

Data Hazards & Dynamic Instruction Scheduling (I)

Hung-Wei Tseng

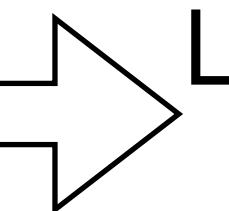
Data hazards

Data hazards

- An instruction currently in the pipeline cannot receive the “logically” correct value for execution
- Data dependencies
 - The output of an instruction is the input of a later instruction
 - May result in data hazard if the later instruction that consumes the result is still in the pipeline

Example: vector scaling

```
i = 0;  
do {  
    vector[i] += scale;  
} while ( ++i < size )
```



LOOP:

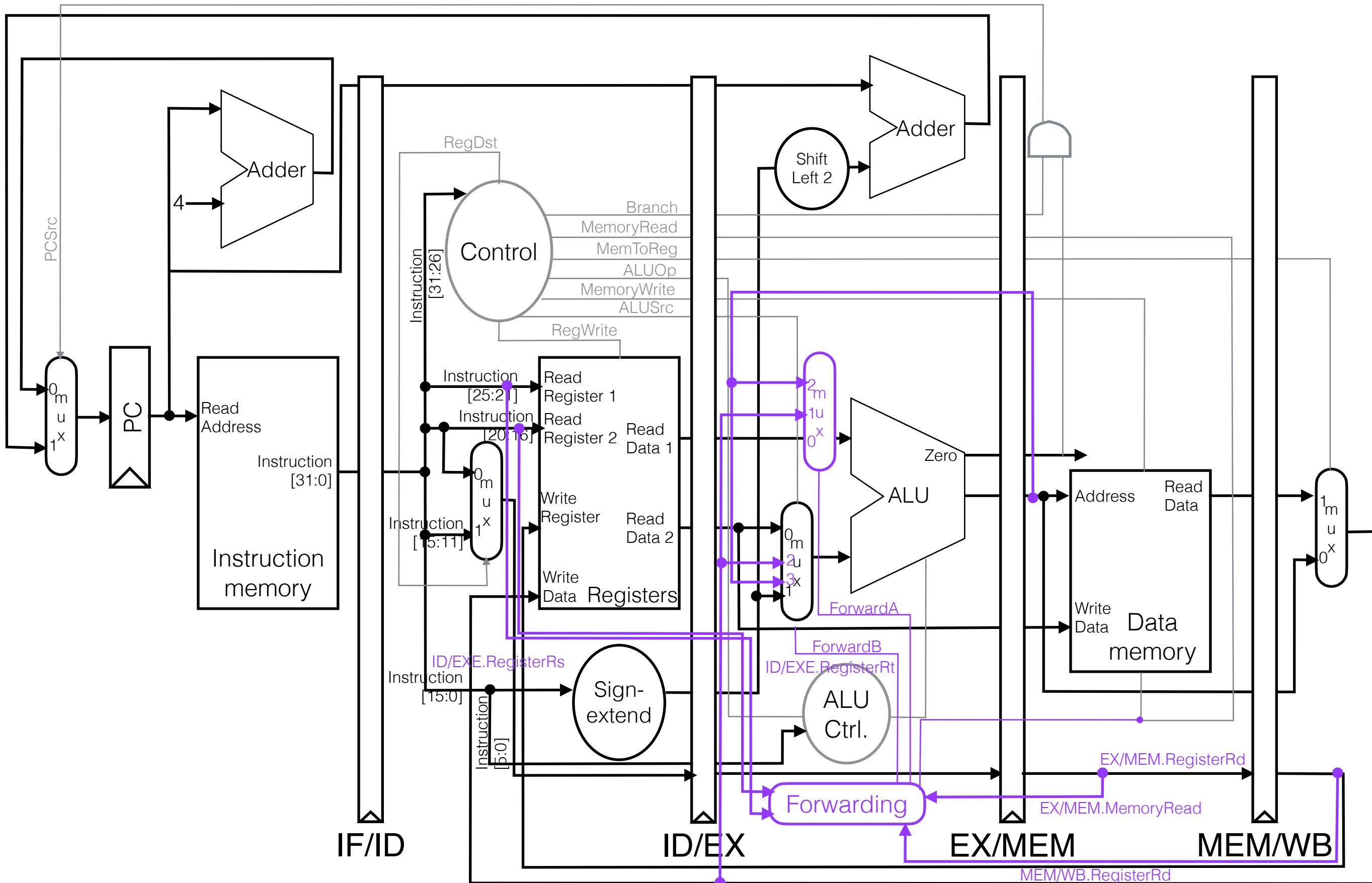
shl	X5, X11, 3
add	X5, X5, X10
ld	X6, 0(X10)
add	X7, X6, X12
sd	X7, 0(X10)
addi	X10, X10, 8
bne	X10, X5, LOOP

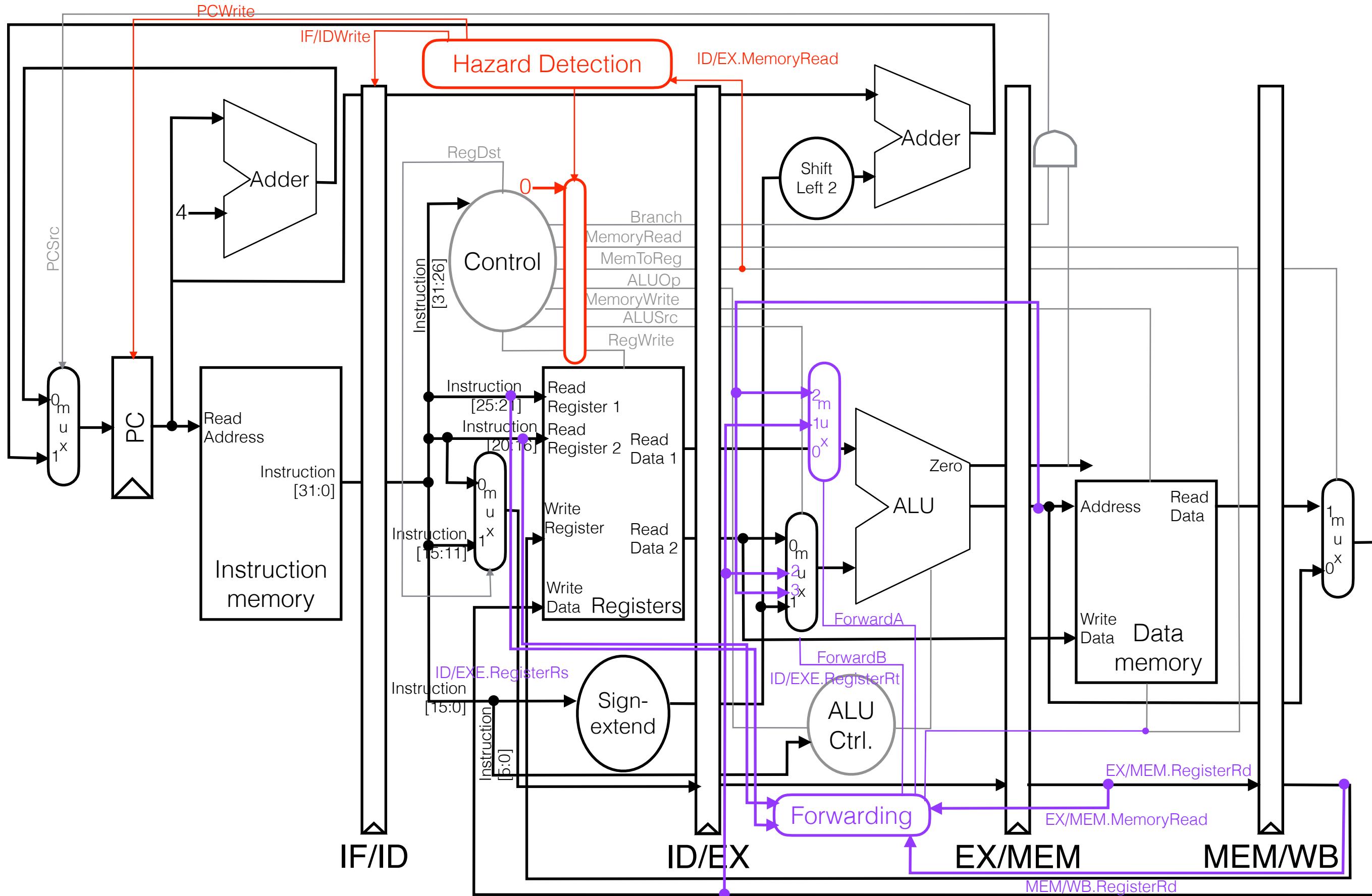
Solution 1: Let's try “stall” again

- Whenever the input is not ready when the consumer is decoding, just stall — the consumer stays at ID.

Solution 2: Data forwarding

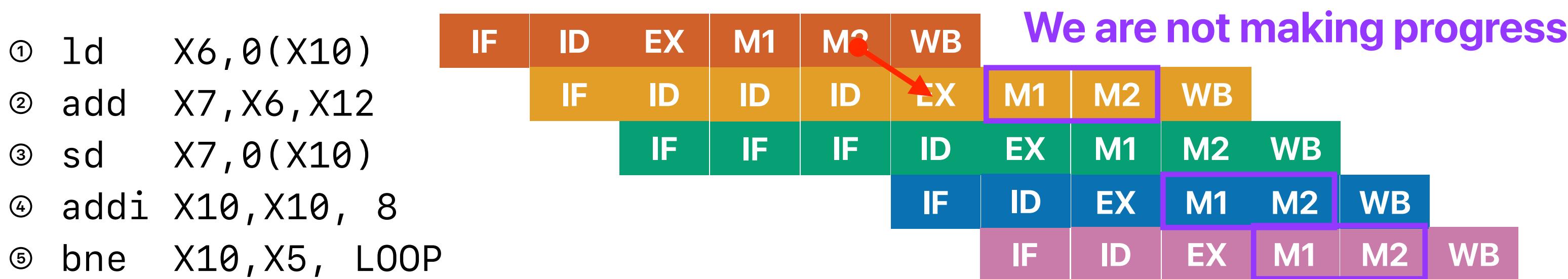
- Add logics/wires to forward the desired values to the demanding instructions
- In our five stage pipeline — if the instruction entering the EXE stage consumes a result from a previous instruction that is entering MEM stage or WB stage
 - A source of the instruction entering EXE stage is the destination of an instruction entering MEM/WB stage
 - The previous instruction must be an instruction that updates register file





Problems with data forwarding

- What if our pipeline gets deeper? — Considering a newly designed pipeline where memory stage is split into 2 stages and the memory access finishes at the 2nd memory stage. By reordering which pair of the following instruction stream can we eliminate all stalls without affecting the correctness of the code?



Dynamic instruction scheduling/ Out-of-order (OoO) execution

Tips of drawing a pipeline diagram

- Each instruction has to go through all 5 pipeline stages: IF, ID, EXE, MEM, WB in order – only valid if it's single-issue, RISC-V 5-stage pipeline
- An instruction can enter the next pipeline stage in the next cycle if
 - No other instruction is occupying the next stage
 - This instruction has completed its own work in the current stage
 - The next stage has all its inputs ready
- Fetch a new instruction only if
 - We know the next PC to fetch
 - We can predict the next PC
 - Flush an instruction if the branch resolution says it's mis-predicted.

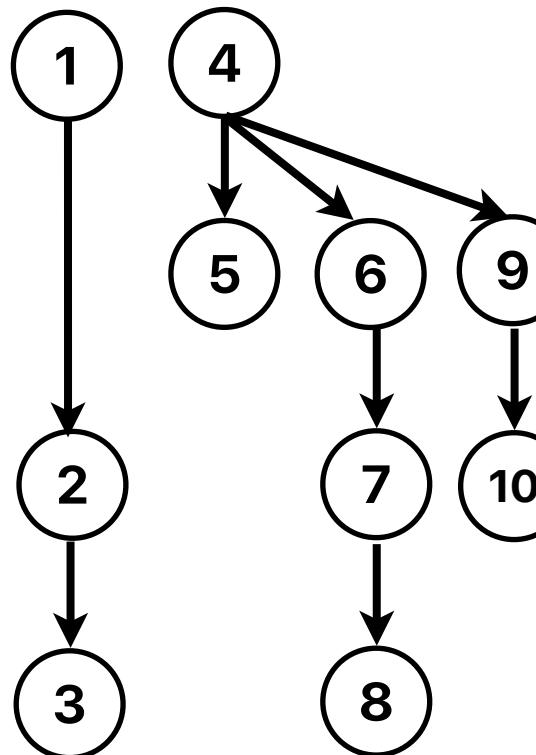
What do you need to execution an instruction?

- 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

Scheduling instructions: based on data dependencies

- Draw the data dependency graph, put an arrow if an instruction depends on the other.

- 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

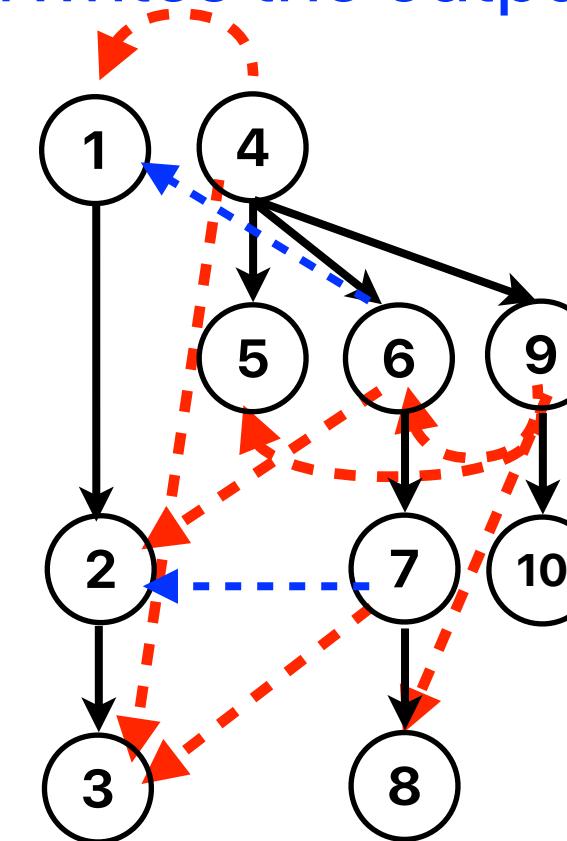


- **In theory**, instructions without dependencies can be executed in parallel or out-of-order
- Instructions with dependencies can never be reordered

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



Out-of-order execution

- Any sequence of instructions has set of RAW, WAW, and WAR hazards that constrain its execution.
- Can we design a processor that extracts as much parallelism as possible, while still respecting these dependences?



Tomasulo's Algorithm

IBM 360 Model 91

- The most powerful commercialized machine in 1968

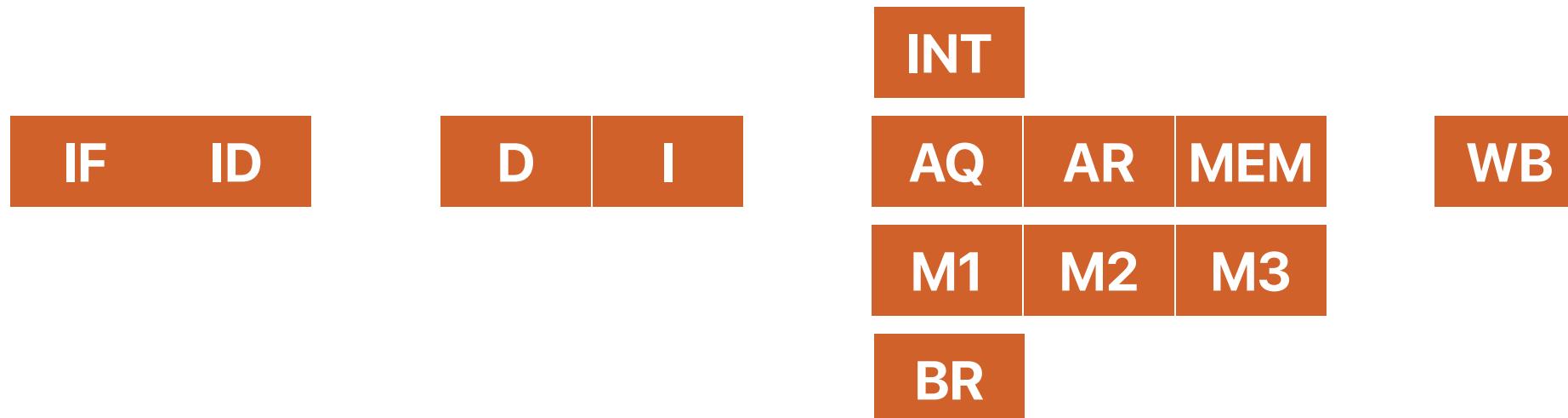


CDC 6600

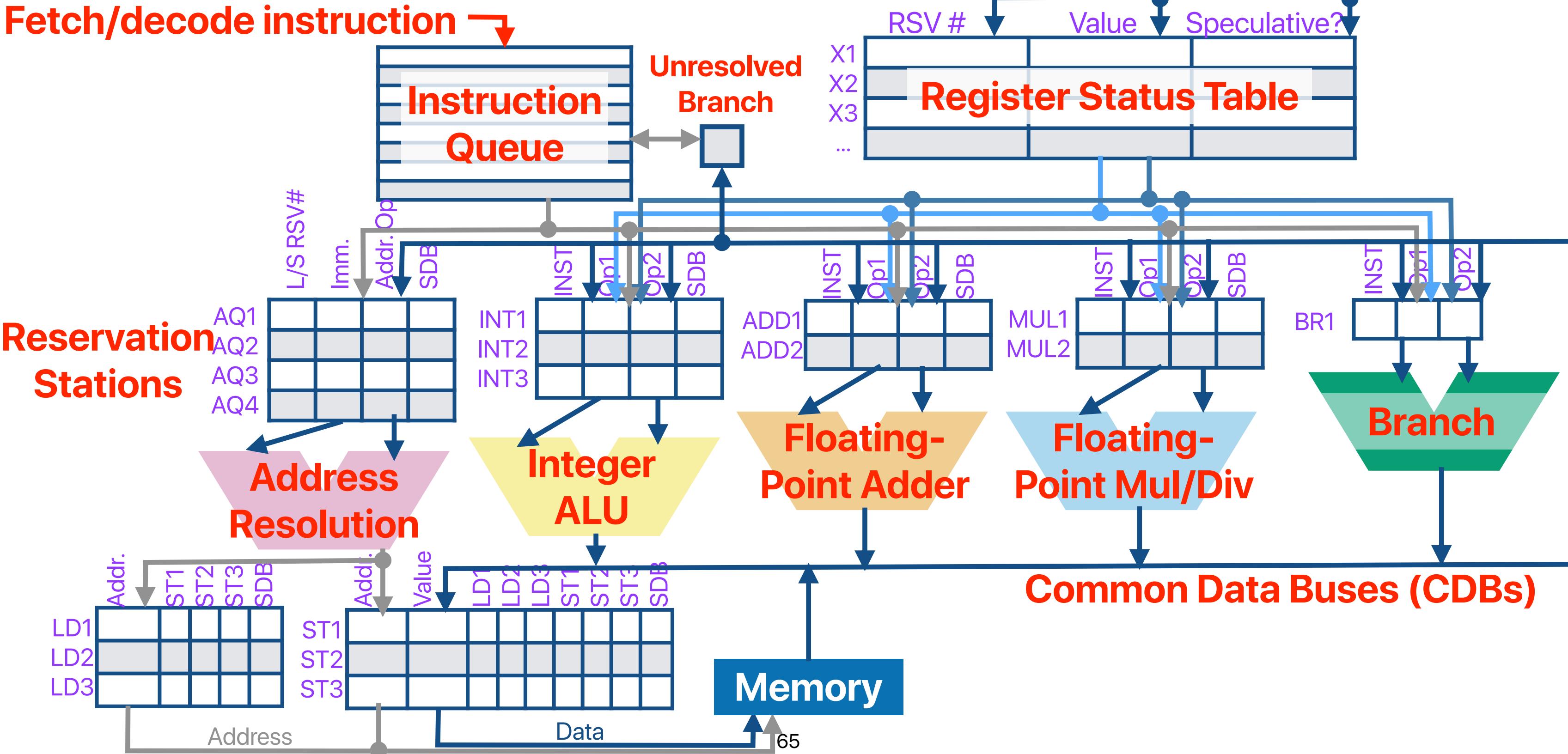
- The first machine with out-of-order execution.
- With performance of up to 3 "M" FLOPS
- It uses score-boarding to achieve so.
 - Instruction storage added to each functional execution unit
 - Instructions issue to FU when no structural hazards, begin execution when dependences satisfied. Thus, instructions issued to different FUs can execute out of order.
 - "scoreboard" tracks RAW, WAR, WAW hazards, tells each instruction when to proceed.
 - No forwarding
 - No register renaming

