Dynamic Instruction Scheduling (II)

Hung-Wei Tseng

Recap: Three pipeline hazards

- Structural hazards — resource conflicts cannot support simultaneous execution of instructions in the pipeline
- Control hazards — the PC can be changed by an instruction in the pipeline
- Data hazards — an instruction depending on a the result that's not yet generated or propagated when the instruction needs that

Recap: addressing hazards

- Structural hazards
  - Stall
  - Modify hardware design
- Control hazards
  - Stall
  - Static prediction
  - Dynamic prediction
- Data hazards
  - Stall
  - Data forwarding
  - Dynamic Scheduling

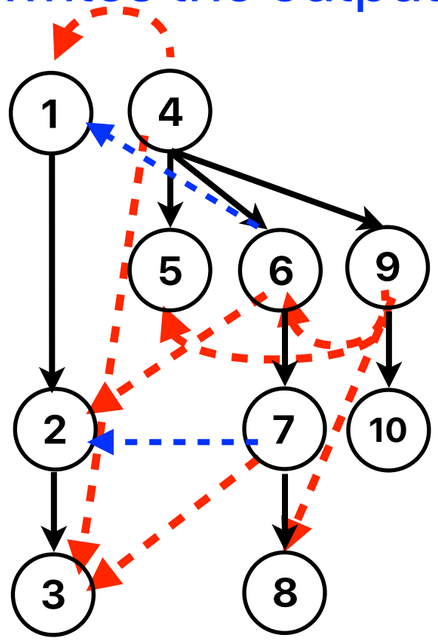
What do you need to execution an instruction?

- 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

False dependencies

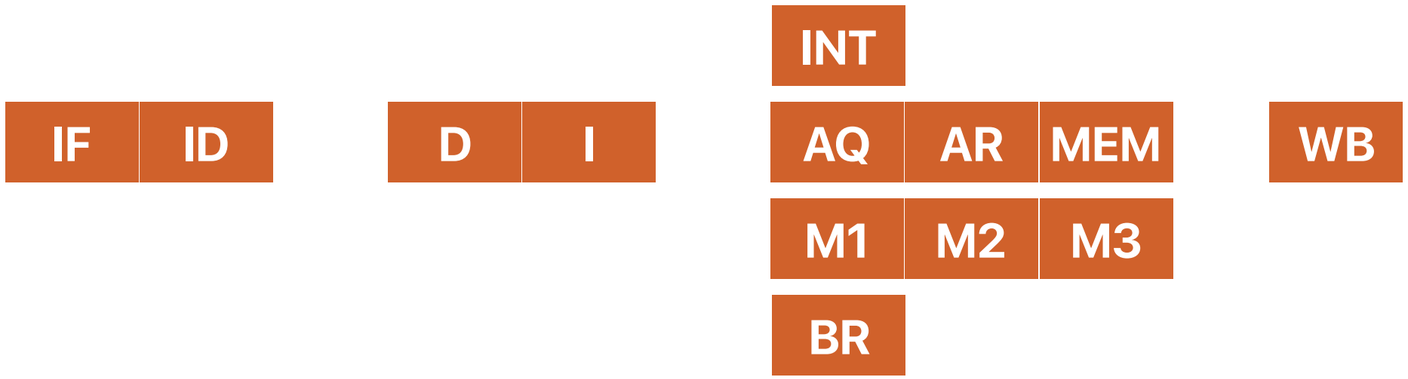
- We are still limited by **false dependencies**
- They are not “true” dependencies because they don’t have an arrow in data dependency graph
  - WAR (Write After Read): a later instruction overwrites the source of an earlier one
    - 4 and 1 4 and 3, 6 and 2, 7 and 3, 9 and 5, 9 and 6, 9 and 8
  - WAW (Write After Write): a later instruction overwrites the output of an earlier one
    - 6 and 1, 7 and 2

```
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
```

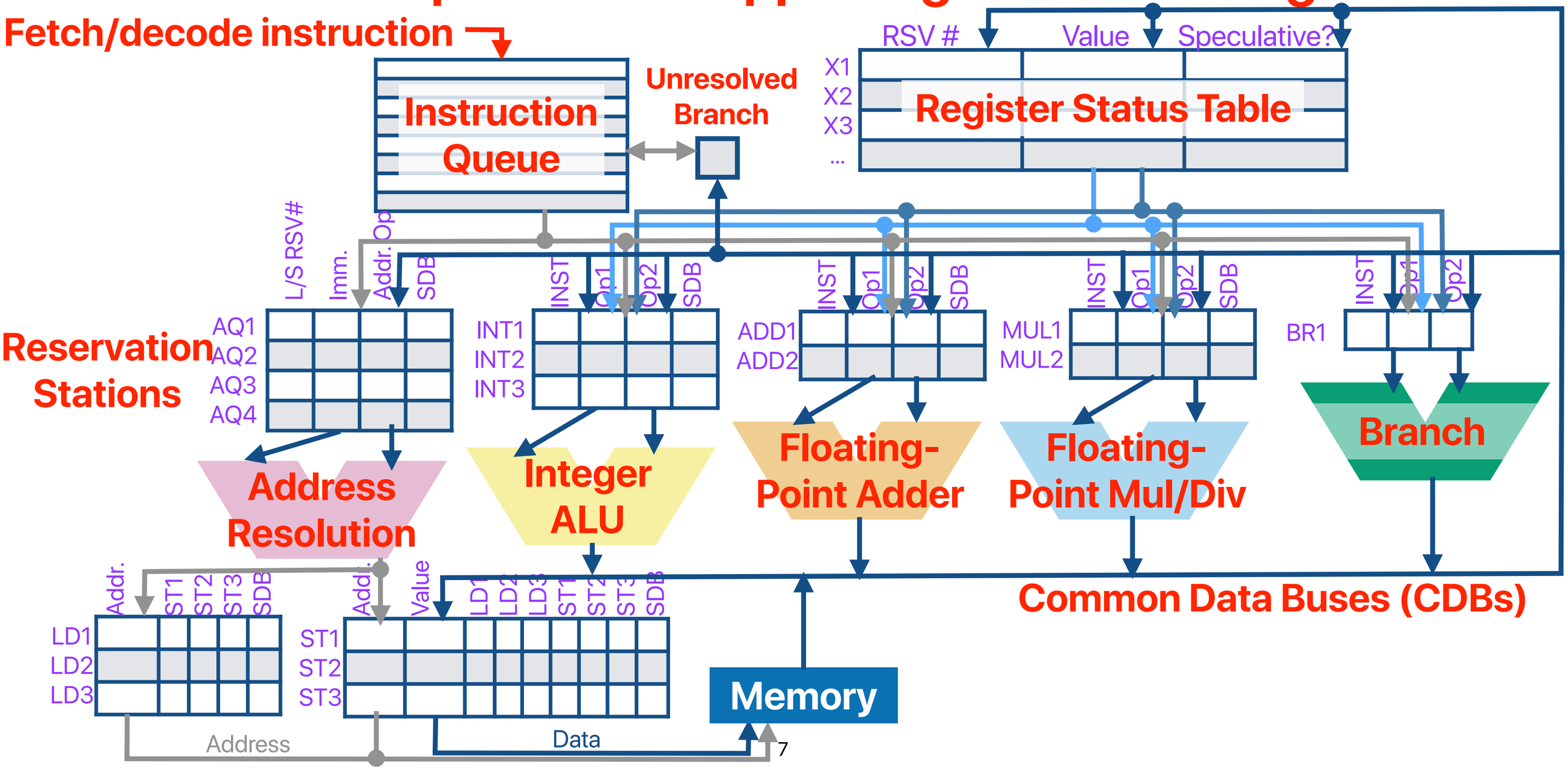


Pipeline in Tomasulo

- Dispatch (D) — allocate a “reservation station” for a decoded instruction
- Issue (I) — collect pending values/branch outcome from common data bus
- Execute (INT, AQ/AQ/MEM, M1/M2/M3, BR) — send the instruction to its corresponding pipeline if no structural hazards
- Write Back (WB) — broadcast the result through CDB

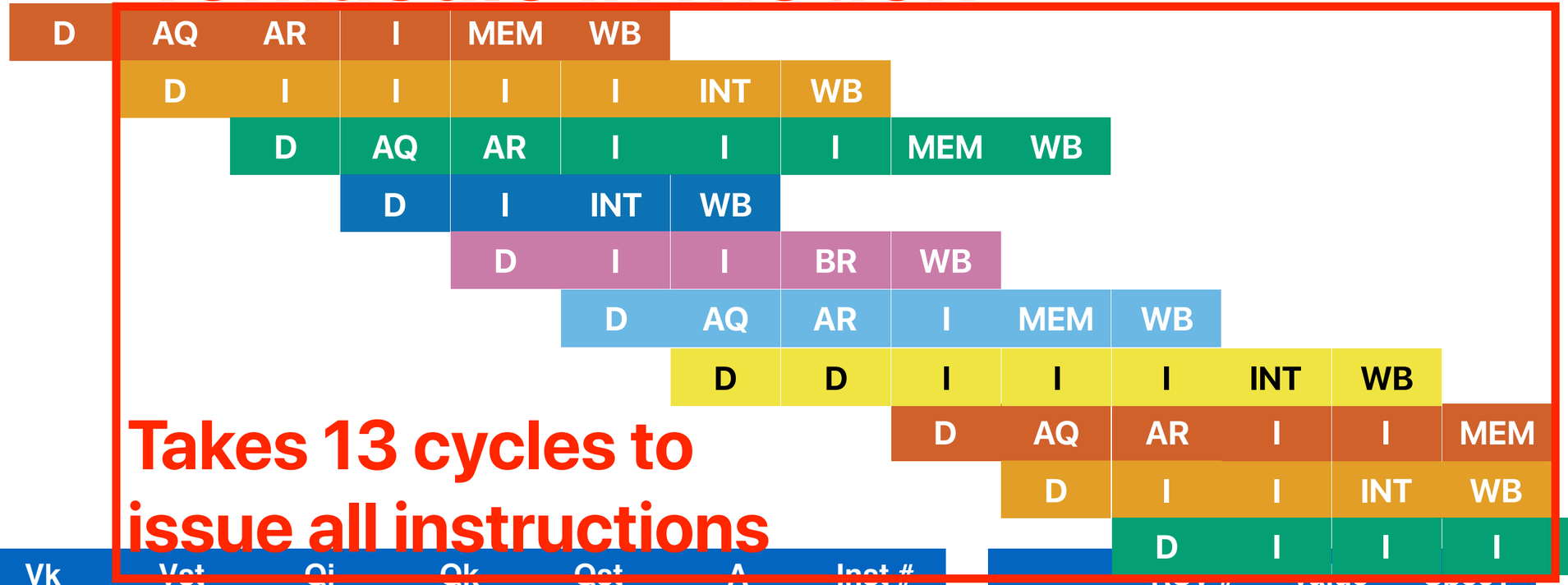


Overview of a processor supporting Tomasulo's algorithm



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
```



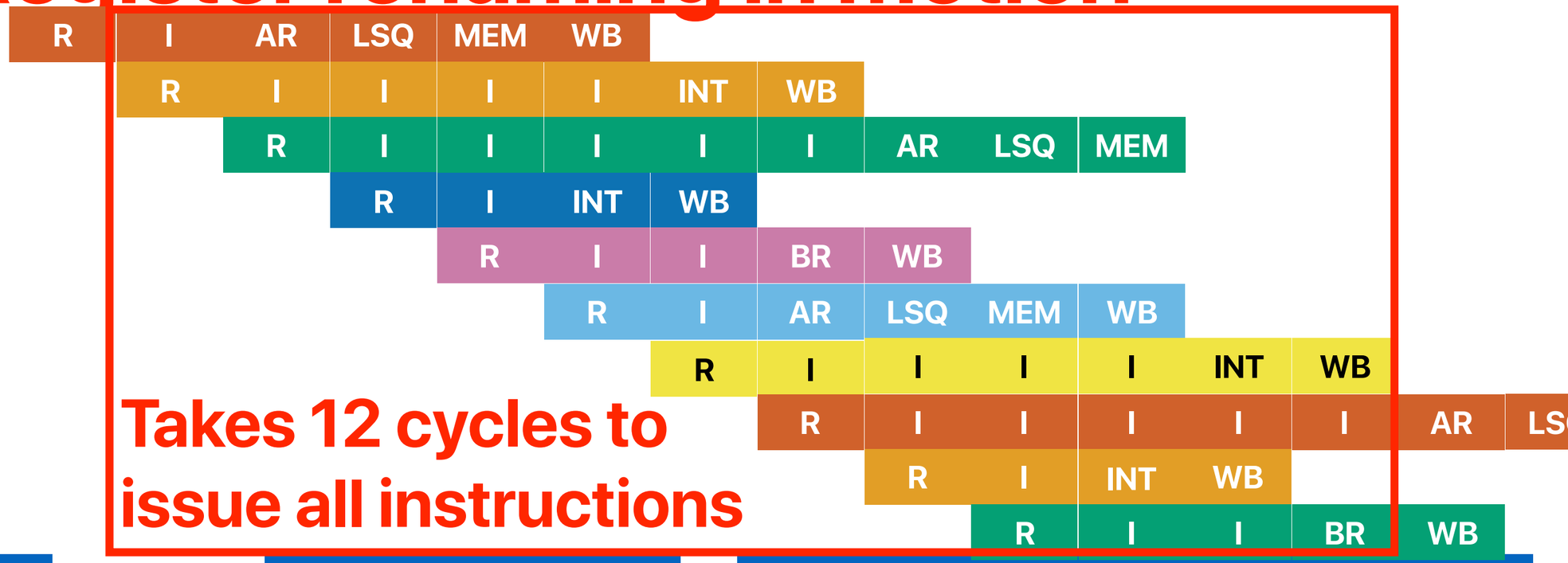
Takes 13 cycles to issue all instructions

	INST	Vj	Vk	Vst	Qj	Qk	Qst	A	Inst #
LD1	~~ld~~	~~0~~	~~[X10]~~						~~1~~
LD2	~~ld~~	~~0~~	~~ADD2~~						~~6~~
LD3									
ST1	~~sd~~	~~0~~	~~[X10]~~	~~ADD1~~					~~3~~
ST2	~~sd~~	~~0~~	~~[X10]~~	~~ADD2~~					~~8~~
ST3									
ADD1	add	8	ADD2						9
ADD2	~~add~~		~~[X12]~~		~~[LD2]~~				~~7~~
MUL1									
MUL2									
BR	br		[X5]	ADD1					10

X5	
X6	LD2
X7	ADD2
X10	ADD1
X12	

Register renaming 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
```



Takes 12 cycles to issue all instructions

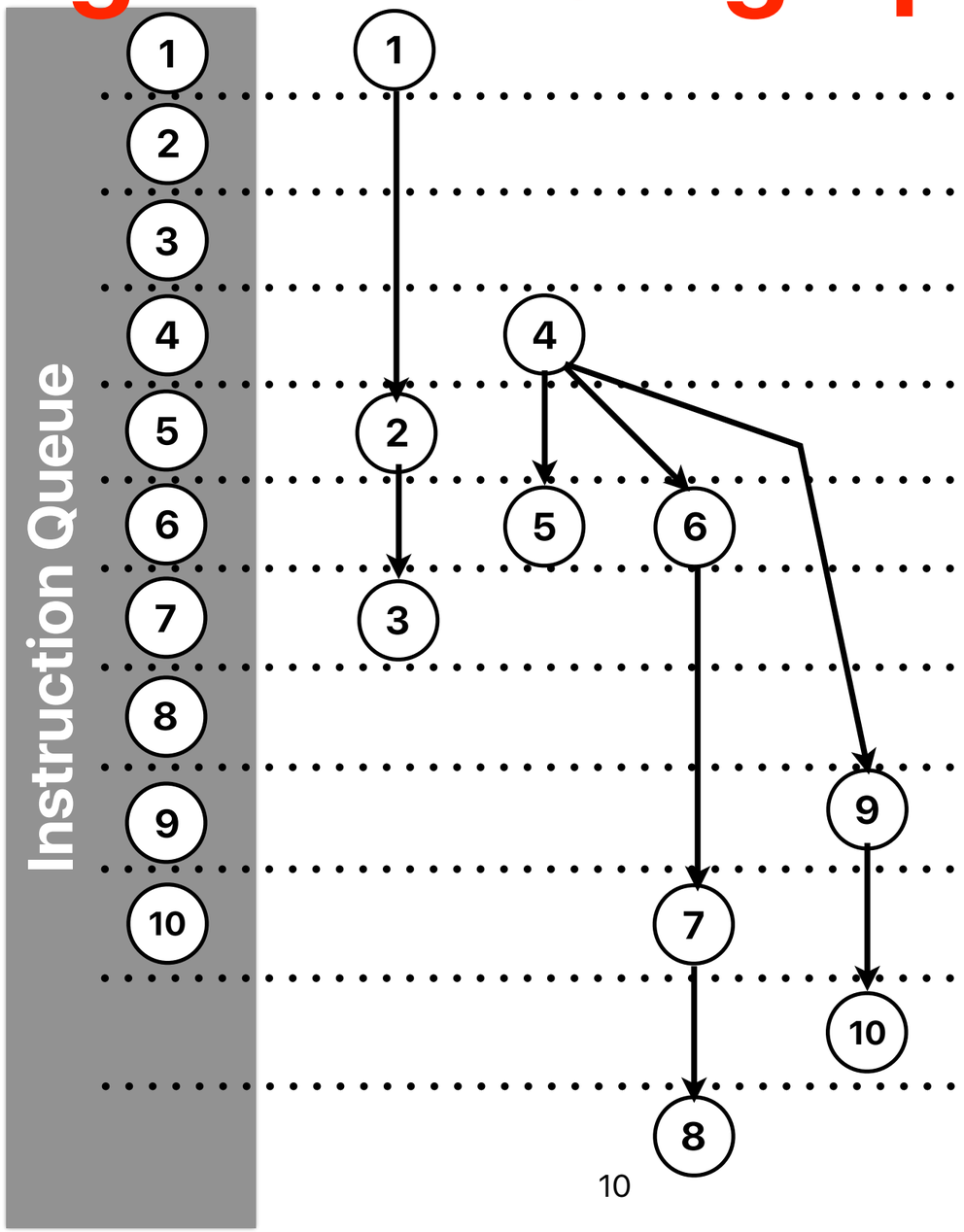
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	~~sd P2, 0(X10)~~
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	~~add P5, P1, X12~~
8	sd P5, 0(P3)
9	~~addi P6, P3, 8~~
10	~~bne P6, 0(X10)~~

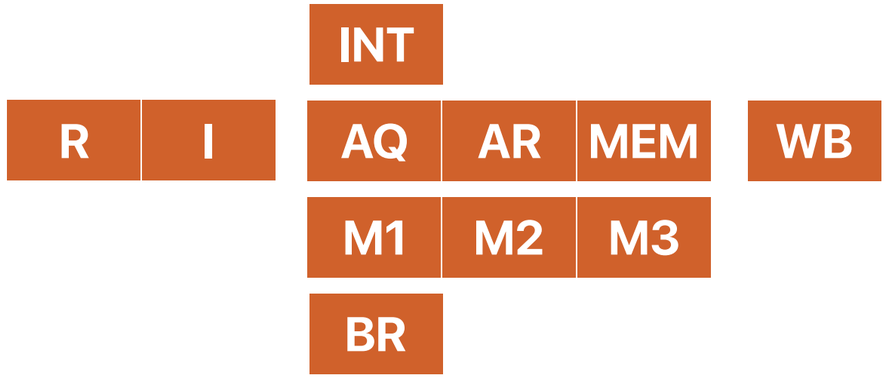
	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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

Through data flow graph analysis

```
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
```





INT — 2 cycles for depending instruction to start
MEM — 4 cycles for the depending instruction to start
MUL/DIV — 4 cycles for the depending instruction to start
BR — 2 cycles to resolve

Outline

- Register renaming (cont.)
- SuperScalar
- Reorder buffer & Speculative execution

Register Renaming (cont.)

