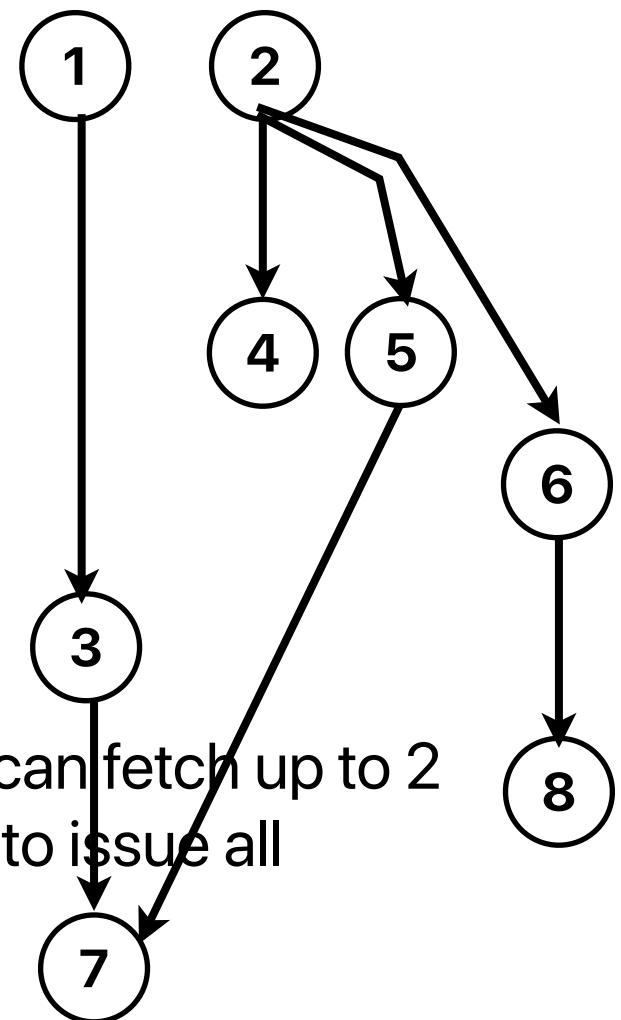
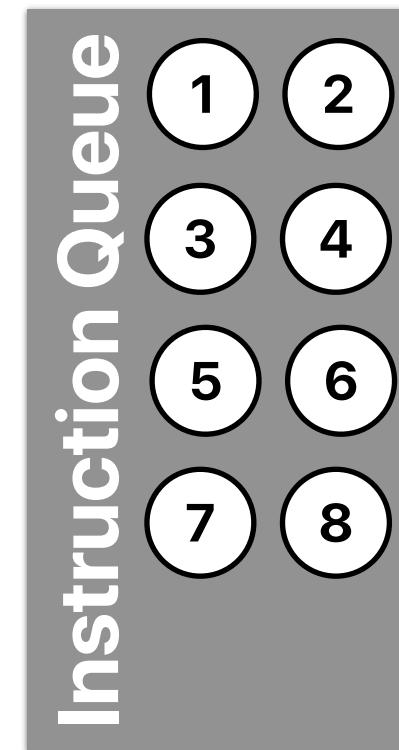
- A. 1
- B. 3
- C. 5
- D. 7
- E. 9



How good is SS/OoO/ROB with this code?

- Consider the following dynamic instructions

- ① 1d X1, 0(X10)
- ② addi X10, X10, 8
- ③ add X20, X20, X1
- ④ bne X10, X2, LOOP
- ⑤ 1d X1, 0(X10)
- ⑥ addi X10, X10, 8
- ⑦ add X20, X20, X1
- ⑧ bne X10, X2, LOOP



Assume a superscalar processor with issue width as 2 & unlimited physical registers that can fetch up to 2 instructions per cycle, 3 cycles to execute a memory instruction how many cycles it takes to issue all instructions?

- A. 1
- B. 3
- C. 5
- D. 7
- E. 9

How good is SS/OoO/ROB with this code?

- Consider the following dynamic instructions

- ① 1d X1, 0(X10)
- ② addi X10, X10, 8
- ③ add X20, X20, X1
- ④ bne X10, X2, LOOP

Assume a superscalar processor with issue width as 2 & unlimited physical registers that can fetch up to 4 instructions per cycle, 3 cycles to execute a memory instruction and the loop will execute for 10,000 times, what's the average CPI?

