Modern Processor Design (V): Try everything

Hung-Wei Tseng

Recap: 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

Recap: Solution 1: Let's try "stall" again

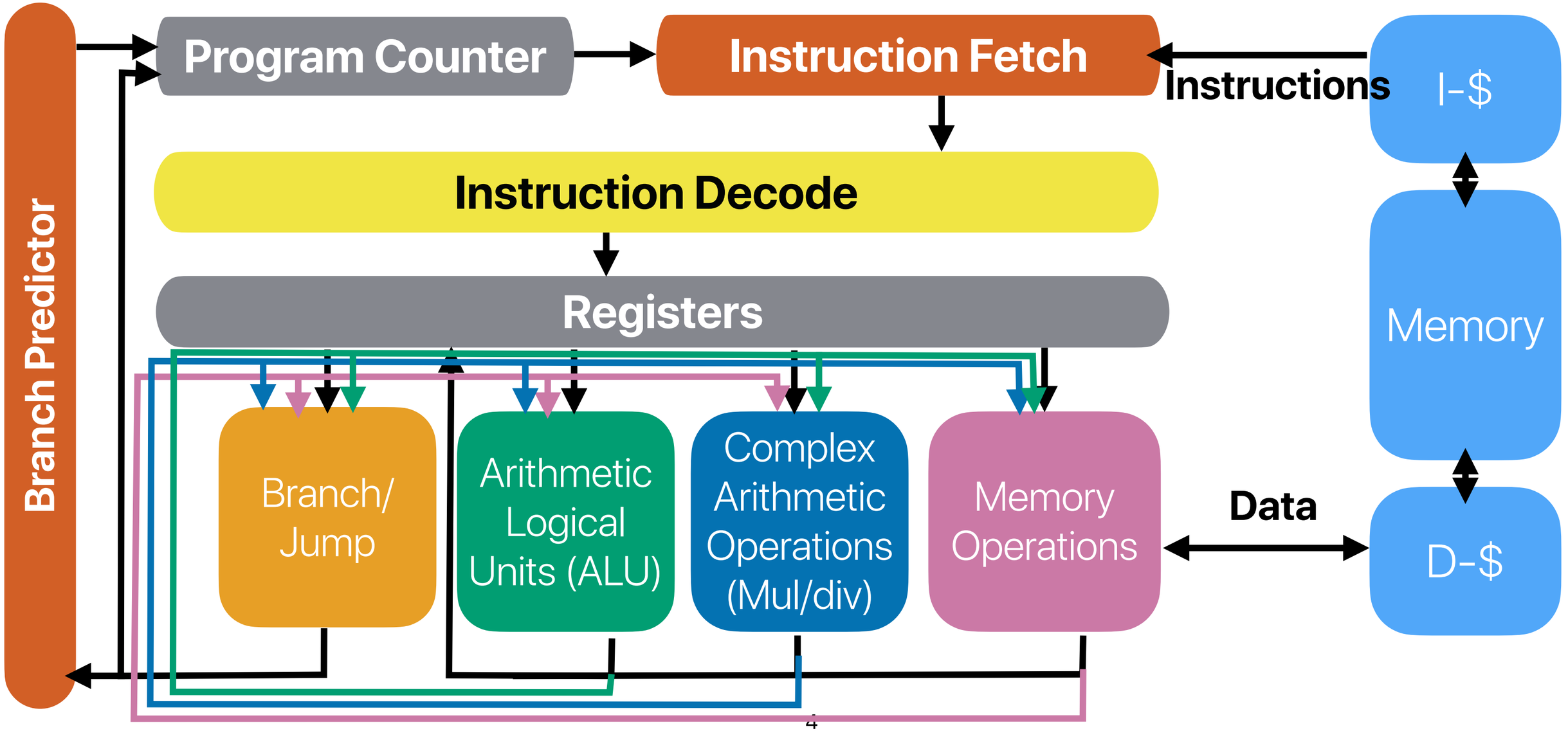
- Whenever the input is not ready when the consumer is decoding, just stall — the consumer stays at ID.

```
① movl    (%rdi), %eax
② movl    (%rsi), %edx
③ movl    %edx, (%rdi)
④ movl    %eax, (%rsi)
```

	IF	ID	ALU/BR/M1	M2	M3	M4	WB
1	(1)						
2	(2)	(1)					
3	(3)	(2)	(1)				
4	(4)	(3)	(2)	(1)			
5	(4)	(3)		(2)	(1)		
6	(4)	(3)			(2)	(1)	
7	(4)	(3)				(2)	(1)
8	(4)	(3)					(2)
9		(4)	(3)				
10			(4)	(3)			
11				(4)	(3)		
12					(4)	(3)	
13						(4)	(3)
14							(4)

we have the value for %edx already!
Why another cycle?