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

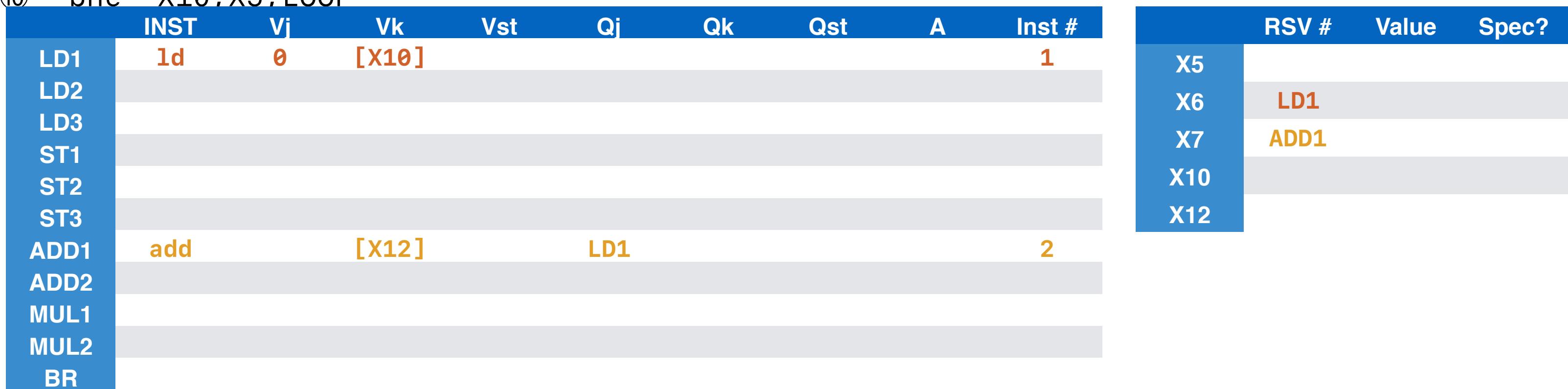
- ① ld X6, 0(X10)
 - ② add X7, X6, X12
 - ③ sd X7, 0(X10)
 - ④ addi X10, X10, 8
 - ⑤ bne X10, X5, LOOP
 - ⑥ ld X6, 0(X10)
 - ⑦ add X7, X6, X12
 - ⑧ sd X7, 0(X10)
 - ⑨ addi X10, X10, 8
 - ⑩ bne X10, X5, LOOP

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

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



Tomasulo in motion

①	ld	X6, 0(X10)	D	AQ	AR
②	add	X7, X6, X12		D	I
③	sd	X7, 0(X10)			D

- 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

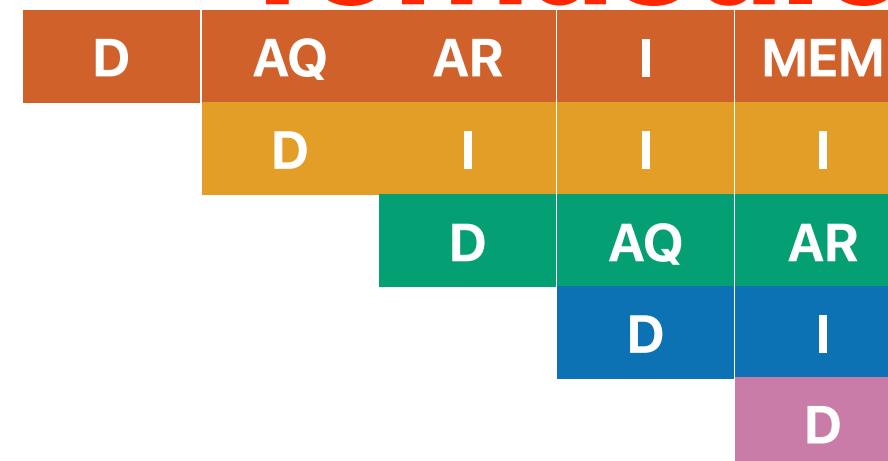
Tomasulo in motion

			D	AQ	AR	I
②	add	X7,X6,X12		D	I	I
③	sd	X7,0(X10)		D		AQ
④	addi	X10,X10,8				D

- 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

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

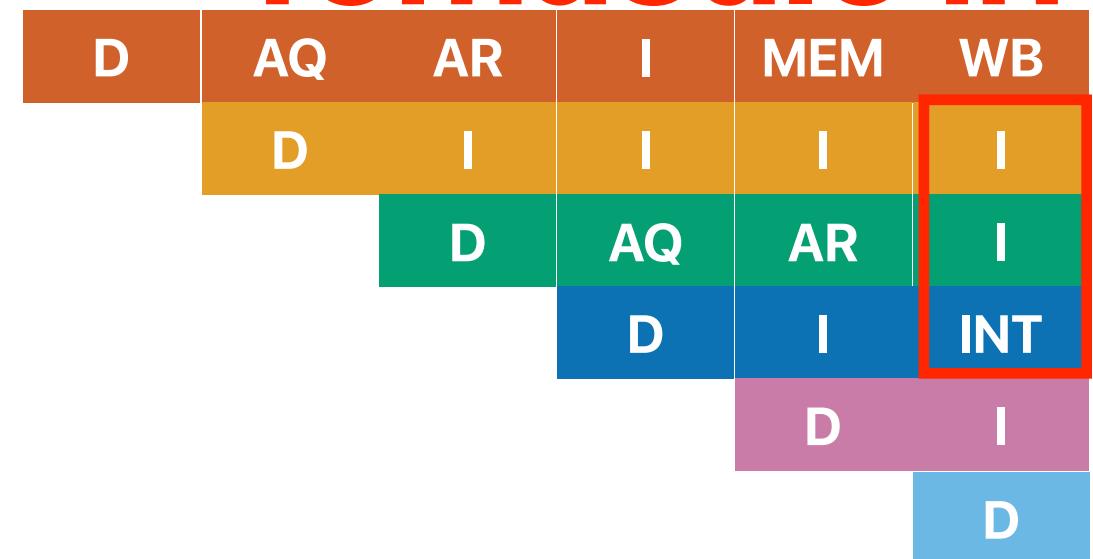


	INST	Vj	Vk	Vst	Qj	Qk	Qst	A	Inst #
LD1	ld	0	[X10]						1
LD2									
LD3									
ST1	sd	0	[X10]				ADD1		3
ST2									
ST3									
ADD1	add		[X12]		LD1				2
ADD2	addi	8	[X10]						4
MUL1									
MUL2									
BR	bne		[X5]		ADD2				5

RSV #	Value	Spec?
X5		
X6	LD1	
X7	ADD1	
X10	ADD2	
X12		

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



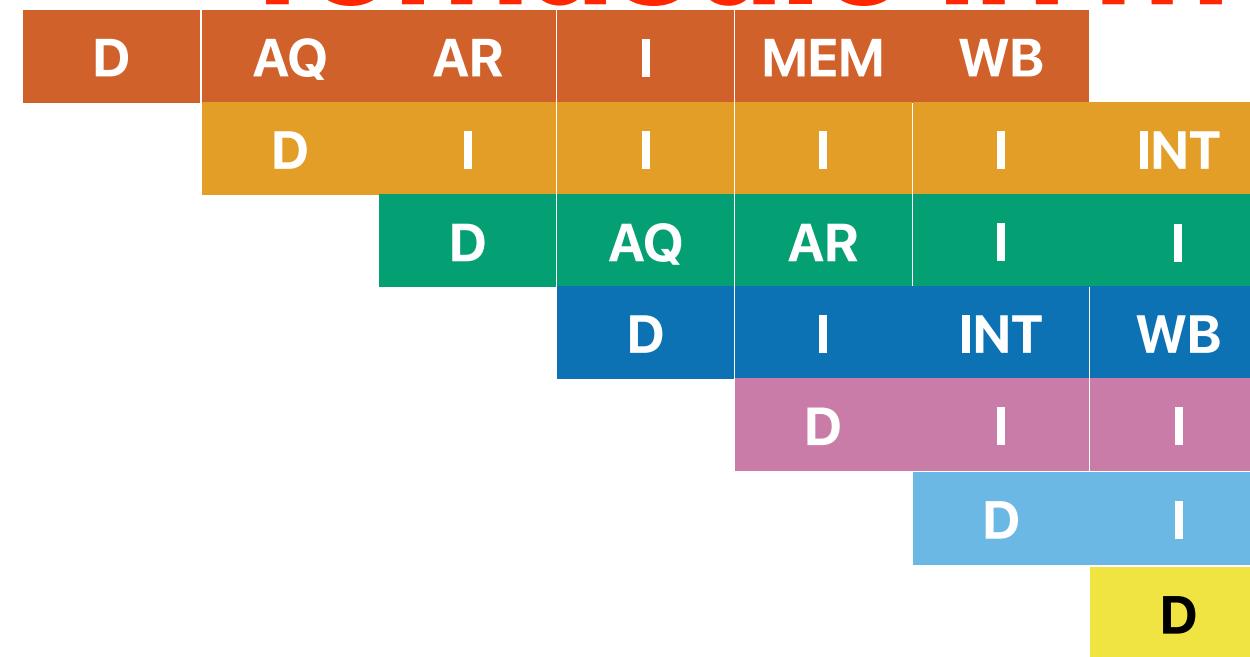
addi is now ahead of **add!**

	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									
ST3									
ADD1	add	LD1	[X12]						2
ADD2	addi	8	[X10]						4
MUL1									
MUL2									
BR	bne		[X5]		ADD2				5

RSV #	Value	Spec?
X5		
X6	LD2	LD1 1
X7	ADD1	
X10	ADD2	
X12		

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

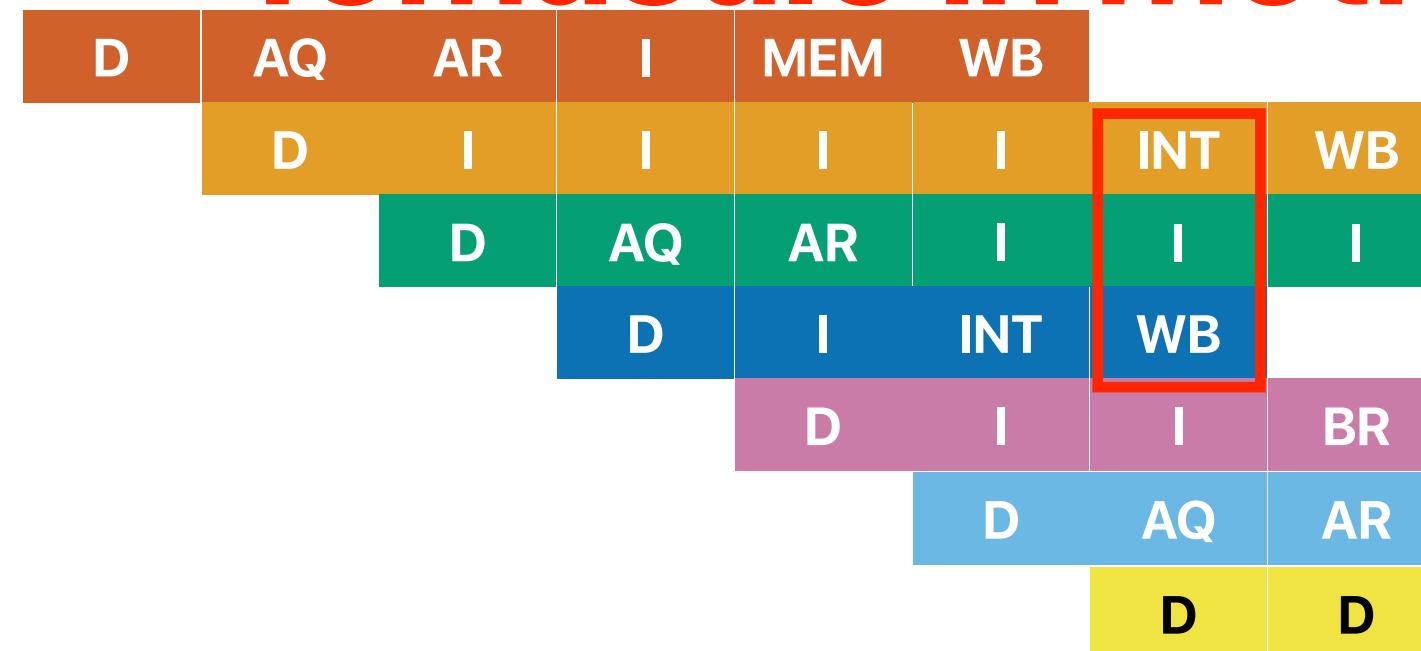


	INST	Vj	Vk	Vst	Qj	Qk	Qst	A	Inst #
LD1	ld	0	[X10]					1	
LD2	ld	0	ADD2					6	
LD3									
ST1	sd	0	[X10]				ADD1	3	
ST2									
ST3									
ADD1	add	LD1	[X12]					2	
ADD2	addi	8	[X10]					4	
MUL1									
MUL2									
BR	bne	ADD2	[X5]						5

RSV #	Value	Spec?
X5		
X6	LD2	1
X7	ADD1	
X10	ADD2	
X12		

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



addi is finished
ahead of add and sd!

	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									
ST3									
ADD1	add	LD1	[X12]					2	
ADD2	add		[X12]		[LD2]			7	
MUL1									
MUL2									
BR	bne	ADD2	[X5]						

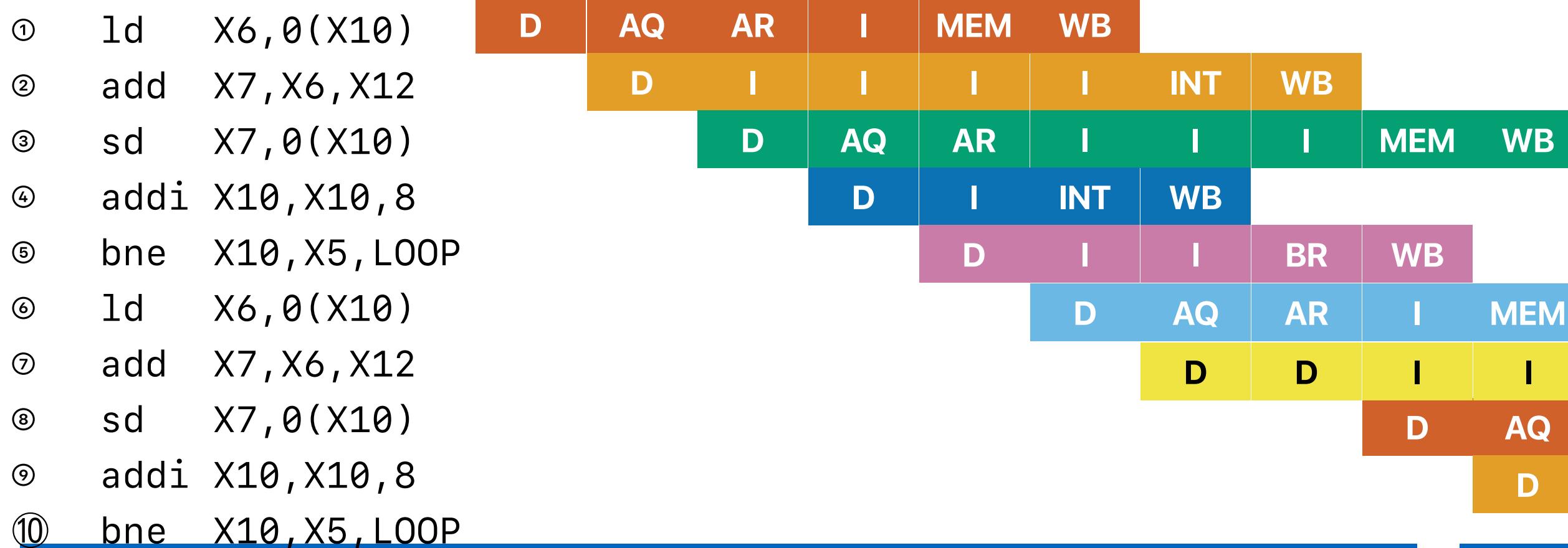
	RSV #	Value	Spec?
X5			
X6	LD2		1
X7	ADD2	ADD1	1
X10			
X12			

Tomasulo in motion

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

	INST	Vj	Vk	Vst	Qj	Qk	Qst	A	Inst #	RSV #	Value	Spec?
LD1	ld	0	[X10]						1	X5		
LD2	ld	0	ADD2						6	X6	LD2	
LD3										X7	ADD2	
ST1	sd	0	[X10]	ADD1					3	X10		ADD2
ST2	sd	0	[X10]				ADD2		8	X12		
ST3												
ADD1	add	LD1	[X12]						2			
ADD2	add		[X12]		[LD2]				7			
MUL1												
MUL2												
BR	bne	ADD2	[X5]						5			

Tomasulo in motion



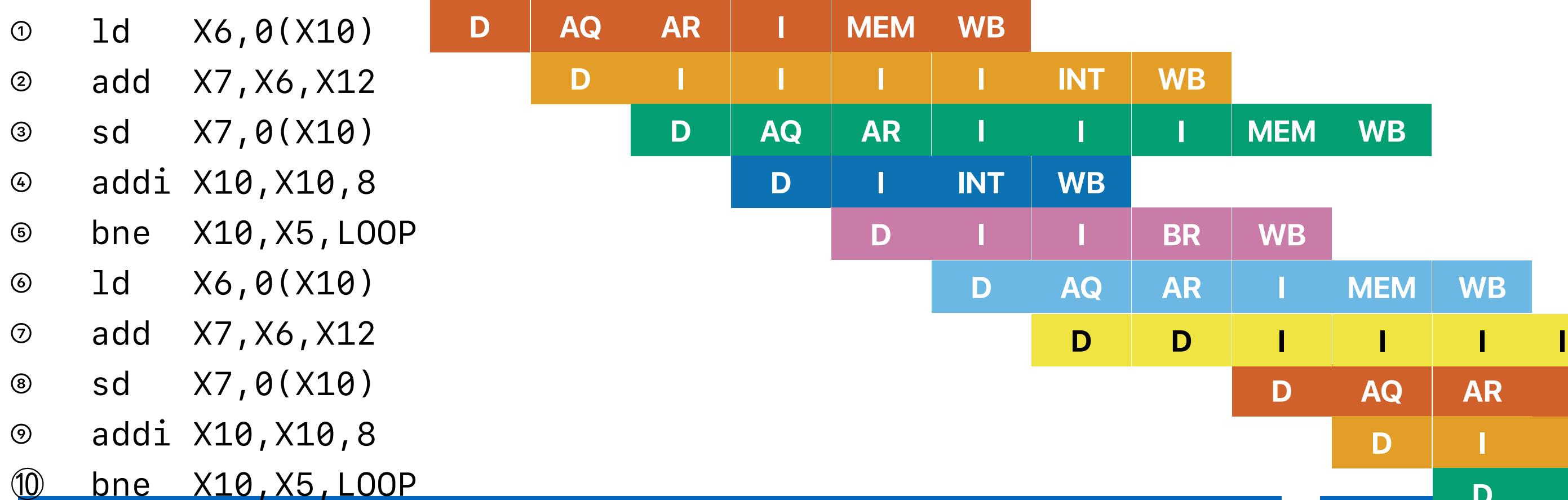
	INST	Vj	Vk	Vst	Qj	Qk	Qst	A	Inst #	RSV #	Value	Spec?
LD1	ld	0	[X10]						1	X5		
LD2	ld	0	ADD2						6	X6	LD2	
LD3										X7	ADD2	
ST1	sd	0	[X10]	ADD1					3	X10	ADD1	
ST2	sd	0	[X10]				ADD2		8	X12		
ST3												
ADD1	add	8	ADD2						9			
ADD2	add		[X12]		[LD2]				7			
MUL1												
MUL2												
BR	bne	ADD2	[X5]						5			