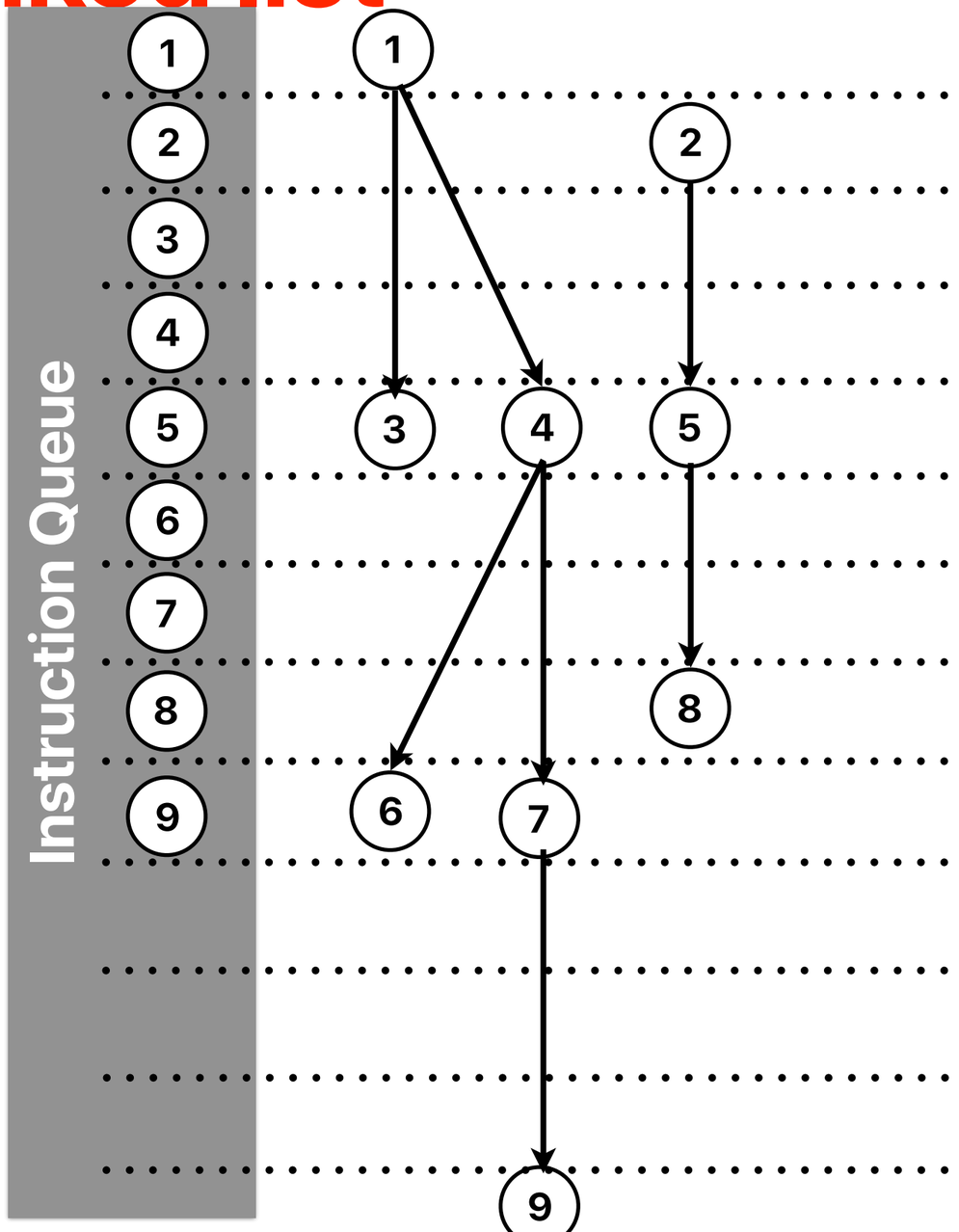
What about “linked list”

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 it’s 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 only fetches 1 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

Register renaming in motion

R

```
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
```

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

	Physical Register
X5	
X6	P1
X7	
X10	
X12	

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

Register renaming in motion


```
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
```

	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
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			

Register renaming in motion


```
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
```

	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	
5	
6	
7	
8	
9	
10	

	Physical Register
X5	
X6	
X7	P2
X10	P1
X12	

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

Register renaming in motion

```
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
```

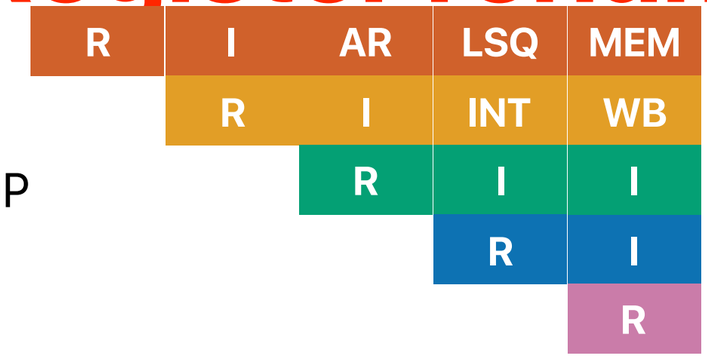
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	
6	
7	
8	
9	
10	

	Physical Register
X5	
X6	
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			

Register renaming in motion

```
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
```



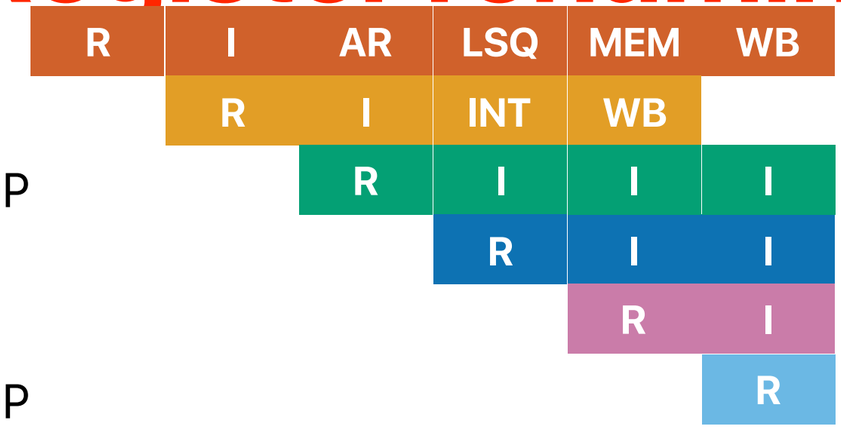
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	
7	
8	
9	
10	

	Physical Register
X5	
X6	
X7	P4
X10	P3
X12	

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

Register renaming in motion

```
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
```



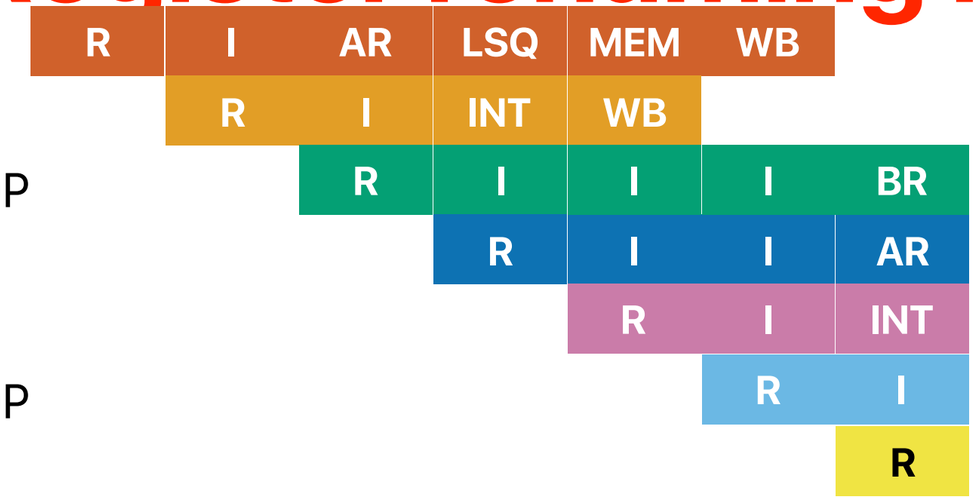
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	
8	
9	
10	

	Physical Register
X5	
X6	
X7	P4
X10	P3
X12	

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

Register renaming in motion

```
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
```



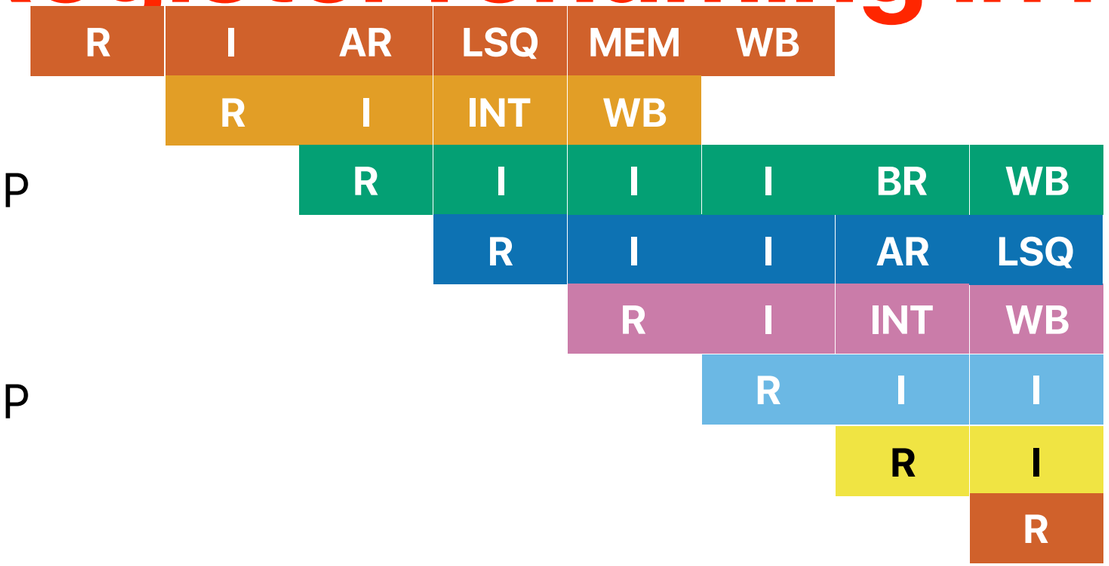
	Renamed instruction		
1	ld	P1,	0(X10)
2	add	P2,	X7, 1
3	bne	P1,	X0, LOOP
4	ld	P3,	0(P1)
5	add	P4,	P2, 1
6	bne	P3,	X0, LOOP
7	ld	P5,	0(P3)
8			
9			
10			

	Physical Register
X5	
X6	
X7	P4
X10	P5
X12	

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

Register renaming in motion

```
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
```



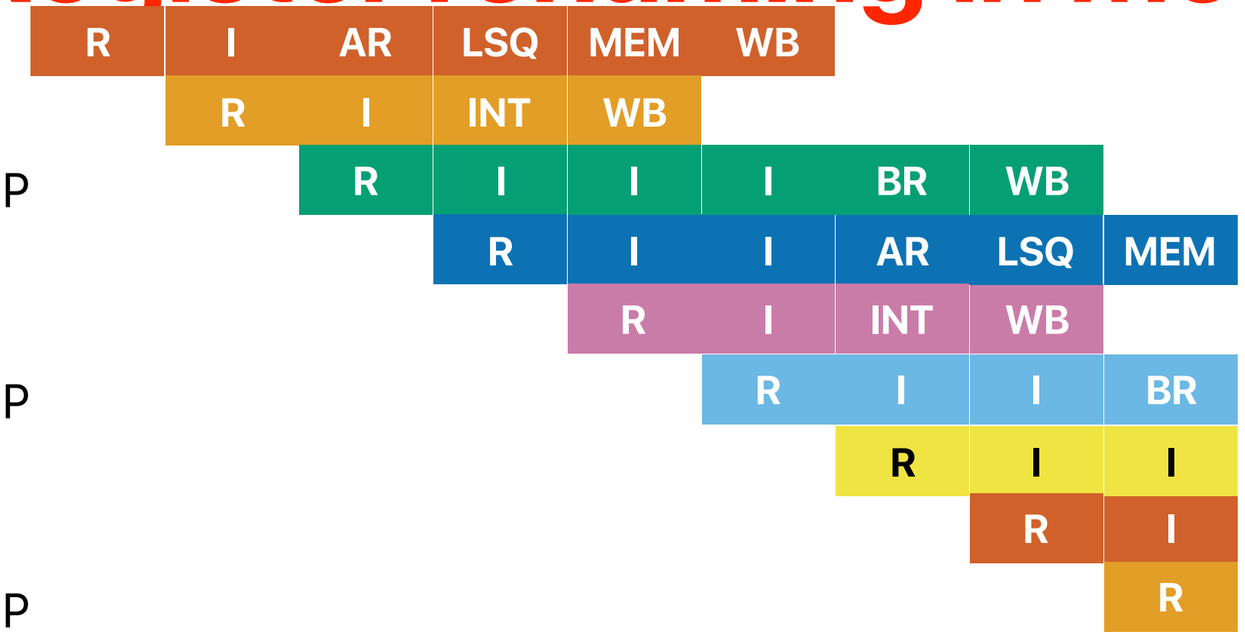
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	
10	

	Physical Register
X5	
X6	
X7	P6
X10	P5
X12	

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

Register renaming in motion

```
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
```



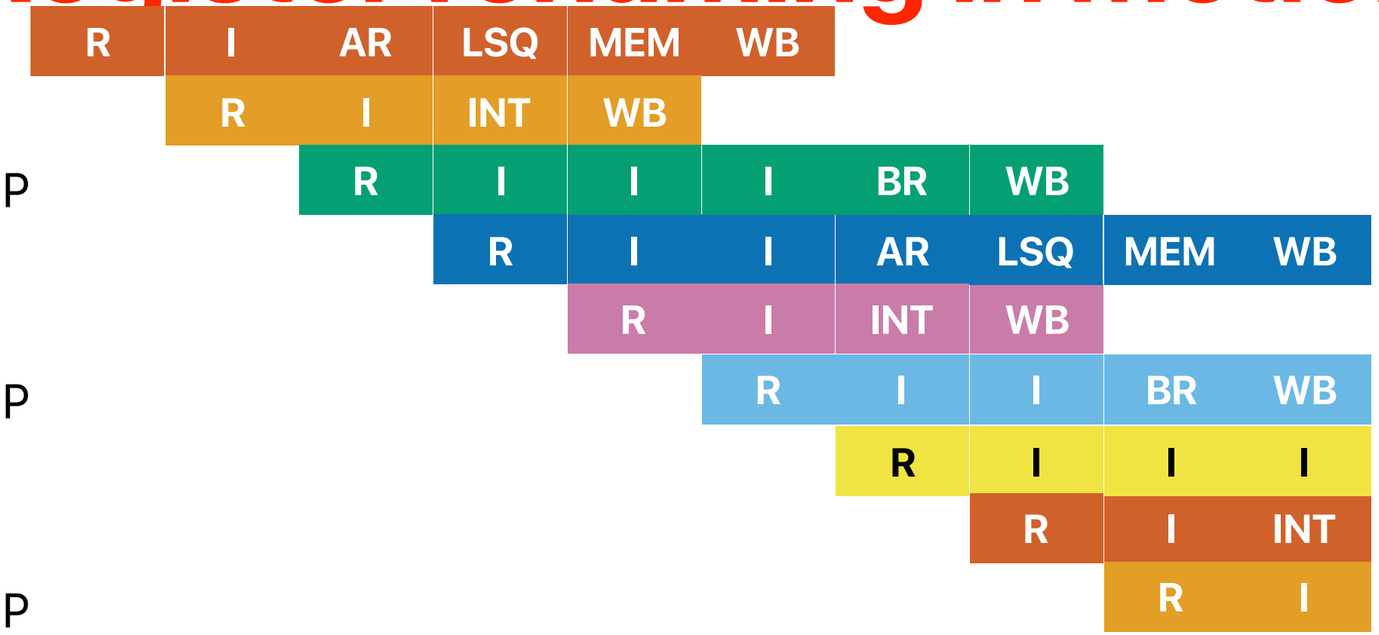
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	bne P5, X0, LOOP
10	

