LUND UNIVERSITY

EITF20: Computer Architecture

Part3.2.1: Pipeline - 3

Liang Liu
liang.liu@eit.lth.se

Outline

- Dynamic scheduling –Tomasulo
- Exception
- Speculation
- What we have done so far (pipeline)

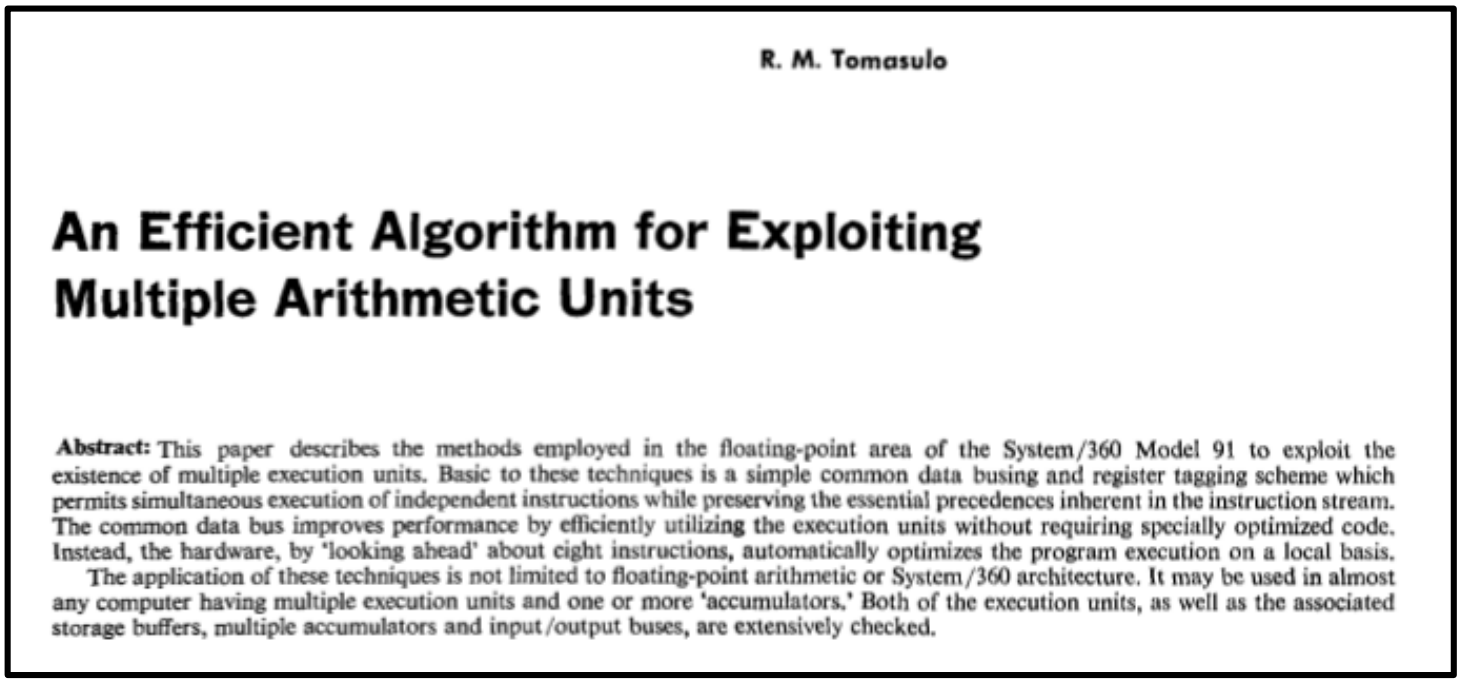
Outline

- Reiteration
- **Dynamic scheduling - Tomasulo**
- Speculation
- ILP limitations
- What we have done so far

Tomasulo algorithm

Another dynamic instruction scheduling algorithm

- **For IBM 360/91, a few years after the CDC 6600 (Scoreboard)**
- **Goal: High performance without compiler support**



R. M. Tomasulo

An Efficient Algorithm for Exploiting Multiple Arithmetic Units

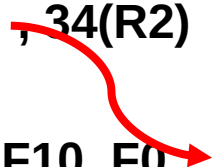
Abstract: This paper describes the methods employed in the floating-point area of the System/360 Model 91 to exploit the existence of multiple execution units. Basic to these techniques is a simple common data busing and register tagging scheme which permits simultaneous execution of independent instructions while preserving the essential precedences inherent in the instruction stream. The common data bus improves performance by efficiently utilizing the execution units without requiring specially optimized code. Instead, the hardware, by 'looking ahead' about eight instructions, automatically optimizes the program execution on a local basis.

The application of these techniques is not limited to floating-point arithmetic or System/360 architecture. It may be used in almost any computer having multiple execution units and one or more 'accumulators.' Both of the execution units, as well as the associated storage buffers, multiple accumulators and input/output buses, are extensively checked.

Registr renaming

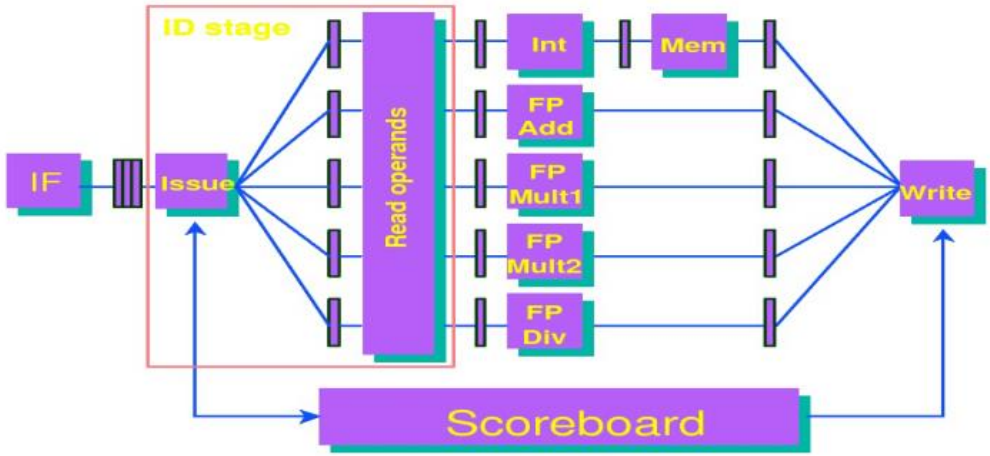
Example:

LD	F6, 34(R2)	LD	FT, 34(R2)	LD	, 34(R2)		
...	...	...		...			
DIVD	F10,F0,F6	DIVD	F10, F0, FT	DIVD	F10, F0,		
ADDD	F6,F8,F2	ADDD	F6, F8, F2	ADDD	F6, F8, F2		