Tomasulo in motion

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

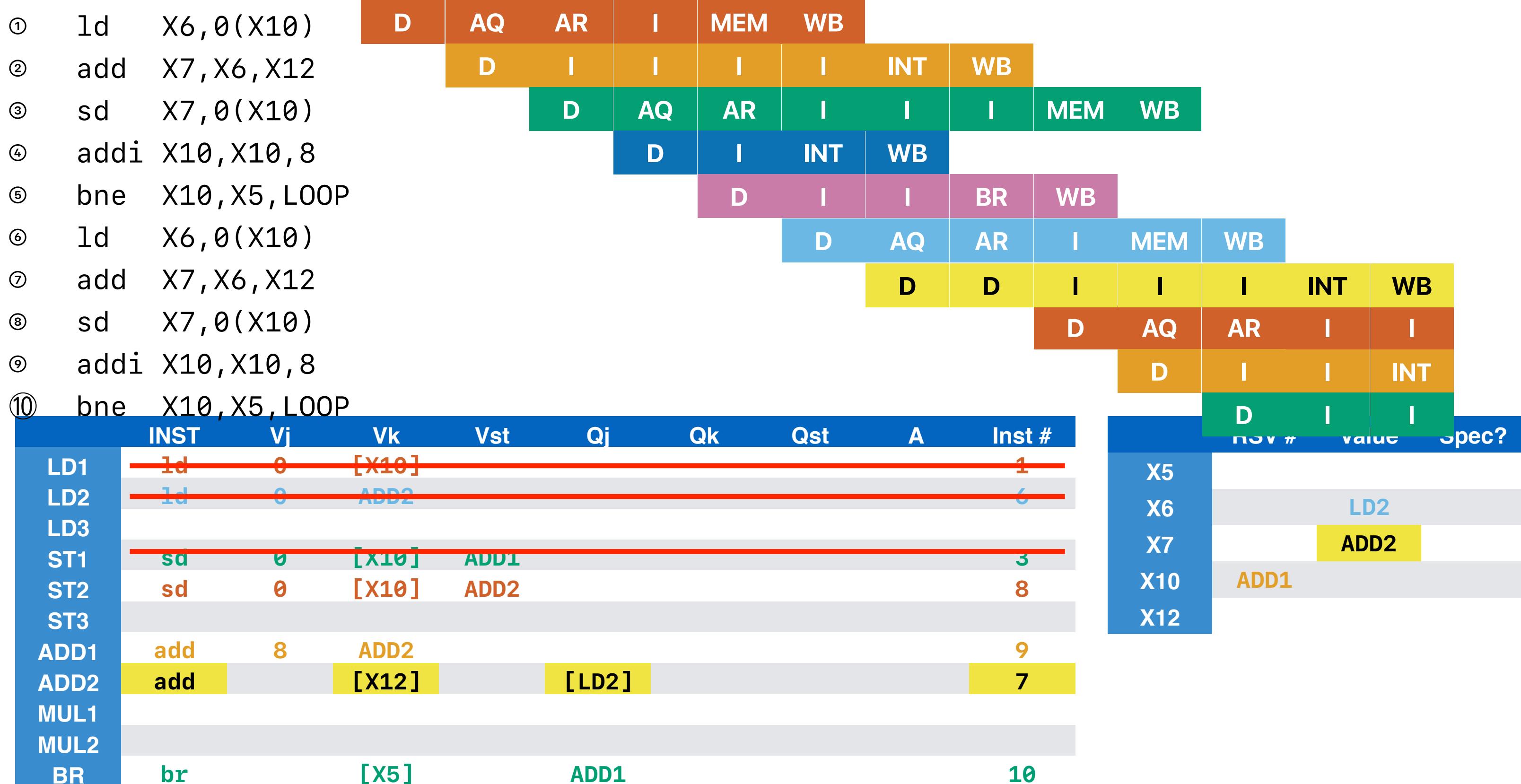
	INST	Vj	Vk	Vst	Qj	Qk	Qst	A	Inst #	Resv #	Value	Spec?
LD1	ld	0	[X10]						1		X5	
LD2	ld	0	ADD2						6		X6	LD2
LD3											X7	ADD2
ST1	sd	0	[X10]	ADD1					3		X10	ADD1
ST2	sd	0	[X10]				ADD2		8		X12	
ST3												
ADD1	add	8	ADD2						9			
ADD2	add		[X12]		[LD2]				7			
MUL1												
MUL2												
BR	br		[X5]		ADD1				10			

Tomasulo in motion

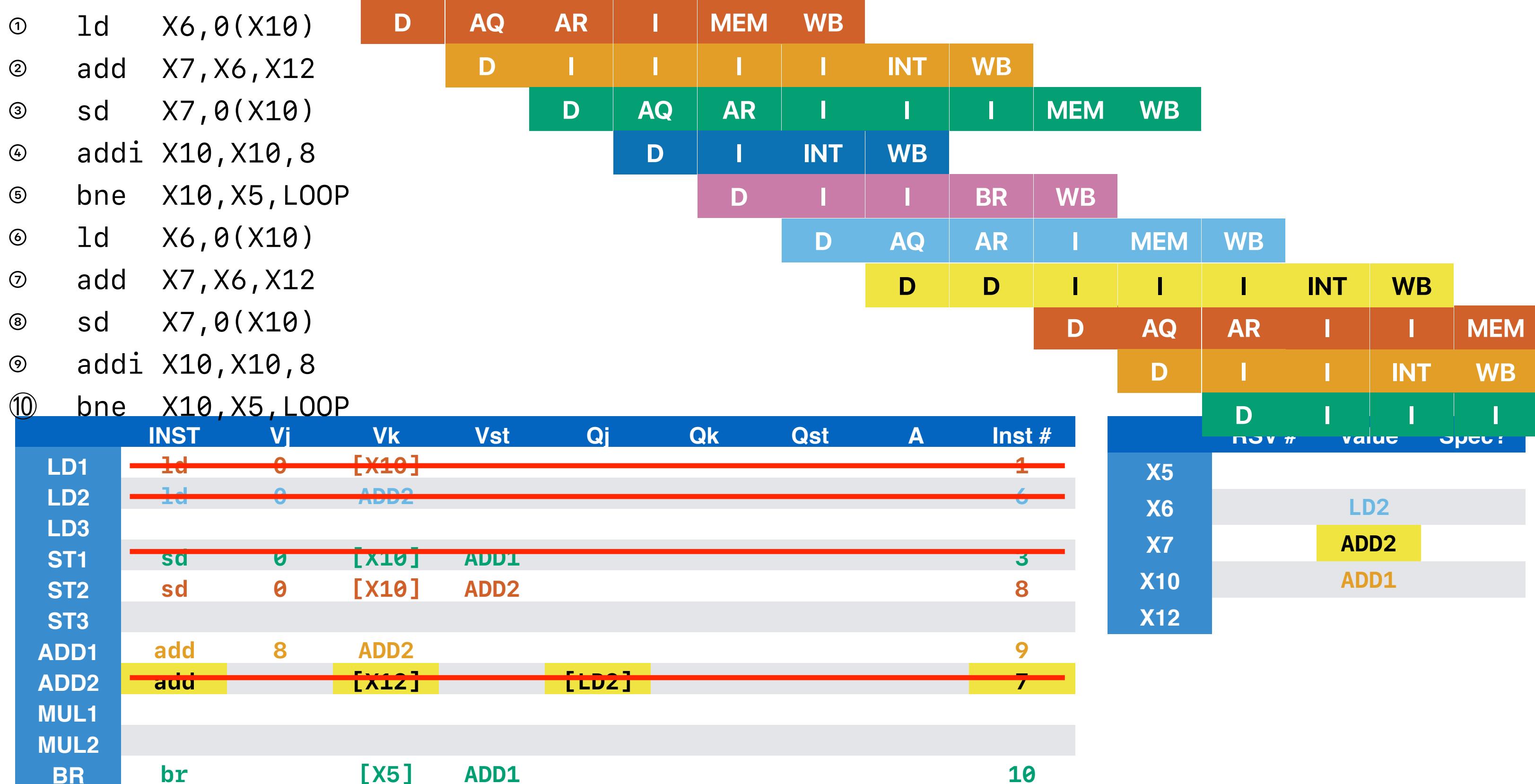


	INST	V _j	V _k	V _{st}	Q _j	Q _k	Q _{st}	A	Inst #	Res v _{st}	v _{value}	Spec?
LD1	ld	0	[X10]						1	X5		
LD2	ld	0	ADD2						6	X6		LD2
LD3										X7	ADD2	
ST1	sd	0	[X10]	ADD1					3	X10	ADD1	
ST2	sd	0	[X10]			ADD2			8	X12		
ST3												
ADD1	add	8	ADD2						9			
ADD2	add		[X12]		[LD2]				7			
MUL1												
MUL2												
BR	br		[X5]	ADD1					10			

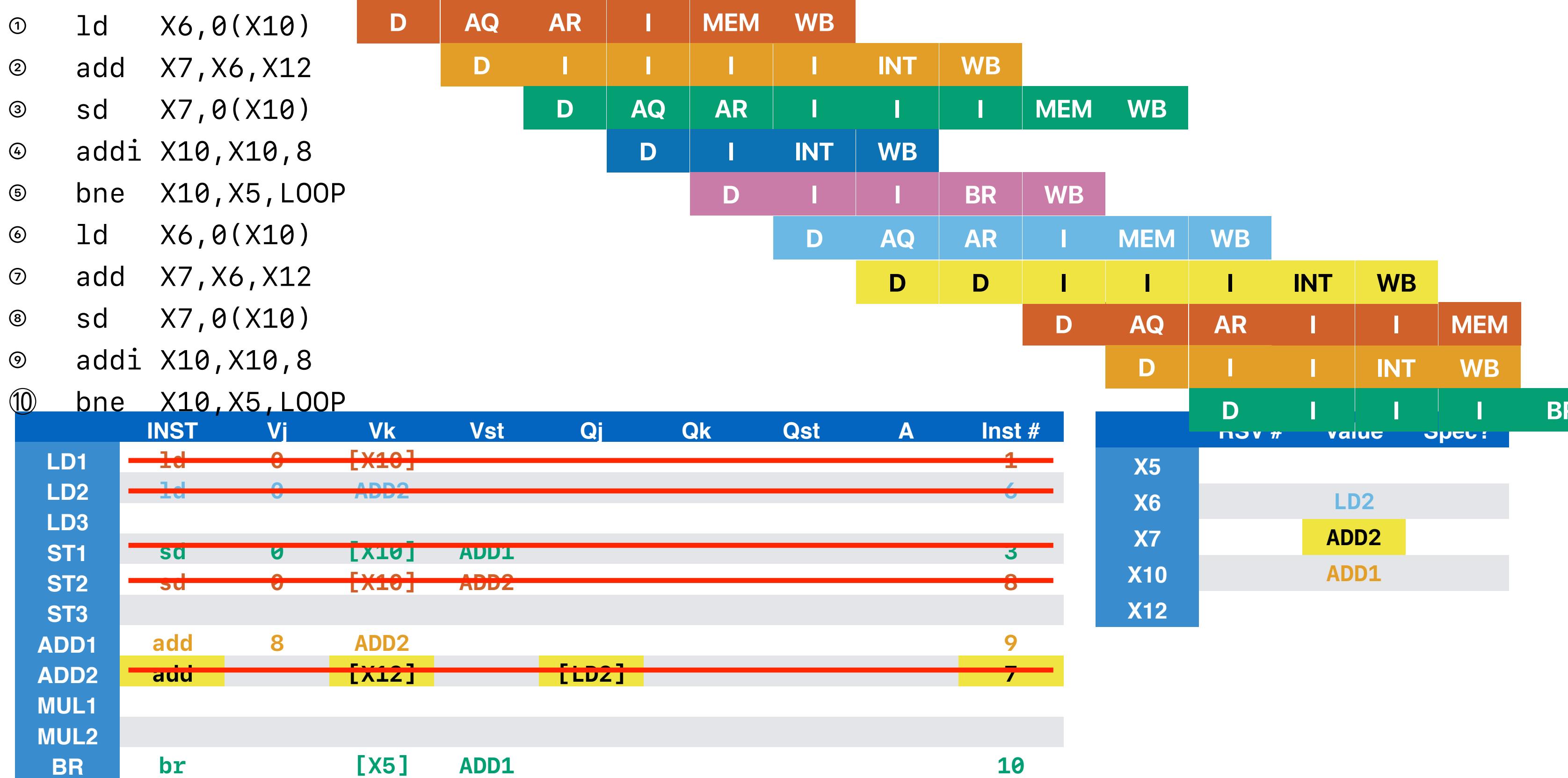
Tomasulo in motion



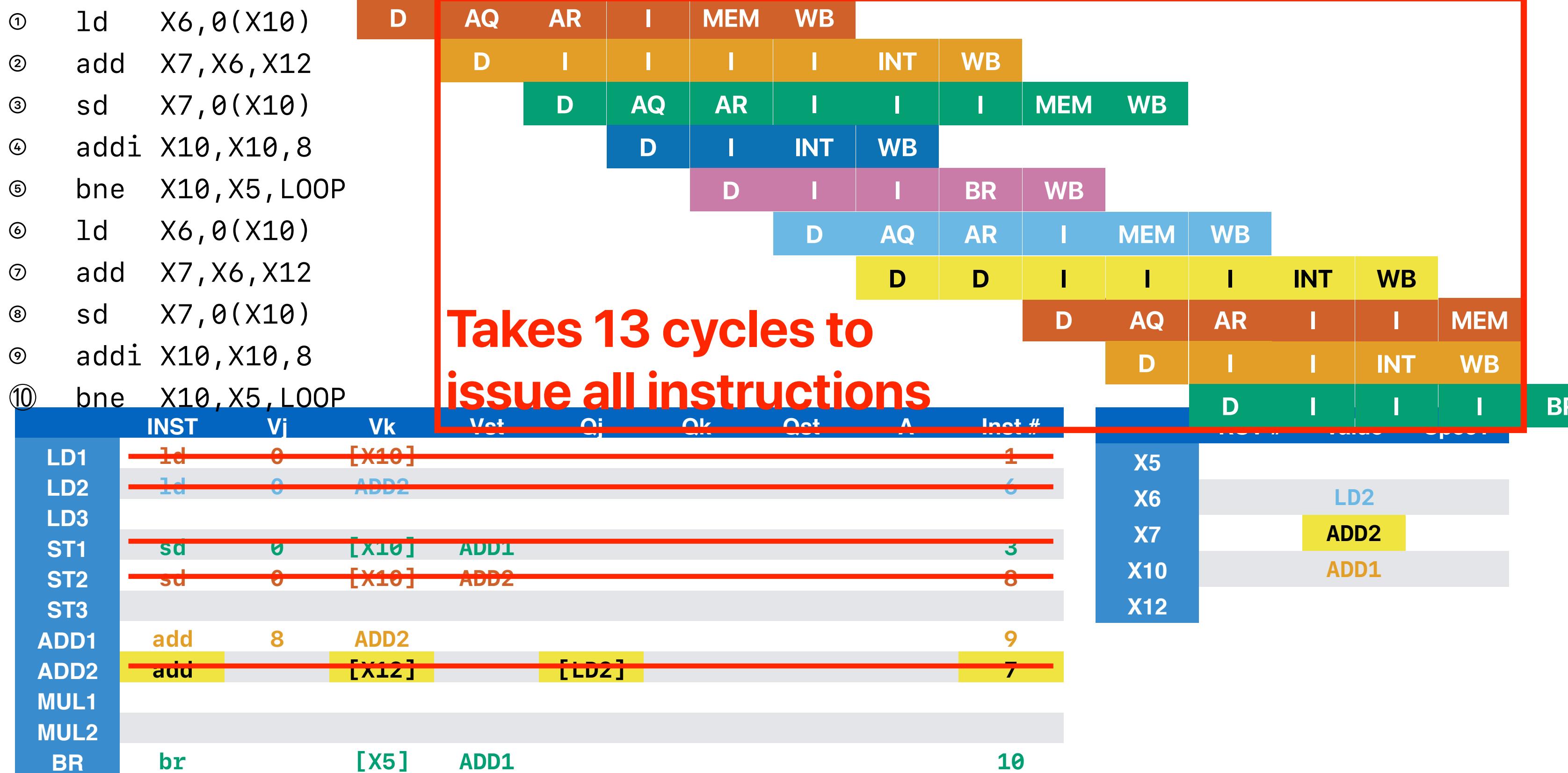
Tomasulo in motion



Tomasulo in motion



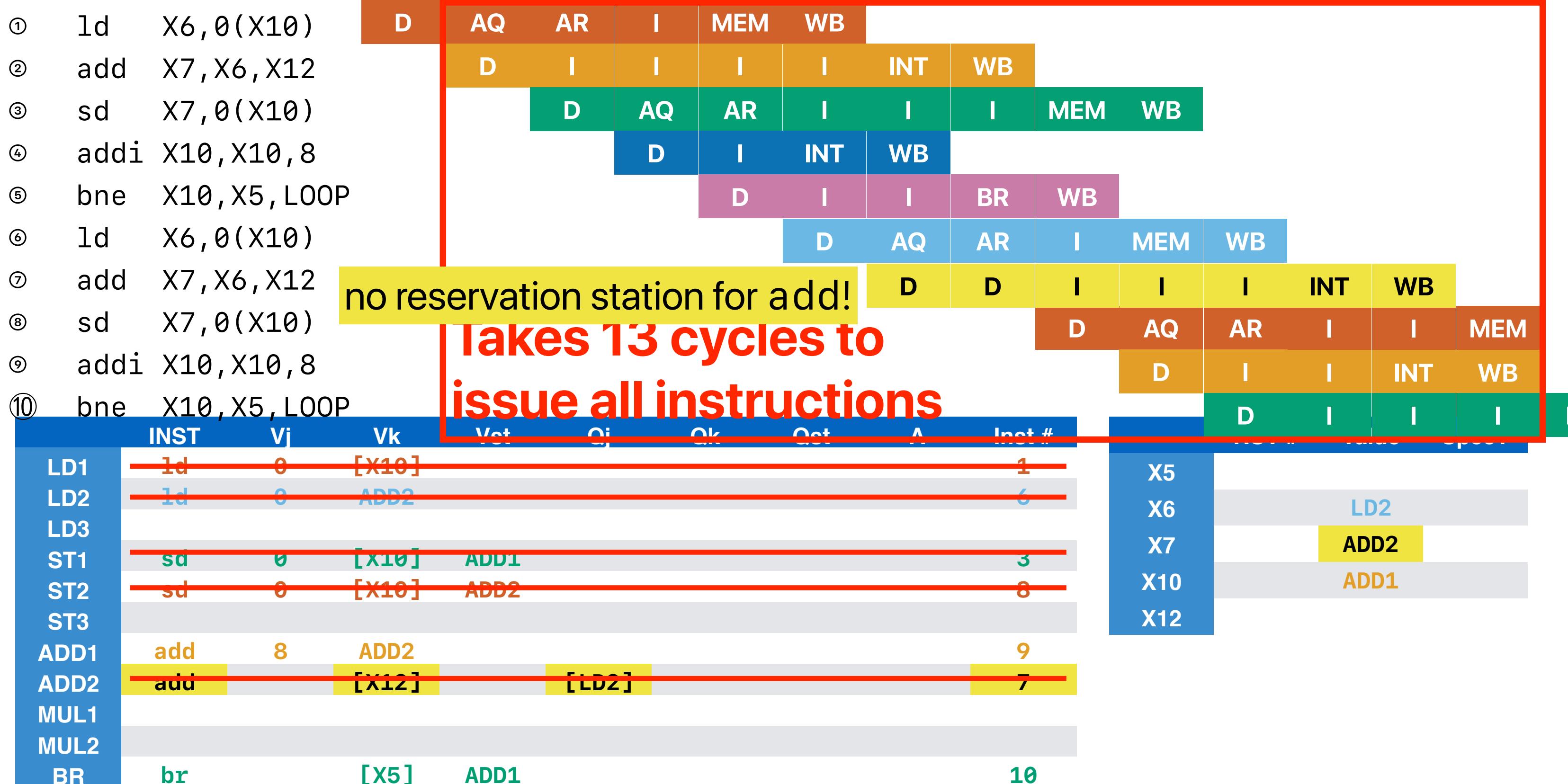
Tomasulo in motion



Register renaming

- K. C. Yeager, "The Mips R10000 superscalar microprocessor," in IEEE Micro, vol. 16, no. 2, pp. 28-41, April 1996.
- R. E. Kessler, "The Alpha 21264 microprocessor," in IEEE Micro, vol. 19, no. 2, pp. 24-36, March-April 1999.

Tomasulo in motion



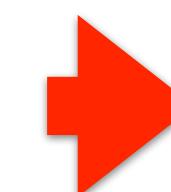
Recap: Why is B better than A?

