COM SCI M151B Week 5

Aidan Jan

November 1, 2024

Out of Order Execution

Tradeoffs of Superscalar

Advantages:

• Higher IPC (instructions per cycle)

Disadvantages:

- Higher complexity for dependency checking
- More hardware resources needed

Pipeline is typically never full due to frequent dependency stalls!

Summary

- Scalar processors are limited (i.e., best case: IPC=1)
- ullet ILP and superscalar could provide this opportunity to parallelize data processing and achieve IPC > 1.
- In-order fetching prevents us from achieving the full potential of superscalar since usually there are not enough independent instructions in the pipeline.
- A solution is out-of-order execution!

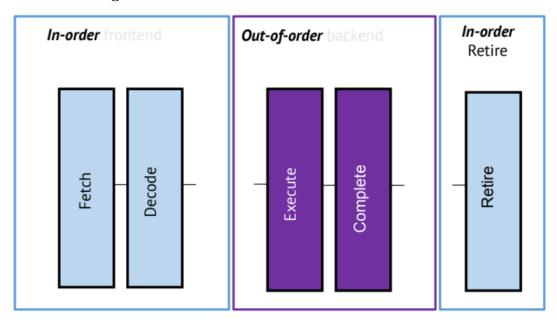
Status Quo

- Superscalar (N = 2 or 4) with pipelining and forwarding and branch prediction!
- Ideally, we can get IPC=N with small CT
- Minimal overhead due to RAW data hazard (due to load-use)
- Minimal stall due to >90% accuracy in branch prediction with only one cycle in miss penalty!
- Bottleneck
 - Not all the pipeline lanes can be always utilized due to dependencies and hazards!
 - Compilers can help but only in a limited form!
- Instead of always executing the next instruction, why don't we find the first available instruction and feed that into the pipeline?
 - If this instruction window is large enough, then we can always fully utilize our pipeline \rightarrow IPC = N.

How to execute instructions out-of-order

- A mechanism for *tracking* instructions
 - Later instructions might be **ready**.
- A mechanism for removing data hazards
 - New hazards: WAW, WAR, load-store
- A mechanism for recovery
 - Speculation (e.g., branch misprediction), etc.

Out of Order Design



Out or Order - Pipeline Recovery

- We need a temporary phase for some instructions so that if they need to be flushed, they can be easily erased!
 - We define a new stage called "retire" (aka. "commit") and separate completion from retire.

"Retire"/"Commit"

What does it mean for an instruction to finish?

- Writing back data to the register file, or
- Writing data to the memory, or
- Updating the PC (branch)

If the data is already writtenm it is too late to flush, thus we should write somewhere temporarily until we are certain!

Complete vs. Retire

- Idea: to enable recovery, allow instructions to complete (i.e., prepare the final result) out of order, but retire them (i.e., actually writing data permanently) in order.
- We call writing data permanently, writing to architectural state (i.e., registers/memory)

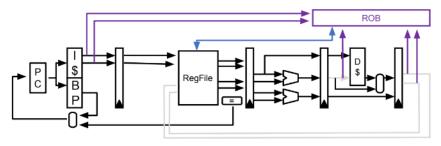
How to Implement retire?

- Need a buffer/storage to store temporary data!
- Need a mechanism to follow instructions in order and out of order to find out when to retire!

How to "retire" in-order?

- Re-order buffer (ROB)
 - ROB is a (circular) table/buffer that holds **completed** instructions.
 - ROB requires an instruction I_x only if all I_j (for j < x) are retired.

Datapath with ROB:



0.1 What if we start executing out of order?

Examples:

• RAW (Read After Write) = "true dependence"

• WAW (Write After Write) = "output dependence"

• WAR (Write After Read) = "anti-dependence"

The WAW and WAR are called "False Dependencies". These can be fixed by simply changing the register, e.g., changing the x2 in the sub instructions to x5 instead.

The RAW case is the true dependence, and in that case, the solution is much more complex. We will discuss this later!

Register Renaming

- The problem is that we have limited architectural registers (ISA registers, e.g., 32 in RISC-V)
- However, we can have much more **physical** registers (e.g., 128 registers).
- One architectural register (A-reg) can be assigned to multiple physical registers (P-reg).