- **Potential WAR harzard on F6**
  - If ADDD finishes before DIVD starts
- **Register renaming**
  - Another temperoray register (FT) instead of F6
  - Or make a link and forward

Scoreboard pipeline



- **Issue:** An instruction is issued if (**in order**):
  - The needed functional unit is free (there is no **structural hazard**)
  - No functional unit has a destination operand equal to the destination of the instruction (resolves **WAW hazards**)
- **Read:** Wait until no data hazards, then read operands
  - Performed in parallel for all functional units
  - Resolves **RAW hazards** dynamically
- **EX:** Normal execution (**out of order**)
  - Notify the scoreboard when ready
- **Write:** The instruction can update destination if:
  - All earlier instructions have read their operands (resolves **WAR hazards**)

Scoreboard example

Instruction status

Instruction		j	k	Issue	Read ops	Exec. compl.	Write result
LD	F6	34+	R2				
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	src 1 Fj	src 2 Fk	FUsrc1 Qj	FUsrc2 Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU								

Clock: 0

Tomasulo orgnizations (renaming with RS)

LD F6, 34(R2)

...

DIVD F10, F0, F6

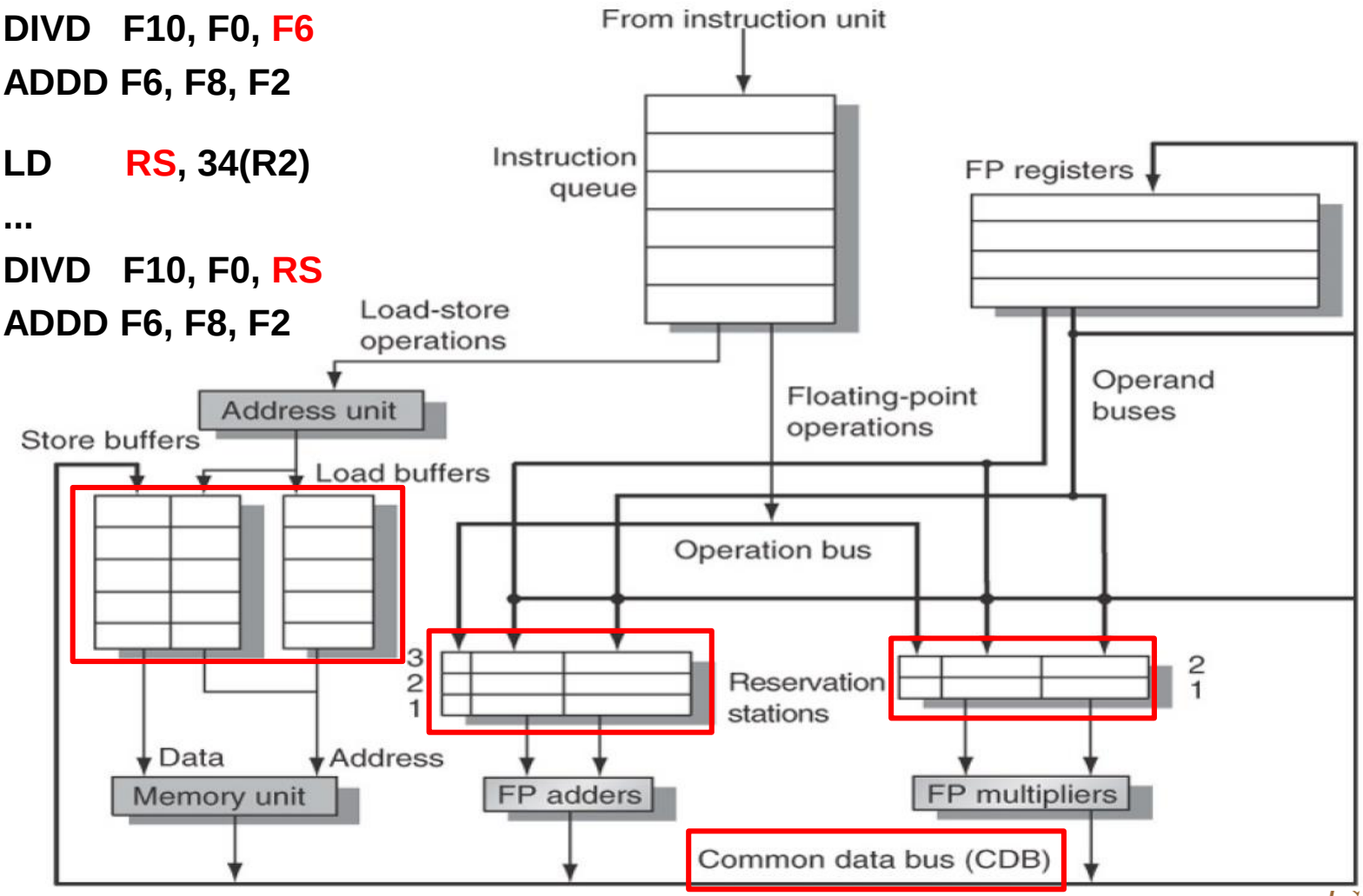
ADDD F6, F8, F2

LD RS, 34(R2)

...

DIVD F10, F0, RS

ADDD F6, F8, F2



Reservation stations

- **Op:Operation to perform (e.g., + or –)**
- **Vj, Vk: Value (instead of reg specifier) of Source operands**
- **Qj, Qk: Reservation stations (instead of FU) producing source registers (value to be written)**
  - Note: Qj,Qk=0 => ready
  - V and Q filed are **mutual exclusive**
- **Busy: Indicates reservation station or FU is busy**

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Three stages of Tomasulo algorithm

- **Issue –** get instruction from instruction Queue
  - If matching reservation station free (no **structural hazard**)
  - Instruction is issued together with its operand's values or RS point (**register rename**, handle **WAR, WAW**)
- **Execution –** operate on operands (EX)
  - When both operands are ready, then execute (handle **RAW**)
  - If not ready, watch **Common Data Bus (CDB)** for operands (snooping)
- **Write result –** finish execution (WB)
  - Write on CDB to all awaiting RS, regs (**forwarding**)
  - Mark reservation station available
  - Data Bus
    - Normal Bus: data + destination
    - Common Data Bus: data + source (snooping)

Tomasulo example, cycle 0

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2			
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Load buffers

	Busy	Address
Load1	No	
Load2	No	
Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status :Indicates which RS will write each register

	F0	F2	F4	F6	F8	F10	...	F30
FU								

Clock: 0

Tomasulo example, cycle 1

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1		
LD	F2	45+	R3			
MULTD	F0	F2	F4			
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Load buffers

Time		Busy	Address
	Load1	Yes	R2+34
	Load2	No	
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU				Load1				