A

```
inline int popcount(uint64_t x){  
    int c=0;  
    while(x) {  
        c += x & 1;  
        x = x >> 1;  
    }  
    return c;  
}
```

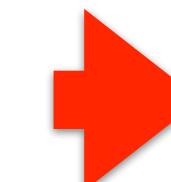
B

```
inline int popcount(uint64_t x) {  
    int c = 0;  
    while(x) {  
        c += x & 1;  
        x = x >> 1;  
        c += x & 1;  
        x = x >> 1;  
        c += x & 1;  
        x = x >> 1;  
        c += x & 1;  
        x = x >> 1;  
    }  
    return c;  
}
```

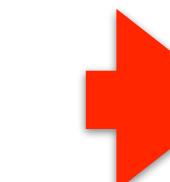


```
and x2, x1, 1  
add x3, x3, x2  
shr x1, x1, 1  
bne x1, x0, LOOP
```

4*n instructions



```
and x2, x1, 1  
add x3, x3, x2  
shr x1, x1, 1  
and x2, x1, 1  
add x3, x3, x2  
shr x1, x1, 1  
and x2, x1, 1  
add x3, x3, x2  
shr x1, x1, 1  
and x2, x1, 1  
add x3, x3, x2  
shr x1, x1, 1  
bne x1, x0, LOOP
```



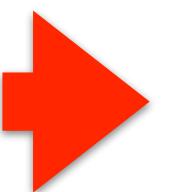
```
and x2, x1, 1  
shr x4, x1, 1  
shr x5, x1, 2  
shr x6, x1, 3  
shr x1, x1, 4  
and x7, x4, 1  
and x8, x5, 1  
and x9, x6, 1  
add x3, x3, x2  
add x3, x3, x7  
add x3, x3, x8  
add x3, x3, x9  
bne x1, x0, LOOP
```

13*(n/4) = 3.25*n instructions

85 Only one branch for four iterations in A

Recap: Why is B better than A?

and x2, x1, 1
add x3, x3, x2
shr x1, x1, 1
and x2, x1, 1
add x3, x3, x2
shr x1, x1, 1
and x2, x1, 1
add x3, x3, x2
shr x1, x1, 1
and x2, x1, 1
add x3, x3, x2
shr x1, x1, 1
bne x1, x0, LOOP

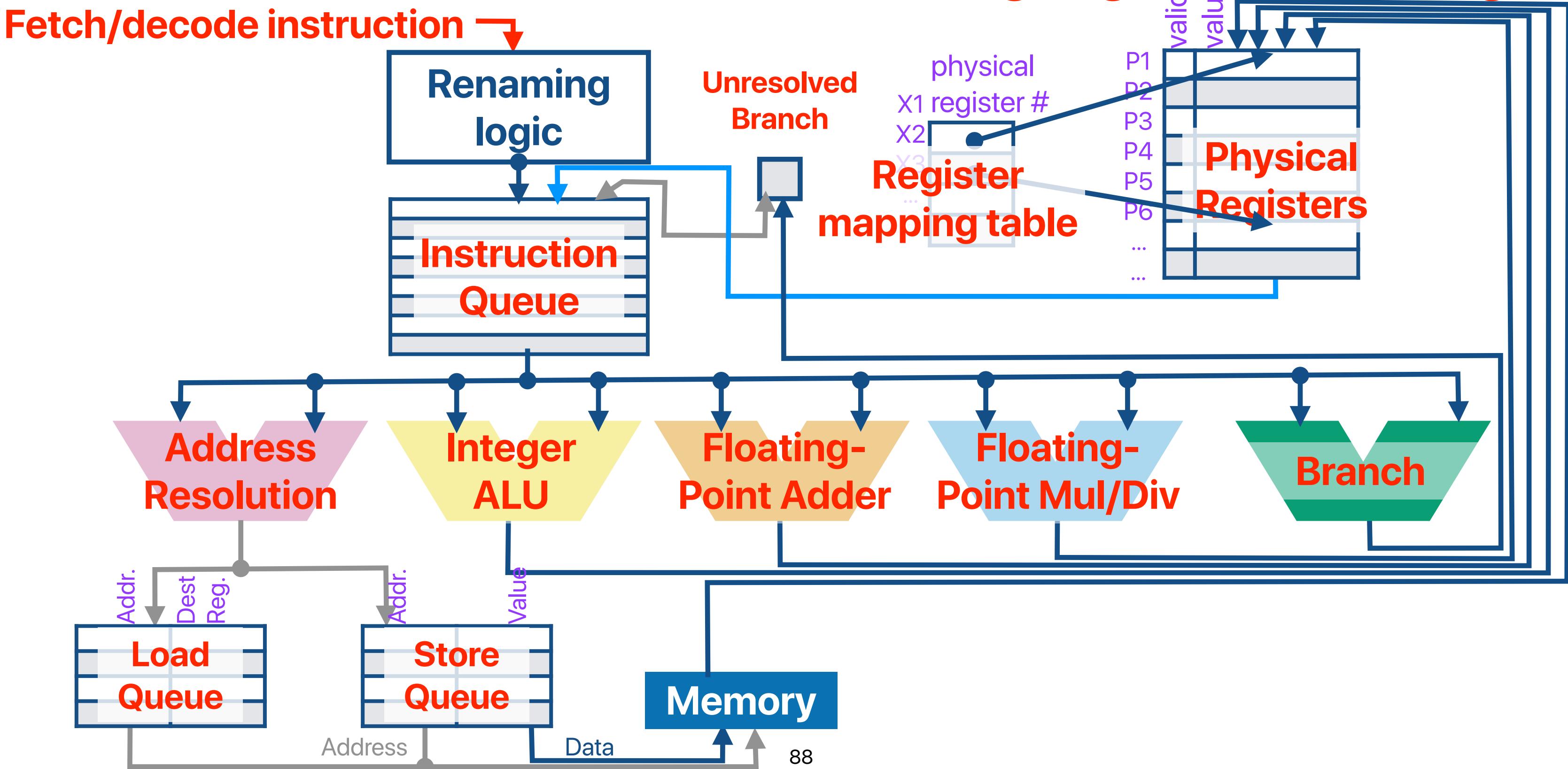


and x2, x1, 1
shr x4, x1, 1
shr x5, x1, 2
shr x6, x1, 3
shr x1, x1, 4
and x7, x4, 1
and x8, x5, 1
and x9, x6, 1
add x3, x3, x2
add x3, x3, x7
add x3, x3, x8
add x3, x3, x9
bne x1, x0, LOOP

Register renaming

- Decouple “reservation stations” from functional units
- Provide a set of “physical registers” and a mapping table mapping “architectural registers” to “physical registers”
- Allocate a physical register for a new output
- Stages
 - Dispatch/Rename (R) — allocate a “physical register” for the output of 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 register renaming



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

R

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 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		
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 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		
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



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		

Register renaming in motion

- | | | R | I | AR | LSQ | MEM |
|---|-------------------|---|---|----|-----|-----|
| ① | ld X6, 0(X10) | R | I | AR | LSQ | MEM |
| ② | add X7, X6, X12 | R | I | I | I | |
| ③ | sd X7, 0(X10) | | R | I | I | |
| ④ | addi X10, X10, 8 | | | R | I | |
| ⑤ | bne X10, X5, LOOP | | | | | R |
| 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	
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			

Register renaming in motion

			R	I	AR	LSQ	MEM	WB
①	ld	X6, 0(X10)	R	I	AR	LSQ	MEM	WB
②	add	X7, X6, X12	R	I	I	I	I	I
③	sd	X7, 0(X10)		R	I	I	I	I
④	addi	X10, X10, 8			R	I	INT	
⑤	bne	X10, X5, LOOP				R	I	
⑥	ld	X6, 0(X10)					R	
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	1		1	P6			
P2	0		1	P7			
P3	0		1	P8			
P4	0		1	P9			
P5				P10			

Register renaming in motion

			R	I	AR	LSQ	MEM	WB	
①	ld	X6, 0(X10)	R						
②	add	X7, X6, X12		R	I	I	I	I	INT
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
⑦	add	X7, X6, X12							R
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		
9		
10		

	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			

Register renaming in motion

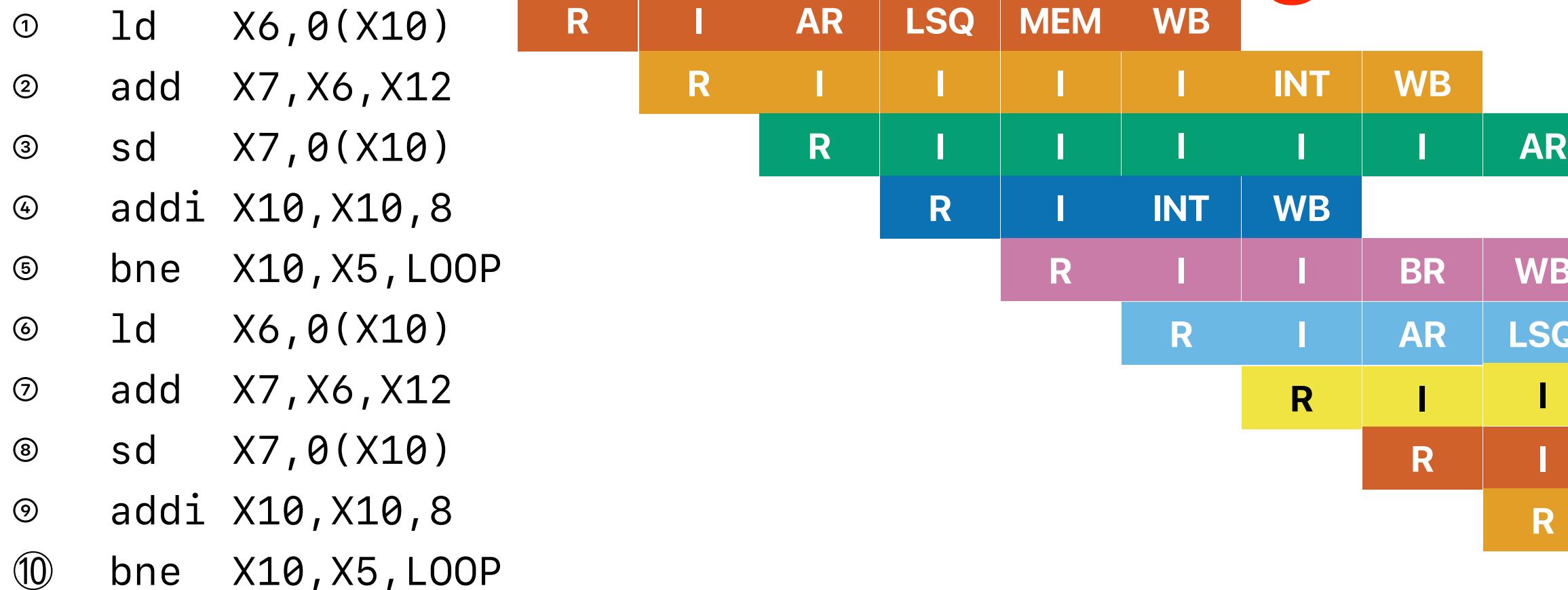
			R	I	AR	LSQ	MEM	WB	
①	ld	X6, 0(X10)	R	I	AR	LSQ	MEM	WB	
②	add	X7, X6, X12	R	I	I	I	I	INT	WB
③	sd	X7, 0(X10)	R	I	I	I	I	I	I
④	addi	X10, X10, 8		R	I	INT		WB	
⑤	bne	X10, X5, LOOP			R	I	I	BR	
⑥	ld	X6, 0(X10)			R	I		AR	
⑦	add	X7, X6, X12			R	I			
⑧	sd	X7, 0(X10)				R			
⑨	addi	X10, X10, 8							
⑩	bne	X10, X5, 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	1		1	P6			
P2	1		1	P7			
P3	1		1	P8			
P4	0		1	P9			
P5	0		1	P10			

Register renaming in motion

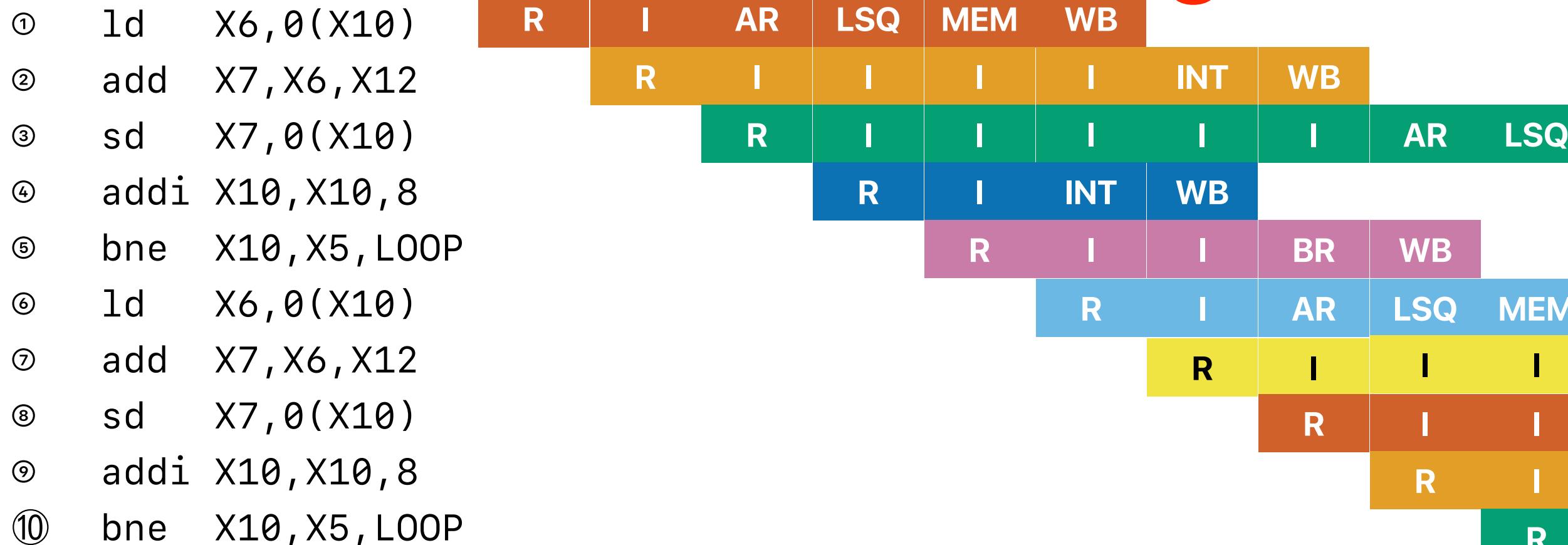


Renamed instruction		
1	ld P1, 0(X10)	
2	add P2, P1, X12	
3	sd P2, 0(X10)	
4	addi P3, X10, 8	
5	bne P3, X5, LOOP	
6	ld P4, 0(P3)	
7	add P5, P1, X12	
8	sd P5, 0(P3)	
9	addi P6, P3, 8	
10		

Physical Register	
X5	
X6	P1
X7	P5
X10	P3
X12	

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

Register renaming in motion

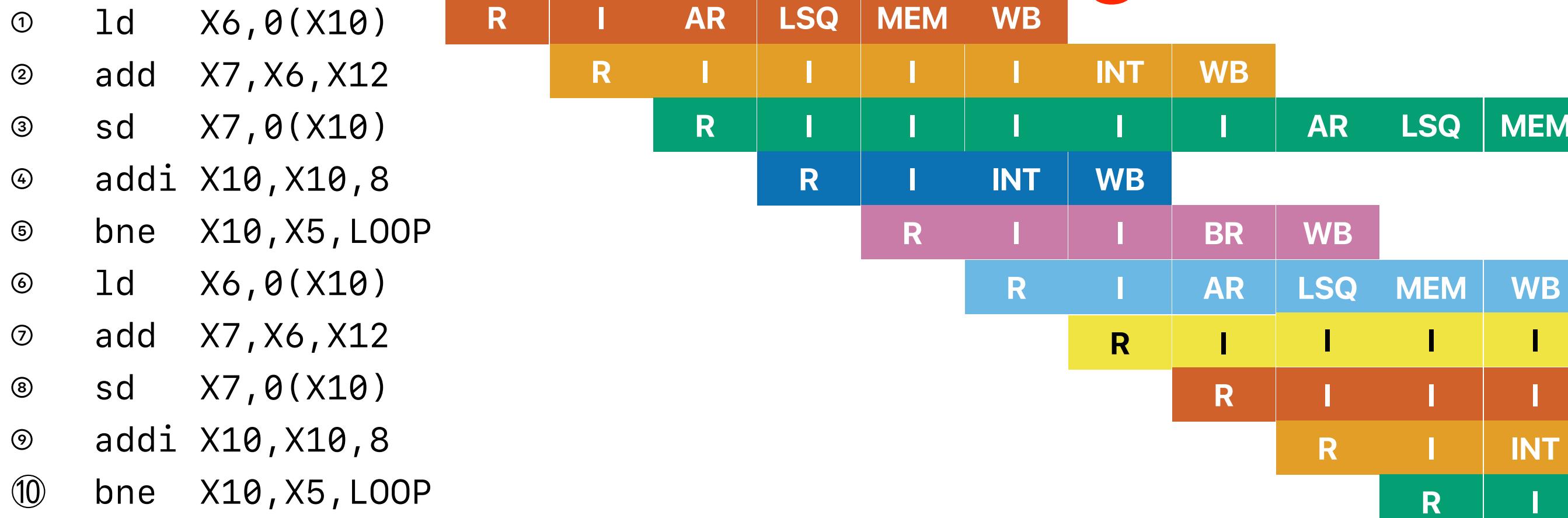


	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	Physical Register
X5	
X6	P1
X7	P5
X10	P3
X12	

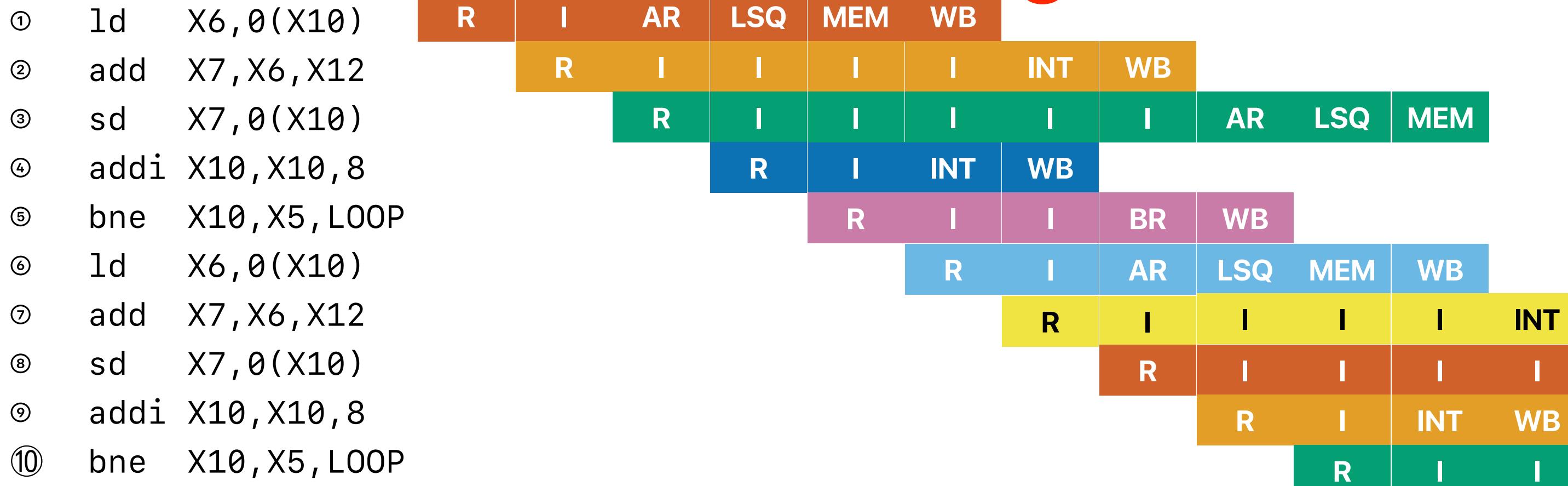
	Valid	Value	In use		Valid	Value	In use
P1	1		1	P6	0		1
P2	1		1	P7			
P3	1		1	P8			
P4	0		1	P9			
P5	0		1	P10			

Register renaming in motion



Renamed instruction		Physical Register		Valid Value In use			Valid Value In use		
1	ld P1, 0(X10)	X5		P1	1	1	P6	0	1
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)								

Register renaming in motion

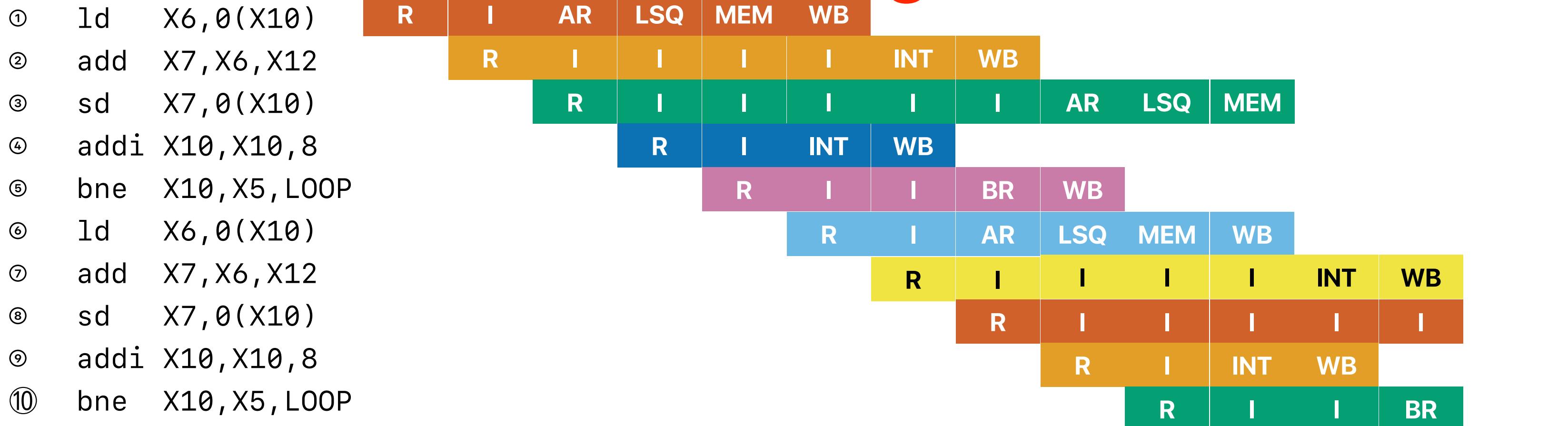


Renamed instruction	
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

Physical Register	
X5	
X6	P1
X7	P5
X10	P3
X12	

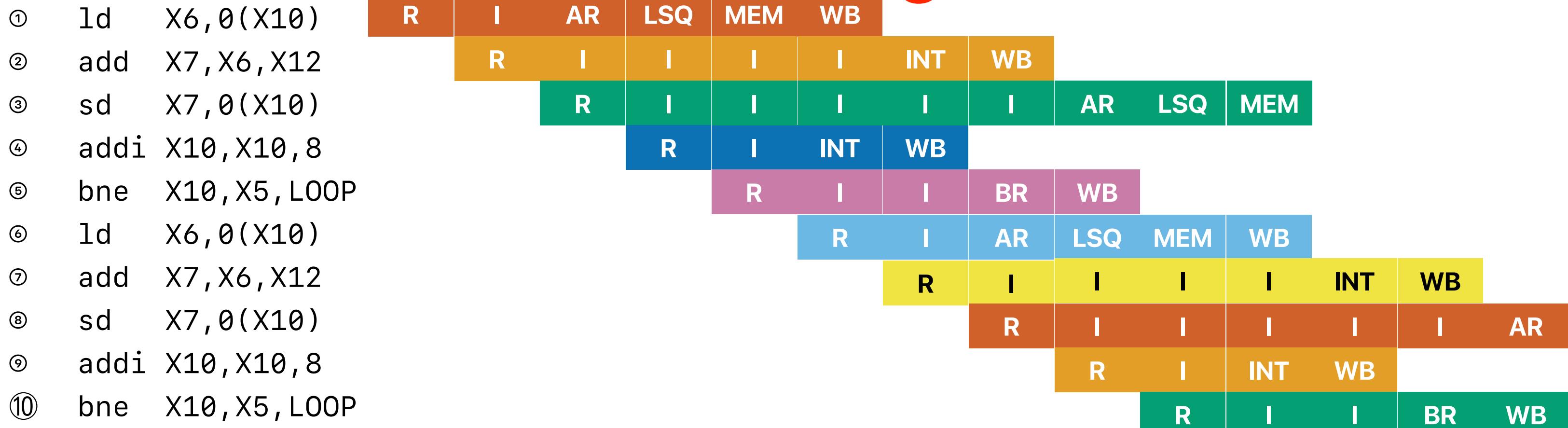
Valid Value In use			Valid Value In use		
P1	1	1	P6	1	1
P2	1	1	P7		
P3	1	1	P8		
P4	1	1	P9		
P5	0	1	P10		

