Modern Processor Design (IV): Try everything

Hung-Wei Tseng

Missing opportunities

```
for(i = 0; i < count; i++) {
    int64_t temp = a[i];
    a[i] = b[i];
    b[i] = temp;
}
```

Processor can predict what should happen and unroll the loop "dynamically"

```
:
movq    (%rcx,%rax), %r8
movq    (%rdi,%rax), %rsi
addq    $8, %rax
movq    %r8, -8(%rdi,%rax)
movq    %rsi, -8(%rcx,%rax)
cmpq    %r9, %rax
jne     .L9
movq    (%rcx,%rax), %r8
movq    (%rdi,%rax), %rsi
addq    $8, %rax
movq    %r8, -8(%rdi,%rax)
movq    %rsi, -8(%rcx,%rax
cmpq    %r9, %rax
jne     .L9
```

```
.L9:
①  movq    (%rcx,%rax), %r8
②  movq    (%rdi,%rax), %rsi
③  addq    $8, %rax
④  movq    %r8, -8(%rdi,%rax)
⑤  movq    %rsi, -8(%rcx,%rax
⑥  movq    (%rcx,%rax), %r8
⑦  movq    (%rdi,%rax), %rsi
⑧  cmpq    %r9, %rax
⑨  jne     .L9
⑩  addq    $8, %rax
⑪  movq    %r8, -8(%rdi,%rax)
⑫  movq    %rsi, -8(%rcx,%rax
⑬  cmpq    %r9, %rax
⑭  jne     .L9
```

	IF	ID	ALU/BR/AG	M1	M2	M3	M4/XORL	WB/Retire
1	(1)							
2	(2)	(1)						
3	(3)	(2)	(1)					
4	(4)	(3)	(2)	(1)				
5	(5)	(4)	(3)	(2)	(1)			
6	(5)	(4)			(2)	(1)		
7	(5)	(4)				(2)	(1)	
8	[illegible]	[illegible]	(4)				(2)	(1)
9	(7)	(6)	(5)	(4)				(2)
10	(8)	(7)	(6)	(5)	(4)			(3)
11	(9)	(8)	(7)	(6)	(5)	(4)		
12	(10)	(9)	(8)	(7)	(6)	(5)	(4)	
13	(11)	(10)	(9)		(7)	(6)	(5)	(4)
14	(12)	(11)	(10)			(7)	(6)	(5)
15	(13)	(12)	(11)	(10)			(7)	(6)
16	(14)	(13)	(12)	(11)	(10)			(7)
17		(14)	(13)	(12)	(11)	(10)		(8)
18			(14)	(13)	(12)	(11)	(10)	(9)
19				(14)	(13)	(12)	(11)	(10)
20					(14)	(13)	(12)	(11)
21						(14)	(13)	(12)
22							(14)	(13)
23								(14)

7 cycles for 7 instructions

CPI = 1

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

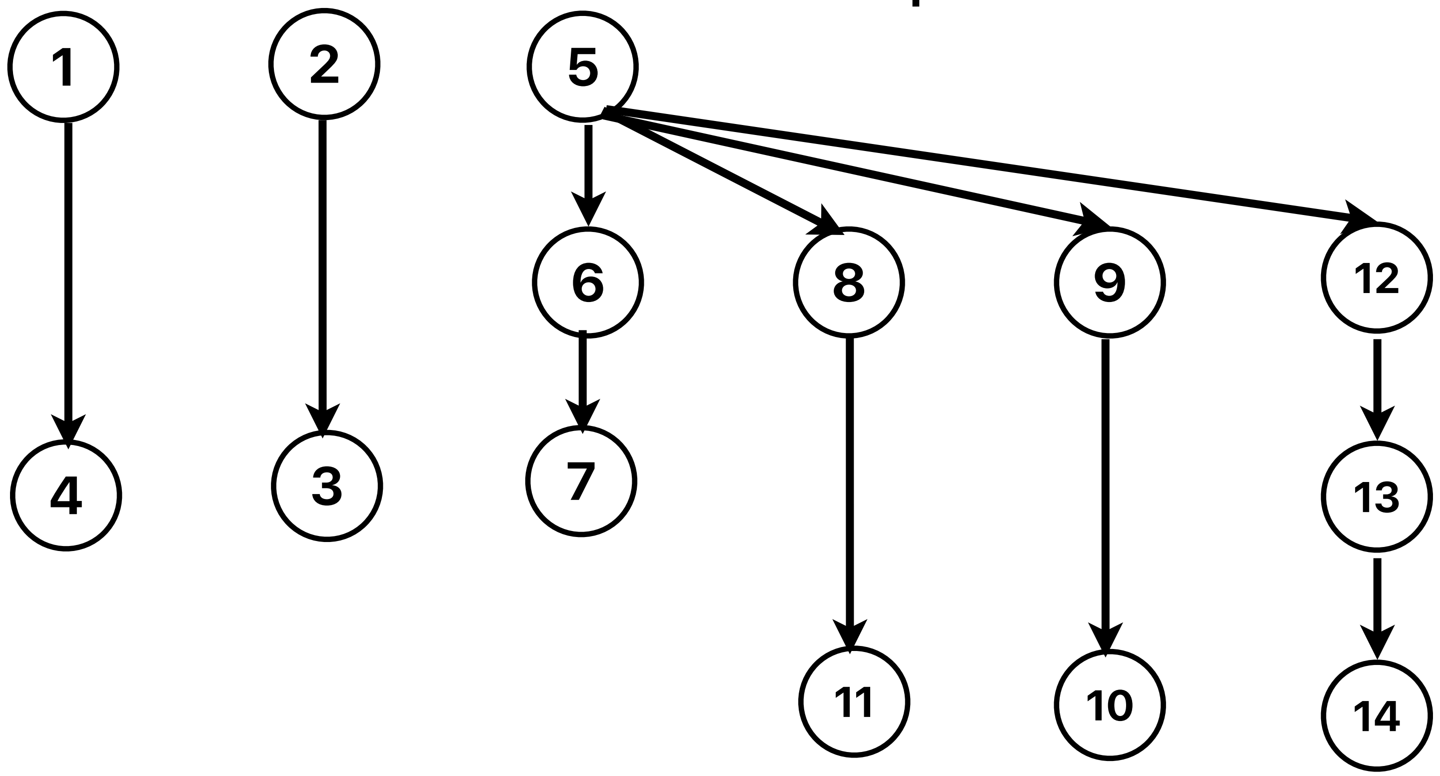
What do you need to execution an instruction?

- 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 movq    (%rdi,%rax), %rsi
② movq    (%rcx,%rax), %r8
③ movq    %r8, (%rdi,%rax)
④ movq    %rsi, (%rcx,%rax)
⑤ addq    $8, %rax
⑥ cmpq    %r9, %rax
⑦ jne     .L9
⑧ movq    (%rdi,%rax), %rsi
⑨ movq    (%rcx,%rax), %r8
⑩ movq    %r8, (%rdi,%rax)
⑪ movq    %rsi, (%rcx,%rax)
⑫ addq    $8, %rax
⑬ cmpq    %r9, %rax
⑭ jne     .L9
```

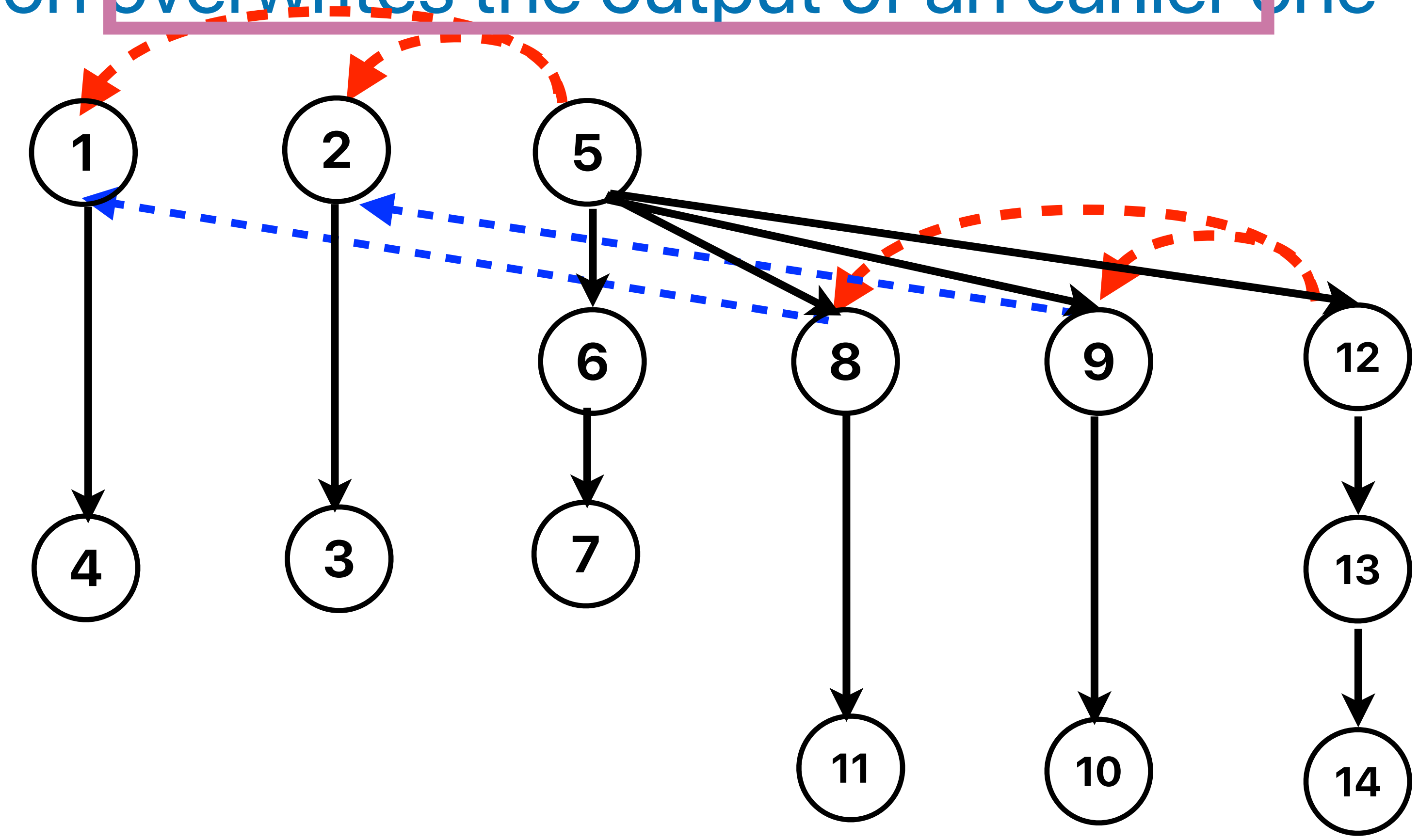


- **In theory**, instructions without dependencies can be executed in parallel or out-of-order
- Instructions with dependencies (on the same path) can never be reordered

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
    - 5 and 1, 5 and 2, 12 and 8, 12 and 9
  - WAW (Write After Write): a later instruction overwrites the output of an earlier one
    - 8 and 1
      9 and 2

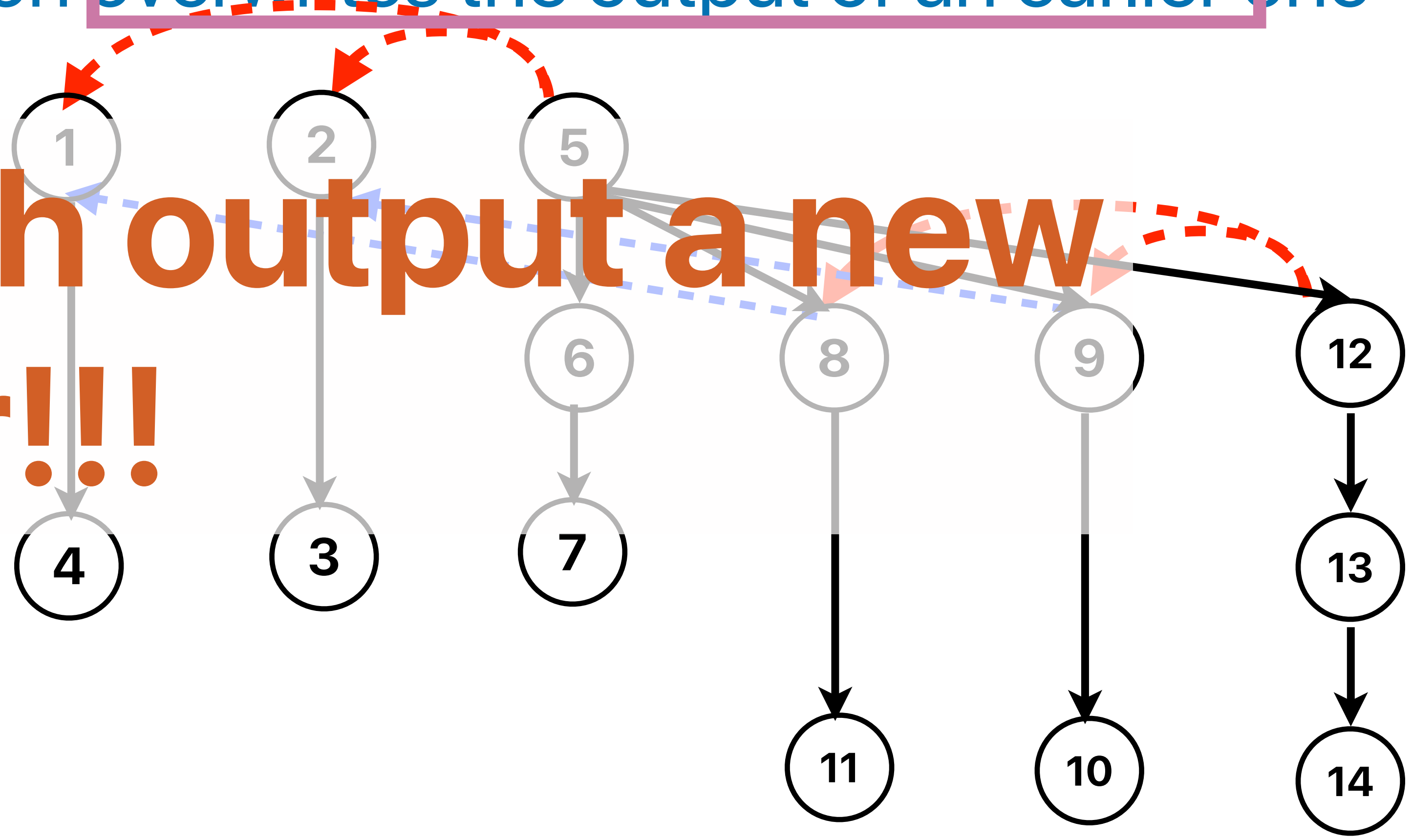
```
①  movq   (%rdi,%rax), %rsi
②  movq   (%rcx,%rax), %r8
③  movq   %r8, (%rdi,%rax)
④  movq   %rsi, (%rcx,%rax)
⑤  addq   $8, %rax
⑥  cmpq   %r9, %rax
⑦  jne    .L9
⑧  movq   (%rdi,%rax), %rsi
⑨  movq   (%rcx,%rax), %r8
⑩  movq   %r8, (%rdi,%rax)
⑪  movq   %rsi, (%rcx,%rax)
⑫  addq   $8, %rax
⑬  cmpq   %r9, %rax
⑭  jne    .L9
```



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
    - 5 and 1, 5 and 2, 12 and 8, 12 and 9
  - WAW (Write After Write): a later instruction overwrites the output of an earlier one
    - 8 and 1
    - 9 and 2