	Physical Register
X5	
X6	
X7	P6
X10	P5
X12	

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

Register renaming in motion

```
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
```



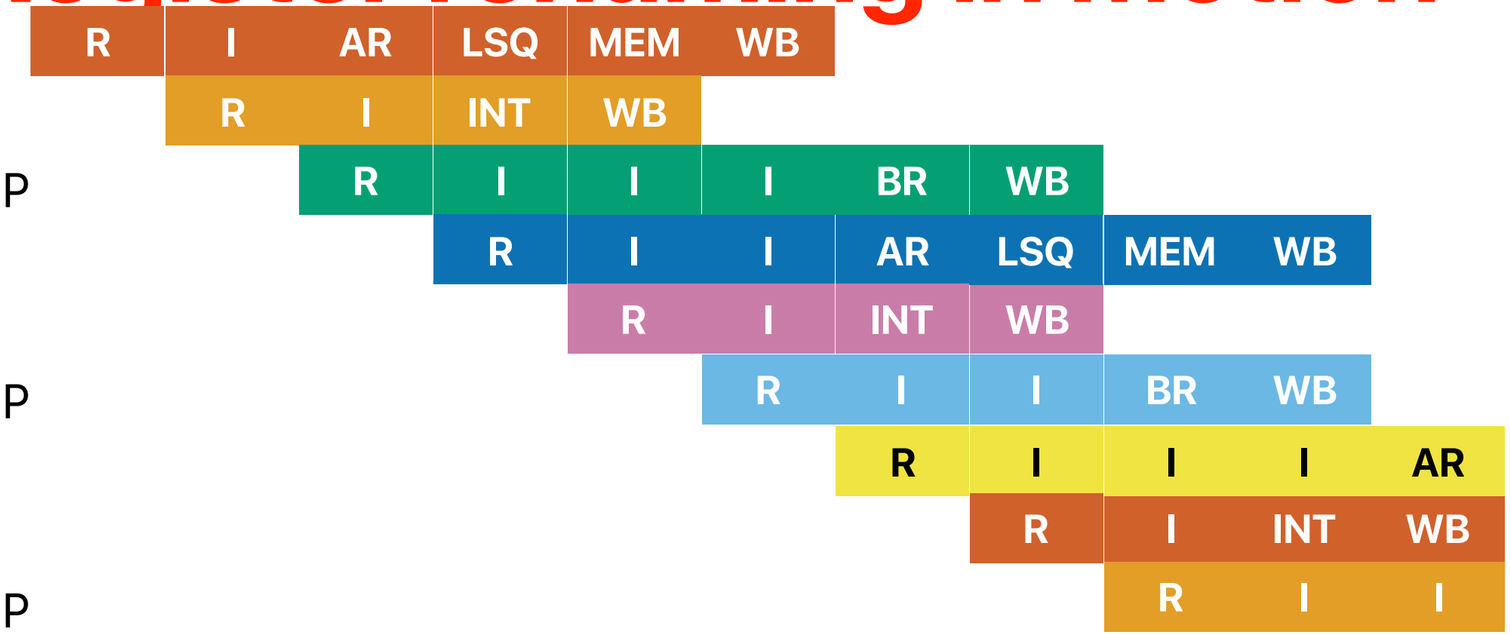
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	bne P5, X0, LOOP
10	

	Physical Register
X5	
X6	
X7	P6
X10	P5
X12	

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

Register renaming in motion

```
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
```



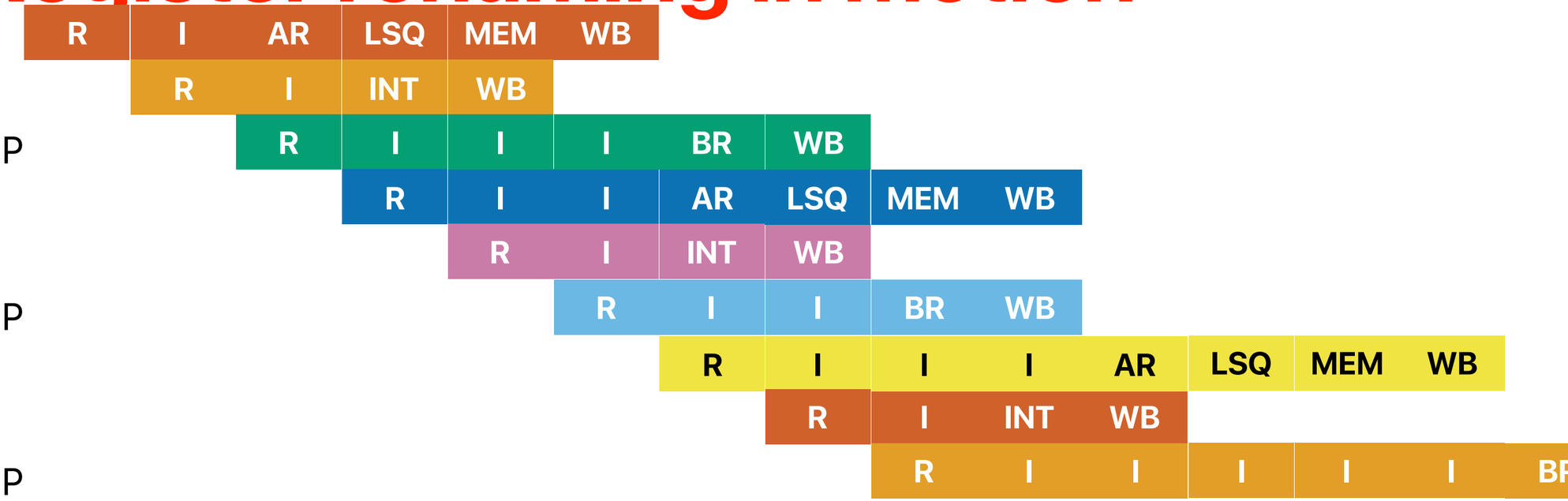
	Renamed instruction		
1	ld	P1,	0(X10)
2	add	P2,	X7, 1
3	bne	P1,	X0, LOOP
4	ld	P3,	0(P1)
5	add	P4,	P2, 1
6	bne	P3,	X0, LOOP
7	ld	P5,	0(P3)
8	add	P6,	P4, 1
9	bne	P5,	X0, LOOP
10			

	Physical Register
X5	
X6	
X7	P6
X10	P5
X12	

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

Register renaming in motion

```
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
```



	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	bne P5, X0, LOOP
10	

	Physical Register
X5	
X6	
X7	P6
X10	P5
X12	

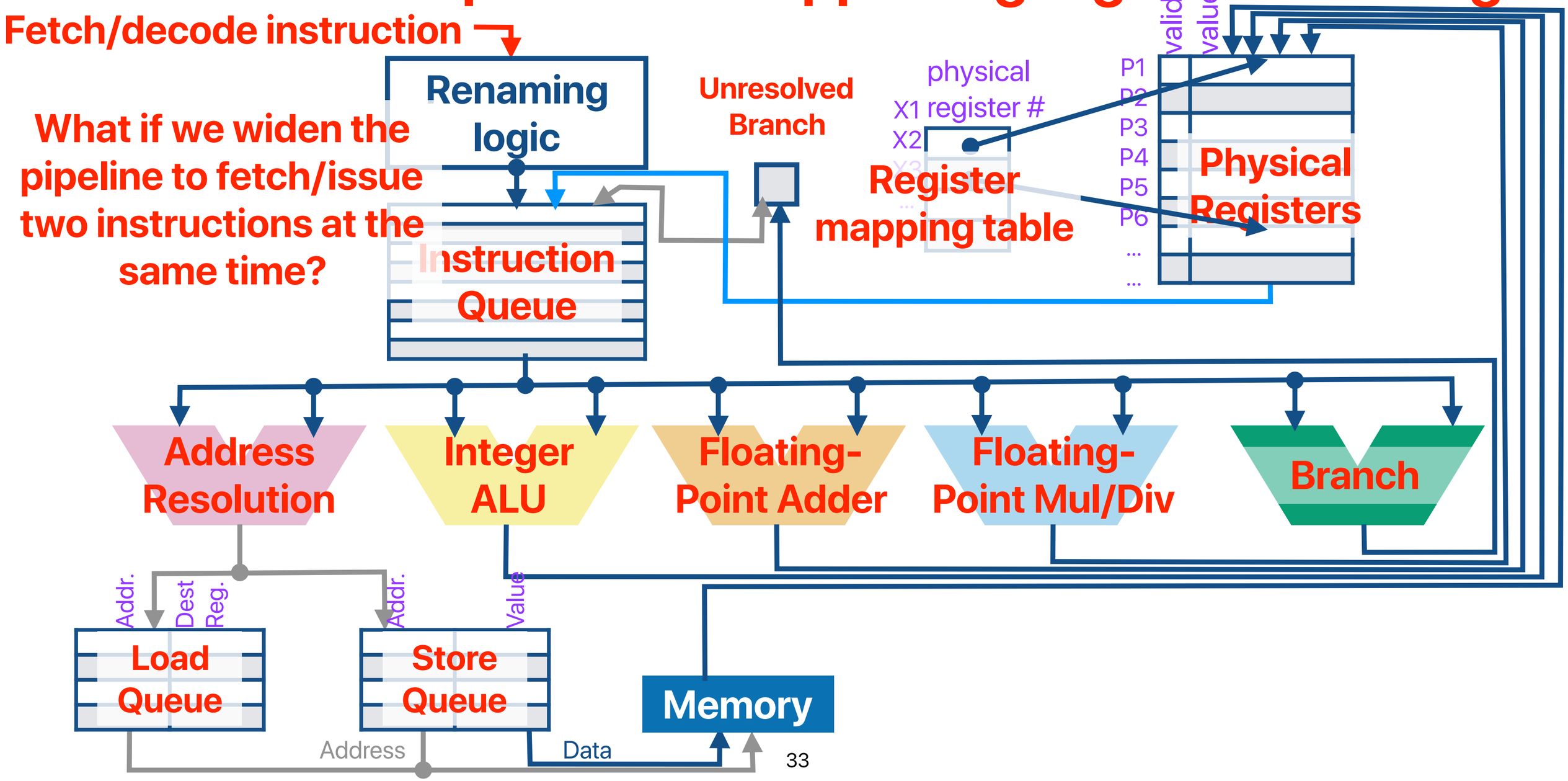
	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6	0		1
P2	1		1	P7			
P3	0		1	P8			
P4	0		1	P9			
P5	0		1	P10			

Super Scalar

Superscalar

- Since we have more functional units now, we should fetch/decode more instructions each cycle so that we can have more instructions to issue!
- Super-scalar: fetch/decode/issue more than one instruction each cycle
  - Fetch width: how many instructions can the processor fetch/decode each cycle
  - Issue width: how many instructions can the processor issue each cycle

Overview of a processor supporting register renaming



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
```

R

R

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

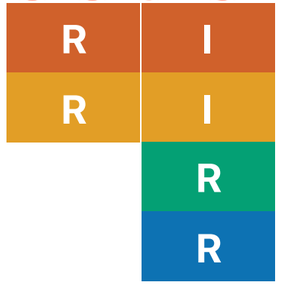
	Physical Register
X5	
X6	P1
X7	P2
X10	
X12	

	Valid	Value	In use
P1	0		1
P2	0		1
P3			
P4			
P5			

	Valid	Value	In use
P6			
P7			
P8			
P9			
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
```



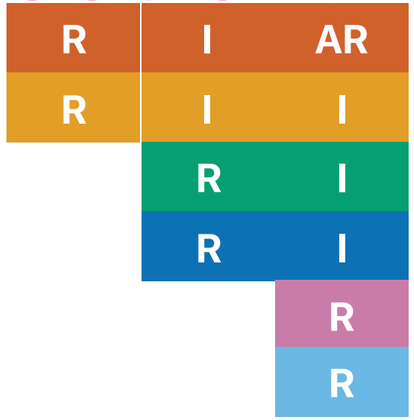
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	
6	
7	
8	
9	
10	

	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
```



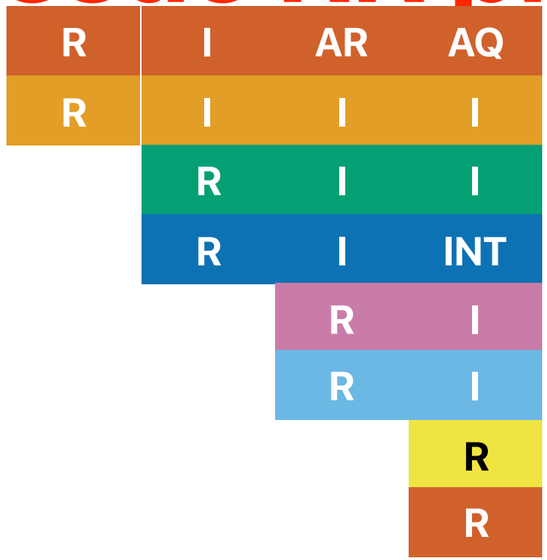
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	
8	
9	
10	

	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)
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
```



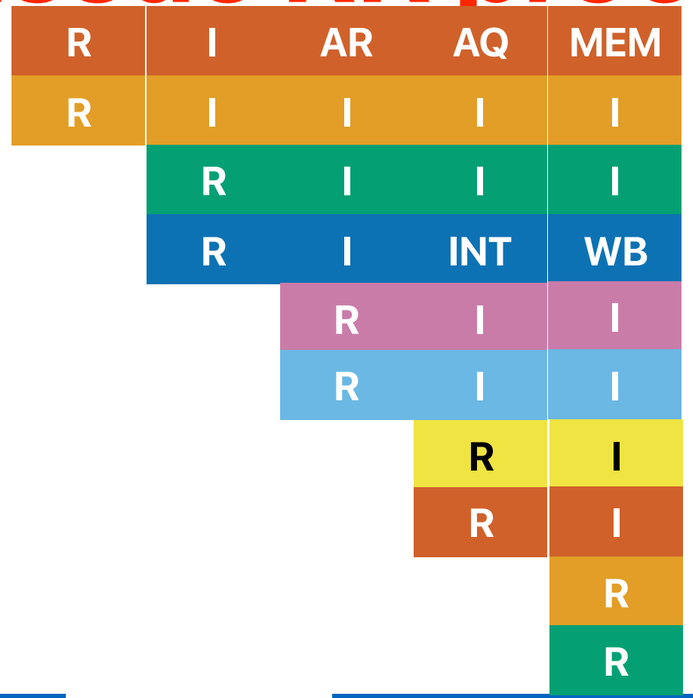
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	
10	

	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)
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
```



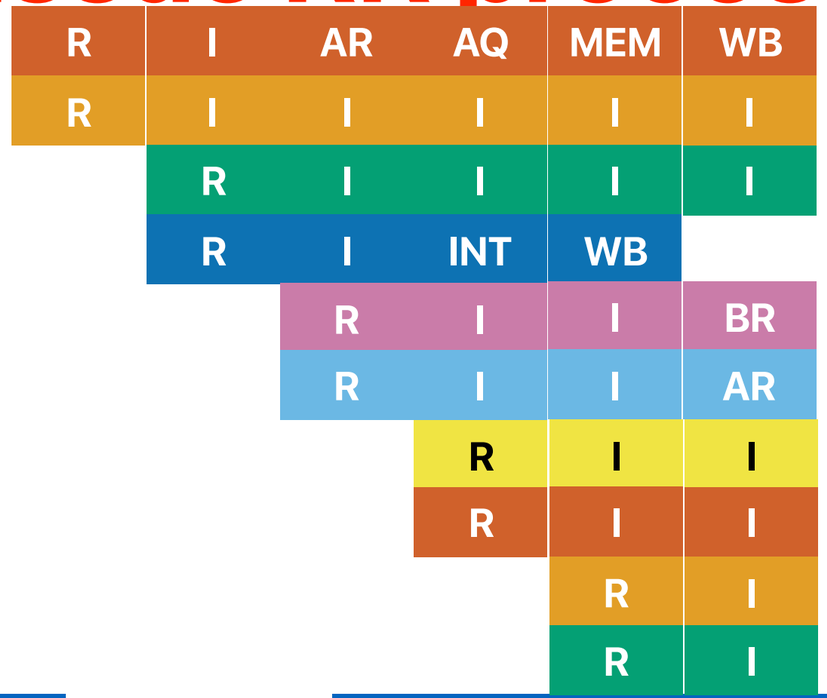
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	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

```
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
```



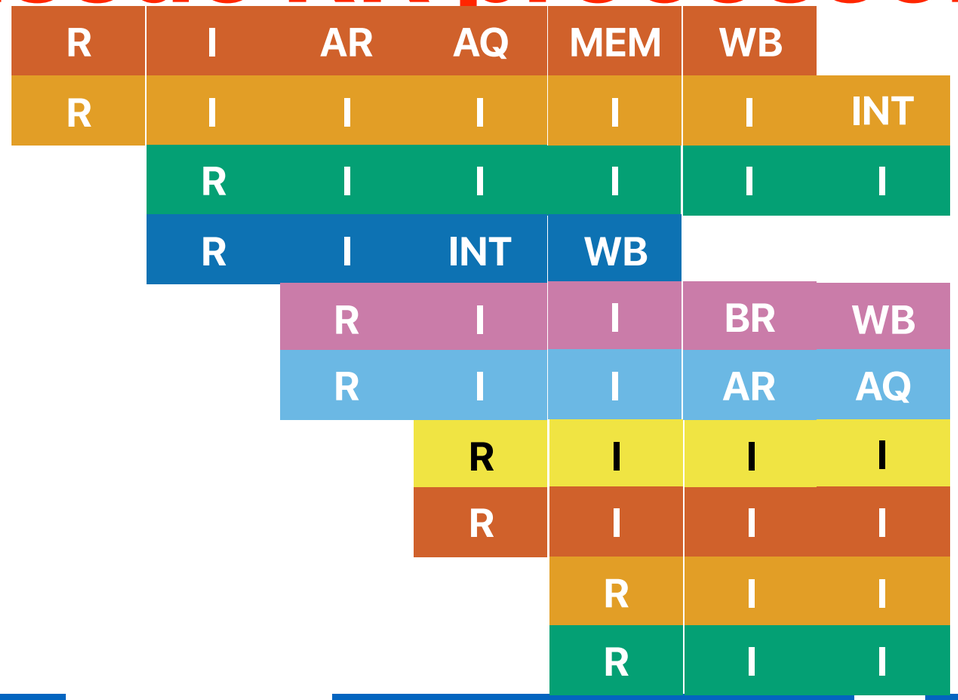
	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	~~addi P3, X10, 8~~
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	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