Clock: 1

Tomasulo example, cycle 3

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	
LD	F2	45+	R3	2		
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2			
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Load buffers

Time		Busy	Address
0	Load1	Yes	R2+32
1	Load2	Yes	R3+45
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	Mult		F4	Load2	
	Mult2	No					

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU	Mult1	Load2		Load1				

Clock: 3

Note:

1. Can have multiple loads
2. Registers names are removed ("renamed") in Reservation Stations

Tomasulo example, cycle 4

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6			
ADDD	F6	F8	F2			

Load buffers

Time		Busy	Address
	Load1	No	
0	Load2	Yes	R3+45
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	Yes	Sub	M(R2+34)			Load2
	Add2	No					
	Add3	No					
	Mult1	Yes	Mult		F4	Load2	
	Mult2	No					

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU	Mult1	Load2		-	Add1			

Clock: 4

Note:

1. CDB forwarding LD1 result to SUBD

Tomasulo example, cycle 5

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4		
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2			

Load buffers

Time		Busy	Address
	Load1	No	
	Load2	No	
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
2	Add1	Yes	Sub	M(R2+34)	M(R3+45)		-
	Add2	No					
	Add3	No					
10	Mult1	Yes	Mult	M(R3+45)	F4	-	
	Mult2	Yes	Div		F6	Mult1	

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU	Mult1	-			Add1	Mult2		

Clock: 5

Tomasulo example, cycle 7

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6		

Load buffers

Time		Busy	Address
	Load1	No	
	Load2	No	
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
0	Add1	Yes	Sub	M(R2+34)	M(R3+45)		
	Add2	Yes	Add		F2	Add1	
	Add3	No					
8	Mult1	Yes	Mult	M(R3+45)	F4		
	Mult2	Yes	Div		F6	Mult1	

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU	Mult1			Add2	Add1	Mult2		

Clock: 7

Tomasulo example, cycle 10

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	

Load buffers

Time		Busy	Address
	Load1	No	
	Load2	No	
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
0	Add2	Yes	Add	F6-F2	F2		
	Add3	No					
5	Mult1	Yes	Mult	M(R3+45)	F4		
	Mult2	Yes	Div		F6	Mult1	

Can we write the result of ADDD?

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU	Mult1			Add2		Mult2		

Clock: 10

Elimation of WAR hazard

Example:

```
        LD      F6, 34(R2)
        ...     ...
        DIVD    F10,F0,F6
        ADDD    F6,F8,F2
```

- **ADDD can safely finish before DIVD has read register F6 because:**
  - DIVD has renamed register F6 to the reservation station

Tomasulo example, cycle 11

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3		
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

Load buffers

Time		Busy	Address
	Load1	No	
	Load2	No	
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
4	Mult1	Yes	Mult	M(R3+45)	F4		
	Mult2	Yes	Div		F6	Mult1	

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU	Mult1			-		Mult2		

Clock: 11

Tomasulo example, cycle 16

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5		
ADDD	F6	F8	F2	6	10	11

Load buffers

Time		Busy	Address
	Load1	No	
	Load2	No	
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
40	Mult2	Yes	Div	F0	F6	-	

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU	-					Mult2		

Clock: 16

Tomasulo example, cycle 57

Instruction status

Instruction		j	k	Issue	Exec. compl.	Write result
LD	F6	34+	R2	1	3	4
LD	F2	45+	R3	2	4	5
MULTD	F0	F2	F4	3	15	16
SUBD	F8	F6	F2	4	7	8
DIVD	F10	F0	F6	5	56	57
ADDD	F6	F8	F2	6	10	11

Load buffers

Time		Busy	Address
	Load1	No	
	Load2	No	
	Load3	No	

Functional unit status

Time	Name	Busy	Op	src 1 Vj	src 2 Vk	RS for j Qj	RS for k Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU						-		

Clock: 57

Comparing: Scoreboard example, CP62

Instruction status

Instruction		j	k	Issue	Read ops	Exec. compl.	Write result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	dest Fi	src 1 Fj	src 2 Fk	FUsrc1 Qj	FUsrc2 Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	...	F30
FU						-		

Clock: 62

Tomasulo vs Scoreboard

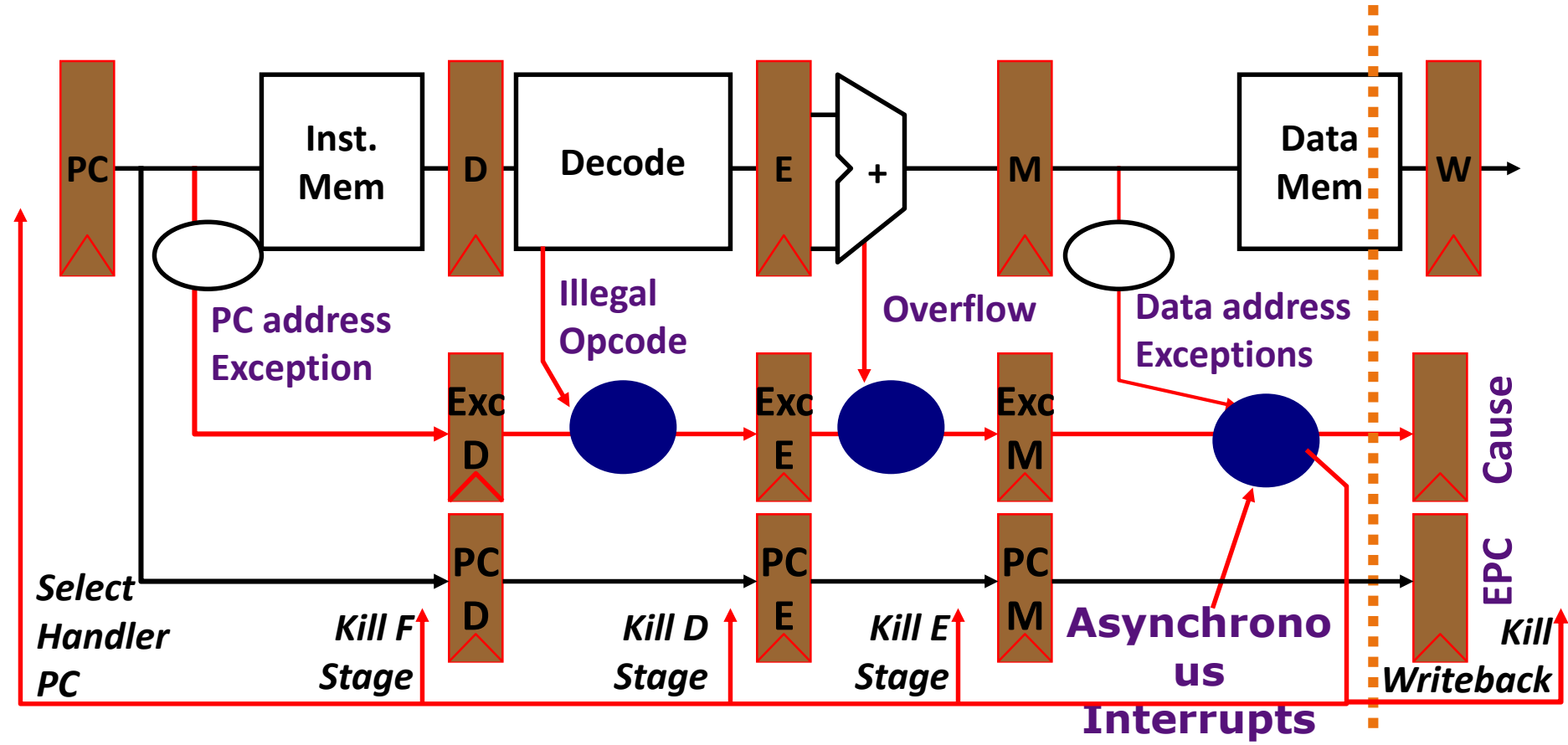
Differences between Tomasulo Algorithm and Scoreboard