How to do renaming?

- Register Alias/Map Table (RAT)
 - One entry per architectural register.
 - Each entry stores the *physical location of the most recent* version of the architectural (logical) register.
 - Algorithm:
 - * For each <u>destination</u> A-reg, the renaming algorithm assigns a new P-reg from a *pool of free* (physical) registers.
 - \ast For each <u>source</u> A-reg, the renaming algorithm accesses RAT and finds the corresponding P-reg.

Renaming Example

- Let's assume that
 - We have 5 architectural registers and 10 physical registers.
 - The *initial* mapping looks like this:

RAT								
A-reg	PMap							
x1	p1							
x2	p2							
х3	р3							
х4	p4							
x5	p5							

"Free" Pool
p6
p7
р8
р9
p10

• Consider the instructions:

• Following the RAT, we get:

• We assign x3 to point to p6 instead of p3

	RAT	
A-reg	PMa	р
x1	р1	
x2	p2	
x3	p3 p	6
x4	p4	
х5	р5	

"Free" Pool

p6

p7

p8

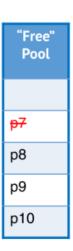
p9

p10

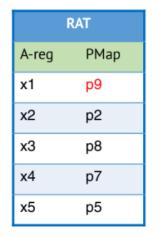
• Now, we fill the second line:

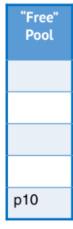
• Now, in this step, we assign x4 to p7.

	RAT
A-reg	PMap
x1	р1
x2	p2
х3	р6
x4	p4 p7
x5	р5



• Et cetera, we do reassignments for the destination registers.

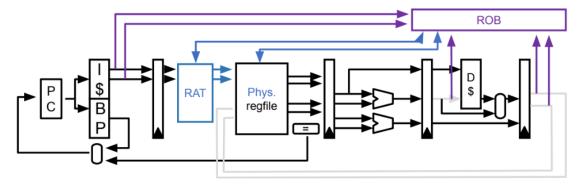




When to free?

We free the previous, "old" destination register when the instruction is retired. e.g., once the first instruction is executed (and retired), we can free p3 (not p6!), because that means that the value of p3 is no longer used since it got reassigned.

Datapath with Renaming and ROB



Recap

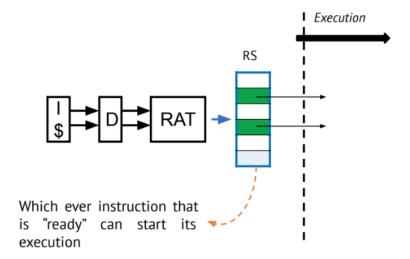
At this point, we have the mechanism for removing data hazards and the mechanism for recovery.

Now, we just need the mechanism for tracking instructions, as described earlier.

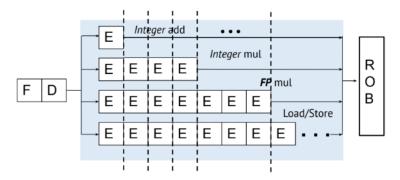
How to Schedule

- We need a mechanism to track who is ready:
 - I3 is not ready, but I4 is!
 - Who is ready?
 - * This also helps to fix RAW hazards (i.e., forwarding!)

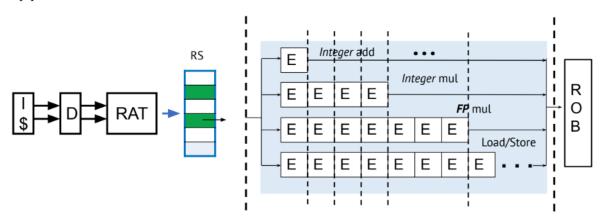
Reservation Station



Now, we can remove the MEM stage and consolidate it to one functional unit, with multiple units (FUs)



Our pipeline looks like this:



Steps

- 1. Fetch
- $2.\,$ Dispatch: putting instructions in reservation station.
 - Copying values (e.g., rsl) to RS.
 - Reserving an entry in ROB