```
①  movq   (%rdi,%rax), %rsi
②  movq   (%rcx,%rax), %r8
③  movq   %r8, (%rdi,%rax)
④  movq   %rsi, (%rcx,%rax)
⑤  addq   $8, %rax
⑥  cmpq   %r9, %rax
⑦  jne    .L9
⑧  movq   (%rdi,%rax), %rsi
⑨  movq   (%rcx,%rax), %r8
⑩  movq   %r8, (%rdi,%rax)
⑪  movq   %rsi, (%rcx,%rax)
⑫  addq   $8, %rax
⑬  cmpq   %r9, %rax
⑭  jne    .L9
```



We need to give each output a new register!!!

Takeaways: data hazards

- More data dependencies, more likelihood of data hazards
- Stalls and data forwarding can both address data hazards to generate correct code execution results — but not very efficient
- Compiler optimizations can help, but to a limited extent
- False dependencies limits the freedom of out-of-order execution

The mechanism of OoO: Register renaming + speculative execution

- K. C. Yeager, "The MIPS R10000 superscalar microprocessor," in IEEE Micro, vol. 16, no. 2, pp. 28-41, April 1996.

Register renaming + OoO

- Redirecting the output of an instruction instance to a **physical register**
- Redirecting inputs of an instruction instance from **architectural registers** to correct **physical registers**
  - You need a mapping table between architectural and physical registers
  - You may also need reference counters to reclaim physical registers
- OoO: Executing an instruction all operands are ready (the values of depending physical registers are generated)
  - You will need an **issue logic** to **issue** an instruction to the target functional unit

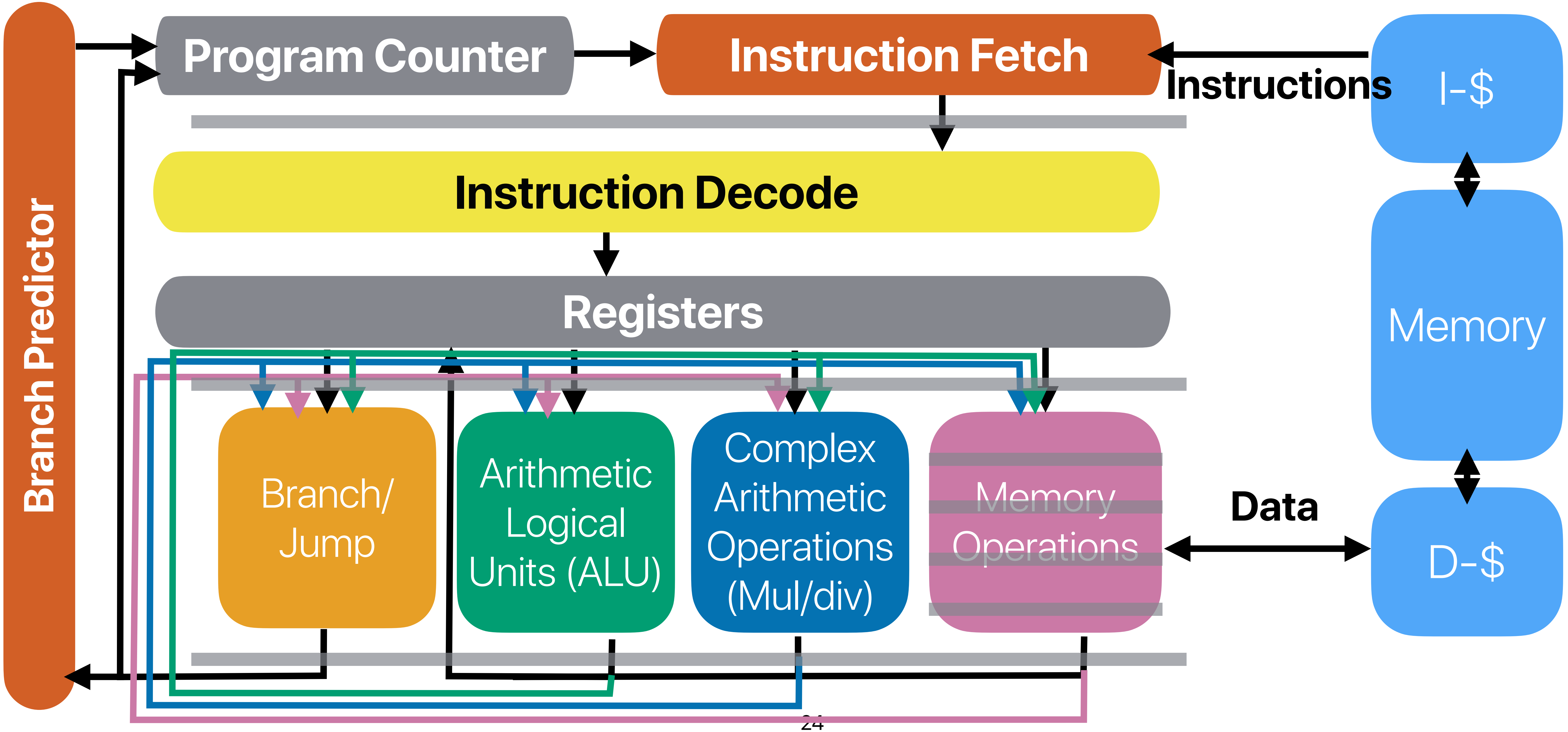
Can we really execute instructions OoO?

- Exceptions may occur anytime — divided by 0, page fault
  - A later instruction cannot write back its own result otherwise the architectural states won't be correct
  - Instructions after the one causes the exception should not be executed
- Hardware can schedule instruction across branch instructions with the help of branch prediction
  - Fetch instructions according to the branch prediction
  - However, branch predictor can never be perfect

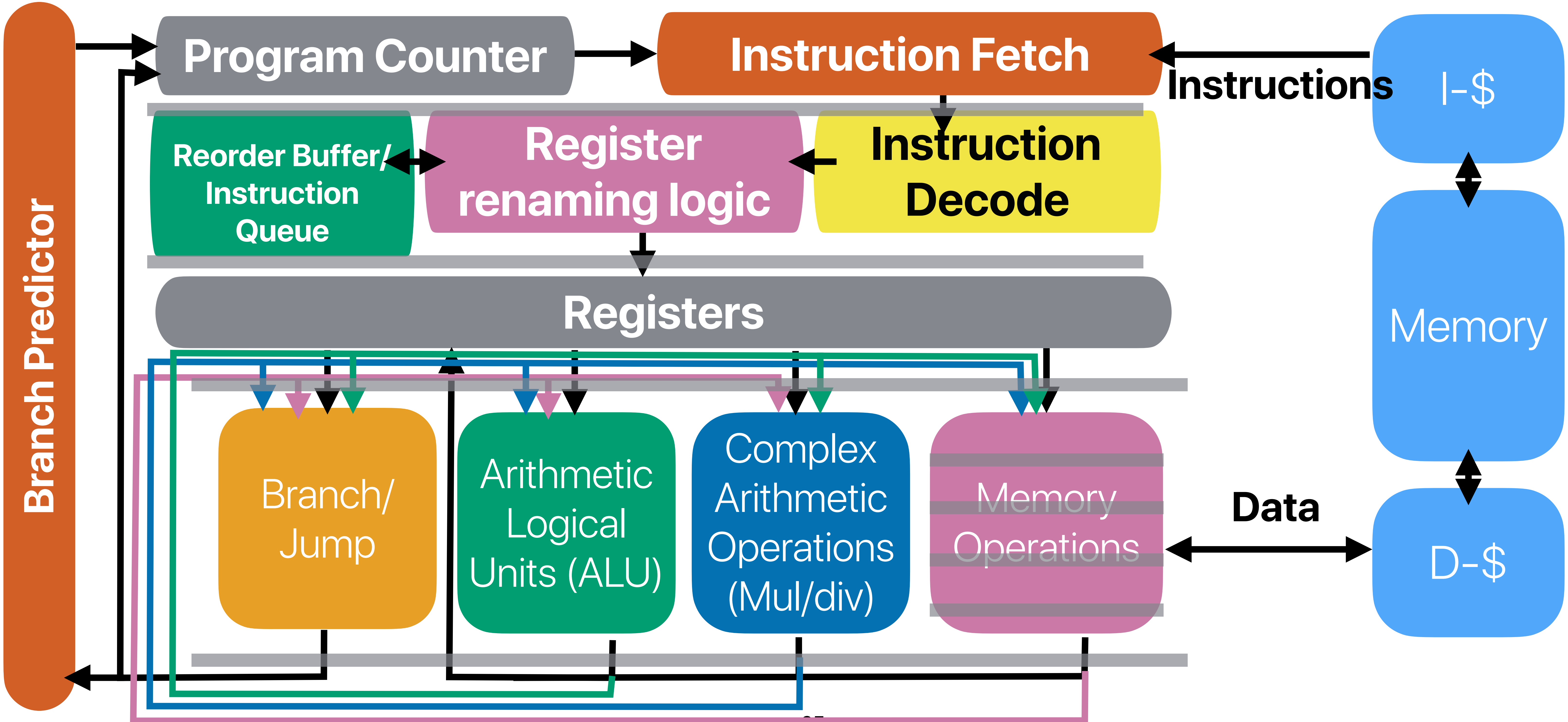
Speculative Execution

- **Speculative** execution mode: an executing instruction is considered as **speculative** before the processor hasn't determined if the instruction should be executed or not
- Reorder buffer (ROB)
  - The processor allocates an entry for each instruction in a reorder buffer
  - Store results in **reorder buffer and physical registers** when the instruction is still speculative
  - If an earlier instruction failed to commit due to an exception or mis-prediction, the physical registers and all ROB entries after the failed-to-commit instruction are flushed
- Commit/Retire
  - Present the execution result to the running program and in architectural registers when all prior instructions are non-speculative
  - Release the ROB entry

Data “forwarding”



Register renaming + OoO + RoB



Register renaming

```
① movq (%rdi,%rax), %rsi
② movq (%rcx,%rax), %r8
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												

	Physical Register
rax	
rcx	
rdi	
rsi	
r8	

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

Register renaming

```
① movq (%rdi,%rax), %rsi
② movq (%rcx,%rax), %r8
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne .L9
```