Recap: Data "forwarding"



Recap: Solution 2: Data forwarding

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

```
① movl    (%rdi), %eax
② movl    (%rsi), %edx
③ movl    %edx, (%rdi)
④ movl    %eax, (%rsi)
```

	IF	ID	ALU/BR/M1	M2	M3	M4	WB
1	(1)						
2	(2)	(1)					
3	(3)	(2)	(1)				
4	(4)	(3)	(2)	(1)			
5	(4)	(3)		(2)	(1)		
6	(4)	(3)			(2)	(1)	
7	(4)	(3)	data forwarding			(2)	(1)
8		(4)	(3)				(2)
9			(4)	(3)			
10				(4)	(3)		
11					(4)	(3)	
12						(4)	(3)
13							(4)
14							

Recap: Let's extend the example a bit…

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

```
.L9:
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
```

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

10 cycles for 7 instructions

CPI = 1.43

The effect of code optimization

- By reordering which pair of the following instruction stream can we reduce stalls without affecting the correctness of the code?

```
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
```

A. (1) & (2)

B. (2) & (3)

C. (3) & (5)

D. (4) & (6)

E. No ordering can help reduce the stalls

Compiler optimization

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

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

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

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

9 cycles for 7 instructions
CPI = 1.29

Outline

- Dynamic instruction scheduling and OoO (out-of-order) execution
- Superscalar: ILP < 1
- Case studies: pipeline in modern processors

Missing opportunities

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

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

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

Compiler cannot optimize across branch since it's predicted during runtime!

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

7 cycles for 7 instructions

CPI = 1

Limitations of Compiler Optimizations

- If the hardware (e.g., pipeline changes), the same compiler optimization may not be that helpful
- The compiler can only optimize on static instructions, but cannot optimize dynamic instruction
  - Compiler cannot predict branches
  - Compiler does not know if cache has the data/instructions

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

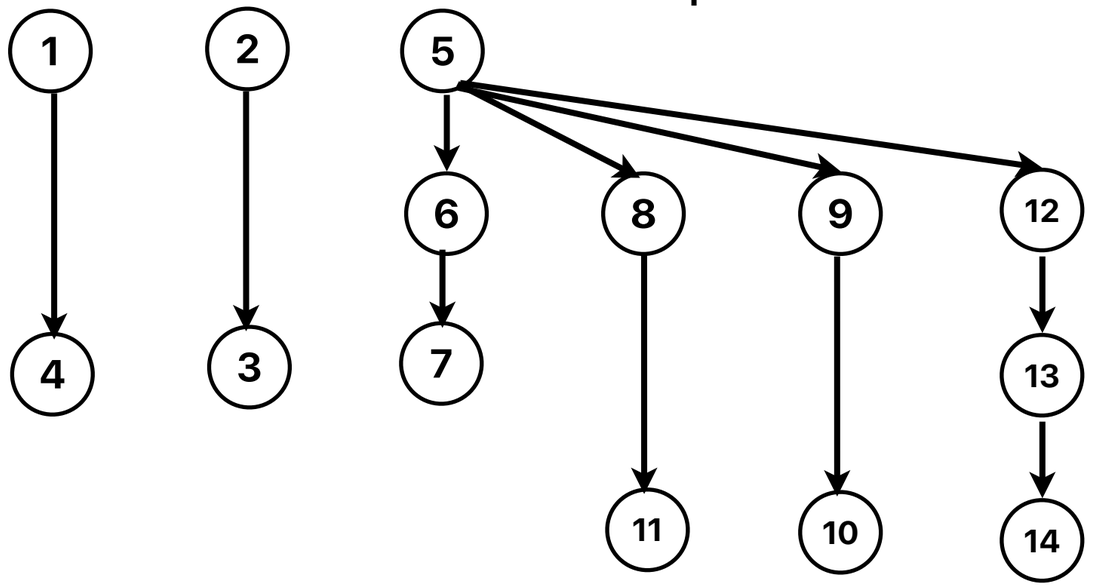
What do you need to execution an instruction?