Register renaming in motion



Renamed instruction		Physical Register		Valid Value In use			Valid Value In use		
1	ld P1, 0(X10)	X5		P1	1	1	P6	1	1
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)								

Register renaming in motion

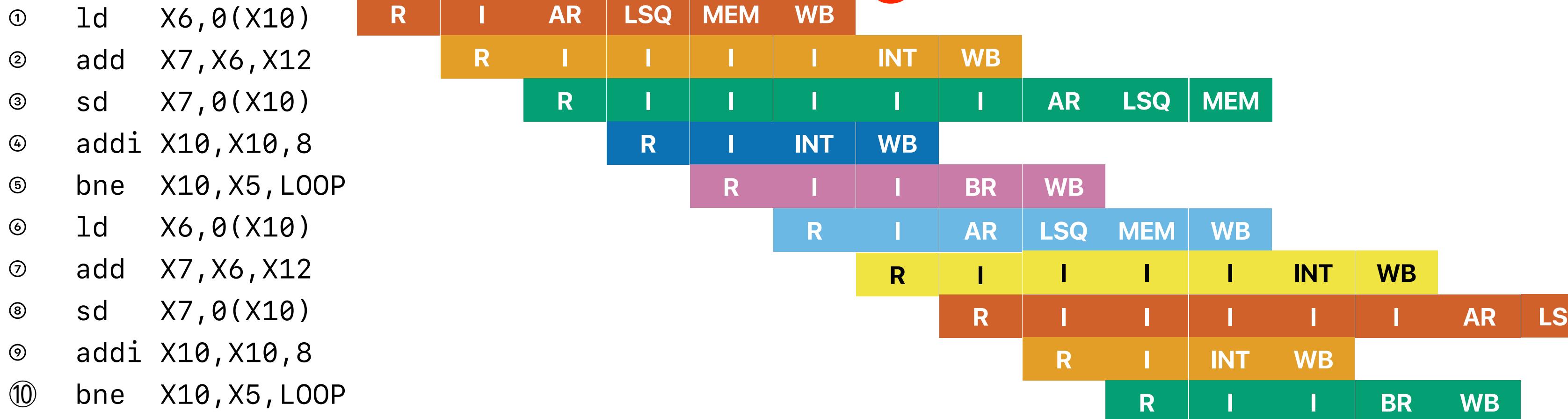


Renamed instruction	
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

Physical Register	
X5	P1
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		

Register renaming in motion

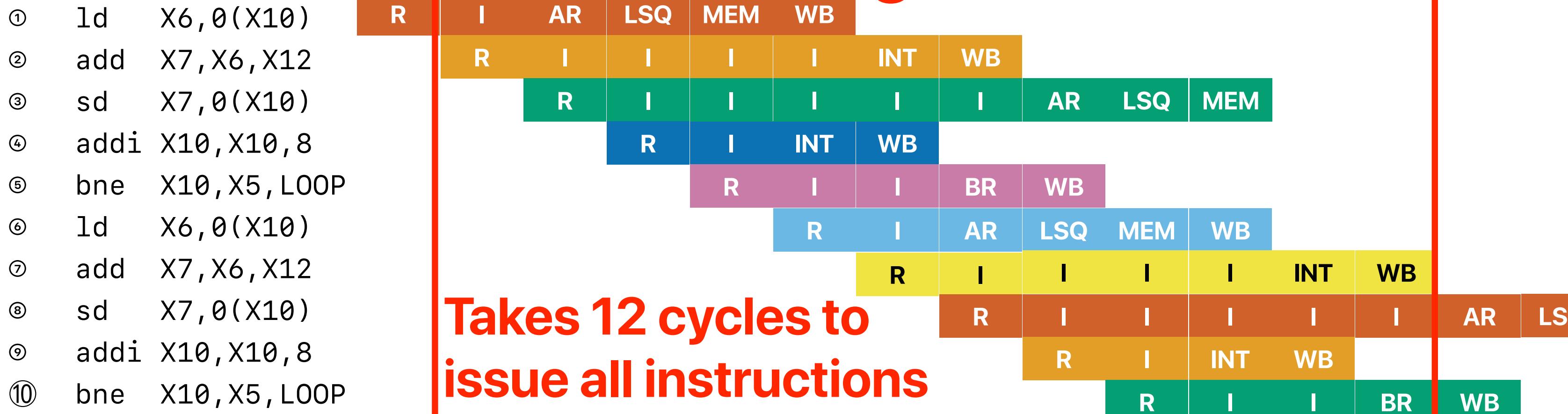


	Renamed instruction
1	ld P1, 0(X10)
2	add P2, P1, X12
3	sd P2, 0(X10)
4	addi P3, X10, 8
5	bne P3, X5, LOOP
6	ld P4, 0(P3)
7	add P5, P1, X12
8	sd P5, 0(P3)
9	addi P6, P3, 8
10	bne P6, 0(X10)

	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			

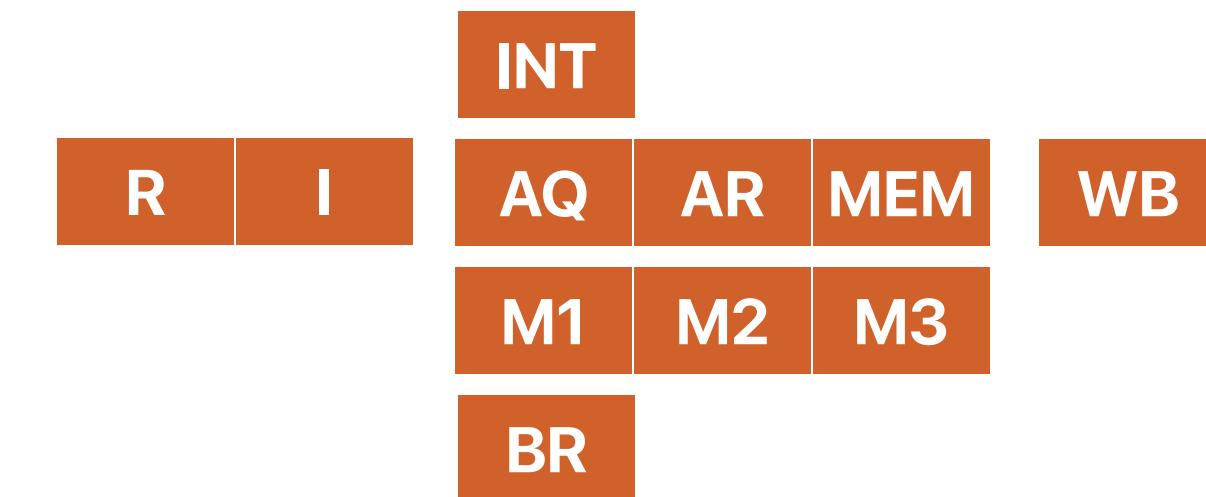
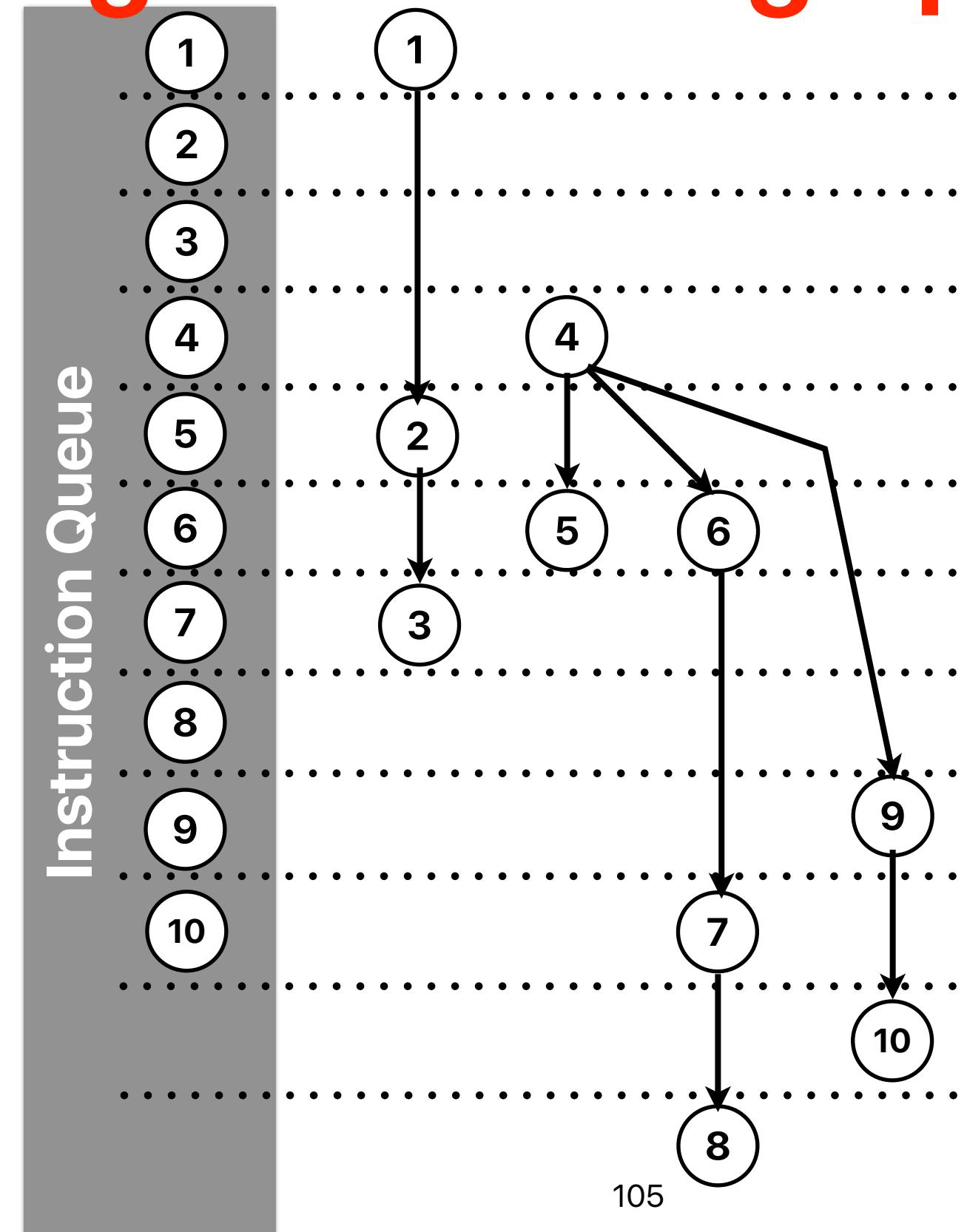
Register renaming in motion



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

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

Register renaming in motion

- ① ld X10, 8(X10) R
- 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)	R	I	AR	LSQ
②	addi	X7, X7, 1	R	I	INT	
③	bne	X10, X0, LOOP		R	I	
④	ld	X10, 8(X10)			R	
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)	R	I	AR	LSQ	MEM
②	addi	X7, X7, 1	R	I	INT	WB	
③	bne	X10, X0, LOOP	R	I	I		
④	ld	X10, 8(X10)		R	I		
⑤	addi	X7, X7, 1			R		
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

		R	I	AR	LSQ	MEM	WB
①	ld X10, 8(X10)	R					
②	addi X7, X7, 1		R	I	INT	WB	
③	bne X10, X0, LOOP			R	I	I	I
④	ld X10, 8(X10)			R	I	I	
⑤	addi X7, X7, 1				R	I	
⑥	bne X10, X0, LOOP					R	
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

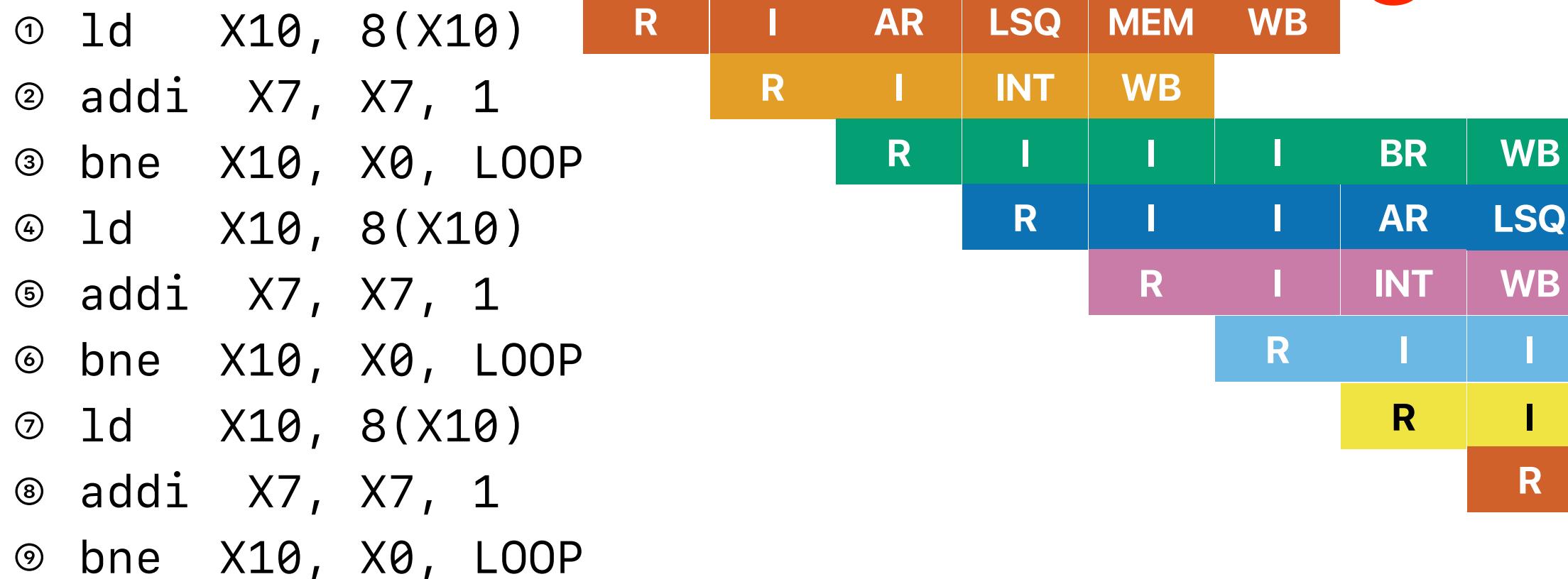
		R	I	AR	LSQ	MEM	WB
①	ld X10, 8(X10)	R					
②	addi X7, X7, 1		R	I	INT	WB	
③	bne X10, X0, LOOP			R	I	I	BR
④	ld X10, 8(X10)			R	I	I	AR
⑤	addi X7, X7, 1				R	I	INT
⑥	bne X10, X0, LOOP				R	I	
⑦	ld X10, 8(X10)					R	
⑧	addi X7, X7, 1						
⑨	bne X10, X0, LOOP						