```
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
```



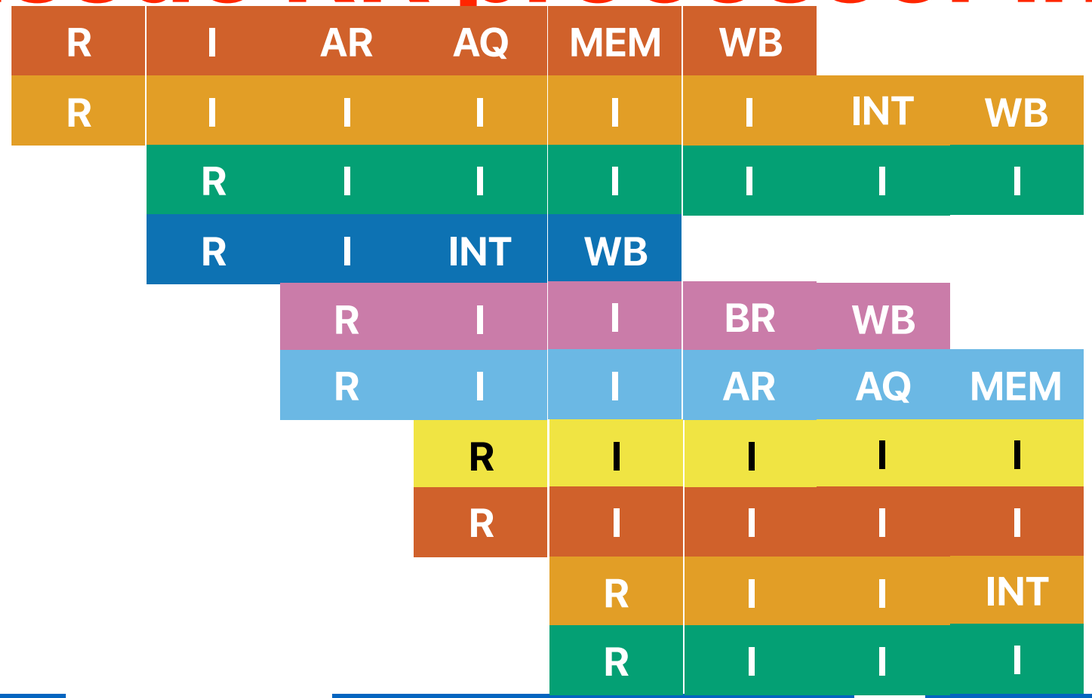
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	add P2, P1, X12
3	sd P2, 0(X10)
4	~~addi P3, X10, 8~~
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	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