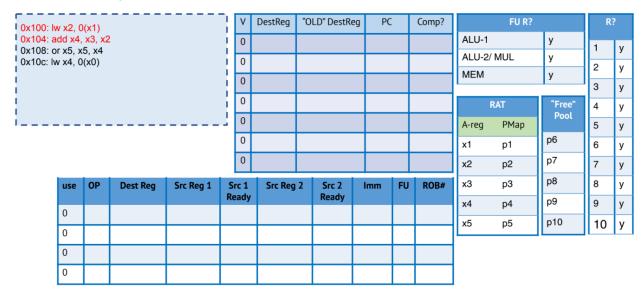
- If RS or ROB is full, stall the backend.
- 3. Issue/Fire: starting the execution once all sources are ready and the FU is available.
 - Releasing the entry in RS (but not ROB).
- 4. Complete: copying results to the ROB.
 - Releasing FU.
 - Mark Rd $ready \rightarrow$ forwarding happens here!
- 5. Commit/Retire: writing the results to architectural state (i.e., register and memory)
 - If this instruction is the top of ROB, commit.
 - Mark old (physical) register as free.

Out of Order Execution Example

CO: Fetch

0x100: lv	x2, 0	(x1)				٧	DestReg	"OL	D" DestRe	g PC		Comp?		FU R?		R?	
0x104: a 0x108: o	dd x4,	x3, x2 5 x4		_		0							ALU-1		У	1	у
0x10c: lw					i	0							ALU-2/	MUL	У	2	У
					i	0							MEM		у	3	у
					l	0					\neg		F	RAT	"Free"	4	y
					j	0							A-reg	РМар	Pool	5	у
						0							x1	p1	p6	6	у
						0							x2	p2	p7	7	у
	use	OP	Dest Reg	Src Reg 1		rc 1	Src Reg	2	Src 2	lmm	FU	ROB#	х3	рЗ	p8	8	у
	0				Re	eady			Ready				x4	p4	p9	9	у
	0							\dashv					х5	p5	p10	10	у
	_																
	0																
	0																

C1: Decode, Fetch



C1: Decode, Fetch

0x100: lv	v x2. 0	(x1)			V	DestReg	"OLD" DestRe	g PC		Comp?		FU R?		R	?
0x104: a 0x108: o	dd x4,	x3, x2			0						ALU-1		у	1	1
0x106. 0	ı xə, x √ x4, 0	o, x4 (x0)			0				┪		ALU-2/	MUL	у	2	1
					0						MEM		у		
					0			_	-					3	
								-	4			RAT	"Free" Pool	4	
					0						A-reg	PMap		5	
					0						x1	p1	p6	6	
					0						x2	p2	p7	7	Ī
	use	OP	Dest Reg	Src Reg 1	Src 1	Src Reg		lmm	FU	ROB#	х3	рЗ	p8	8	1
	0				Ready	'	Ready				x4	p4	p9	9	1
	0										x5	p5	p10	10	
	0														_
	0														
	-														

C2: Rename, Decode

0x100: l	w x2, 0	(x1)	lw, p6, 0(p1))	٧	DestReg	"OLD" DestRe	g PC		Comp?		FU R?		R	?
	add x4,	x3, x2	add p7, p3,		0						ALU-1		у	1	у
0x106: 0					0				┪		ALU-2/	MUL	у	2	1
					0						MEM		у		У
					0			_	-					3	У
									_		ı	RAT	"Free" Pool	4	У
1					0						A-reg	РМар	Foot	5	у
					0						x1	p1		6	у
					0						x2	6 (2)		7	у
	use	OP	Dest Reg	Src Reg 1	Src 1	Src Reg		lmm	FU	ROB#	х3	р3	p8	8	у
					Ready	'	Ready				x4	7 (4)	p9	9	у
	0										x5	p5	p10	10	v
	0										AO .	PO	P · ·	10	У
	0														
	0														
	U														

C2: Rename, Decode

0x100: lv	v x2, 0	(x1)	lw, p6, 0(p1		l V	DestReg	"OLD" DestRe	g PC		Comp?		FU R?		R	?
	dd x4,	x3, x2	2 add p7, p3,		0						ALU-1		у	1	У
0x108. 0					0				\neg		ALU-2/	MUL	у	2	У
					0						MEM		у	3	-
					0				+			RAT	"Free"	_	У
					0				-				Pool	4	У
'					0			+	-		A-reg	PMap		5	У
									_		x1	p1		6	У
					0						x2	6 (2)		7	у
	use	OP	Dest Reg	Src Reg 1	Src 1			lmm	FU	ROB#	x3	рЗ	p8	8	у
	0				Read	у	Ready				x4	7 (4)	p9	9	у
	-										x5	p5	p10	10	у
	0														ت_
	0														
	0														

