Superscalar* Club Meeting #4

*we really mean: superscalar speculative out-of-order

James C. Hoe Department of ECE Carnegie Mellon University

[Internal Use Draft] Superscalar Club Meeting #4, Slide 1, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Goal set when we started

- Achieve an RTL-precise understanding of superscalar speculative out-of-order register dataflow—as with 5-stage pipeline in 18-447
 - know you could make a working design and know what it does (how good is it?)
 - know shortcomings and limits in the simplifications to help get started
 - know what you have not figured out yet

Test: Able to read the R10K paper and say definitely what you understand and what you don't

Plan of Attack

- Focus
 - mainly on register dataflow
 - lightly on memory dataflow
 - not at all on i-fetch (a well decoupled subject both conceptually and physically)
- Path
 - further develop concepts in L20 we didn't have time for
 - study Metaflow DRIS to flesh out conceptual-level understanding
 - study how things were really done in R10K
 - play with an RTL-precise executable model (in C++)

Review: Tomasulo with Physical Register File Rename

Problem Setup

I0:
$$r4 \leftarrow r4 + r6$$
I1: $r2 \leftarrow r2 + r6$ I2: $r4 \leftarrow r2 * r4$ I3: $r4 \leftarrow r4 + r6$ I4: $r2 \leftarrow r2 + r6$ I5: $r2 \leftarrow r2 + r4$

ROB	Dst	Sr	rc1	Sr	rc2	ROF	B Dst	S	rc1	S	rc2	Μ	ap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	ID		Tag	Rdy	Tag	Rdy	Ta	ble	Α	6	У
		8	•	8				8		8	•	r2	J	В	1	У
							_							С	6	У
												F4	U	D	2	У
						Mu	lt: stage	1() stag	e 2 ()	r6	I	E	10	У
-	Ado	l: stage	e1()								r8	C	F	5	У
		0												G	0	У
	7	6	5	4	3	2	1	0			head		tail	Н	10	У
DOD	, 			-			<u> </u>		Ence I	•••• [CEE		D	Ι	3	У
ROB									r ree 1		θ, Ε, Γ,	, n, n,	D	J	2	У

[Internal Use Draft] Superscalar Club Meeting #4, Slide 5, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Timing Assumptions (Arbitrary)

- Up to two instructions of any type can be dispatched to their respective RS in each cycle
- An instruction can begin execution as early as the same cycle it is dispatched to an RS
- Execution priority is given to the older instruction in RS
- One 1-cyc adder and one pipelined 2-cyc multiplier
- Producer notifies its dependent insts in its last execution cycle (CBD-based, no "dead reckoning")
- An instruction is removed from RS after its last execution cycle
- If appropriate, an instruction can commit/retire as early as the cycle after its last execution cycle.

Worksheet for You to Try



Initial State @ Start of Cycle #1

ROB	Dst	Sr	c1	Sr	rc2	ROB	Dst	Sı	rc1	S	rc2	Μ	ap	Phy.	Reg	Rdy
ID		Тад	Rdv	Tag	Rdv	ID		Тад	Rdv	Tag	Rdv	Ta	ble	Α	6	У
								8		8		r2	J	В	1	У
														С	6	У
												r4	U	D	2	У
						Mul	t: stage	1 () stag	e 2 ()	r6	I	E	10	У
	Ado	l: stage	e1()								r8	C	F	5	У
						l.								G	0	У
	7	6	5	4	3	2	1	0			head		tail	H	10	У
DOD				-					Ence I	••• [GEE		D	Ι	3	У
KOB									r ree 1		Θ, Ε, Γ,	, п, п,	D	J	2	У

End of Cycle #1

I0: r4 ← r4 + r6; **I1:** r2 ← r2 + r6

ROB	Dst	Sr	c1	Sr	c2	ROB	Dst	Sı	·c1	Sr	c2	N	Iap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	ID		Tag	Rdy	Tag	Rdy	T	able	A	6	У
0	G	0	V	т	V							r2	E	B	1	У
0	0	0	,	1	,	r4 G						С	6	У		
1	E	J	У	Ι	У							r4	6	D	2	У
						Mul	t <mark>: stag</mark> e	1 () stag	e 2 ()	r6	I	E	10	N
	Add	: stage	1(0)								r8	С	F	5	У
L		0				J								G	0	N
	7	6	5	4	3	2	1	0			head		tail	Н	10	У
DOD			1			_	T4	TO	Б. Т	· . [6		I	3	У
ROB							11/J	1VD	Free L		г, А, Н	, в		J	2	У

[Internal Use Draft] Superscalar Club Meeting #4, Slide 8, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Carnegie Mellon