→ P1

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												

	Physical Register
rax	
rcx	
rdi	
rsi	P1
r8	

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

Register renaming

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												

	Physical Register
rax	
rcx	
rdi	
rsi	P1
r8	P2

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

Register renaming

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												

	Physical Register
rax	
rcx	
rdi	
rsi	P1
r8	P2

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

Register renaming

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8												
9												
10												
11												
12												
13												
14												
15												
16												

	Physical Register
rax	
rcx	
rdi	
rsi	P1
r8	P2

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

Register renaming

```
① movq (%rdi,%rax), %rsi   → P1
② movq (%rcx,%rax), %r8    → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax            → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8												
9												
10												
11												
12												
13												
14												
15												
16												

	Physical Register
rax	P3
rcx	
rdi	
rsi	P1
r8	P2

	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

① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8 → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax → P3
⑥ cmpq %r9, %rax
⑦ jne .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne .L9

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9												
10												
11												
12												
13												
14												
15												
16												

Instruction (5) is running ahead of (3)

	Physical Register
rax	P3
rcx	
rdi	
rsi	P1
r8	P2

	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

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax          → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

Instruction (4) is running ahead of (3)

Instruction (5) is running ahead of (3)

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				(1)(5)
10												
11												
12												
13												
14												
15												
16												

	Physical Register
rax	P3
rcx	
rdi	
rsi	P1
r8	P2

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

Register renaming

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax          → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10												
11												
12												
13												
14												
15												
16												

Retire/Commit (1)

	Physical Register
rax	P3
rcx	
rdi	
rsi	P1
r8	P2

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

Register renaming

```
①  movq (%rdi,%rax), %rsi → P1
②  movq (%rcx,%rax), %r8  → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax          → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi → P4
⑨  movq (%rcx,%rax), %r8
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax
⑬  cmpq %r9, %rax
⑭  jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							(2)(5)
11												
12												
13												
14												
15												
16												

	Physical Register
rax	P3
rcx	
rdi	
rsi	P4
r8	P2

	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				P10			

Register renaming

① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8 → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax → P3
⑥ cmpq %r9, %rax
⑦ jne .L9
⑧ movq (%rdi,%rax), %rsi → P4
⑨ movq (%rcx,%rax), %r8 → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne .L9

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12												
13												
14												
15												
16												

	Physical Register
rax	P3
rcx	
rdi	
rsi	P4
r8	P5

	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

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax          → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi → P4
⑨ movq (%rcx,%rax), %r8  → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13												
14												
15												
16												

	Physical Register
rax	P3
rcx	
rdi	
rsi	P4
r8	P5

	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

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax          → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi → P4
⑨ movq (%rcx,%rax), %r8  → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14												
15												
16												

	Physical Register
rax	P3
rcx	
rdi	
rsi	P4
r8	P5

	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

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8 → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax → P3
⑥ cmpq %r9, %rax
⑦ jne .L9
⑧ movq (%rdi,%rax), %rsi → P4
⑨ movq (%rcx,%rax), %r8 → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax → P6
⑬ cmpq %r9, %rax
⑭ jne .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15												
16												

	Physical Register
rax	P6
rcx	
rdi	
rsi	P4
r8	P5

	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

```
①  movq (%rdi,%rax), %rsi → P1
②  movq (%rcx,%rax), %r8  → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax          → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi → P4
⑨  movq (%rcx,%rax), %r8  → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax          → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			(3)(4)(5)(6)(7)
16												

	Physical Register
rax	P6
rcx	
rdi	
rsi	P4
r8	P5

	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

```
① movq (%rdi,%rax), %rsi   → P1
② movq (%rcx,%rax), %r8    → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax            → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi   → P4
⑨ movq (%rcx,%rax), %r8    → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax            → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)

	Physical Register
rax	P6
rcx	
rdi	
rsi	P4
r8	P5

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

Register renaming

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8  → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax          → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi → P4
⑨ movq (%rcx,%rax), %r8  → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax          → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18												
19												
20												
21												
22												
23												

Register renaming

```
①  movq (%rdi,%rax), %rsi → P1
②  movq (%rcx,%rax), %r8  → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax          → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi → P4
⑨  movq (%rcx,%rax), %r8  → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax          → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				(8)(12)(13)(14)
19												
20												
21												
22												
23												

Register renaming

```
① movq (%rdi,%rax), %rsi  → P1
② movq (%rcx,%rax), %r8   → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax           → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi  → P4
⑨ movq (%rcx,%rax), %r8   → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax           → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				~~(8)~~(12)(13)(14)
19	(19)	(18)	(15)(16)(17)	(10)	(11)							(9)(12)(13)(14)
20												
21												
22												
23												

Register renaming

```
①  movq (%rdi,%rax), %rsi  → P1
②  movq (%rcx,%rax), %r8   → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax           → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi  → P4
⑨  movq (%rcx,%rax), %r8   → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax           → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				~~(8)~~(12)(13)(14)
19	(19)	(18)	(15)(16)(17)	(10)	(11)							~~(9)~~(12)(13)(14)
20	(20)	(19)	(16)(17)(18)	(15)	(10)	(11)						(12)(13)(14)
21												
22												
23												

Register renaming

```
①  movq (%rdi,%rax), %rsi  → P1
②  movq (%rcx,%rax), %r8   → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax           → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi  → P4
⑨  movq (%rcx,%rax), %r8   → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax           → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				~~(8)~~(12)(13)(14)
19	(19)	(18)	(15)(16)(17)	(10)	(11)							~~(9)~~(12)(13)(14)
20	(20)	(19)	(16)(17)(18)	(15)	(10)	(11)						(12)(13)(14)
21	(21)	(20)	(17)(18)(19)	(16)	(15)	(10)	(11)					(12)(13)(14)
22												
23												

Register renaming

```
①  movq (%rdi,%rax), %rsi  → P1
②  movq (%rcx,%rax), %r8   → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax           → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi  → P4
⑨  movq (%rcx,%rax), %r8   → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax           → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

Only 1 of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				~~(8)~~(12)(13)(14)
19	(19)	(18)	(15)(16)(17)	(10)	(11)							~~(9)~~(12)(13)(14)
20	(20)	(19)	(16)(17)(18)	(15)	(10)	(11)						(12)(13)(14)
21	(21)	(20)	(17)(18)(19)	(16)	(15)	(10)	(11)					(12)(13)(14)
22		(21)	(17)(18)(20)		(16)	(15)	(10)	(11)	(19)			(12)(13)(14)
23												

Register renaming

Only 1 of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi → P1
②  movq (%rcx,%rax), %r8  → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax          → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi → P4
⑨  movq (%rcx,%rax), %r8  → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax          → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				~~(8)~~(12)(13)(14)
19	(19)	(18)	(15)(16)(17)	(10)	(11)							~~(9)~~(12)(13)(14)
20	(20)	(19)	(16)(17)(18)	(15)	(10)	(11)						(12)(13)(14)
21	(21)	(20)	(17)(18)(19)	(16)	(15)	(10)	(11)					(12)(13)(14)
22		(21)	(17)(18)(20)		(16)	(15)	(10)	(11)	(19)			(12)(13)(14)
23			(17)(20)(21)	(18)		(16)	(15)	(10)				(11)(12)(13)(14)(19)
24												
25												

Register renaming

Only 1 of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi  → P1
②  movq (%rcx,%rax), %r8   → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax           → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi  → P4
⑨  movq (%rcx,%rax), %r8   → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax           → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				~~(8)~~(12)(13)(14)
19	(19)	(18)	(15)(16)(17)	(10)	(11)							~~(9)~~(12)(13)(14)
20	(20)	(19)	(16)(17)(18)	(15)	(10)	(11)						(12)(13)(14)
21	(21)	(20)	(17)(18)(19)	(16)	(15)	(10)	(11)					(12)(13)(14)
22		(21)	(17)(18)(20)		(16)	(15)	(10)	(11)	(19)			(12)(13)(14)
23			(17)(20)(21)	(18)		(16)	(15)	(10)				(11)(12)(13)(14)(19)
24			(20)(21)	(17)	(18)		(16)	(15)				(10)(11)(12)(13)(14)(19)
25												

Register renaming

```
① movq (%rdi,%rax), %rsi  → P1
② movq (%rcx,%rax), %r8   → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax           → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi  → P4
⑨ movq (%rcx,%rax), %r8   → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax           → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

Only 1 of them can have a instruction at the same cycle

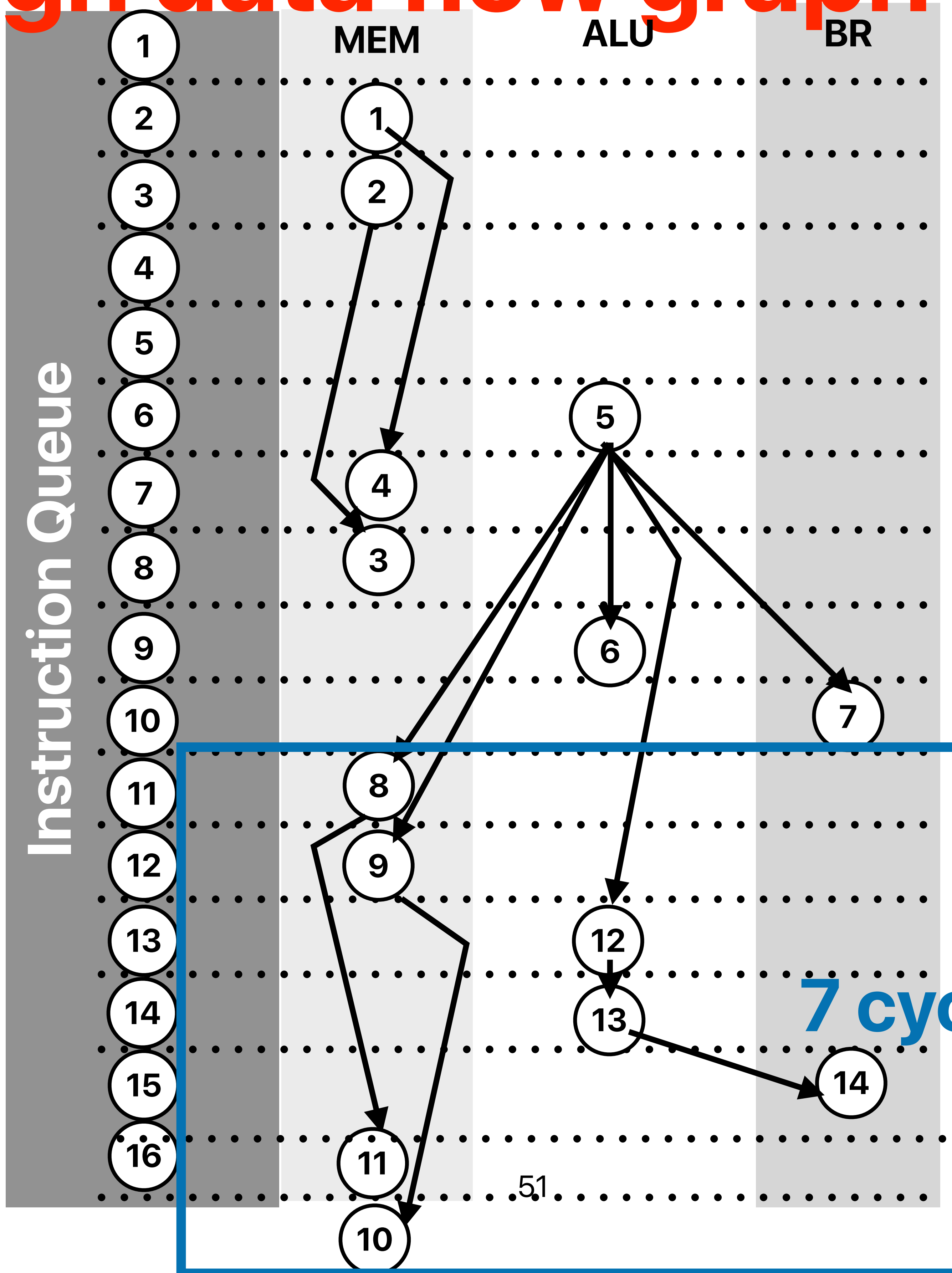
	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)											
2	(2)	(1)										
3	(3)	(2)	(1)									
4	(4)	(3)	(2)	(1)								
5	(5)	(4)	(3)	(2)	(1)							
6	(6)	(5)	(3)(4)		(2)	(1)						
7	(7)	(6)	(3)(4)(5)			(2)	(1)					
8	(8)	(7)	(3)(4)(6)				(2)	(1)	(5)			
9	(9)	(8)	(3)(6)(7)	(4)				(2)				~~(1)~~(5)
10	(10)	(9)	(6)(7)(8)	(3)	(4)							~~(2)~~(5)
11	(11)	(10)	(7)(8)(9)		(3)	(4)			(6)			
12	(12)	(11)	(8)(9)(10)			(3)	(4)				(7)	(5)(6)
13	(13)	(12)	(9)(10)(11)	(8)			(3)	(4)				(5)(6)(7)
14	(14)	(13)	(10)(11)(12)	(9)	(8)			(3)				(4)(5)(6)(7)
15	(15)	(14)	(10)(11)(13)		(9)	(8)			(12)			~~(3)(4)(5)(6)(7)~~
16	(16)	(15)	(10)(11)(14)			(9)	(8)		(13)			(12)
17	(17)	(16)	(10)(11)(15)				(9)	(8)			(14)	(12)(13)
18	(18)	(17)	(10)(15)(16)	(11)				(9)				~~(8)~~(12)(13)(14)
19	(19)	(18)	(15)(16)(17)	(10)	(11)							~~(9)~~(12)(13)(14)
20	(20)	(19)	(16)(17)(18)	(15)	(10)	(11)						(12)(13)(14)
21	(21)	(20)	(17)(18)(19)	(16)	(15)	(10)	(11)					(12)(13)(14)
22		(21)	(17)(18)(20)		(16)	(15)	(10)	(11)	(19)			(12)(13)(14)
23			(17)(20)(21)	(18)		(16)	(15)	(10)				(11)(12)(13)(14)(19)
24			(20)(21)	(17)	(18)		(16)	(15)				~~(10)(11)(12)(13)(14)(19)~~
25			(21)		(17)	(18)		(16)	(20)			(15)(19)

