EE 660: Computer Architecture Out-of-Order Processors

Yao Zheng

Department of Electrical Engineering

University of Hawai‘i at Mānoa

UNIVERSITY of HAWAI‘I®
MĀNOA

Based on the slides of Prof. David Wentzlaff

Agenda

- I4 Processors
- I2O2 Processors
- I2OI Processors
- IO3 Processors
- IO2I Processors

Agenda

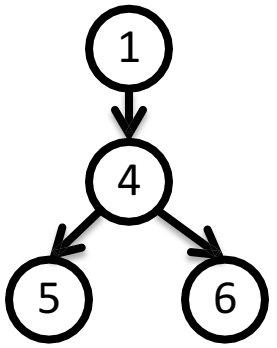
- I4 Processors
- I2O2 Processors
- I2OI Processors
- IO3 Processors
- IO2I Processors

Out-Of-Order (OOO) Introduction

Name	Frontend	Issue	Writeback	Commit	
I4	IO	IO	IO	IO	Fixed Length Pipelines Scoreboard
I2O2	IO	IO	OOO	OOO	Scoreboard
I2OI	IO	IO	OOO	IO	Scoreboard, Reorder Buffer, and Store Buffer
I03	IO	OOO	OOO	OOO	Scoreboard and Issue Queue
IO2I	IO	OOO	OOO	IO	Scoreboard, Issue Queue, Reorder Buffer, and Store Buffer

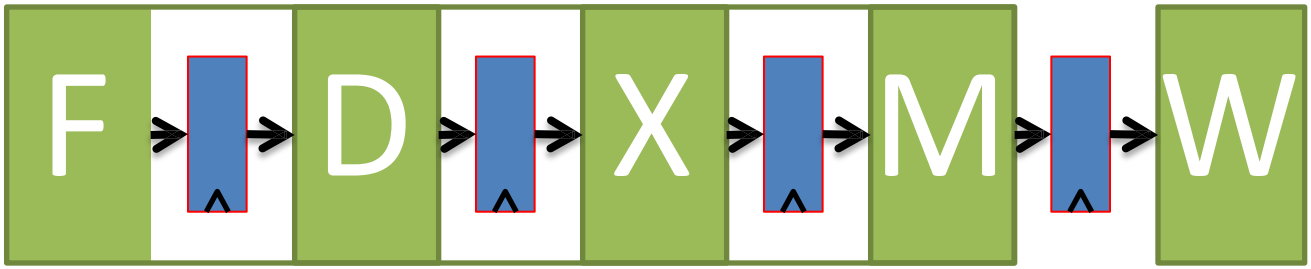
OOO Motivating Code Sequence

```
0 MUL   R1, R2, R3
1 ADDIU R11,R10,1
2 MUL   R5, R1, R4
3 MUL   R7, R5, R6
4 ADDIU R12,R11,1
5 ADDIU R13,R12,1
6 ADDIU R14,R12,2
```

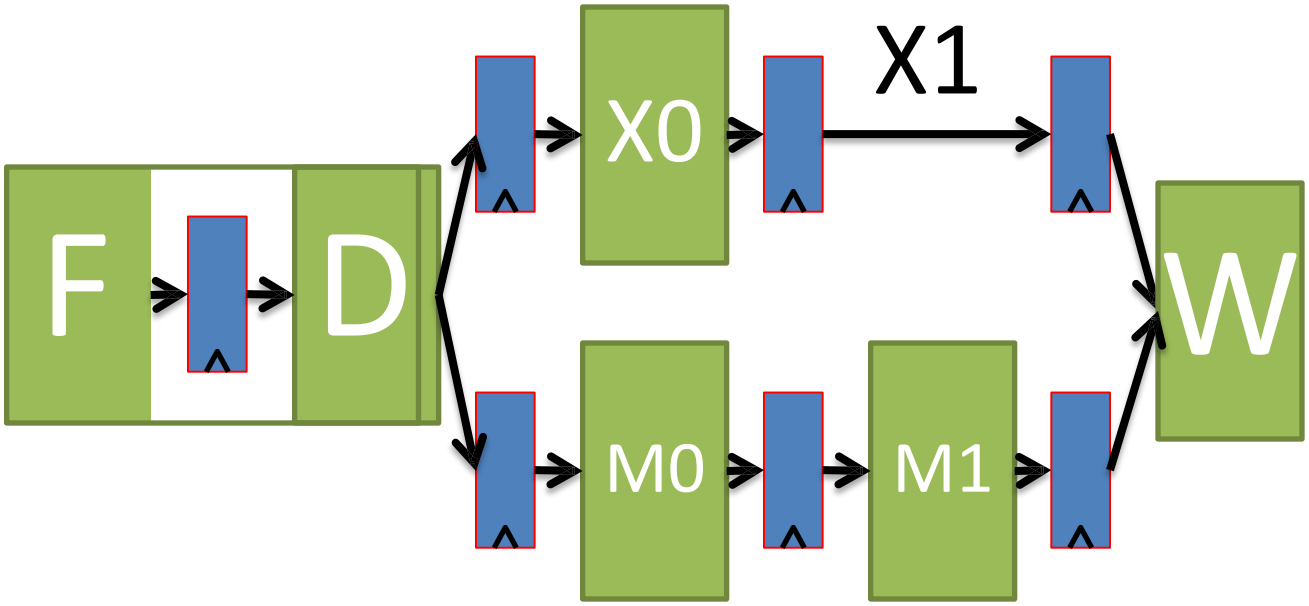


- Two independent sequences of instructions enable flexibility in terms of how instructions are scheduled in total order
- We can schedule statically in software or dynamically in hardware