End of Cycle #1

Е

Α

1

3

J

F

I0: r4 ← r4 + r6; **I1:** r2 ← r2 + r6

r2

r4

r6

r8

E

A

Ι

Y

Y

N

N

Y

Y

Y

Y

6

2

10

С

D

E

ROB	Dst	Sr	·c1	Sr	c2	RO	B Dst	S	rc1	Sı	·c2	M	lap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	II		Tag	Rdy	Tag	Rdy	Ta	ible	A	6	У
0	G	D	У	I	У							r2	E	B	1	y V
1	E	т	V	т	V						1. 	r4	G	C	6	У
1	C	J	y	L	y							-	T	D	2	У
						Mult: stage 1 () stage 2 () r6 I									10	N
	Add	: stage	1(0)	1							r8	С	F	5	У
<u> </u>		0				1								G	0	Ν
	7	6	5	4	3	2	1	0			head		tail	Н	10	У
DOD		-	1	1				TO	EI	са Г	E A L			Ι	3	У
ROR							<u> </u> /J		Free I		Г, А, Р	і, В		J	2	У
End of	Cycle	#2					11:	r2 ←	r2 + r	⁻ 6; 2	:r4 ←	– r2 *	* r4; I	3: r4 ·	← r4	+ r6
ROB	Dst	Sr	c1	Sr	c2	RO	B Dst	Si	rc1	Sr	rc2	Μ	lap	Phy.	Reg	Rdy
ID		Тад	Rdv	Тад	Rdy	ID		Тад	Rdy	Тад	Rdy	Ta	ble	Α	6	N
		Lag	Ruy	Lag	Ruy			rag	Ruy	Tag	nuy		E	В	1	У

	Add	: stage	1(1)							r8	С	F	5
		0]								G	5
	7	6	5	4	3	2	1	0		head		tail	Η	10
DOD	,				т <u>э</u>	- T2	T1.	v	Enco Lint				I	3
ROB					13/F	14/G	11/J		Free List	п, в, о			J	2

F

Mult: stage 1 (

2

Е

G

y

)

N

) stage 2 (

[Internal Use Draft] Superscalar Club Meeting #4, Slide 9, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Ι