	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

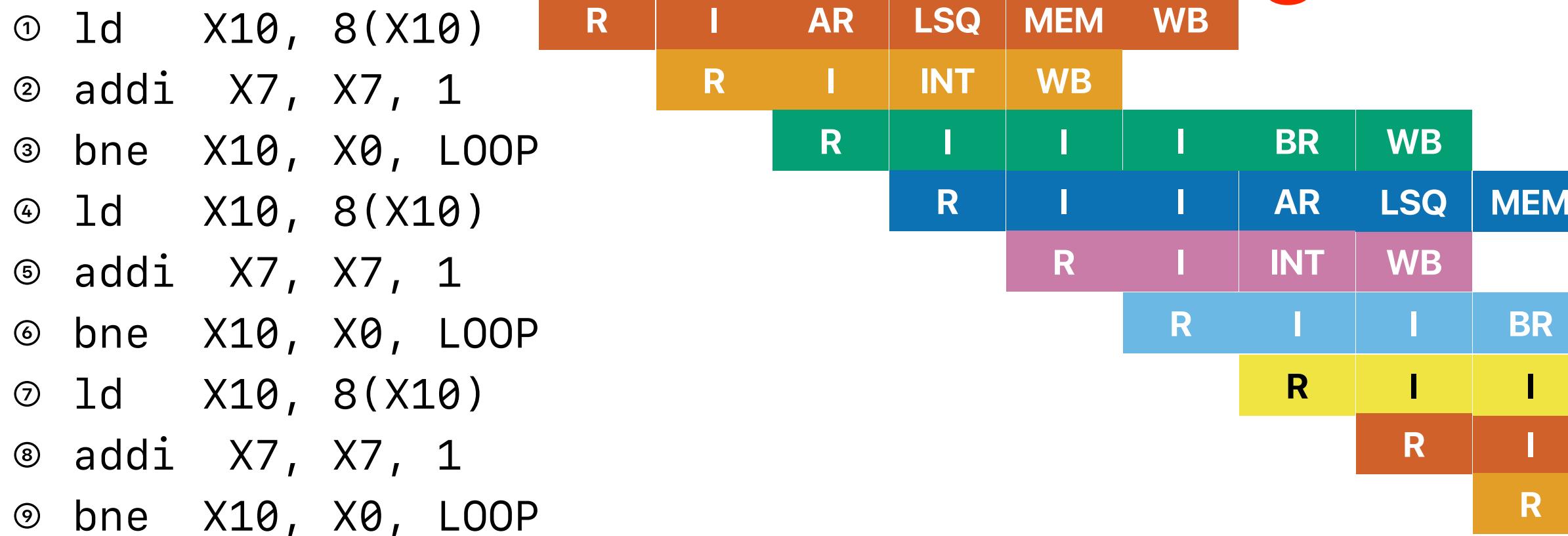


	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	
10	

	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

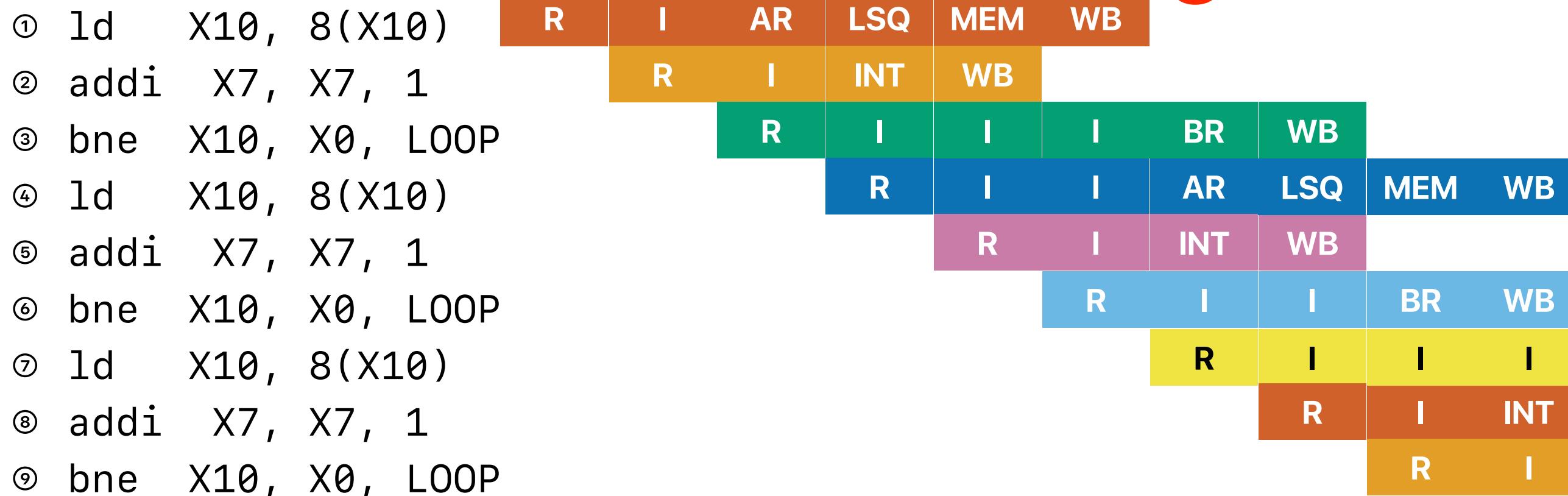


	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	bne P5, X0, LOOP
10	

	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

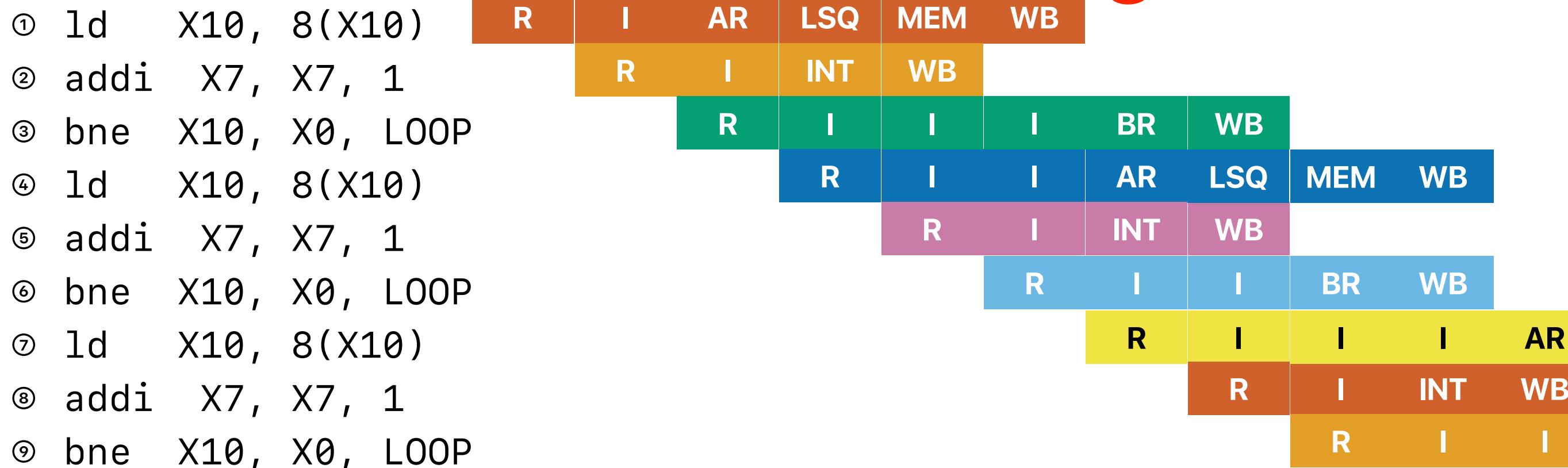


	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	bne P5, X0, LOOP
10	

	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

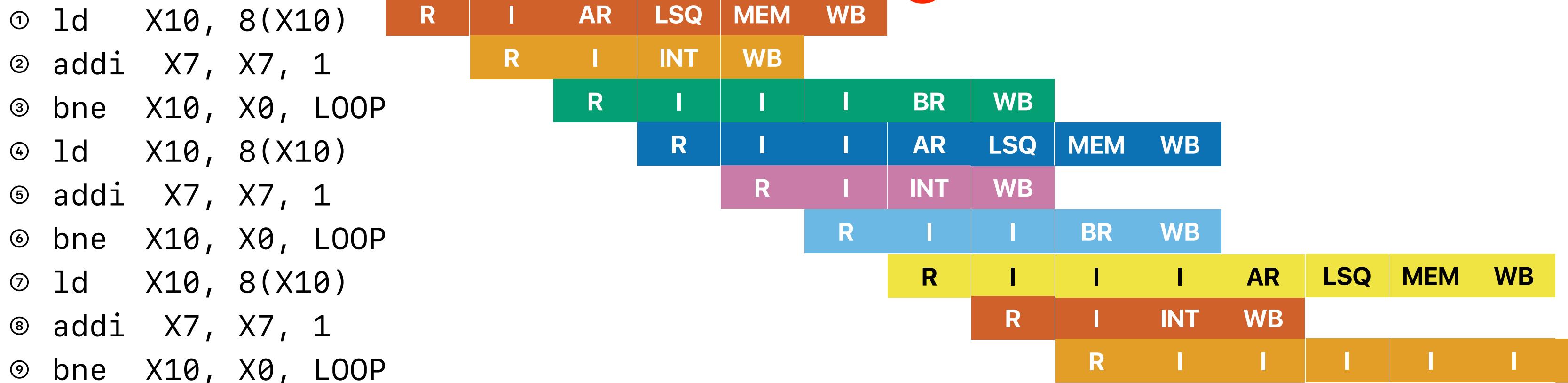


	Renamed instruction
1	ld P1, 0(X10)
2	add P2, X7, 1
3	bne P1, X0, LOOP
4	ld P3, 0(P1)
5	add P4, P2, 1
6	bne P3, X0, LOOP
7	ld P5, 0(P3)
8	add P6, P4, 1
9	bne P5, X0, LOOP
10	

	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



	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

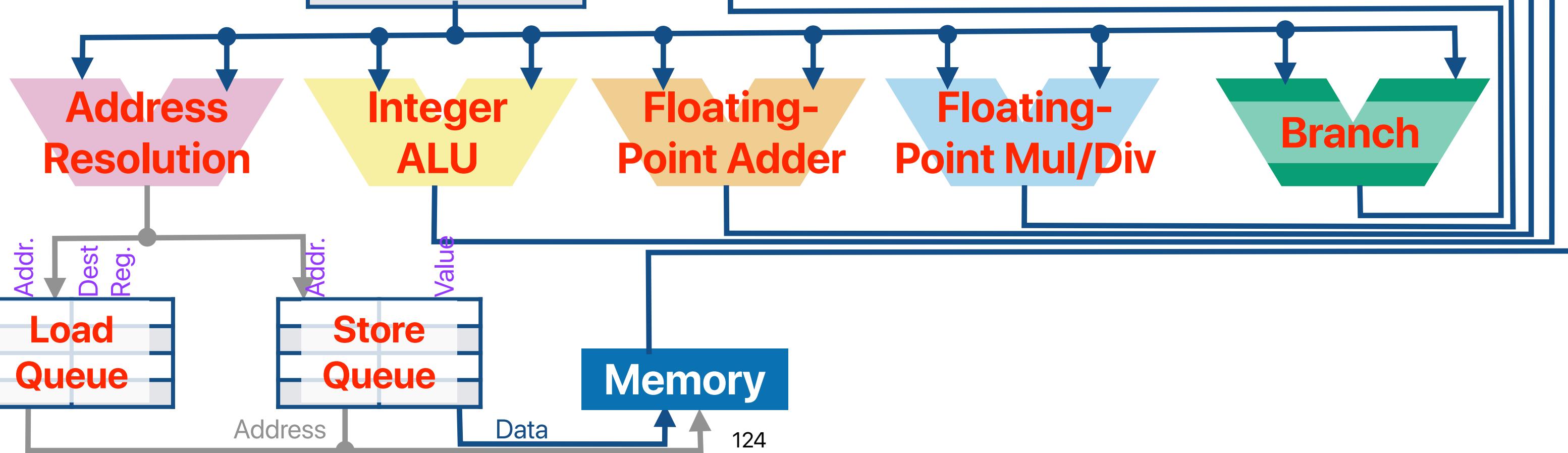
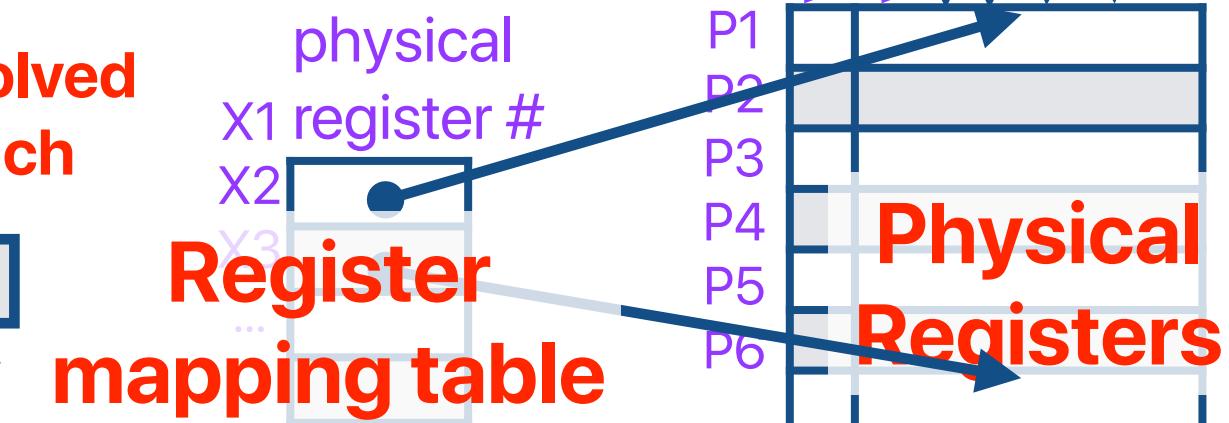
- Since we have more functional units now, we should fetch/decode more instructions each cycle so that we can have more instructions to issue!
- Super-scalar: fetch/decode/issue more than one instruction each cycle
 - Fetch width: how many instructions can the processor fetch/decode each cycle
 - Issue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

Overview of a processor supporting register renaming

Fetch/decode instruction →



Unresolved Branch



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

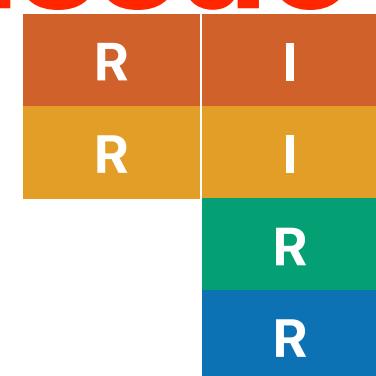
Renamed instruction		
1	ld P1, 0(X10)	
2	add P2, P1, X12	
3		
4		
5		
6		
7		
8		
9		
10		

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



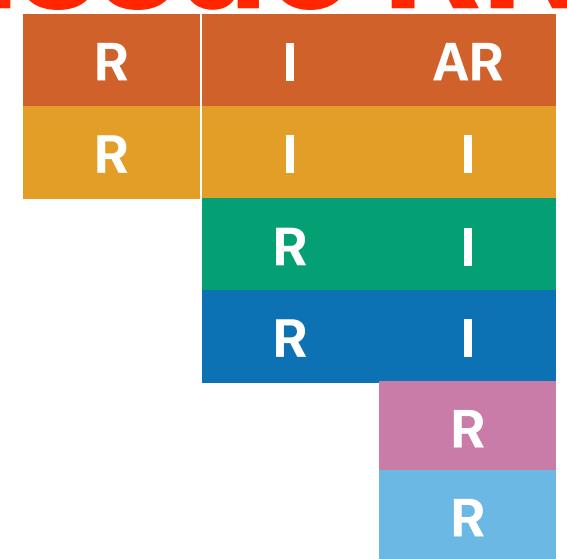
Renamed instruction		
1	ld P1, 0(X10)	
2	add P2, P1, X12	
3	sd P2, 0(X10)	
4	addi P3, X10, 8	
5		
6		
7		
8		
9		
10		

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

			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		
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
⑧	sd	X7, 0(X10)			R	I	I	I	I
⑨	addi	X10, X10, 8			R	I	I	INT	WB
⑩	bne	X10, X5, LOOP			R	I	I	I	I

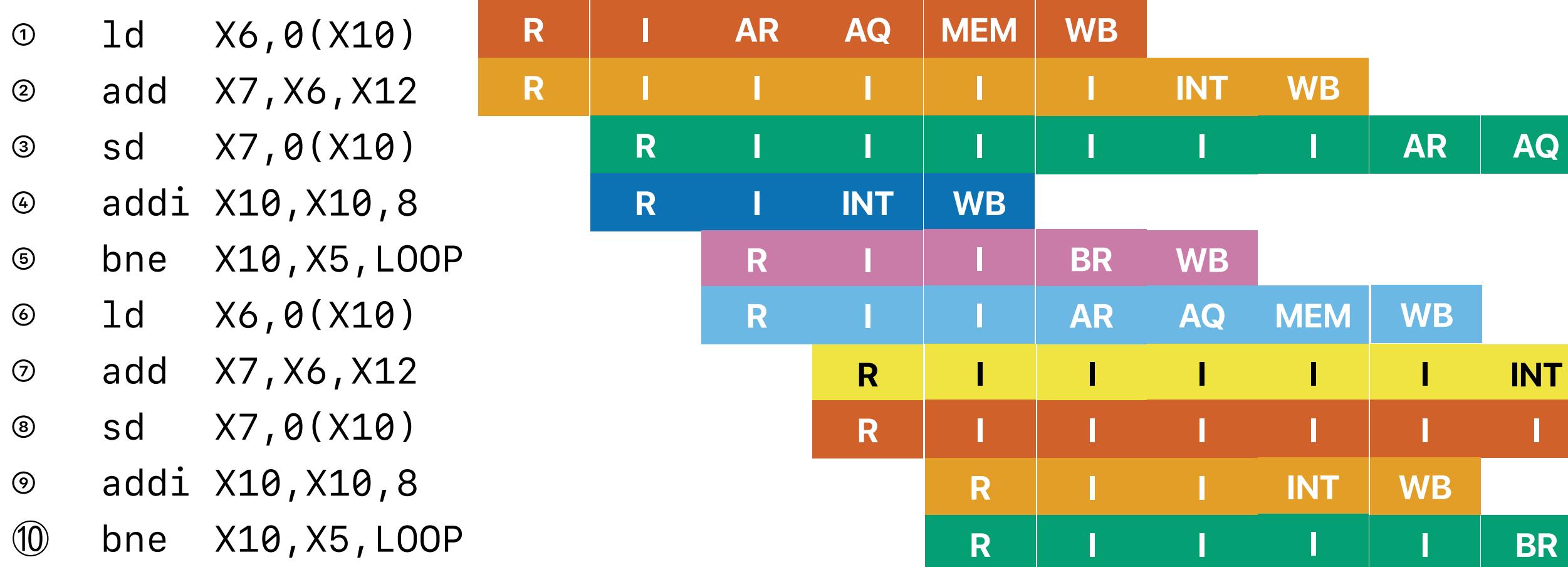
Renamed instruction	Physical Register	Valid			Valid		
		Value	In use		Value	In use	
1 ld P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	1	1	P9
5 bne P3, X5, LOOP	X12			P5	0	1	P10
6 ld P4, 0(P3)							
7 add P5, P1, X12						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

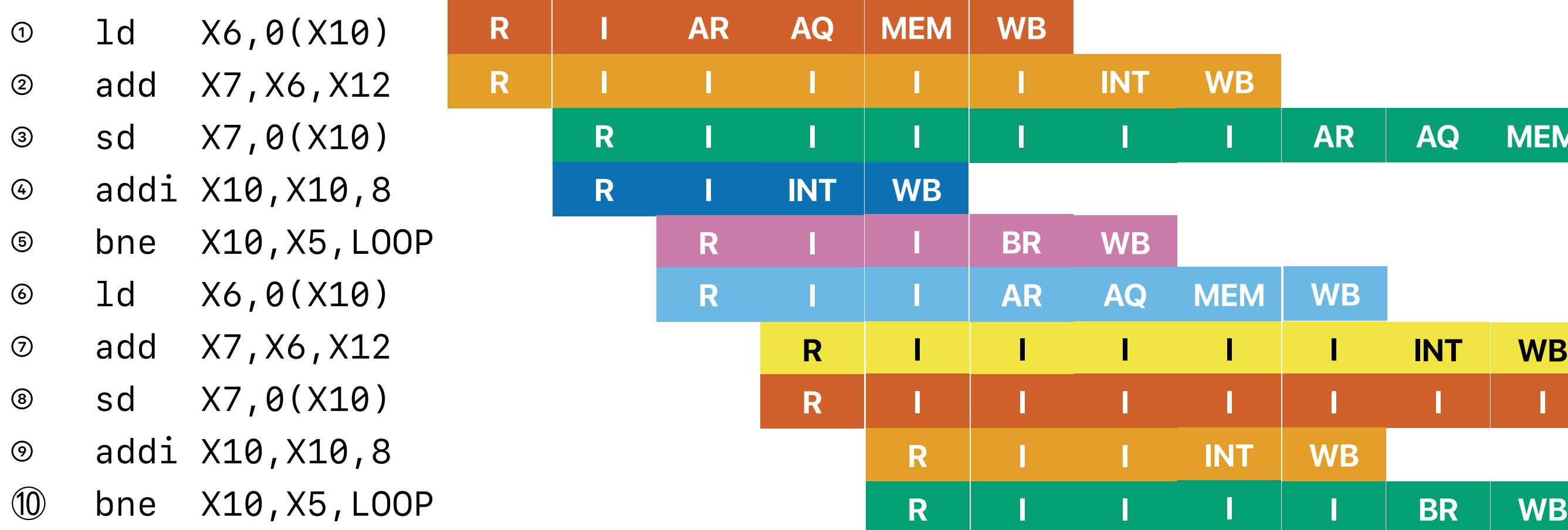
Renamed instruction	Physical Register	Valid			Valid		
		Value	In use		Value	In use	
1 ld P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	1	1	P9
5 bne P3, X5, LOOP	X12			P5	0	1	P10
6 ld P4, 0(P3)							
7 add P5, P1, X12						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

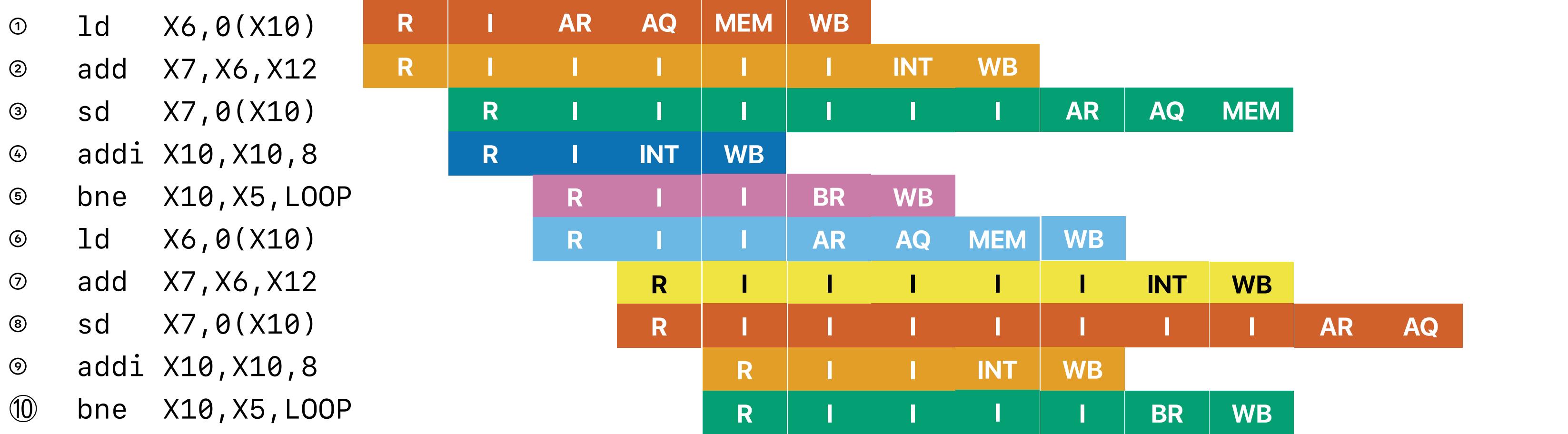
2-issue RR processor in motion

			R	I	AR	AQ	MEM	WB												
①	ld	X6, 0(X10)	R																	
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	I	AR	AQ	MEM							
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	WB									
⑧	sd	X7, 0(X10)		R	I	I	I	I	I	I	I	I	AR							
⑨	addi	X10, X10, 8		R	I	I	INT	WB												
⑩	bne	X10, X5, LOOP		R	I	I	I	I	BR	WB										