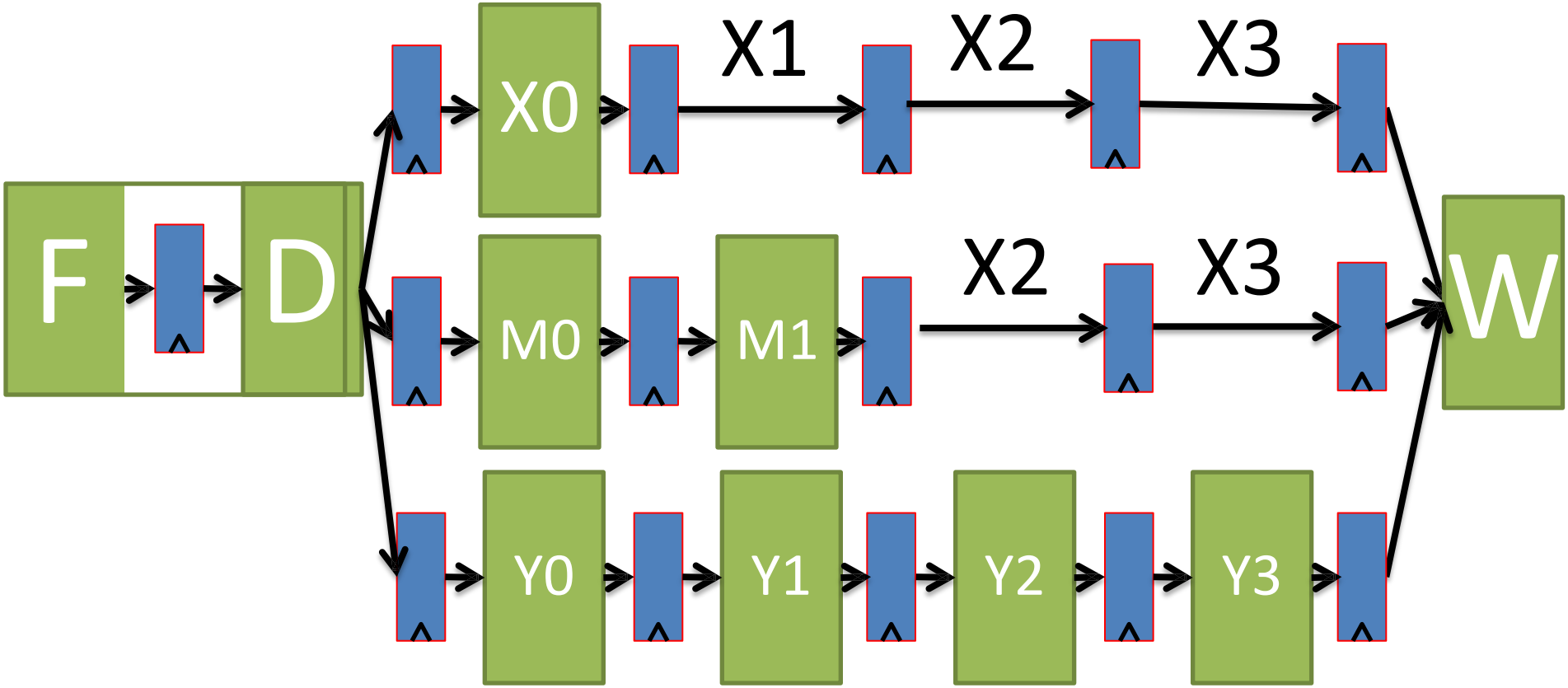
I4: In-Order Front-End, Issue, Writeback, Commit



I4: In-Order Front-End, Issue, Writeback, Commit

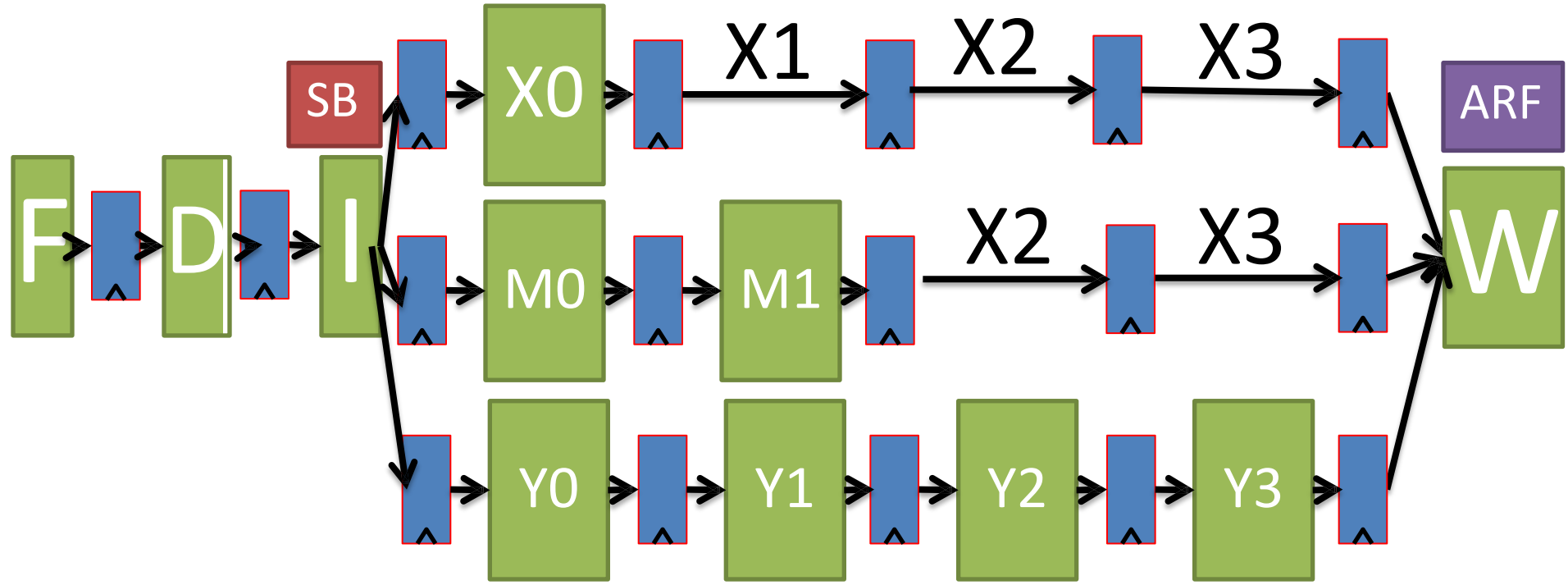


I4: In-Order Front-End, Issue, Writeback, Commit (4-stage MUL)



To avoid increasing CPI, needs full bypassing which can be expensive. To help cycle time, add Issue stage where register file read and instruction "issued" to Functional Unit

I4: In-Order Front-End, Issue, Writeback, Commit (4-stage MUL)



ARF	R	W
SB	R/W	W

Basic Scoreboard

Data Avail.