Ι

y

y

Y

N

Carnegie Mellon

J

2

У

End of Cycle #2

I1: r2 ← r2 + r6; **I2:** r4 ← r2 * r4; **I3:** r4 ← r4 + r6

	-															
ROB	Dst	Sr	rc1	Sr	rc2	ROB	Dst	Sr	rc1	Sr	rc2	Μ	ap	Phy	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	ID		Tag	Rdy	Tag	Rdy	Ta	ble	A	6	N
						2	F	F	N	G	y	r2	Е	B	1	У
1	E	т	V	т	V			_				r4	A	С	6	У
1		5	,	-	,							r6	Т	D	2	Y
3	A	۲	N	1	У	Mult	: stage	1() stag	e 2 ()	0	-	E	10	N
	Add	l: stage	1(1)								rð	C	F	5	N
						-								G	5	Y
	7	6	5	4	3	2	1	0			head		tail	Н	10	Y V
					тэ	T2.	T1 /		Free I	ist [HRD	8		I	3	У
ROB					L3/		TT/I		I I CC I	100	· ·, Þ, Ö	243 - Contra 1997 - Contra 199			-	
ROB					13/1		11/J		TICC I		, 0, 0			J	2	У
<mark>ков</mark> End of	Cycle	e #3		12:	<u> 13/</u> r4 ←	r2 * r4;	; I3: r	 4 ←	r4 + i	r6; 4	r, b, b	– r2 +	- r6;	 I5: r2	∣ ² ← r2	y + r4
ROB End of ROB	Cycle Dst	e #3	rc1	12: Sr	13/1 r4 \leftarrow rc2	r2 * r4;	; I3: r	 4 ← 	r4 + 1 vc1	r6; 4 Sr	: r2 ←	– r2 +	- r6; ap	J 15: r2 Phy	2 ← r2	y + r4 Rdy
ROB End of ROB ID	Cycle Dst	e #3 Si Tag	rc1 Rdy	I2: Sr Tag	$r4 \leftarrow rc2$	r2 * r4;	; I3: r	24 ← Sr Tag	r4 + 1 rc1 Rdy	r6; 4 Sr Tag	r^{0} r2 \leftarrow rc2 Rdy	– r2 + M Ta	- r6; ap ble	J 15: r2 Phy. A	2 ← r2 Reg 6	y + r4 Rdy N
ROB End of ROB ID	Cycle Dst	e #3 Si Tag	rc1 Rdy	I2: Sr Tag	$r4 \leftarrow rc2$ Rdy	r2 * r4	13: r	4 ← Sr Tag	r4 + 1 rc1 Rdy	r6; 14 Sr Tag	: r2 ← •c2 Rdy	- r2 + M Ta r2	- r6; ap ble B	J 15: r2 Phy. A B	2 ← r2 Reg 6 1	y + r4 Rdy N N
ROB End of ROB ID 4	Cycle Dst H	e #3 Si Tag E	rc1 Rdy Y	I2: Sr Tag I	$r4 \leftarrow rc2$ Rdy y	r2 * r4; ROB ID 2	13: r Dst	4 ← Sr Tag E	r4 + i rc1 Rdy y	r6; 4 Sr Tag <i>G</i>	: r2 ← ·c2 Rdy Y	- r2 + M Ta r2 r4	r6; ap ble B	J 15: r2 Phy. A B C	2 ← r2 Reg 6 1 6	y + r4 Rdy N N y
ROB End of ROB ID 4 5	Cycle Dst H B	e #3 Si Tag E H	rc1 Rdy Y N	I2: Sr Tag I A	$r4 \leftarrow rc2$ Rdy V N	r2 * r4; ROB ID 2	I3: r Dst	4 ← Sr Tag E	r4 + i c1 Rdy y	r6; 4 Sr Tag <i>G</i>	: r2 ← ·c2 Rdy Y	- r2 + M Ta r2 r4	r6; ap ble B A	J I5: r2 Phy A B C D	2 ← r2 Reg 6 1 6 2	y +r4 Rdy N N y y
ROB End of ROB ID 4 5 3	Cycle Dst H B A	e #3 Si Tag E H F	rc1 Rdy Y N N	I2: Sr Tag I A I	r4 ← c2 Rdy y N y	r2 * r4; ROB ID 2 Mul	I3: r Dst F t: stage	24 ← Sr Tag E 21 (2	r4 + r c1 Rdy y) stag	r6; 14 Sr Tag <i>G</i> ge 2 (r2 ← c2 Rdy y)	- r2 + M Ta r2 r4 r6	- r6; ap ble B A I	J I5: r2 Phy A B C D E	2 ← r2 Reg 6 1 6 2 5	y + r4 Rdy N N y y y
ROB End of ROB ID 4 5 3	Cycle Dst H B A Add	e #3 Si Tag E H F : stage	rc1 Rdy Y N N 1 (4	I2: Sr Tag I A I)	r4 ← c2 Rdy y N y	r2 * r4; r2 * r4; ID 2 Mul	I3: r Dst F t: stage	4 ← Sr Tag E 21(2	r4 + 1 c1 Rdy y) stag	r6; 14 Sr Tag <i>G</i> ge 2 (r2 ← c2 Rdy y)	- r2 + M Ta r2 r4 r6 r8	- r6; ap ble B A I C	J I5: r2 Phy A B C D E F	2 ← r2 Reg 6 1 6 2 5 5	y + r4 Rdy N N V y y V N
ROB End of ROB ID 4 5 3	Cycle Dst H B A Add	e #3 Si Tag E H F : stage	rc1 Rdy y N N 1 (4	I2: Sr Tag I A I)	r4 ← rc2 Rdy y N y	r2 * r4; ROB ID 2 Mul	I3: r Dst F t: stage	4 ← Sr Tag E 21(2	r4 + 1 ·c1 Rdy y) stag	r6; 14 Sr Tag <i>G</i> ge 2 (: r2 ← ·c2 Rdy y	- r2 + M Ta r2 r4 r6 r8	- r6; ap ble B A I C	J I5: r2 Phy A B C D E F G	2 ← r2 Reg 6 1 6 2 5 5 5 5	y + r4 Rdy N N y y N y N
ROB End of ROB ID 4 5 3	Cycle Dst H B A Add	e #3 Si Tag E H F : stage	rc1 Rdy Y N N 1(4	I2: Sr Tag I A I)	$r 4 \leftarrow rc2$ $R dy$ y N y 3	r2 * r4; r2 * r4; ID 2 Mul	1 17J	64 ← Sr Tag E e 1 (2	r4 + 1 rc1 Rdy y) stag	r6; 14 Sr Tag <i>G</i> ge 2 (r, b, b r, c 2 Rdy y)	- r2 + M Ta r2 r4 r6 r8	- r6; ap ble B A I C tail	J I5: r2 Phy A B C D E F G H	2 ← r2 Reg 6 1 6 2 5 5 5 10	y + r4 Rdy N N y y N y N
ROB End of ROB ID 4 5 3	Cycle Dst H B A Add 7	e #3 Si Tag E H F : stage 6	rc1 Rdy Y N N 1(4 5	I2: Sr Tag I A I) 4 T4/	$r 4 \leftarrow rc2$ $rc2$ Rdy y N y 3 $T3/$	r2 * r4; r2 * r4; ID 2 Mul	1	64 ← Sr Tag E e 1 (2	r4 + 1 c1 Rdy y) stag	r6; 14 Sr Tag <i>G</i> ge 2 (: r2 ← ·c2 Rdy y)	- r2 + M Ta r2 r4 r6 r8	- r6; ap ble B A I C tail	J I5: r2 Phy A B C D E F G H H I	2 ← r2 Reg 6 1 6 2 5 5 5 10 3	y + r4 Rdy N N y y N y N y