C3: Dispatch, Rename

		•	,														
0x100: l	w x2. 0)(x1)	lw, p6, 0(p1))	1	٧	DestReg	"OLD" DestR	eg	PC		Comp?		FU R?		R	?
	add x4,	x3, x2	add p7, p3,			1	p6	p2		100)		ALU-1		у	1	У
0x108: 0					i i	1	р7	p4	\top	104			ALU-2/	MUL	у	2	У
i					1	0			+		_		MEM		у		-
!						0			+		-					3	У
i					i L				4		4		F	RAT	"Free" Pool	4	У
1					- 1	0			4		_		A-reg	PMap	1 001	5	у
			_	0			\perp		\perp		x1	p1		6	n		
						0							x2	6 (2)		7	n
	use	ОР	Dest Reg	Src Reg 1	Src		Src Reg		In	nm	FU	ROB#	х3	р3	p8	8	у
					Rea			Ready					x4	7 (4)	p9	9	у
	1	lw	р6	p1	У		0	У		0	2	0	x5	p5	p10	10	٧
	1	ad	р7	р3	у	•	р6	n		-	0	1		P	1	10	,
	0																
	0																

C3: Dispatch, Rename

0x100: lv	v x2. 0	(x1)	lw, p6, 0(p1))	٧	DestReg	"OLD" DestR	eg PC		Comp?		FU R?		R	?
0x104: a	dd x4,	x3, x2	add p7, p3,	p6	1	р6	p2	10	0		ALU-1		у	1	у
			or p8, p5,p7 lw p9, 0(p0)		1	р7	p4	10	4		ALU-2/	MUL	у	2	1
					0						MEM		у	3	у
					! 0				_			RAT	"Free"		У
!					0								Pool	4	У
'					-			_	_		A-reg	PMap		5	У
					0				_		x1	p1		6	n
					0						x2	6 (2)		7	n
	use	OP	Dest Reg	Src Reg 1	Src 1			lmm	FU	ROB#	x3	р3		8	у
					Read		Ready				x4	9 (7)		9	у
	1	lw	р6	p1	У	0	У	0	2	0	x5	8 (5)	p10	10	٧
	1	ad	р7	р3	У	p6	n	-	0	1		- (3)			,
	0														
	0														

C4: Issue, Dispatch

0x100: lv	v x2, 0	(x1)	lw, p6, 0(p1))	IV	DestRe	g "O	LD" DestRe	g Po		Comp?		FU R?		R	?
0x104: a	dd x4,	x3, x2	add p7, p3, or p8, p5,p7	p6	1	р6	Т	p2	10	0		ALU-1		у	1	У
			lw p9, 0(p0)		1	р7	Т	p4	10	4		ALU-2/	MUL	У	2	у
					0							MEM		У	3	у
					0							F	RAT	"Free"	4	у
i					0							A-reg	РМар	Pool	5	у
					0		┸					x1	p1		6	n
					0							x2	6 (2)		7	n
	use	OP	Dest Reg	Src Reg 1	Src 1		eg 2	Src 2	lmm	FU	ROB#	х3	рЗ		8	у
	1	lw	р6	p1	Read V	y		Ready y	0	2	0	x4	9 (7)		9	у
	1	ad	p7	р3		p		n		0	1	x5	8 (5)	p10	10	у
		au	p/	рэ	у	P	,	"		0	1					
	0															
	0															

C4: Issue, Dispatch

ı		
I	0x100: lw x2, 0(x1)	lw, p6, 0(p1)
I	0x104: add x4, x3, x2	add p7, p3, p6
	0x108: or x5, x5, x4	or p8, p5,p7
!	0x10c: lw x4, 0(x0)	lw p9, 0(p0)
1		