- 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
```



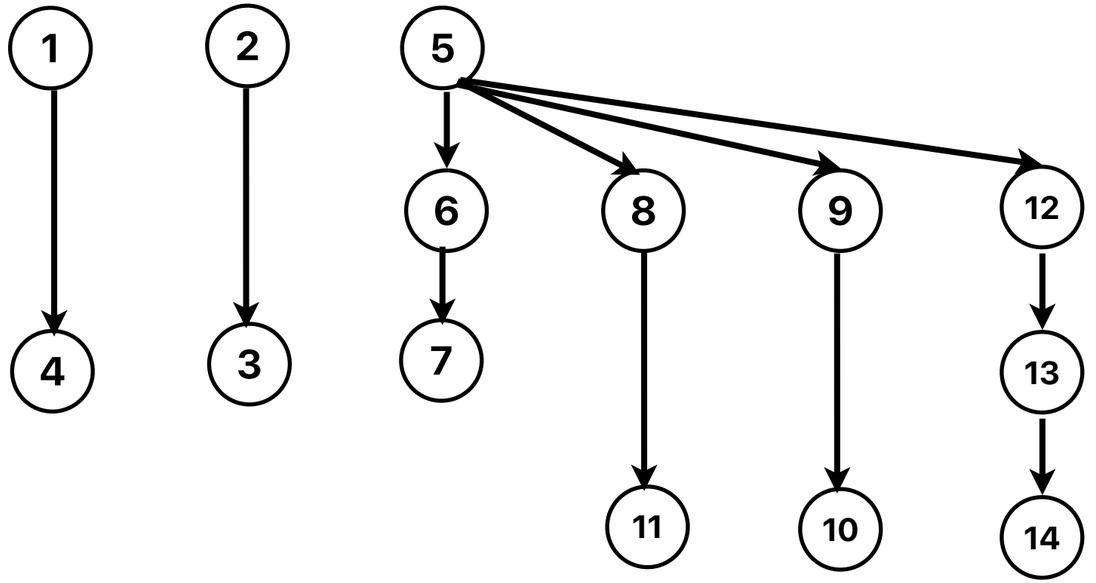
- **In theory**, instructions without dependencies can be executed in parallel or out-of-order
- Instructions with dependencies (on the same path) can never be reordered

https://www.pollev.com/hungweitseng close in 1:30

If we can predict the future …

- Consider the following dynamic instructions:

```
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
```



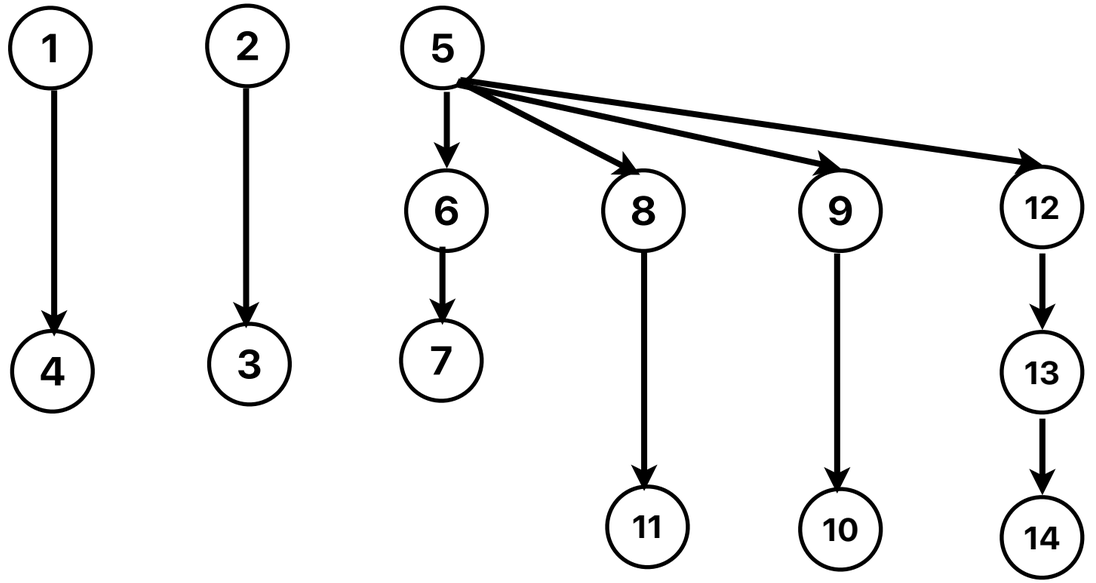
Which of the following pair can we reorder without affecting the correctness if the **branch prediction is perfect**?

A. (1) and (2)
B. (3) and (5)
C. (4) and (6)
D. (3) and (8)
E. (11) and (12)

If we can predict the future …

- Consider the following dynamic instructions:

```
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
```



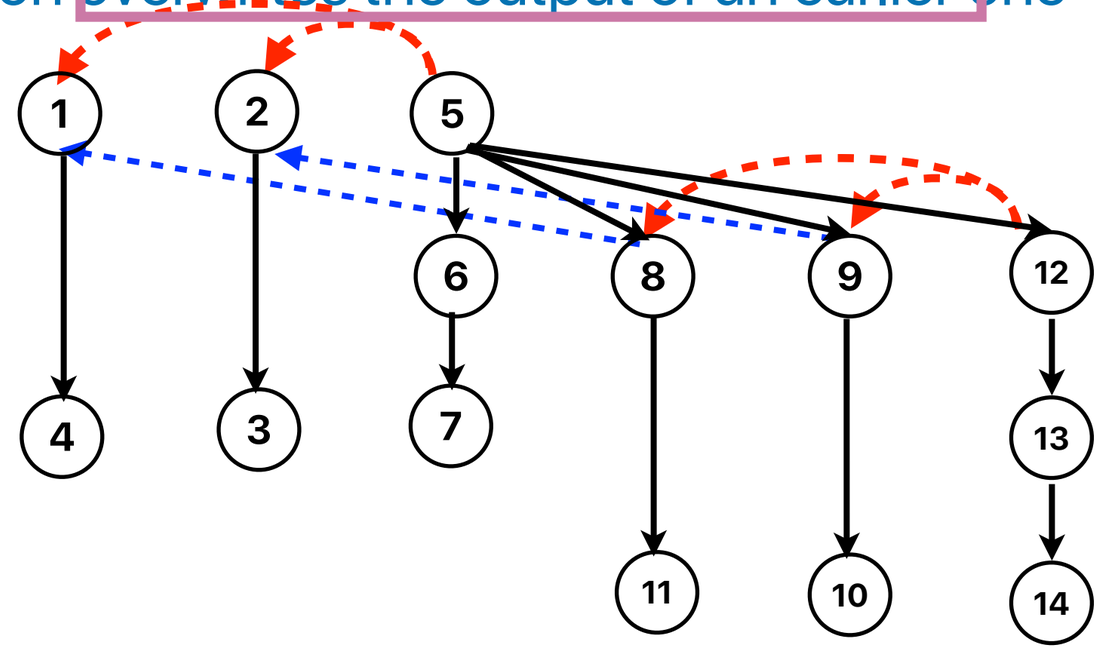
Which of the following pair can we reorder without affecting the correctness if the **branch prediction is perfect**?

A. (1) and (2)
B. (3) and (5)
C. (4) and (6)
D. (3) and (8)
E. (11) and (12)

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
```

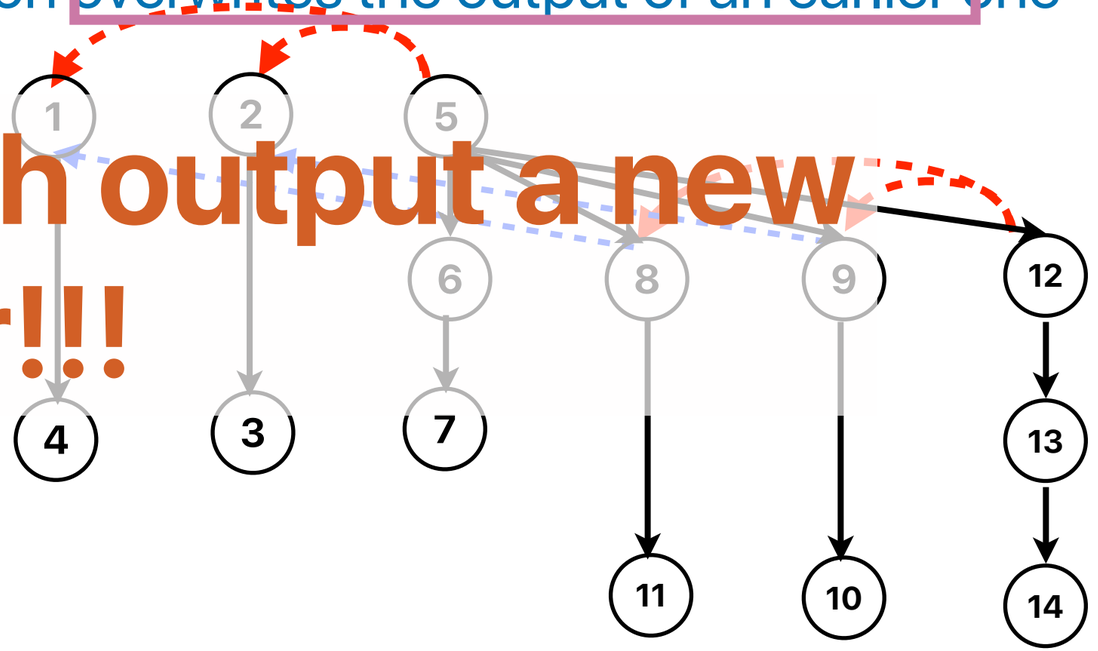


False dependencies

- 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

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

```
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
```



What if we can use more registers...

```
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
```

```
①  movq    (%rdi,%rax), %t0
②  movq    (%rcx,%rax), %t1
③  movq    %t1, (%rdi,%rax)
④  movq    %t0, (%rcx,%rax)
⑤  addq    $8, %t2
⑥  cmpq    %r9, %t2
⑦  jne     .L9
⑧  movq    (%rdi, %t2), %t3
⑨  movq    (%rcx, %t2), %t4
⑩  movq    %t4, (%rdi,%t2)
⑪  movq    %t3, (%rcx,%t2)
⑫  addq    $8, %t5
⑬  cmpq    %r9, %t5
⑭  jne     .L9
```

All false dependencies are gone!!!