7 cycles for 7 instructions

CPI = 1

Through data flow graph analysis

```
①  movq (%rdi,%rax), %rsi
②  movq (%rcx,%rax), %r8
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi
⑨  movq (%rcx,%rax), %r8
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```



7 cycles every iteration

CPI $= \frac{7}{7} = 1!$

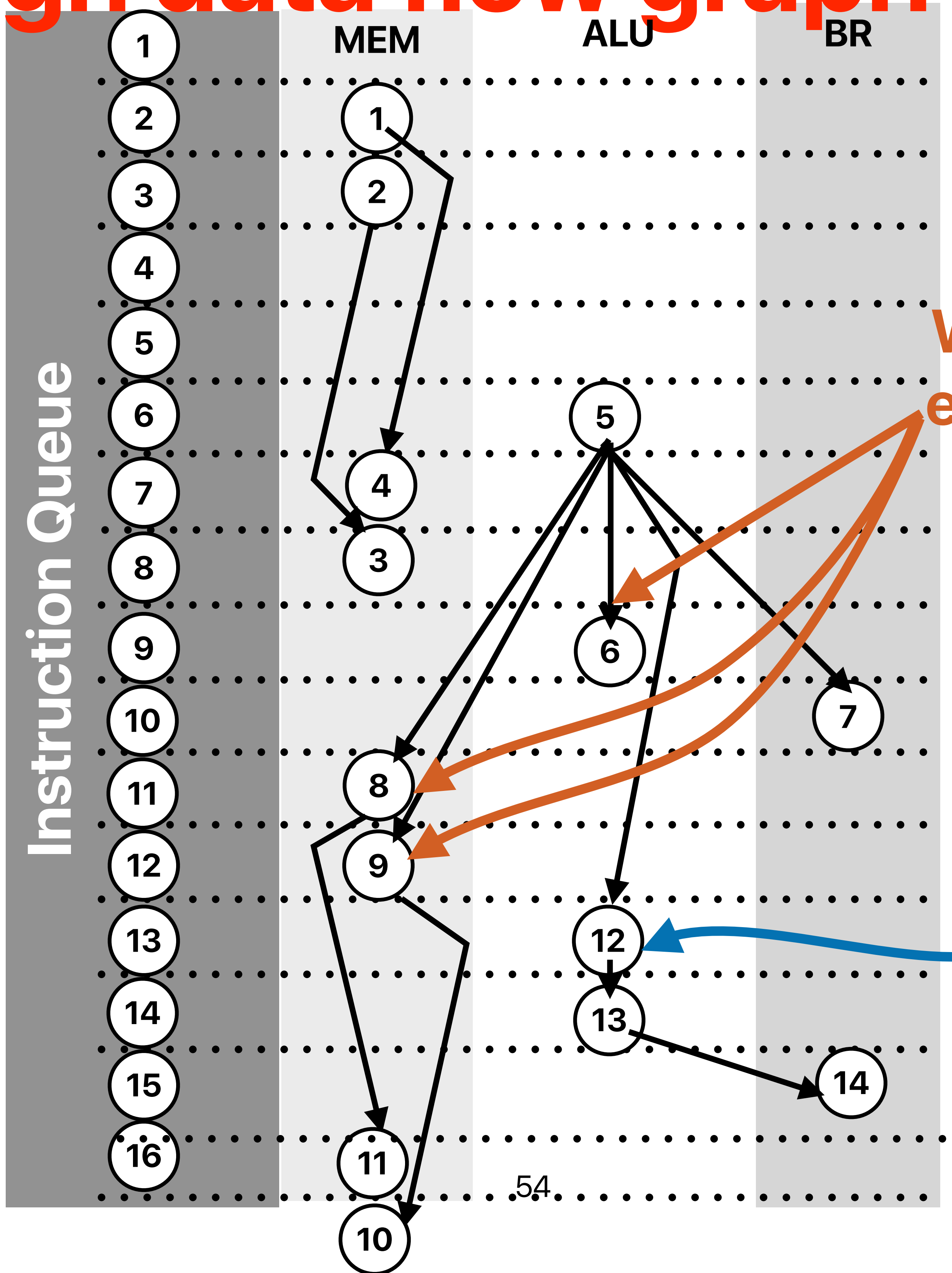
Takeaways: data hazards

- More data dependencies, more likelihood of data hazards
- Stalls and data forwarding can both address data hazards to generate correct code execution results — but not very efficient
- Compiler optimizations can help, but to a limited extent
- False dependencies limits the freedom of out-of-order execution
- Register renaming + Speculative execution enables more efficient execution by dynamically scheduling instructions whenever their data dependencies are resolved

If CPI==1 the limitation?

Through data flow graph analysis

```
①  movq (%rdi,%rax), %rsi
②  movq (%rcx,%rax), %r8
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi
⑨  movq (%rcx,%rax), %r8
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```



We cannot issue them earlier simply because structural hazards!

We could have this executed earlier if it's in the queue earlier

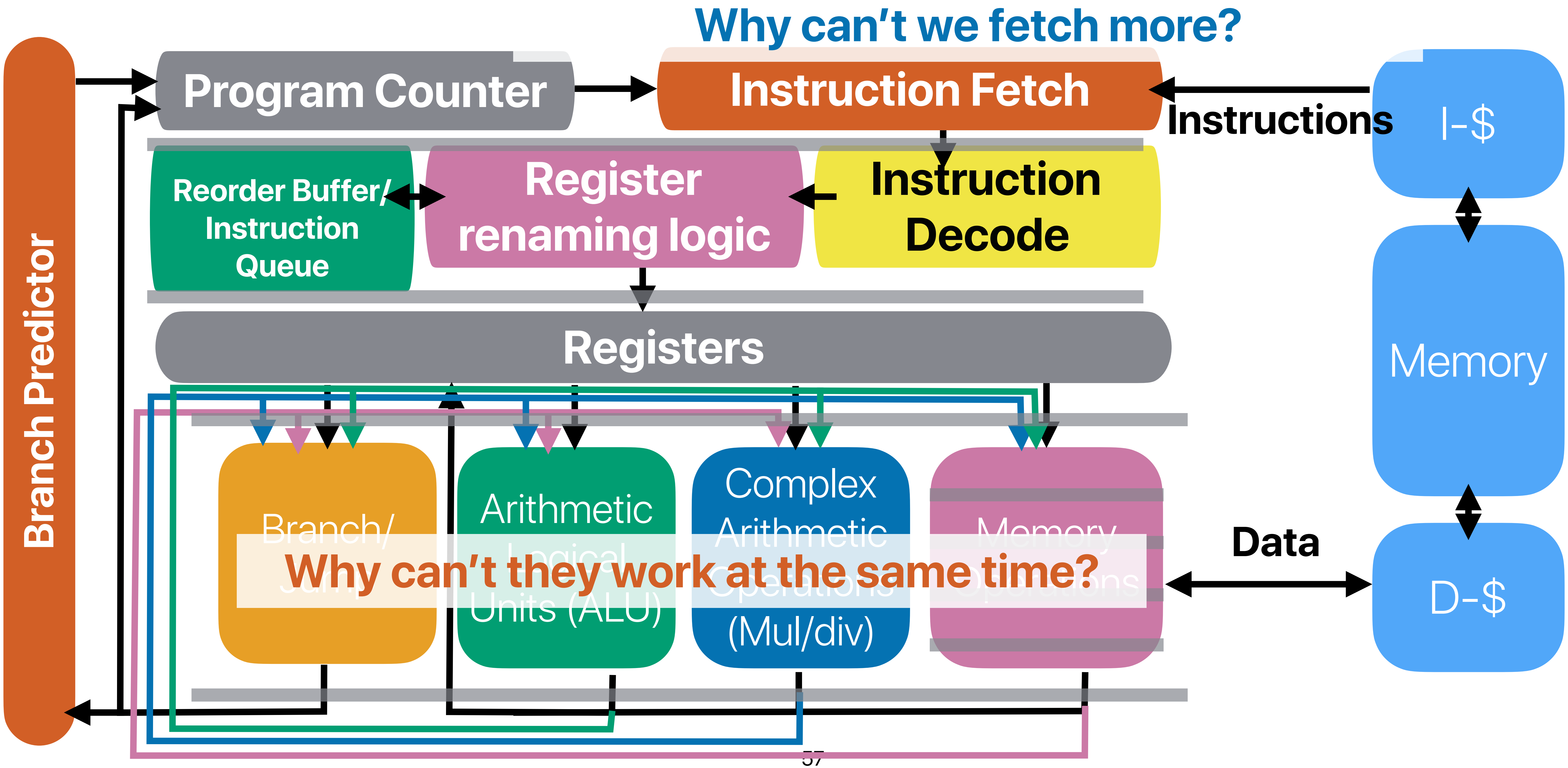
Super Scalar

Superscalar

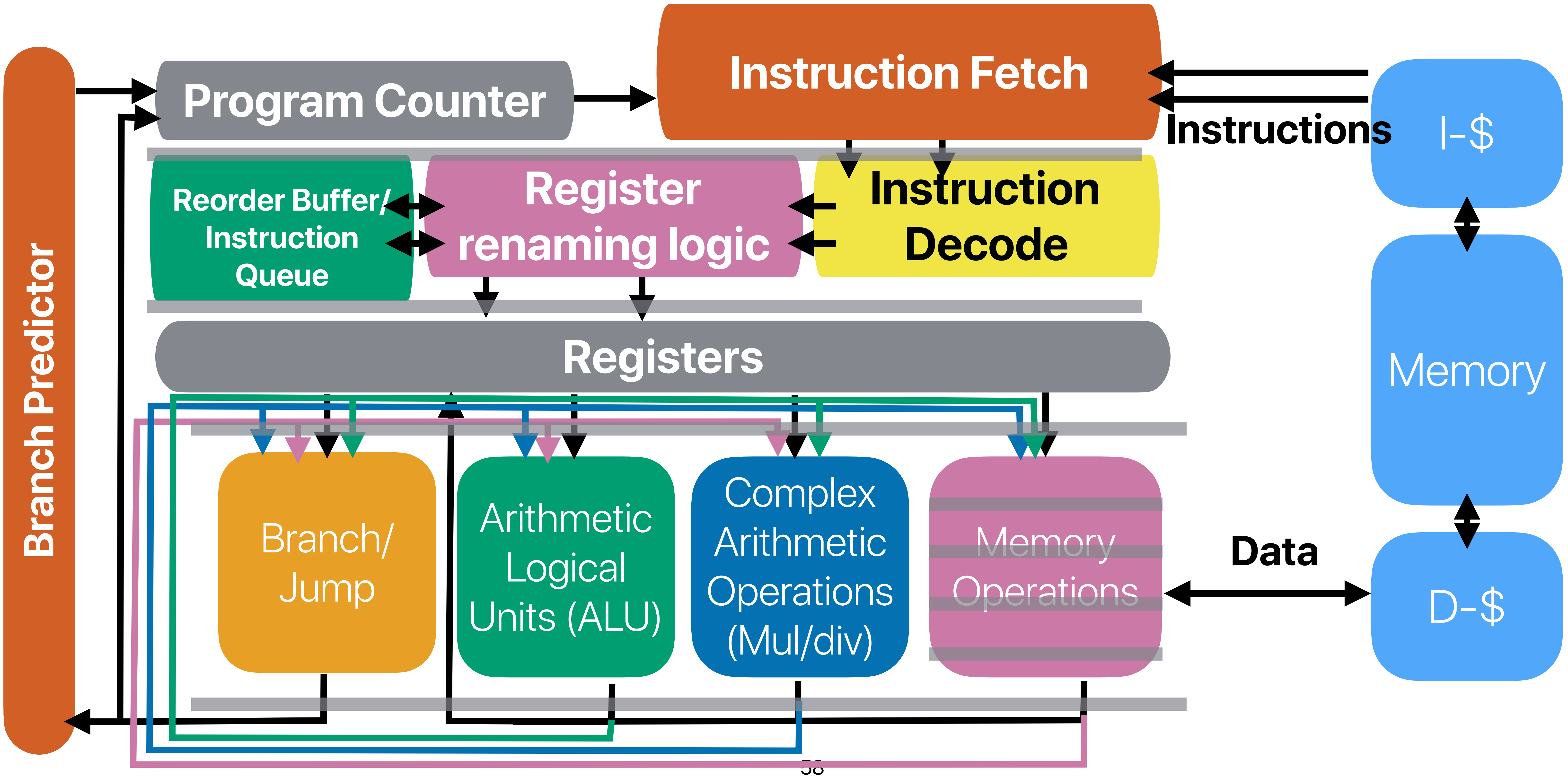
- Since we have many functional units now, we should fetch/decode more instructions each cycle so that we can have more instructions to issue!
- Super-scalar: fetch/decode/issue more than one instruction each cycle
  - **Fetch width:** how many instructions can the processor fetch/decode each cycle
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