٧	DestReg	"OLD" DestReg	PC	Comp?
1	p6	p2	100	
1	р7	p4	104	
0				
0				
0				
0				
0				

	FU R?			
ALU-1		у	1	.,
ALU-2/	MUL	у	1	У
MEM		n	2	у
			3	у
·	RAT	"Free" Pool	4	у
A-reg	РМар	FUUL	5	у
x1	p1		6	n
x2	6 (2)		7	n
х3	рЗ		8	у
x4	9 (7)		9	у
x5	8 (5)	p10	10	у

use	OP	Dest Reg	Src Reg 1	Src 1 Ready	Src Reg 2	Src 2 Ready	lmm	FU	ROB#
0									
1	ad	р7	р3	у	р6	n	-	0	1
0									
0									

C4: Issue, Dispatch

0x100: lw x2, 0(x1) lw, p6, 0(p1) 0x104: add x4, x3, x2 add p7, p3, p6 0x108: or x5, x5, x4 or p8, p5,p7 0x10c: lw x4, 0(x0) lw p9, 0(p0)

٧	DestReg	"OLD" DestReg	PC	Comp?
1	р6	p2	100	
1	р7	p4	104	
1	p8	р5	108	
1	р9	р7	10c	
 0				
0				
0				

ALU-1		V		
		У	1	у
ALU-2/	MUL	у		
MEM		n	2	у
			3	у
	RAT	"Free" Pool	4	у
A-reg	PMap	Foot	5	у
x1	р1		6	n
x2	6 (2)		7	n
x3	р3		8	n
x4	9 (7)		9	n
x5	8 (5)	p10	10	у

use	OP	Dest Reg	Src Reg 1	Src 1 Ready	Src Reg 2	Src 2 Ready	lmm	FU	ROB#
0									
1	ad	р7	р3	у	p6	n	-	0	1
1	or	р8	p5	у	р7	n	-	1	2
1	lw	р9	p0	у	p0	у	0	2	3

C5: Complete, Issue

ı		
ı	0x100: lw x2, 0(x1)	lw, p6, 0(p1)
I	0x104: add x4, x3, x2	add p7, p3, p6
l	0x108: or x5, x5, x4	or p8, p5,p7
١	0x10c: lw x4, 0(x0)	lw p9, 0(p0)
ŀ		

٧	DestReg	"OLD" DestReg	PC	Comp?
1	p6	p2	100	у
1	р7	p4	104	
1	p8	p5	108	
1	р9	р7	10c	
0				
0				
0				

ALU-1		1/		
ALU-1		У	1	у
ALU-2/	MUL	у		1
MEM		у	2	у
			3	у
ı	RAT	"Free" Pool	4	у
A-reg	PMap	1 001	5	у
x1	р1		6	n
x2	6 (2)		7	n
x3	рЗ		8	n
x4	9 (7)		9	n
x5	8 (5)	p10	10	у

FU R?

use	OP	Dest Reg	Src Reg 1	Src 1 Ready	Src Reg 2	Src 2 Ready	lmm	FU	ROB#
0									
1	ad	р7	р3	у	p6	n	-	0	1
1	or	p8	p5	у	р7	n	-	1	2
1	lw	р9	p0	у	p0	у	0	2	3

C5: Complete, Issue

•	
0x100: lw x2, 0(x1)	lw, p6, 0(p1)
0x104: add x4, x3, x2	add p7, p3, p6
0x108: or x5, x5, x4	or p8, p5,p7
0x10c: lw x4, 0(x0)	lw p9, 0(p0)

٧	DestReg	"OLD" DestReg	PC	Comp?
1	р6	p2	100	у
1	р7	p4	104	
1	p8	p5	108	
1	р9	р7	10c	
0				
0				
0				

,,		,	1	у	ı
ALU-2	/ MUL	у	'	У	
MEM		у	2	у	
		,	3	у	
	RAT	"Free" Pool	4	у	
A-reg	РМар	Foot	5	у	
x1	p1		6	у	
x2	6 (2)		7	n	
x3	р3		8	n	
x4	9 (7)		9	n	
x5	8 (5)	p10	10	у	

FU R?