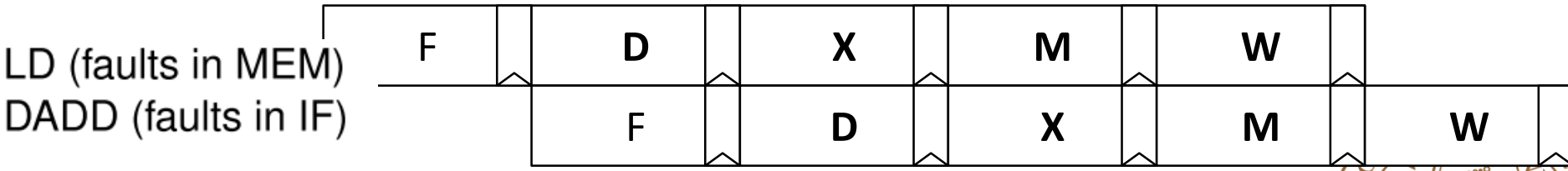
- **Control and buffers distributed with Function Units versus centralized in scoreboard**
- **Registers in instructions replaced by pointers to reservation stations**
  - Register renaming, helps avoid WAR and WAW hazards
  - More reservation stations than registers; so allow optzns compilers can't do
  - Operands stays in register in Scoreboard (stall for WAR and WAW)
- **Common Data Bus broadcasts results to all FUs (forwarding!)**

Outline

- Reiteration
- Dynamic scheduling - Tomasulo
- Superscalar, VLIW
- **Exception**
- Speculation
- What we have done so far

Solution for simple MIPS





How to handle exception in OOO ExE?

Outline

- Reiteration
- Dynamic scheduling - Tomasulo
- Superscalar, VLIW
- **Speculation**
- ILP limitations
- What we have done so far

Hardware-base speculation

- **Trying to exploit more ILP (e.g., multiple issue) while maintaining control dependencies becomes a burden**

```
DIVD   F10, F0, F6
if (F10=0)
MULT  F2,  F4, F5
else
ADD    F2,  F4, F5
```

- **Overcome control dependencies**
  - By speculating on the outcome of branches and **executing** the program as if our guesses were correct
  - Need to handle incorrect guesses
- **Methodologies (combine):**
  - **Dynamic branch prediction**: allows instruction scheduling across branches (choose which instr. to execute)
  - **Speculation**: **execute** instructions before all control dependencies are resolved
  - **Dynamic scheduling**: take advantage of ILP (scheduling speculated instr.)

Implementing speculation

Key idea

- Allow instructions to execute **out of order**
- Force instructions to **commit in order**
- Prevent any **irrevocable action** (such as updating state or taking an exception) until an instruction commits

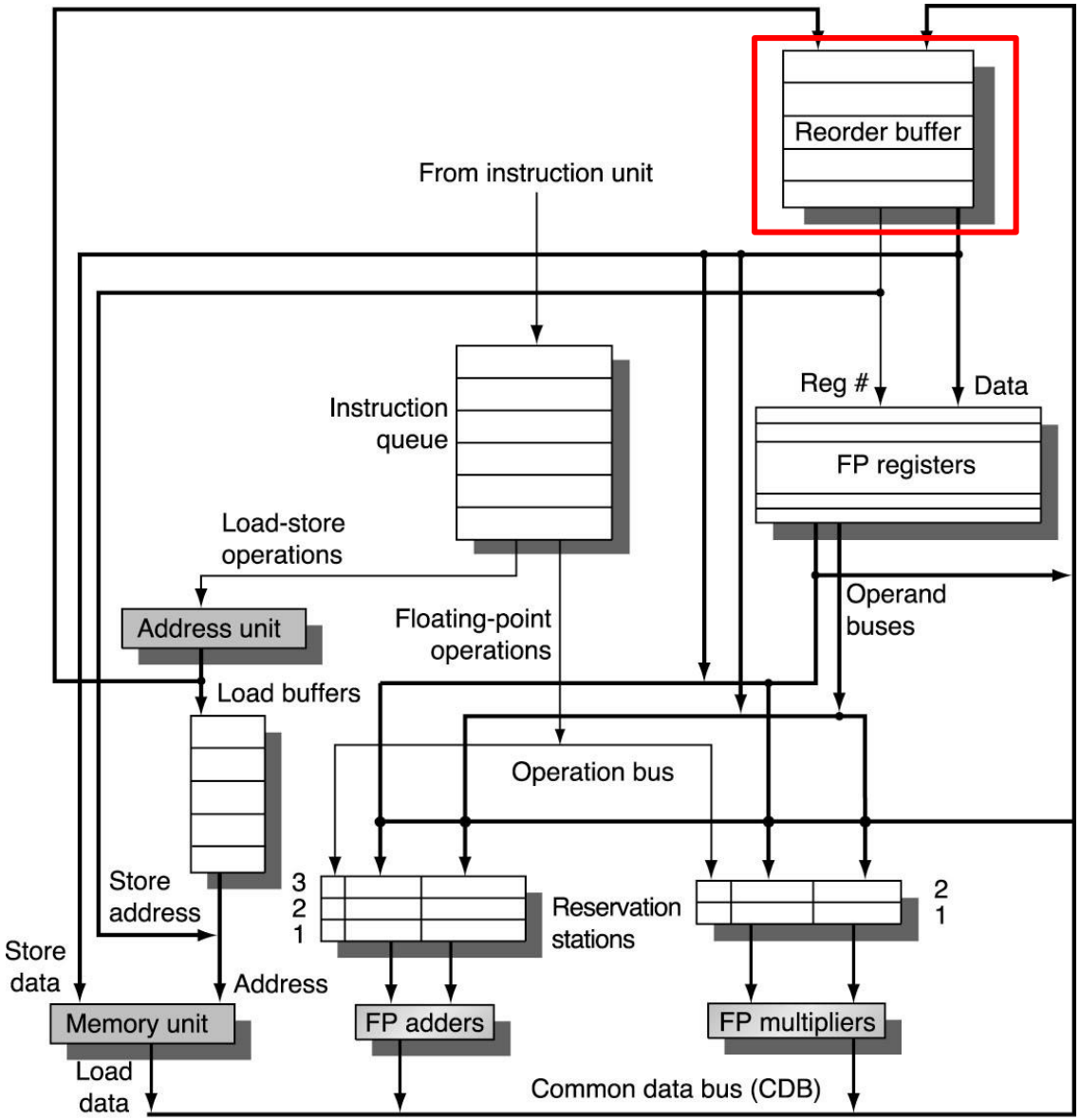
Strategies:

- Must separate **bypassing of results** among instructions from **actual completion** (write-back) of instructions
- Instruction commit updates register or memory when instruction **no longer speculative**

Need to add re-order buffer

- Hold the results of inst. that have finished exe but have not commiteed

Tomasulo extended to support speculation



ROB (reorder buffer)

entry	instruction type	destination	value	ready
1				
2				
...				
n				

- **Contains 4 fields:**
  - **Instruction type** indicates whether branch, store, or register op
  - **Destination field** memory or register
  - **Value field** hold the inst. result until commit
  - **Ready flag** indicates instruction has completed operation
- **Every instruction has a ROB entry until it commits**
  - Therefore tag results using ROB entry number
  - The renaming function of the reservation stations is **partially** replaced by the ROB

Four-step speculation

- **Issue:**
  - Get instruction from instruction queue and issue if reservation station and **ROB slots** available – sometimes called dispatch
  - Send operands or **ROB entry # (instead of RS #)**
- **Execution – operate on operands (EX)**
  - If both operands ready: execute; if not, watch CDB for result;
  - When both operands are in reservation station: execute
- **Write result – complete execution**
  - Write on CDB to all awaiting **FUs (RSs) & ROB** (tagged by ROB entry #)
  - Mark reservation station available
- **Commit – update register with reorder result**
  - When instr. **is at head of ROB & result is present & no longer speculative**; update register with result (or store to memory) and remove instr. from ROB;
  - handle mis-speculations and precise exceptions

ROB (reorder buffer)

- **When MUL.D is ready to commit**

```
L.D     F6,32(R2)
L.D     F2,44(R3)
MUL.D   F0,F2,F4
SUB.D   F8,F2,F6
DIV.D   F10,F0,F6
ADD.D   F6,F8,F2
```

Reorder buffer

Entry	Busy	Instruction		State	Destination	Value
1	No	L.D	F6,32(R2)	Commit	F6	Mem[32 + Regs[R2]]
2	No	L.D	F2,44(R3)	Commit	F2	Mem[44 + Regs[R3]]
3	Yes	MUL.D	F0,F2,F4	Write result	F0	#2 × Regs[F4]
4	Yes	SUB.D	F8,F2,F6	Write result	F8	#2 – #1
5	Yes	DIV.D	F10,F0,F6	Execute	F10	
6	Yes	ADD.D	F6,F8,F2	Write result	F6	#4 + #2

Reservation stations

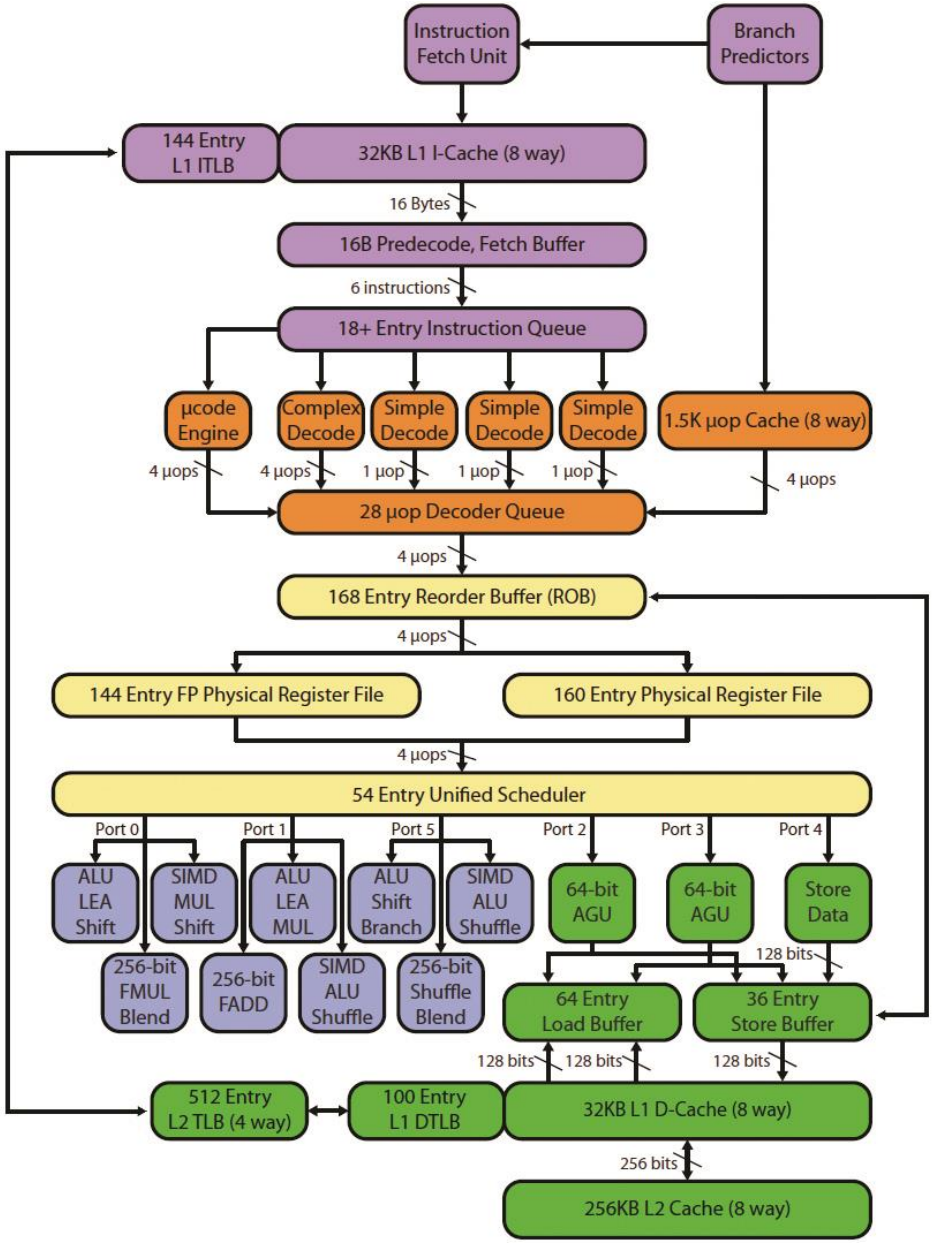
Name	Busy	Op	Vj	Vk	Qj	Qk	Dest	A
Load1	No							
Load2	No							
Add1	No							
Add2	No							
Add3	No							
Mult1	No	MUL.D	Mem[44 + Regs[R3]]	Regs[F4]			#3	
Mult2	Yes	DIV.D		Mem[32 + Regs[R2]]	#3		#5	