	P	F	4	3	2	1	0
R1							
R2							
R3							
…							
R31							

P: Pending, Write to Destination in flight
F: Which functional unit is writing register
Data Avail.: Where is the write data in the functional unit pipeline

- A One in Data Avail. In column ‘I’ means that result data is in stage ‘I’ of functional unit F
- Can use F and Data Avail. fields to determine when to bypass and where to bypass from
- A one in column zero means that cycle functional unit is in the Writeback stage
- Bits in Data Avail. field shift right every cycle.

Basic Scoreboard

Data Avail.

	P	F	4	3	2	1	0
R1			1				
R2							
R3							
…							
R31							

P: Pending, Write to Destination in flight
F: Which functional unit is writing register
Data Avail.: Where is the write data in the functional unit pipeline

- A One in Data Avail. In column ‘I’ means that result data is in stage ‘I’ of functional unit F
- Can use F and Data Avail. fields to determine when to bypass and where to bypass from
- A one in column zero means that cycle functional unit is in the Writeback stage
- Bits in Data Avail. field shift right every cycle.

```
0 MUL   R1, R2, R3 F  D  I  Y0 Y1 Y2 Y3 W
1 ADDIU R11,R10,1     F  D  I  X0 X1 X2 X3 W
2 MUL   R5, R1, R4       F  D  I  I  I  Y0 Y1 Y2 Y3 W
3 MUL   R7, R5, R6          F  D  D  D  I  I  I  I  Y0 Y1 Y2 Y3 W
4 ADDIU R12,R11,1              F  F  F  D  D  D  D  I  X0 X1 X2 X3 W
5 ADDIU R13,R12,1                    F  F  F  F  D  I  X0 X1 X2 X3 W
6 ADDIU R14,R12,2                             F  D  I  X0 X1 X2 X3 W
```

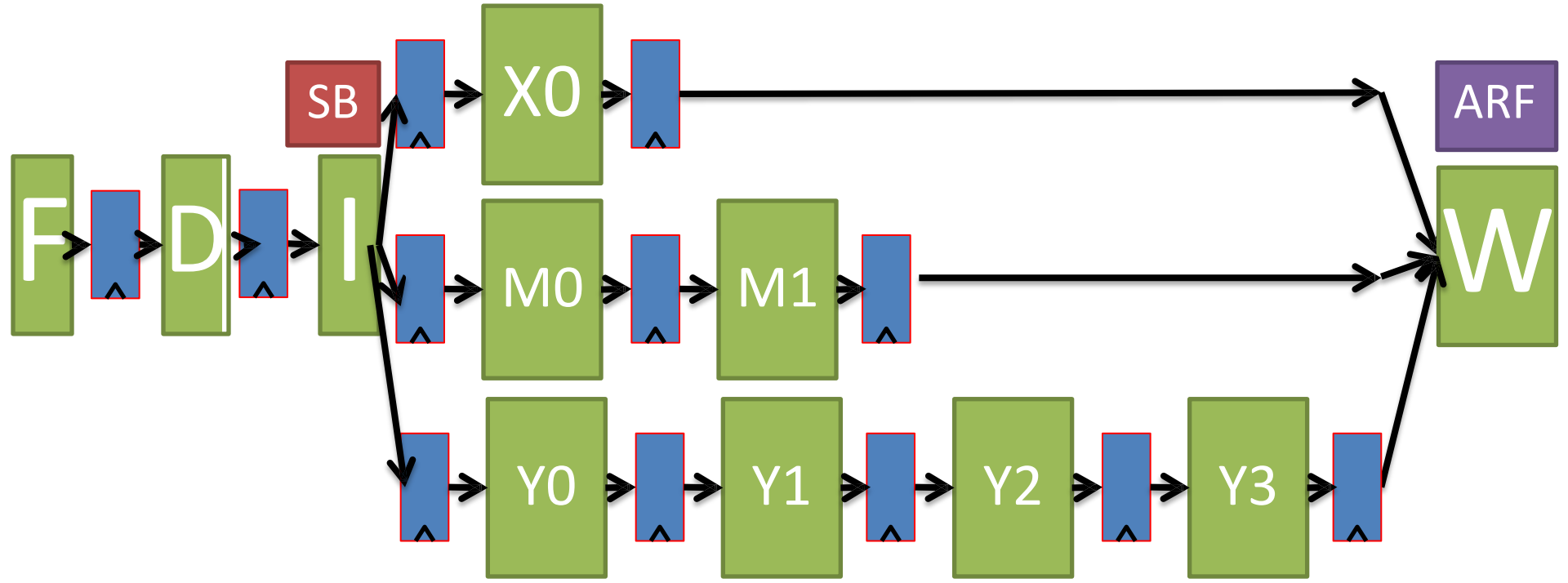
```
Cyc  D I          4 3 2 1 0      Dest Regs
1    0
2    1 0
3    2 1          1              R1
4                 1 1               R11
5                   1 1
6    3 2              1 1
7                 1      1 1            R5
8                   1      1
9                      1
10   4 3                  1
11   5 4          1          1      R7
12   6 5          1 1                   R12
13     6          1 1 1                    R13
14                1 1 1 1                     R14
15                  1 1 1 1
16                     1 1 1
17                       1 1
18                         1
```

RED Indicates if we look at F Field, we can bypass on this cycle

Agenda

- I4 Processors
- I2O2 Processors
- I2OI Processors
- IO3 Processors
- IO2I Processors