- A. 0.5
- B. 0.75
- C. 1
- D. 1.25
- E. 1.5



How good is SS/OoO/ROB with s code?

- Consider the following dynamic instructions

- ① ld X1, 0(X10)
- ② addi X10, X10, 8
- ③ add X20, X20, X1
- ④ bne X10, X2, LOOP

Assume a superscalar processor with issue width as 2 & unit that can fetch up to 4 instructions per cycle, 3 cycles to execute and the loop will execute for 10,000 times, what's the average

A. 0.5

B. 0.75

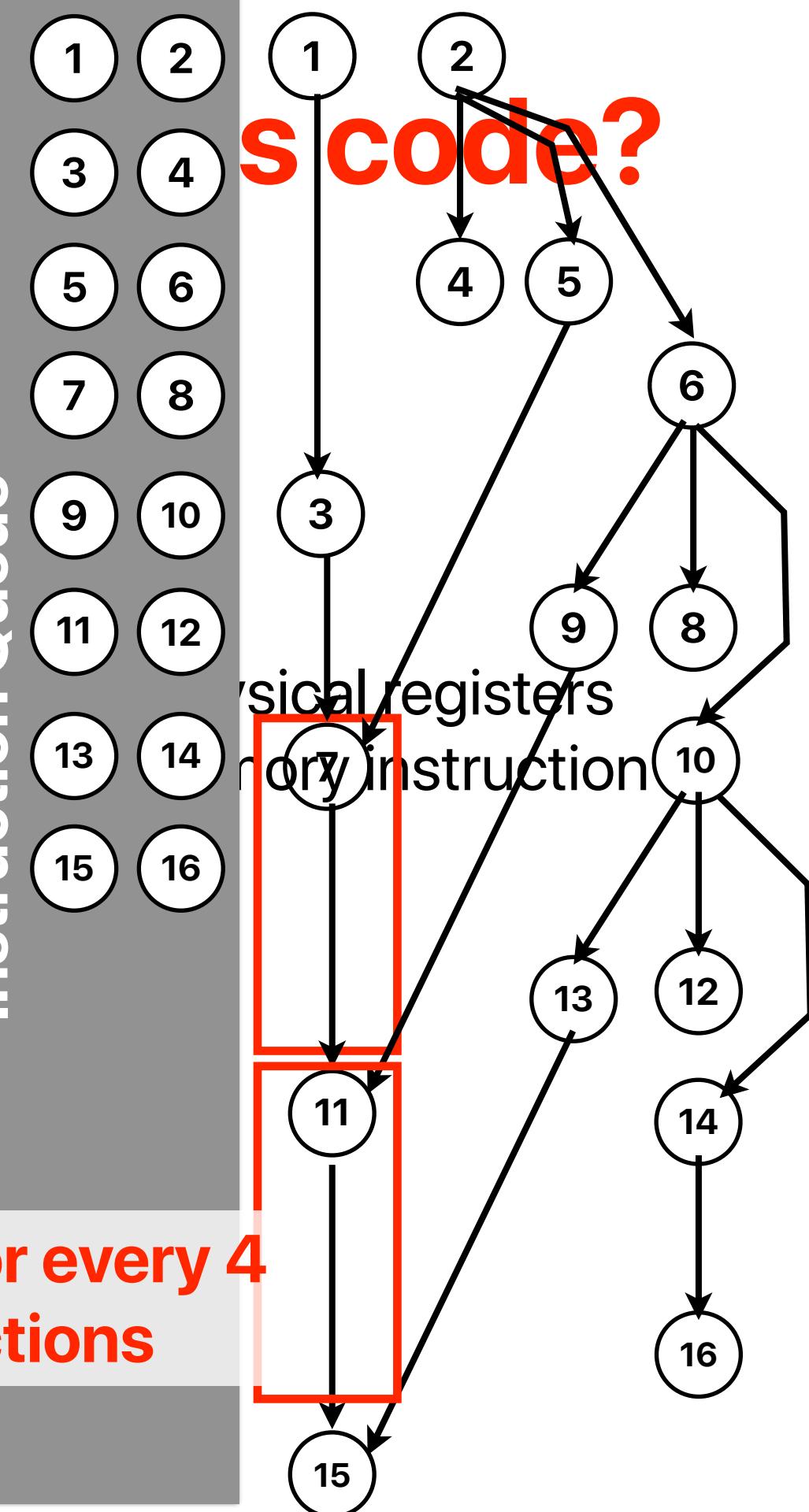
C. 1

D. 1.25

E. 1.5

- ① ld X1, 0(X10)
- ② addi X10, X10, 8
- ③ add X20, X20, X1
- ④ bne X10, X2, LOOP
- ⑤ ld X1, 0(X10)
- ⑥ addi X10, X10, 8
- ⑦ add X20, X20, X1
- ⑧ bne X10, X2, LOOP
- ⑨ ld X1, 0(X10)
- ⑩ addi X10, X10, 8
- ⑪ add X20, X20, X1
- ⑫ bne X10, X2, LOOP