```
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
```



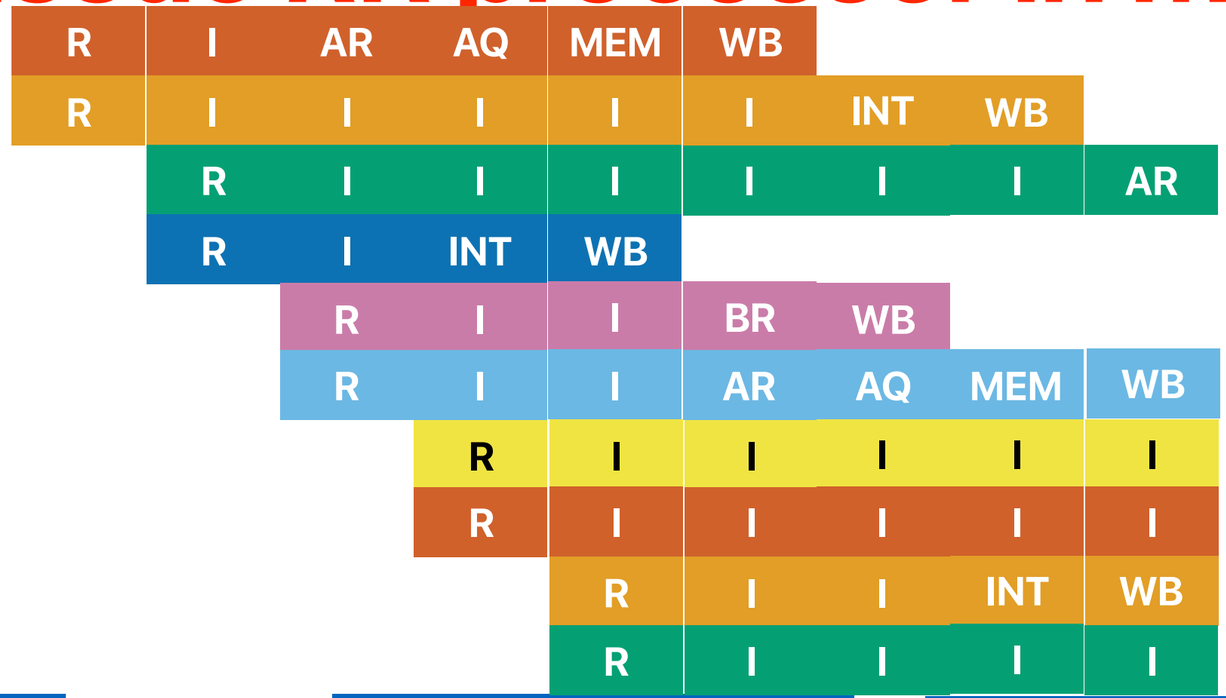
	Renamed instruction		
1	~~ld~~	~~P1, 0(X10)~~	
2	~~add~~	~~P2, P1, X12~~	
3	sd	P2, 0(X10)	
4	~~addi~~	~~P3, X10, 8~~	
5	~~bne~~	~~P3, X5, LOOP~~	
6	ld	P4, 0(P3)	
7	add	P5, P1, X12	
8	sd	P5, 0(P3)	
9	addi	P6, P3, 8	
10	bne	P6, 0(X10)	

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



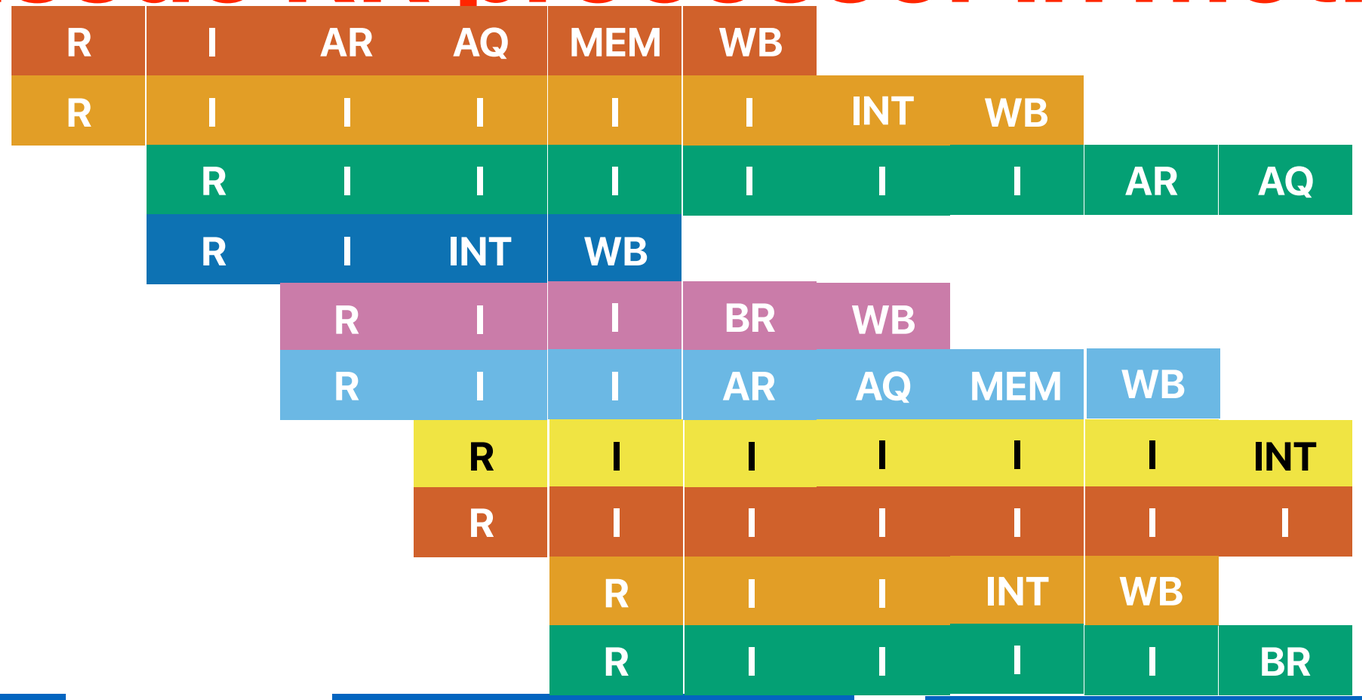
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	sd P2, 0(X10)
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



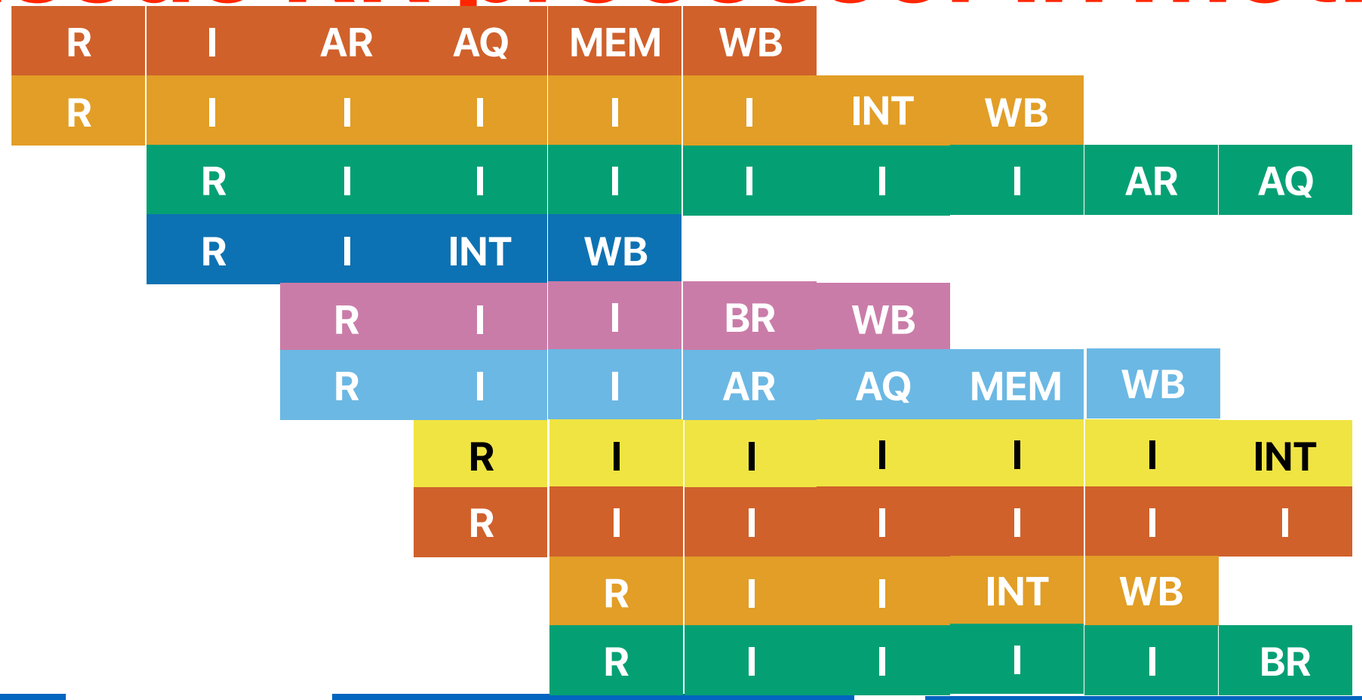
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	sd P2, 0(X10)
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	add P5, P1, X12
8	sd P5, 0(P3)
9	~~addi P6, P3, 8~~
10	bne P6, 0(X10)

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



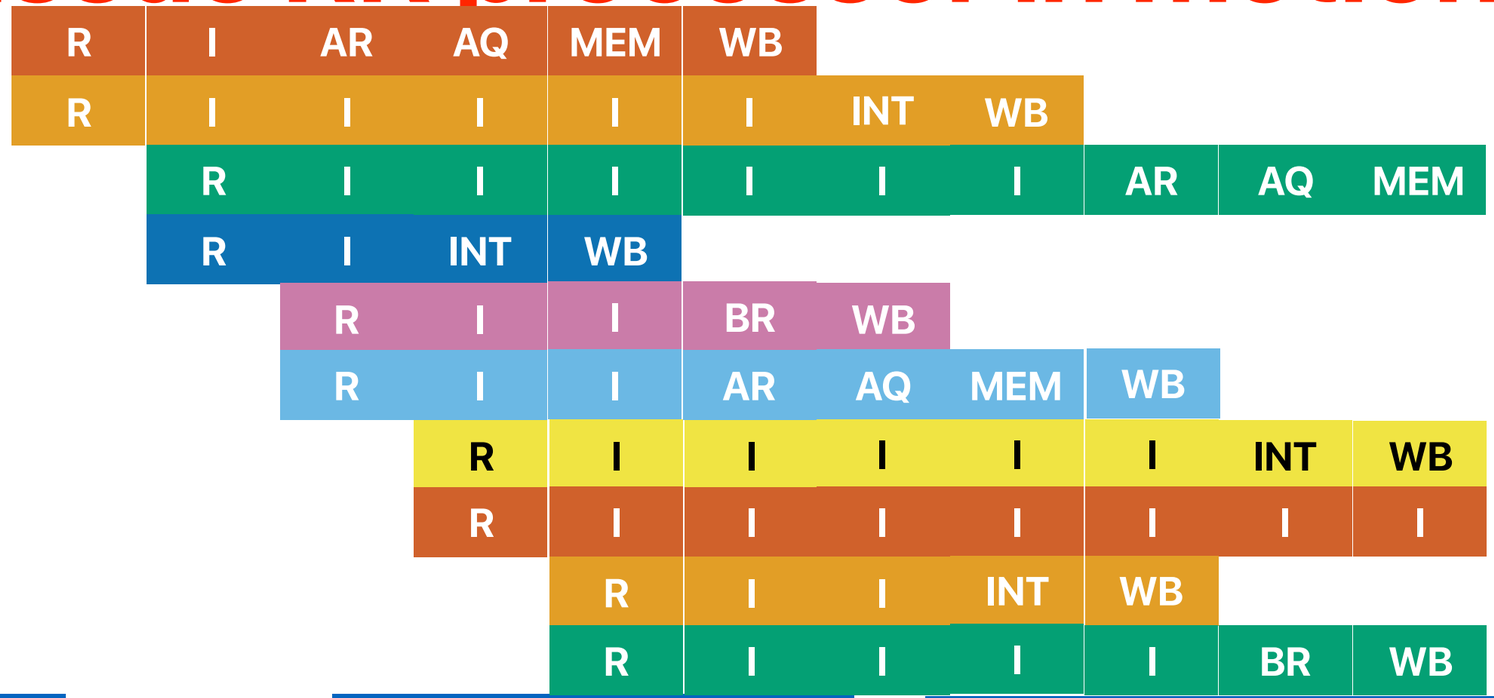
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	sd P2, 0(X10)
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	add P5, P1, X12
8	sd P5, 0(P3)
9	~~addi P6, P3, 8~~
10	bne P6, 0(X10)

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



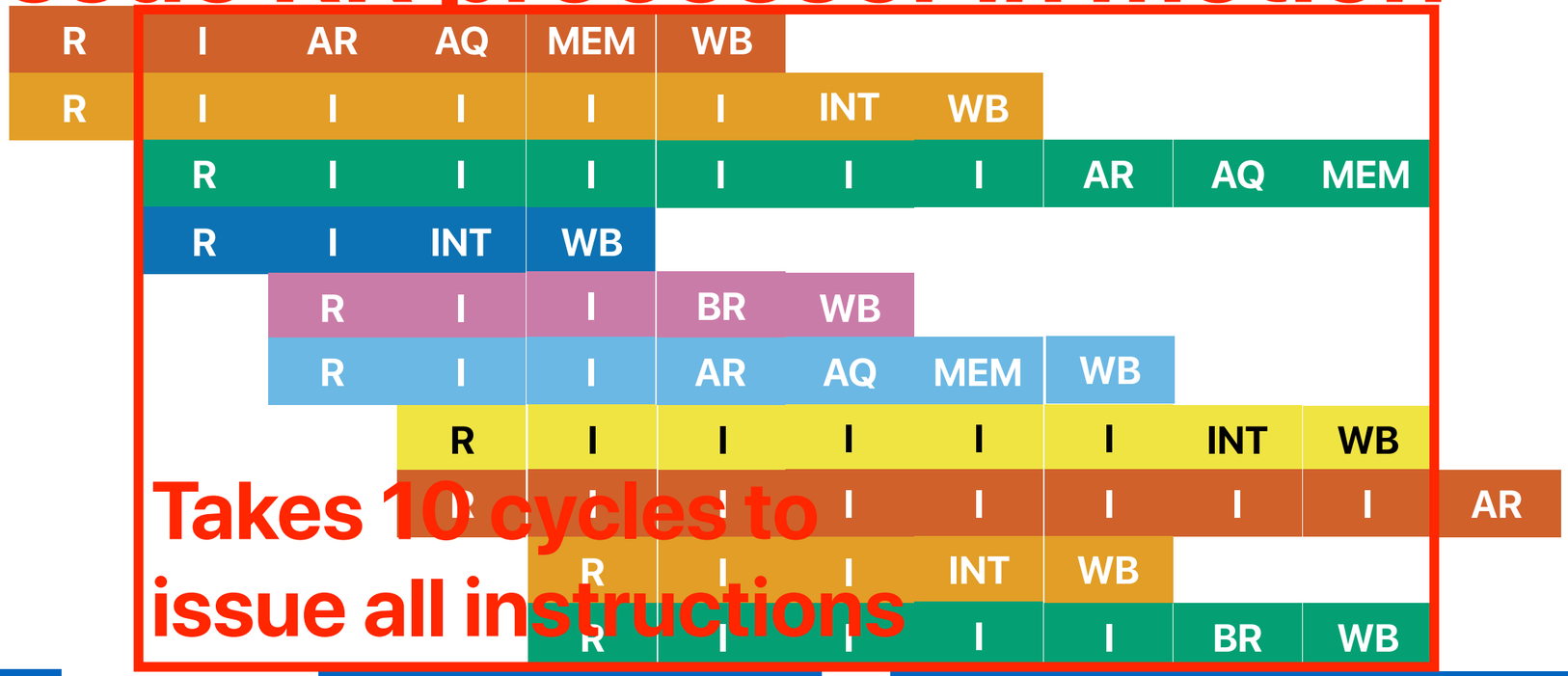
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	~~sd P2, 0(X10)~~
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	add P5, P1, X12
8	sd P5, 0(P3)
9	~~addi P6, P3, 8~~
10	bne P6, 0(X10)

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



Takes 10 cycles to issue all instructions

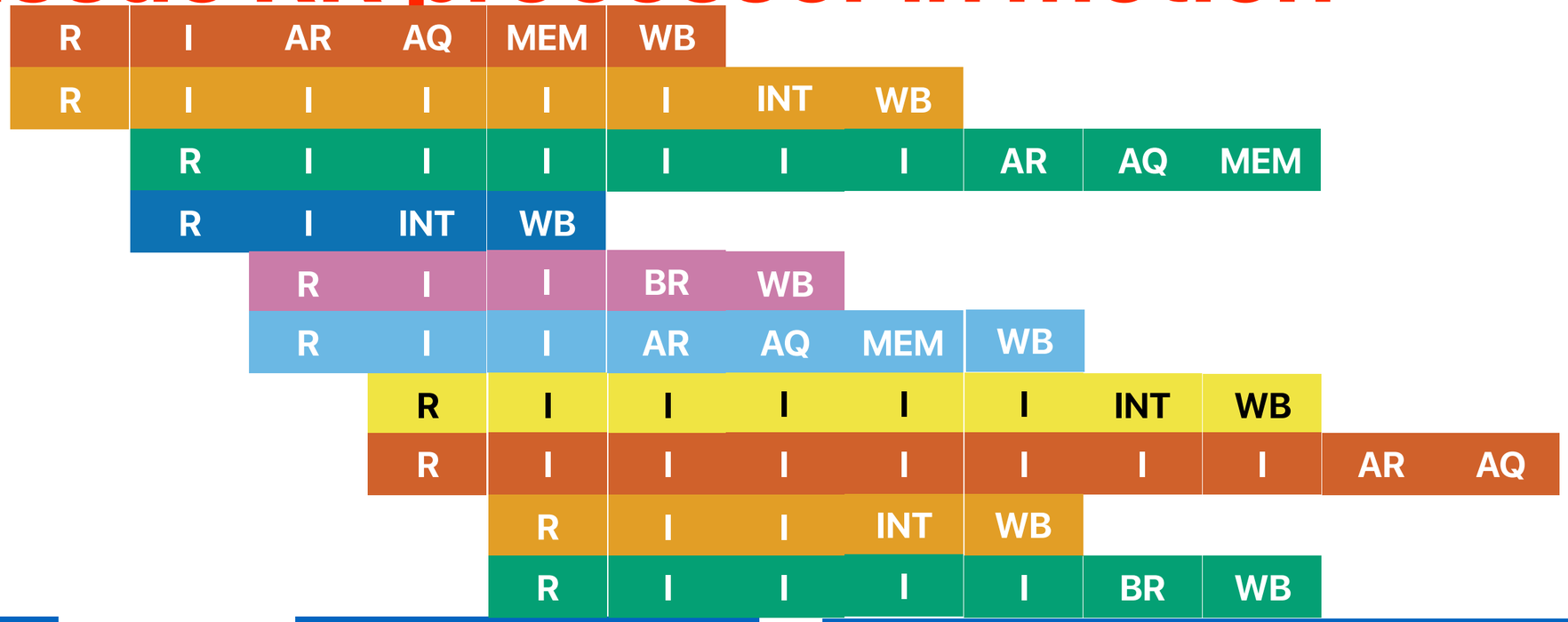
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	~~sd P2, 0(X10)~~
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	~~add P5, P1, X12~~
8	sd P5, 0(P3)
9	~~addi P6, P3, 8~~
10	~~bne P6, 0(X10)~~

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



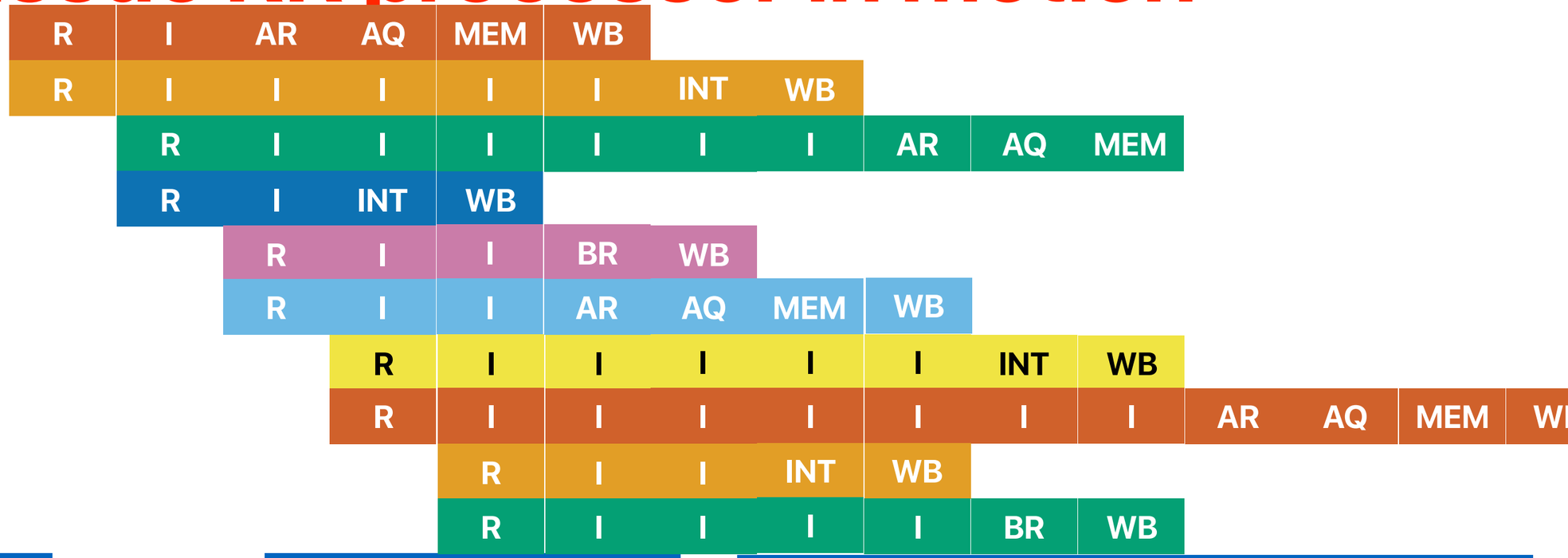
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	~~sd P2, 0(X10)~~
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	~~add P5, P1, X12~~
8	sd P5, 0(P3)
9	~~addi P6, P3, 8~~
10	~~bne P6, 0(X10)~~

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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

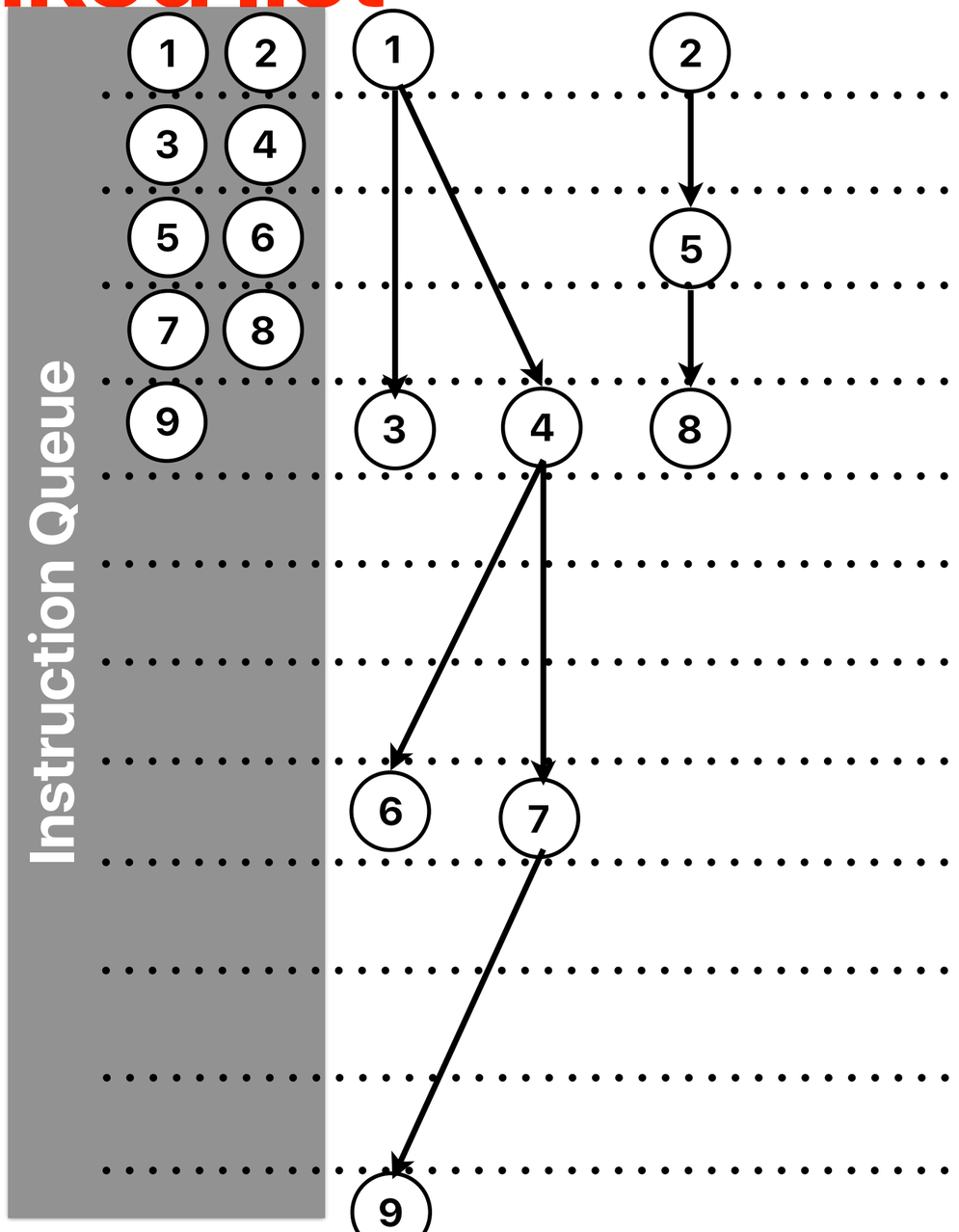
What about “linked list”

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 it’s 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

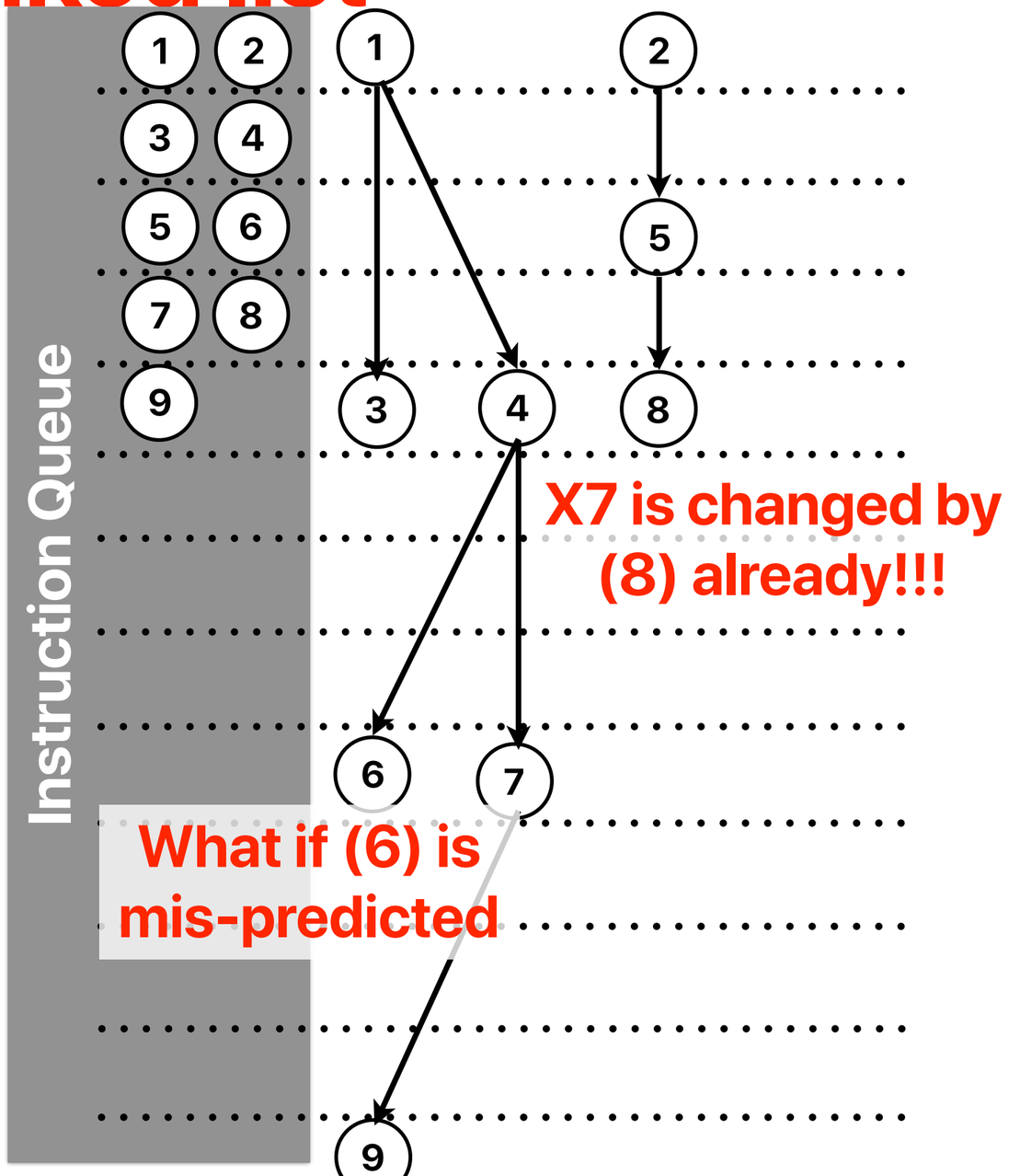
What about “linked list”

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 have exceptions?

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

  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

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)

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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

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

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
```

R

R

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

head

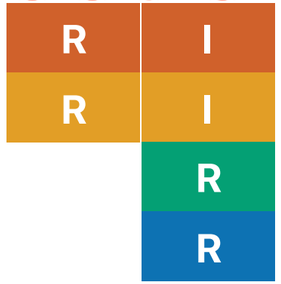
tail

	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	

head

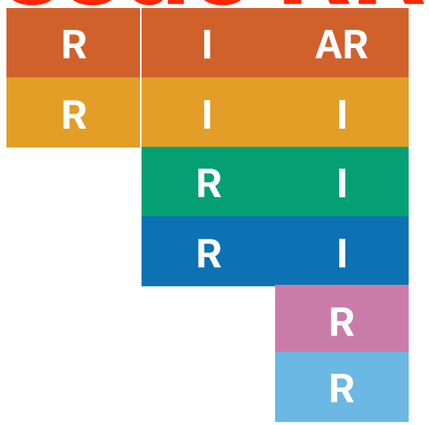
tail

	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		

head

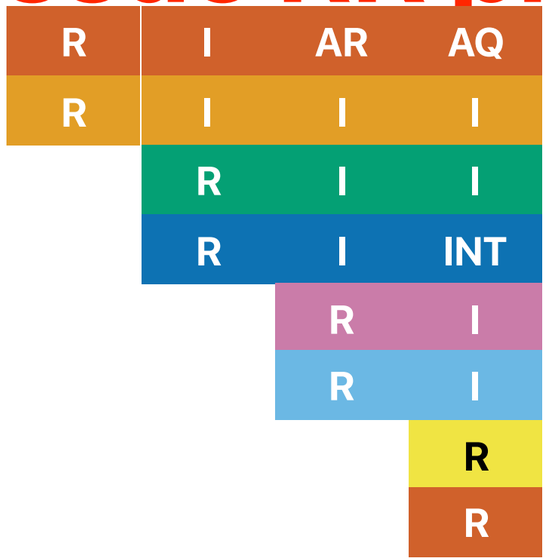
tail

	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)
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	
10	

head

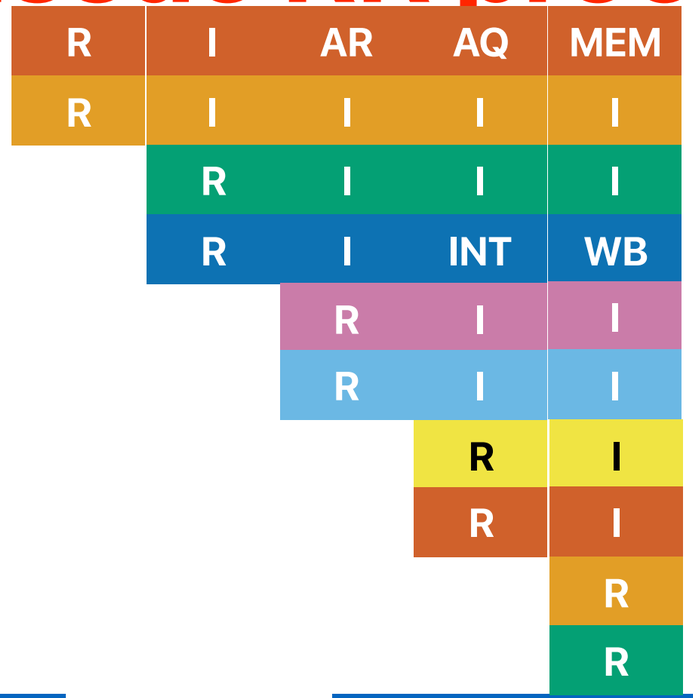
tail

	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)
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