I2O2: In-order Frontend/Issue, Out-of-order Writeback/Commit

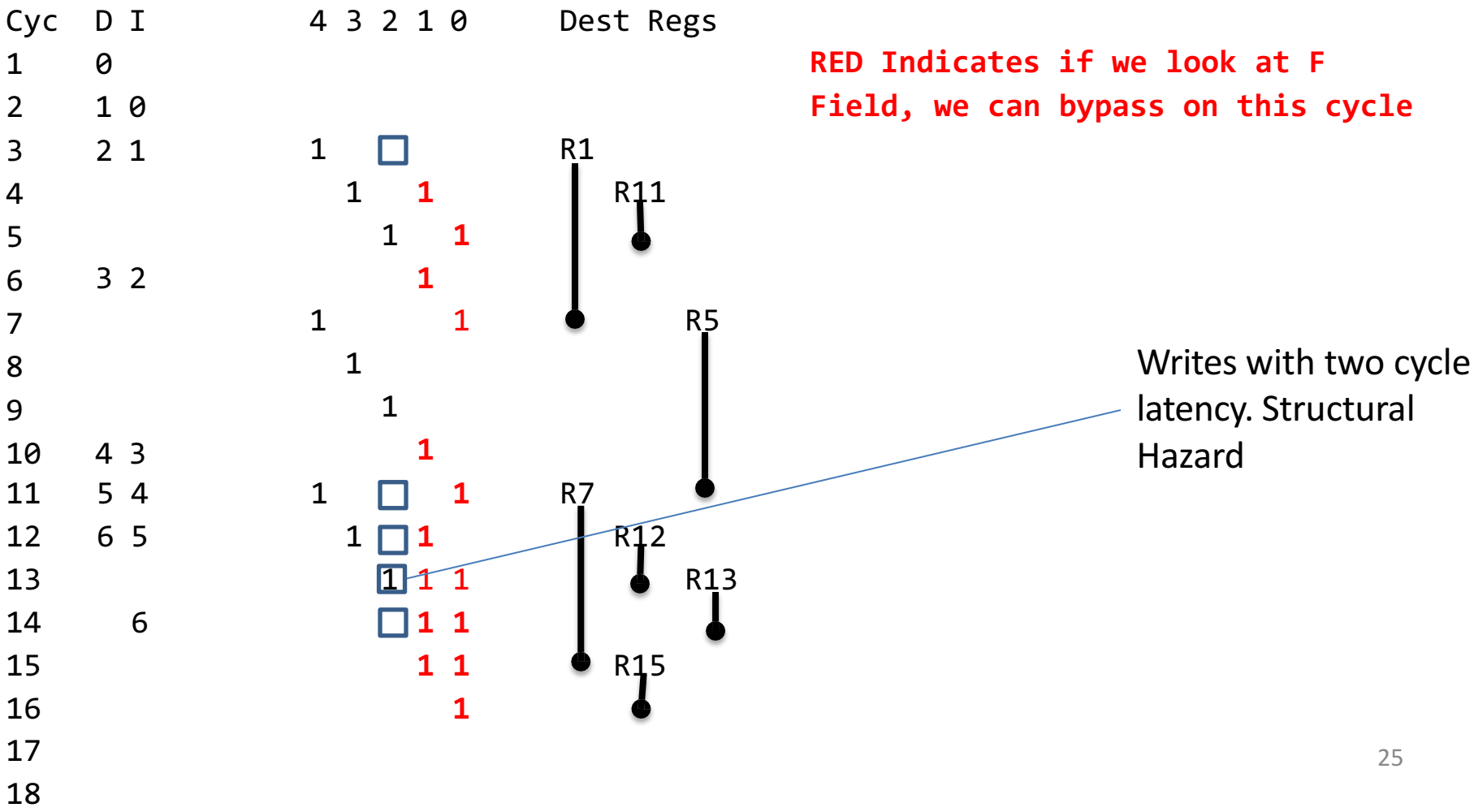


ARF	R		W
SB	R	R/W	W

I2O2 Scoreboard

- Similar to I4, but we can now use it to track structural hazards on Writeback port
- Set bit in Data Avail. according to length of pipeline
- Architecture conservatively stalls to avoid WAW hazards by stalling in Decode therefore current scoreboard sufficient. More complicated scoreboard needed for processing WAW Hazards

```
0 MUL   R1, R2, R3 F  D  I  Y0 Y1 Y2 Y3 W
1 ADDIU R11,R10,1     F  D  I  X0 W
2 MUL   R5, R1, R4       F  D  I  I  I  Y0 Y1 Y2 Y3 W
3 MUL   R7, R5, R6          F  D  D  D  I  I  I  I  Y0 Y1 Y2 Y3 W
4 ADDIU R12,R11,1              F  F  F  D  D  D  D  I  X0 W
5 ADDIU R13,R12,1                       F  F  F  F  D  I  X0 W
6 ADDIU R14,R12,2                                   F  D  I  I X0 W
```



Early Commit Point?