ALU-1

use	OP	Dest Reg	Src Reg 1	Src 1 Ready	Src Reg 2	Src 2 Ready	lmm	FU	ROB#
0									
1	ad	р7	р3	у	p6	n	-	0	1
1	or	р8	p5	у	р7	n	-	1	2
1	lw	р9	p0	у	p0	у	0	2	3

C5: Complete, Issue

ļ		
	0x100: lw x2, 0(x1)	lw, p6, 0(p1)
	0x104: add x4, x3, x2	add p7, p3, p6
	0x108: or x5, x5, x4	or p8, p5,p7
	0x10c: lw x4, 0(x0)	lw p9, 0(p0)
l		
ı		

٧	DestReg	"OLD" DestReg	PC	Comp?
1	p6	p2	100	у
1	р7	p4	104	
1	p8	p5	108	
1	р9	р7	10c	
0				
0				
0				

			1	У
ALU-2/	MUL	у		,
MEM		у	2	у
		,	3	у
	RAT	"Free" Pool	4	у
A-reg	РМар	7 001	5	у
x1	p1		6	у
x2	6 (2)		7	n
х3	р3		8	n
x4	9 (7)		9	n
x5	8 (5)	p10	10	у

ALU-1

use	OP	Dest Reg	Src Reg 1	Src 1 Ready	Src Reg 2	Src 2 Ready	lmm	FU	ROB#
0									
1	ad	р7	р3	у	p6	у	-	0	1
1	or	р8	p5	у	р7	n	-	1	2
1	lw	р9	р0	у	p0	у	0	2	3

C5: Complete, Issue

0x100: lw x2, 0(x1) lw, p6, 0(p1) 0x104: add x4, x3, x2 add p7, p3, p6 0x108: or x5, x5, x4 or p8, p5,p7 0x10c: lw x4, 0(x0) lw p9, 0(p0)

٧	DestReg	"OLD" DestReg	PC	Comp?
1	p6	p2	100	у
1	р7	p4	104	
1	p8	p5	108	
1	р9	р7	10c	
0				
0				
0				

ALU-1		У	1	у
ALU-2/	MUL	у	'	у
MEM		у	2	у
		,	3	у
ı	RAT	"Free" Pool	4	у
A-reg	РМар	Foot	5	у
x1	p1		6	у
x2	6 (2)		7	n
x3	р3		8	n
x4	9 (7)		9	n
х5	8 (5)	p10	10	у

FU R?

use	OP	Dest Reg	Src Reg 1	Src 1 Ready	Src Reg 2	Src 2 Ready	lmm	FU	ROB#
0									
1	ad	р7	р3	У	р6	у	-	0	1
1	or	р8	p5	у	р7	n	-	1	2
1	lw	р9	p0	у	p0	у	0	2	3

C5: Complete, Issue

0x100: lw	x2. 0	(x1)	lw, p6, 0(p1)		V	/ De	estReg "	OLD" DestRe	eg Po	PC Co		FU R?			R?	
0x104: ad	d x4,	x3, x2	add p7, p3,	p6	1		р6	p2	10	0	у	ALU-1		n	1	у
			or p8, p5,p7 lw p9, 0(p0)		1		p7	p4	10	4		ALU-2/	MUL	у	2	ļ.
		,	,		1		p8	p5	10	8		MEM		n		У
						-				-					3	У
					1	-	p9	р7	10	С		F	RAT	"Free"	4	у
			0)				A-reg	РМар	Pool	5	у				
					0)						x1	p1		6	у
					0)						x2	6 (2)		7	n
	use	OP	Dest Reg	Src Reg 1	Src :	1	Src Reg 2	Src 2	lmm	FU	ROB#	х3	р3		8	n
					Read	dy		Ready				x4	9 (7)		9	n
	0											x5	8 (5)	p10	10	у
	0											Α.Θ	0 (0)	•	10	,
	1	or	р8	p5	у		р7	n	-	1	2					
	0															