Takes 10 cycles to issue all instructions

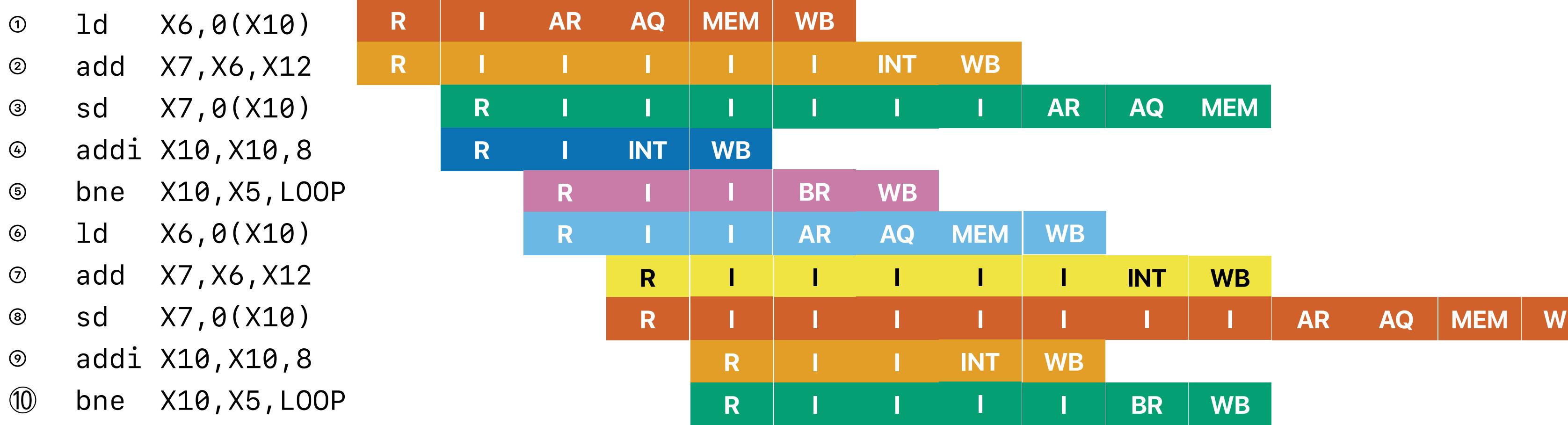
Renamed instruction	Physical Register	Valid			Valid		
		Value	In use		Value	In use	
1 ld P1, 0(X10)	X5			P1	1	1	P6
2 add P2, P1, X12	X6	P1		P2	1	1	P7
3 sd P2, 0(X10)	X7	P5		P3	1	1	P8
4 addi P3, X10, 8	X10	P3		P4	1	1	P9
5 bne P3, X5, LOOP				P5	1	1	P10
6 ld P4, 0(P3)							
7 add P5, P1, X12							
8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

2-issue RR processor in motion



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



Renamed instruction		Physical Register		Valid Value In use			Valid 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)								

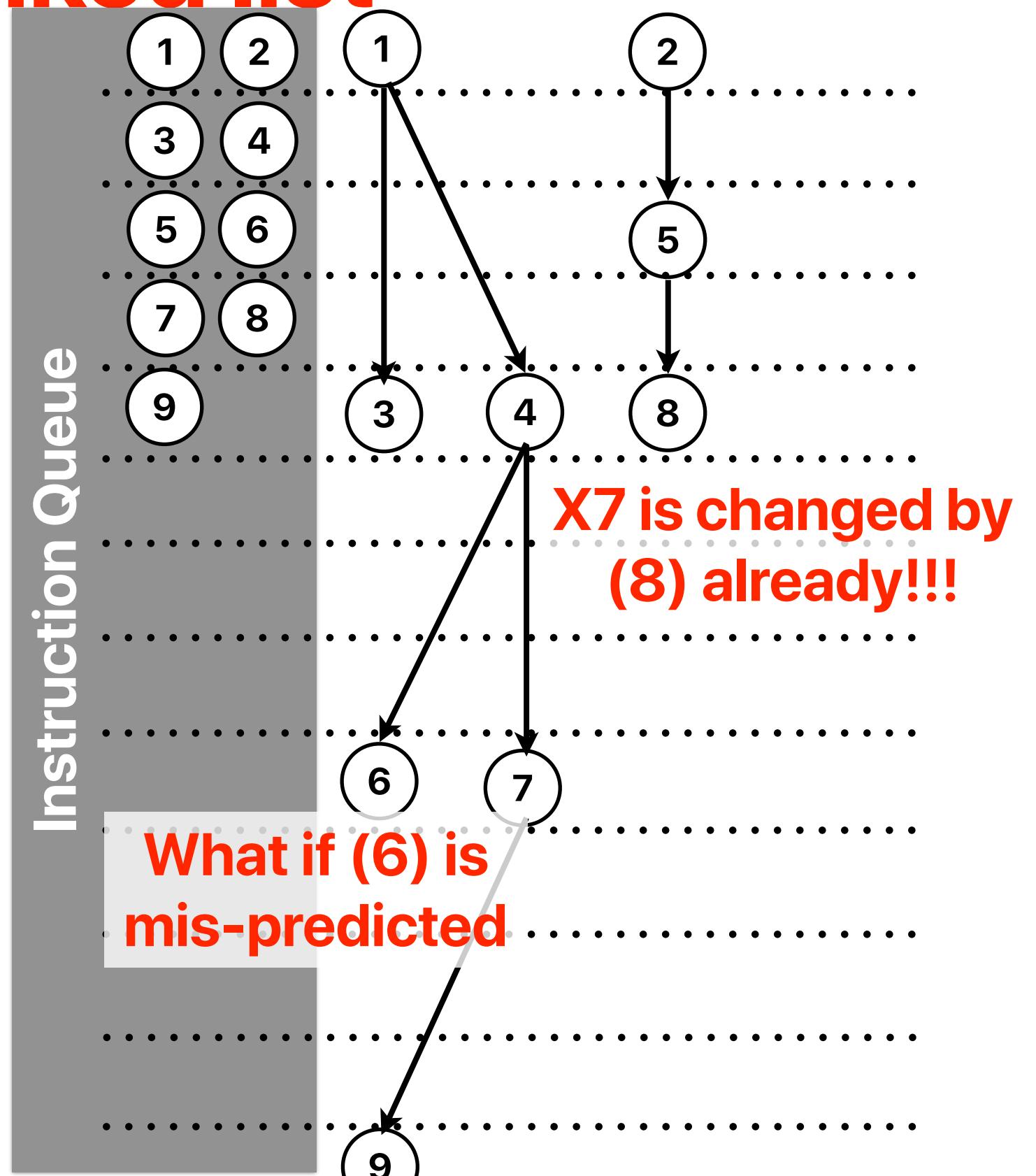
What about “linked list”

Static instructions

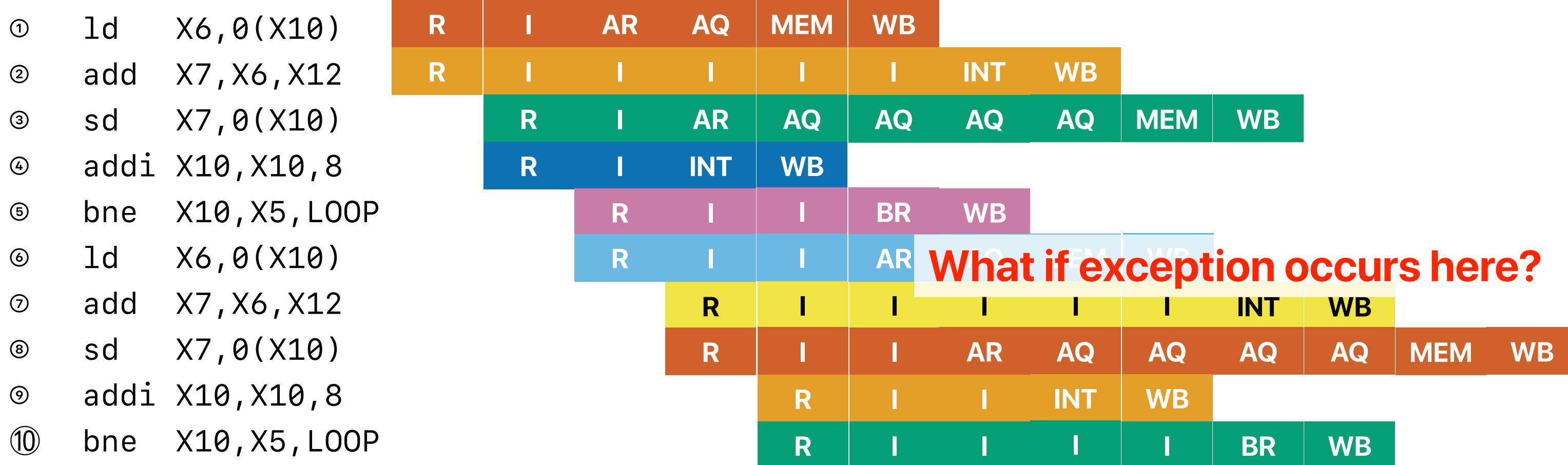
```
LOOP: ld    X10, 8(X10)  
      addi  X7, X7, 1  
      bne   X10, X0, LOOP
```

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



2-issue RR processor in motion



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	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

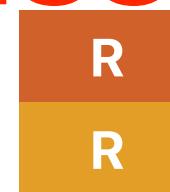
Reorder Buffer (ROB)

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

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

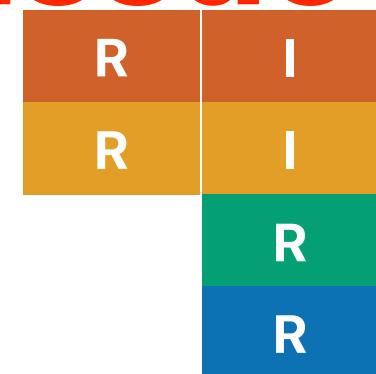


Renamed instruction			Physical Register			Valid Value In use			Valid Value In use								
			X5		X6	P1		P2	0	1	P6		P7		P8	P9	P10
1	ld	P1, 0(X10)															
2	add	P2, P1, X12															
3																	
4																	
5																	
6																	
7																	
8											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



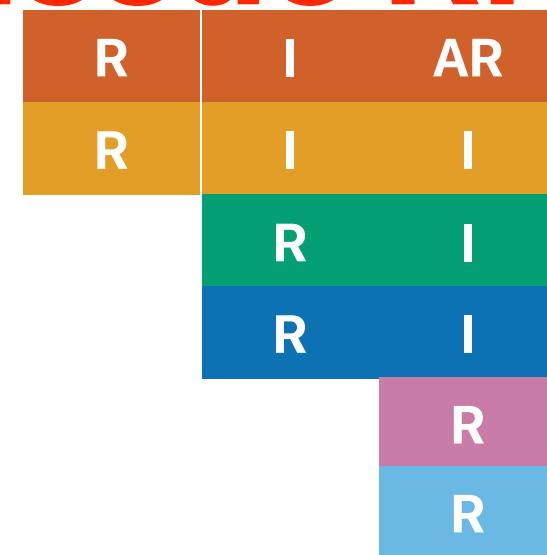
Renamed instruction		Physical Register			Valid Value In use			Valid Value In use			
		X5		X6	P1	X7	P2	X10	P3	X12	
1	ld P1, 0(X10)										P6
2	add P2, P1, X12										P7
3	sd P2, 0(X10)										P8
4	addi P3, X10, 8										P9
5											P10
6											
7											
8											
9											
10											

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 **Physical Register** **Register File**

	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**

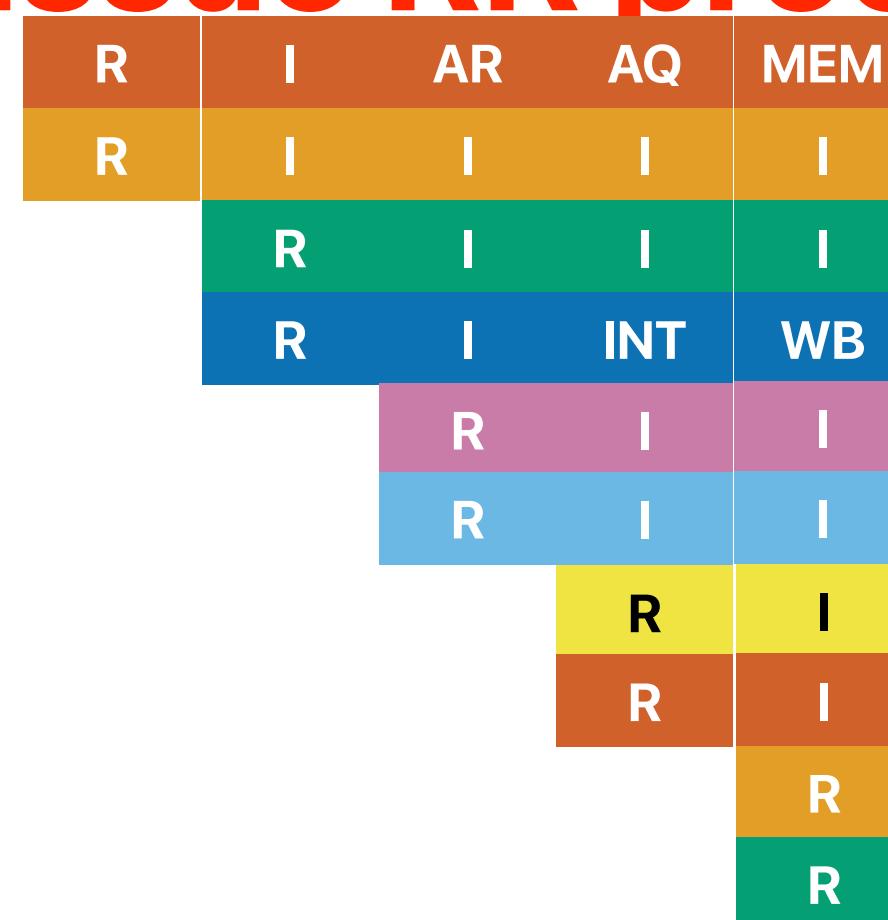
2-issue RR processor in motion

			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)	head	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	tail									
8	sd P5, 0(P3)										
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		
1	ld P1, 0(X10)	head
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	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	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	I	
⑧	sd X7, 0(X10)	R	I	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	P1
3	sd P2, 0(X10)	X7	P5
4	addi P3, X10, 8	X10	P3
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

157

	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			

tail

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	
⑧	sd	X7, 0(X10)			R	I	I	I	
⑨	addi	X10, X10, 8			R	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		

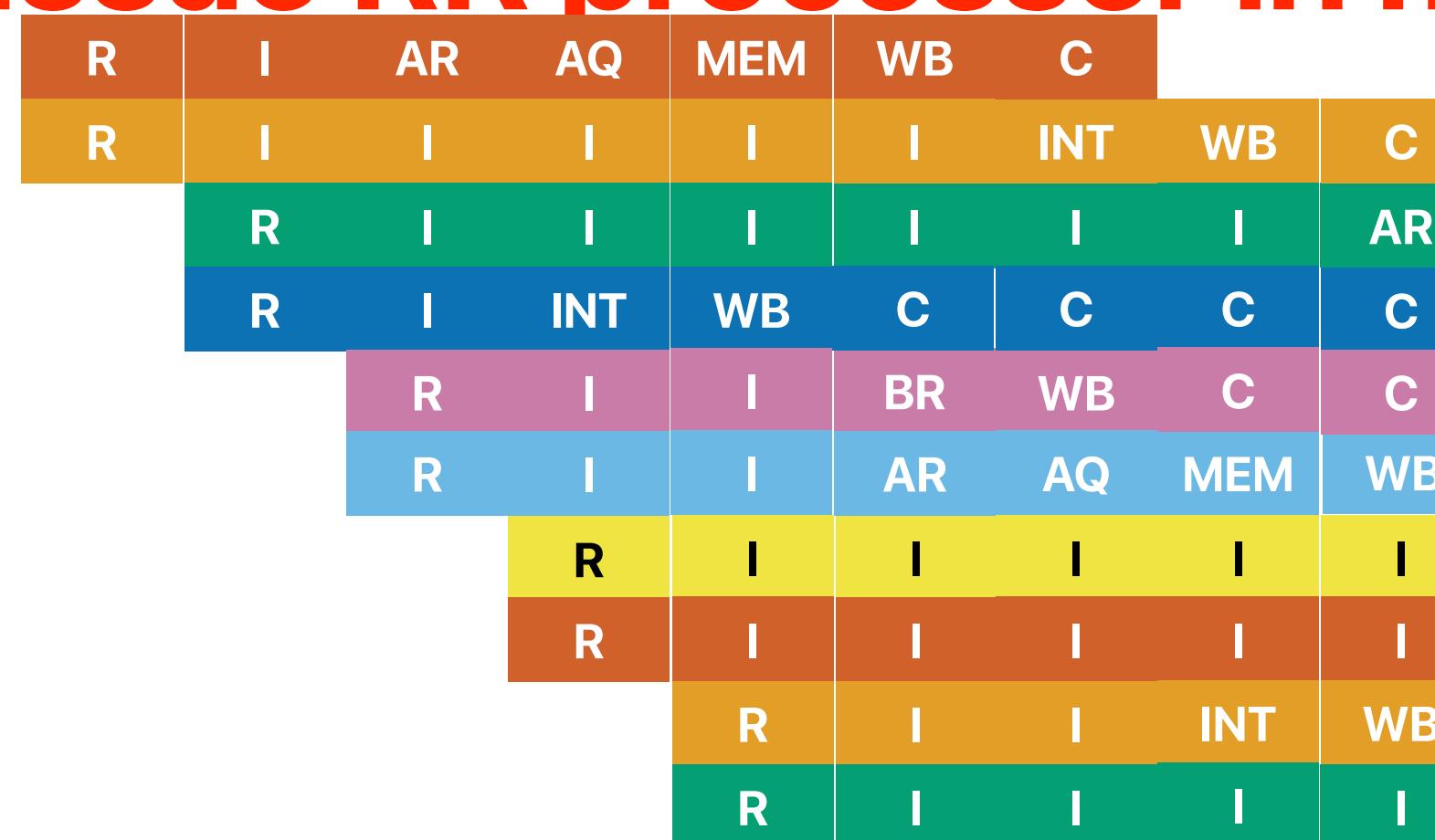
Renamed instruction		Physical Register		Valid	Value	In use	Valid	Value	In use
1	ld P1, 0(X10)	X5		P1	1	1	P6		
2	add P2, P1, X12	X6	P1	P2	1	1	P7		
3	sd P2, 0(X10)	X7	P5	P3	1	1	P8		
4	addi P3, X10, 8	X10	P3	P4	0	1	P9		
5	bne P3, X5, LOOP	X12		P5	0	1	P10		
6	ld P4, 0(P3)								
7	add P5, P1, X12						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

← head

← tail

2-issue RR processor in motion

- 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	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
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	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		

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

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		
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	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		

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	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	MEM
⑨	addi	X10, X10, 8		R	I	I	INT	WB	C	C	C	C	C	C
⑩	bne	X10, X5, LOOP		R	I	I	I	I	I	BR	WB	C	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		
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

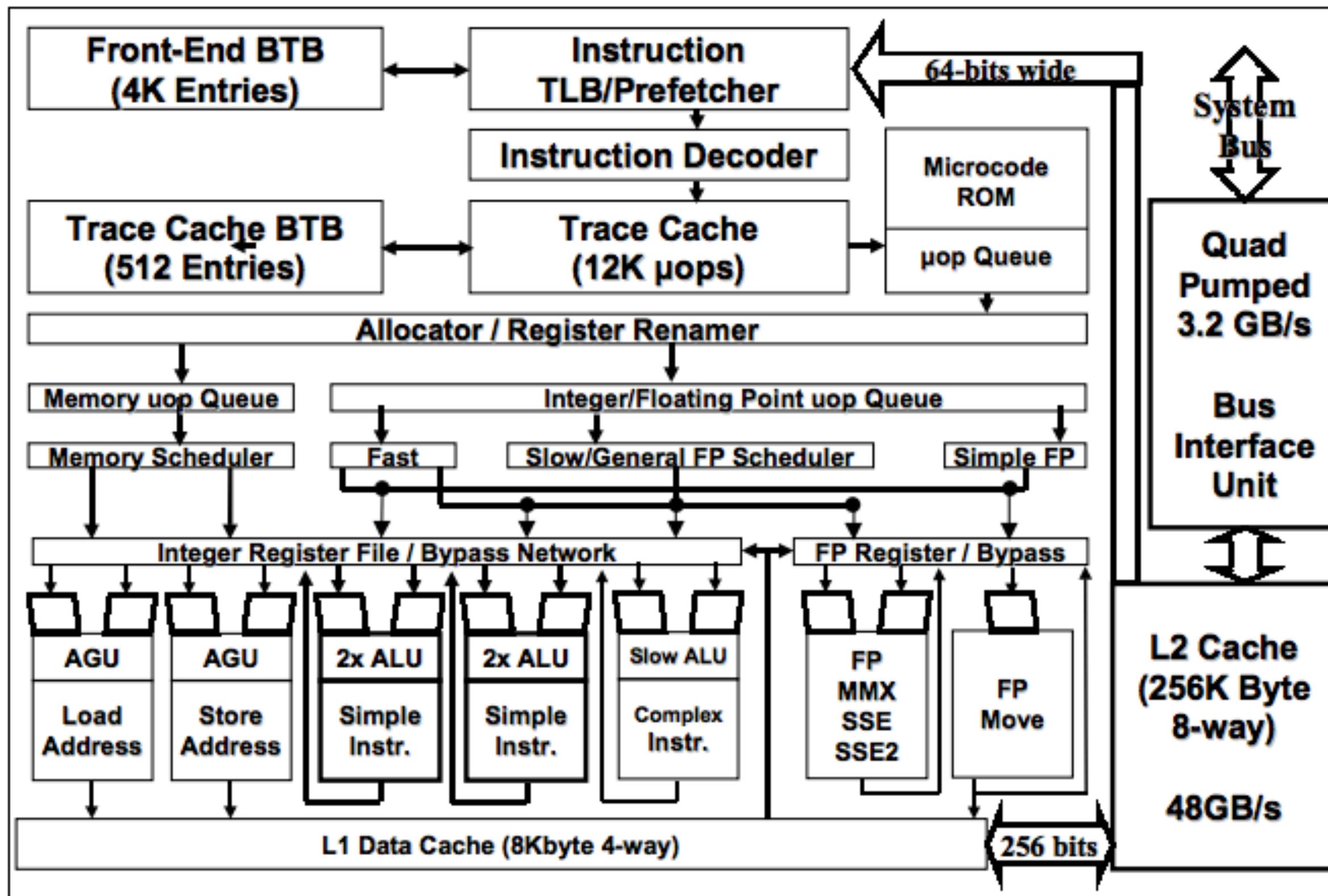
		R	I	AR	AQ	MEM	WB	C				
①	ld	X6, 0(X10)	R									
②	add	X7, X6, X12	R	I	I	I	I	INT	WB	C		
③	sd	X7, 0(X10)	R	I	I	I	I	I	AR	AQ	MEM	C
④	addi	X10, X10, 8	R	I	INT	WB	C	C	C	C	C	C
⑤	bne	X10, X5, LOOP		R	I	I	BR	WB	C	C	C	C
⑥	ld	X6, 0(X10)	R	I	I	AR	AQ	MEM	WB	C	C	C
⑦	add	X7, X6, X12		R	I	I	I	I	INT	WB	C	
⑧	sd	X7,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