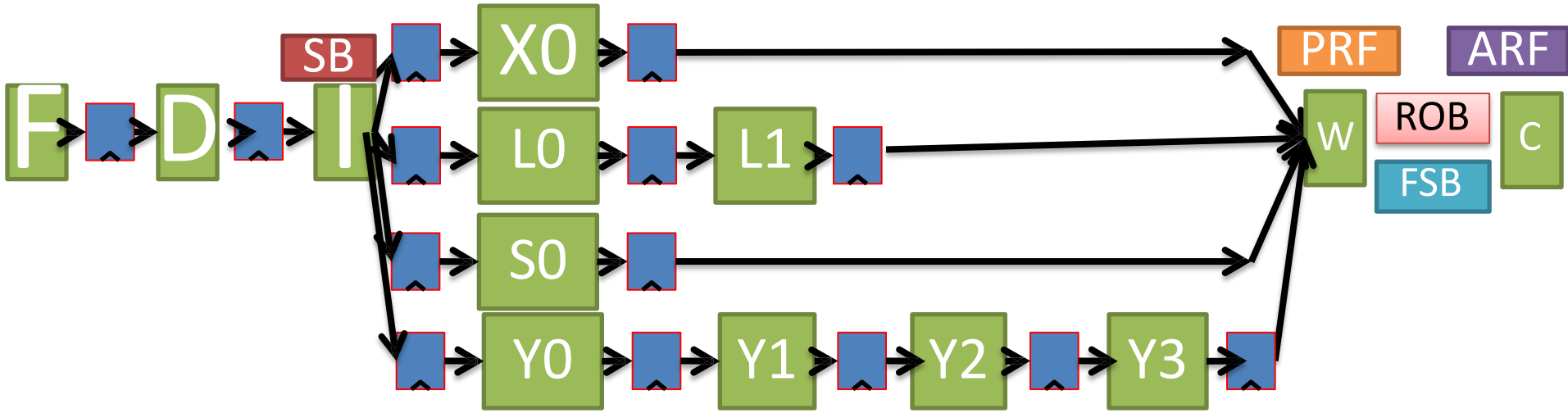
```
0 MUL   R1, R2, R3 F  D  I  Y0 Y1 Y2 Y3 /
1 ADDIU R11,R10,1     F  D  I  X0 W     /
2 MUL   R5, R1, R4       F  D  I  I  I  /
3 MUL   R7, R5, R6          F  D  D  D  /
4 ADDIU R12,R11,1              F  F  F  /
5 ADDIU R13,R12,1                       /
6 ADDIU R14,R12,2
```

- Limits certain types of exceptions.

Agenda

- I4 Processors
- I2O2 Processors
- I2OI Processors
- IO3 Processors
- IO2I Processors

I2OI: In-order Frontend/Issue, Out-of-order Writeback, In-order Commit



ARF				W
SB		R/W	W	
PRF		R	W	
ROB	R/W		W	R/W
FSB			W	R/W

PRF=Physical Register File(Future File), ROB=Reorder Buffer, FSB=Finished Store Buffer (1 entry)

Reorder Buffer (ROB)

State	S	ST	V	Preg
--				
P	1			
F	1			
P	1			
P				
F				
P				
P				
--				
--				

State: {Free, Pending, Finished}
S: Speculative
ST: Store bit
V: Physical Register File Specifier Valid
Preg: Physical Register File Specifier

Reorder Buffer (ROB)

State	S	ST	V	Preg
--				
P	1			
F	1			
P	1			
P				
F				
P				
P				
--				
--				

Next instruction allocates here in D

Tail of ROB

Speculative because branch is in flight

Instruction wrote ROB out of order

Head of ROB

Commit stage is waiting for Head of ROB to be finished

State: {Free, Pending, Finished}
S: Speculative
ST: Store bit
V: Physical Register File Specifier Valid
Preg: Physical Register File Specifier

Finished Store Buffer (FSB)

V	Op	Addr	Data
--			

- Only need one entry if we only support one memory instruction inflight at a time.
- Single Entry FSB makes allocation trivial.
- If support more than one memory instruction, we need to worry about Load/Store address aliasing.

```
0 MUL   R1, R2, R3 F  D  I  Y0 Y1 Y2 Y3 W  C
1 ADDIU R11,R10,1     F  D  I  X0 W  r        C
2 MUL   R5, R1, R4       F  D  I  I  I  Y0 Y1 Y2 Y3 W  C
3 MUL   R7, R5, R6          F  D  D  D  I  I  I  I  Y0 Y1 Y2 Y3 W  C
4 ADDIU R12,R11,1              F  F  F  D  D  D  D  I  X0 W  r        C
5 ADDIU R13,R12,1                       F  F  F  F  D  I  X0 W  r        C
6 ADDIU R14,R12,2                                   F  D  I  I  X0 W  r     C
```

```
Cyc  D I    ROB  0    1   2   3
0
1    0
2    1 0          R1
3    2 1               R11
4                           R5
5
6    3 2               R11
7                                R7
8               R1
9
10   4 3
11   5 4          R12
12   6 5               R13  R5
13                          R14
14     6          R12
15               R13
16                               R7
17                          R14
18
19
```

Empty = free entry in ROB

State of ROB at beginning of cycle

Pending entry in ROB

Circle=Finished (Cycle after W)

Last cycle before entry is freed from ROB (Cycle in C stage)

Entry becomes free and is freed on next cycle

What if First Instruction Causes an Exception?

```
0 MUL   R1, R2, R3 F  D  I  Y0 Y1 Y2 Y3 W  /
1 ADDIU R11,R10,1     F  D  I  X0 W  r  -- /
2 MUL   R5, R1, R4       F  D  I  I  I  Y0 /
3 MUL   R7, R5, R6          F  D  D  D  I  /
4 ADDIU R12,R11,1              F  F  F  D  /
                                        F  D  I. . .
```

What About Branches?

Option 2

```
0 BEQZ  R1, target F  D  I  X0 W  C
1 ADDIU R11,R10,1     F  D  I  X0 /
2 ADDIU R5, R1, R4       F  D  I  /
3 ADDIU R7, R5, R6          F  D  /
T ADDIU R12,R11,1              F  D  I . . .
```

Squash instructions in ROB when Branch commits

Option 1

```
0 BEQZ  R1, target F  D  I  X0 W  C
1 ADDIU R11,R10,1     F  D  I  -
2 ADDIU R5, R1, R4       F  D  -
3 ADDIU R7, R5, R6          F  -
T ADDIU R12,R11,1              F  D  I . . .
```

Squash instructions earlier. Has more complexity. ROB needs many ports.

Option 3

```
0 BEQZ  R1, target F  D  I  X0 W  C
1 ADDIU R11,R10,1     F  D  I  X0 W  /
2ADDIU R5, R1, R4        F  D  I  X0 W  /
3ADDIU R7, R5, R6           F  D  I  X0 W  /
T ADDIU R12,R11,1              F  D  I  X0 W  C
```

Wait for speculative instructions to reach the Commit stage and squash in Commit stage

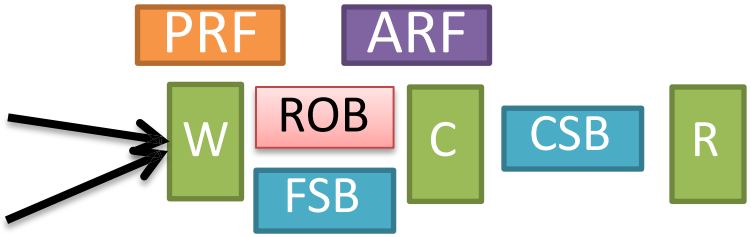
What About Branches?