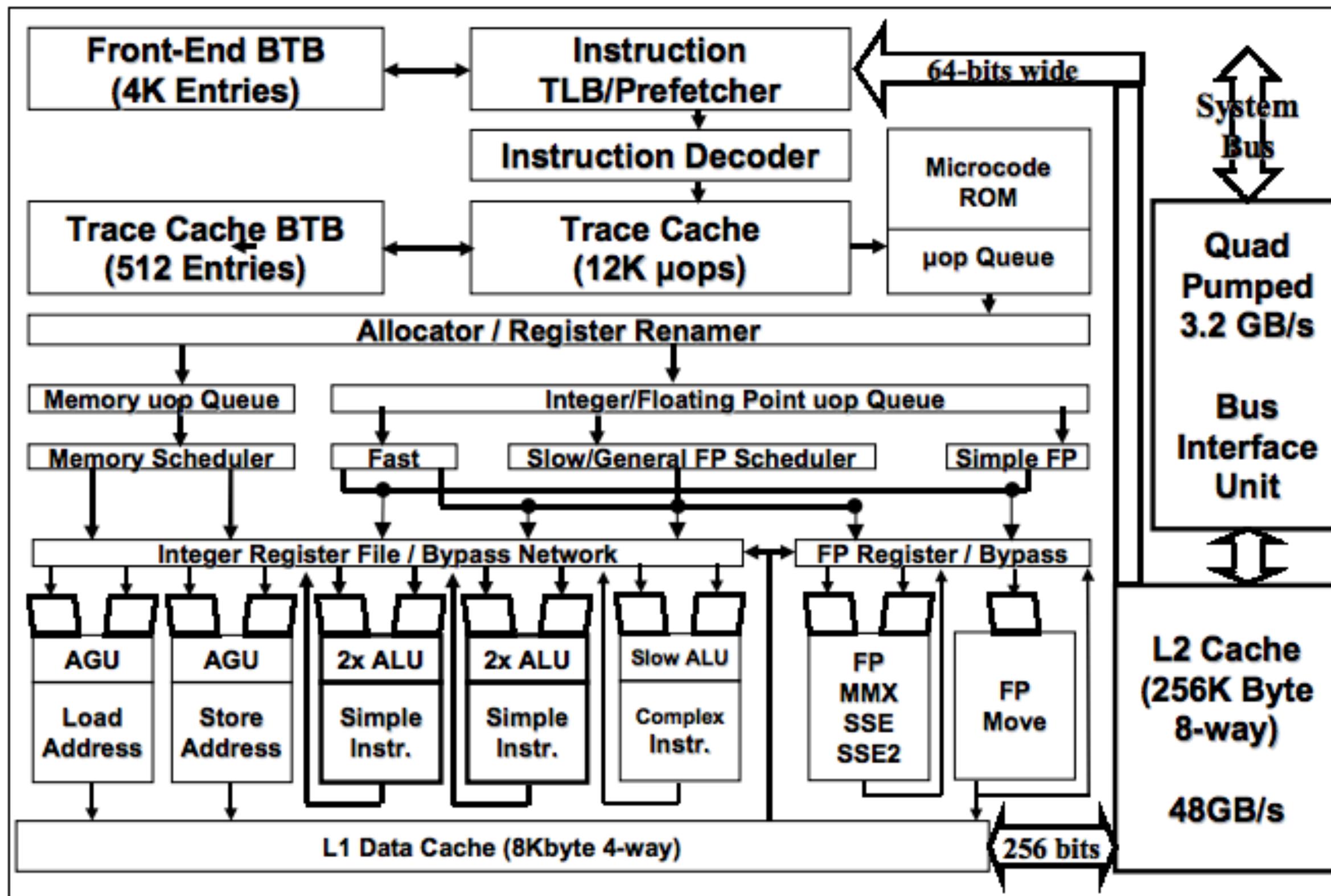
Instruction Queue



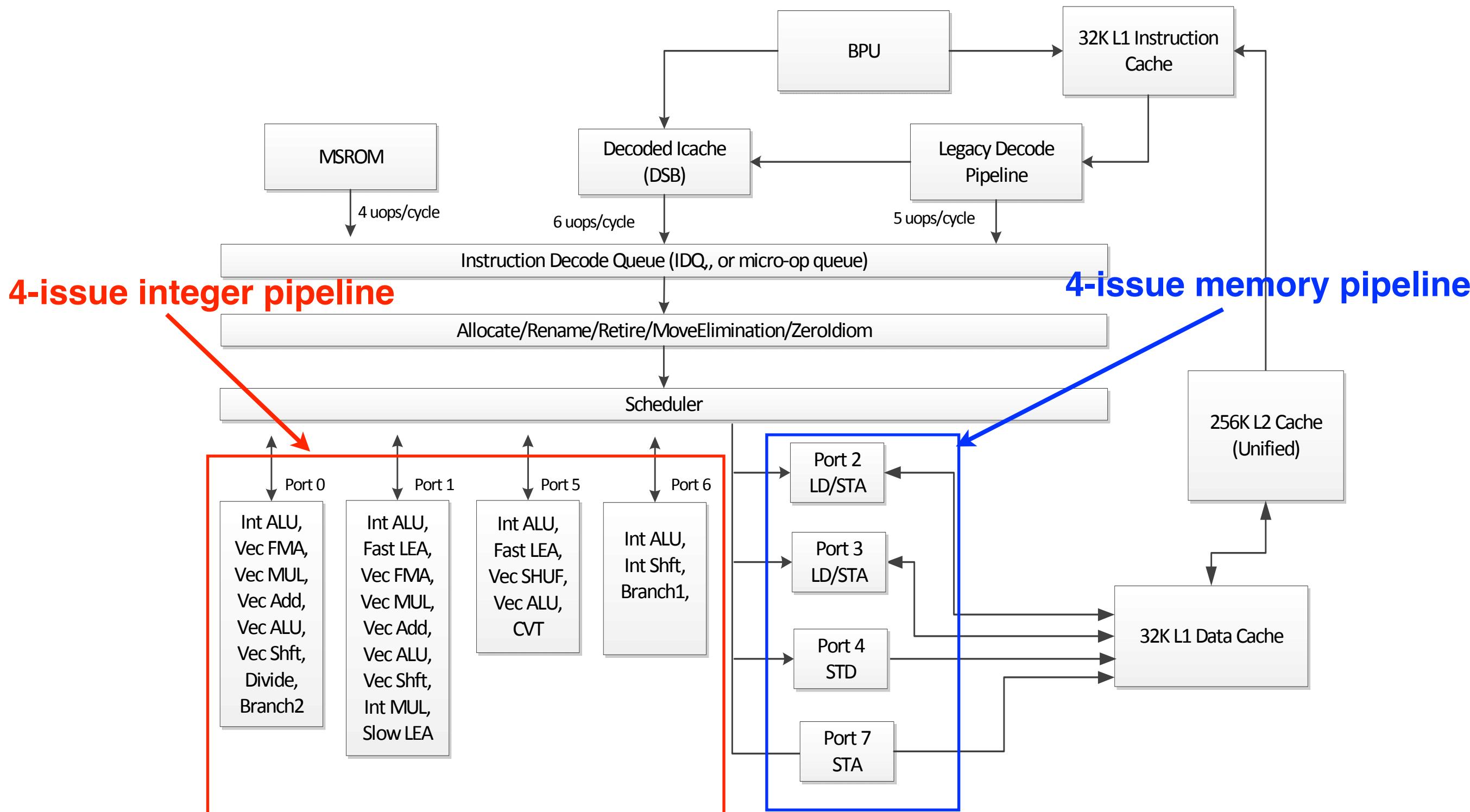
3 cycles for every 4 instructions

The pipelines of Modern Processors