$$\frac{1}{min(issue\ width, fetch\ width, decode\ width)}$$

Register renaming + OoO + RoB



Register renaming + SuperScalar



2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi  → P1
②  movq (%rcx,%rax), %r8   → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax           → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi  → P4
⑨  movq (%rcx,%rax), %r8   → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax           → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

```
① movq (%rdi,%rax), %rsi
② movq (%rcx,%rax), %r8
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi
⑨ movq (%rcx,%rax), %r8
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

→ P1 (①)
→ P2 (②)
→ P3 (④)
→ P4 (⑧)
→ P5 (⑨)
→ P6 (⑪)

2 issue: "2" of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
① movq (%rdi,%rax), %rsi → P1
② movq (%rcx,%rax), %r8 → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax → P3
⑥ cmpq %r9, %rax
⑦ jne .L9
⑧ movq (%rdi,%rax), %rsi → P4
⑨ movq (%rcx,%rax), %r8 → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax → P6
⑬ cmpq %r9, %rax
⑭ jne .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi   → P1
②  movq (%rcx,%rax), %r8    → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax            → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi   → P4
⑨  movq (%rcx,%rax), %r8    → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax            → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi   → P1
②  movq (%rcx,%rax), %r8    → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax            → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi   → P4
⑨  movq (%rcx,%rax), %r8    → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax            → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi  → P1
②  movq (%rcx,%rax), %r8   → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax           → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi  → P4
⑨  movq (%rcx,%rax), %r8   → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax           → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi  → P1
②  movq (%rcx,%rax), %r8   → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)  → P3
⑤  addq $8, %rax
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi  → P4
⑨  movq (%rcx,%rax), %r8   → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax           → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: “2” of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi   → P1
②  movq (%rcx,%rax), %r8    → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax            → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi   → P4
⑨  movq (%rcx,%rax), %r8    → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax            → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: “2” of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi   → P1
②  movq (%rcx,%rax), %r8    → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax            → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi   → P4
⑨  movq (%rcx,%rax), %r8    → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax            → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12												
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
① movq (%rdi,%rax), %rsi    → P1
② movq (%rcx,%rax), %r8     → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax             → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi    → P4
⑨ movq (%rcx,%rax), %r8     → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax             → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13												
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: “2” of them can have a instruction at the same cycle

```
① movq (%rdi,%rax), %rsi     → P1
② movq (%rcx,%rax), %r8      → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)     → P3
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi     → P4
⑨ movq (%rcx,%rax), %r8      → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax              → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20)(21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14												
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi   → P1
②  movq (%rcx,%rax), %r8    → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax            → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi   → P4
⑨  movq (%rcx,%rax), %r8    → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax            → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9) (10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11) (12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12) (13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14) (15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16) (17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19) (20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20) (21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14				(16)	(10)	(11)	(15)	(3)	(20)			(4)(5)(6)(7)(8)(9)(12)(13)(14) (19)
15												
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
① movq (%rdi,%rax), %rsi   → P1
② movq (%rcx,%rax), %r8    → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)   → P3
⑤ addq $8, %rax
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi   → P4
⑨ movq (%rcx,%rax), %r8    → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax            → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20)(21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14				(16)	(10)	(11)	(15)	(3)	(20)			(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)
15					(16)	(10)	(11)	(15)			(21)	(3)(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)(20)
16												
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

```
① movq (%rdi,%rax), %rsi    → P1
② movq (%rcx,%rax), %r8     → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax             → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi    → P4
⑨ movq (%rcx,%rax), %r8     → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax             → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

2 issue: "2" of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20)(21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14				(16)	(10)	(11)	(15)	(3)	(20)			(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)
15					(16)	(10)	(11)	(15)			(21)	(3)(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)(20)
16				(17)		(16)	(10)	(11)				(12)(13)(14)(15)(19)(20(21)
17												
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
①  movq (%rdi,%rax), %rsi   → P1
②  movq (%rcx,%rax), %r8    → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax            → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi   → P4
⑨  movq (%rcx,%rax), %r8    → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax            → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20)(21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14				(16)	(10)	(11)	(15)	(3)	(20)			(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)
15					(16)	(10)	(11)	(15)			(21)	(12)(13)(14)(19)(20)
16				(17)		(16)	(10)	(11)				(3)(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)(20)
17					(17)		(16)	(10)				(11)(12)(13)(14)(15)(19)(20(21)
18												
19												
20												

2-issue SS + Register renaming +OoO

2 issue: "2" of them can have a instruction at the same cycle

```
① movq (%rdi,%rax), %rsi  → P1
② movq (%rcx,%rax), %r8   → P2
③ movq %r8, (%rdi,%rax)
④ movq %rsi, (%rcx,%rax)
⑤ addq $8, %rax           → P3
⑥ cmpq %r9, %rax
⑦ jne  .L9
⑧ movq (%rdi,%rax), %rsi  → P4
⑨ movq (%rcx,%rax), %r8   → P5
⑩ movq %r8, (%rdi,%rax)
⑪ movq %rsi, (%rcx,%rax)
⑫ addq $8, %rax           → P6
⑬ cmpq %r9, %rax
⑭ jne  .L9
⑮ movq (%rdi,%rax), %rsi
⑯ movq (%rcx,%rax), %r8
⑰ movq %r8, (%rdi,%rax)
⑱ movq %rsi, (%rcx,%rax)
⑲ addq $8, %rax
⑳ cmpq %r9, %rax
㉑ jne  .L9
```

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20)(21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14				(16)	(10)	(11)	(15)	(3)	(20)			(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)
15					(16)	(10)	(11)	(15)			(21)	(3)(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)(20)
16				(17)		(16)	(10)	(11)				(12)(13)(14)(15)(19)(20(21)
17					(17)		(16)	(10)				(11)(12)(13)(14)(15)(19)(20(21)
18						(17)		(16)				(10)(11)(12)(13)(14)(15)(19)(20)(21)
19												
20												

2-issue SS + Register renaming +OoO

```
①  movq (%rdi,%rax), %rsi      → P1
②  movq (%rcx,%rax), %r8       → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax               → P3
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi      → P4
⑨  movq (%rcx,%rax), %r8       → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax               → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

2 issue: "2" of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	(1)(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			(2)(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20)(21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14				(16)	(10)	(11)	(15)	(3)	(20)			(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)
15					(16)	(10)	(11)	(15)			(21)	(3)(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)(20)
16				(17)		(16)	(10)	(11)				(12)(13)(14)(15)(19)(20(21)
17					(17)		(16)	(10)				(11)(12)(13)(14)(15)(19)(20(21)
18						(17)		(16)				(10)(11)(12)(13)(14)(15)(19)(20)(21)
19				(18)			(17)					(16)(19)(20)(21)
20												

2-issue SS + Register renaming +OoO

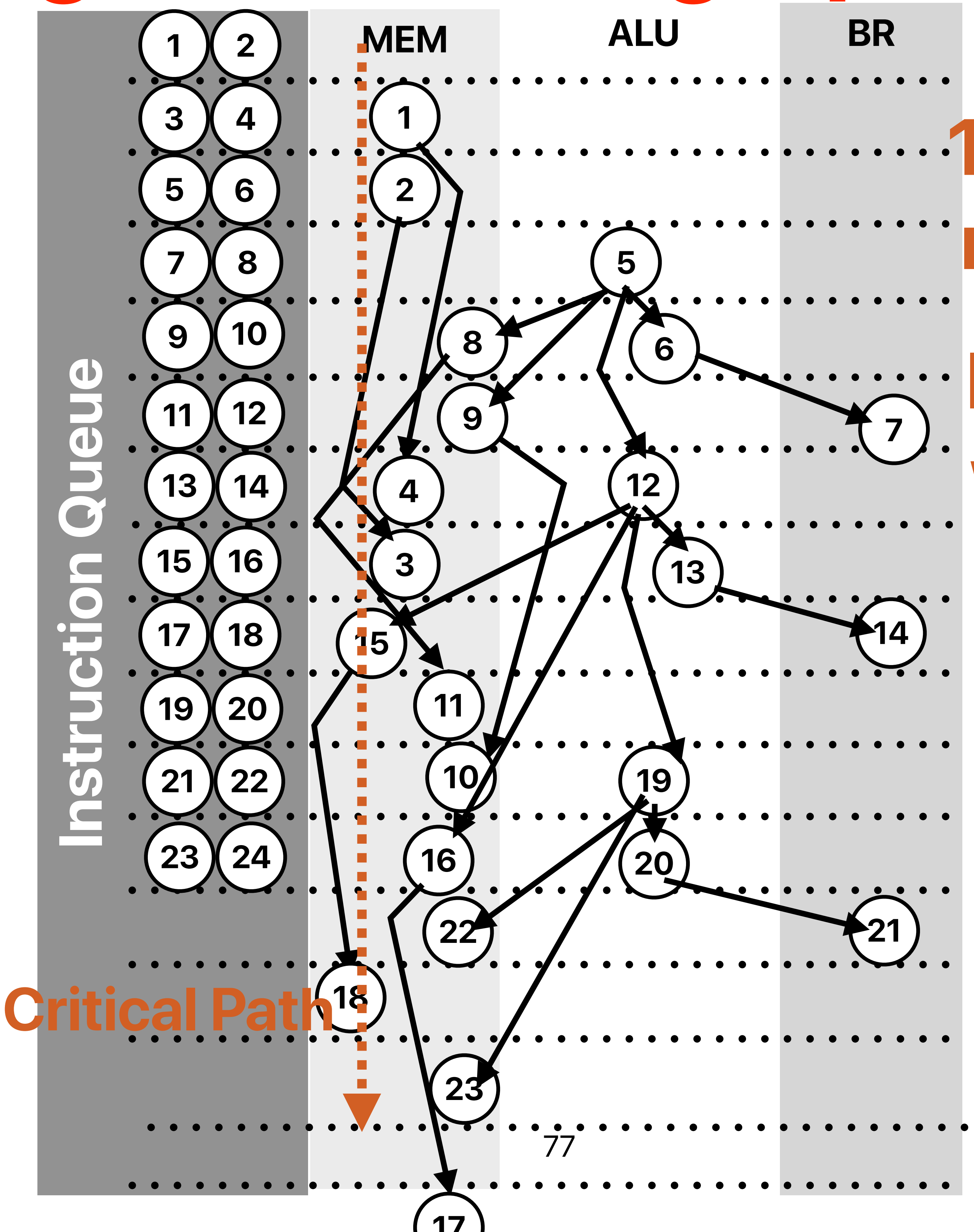
```
①  movq (%rdi,%rax), %rsi    → P1
②  movq (%rcx,%rax), %r8     → P2
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)    → P3
⑤  addq $8, %rax
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi    → P4
⑨  movq (%rcx,%rax), %r8     → P5
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax             → P6
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
```

2 issue: "2" of them can have a instruction at the same cycle

	IF	ID	REN	AG	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)(2)											
2	(3)(4)	(1)(2)										
3	(5)(6)	(3)(4)	(1)(2)									
4	(7)(8)	(5)(6)	(2)(3)(4)	(1)								
5	(9)(10)	(7)(8)	(3)(4)(5)(6)	(2)	(1)							
6	(11)(12)	(9)(10)	(3)(4)((6)(7)(8)		(2)	(1)						
7	(13)(14)	(11)(12)	(3)(4)((6)(7)(9)(10)	(8)		(2)	(1)		(5)			(5)
8	(15)(16)	(13)(14)	(3)(4)(7)(10)(11)(12)	(9)	(8)		(2)	(1)	(6)			(5)(6)
9	(17)(18)	(15)(16)	(3)(10)(11)(12)(13)(14)	(4)	(9)	(8)		(2)			(7)	~~(1)~~(5)(6)
10	(19)(20)	(17)(18)	(10)(11)(13)(14)(15)(16)	(3)	(4)	(9)	(8)		(12)			~~(2)~~(5)(6)(7)
11	(21)(22)	(19)(20)	(10)(11)(14)(16)(17)(18)	(15)	(3)	(4)	(9)	(8)	(13)			(5)(6)(7)(12)
12		(21)(22)	(16)(17)(18)(19)(20	(11)	(15)	(3)	(4)	(9)			(14)	(5)(6)(7)(8)(12)(13)
13			(16)(17)(18)(20)(21)(22)	(10)	(11)	(15)	(3)	(4)	(19)			(5)(6)(7)(8)(9)(12)(13)(14)
14				(16)	(10)	(11)	(15)	(3)	(20)			(4)(5)(6)(7)(8)(9)(12)(13)(14)(19)
15					(16)	(10)	(11)	(15)			(21)	~~(3)(4)(5)(6)(7)(8)~~(9)(12)(13)(14)(19)(20)
16				(17)		(16)	(10)	(11)				(12)(13)(14)(15)(19)(20(21)
17					(17)		(16)	(10)				(11)(12)(13)(14)(15)(19)(20(21)
18						(17)		(16)				~~(10)(11)(12)(13)(14)(15)~~(19)(20)(21)
19				(18)			(17)					~~(16)~~(19)(20)(21)
20					(18)			(17)				(19)(20)(21)