[Internal Use Draft] Superscalar Club Meeting #4, Slide 10, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

G

F

E

H

Carnegie Mellon End of Cycle #3 **I2:** r4 ← r2 * r4; **I3:** r4 ← r4 + r6; **I4:** r2 ← r2 + r6; **I5:** r2 ← r2 + r4 Phy. Reg Rdy **ROB** Src1 Src2 ROB Dst Src1 Src2 Dst Map Table 6 N A ID ID Tag Tag Rdy Rdy Tag Rdy Tag Rdy 1 N B **r**2 В Y F Y Y H F Y Τ 2 E G 4 Y 6 С A r4 5 B Н N A N 2 Y D Ι r6 3 A F N Ι y Mult: stage 1 (2) stage 2 () E 5 Y С **r8** 5 F N Add: stage 1 (4) 5 G Y 10 H N 6 2 1 head tail 7 5 4 3 0 Y 3 I D, J I4/ I3/ I2/ **Free List** ROB I5/ Y 2 J H F F G End of Cycle #4 **12:** r4 ← r2 * r4; **13:** r4 ← r4 + r6; **14:** r2 ← r2 + r6; **15:** r2 ← r2 + r4 ROB Dst Src1 Src2 ROB Dst Src1 Src2 Map Phy. Reg Rdy Table A 6 N ID ID Tag Rdy Tag Rdy Tag Rdy Tag Rdy B 1 N В **r**2 F E Y Y 2 G 6 Y C A r4 Y В H 5 A N Y 2 D Ι r6 3 A F N Ι Y Mult: stage 1 () stage 2 (2 Y E 5) С **r8**

5

5

8

3

2

N

Y y

Y

Y

F

G

H

Ι

J

tail

head

D, J

Free List

[Internal Use Draft] Superscalar Club Meeting #4, Slide 11, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

3

I3/

F

2

I2/

G

1

0

Add: stage 1 (

6

7

ROB

)

4

I4/

F

5

I5/

H

Carnegie Mellon

End of	Cycle	e #4		12: I	ŕ4 ←	r2 * r4	; I3: r	4 ←	r4 + r	6; 4	r2 ←	- r2 +	r6; I	<mark>5:</mark> r2 ∢	— r2	+ r4
ROB	Dst	Sr	rc1	Sr	c2	ROB	Dst	Sı	rc1	Sr	c2	M	ap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	ID		Tag	Rdy	Tag	Rdy	Ta	ble	Α	6	N
		0				2	F	F	y y	G	y	r2	B	B	1	N
5	R	Ц	V	Δ	N		· ·	_			•	r4	A	С	6	У
								1 (<u> </u>	r6	I	D	2	y V
3	A	r	IN	T	У	Mu	lt: stage	e I () stage	2(2)	r8	C	E	5	y N
	Add	l: stage	e 1 ()								10	~	r C	5	N
														G U	8	y V
	7	6	5	4	3	2	1	0		_	head		tail	II I	3	v
ROB			15/	I4/	I3/	I2/			Free I	list	D, J			1	2	y
	- 1		Н	E	F	G		-	_		-	-			_	
End of	Cycle	e #5					I3: r	4 ←	r4 + r	6; 14	r2 ←	- r2 +	r6;	5:r2 ∢	— r2	+ r4
ROB	Dst	Sr	rc1	Sr	c2	ROB	Dst	Sı	rc1	Sr	rc2	M	ap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	ID		Tag	Rdy	Tag	Rdv	Ta	ble	Α	6	N
		0		0	•		-	0								
5											Itaj	r2	В	B	1	N
	D	L.	V	Δ	N							r2 r4	B	B C	1 6	N y
	В	н	У	A	N							r2 r4	B A T	B C D	1 6 2	N y y
3	B A	H F	y y	A I	N Y	Mı	llt: stag	ge 1 () stage	e 2 ()	r2 r4 r6	B A I	B C D E	1 6 2 5	N Y Y Y
3	B A Add	H F : stage	У У 1(3	A I)	N y	Mı	ılt: stag	ge 1 () stage	e 2 ()	r2 r4 r6 r8	B A I C	B C D E F	1 6 2 5 25	N Y Y Y Y Y
3	B A Add	H F : stage	у У 1(3	A I)	N y	Mı	ılt: stag	je 1 () stage	e 2 ()	r2 r4 r6 r8	B A I C	B C D E F G	1 6 2 5 25 5 5	N Y Y Y Y Y Y Y
3	B A Add 7	H F : stage 6	y y 1(3 5	A I) 4	N y 3	2	ılt: stag	ge 1 () stage	e 2 () head	r2 r4 r6 r8	B A I C tail	B C D E F G H	1 6 2 5 25 5 5 8 25	N Y Y Y Y Y Y Y Y Y Y Y
3 ROB	B A Add 7	H F : stage 6	y y 1(3 5 I5/	A I) 4 I4/	N y 3 I3/	2	llt: stag	0) stage	e 2 () head D, J,G	r2 r4 r6 r8	B A I C tail	B C D F G H I	1 6 2 5 25 5 8 3 3 2	N y y y y y y y y