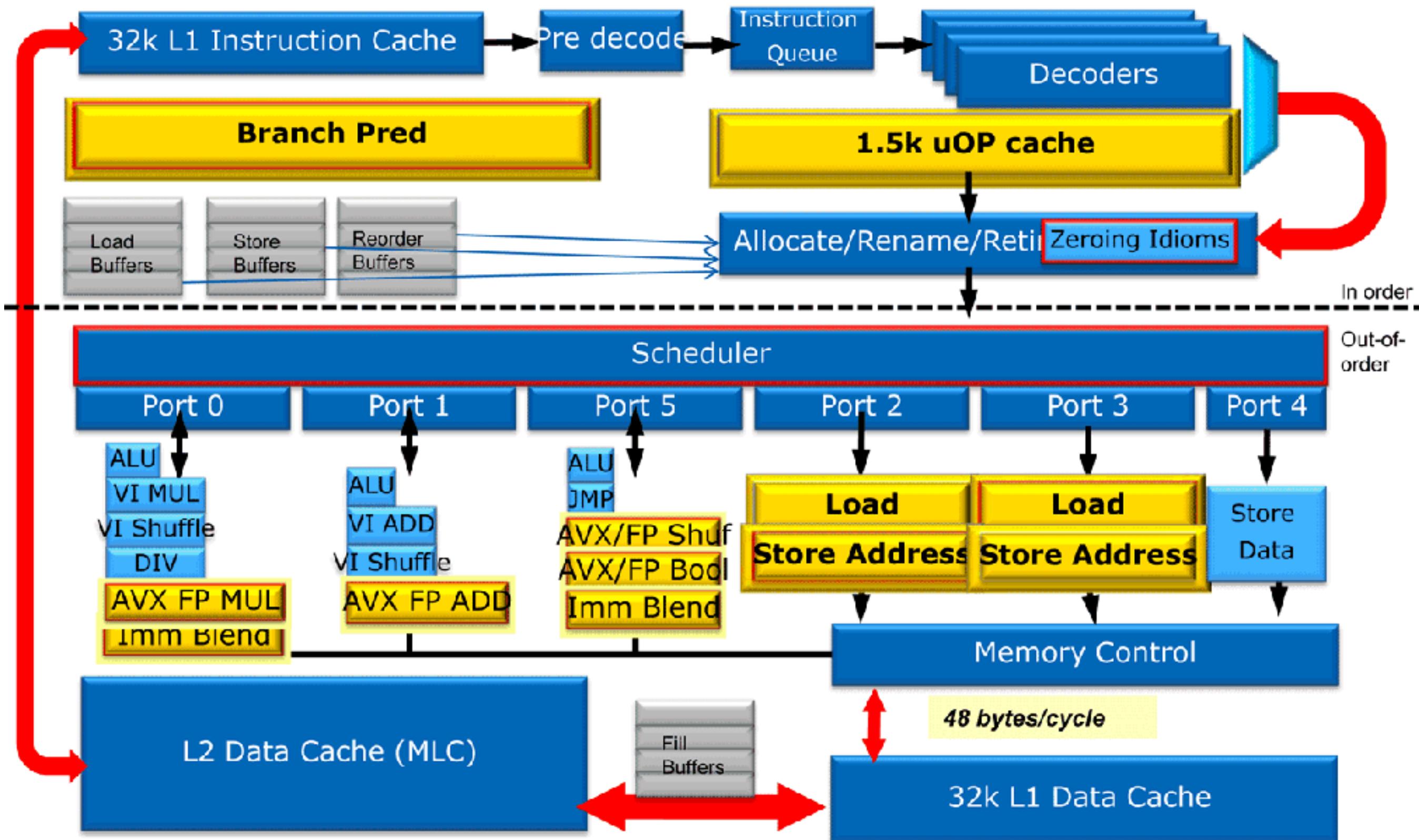
Intel Pentium 4

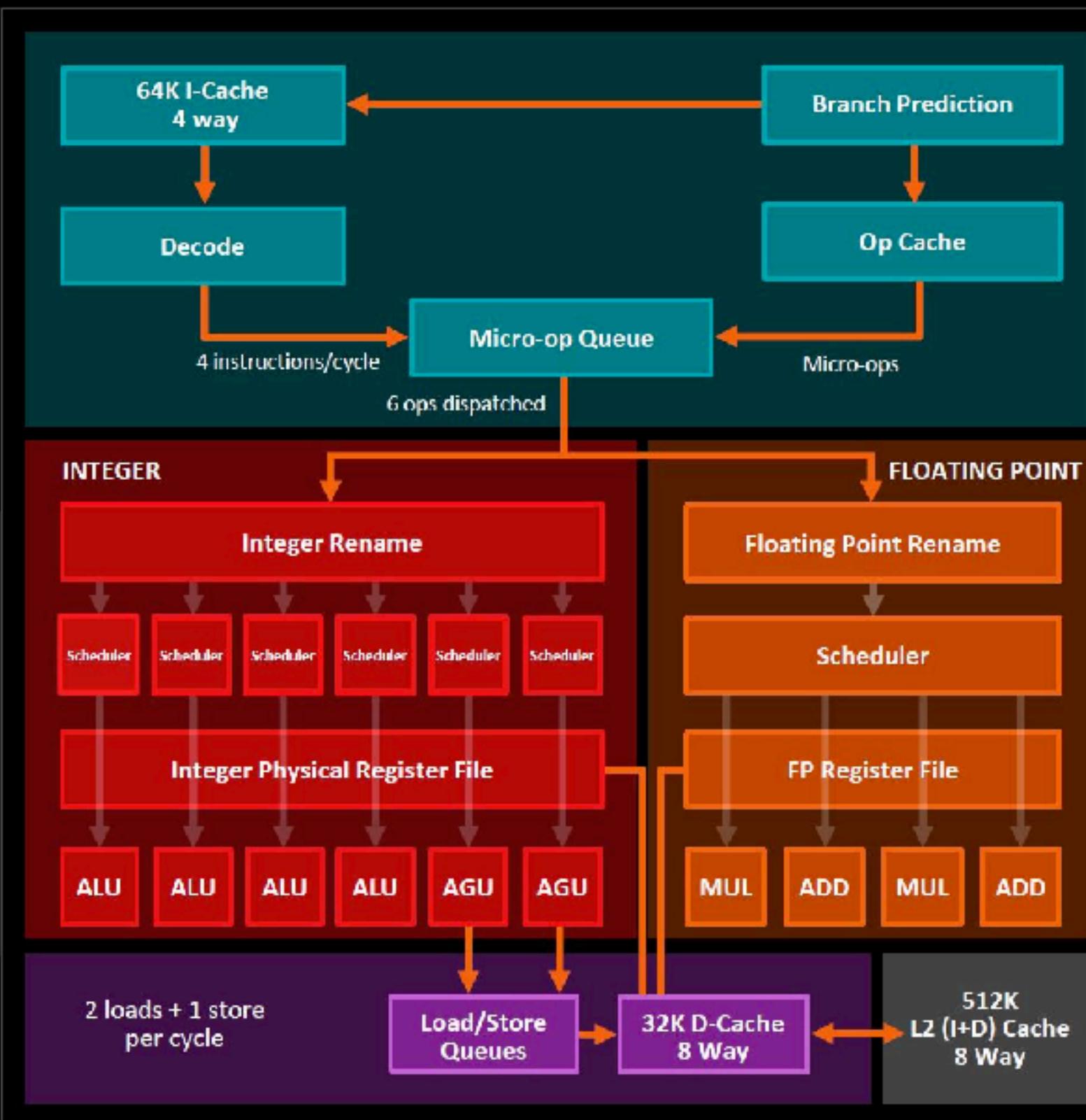


Intel Skylake



Intel Sandy Bridge



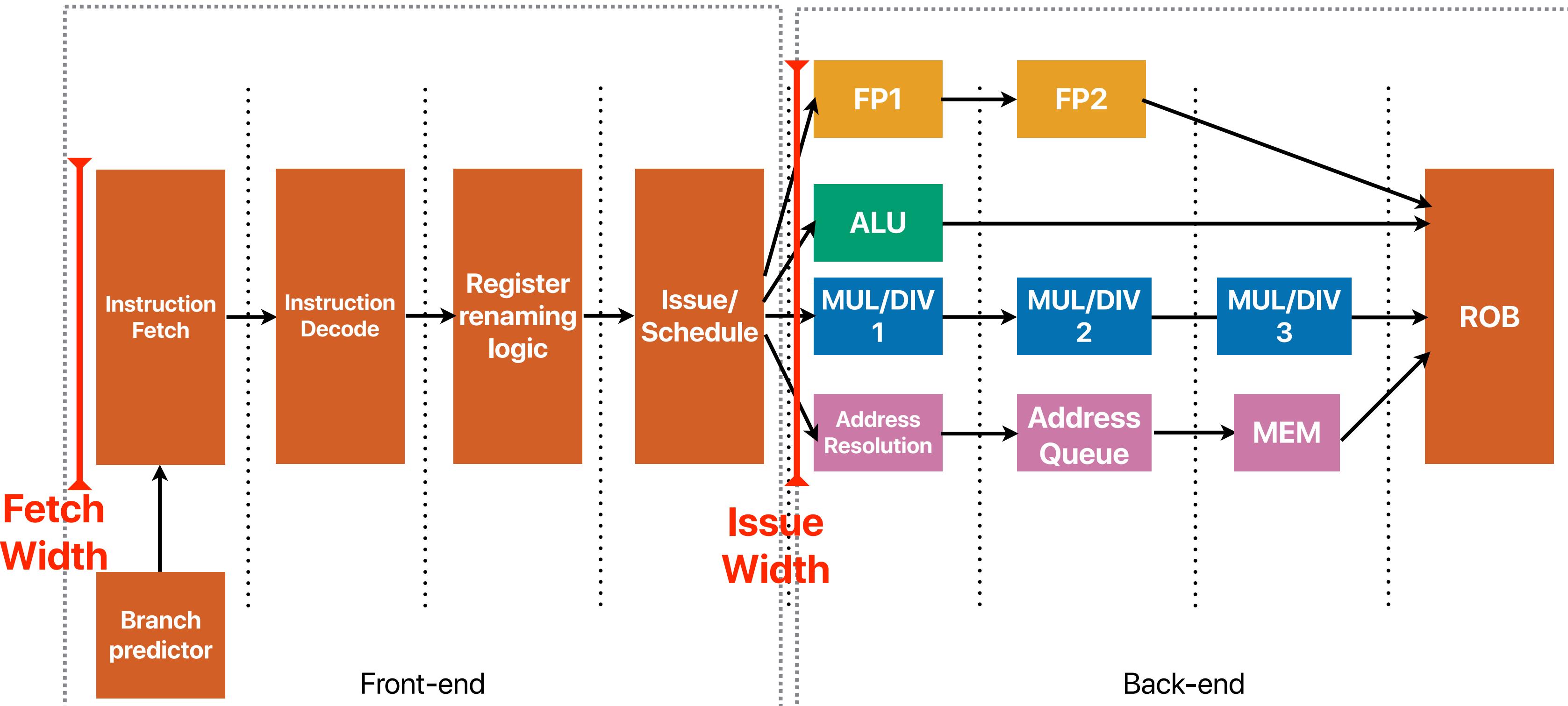


ZEN MICROARCHITECTURE

- ▲ Fetch Four x86 instructions
- ▲ Op Cache instructions
- ▲ 4 Integer units
 - Large rename space – 168 Registers
 - 192 instructions in flight/8 wide retire
- ▲ 2 Load/Store units
 - 72 Out-of-Order Loads supported
- ▲ 2 Floating Point units x 128 FMACs
 - built as 4 pipes, 2 Fadd, 2 Fmul
- ▲ I-Cache 64K, 4-way
- ▲ D-Cache 32K, 8-way
- ▲ L2 Cache 512K, 8-way
- ▲ Large shared L3 cache
- ▲ 2 threads per core

Putting it all together

Pipeline SuperScalar/OoO/ROB



Recap: What about "linked list"