Four-step speculation

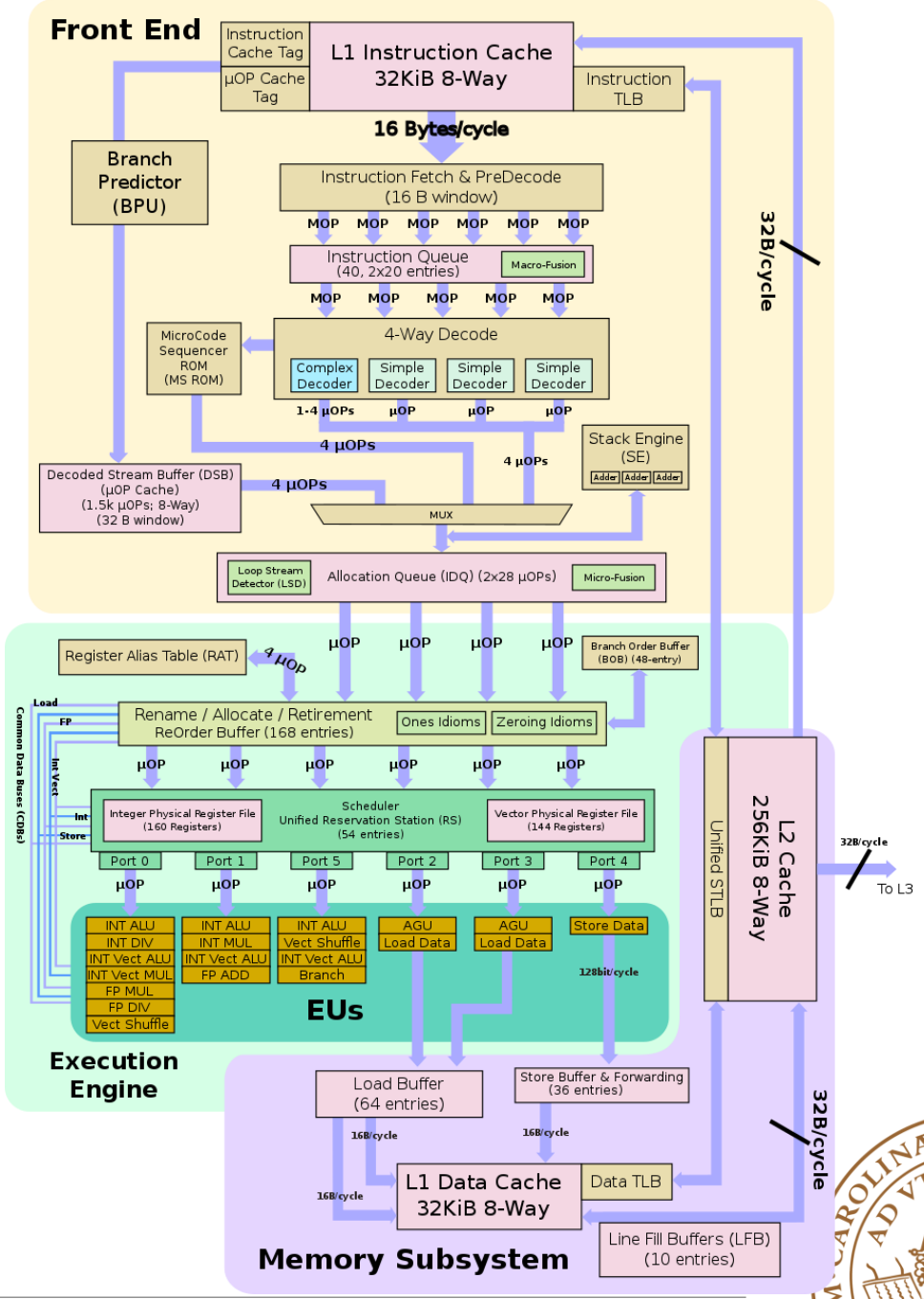
Commit – branch prediction wrong

- When branch instr. is at head of ROB & incorrect prediction (or exception): remove all instr. from reorder buffer (flush); restart execution at correct instruction
- Expensive ⇒ try to recover as early as possible (delay in ROB)
- Performance sensitive to branch prediction/speculation (waste computation power & time if wrong)