[Internal Use Draft] Superscalar Club Meeting #4, Slide 12, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

End of Cycle #5

I3: r4 ← r4 + r6; **I4:** r2 ← r2 + r6; **I5:** r2 ← r2 + r4

ROB	Dst	Sr	rc1	Sr	c2	ROI	B Dst	S	rc1	Sı	rc2	N	Iap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	ID		Tag	Rdy	Tag	Rdy	T	able	Α	6	N
		-										r2	B	B	1	N
		212						-	-			r/	Δ	С	6	У
5	B	Н	У	A	Ν							17		D	2	У
3	A	F	У	Ι	У	N	Iult: sta	ge 1 () stage	e <mark>2 (</mark>)	r6	I	E	5	У
7	Add	: stage	1(3)								r8	C	F	25	У
														G	5	Y
	7	6	5	4	3	2	1	0			head		tail	Н	8	У
DOD			TE/	TAI	T2/				Ence I	•	D TG			Ι	3	У
ROB			H	E	F				Free I		0, 1,9			J	2	У
End of	Cycle	#6			1									<mark>5:</mark> r2 ∢	← r2	+ r4

ROB	Dst	Sr	rc1	Sr	rc2	R	OB	Dst	S	rc1	Sr	rc2		M	ap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy]	D		Tag	Rdy	Tag	Rdy		Ta	ble	Α	28	У
			•						8		8			r2	B	В	1	N
															Δ	С	6	У
5	В	Н	У	A	У									r4	~	D	2	У
							Mu	lt: stag	e 1 () stage	e 2 ()		r6	I	E	5	У
	Add	: stage	1(5)					· · · · · ·	2000 C	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		' [r8	С	F	25	У
						ļ										G	5	У
	7	6	5	4	3	2		1	0			head			tail	H	8	y.
DOD	,					_		-		E.s. I	••• [D TC	F	E		Ι	3	У
ROB			H							r ree 1		0, J,0	<u>،</u> ۲	,C		J	2	У

[Internal Use Draft] Superscalar Club Meeting #4, Slide 13, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Carnegie Mellon

G

Н

Ι

J

5

8

3

2

y y

Y

Y

End of Cycle #6

I5: r2 ← r2 + r4

ROB	Dst	Sr	·c1	Sr	c2	ROB	Dst	Sr	rc1	Sr	c2	M	ap	Phy.	Reg	Rdy
ID		Tag	Rdy	Tag	Rdy	ID		Tag	Rdy	Tag	Rdy	Ta	ble	Α	28	У
		0			•			8		8		r2	B	В	1	N
	0		~									r4	A	С	6	У
5	R	н	У	A	У								т	D	2	У
						M	ult: stag	e 1 () stage	e 2 ()	ro	T	Е	5	У
	Add	: stage	1(5)								r8	C	F	25	У
		0												G	5	Y
	7	6	5	4	3	2	1	0			head		tail	Η	8	У
DOD	,	<u> </u>				-	-		Ence I	••• [D TG	FE		Ι	3	У
ROB			15/ H						r ree 1		0, 3,8,	, , C	n	J	2	У
End of	Cycle	e #7														
ROB	Dst	Sr	rc1	Sr	c2	ROB	Dst	Sr	rc1	Sr	c2	M	ap	Phy.	Reg	Rdy
ID		Tag	Rdv	Tag	Rdv	ID		Tag	Rdv	Tag	Rdy	Ta	ble	Α	28	У
							-	8				r2	В	В	36	У
												r4	A	С	6	У
												14	T	D	2	У
						M	ult: stag	e 1 () stage	e 2 ()	ro	L	E	5	У
	Ado	l: stage	e 1 ()								r8	C	F	25	У

	7	6	5	4	3	2	1	0		head	tail	
ROB									Free List	D, J,G,F,E	,Н	╞

[Internal Use Draft] Superscalar Club Meeting #4, Slide 14, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Precise Exception Rewind

 Suppose at the end of I2's second cycle of execution, I2 encounters an exception and cannot complete.