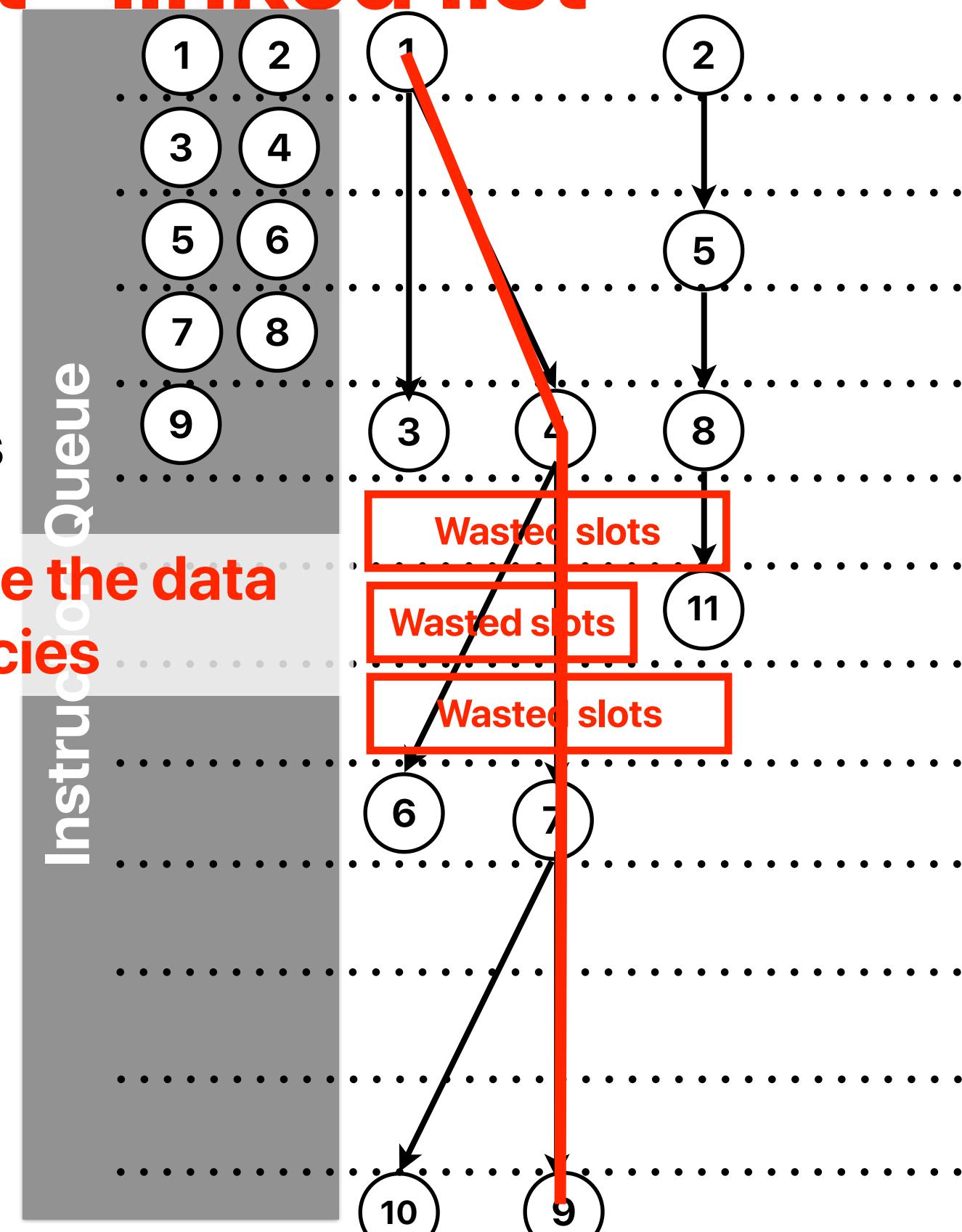
Static instructions

```
LOOP: ld X10, 8(X10)  
      addi X7, X7, 1  
      bne X10, X0, LOOP
```

Dynamic instructions

```
① ld X10, 8(X10)  
② addi X7, X7, 1  
③ bne X10, X0, LOOP  
④ ld X10, 8(X10)  
⑤ addi X7, X7, 1  
⑥ bne X10, X0, LOOP  
⑦ ld X10, 8(X10)  
⑧ addi X7, X7, 1  
⑨ bne X10, X0, LOOP
```

ILP is low because the data dependencies



Demo: ILP within a program

- perf is a tool that captures performance counters of your processors and can generate results like branch mis-prediction rate, cache miss rates and ILP.

Announcements

- Assignment #3 due this **Wednesday**
- **Final exam will be online**
- Reading Quiz due next Monday
- Project is released
 - Please check website to the link of GitHub repo
 - You may discuss, but each needs an individual/distinguishable version of code
 - You need to write a brief report
 - Grading rubrics
 - 20% — report
 - 20% — if your code can compile and run
 - 60% — performance based. The sample prefetcher is the baseline. We calculate your score at this part using $\min(\text{Speedup}-1, 1)$. If you can speedup by 2, you score full credits in this part
 - Due 11/29 — **no extension**