Through data flow graph analysis

```
①  movq (%rdi,%rax), %rsi
②  movq (%rcx,%rax), %r8
③  movq %r8, (%rdi,%rax)
④  movq %rsi, (%rcx,%rax)
⑤  addq $8, %rax
⑥  cmpq %r9, %rax
⑦  jne  .L9
⑧  movq (%rdi,%rax), %rsi
⑨  movq (%rcx,%rax), %r8
⑩  movq %r8, (%rdi,%rax)
⑪  movq %rsi, (%rcx,%rax)
⑫  addq $8, %rax
⑬  cmpq %r9, %rax
⑭  jne  .L9
⑮  movq (%rdi,%rax), %rsi
⑯  movq (%rcx,%rax), %r8
⑰  movq %r8, (%rdi,%rax)
⑱  movq %rsi, (%rcx,%rax)
⑲  addq $8, %rax
⑳  cmpq %r9, %rax
㉑  jne  .L9
㉒  movq (%rdi,%rax), %rsi
㉓  movq (%rcx,%rax), %r8
㉔  movq %r8, (%rdi,%rax)
㉕  movq %rsi, (%rcx,%rax)
㉖  addq $8, %rax
㉗  cmpq %r9, %rax
```



12 cycles for every 11 memory instructions

If we have 11 loops, it will have 44 memory instructions, 77 instructions in total and take 48 cycles

CPI: $\frac{48}{77} = 0.62$

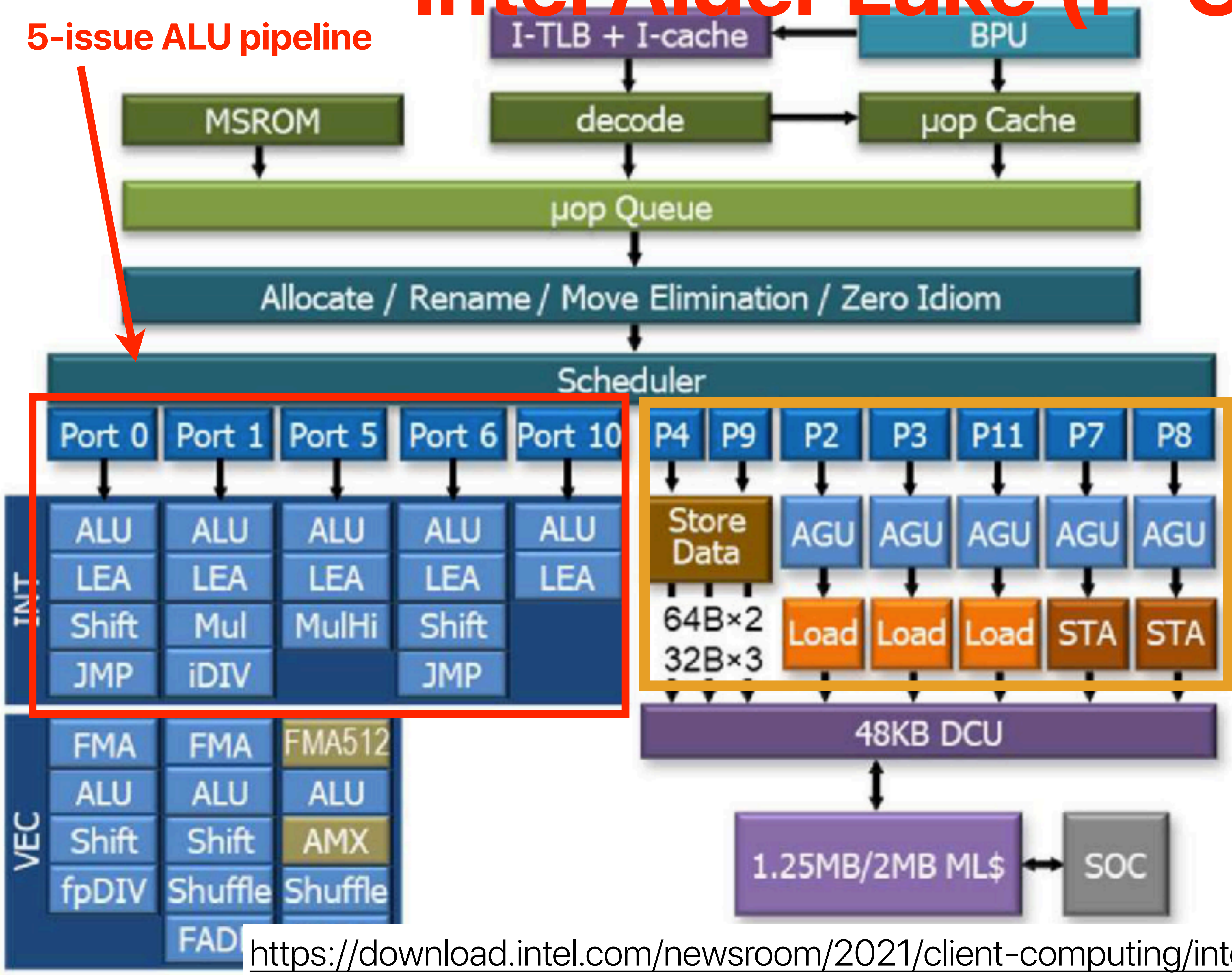
Takeaways: data hazards

- More data dependencies, more likelihood of data hazards
- Stalls and data forwarding can both address data hazards to generate correct code execution results — but not very efficient
- Compiler optimizations can help, but to a limited extent
- False dependencies limits the freedom of out-of-order execution
- Register renaming + Speculative execution enables more efficient execution by dynamically scheduling instructions whenever their data dependencies are resolved
- Super scalar further improves the utilization of hardware and throughput

The pipelines of Modern Processors

Intel Alder Lake (P-Core)

5-issue ALU pipeline



7-issue memory pipeline

$$MinCPI = \frac{1}{12}$$

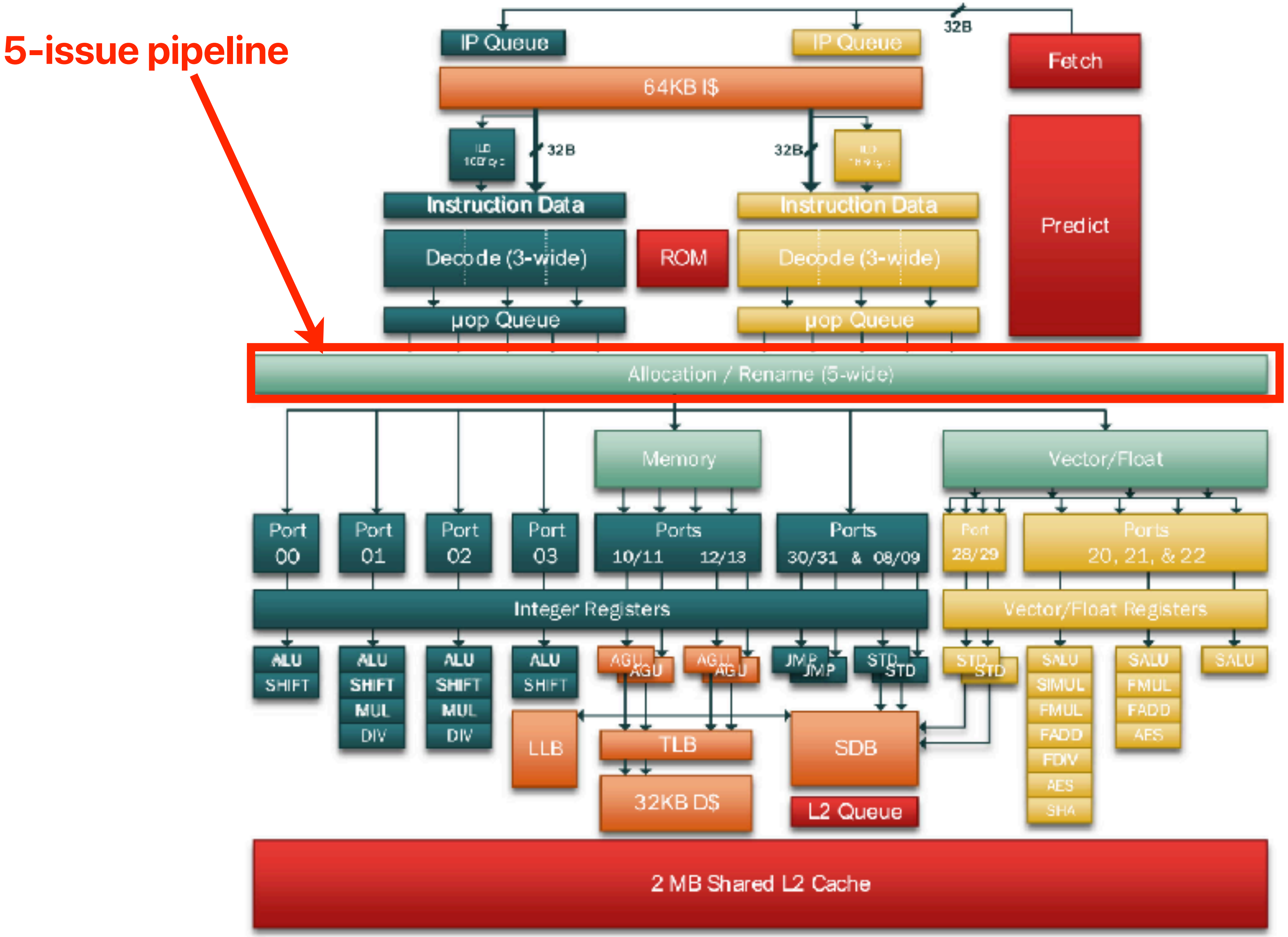
$$MinINTInst.CPI = \frac{1}{5}$$

$$MinMEMInst.CPI = \frac{1}{7}$$

$$MinBRInst.CPI = \frac{1}{2}$$

https://download.intel.com/newsroom/2021/client-computing/intel-architecture-day-2021-presentation.pdf

Intel Alder Lake (E-Core)



AMD Zen 3 (RyZen 5000 Series)