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

https://www.pollev.com/hungweitseng close in 1:30

Abort an instruction?

- During how many of the following states can we still abort an instruction and change the execution flow even without a branch instruction or branch misprediction?
  - ① Fetching instruction from the memory
  - ② Decoding an instruction
  - ③ Executing arithmetic operations in an ALU
  - ④ Accessing memory
  - ⑤ Write back register values
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

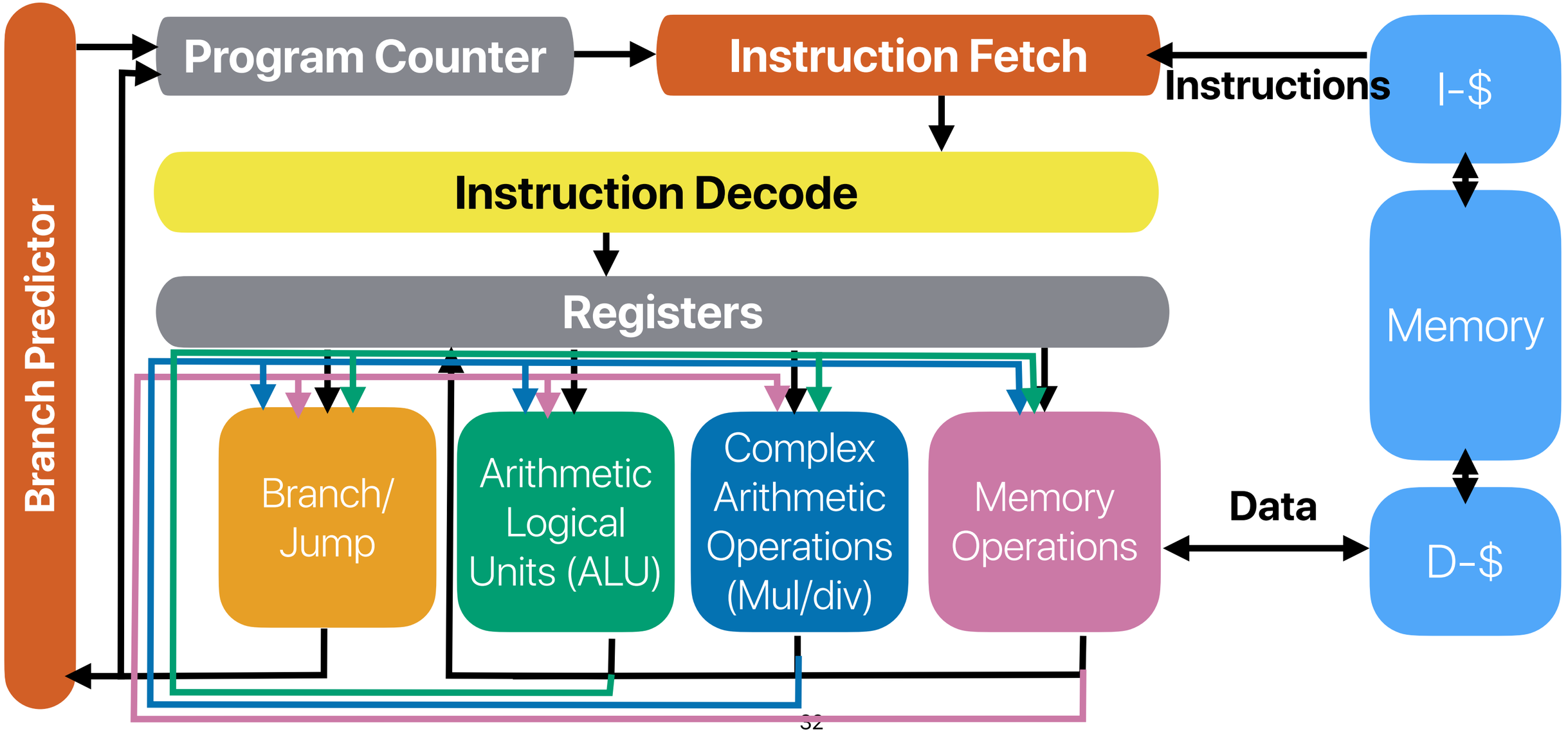
Abort an instruction?