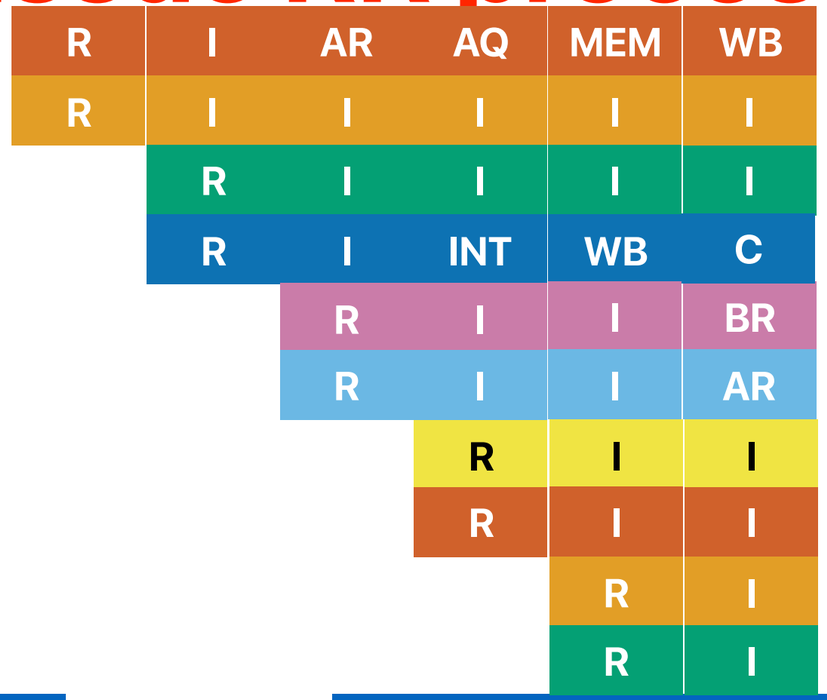
tail

	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

```
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
```



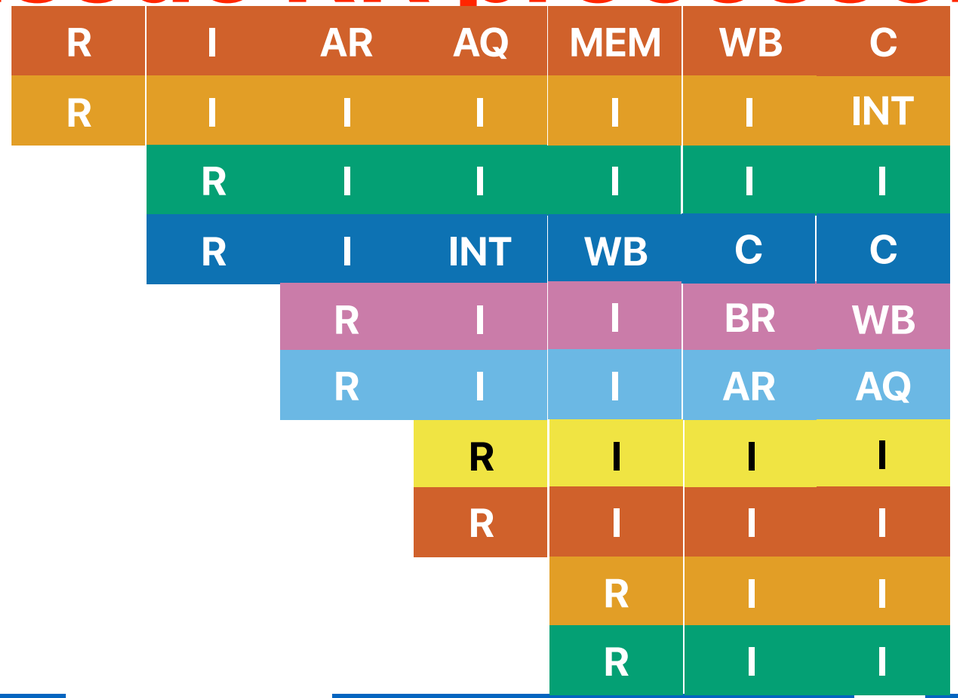
	Renamed instruction			head
1	ld	P1,	0(X10)	
2	add	P2,	P1, X12	
3	sd	P2,	0(X10)	
4	addi	P3,	X10, 8	
5	bne	P3,	X5, LOOP	
6	ld	P4,	0(P3)	
7	add	P5,	P1, X12	
8	sd	P5,	0(P3)	
9	addi	P6,	P3, 8	
10	bne	P6,	0(X10)	tail

	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

```
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

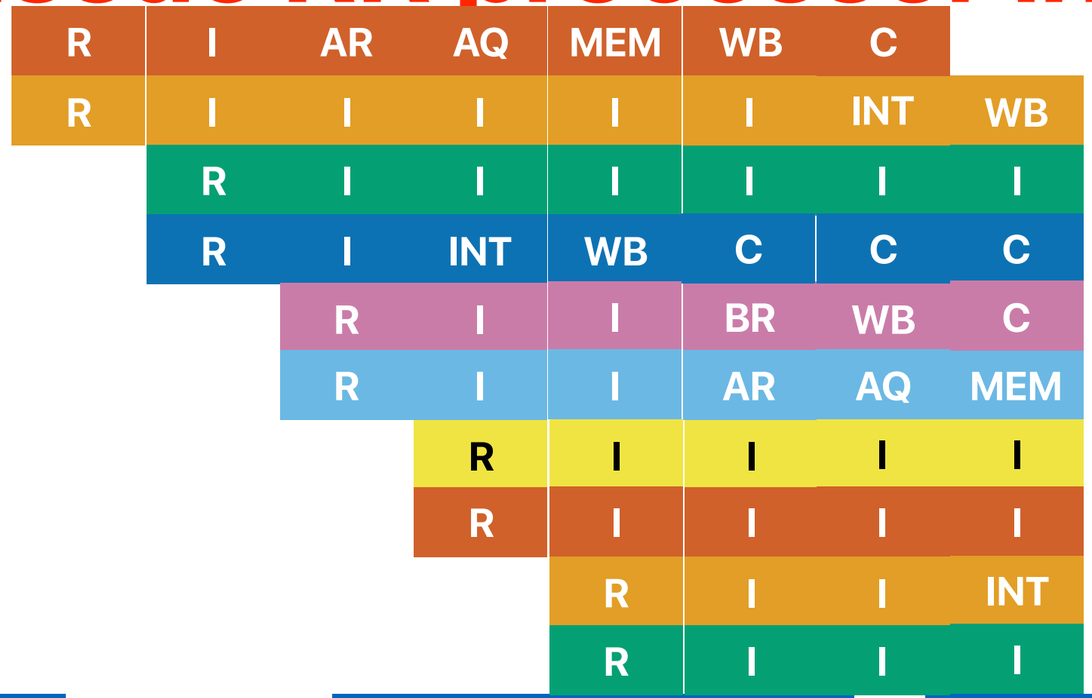
tail

	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