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	1	1	P7		
3 sd P2, 0(X10)	X7	P5	1	1	P8		
4 addi P3, X10, 8	X10	P3	1	1	P9		
5 bne P3, X5, LOOP					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)							

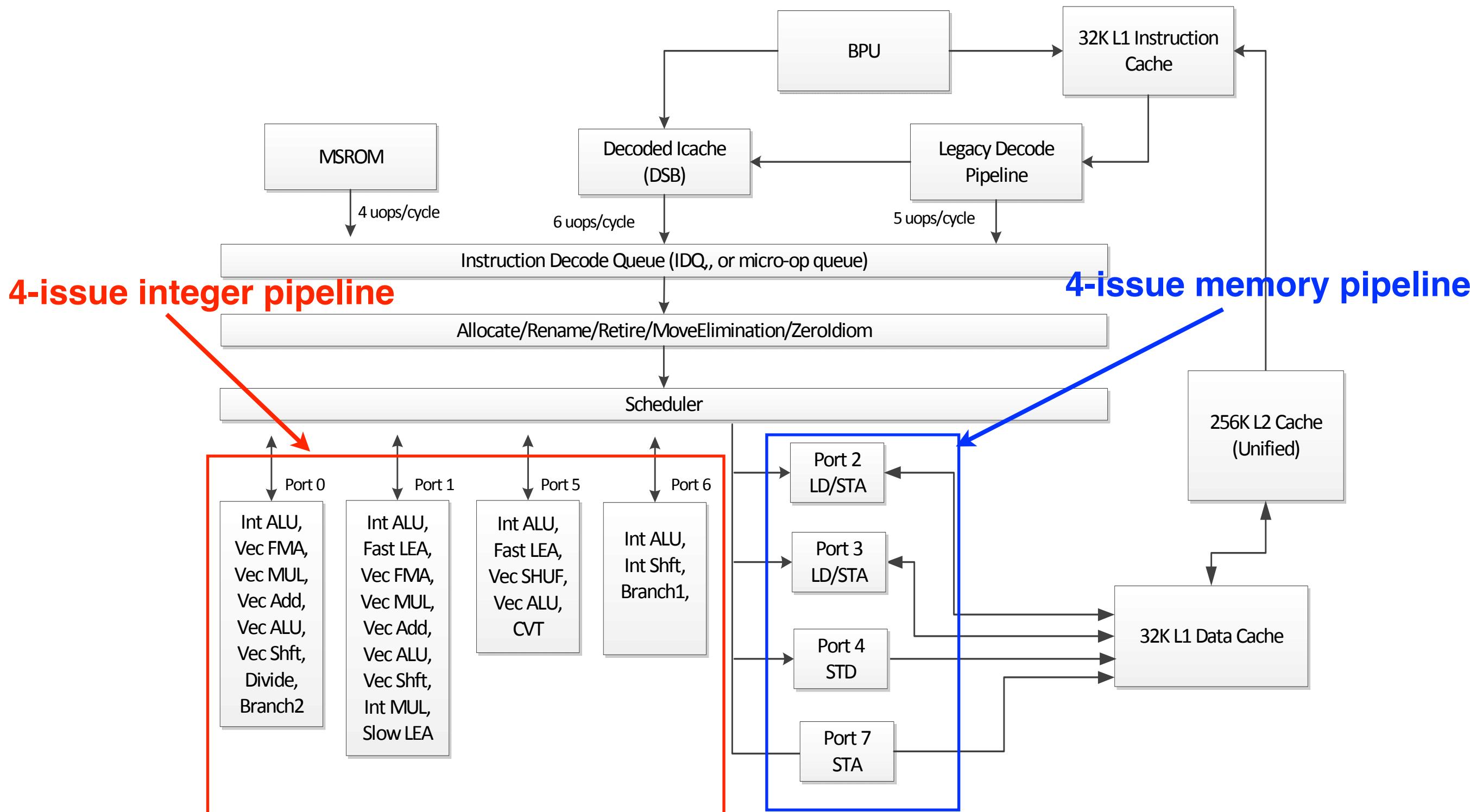
← taibd

The pipelines of Modern Processors

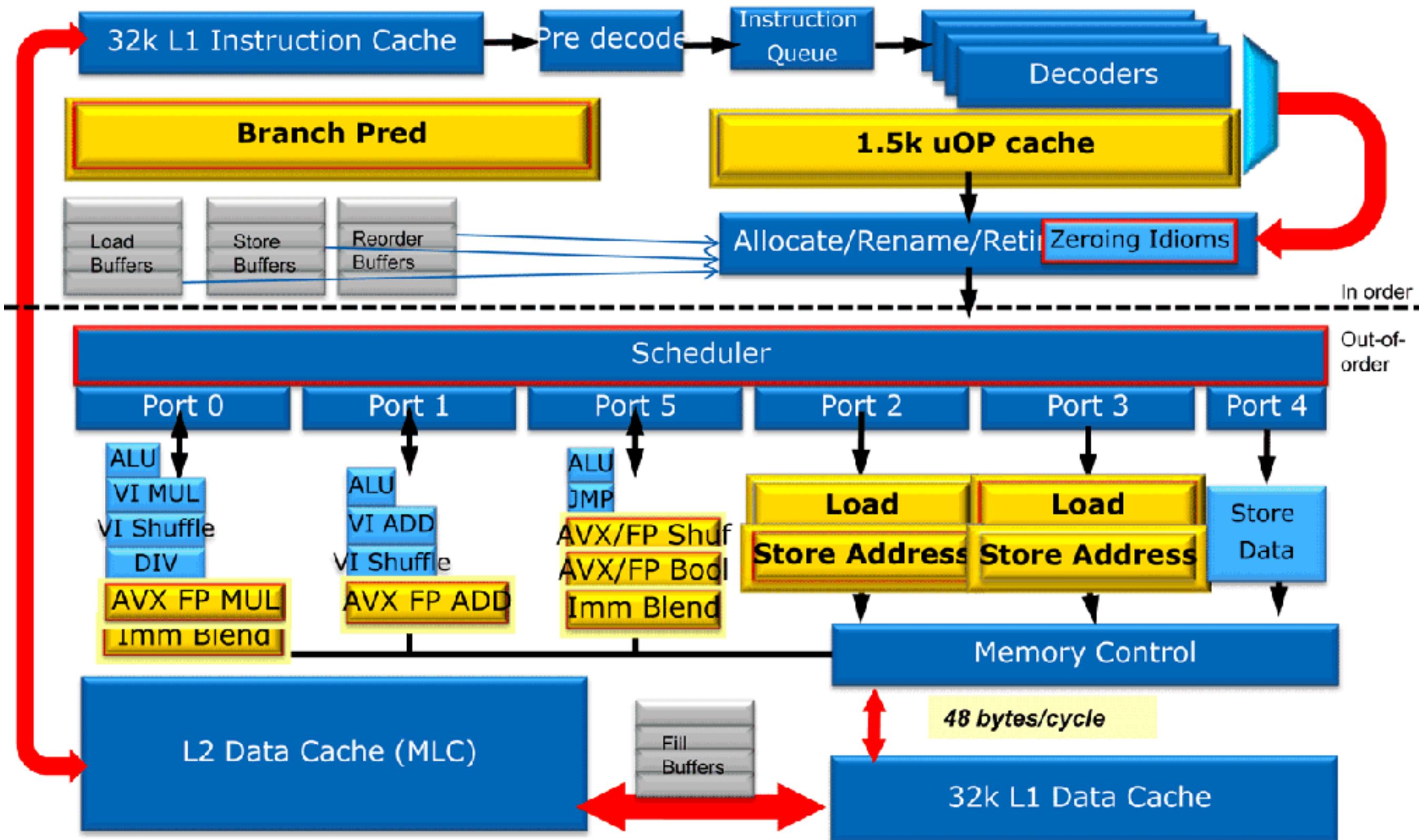
Intel Pentium 4

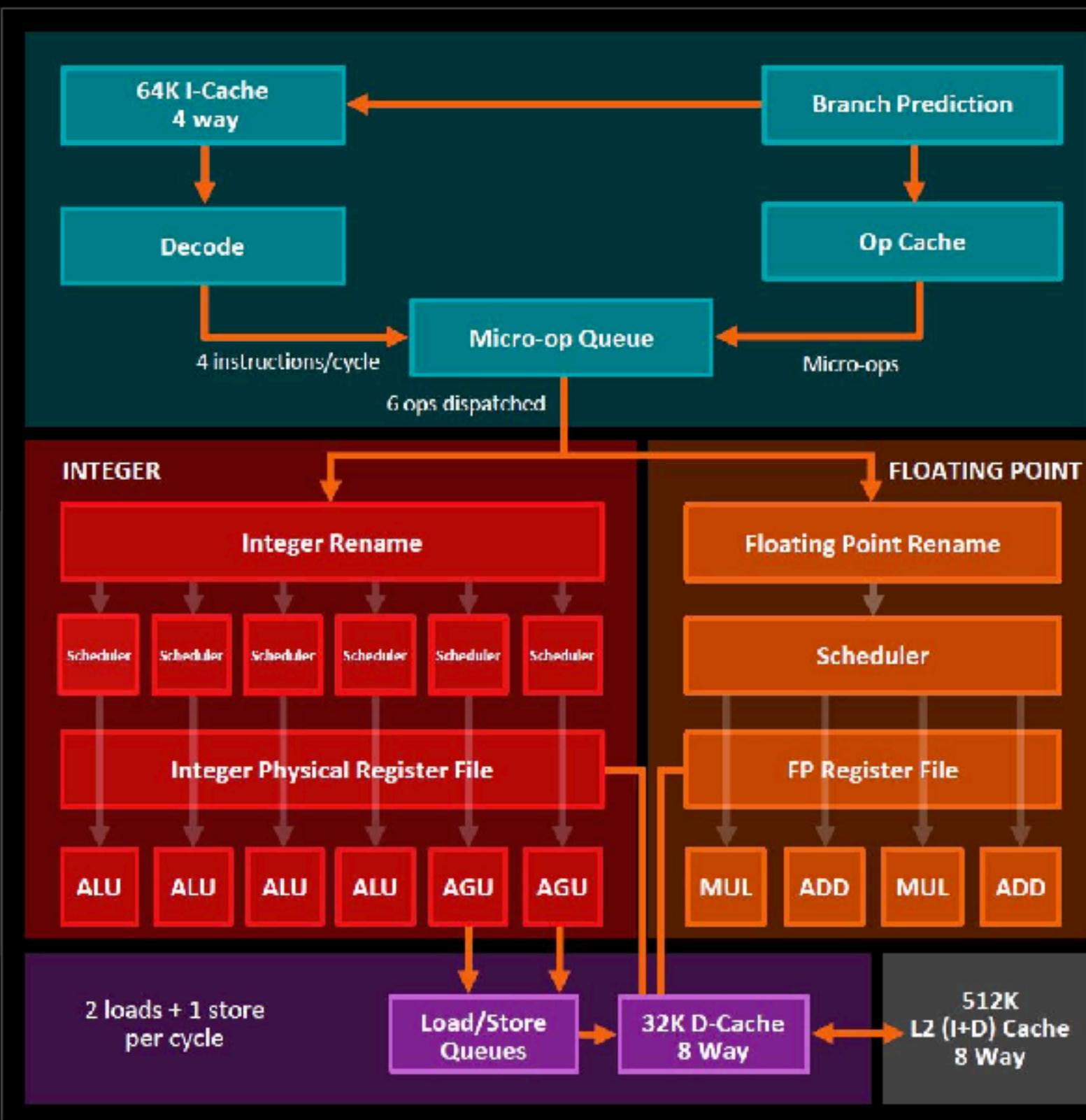


Intel Skylake



Intel Sandy Bridge



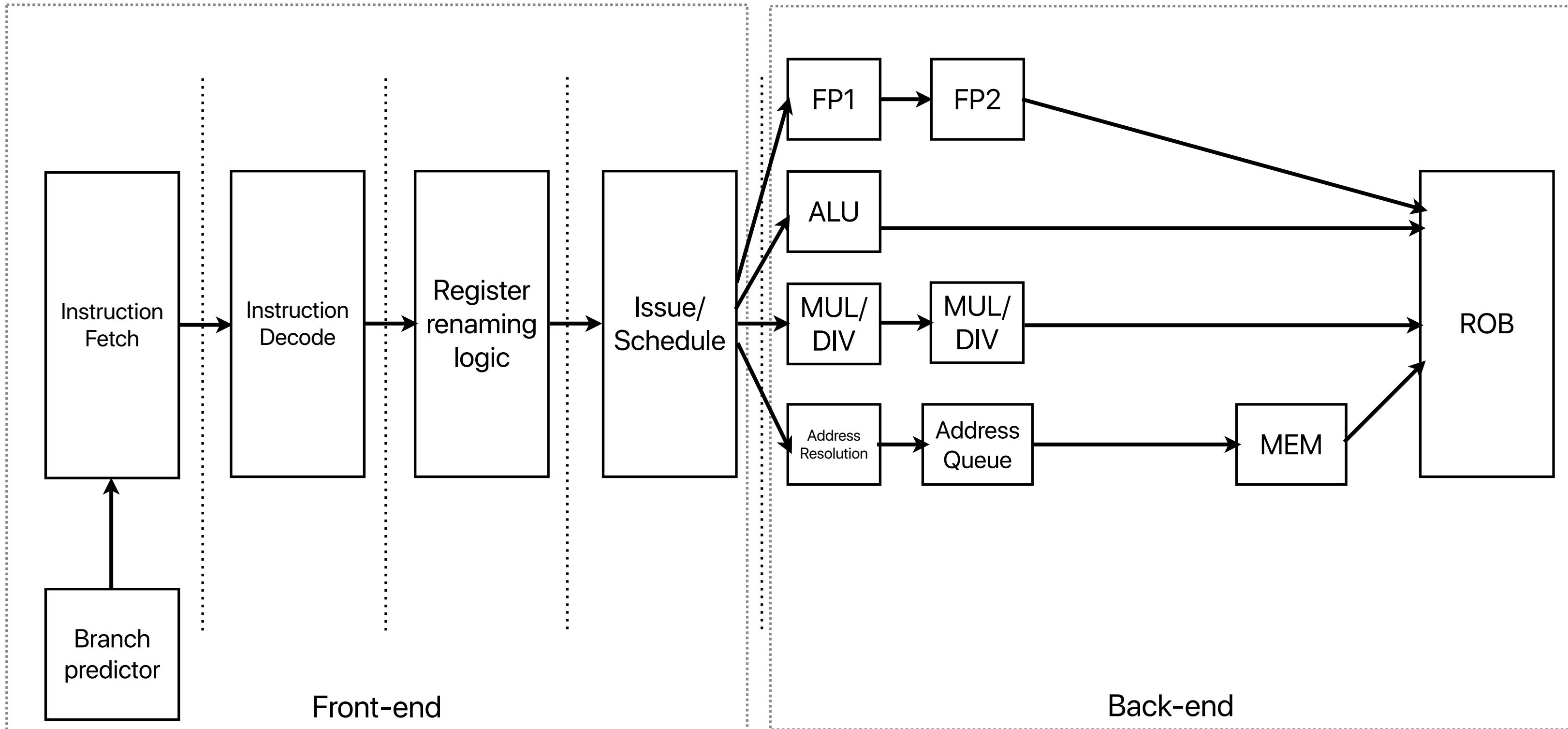


ZEN MICROARCHITECTURE

- ▲ 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"

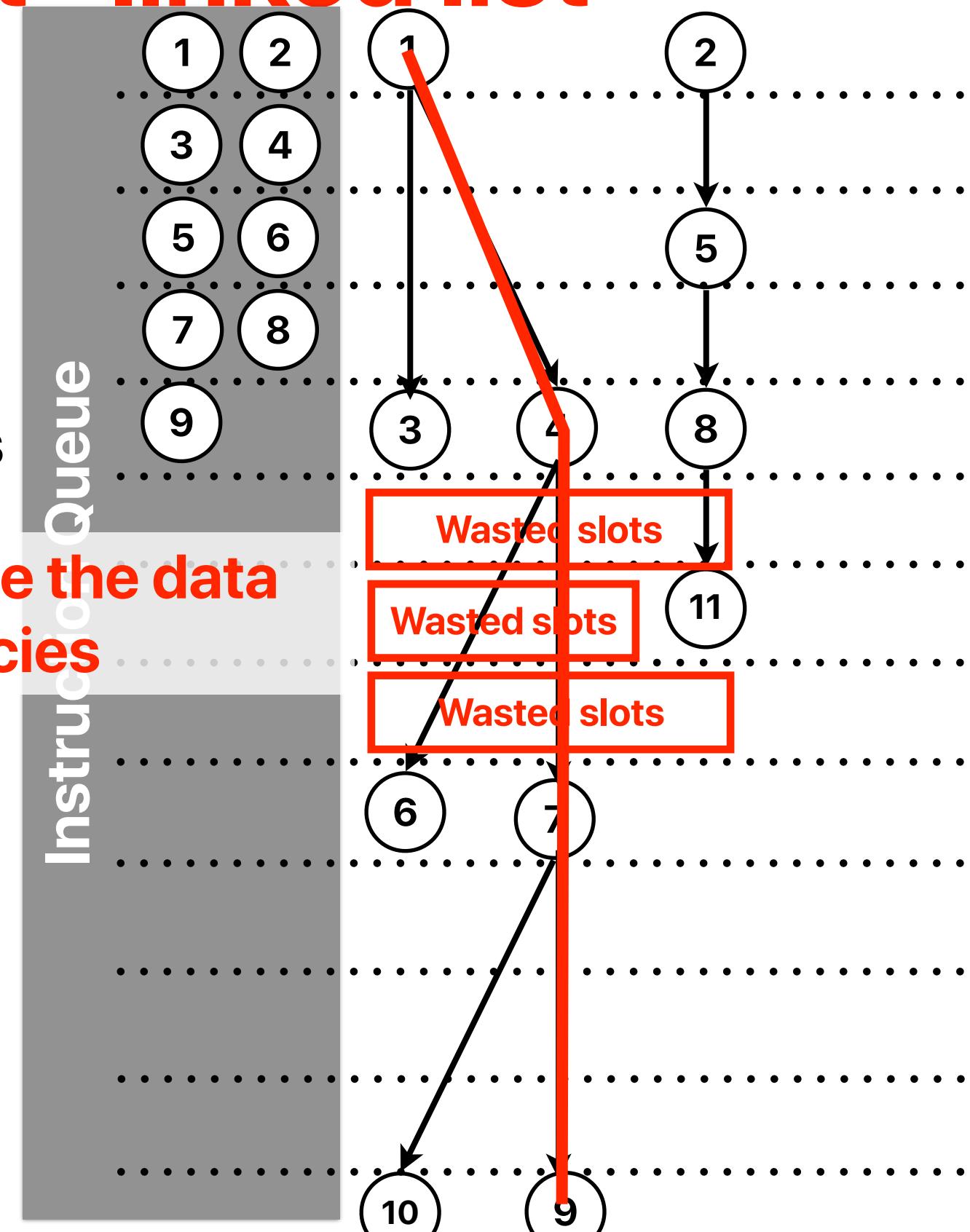
Static instructions

```
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.