See <u>https://github.com/jhoecmu/ooo-beta</u> for more

[Internal Use Draft] Superscalar Club Meeting #4, Slide 15, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Memory Dataflow Concepts

Superscalar Speculative Out-of-Order

- Modern CPUs can have over 100 instructions in out-of-order execution scope
 - ROB has 100's of entries
 - RS do not have 100's of entries
- Keep in mind, if average basic block is 7 inst, over 10 levels of branch prediction
- Important Question:
 - how much more ILP is uncovered with look ahead
 - how much <u>useful</u> work is done during look ahead
 Ans: not much and not much

Truth about Superscalar Speculative OOO

- If memory speed kept up with core speed, we would still be building in-order pipelines
- But, by 2005 we were seeing e.g., Intel P4 at 4+GHz
- 16KB L1 D-cache
 - t₁ = 4 cyc int (9 cycle fp)
 - 1024KB L2 D-cache
 - t₂ = 18 cyc int (18 cyc fp)
 - Main memory

- t₃ = ~ 50ns or 180 cyc

- Speculative OOO has really been about
 - finding independent work to do after cache hit&miss
 - getting to future cache misses as early as possible
 - overlapping multiple cache misses for BW (aka MLP)

Scheduling Memory Operations

- Memory data dependence \Rightarrow RAW, WAR, WAW
- Addresses use GPR ⇒ register dependence
- Storing has side-effect that cannot be undone
 ⇒ must wait until commit
- When earliest to safely start LW <u>on uniprocessor</u>?
 - no more older pending SW OR
 - no older SW with conflicting address (requires knowing all older SW addresses) OR
 - just go if no known conflict; reload if new RAW hazard later

What about MP memory consistency?

IBM 360/91 FP Module [1967]



[Internal Use Draft] Superscalar Club Meeting #4, Slide 20, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Tomasulo's Algorithm

Instruction state	Wait until	Action or bookkeeping	A
Issue FP operation	Station r empty	<pre>if (RegisterStat[rs].Qi≠0) {RS[r].Qj ← RegisterStat[rs].Qi} else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; if (RegisterStat[rt].Qi≠0) {RS[r].Qk ← RegisterStat[rt].Qi else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0}; RS[r].Busy ← yes; RegisterStat[rd].Q ← r;</pre>	Patterson, CAA
Load or store	Buffer r empty	<pre>if (RegisterStat[rs].Qi≠0) {RS[r].Qj ← RegisterStat[rs].Qi} else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0}; RS[r].A ← imm; RS[r].Busy ← yes;</pre>	ennessy&
Load only		RegisterStat[rt].Qi \leftarrow r;	Ľ.
Store only	· · · · · ·	<pre>if (RegisterStat[rt].Qi≠0) {RS[r].Qk ← RegisterStat[rs].Qi} else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0};</pre>	trom
Execute FP operation	(RS[r].Qj = 0) and (RS[r].Qk = 0)	Compute result: operands are in Vj and Vk	
Load-store step 1	RS[r].Qj = 0 & r is head of load-store queue	$RS[r].A \leftarrow RS[r].Vj + RS[r].A;$	
Load step 2	Load step 1 complete	Read from Mem[RS[r].A]	
Write Result FP operation or load	Execution complete at r & CDB available	<pre>∀x(if (RegisterStat[x].Qi=r) {Regs[x] ← result; RegisterStat[x].Qi ← 0}); ∀x(if (RS[x].Qj=r) {RS[x].Vj ← result;RS[x].Qj ← 0}); ∀x(if (RS[x].Qk=r) {RS[x].Vk ← result;RS[x].Qk ← 0}); RS[r].Busy ← no;</pre>	
Store	Execution complete at $r \& RS[r].Qk = 0$	$\begin{array}{l} Mem[RS[r].A] \ \leftarrow \ RS[r].Vk;\\ RS[r].Busy \ \leftarrow \ no; \end{array}$	

RS entry

- Busy: in use
- **Op**: operand
- Vj: op1 value
- Vk: op2 value
- Qj: op1 RS-tag
- Qk: op2 RS-tag

case (RegisterStat)
O: RF val current
RS-tag:
 to be produce by
 corresponding
 instruction

[Internal Use Draft] Superscalar Club Meeting #4, Slide 21, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Tomasulo + Speculative Execution



[Internal Use Draft] Superscalar Club Meeting #4, Slide 22, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute:] All rights reserved.