```
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
```



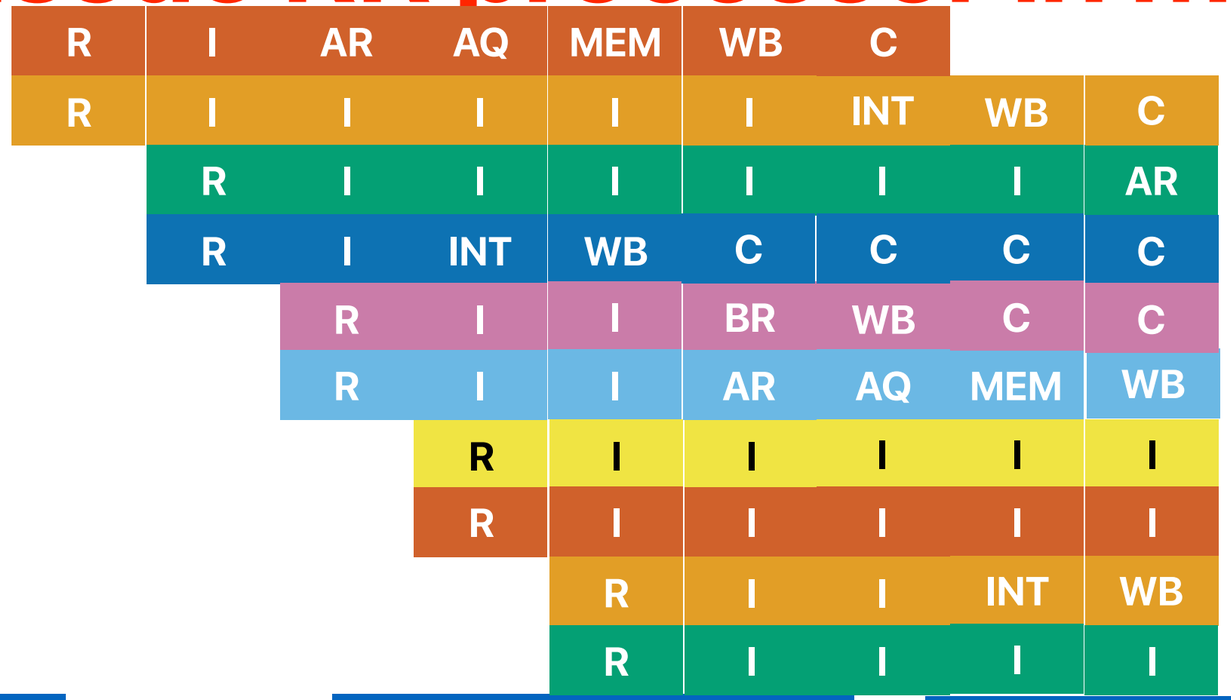
	Renamed instruction		
1	~~ld~~	~~P1, 0(X10)~~	
2	add	P2, P1, X12	← head
3	sd	P2, 0(X10)	
4	addi	P3, X10, 8	
5	bne	P3, X5, LOOP	
6	ld	P4, 0(P3)	
7	add	P5, P1, X12	
8	sd	P5, 0(P3)	
9	addi	P6, P3, 8	
10	bne	P6, 0(X10)	← tail

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6			
P2	1		1	P7			
P3	1		1	P8			
P4	0		1	P9			
P5	0		1	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	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

head

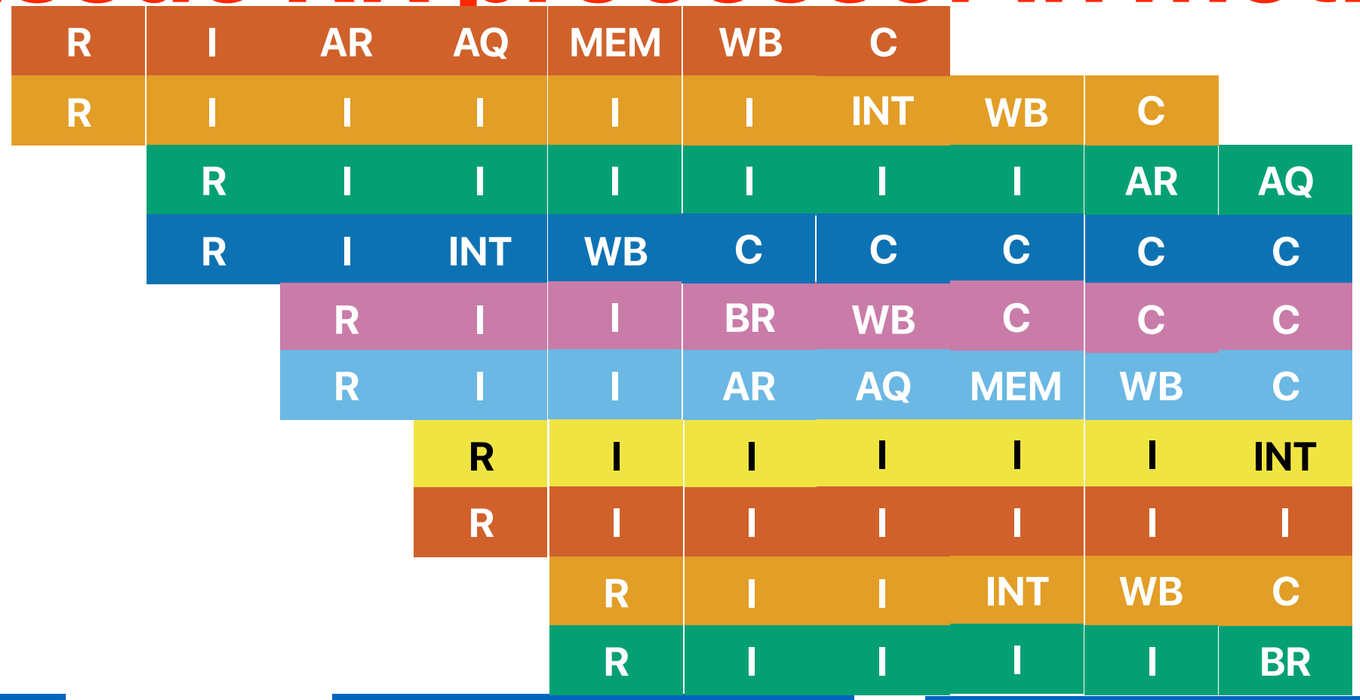
tail

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6			
P2	1		1	P7			
P3	1		1	P8			
P4	1		1	P9			
P5	0		1	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	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

head

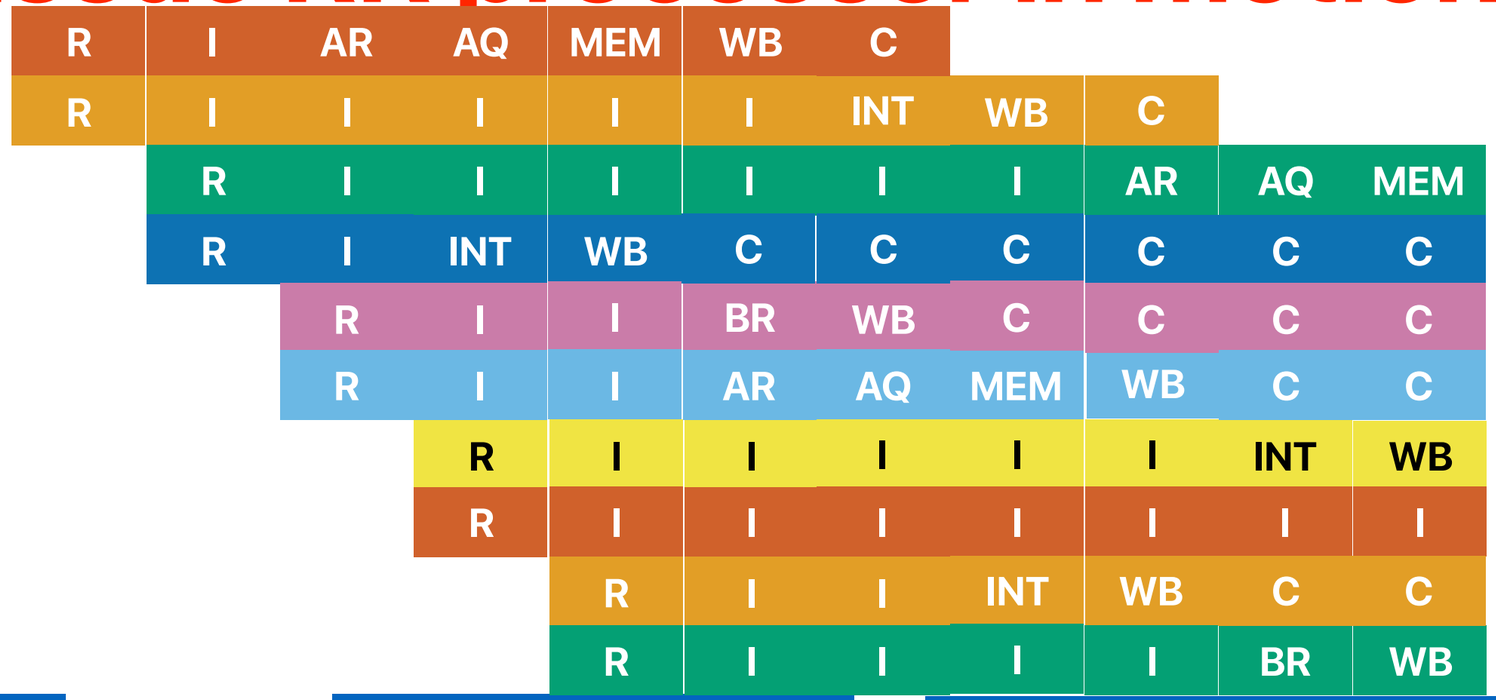
tail

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



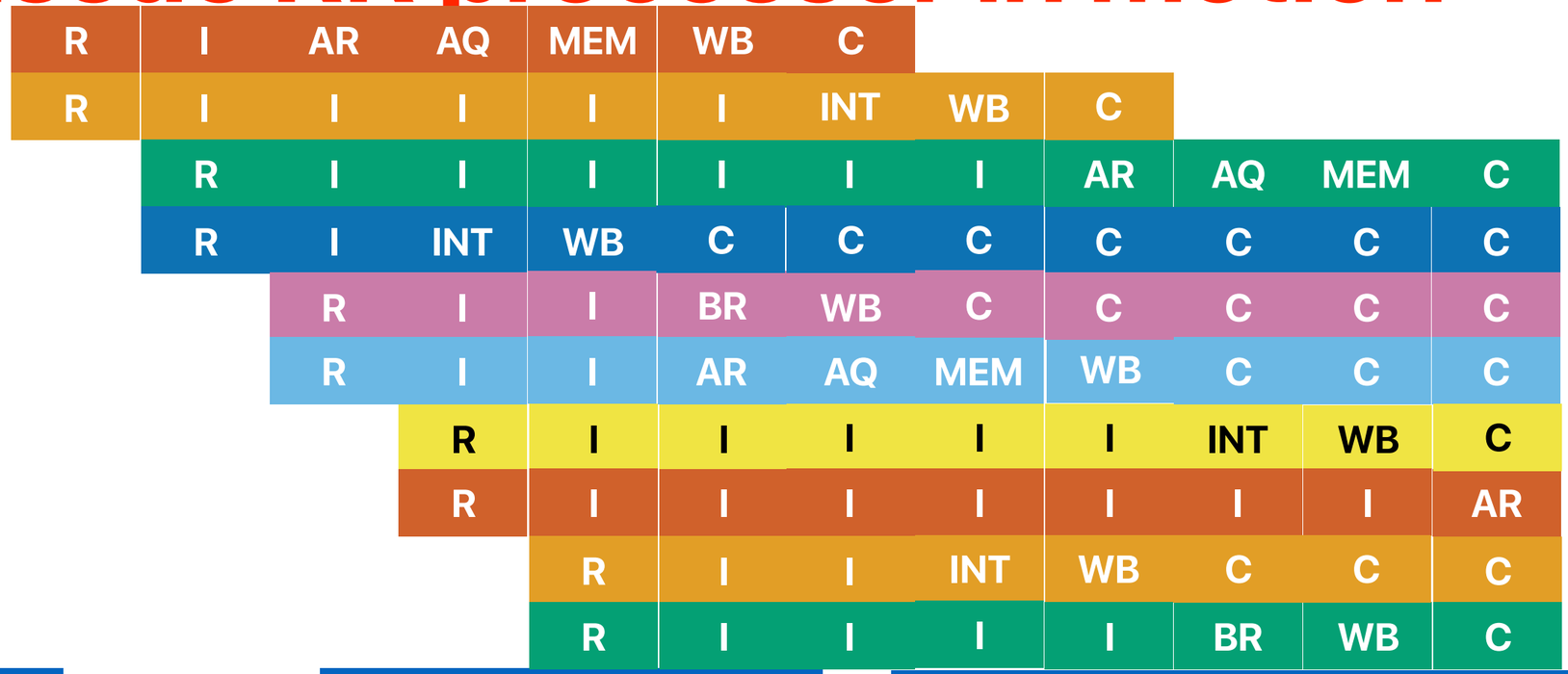
	Renamed instruction		
1	~~ld~~	~~P1, 0(X10)~~	
2	~~add~~	~~P2, P1, X12~~	
3	sd	P2, 0(X10)	← head
4	addi	P3, X10, 8	
5	bne	P3, X5, LOOP	
6	ld	P4, 0(P3)	
7	add	P5, P1, X12	
8	sd	P5, 0(P3)	
9	addi	P6, P3, 8	
10	bne	P6, 0(X10)	← tail

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



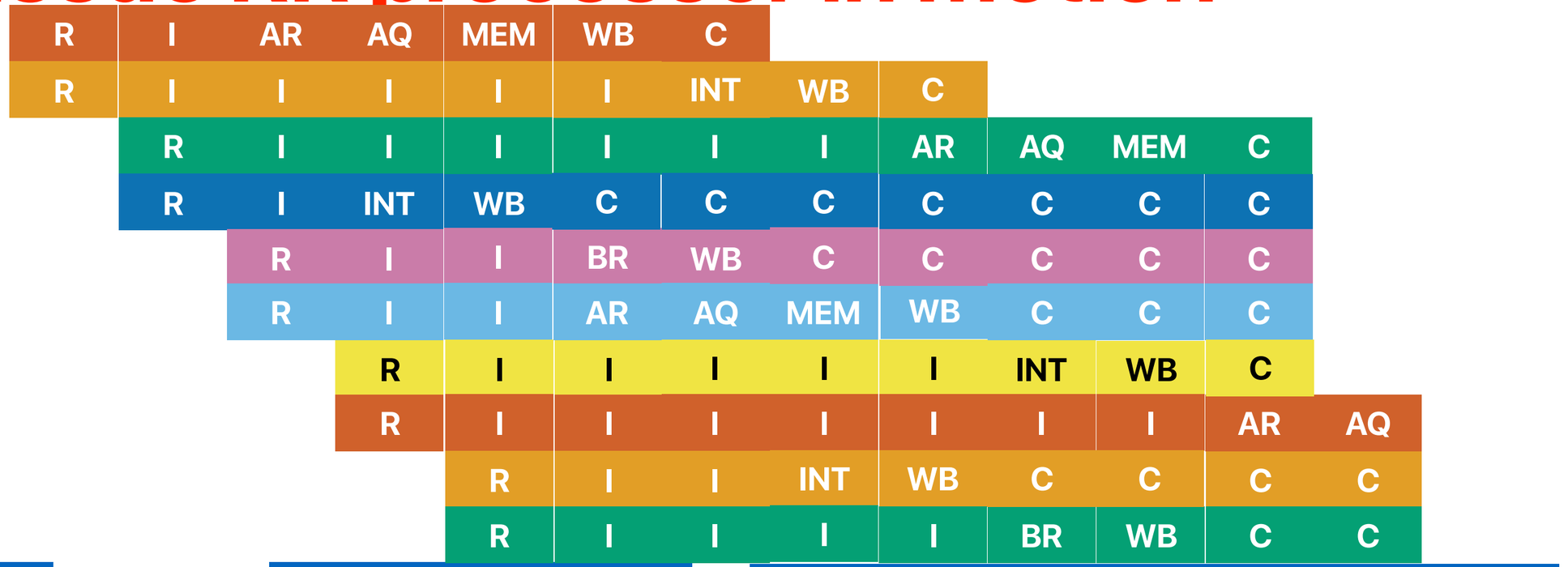
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	~~sd P2, 0(X10)~~
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	~~add P5, P1, X12~~
8	sd P5, 0(P3) ← head
9	addi P6, P3, 8
10	bne P6, 0(X10) ← tail

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



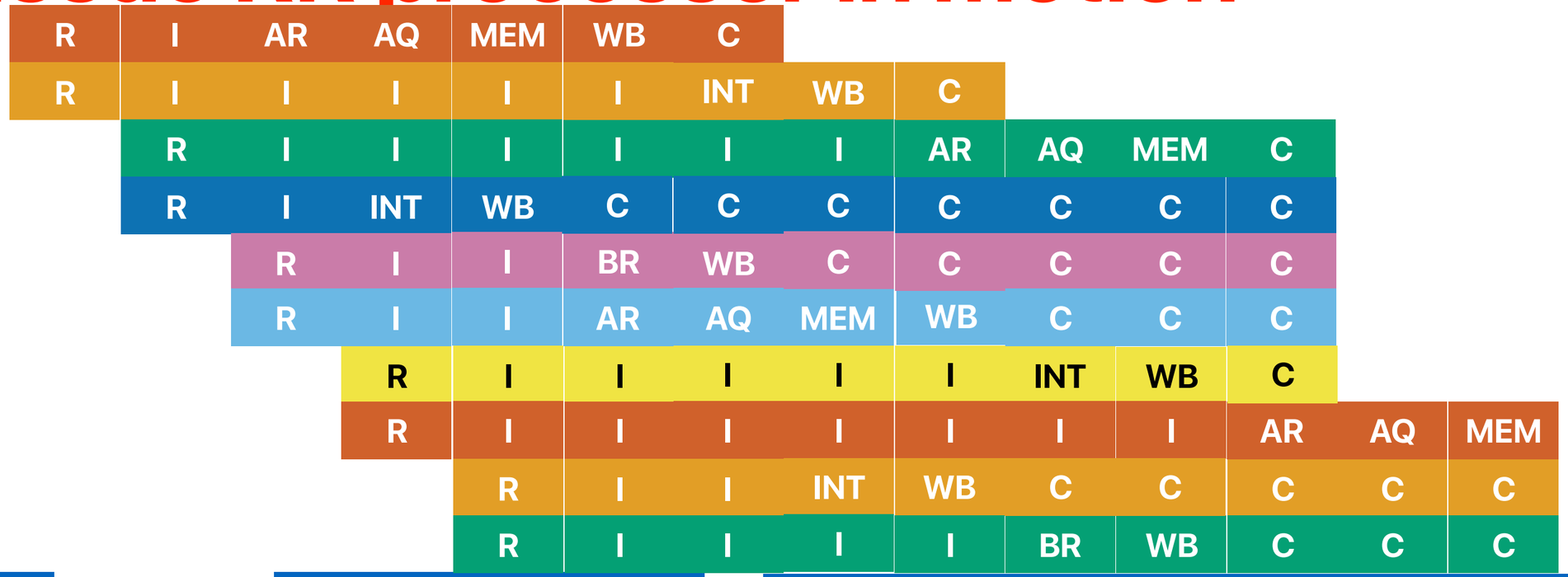
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	~~sd P2, 0(X10)~~
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	~~add P5, P1, X12~~
8	sd P5, 0(P3) ◀ head
9	addi P6, P3, 8
10	bne P6, 0(X10) ◀ tail

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

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



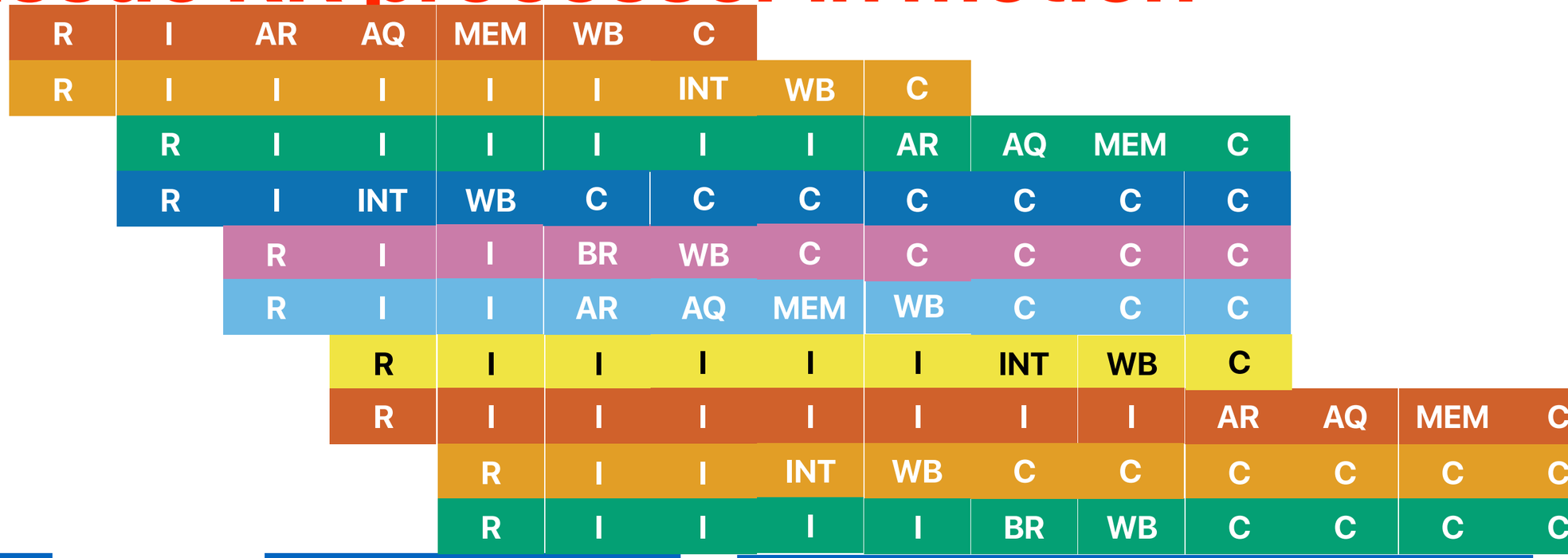
	Renamed instruction
1	~~ld P1, 0(X10)~~
2	~~add P2, P1, X12~~
3	~~sd P2, 0(X10)~~
4	~~addi P3, X10, 8~~
5	~~bne P3, X5, LOOP~~
6	~~ld P4, 0(P3)~~
7	~~add P5, P1, X12~~
8	sd P5, 0(P3) ← head
9	addi P6, P3, 8
10	bne P6, 0(X10) ← tail