C6: Retire, Complete, Issue

0x100:	lw x2, 0)(x1)	lw, p6, 0(p1))	\	٧	DestReg	"OLD" DestR	eg P	С	Comp?		FU R?		R	?
0x104:	add x4,	x3, x2	add p7, p3,	p6	(0						ALU-1		n	1	у
			or p8, p5,p7 lw p9, 0(p0)		1	1	р7	p4	10)4		ALU-2/	MUL	у	2	У
i					1 1	1	p8	p5	10)8		MEM		n	3	У
			! 1	1	р9	р7	10	10c		217		"Free"				
i						0	P.	P.		~			RAT	Pool	4	У
					- 1	_						A-reg	PMap		5	у
						0						x1	p1	p2	6	у
					(0						x2	6 (2)		7	n
	use	OP	Dest Reg	Src Reg 1	Src		Src Reg		lmm	FU	ROB#	x3	рЗ		8	n
					Read	dy		Ready				x4	9 (7)		9	n
	0									_		x5	8 (5)	p10	10	У
	0												. ,			,
			-0	p5	У		р7	n	-	1	2					
	1	or	p8	þο	у		P -									
	0		-0	25			p7		-	1	2		9 (7)	p10	10	

C6: Retire, Complete, Issue

0x100: l	w x2, 0	(x1)	lw, p6, 0(p1)	_ 	R	٧	DestReg	"OLD" DestRe	eg PC		Comp?		FU R?		R	?
0x104: a	dd x4,	x3, x2	add p7, p3,	p6		0						ALU-1		у	1	у
			or p8, p5,p7 lw p9, 0(p0)		⇒ľ	1	р7	p4	104	4	у	ALU-2/	MUL	у	2	У
			11	1	p8	р5	108	3		MEM		n		1		
!					i I	1	p9	n7	100						3	У
					K L	_	þЭ	р7	100	٠		F	RAT	"Free" Pool	4	у
1					3	0						A-reg	РМар		5	у
						0						x1	p1	p2	6	у
						0						x2	6 (2)		7	у
	use	OP	Dest Reg	Src Reg 1	Sro	c 1	Src Reg	2 Src 2	lmm	FU	ROB#	х3	р3		8	n
					Rea	ady		Ready				x4	9 (7)		9	n
	0											x5	8 (5)	p10	10	V
	0											A.J	0 (0)	P . G	10	У
	1	or	p8	p5	У	/	р7	у	-	1	2					
	0															
	U						I									

C6: Retire, Complete, Issue

0x100: lv	0x100: lw x2, 0(x1) lw, p6, 0(p1)						DestReg	"OLD" DestReg		PC		Comp?		R	?				
0x104: a	idd x4,	x3, x2	add p7, p3,	p6		0							ALU-1		у	1	У		
	0x108: or x5, x5, x4 or p8, p5,p7 0x10c: lw x4, 0(x0) lw p9, 0(p0)						р7	p4	┪	104		У	ALU-2/	n	2	у			
i							р8	p5	1	108			MEM		n	3			
							р9	р7	+	10c					"F"		У		
i						0	Ρ,	P'					,	RAT	"Free" Pool	4	У		
									4				A-reg	PMap		5	у		
						0			4		_		x1	p1	p2	6	у		
								0							x2	6 (2)		7	у
	use	OP	Dest Reg	Src Reg 1	Src 1		Src Reg		In	Imm FU		ROB#	x3	р3		8	n		
	0				R	eady		Ready					x4	9 (7)		9	n		
	0												x5	8 (5)	p10	10	у		
	0													. ,			1,		
	1	or	р8	р5		у	р7	у		-	1	2							
	0																		

C7: Retire, Complete

0x100: lw	, x2, 0	(x1)	lw, p6, 0(p1)	d p7, p3, p6		0(p1)		V	DestReg	"OLD" DestRe	eg PC		Comp?		FU R?		R	?										
0x104: ad	dd x4,	x3, x2	add p7, p3,			0						ALU-1		у	1	У												
			lw p9, 0(p0)										or p8, p5,p7 I w p9, 0(p0) I									\neg		ALU-2/	MUL	n	2	
						_		1	p8	p5	10	8		MEM		n		У										
!									10						3	у												
							р9	р7	10	C		F	RAT	"Free"	4	у												
						0						A-reg	РМар		5	у												
					C	0						x1 p1		p2	6	у												
					C	0						x2	6 (2)	p4	7	у												
	use	ОР	Dest Reg	Src Reg 1	Src 1		Src Reg		lmm	FU	ROB#	х3	рЗ		8	n												
					Read	dy		Ready				x4	9 (7)		9	n												
	0											x5	8 (5)	p10	10	у												
	0												, ,															
	0										p1 p2 p4 p3 p 9 (7)																	