- Three possible designs with decreasing complexity based on when to squash speculative instructions and de-allocate ROB entry:

1. As soon as branch resolves
2. When branch commits
3. When speculative instructions reach commit

- Base design only allows one branch at a time. Second branch stalls in decode. Can add more bits to track multiple in-flight branches.

Avoiding Stalling Commit on Store Miss



CSB=Committed Store Buffer

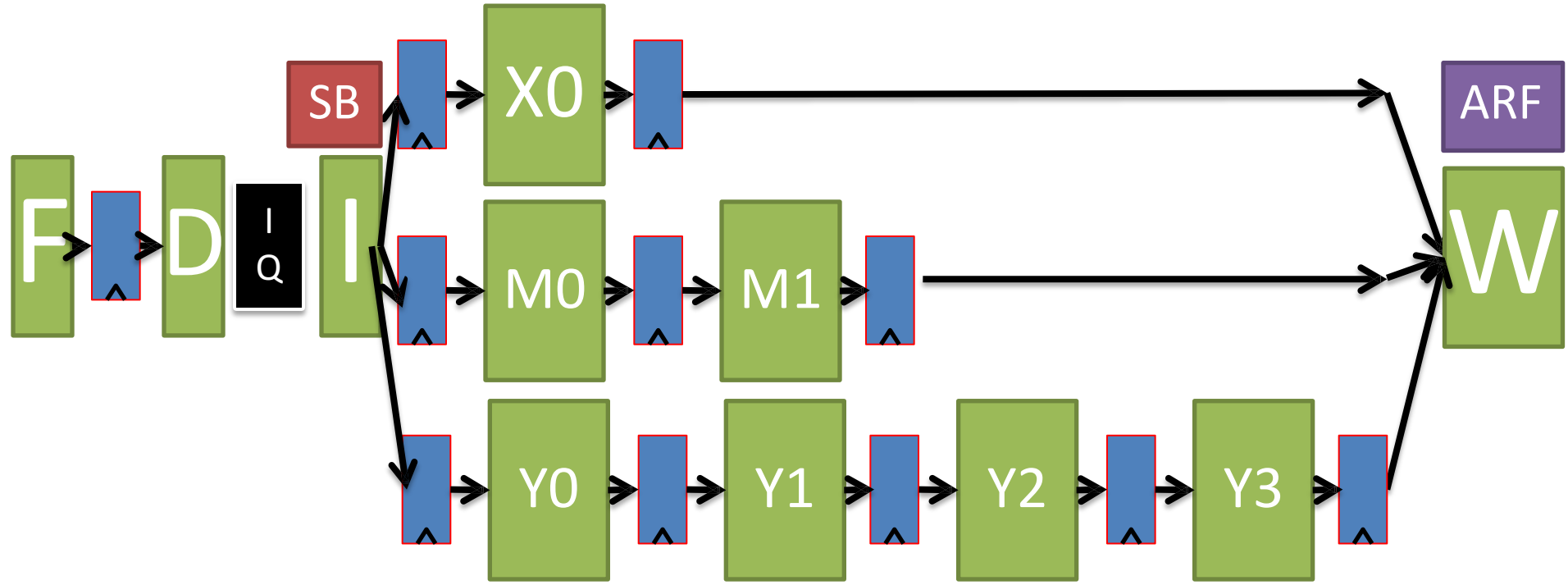
```
0 OpA    F   D   I   X0 W   C
1 SW         F   D   I   S0 W   C   C   C   C
2 OpB            F   D   I   X0 W   W   W   W   C
3 OpC                F   D   I   X   X   X   X   W   C
4 OpD                    F   D   I   I   I   I   X   W   C

With Retire Stage
0 OpA    F   D   I   X0 W   C
1 SW         F   D   I   S0 W   C   R   R   R
2 OpB            F   D   I   X0 W   C
3 OpC                F   D   I   X   W   C
4 OpD                    F   D   I   X   W   C
```

Agenda

- I4 Processors
- I2O2 Processors
- I2OI Processors
- IO3 Processors
- IO2I Processors

IO3: In-order Frontend, Out-of-order Issue/Writeback/Commit



ARF	R	W
SB	R R/W	W
IQ	W R/W	W

Issue Queue (IQ)

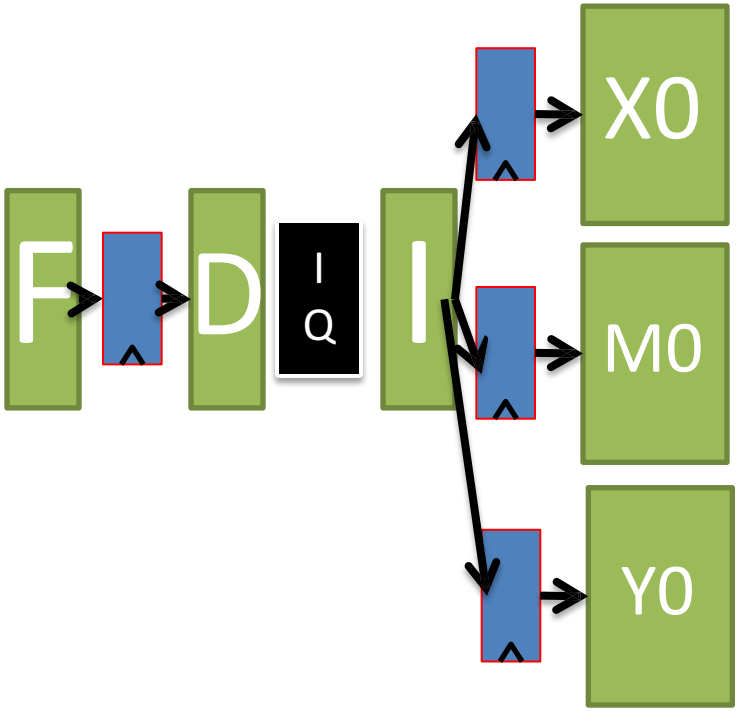
Op	Imm	S	V	Dest	V	P	Src0	V	P	Src1

Op: Opcode
Imm.: Immediate
S: Speculative Bit
V: Valid (Instruction has corresponding Src/Dest)
P: Pending (Waiting on operands to be produced)