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6			
P2	1		1	P7			
P3	1		1	P8			
P4	1		1	P9			
P5	1		1	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	~~add P5, P1, X12~~
8	~~sd P5, 0(P3)~~
9	~~addi P6, P3, 8~~
10	~~bne P6, 0(X10)~~

tail head

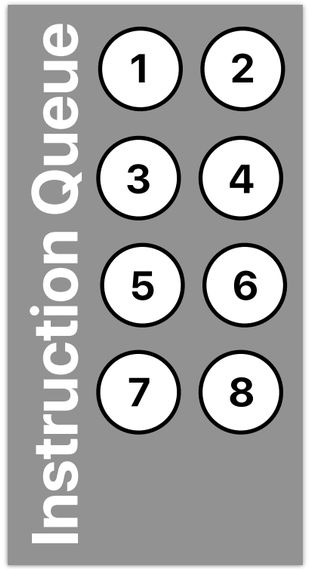
	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

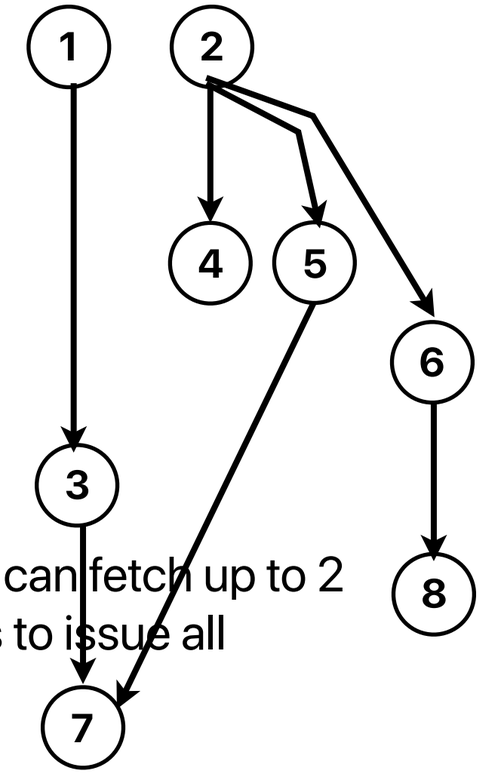
	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6			
P2	1		1	P7			
P3	1		1	P8			
P4	1		1	P9			
P5	1		1	P10			

How good is SS/OoO/ROB with this 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
⑤  ld   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 wi 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 & ur sical registers that can fetch up to 4 instructions per cycle, 3 cycles to exe ory instruction and the loop will execute for 10,000 times, what's the avera

A. 0.5

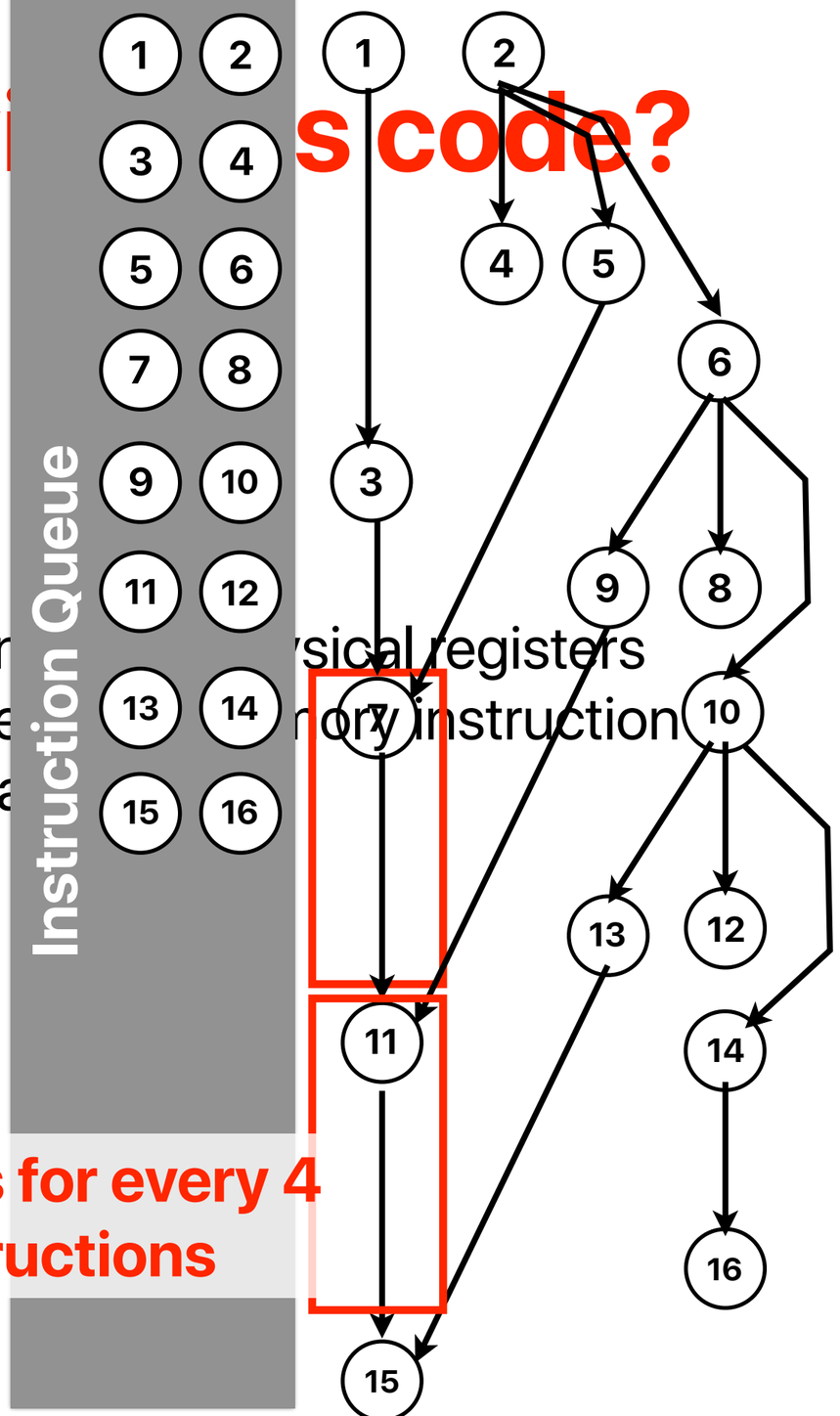
B. 0.75

C. 1

D. 1.25

E. 1.5

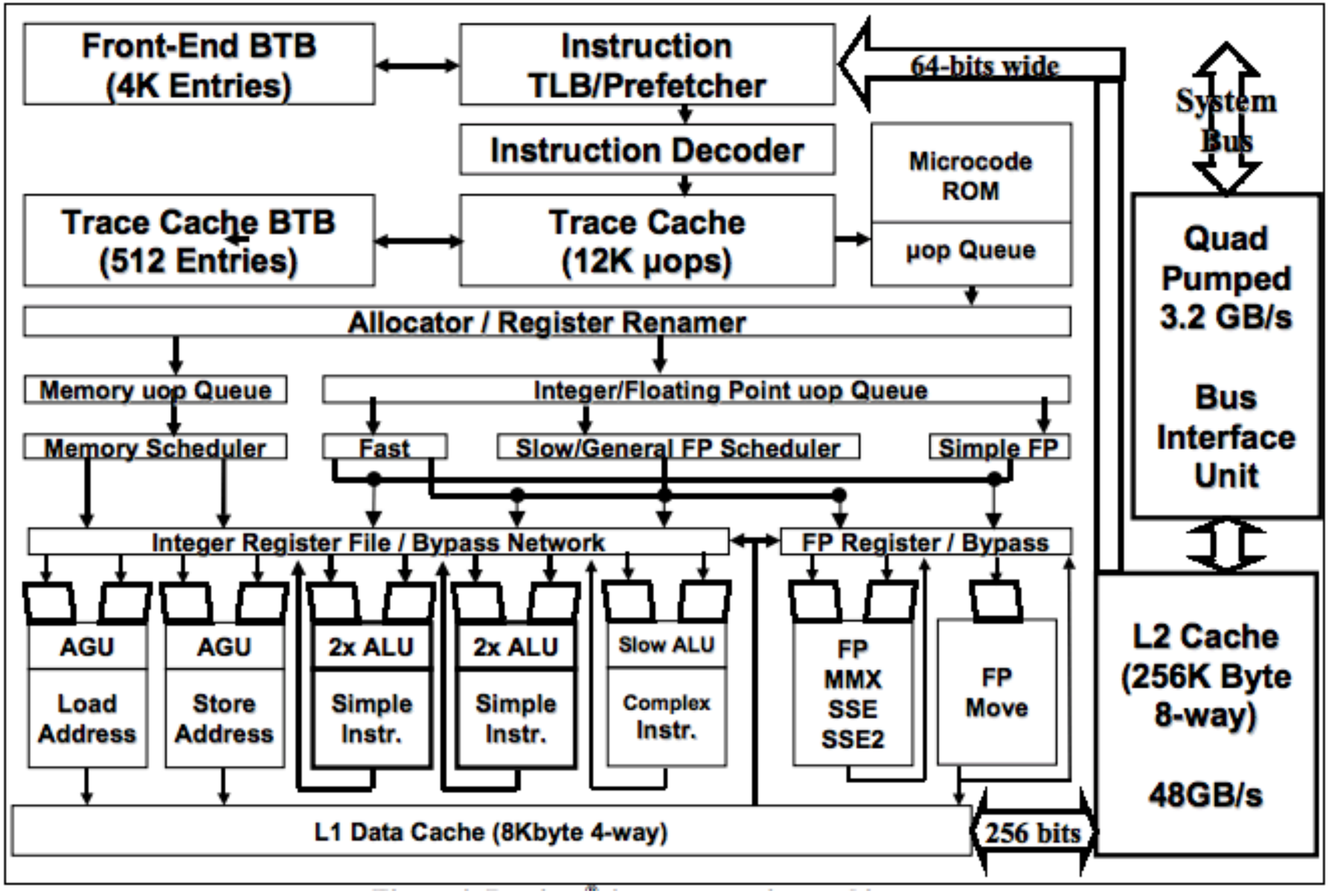
```
①  ld   X1, 0(X10)
②  addi X10, X10, 8
③  add  X20, X20, X1
④  bne  X10, X2, LOOP
⑤  ld   X1, 0(X10)
⑥  addi X10, X10, 8
⑦  add  X20, X20, X1
⑧  bne  X10, X2, LOOP
⑨  ld   X1, 0(X10)
⑩  addi X10, X10, 8
⑪  add  X20, X20, X1
⑫  bne  X10, X2, LOOP
```



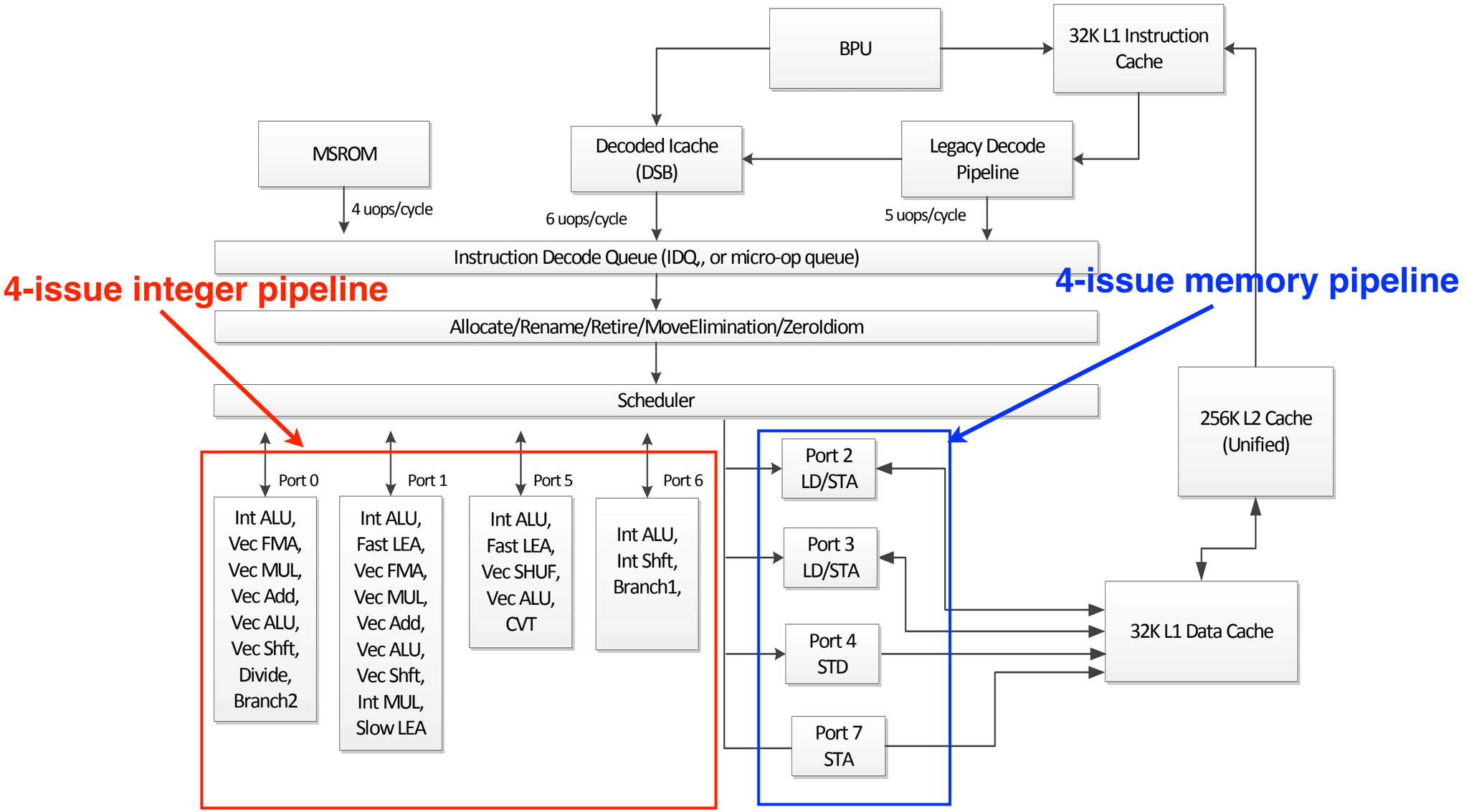
3 cycles for every 4 instructions

The pipelines of Modern Processors

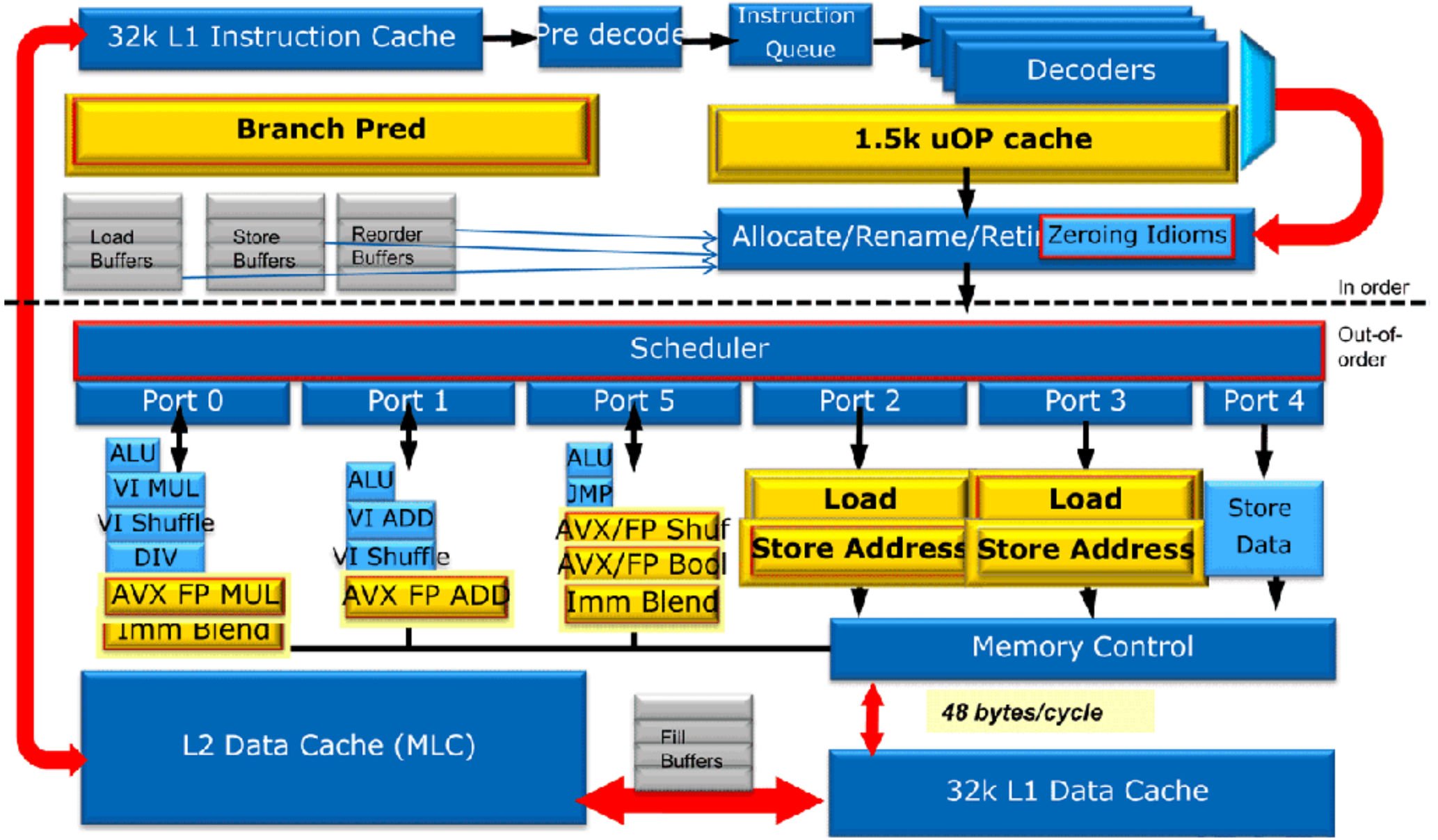
Intel Pentium 4

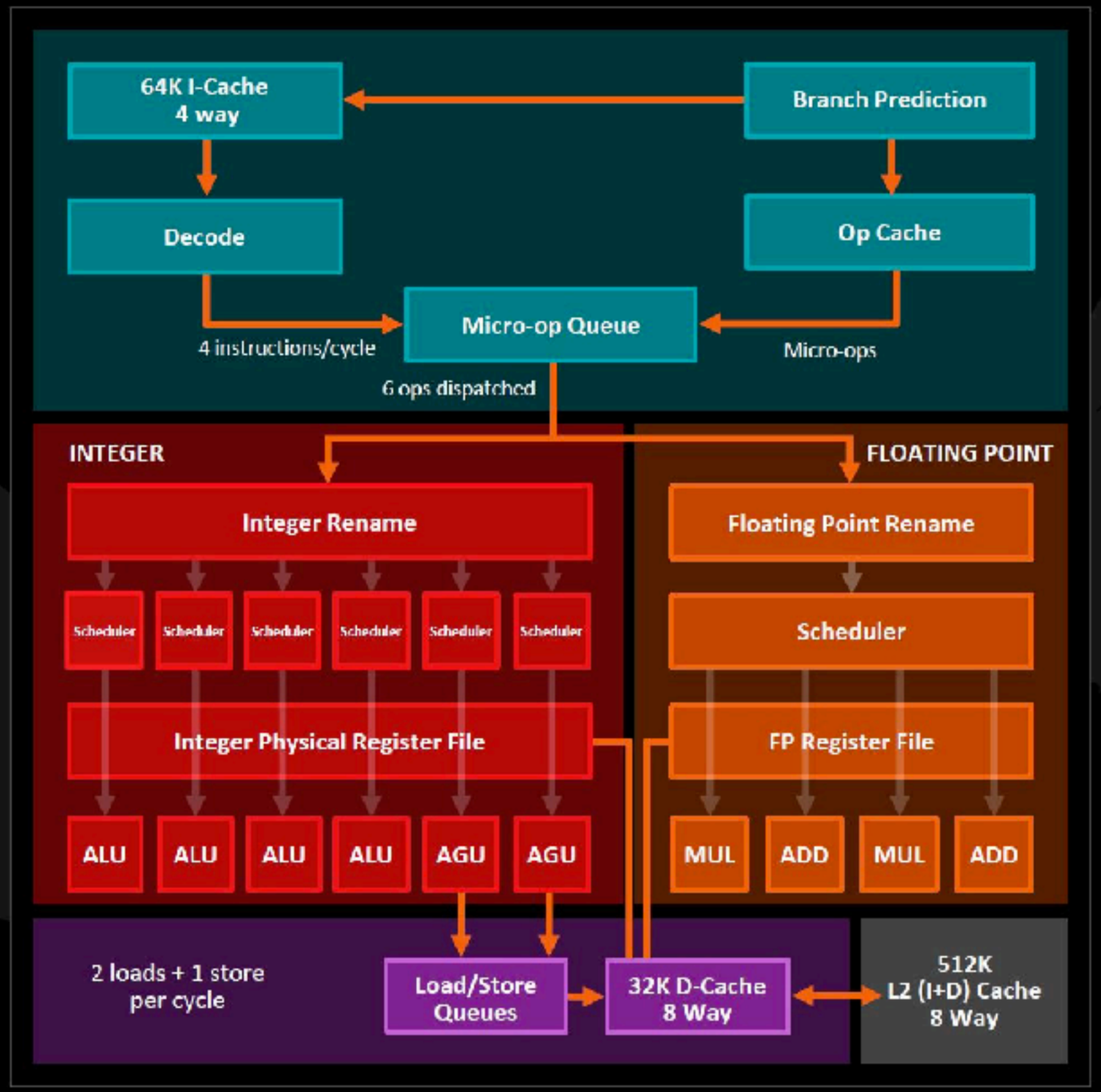


Intel Skylake



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

Announcement

- Reading quiz due next Monday
- Homework #4 due 12/4
- iEval submission — attach your "confirmation" screen, you get an extra/bonus homework
- Project due on 12/2
  - You can only turn-in "helper.c"
    - `mcfutil.c:refresh_potential()` creates helper threads
    - `mcfutil.c:refresh_potential()` calls `helper_thread_sync()` function periodically
    - It's your task to think what to do in `helper_thread_sync()` and `helper_thread()` functions
    - Please DO READ papers before you ask what to do
  - Formula for grading — **min(100, speedup*100)**
  - No extension
- Office hour for Hung-Wei **next** week — MWF 1p-2p — no office hour this week