C7: Retire, Complete

0x100: lw x2, 0(x1)	lw, p6, 0(p1))	٧	DestReg	"OLD" DestRe	PC		Comp?		FU R?		R?							
0x104: add x4, x3, x2 0x108: or x5, x5, x4	add p7, p3,	p6	0						ALU-1		у	1	у						
0x10c: lw x4, 0(x0)													\neg		ALU-2/	MUL	у	2	у
		1	р8	р5	108	108 y		MEM		У	3	У							
			1	р9	р7	10c	T	у		RAT		4	y						
		0				1		A-reg	PMap	Pool	5	У							
							7				p2	6	У						
							+		x1 p1 x2 6 (2)	p4	7	У							
use OP				Sua Dan	2 5-2	lum	FU	ROB#	x3	p3		8	У						
use OP	Dest Reg	Src Reg 1	Src 1 Ready	Src Reg	2 Src 2 Ready	lmm	FU	KUB#				9							
0									x4	9 (7)	n10		У						
0									x5	8 (5)	p10	10	У						
0																			
0																			

C8: Retire

0x100: lv	0x100: lw x2, 0(x1) lw, p6, 0(p1)							"OLD" Dest	'OLD" DestReg PC			Comp?			R?		
			add p7, p3,		1	0							ALU-1		у	1	у
	0x108: or x5, x5, x4 or p8, p5,p7 0x10c: lw x4, 0(x0) lw p9, 0(p0)										┪		ALU-2/ MUL		у	2	у
													MEM		у	3	
	!										-				" — ••		У
i														RAT	"Free" Pool	4	У
1					•	0					_		A-reg	РМар		5	у
						0					_		x1	p1	p2	6	у
						0							x2	6 (2)	p4	7	у
	use	OP	Dest Reg	Src Reg 1		rc 1	Src Reg			lmm	FU	ROB#	х3	р3	p5	8	у
	0				Re	eady		Ready					x4	9 (7)	p7	9	У
	0								_				x5	8 (5)	p10	10	у
	0																-
	0																
	0																

C9:...

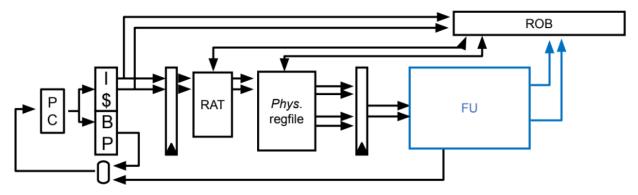
0x100: lv	0x100: lw x2, 0(x1) lw, p6, 0(p1)							"OLD" DestR	LD" DestReg			Comp?			R	?			
0x104: a	dd x4,	x3, x2	add p7, p3,	p6			! [П				ALU-1		у	1	у
0x106: 0	0x108: or x5, x5, x4 or p8, p5,p7 0x10c: lw x4, 0(x0) lw p9, 0(p0)								┪		\neg		ALU-2/	MUL	у	2	у		
i													MEM		у	1 2 3 4 5 6 7 8	-		
!									-		-					3	У		
						0					4		RAT		"Free" Pool	4	У		
	! =								┙				A-reg	РМар		5	у		
									┙				x1	p1	p2	6	у		
						0							x2	6 (2)	p4	7	у		
	use	OP	Dest Reg	Src Reg 1	Src 1		Src Reg		lr	lmm		ROB#	х3	рЗ	p5	8	у		
					Re	eady		Ready					x4	9 (7)	p7	9	у		
	0												x5	8 (5)	p10	10	у		
	0													- (3)			,		
	0																		
	0																		

Summary

We now have a way to execute instructions out of order with the three parts discussed earlier:

- $\bullet\,$ A mechanism for tracking instructions
 - Later instructions might be **ready**
- \bullet A mechanism for removing data hazards
 - ${\bf New}$ hazards: WAW, WAR, load-store
- A mechanism for recovery
 - Speculation (e.g., branch misprediction), etc.

Datapath with ROB, RAT, FU



• Each unit can be further split into multiple stages (not shown here).