- During how many of the following states can we still abort an instruction and change the execution flow even without a branch instruction or branch misprediction?
  - ① Fetching instruction from the memory — **page fault, illegal address**
  - ② Decoding an instruction — **unknown instruction**
  - ③ Executing arithmetic operations in an ALU — **divide by zero, overflow, underflow**
  - ④ Accessing memory — **page fault, illegal address**
  - ⑤ Write back register values
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

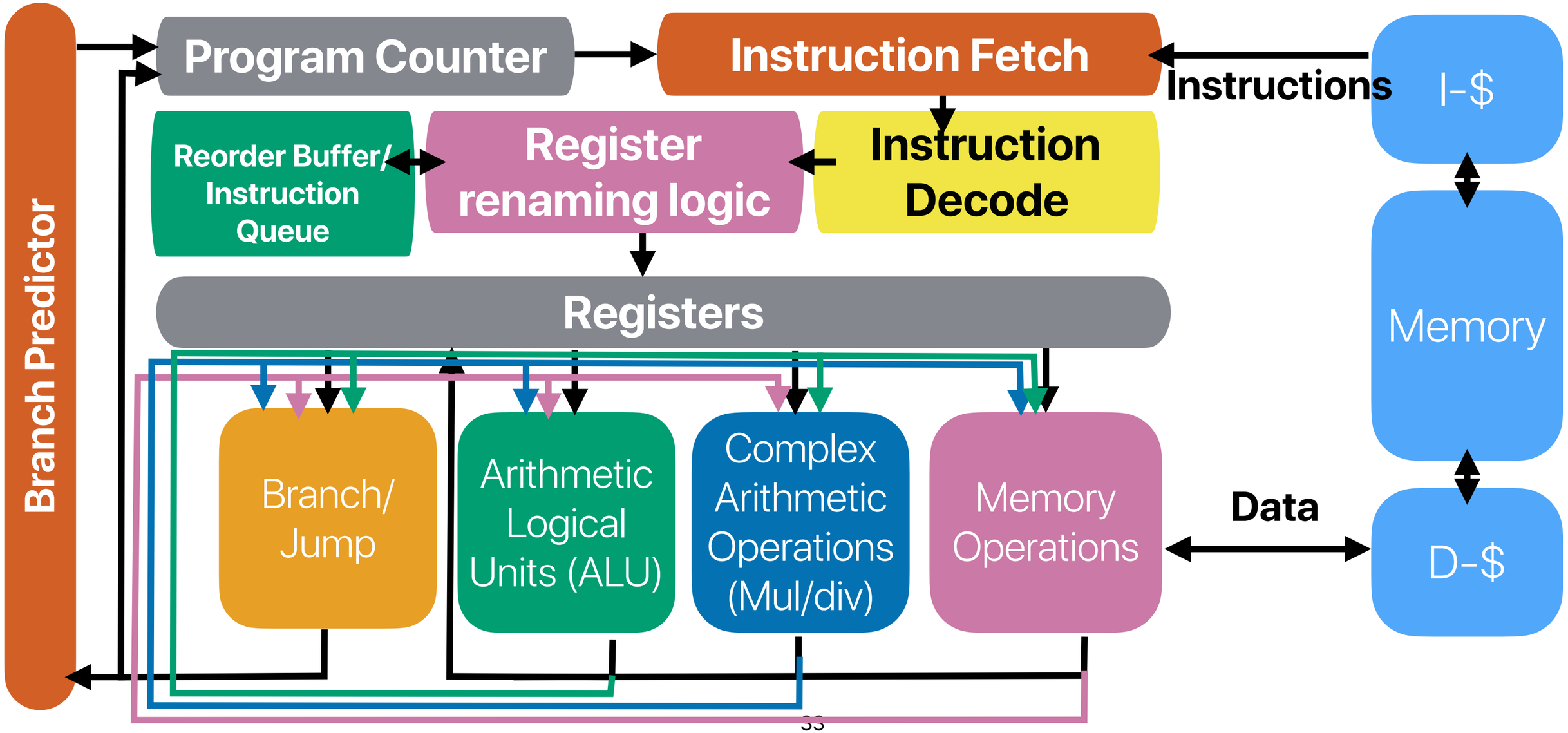
Speculative Execution

- **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	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	P6
rcx	
rdi	
rsi	P4
r8	P5

	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 → P1
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	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	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	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            → 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	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	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)(5)(6)	(4)			(2)				(1)
9											
10											
11											
12											
13											
14											
15											
16											

Instruction (4) is running ahead of (3)