Tomasulo's Algorithm + ROB

Status	Wait until	Action or bookkeeping
Issue all instructions	Reservation station (r) and ROB (b)	<pre>if (RegisterStat[rs].Busy)/*in-flight instr. writes rs*/ {h ← RegisterStat[rs].Reorder; if (ROB[h].Ready)/* Instr completed already */ {RS[r].Vj ← ROB[h].Value; RS[r].Qj ← 0;} else {RS[r].Qj ← h;} /* wait for instruction */ } else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0;}; RS[r].Busy ← yes; RS[r].Dest ← b; ROB[b].Instruction ← opcode; ROB[b].Dest ← rd;ROB[b].Ready ← no; </pre>
FP operations and stores	both available	if (RegisterStat[rt].Busy) /*in-flight instr writes rt*/ {h \leftarrow RegisterStat[rt].Reorder; if (ROB[h].Ready)/* Instr completed already */ {RS[r].Vk \leftarrow ROB[h].Value; RS[r].Qk \leftarrow 0;} else {RS[r].Qk \leftarrow h;} /* wait for instruction */ } else {RS[r].Vk \leftarrow Regs[rt]; RS[r].Qk \leftarrow 0;};
FP operations		RegisterStat[rd].Reorder \leftarrow b; RegisterStat[rd].Busy \leftarrow yes; ROB[b].Dest \leftarrow rd;
Loads		$\begin{array}{rl} \text{RS[r].A} \leftarrow \text{imm; RegisterStat[rt].Reorder} \leftarrow \text{b;} \\ \text{RegisterStat[rt].Busy} \leftarrow \text{yes; ROB[b].Dest} \leftarrow \text{rt;} \end{array}$
Stores	mallor.	$RS[r].A \leftarrow imm;$

ROB Entry:

is **Busy** / **Instruction** / logical **Dest** reg / dest **Value** / value is **Ready** RegisterStat: reg is **Busy** / renamed to **Reorder** buffer entry #

[Internal Use Draft] Superscalar Club Meeting #4, Slide 23, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Tomasulo's Algorithm + ROB

Execute FP op	(RS[r].Qj == 0) and (RS[r].Qk == 0)	Compute results—operands are in Vj and Vk	
Load step 1	(RS[r].Qj == 0) and there are no stores earlier in the queue	$RS[r].A \leftarrow RS[r].Vj + RS[r].A;$	
Load step 2	Load step 1 done and all stores earlier in ROB have different address	Read from Mem[RS[r].A] subtle	
Store	$\frac{(RS[r].Qj == 0) \text{ and}}{\text{store at queue head}}$	ROB[h].Address ← RS[r].Vj + RS[r].A;	
Write result all but store	Execution done at <i>r</i> and CDB available	b ← RS[r].Dest; RS[r].Busy ← no; $\forall x(if (RS[x].Qj==b) \{RS[x].Vj \leftarrow result; RS[x].Qj \leftarrow 0\});$ $\forall x(if (RS[x].Qk==b) \{RS[x].Vk \leftarrow result; RS[x].Qk \leftarrow 0\});$ ROB[b].Value ← result; ROB[b].Ready ← yes;	AADAI
Store	Execution done at r and (RS[r].Qk == 0)	ROB[h].Value ← RS[r].Vk;	Prson, C
Commit	Instruction is at the head of the ROB (entry h) and ROB[h].ready == yes	<pre>d ← ROB[h].Dest; /* register dest, if exists */ if (ROB[h].Instruction==Branch) {if (branch is mispredicted) {clear ROB[h]. RegisterStat; fetch branch dest;];} else if (ROB[h].Instruction==Store) {Mem[ROB[h].Destination] ← ROB[h].Value;} else /* put the result in the register destination */ {Regs[d] ← ROB[h].Value;}; ROB[h].Busy ← no; /* free up ROB entry */ /* free up dest register if no one else writing it */ if (RegisterStat[d].Reorder==h) {RegisterStat[d].Busy ← no;};</pre>	from [Hennessv&Patte

[Internal Use Draft] Superscalar Club Meeting #4, Slide 24, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

DRIS is also Address Queue

- LW/SW issued 1st-time as "ADD"
 - compute base+offset
- COMPLET
 result written to Data; complete ...
 SW issue 2nd-time on retire Comparators do you imaginess width and time when RAW-free Comparators do you imagines?
- - check (by CAM) LW addr against older SW addr's 2.
 - 3. If conflict, forward youngest matching SW to LW
- OOO for shared-memory MP must obey additional ordering rules

Memory Dataflow Realities

[Internal Use Draft] Superscalar Club Meeting #4, Slide 26, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

What makes it hard

- Memory dependencies are not static
 - LW and SW addresses need to be calculated, and
 - translated (to check RAW/WAR/WAW synonyms)
- Memory addresses are wider than register names
- Memory instructions take longer to execute relative to other instructions types
- Scheduling takes into account O(N²) dependency relationships in program order
- More special ordering rules if shared memory MP
- Want to load as soon as possible!!!