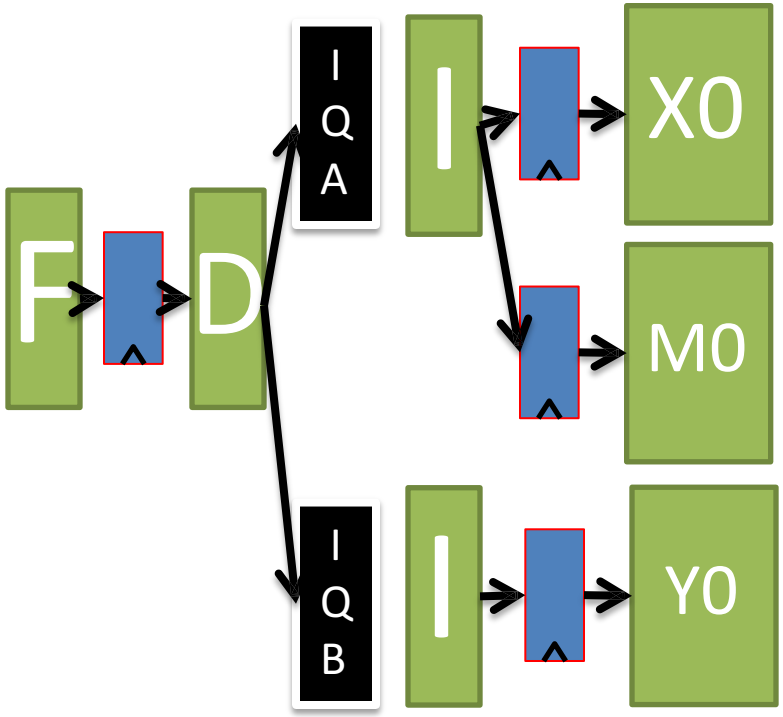
Instruction Ready = (!Vsrc0 || !Psrc0) && (!Vsrc1 || !Psrc1) && no structural hazards

- For high performance, factor in bypassing

Centralized vs. Distributed Issue Queue



Centralized



Distributed

Advanced Scoreboard

Data Avail.

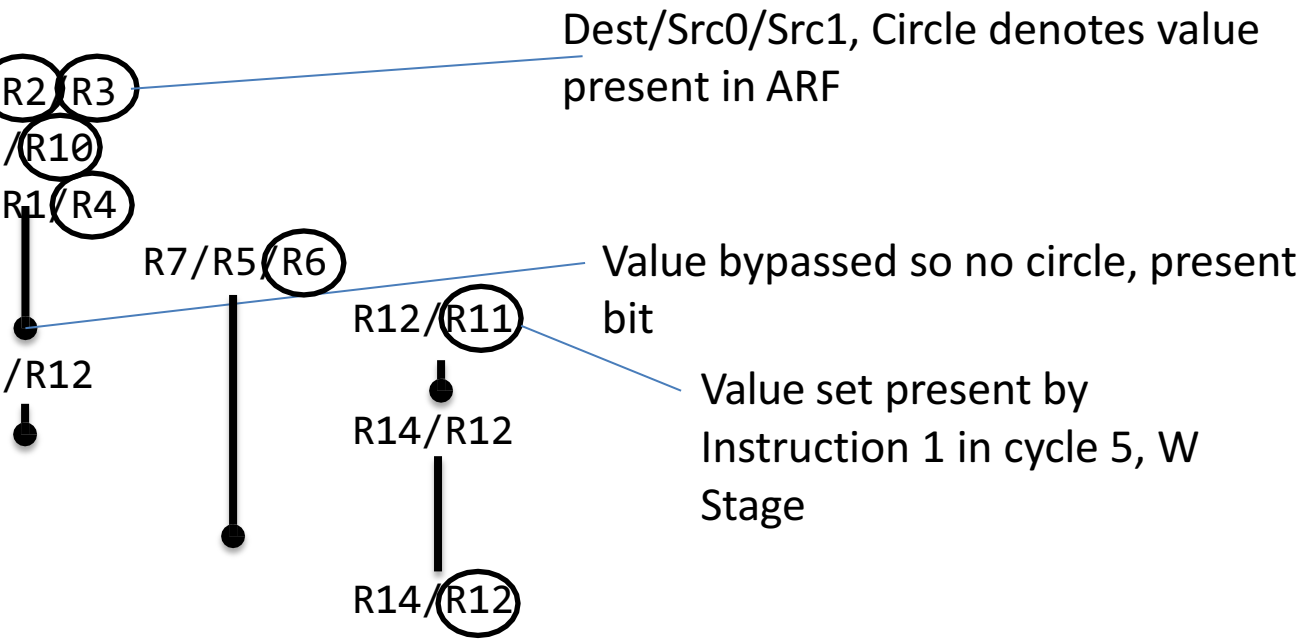
	P	4	3	2	1	0
R1						
R2						
R3						
…						
R31						

P: Pending, Write to Destination in flight
Data Avail.: Where is the write data in the pipeline and which functional unit