	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	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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						(2)
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	1		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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
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	1		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 → 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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(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	0		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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(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
⑬  cmpq %r9, %rax
⑭  jne  .L9
```

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

	IF	ID	REN	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14				(9)	(8)						
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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15		(14)	(10)(11)(12)(13)		(9)	(8)					
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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15		(14)	(10)(11)(12)(13)		(9)	(8)					
16						(9)	(8)				

	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
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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15	(15)	(14)	(10)(11)(13)		(9)	(8)		(12)			
16	(16)	(15)	(10)(11)(14)			(9)	(8)	(13)			(12)
17	(17)	(16)	(10)(14)(15)	(11)			(9)				~~(8)~~(12)(13)

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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15	(15)	(14)	(10)(11)(13)		(9)	(8)		(12)			
16	(16)	(15)	(10)(11)(14)			(9)	(8)	(13)			(12)
17	(17)	(16)	(10)(14)(15)	(11)			(9)				~~(8)~~(12)(13)
18	(18)	(17)	(14)(15)(16)	(10)	(11)						~~(9)~~(12)(13)

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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15	(15)	(14)	(10)(11)(13)		(9)	(8)		(12)			
16	(16)	(15)	(10)(11)(14)			(9)	(8)	(13)			(12)
17	(17)	(16)	(10)(14)(15)	(11)			(9)				~~(8)~~(12)(13)
18	(18)	(17)	(14)(15)(16)	(10)	(11)						~~(9)~~(12)(13)
19	(19)	(18)	(15)(16)(17)		(10)	(11)				(14)	(12)(13)

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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15	(15)	(14)	(10)(11)(13)		(9)	(8)		(12)			
16	(16)	(15)	(10)(11)(14)			(9)	(8)	(13)			(12)
17	(17)	(16)	(10)(14)(15)	(11)			(9)				~~(8)~~(12)(13)
18	(18)	(17)	(14)(15)(16)	(10)	(11)						~~(9)~~(12)(13)
19	(19)	(18)	(15)(16)(17)		(10)	(11)				(14)	(12)(13)
20	(20)	(19)	(16)(17)(18)	(15)		(10)	(11)				(12)(13)(14)

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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15	(15)	(14)	(10)(11)(13)		(9)	(8)		(12)			
16	(16)	(15)	(10)(11)(14)			(9)	(8)	(13)			(12)
17	(17)	(16)	(10)(14)(15)	(11)			(9)				~~(8)~~(12)(13)