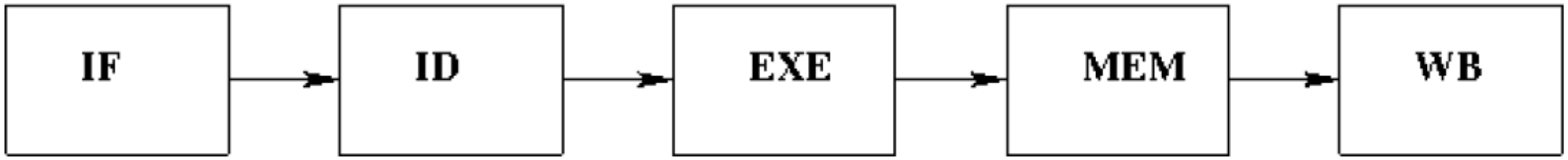
Sandy bridge microarchitecture



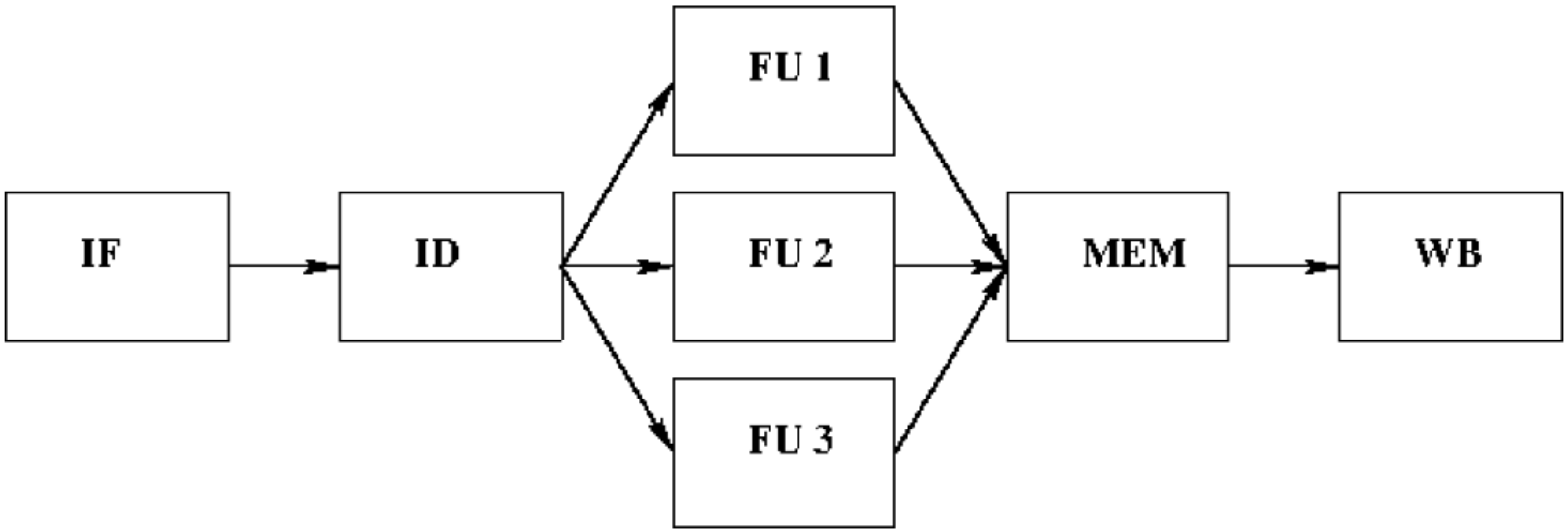
Sandy bridge microarchitecture



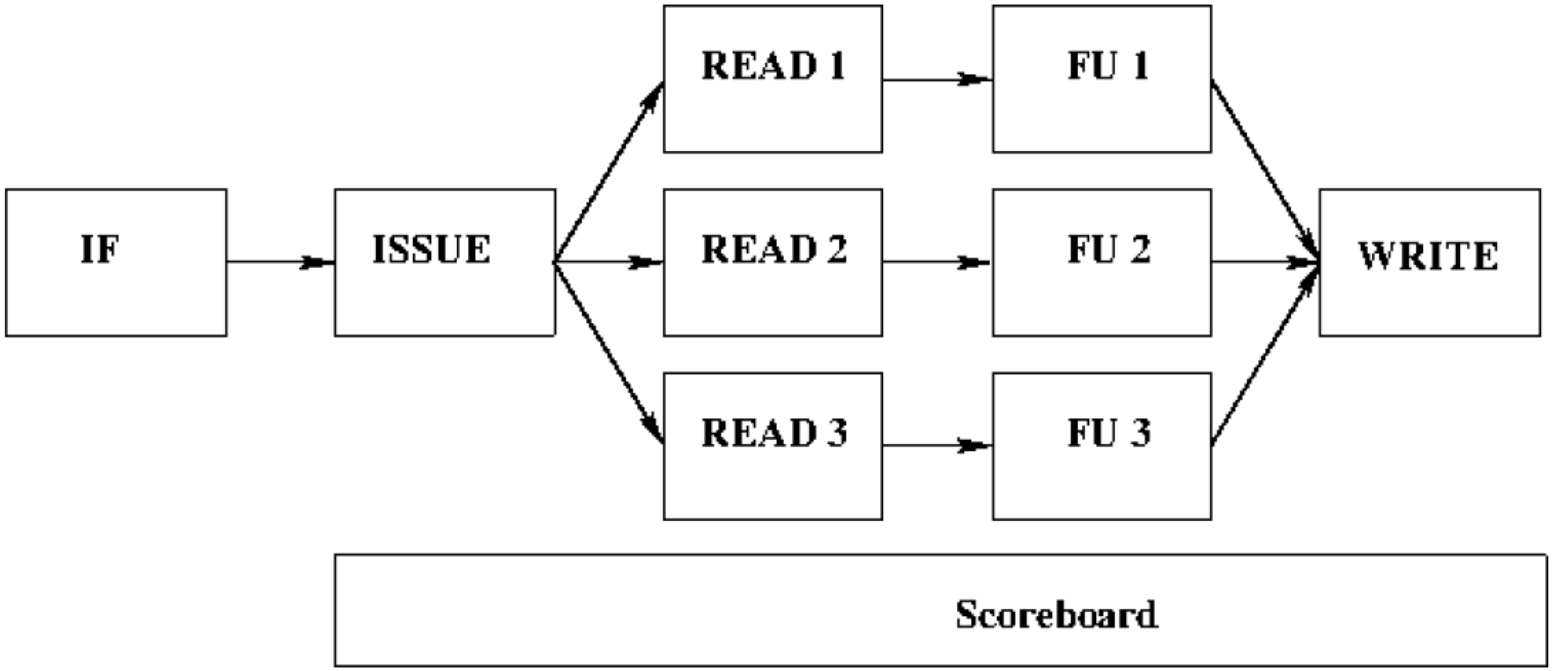
Basic 5-stage pipeline



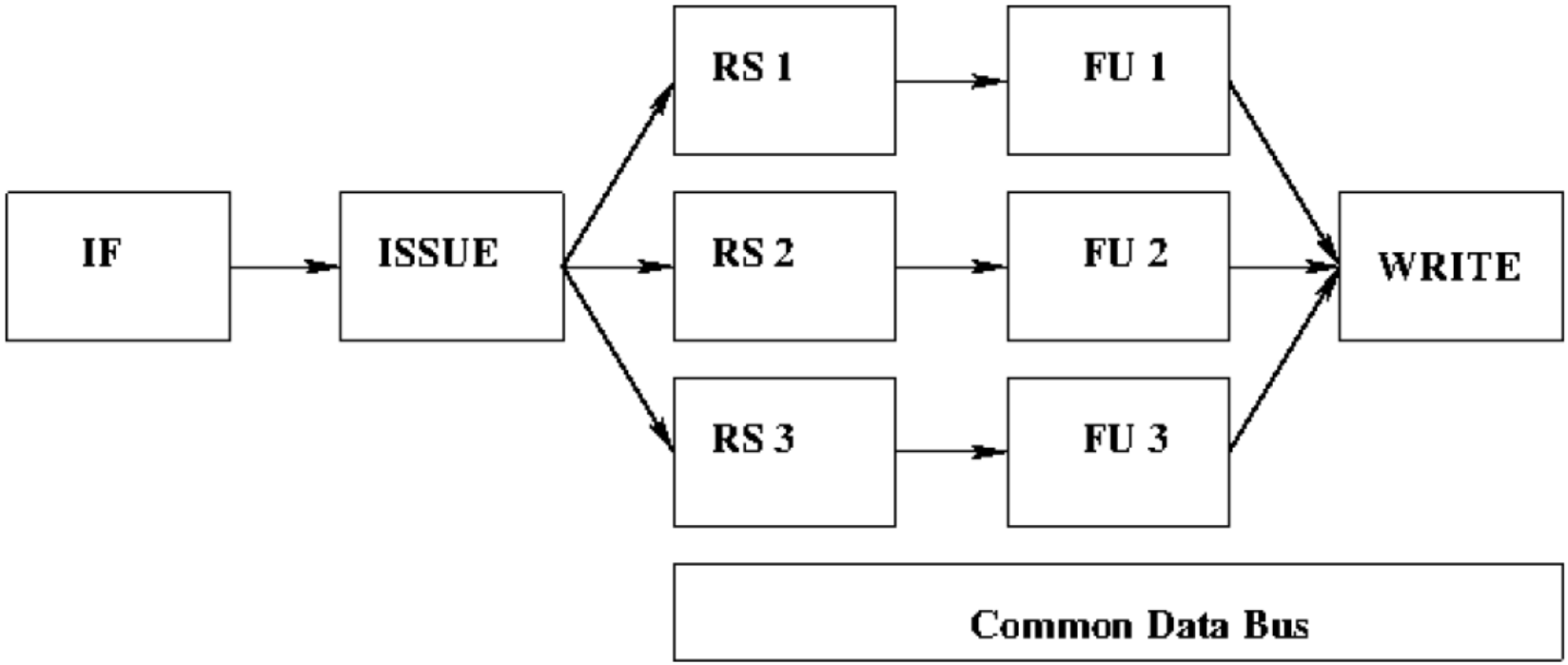
Pipeline with several FUs



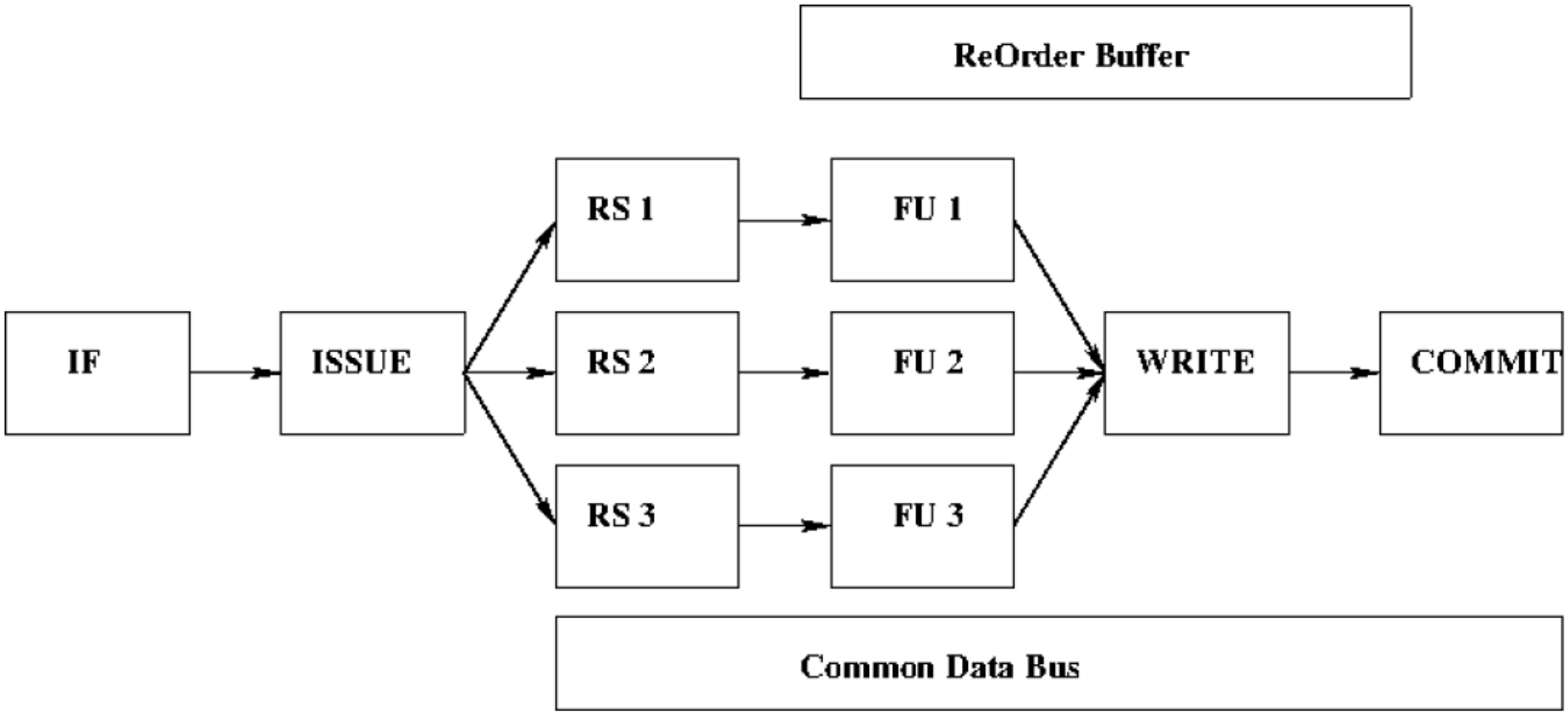
Scoreboard pipeline



Tomasulo pipeline



Tomasulo pipeline with speculation



Summary pipeline - implementation

Problem	Simple	Scoreboard	Tomasulo	Tomasulo + Speculation
	Static Sch	Dynamic Scheduling		
RAW				
WAR				
WAW				
Exceptions				
Issue				
Execution				
Completion				
Structural hazard				
Control hazard				

Summary pipeline - implementation

Problem	Simple	Scoreboard	Tomasulo	Tomasulo + Speculation
	Static Sch	Dynamic Scheduling		
RAW	forwarding stall	wait (Read)	CDB stall	CDB stall
WAR	-	wait (Write)	Reg. rename	Reg. rename
WAW	-	wait (Issue)	Reg. rename	Reg. rename
Exceptions	precise	?	?	precise, ROB
Issue	in-order	in-order	in-order	in-order
Execution	in-order	out-of-order	out-of-order	out-of-order
Completion	in-order	out-of-order	out-of-order	in-order
Structural hazard	-	many FU stall	many FU, CDB, stall	many FU, CDB, stall
Control hazard	Delayed br., stall	Branch prediction	Branch prediction	Br. pred, speculation

CPU performance equation