3-issue memory pipeline

4-issue integer pipeline + 1 additional branch

$$MinCPI = \frac{1}{8}$$

$$MinINTInst.CPI = \frac{1}{4}$$

$$MinMEMInst.CPI = \frac{1}{3}$$

$$MinBRInst.CPI = \frac{1}{2}$$

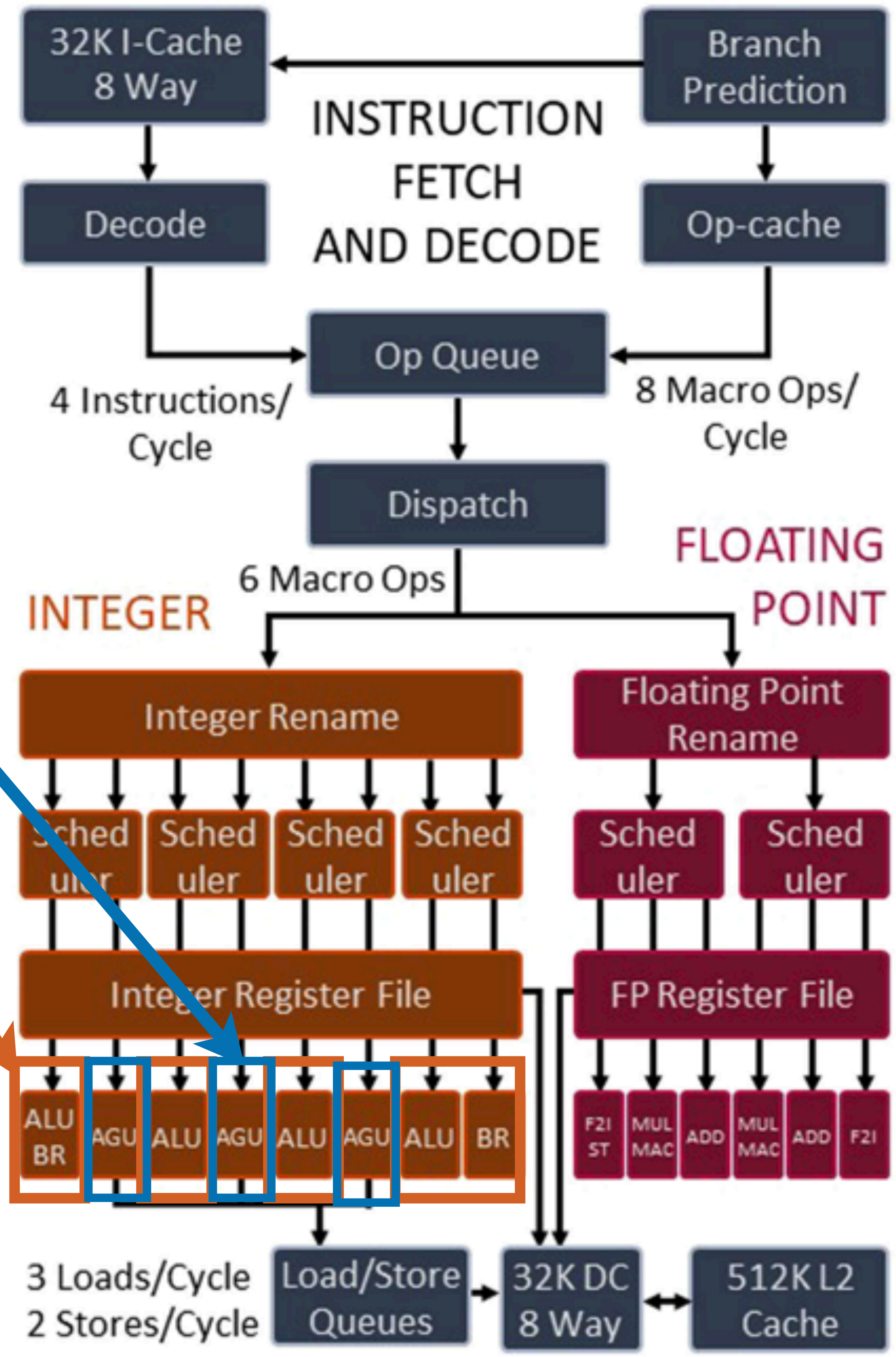


FIGURE 1. "Zen 3" block diagram.

Summary: Characteristics of modern processor architectures

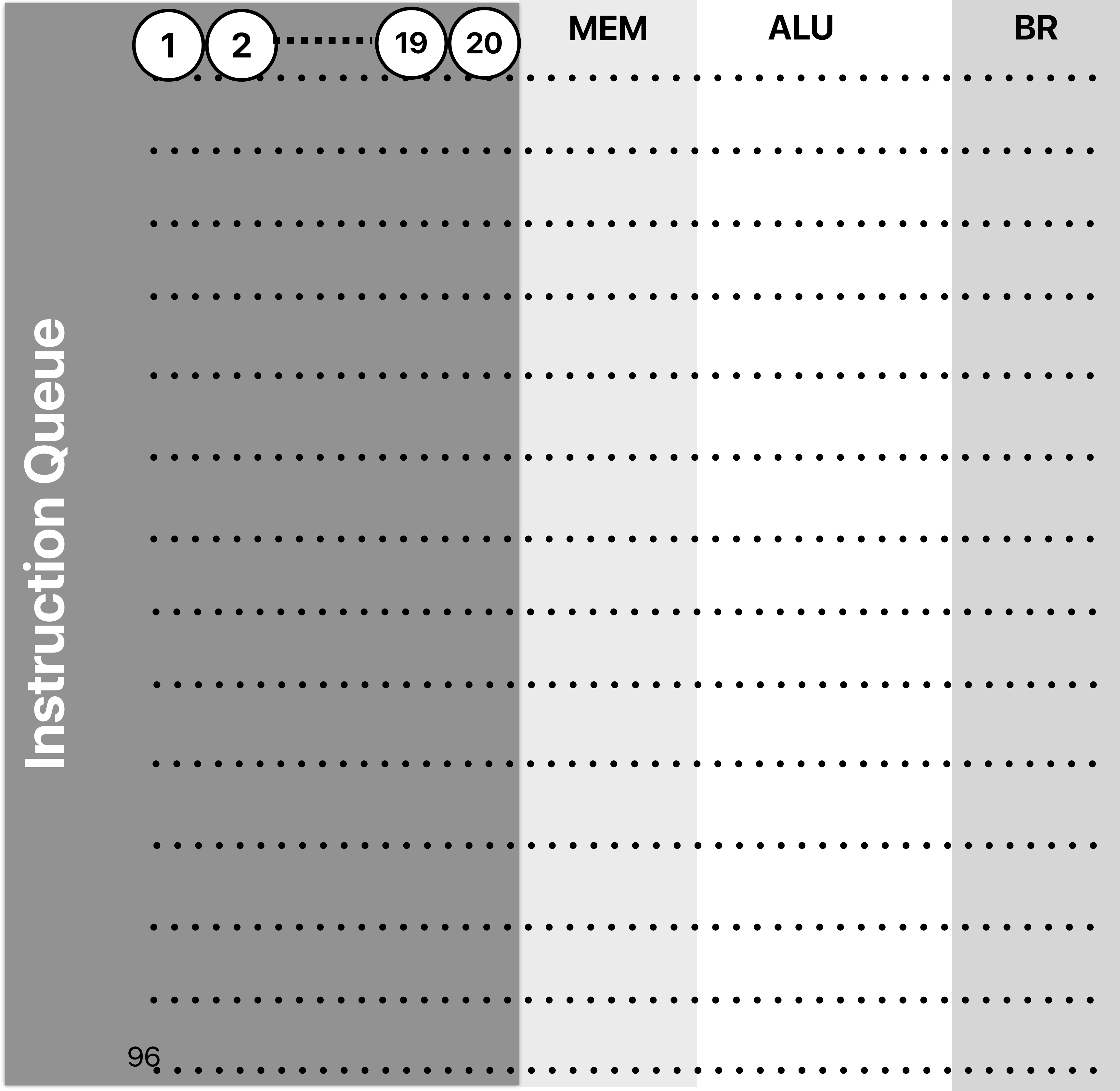
- Multiple-issue pipelines with multiple functional units available
  - Multiple ALUs
  - Multiple Load/store units
  - Dynamic OoO scheduling to reorder instructions whenever possible
- Cache — very high hit rate **if your code has good locality**
  - Very matured data/instruction prefetcher
- Branch predictors — very high accuracy **if your code is predictable**
  - Perceptron
  - TAGE

Takeaways: data hazards

- More data dependencies, more likelihood of data hazards
- Stalls and data forwarding can both address data hazards to generate correct code execution results — but not very efficient
- Compiler optimizations can help, but to a limited extent
- False dependencies limits the freedom of out-of-order execution
- Register renaming + Speculative execution enables more efficient execution by dynamically scheduling instructions whenever their data dependencies are resolved
- Super scalar further improves the utilization of hardware and throughput
- Modern processors are all very wide-issue super scalar processors with OoO capabilities

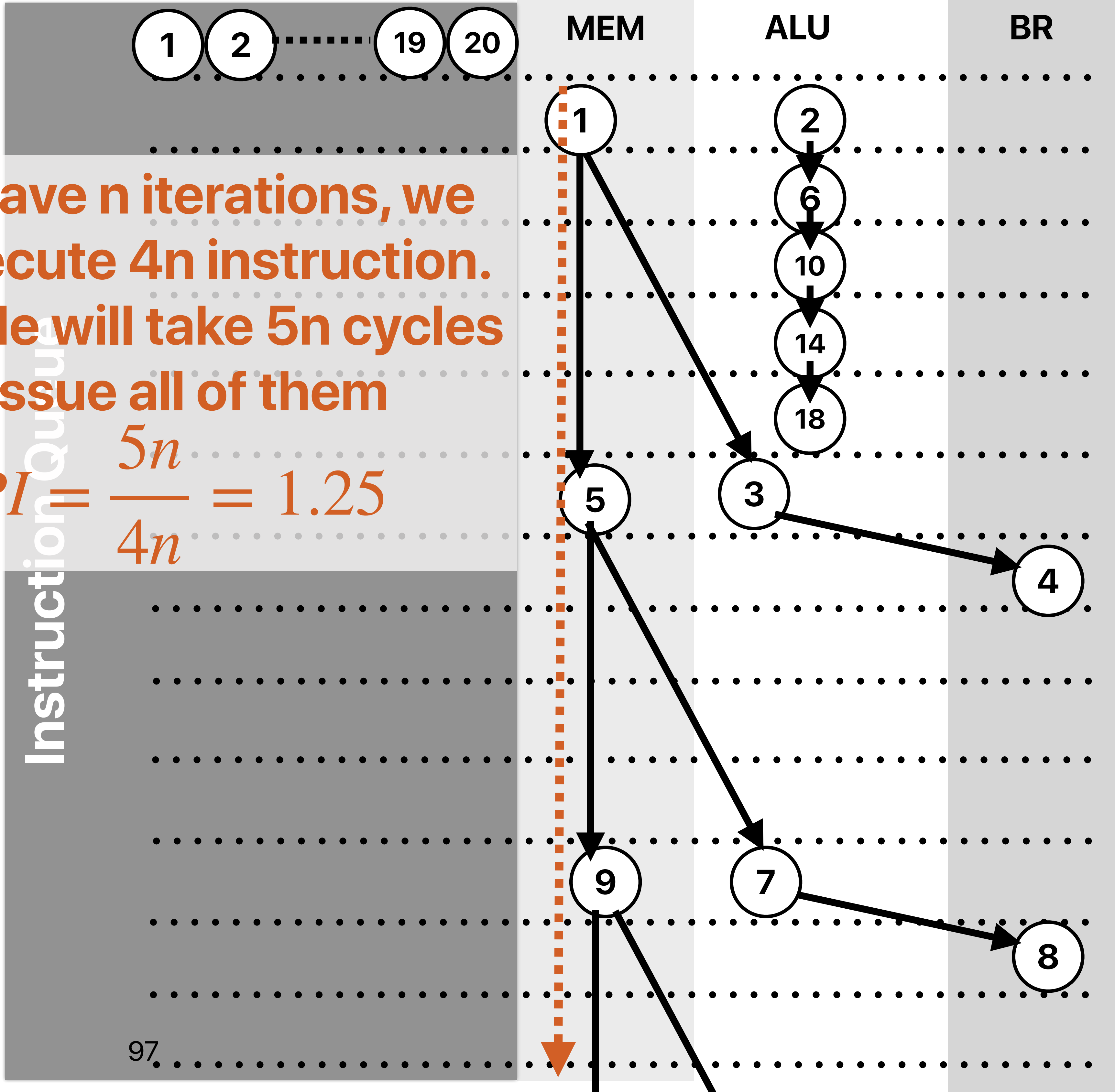
What if we have "unlimited" fetch/issue width — "linked list"

```
① .L3:   movq    8(%rdi), %rdi
②        addl    $1, %eax
③        testq   %rdi, %rdi
④        jne     .L3
⑤ .L3:   movq    8(%rdi), %rdi
⑥        addl    $1, %eax
⑦        testq   %rdi, %rdi
⑧        jne     .L3
⑨ .L3:   movq    8(%rdi), %rdi
⑩        addl    $1, %eax
⑪        testq   %rdi, %rdi
⑫        jne     .L3
⑬ .L3:   movq    8(%rdi), %rdi
⑭        addl    $1, %eax
⑮        testq   %rdi, %rdi
⑯        jne     .L3
⑰ .L3:   movq    8(%rdi), %rdi
⑱        addl    $1, %eax
⑲        testq   %rdi, %rdi
⑳        jne     .L3
```



What if we have "unlimited" fetch/issue width — "linked list"

```
① .L3:  movq   8(%rdi), %rdi
②       addl   $1, %eax
③       testq  %rdi, %rdi
④       jne    .L3
⑤ .L3:  movq   8(%rdi), %rdi
⑥       addl   $1, %eax
⑦       testq  %rdi, %rdi
⑧       jne    .L3
⑨ .L3:  movq   8(%rdi), %rdi
⑩       addl   $1, %eax
⑪       testq  %rdi, %rdi
⑫       jne    .L3
⑬ .L3:  movq   8(%rdi), %rdi
⑭       addl   $1, %eax
⑮       testq  %rdi, %rdi
⑯       jne    .L3
⑰ .L3:  movq   8(%rdi), %rdi
⑱       addl   $1, %eax
⑲       testq  %rdi, %rdi
⑳       jne    .L3
```



If we have n iterations, we will execute 4n instruction. The code will take 5n cycles to issue all of them