18	(18)	(17)	(14)(15)(16)	(10)	(11)						~~(9)~~(12)(13)
19	(19)	(18)	(15)(16)(17)		(10)	(11)				(14)	(12)(13)
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)

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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15	(15)	(14)	(10)(11)(13)		(9)	(8)		(12)			
16	(16)	(15)	(10)(11)(14)			(9)	(8)	(13)			(12)
17	(17)	(16)	(10)(14)(15)	(11)			(9)				~~(8)~~(12)(13)
18	(18)	(17)	(14)(15)(16)	(10)	(11)						~~(9)~~(12)(13)
19	(19)	(18)	(15)(16)(17)		(10)	(11)				(14)	(12)(13)
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)		(19)			(10)(11)(12)(13)(14)

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	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)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)	(9)	(6)(7)(8)		(3)	(4)		(5)			
11	(11)	(10)	(7)(8)(9)			(3)	(4)	(6)			(5)
12	(12)	(11)	(9)(10)				(3)			(7)	(4)(5)(6)
13	(13)	(12)	(10)(11)	(8)							~~(3)(4)(5)(6)(7)~~
14	(14)	(13)	(10)(11)(12)	(9)	(8)						
15	(15)	(14)	(10)(11)(13)		(9)	(8)		(12)			
16	(16)	(15)	(10)(11)(14)			(9)	(8)	(13)			(12)
17	(17)	(16)	(10)(14)(15)	(11)			(9)				~~(8)~~(12)(13)
18	(18)	(17)	(14)(15)(16)	(10)	(11)						~~(9)~~(12)(13)
19	(19)	(18)	(15)(16)(17)		(10)	(11)				(14)	(12)(13)
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)		(19)			~~(10)(11)(12)(13)(14)~~
23			(17)(18)(21)			(16)	(15)	(20)			(19)

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

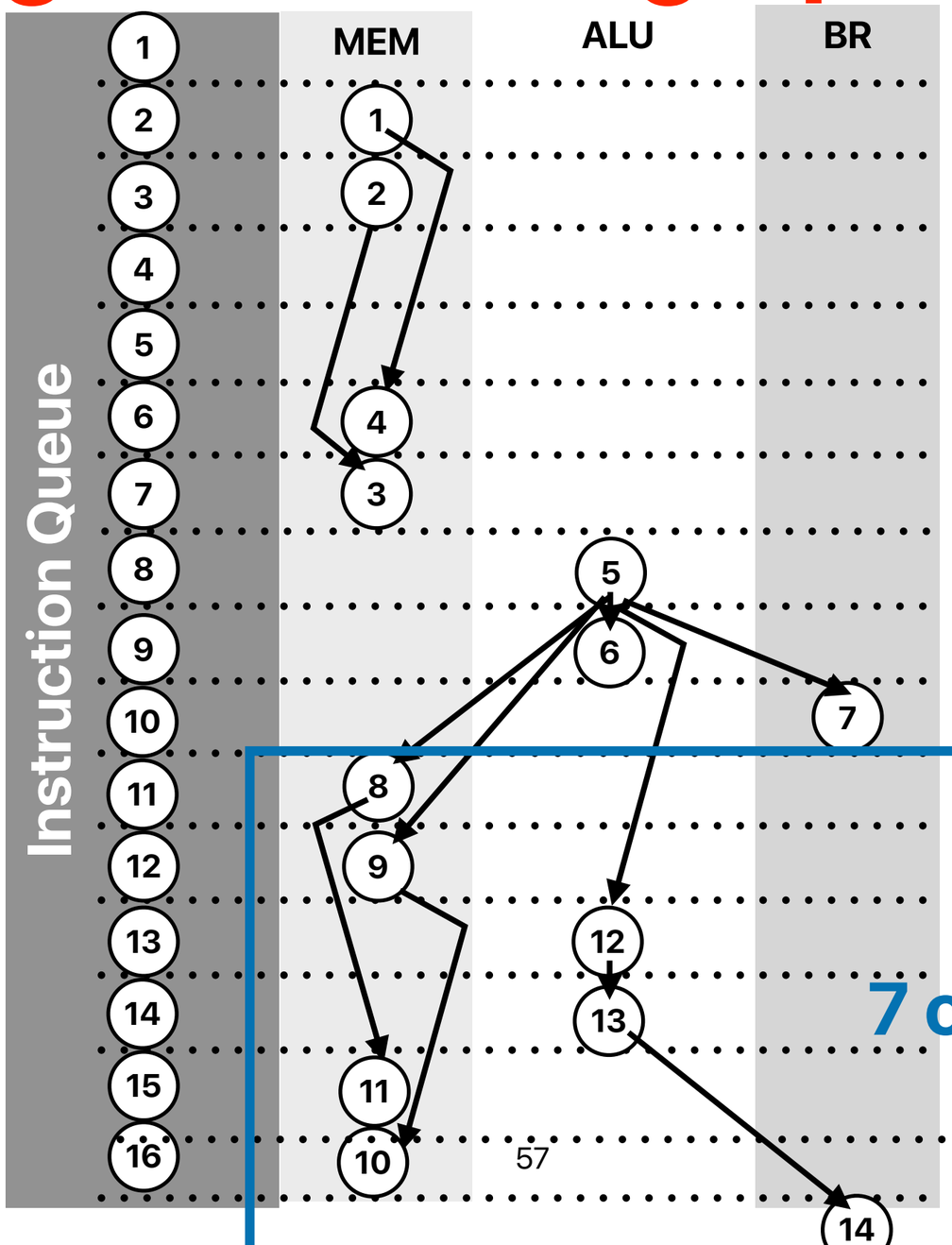
	IF	ID	REN	M1	M2	M3	M4	ALU	MUL	BR	ROB
1	(1)										
2	(2)	(1)									
3	(3)	(2)	(1)								
4	(4)	(3)	(2)	(1)							
5	(5)	(4)	(3)	(2)	(1)						
6	(6)	(5)	(3)(4)		(2)	(1)					
7	(7)	(6)	(3)(4)(5)			(2)	(1)				
8	(8)	(7)	(3)(5)(6)	(4)			(2)				~~(1)~~
9	(9)	(8)	(5)(6)(7)	(3)	(4)						~~(2)~~
10	(10)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

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

Announcements

- **Assignment 4** due tonight
  - A "Completeness" question means we don't grade on the correctness of answer itself.
  - It does not mean you can put random answer and you must complete all completeness question to have credits
  - You should check the generated "assignment.turnin.ipynb" before submission
- **Reading Quiz 7** due **next Tuesday** before the lecture
- **iEVAL** starting from the next Monday
  - Submit the prove of your participation in iEVAL through Gradescope
  - It can become a full credit reading quiz (it helps to amortize the penalty of another least performing one)
- **Assignment 5** is **released** due 6/06/2024
  - The same programming assignment as Assignment 3 but you need to speedup by 4x on Gradescope this time
- Final exam
  - 6/14 8a-11a @ **BOYHL 1471**
  - Closed book, no cheatsheet — the same rules as the midterm
  - Two questions can be used as CSMS comprehensive examine questions — one is memory-hierarchy related, and the other is OoO scheduling and code optimization (yes — today's lecture)

Computer Science & Engineering

つづく