- Data Avail. now contains functional unit identfier
- A non-empty value in column zero means that cycle functional unit is in the Writeback stage
- Bits in Data Avail. field shift right every cycle.

```
                    0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
0 MUL   R1, R2, R3  F  D  I  Y0 Y1 Y2 Y3 W
1 ADDIU R11,R10,1      F  D  I  X0 W
2 MUL   R5, R1, R4        F  D  i     I  Y0 Y1 Y2 Y3 W
3 MUL   R7, R5, R6           F  D  i              I  Y0 Y1 Y2 Y3 W
4 ADDIU R12,R11,1               F  D  i  I  X0 W
5 ADDIU R13,R12,1                  F  D  i  I  X0 W
6 ADDIU R14,R12,2                  F  D  i        I  X0 W
```

```
Cyc  D I    IQ   0        1         2
0
1    0
2    1 0          R1/R2/R3
3    2 1          R11/R10
4    3            R5/R1/R4
5    4                      R7/R5/R6
6    5 2                                R12/R11
7    6 4          R13/R12
8      5                                R14/R12
9
10     3
11     6                                R14/R12
12
13
14
```



Assume All Instruction in Issue Queue

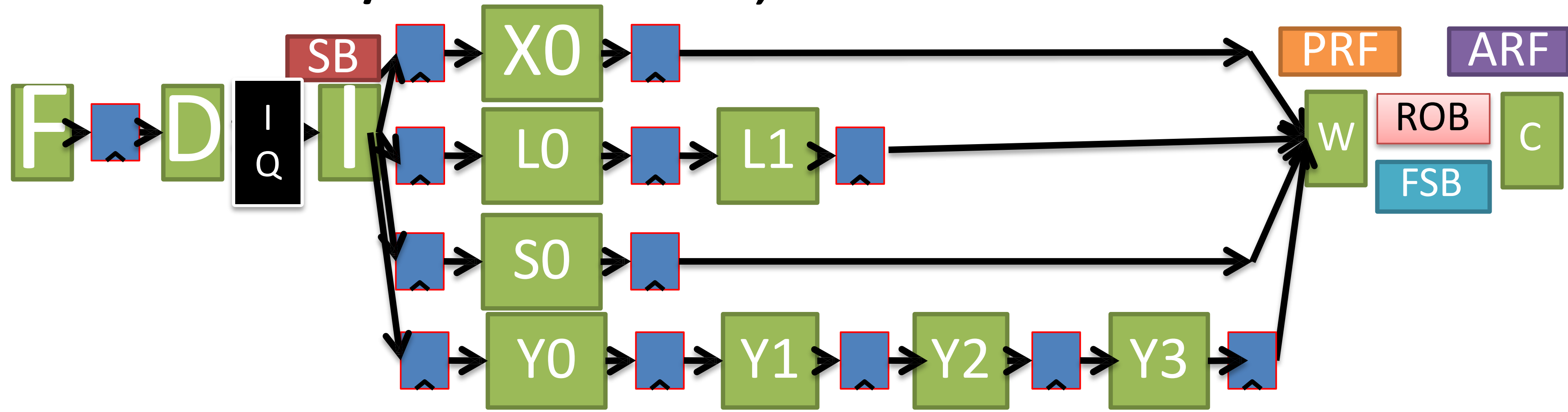
```
                                  0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
0 MUL   R1, R2, R3 F  D  i                  I  Y0 Y1 Y2 Y3 W
1 ADDIU R11,R10,1     F  D  i                  I  X0 W
2 MUL   R5, R1, R4       F  D  i                        I  Y0 Y1 Y2 Y3 W
3 MUL   R7, R5, R6          F  D  i                           I  Y0 Y1 Y2 Y3 W
4 ADDIU R12,R11,1              F  D  i               I  X0 W
5 ADDIU R13,R12,1                 F  D  i                  I  X0 W
6 ADDIU R14,R12,2                    F  D  i                  I  X0 W
```

- Better performance than previous?

Agenda

- I4 Processors
- I2O2 Processors
- I2OI Processors
- IO3 Processors
- IO2I Processors

IO2I: In-order Frontend, Out-of-order Issue/Writeback, In-order Commit



ARF				W
SB		R/W	W	
PRF		R	W	
ROB	R/W		W	R/W
FSB			W	R/W
IQ	W	R/W		

```
                    0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
0 MUL   R1, R2, R3  F  D  I  Y0 Y1 Y2 Y3 W  C
1 ADDIU R11,R10,1      F  D  I  X0 W  r        C
2 MUL   R5, R1, R4        F  D  i     I  Y0 Y1 Y2 Y3 W  C
3 MUL   R7, R5, R6           F  D  i              I  Y0 Y1 Y2 Y3 W  C
4 ADDIU R12,R11,1               F  D  i  I  X0 W  r                    C
5 ADDIU R13,R12,1                  F  D  i  I  X0 W  r                    C
6 ADDIU R14,R12,2                     F  D  i        I  X0 W  r                 C

0 MUL   R1, R2, R3  F  D  I  Y0 Y1 Y2 Y3 W  C
1 ADDIU R11,R10,1      F  D  I  X0 W  r        C
2 MUL   R5, R1, R4        F  D  i     I  Y0 Y1 Y2 Y3 W  C
3 MUL   R7, R5, R6           F  D  i              I  Y0 Y1 Y2 Y3 W  C
4 ADDIU R12,R11,1               F  D  i  I  X0 W  r                    C
5 ADDIU R13,R12,1                  F  D  i  I  X0 W  r                    C
6 ADDIU R14,R12,2                     F  D  i     I  X0 W  r                 C
```

Out-of-order 2-Wide Superscalar with 1 ALU

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	MUL R1, R2, R3	F	D	I	Y0	Y1	Y2	Y3	W	C											
1	ADDIU R11,R10,1	F	D	I	X0	W	r				C										
2	MUL R5, R1, R4		F	D	i			I	Y0	Y1	Y2	Y3	W	C							
3	MUL R7, R5, R6		F	D	i							I	Y0	Y1	Y2	Y3	W	C			
4	ADDIU R12,R11,1			F	D	I	X0	W	r										C		
5	ADDIU R13,R12,1			F	D	i	I	X0	W	r										C	
6	ADDIU R14,R12,2				F	D	i	I	X0	W	r										C

Acknowledgements

- These slides contain material developed and copyright by:
  - Arvind (MIT)
  - Krste Asanovic (MIT/UCB)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)
  - Christopher Batten (Cornell)

- MIT material derived from course 6.823
- UCB material derived from course CS252 & CS152
- Cornell material derived from course ECE 4750