R10K Load Data Path



stage 2 ^I stage 3 Control stage 4 ^I stage 5 [Fig 10, Yeager 1996, IEEE Micro] [Internal Use Draft] Superscalar Club Meeting #4, Slide 28, James C. Hoe, CMU/ECE/CALCM, © 2023 [Do not redistribute.]

Memory Disambiguation



FIGURE 6.8: Schematics of the MIPS R10000 pipeline to implement the partial ordering memory disambiguation policy. from [Gonzalez, et al., 2010]

Address Queue (16 entries)

- Track LW/SW and addr in program order
- Addr calculation issued in dataflow order
- N-comparators compares new/external addr to each known-so-far addr to detect
 - LW/SW RAW hazards
 - cache line return after load miss
 - external invalidations (more later)
- Pair-wise RAW hazards matrix (N² bits) updated incrementally after each LW/SW addr becomes known
- LW do not issue against known/potential RAW hazard or pending cache miss



&translate



[Internal Use Draft] Superscalar Club Meeting #4, Slide 30, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Dependence Matrix



Best Case Load Timing



Last Slide is only for "Best Case"

- 1. TLB, tag bank, and data bank all available
 - 3 independently contended/arbitrated resources
 - external CC lookup has priority use of tag bank
- 2. TLB and tag bank both have to hit
- 3. No load-after-store dependence
 - Probably requires <u>no older stores at all</u> to succeed; no time to check address queue after TLB lookup
- Extra Note: no store-to-load forwarding in R10K
 - quite hard (e.g., how to know store data is ready?)
 - little value added for large GPR ISAs (whereas x86 required it due to frequent register spills)

OOO and Memory Consistency

SC doesn't come freely in OOO

- Threads T1 and T2 communicate via shared memory locations X and Y
 - T1 produces result in X to be consumed by T2
 - T1 signals readiness to T2 by setting Y



 This works because SC says T1 and T2 must see the stores to X and Y in the same order

[Internal Use Draft] Superscalar Club Meeting #4, Slide 35, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

OOO and Consistency

- **T1:** inorder store commit by core no guarantee
 - reorder by cache (e.g., non-blocking miss handling)
 - non-atomic write (e.g., write-through cache, or MSI that allow M and S to overlap transiently)
- T2: speculative OOO allows reorder across datadependent control dependence (whether load X happens depends on value of load Y)



[Internal Use Draft] Superscalar Club Meeting #4, Slide 36, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

Aside: Write Atomicity



Q: when can writer's cache promote $S \rightarrow M$ after issuing invalidate? **A:** if WC, go for it; if SC, <u>strictly</u> after all $S \rightarrow I$ (how to know?).

Weak Consistency (WC)

- WC only imposes uniprocessor memory dependence: R_i(x)<W_j(x); W_i(x)<R_j(x); W_i(x)<W_j(x) All optimizations apply on sequential parts of prgm
- Program insert explicit memory fence instructions to force ordering when it matters



 Implementation wise, heavy-weight fence stalls subsequent LW/SW progress until full ordering

[Internal Use Draft] Superscalar Club Meeting #4, Slide 38, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

It can get weird: "Ghost Loads"

• Assume X and Y initially 0

T1 :	T2:
v1=load X	v2=load Y
store (Y, v1)	store (X, 3)

What are allowed values for v2 under WC?
 Only values on the table are 0 and 3, can it be 3?

WC says T2's load can see the value 3 introduced by its own future store!!

[Internal Use Draft] Superscalar Club Meeting #4, Slide 39, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

R10K Speculative SC

- Stores commit in-order when retired, but also need
 - cache block in M ready and locked at SW retirement
 - full invalidation ack in acquiring M (write atomicity)
- Loads attempted speculatively and out-of-order
 - address of invalidated cache blocks checked against speculative loads in load/store queue
 - if a loaded cache-block invalidated before load retires,
 "soft exception" restart from oldest affected load
 - load retires only if the fetched value is still "current"
- ⇒ Loads and stores appears executed in program order at the commit point
- \Rightarrow No penalty for sequential programs!!!

[Internal Use Draft] Superscalar Club Meeting #4, Slide 40, James C. Hoe, CMU/ECE/CALCM, ©2023 [Do not redistribute.]

TSO if just this