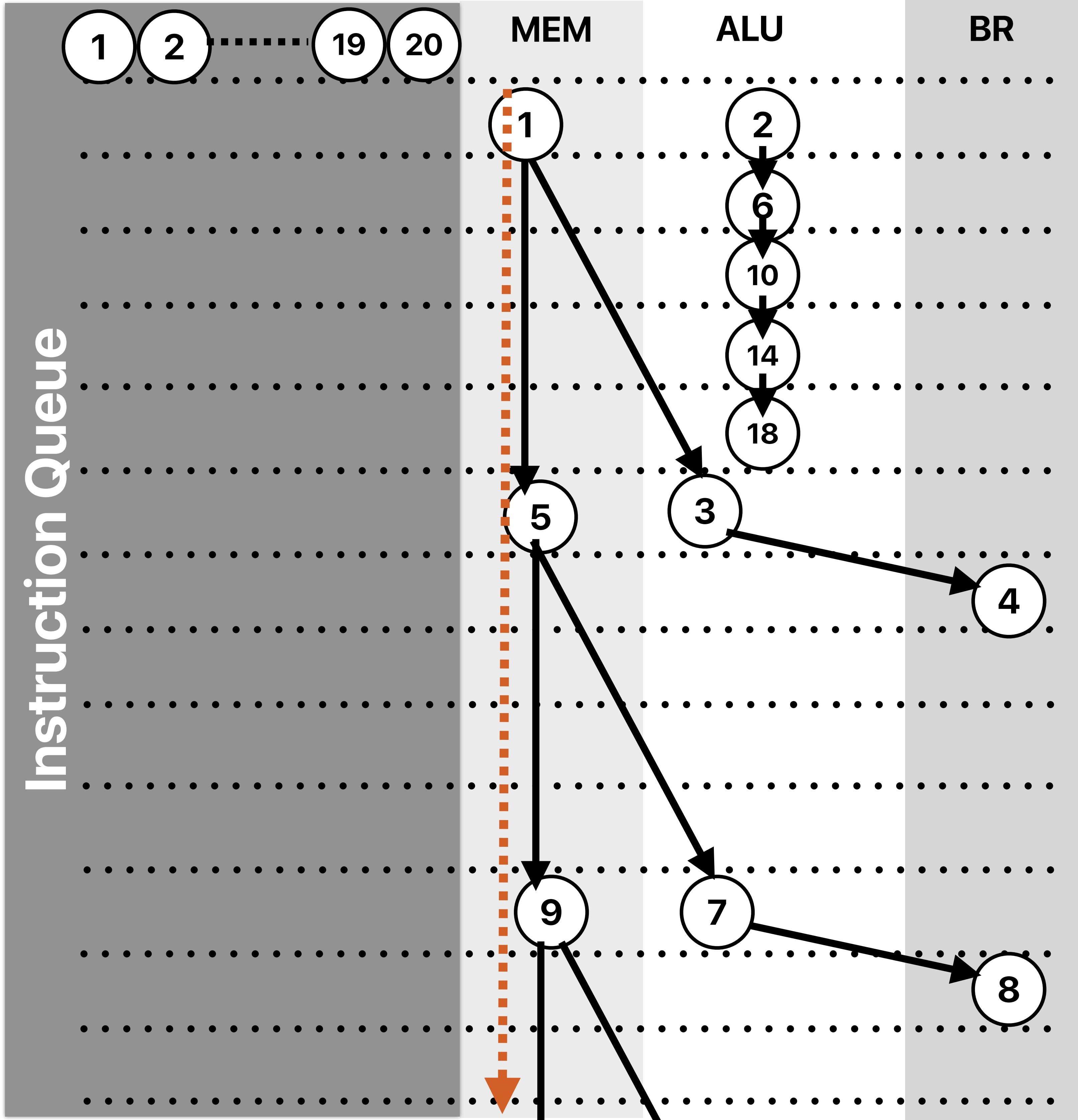
$$CPI = \frac{5n}{4n} = 1.25$$

What if we have "unlimited" fetch/issue width — "linked list"

If we cannot improve the performance of executing
`movq 8(%rdi), %rdi`
we cannot improve the execution time.
That's the "critical path"!

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

```
① .L3:    movq    8(%rdi), %rdi
②         addl    $1, %eax
③         testq   %rdi, %rdi
④         jne     .L3
```

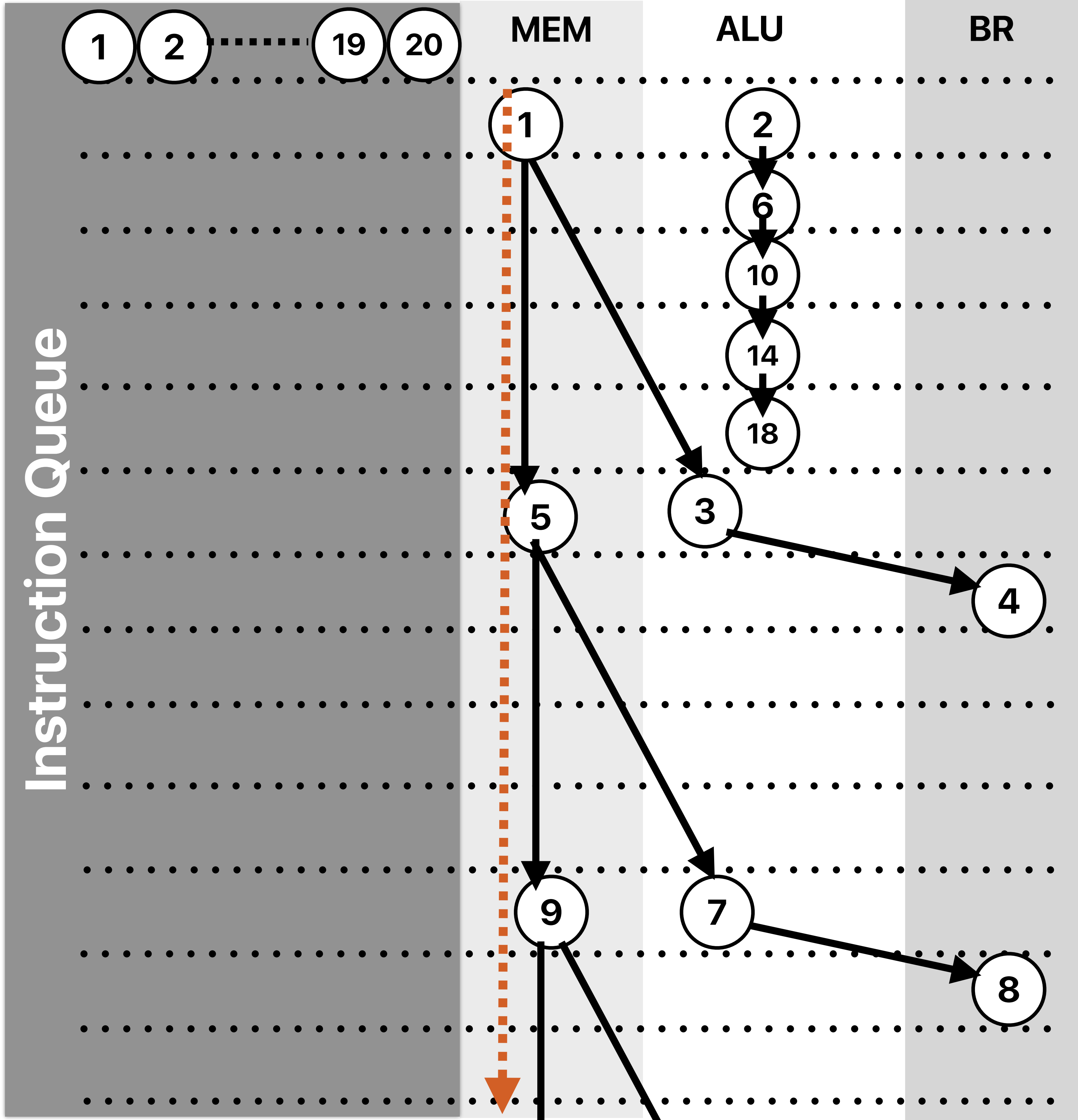


What if we have "unlimited" fetch/issue width — "linked list"

If we cannot improve the performance of executing
`movq 8(%rdi), %rdi`
we cannot improve the execution time.
That's the "critical path"!

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

```
① .L3:   movq    8(%rdi), %rdi
②        addl    $1, %eax
③        testq   %rdi, %rdi
④        jne     .L3
```



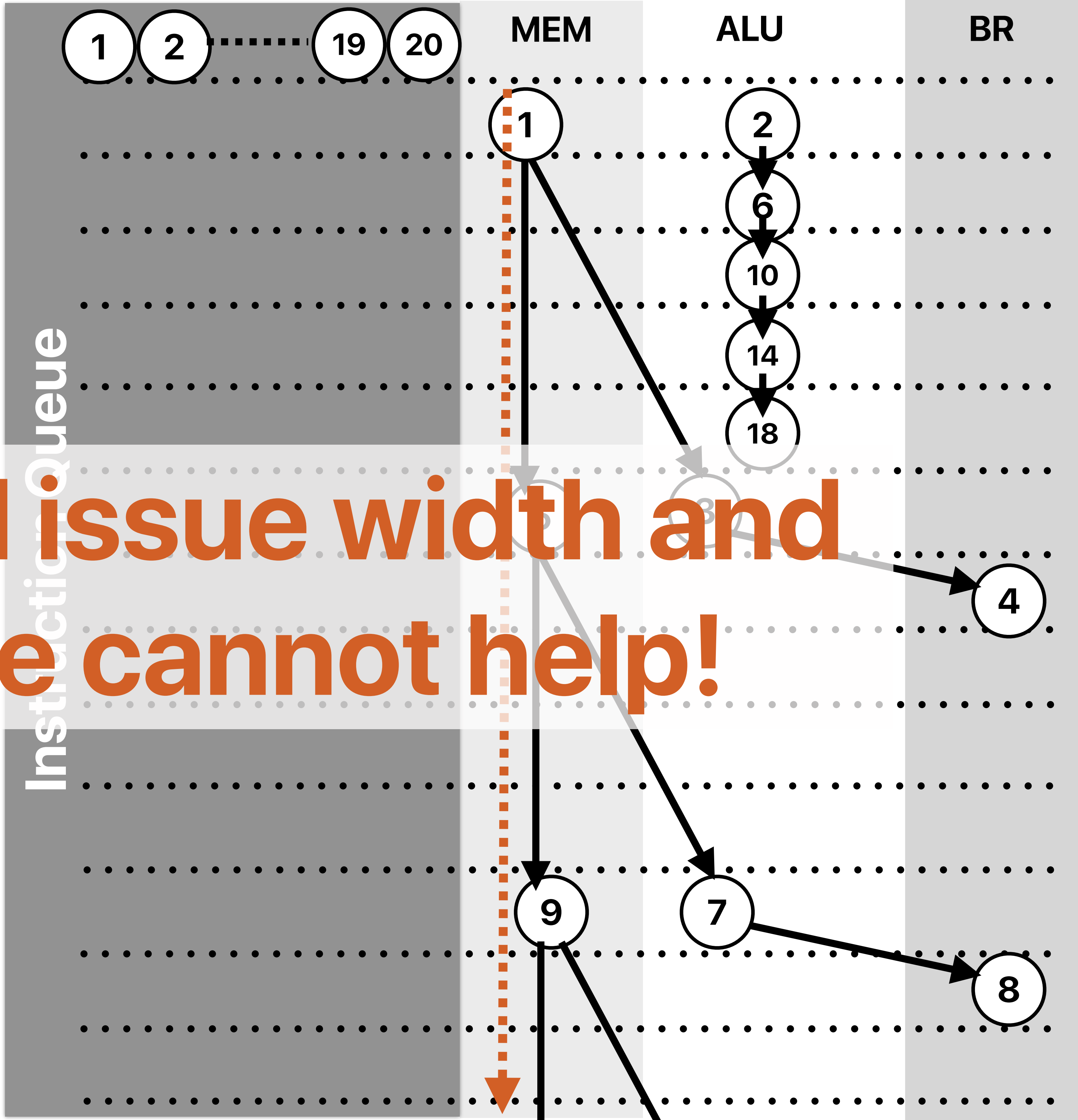
What if we have "unlimited" fetch/issue width — "linked list"

If we cannot improve the performance of executing
`movq 8(%rdi), %rdi`
we cannot improve the execution time.
That's the "critical path"!

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

Even unlimited issue width and a perfect cache cannot help!

```
① .L3:   movq    8(%rdi), %rdi
②        addl    $1, %eax
③        testq   %rdi, %rdi
④        jne     .L3
```



Takeaways: data hazards

- More data dependencies, more likelihood of data hazards
- Stalls and data forwarding can both address data hazards to generate correct code execution results — but not very efficient
- Compiler optimizations can help, but to a limited extent
- False dependencies limits the freedom of out-of-order execution
- Register renaming + Speculative execution enables more efficient execution by dynamically scheduling instructions whenever their data dependencies are resolved
- Super scalar further improves the utilization of hardware and throughput
- Modern processors are all very wide-issue super scalar processors with OoO capabilities
- If your code cannot exploit the rich ILP on modern processors, your code cannot be efficient

Computer
Science &
Engineering

つづく