高等计算机系统结构

指令级并行处理

（第三讲）

程 旭

2014年3月31日

三种数据相关

1. *Data dependences* (also called true data dependences)
2. *name dependences*
3. *control dependences*

An instruction *j* is *data dependent* on instruction *i* if either of the following holds:

- Instruction *i* produces a result that may be used by instruction *j*.
- Instruction *j* is data dependent on instruction *k*, and instruction *k* is data dependent on instruction *i*.

名字相关（*Name Dependences*）

A name dependence occurs when two instructions use the same register or memory location, called a *name*, but there is no flow of data between the instructions associated with that name. （1）Antidependence; (2) output dependence

对于执行如下类型的指令序列：

$r_k \leftarrow (r_i)\ \text{op}\ (r_j)$

真数据相关（True Data-dependence, or Flow Denpendence）

$r_3 \leftarrow (r_1)\ \text{op}\ (r_2)$　　Read-after-Write

$r_5 \leftarrow (r_3)\ \text{op}\ (r_4)$　　(RAW) hazard

反相关（Anti-dependence）

$r_3 \leftarrow (r_1)\ \text{op}\ (r_2)$　　Write-after-Read

$r_1 \leftarrow (r_4)\ \text{op}\ (r_5)$　　(WAR) hazard

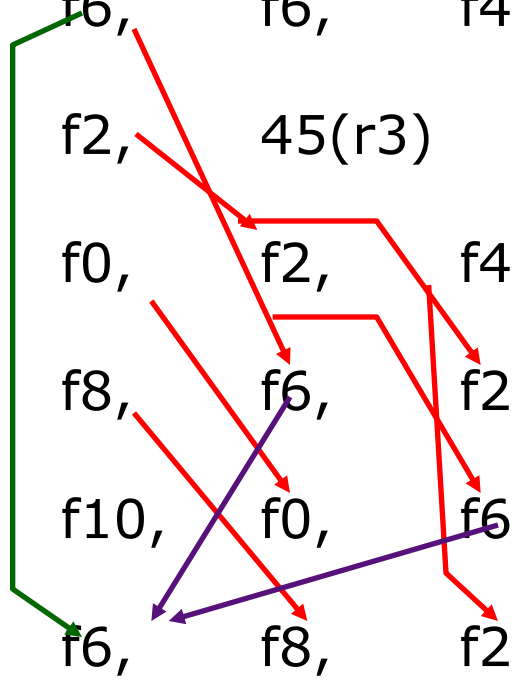
输出相关（Output-dependence）

$r_3 \leftarrow (r_1)\ \text{op}\ (r_2)$　　Write-after-Write

$r_3 \leftarrow (r_6)\ \text{op}\ (r_7)$　　(WAW) hazard

数据冒险示例

		dest	*src1*	*src2*
I_1	DIVD	f6,	f6,	f4
I_2	LD	f2,	45(r3)	
I_3	MULTD	f0,	f2,	f4
I_4	DIVD	f8,	f6,	f2
I_5	SUBD	f10,	f0,	f6
I_6	ADDD	f6,	f8,	f2

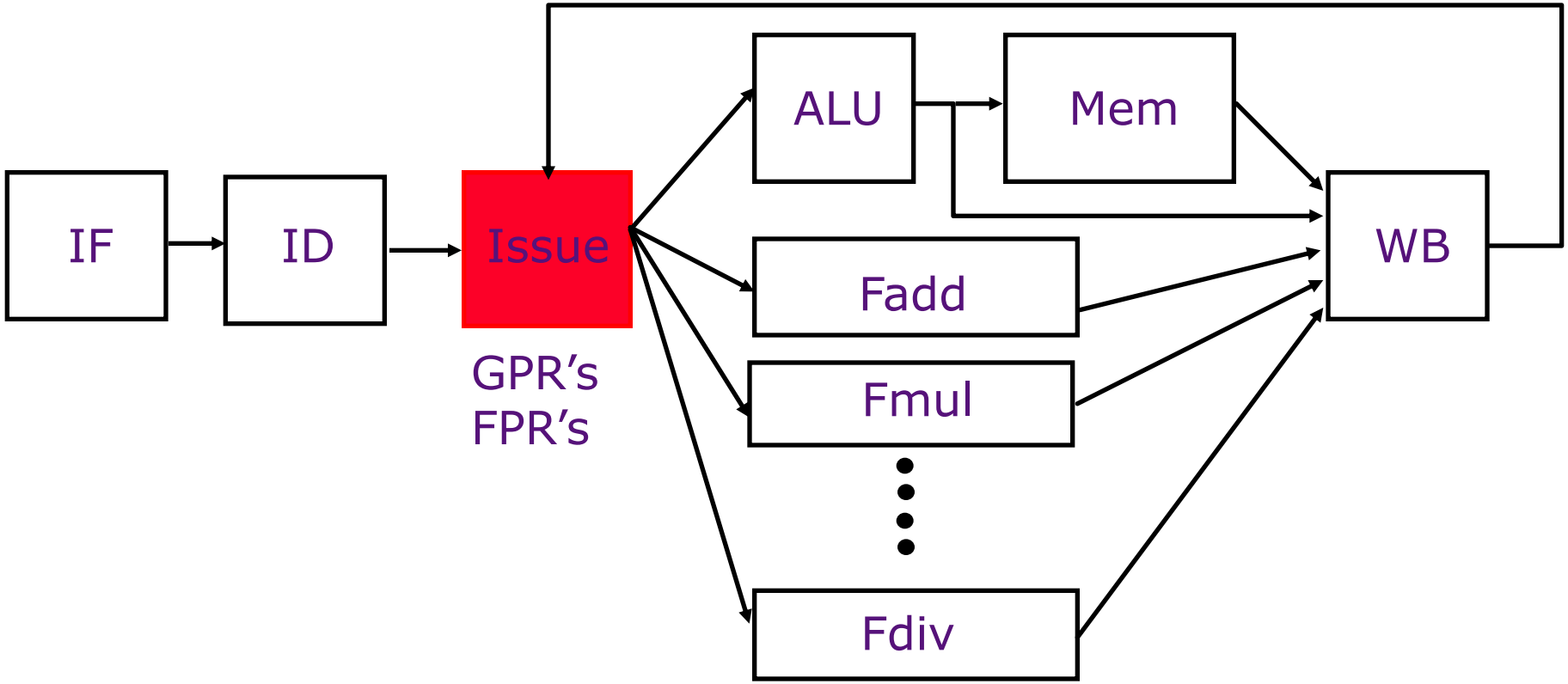


先写后读冒险（RAW Hazards）

先读后写冒险（WAR Hazards）

写写冒险（WAW Hazards）

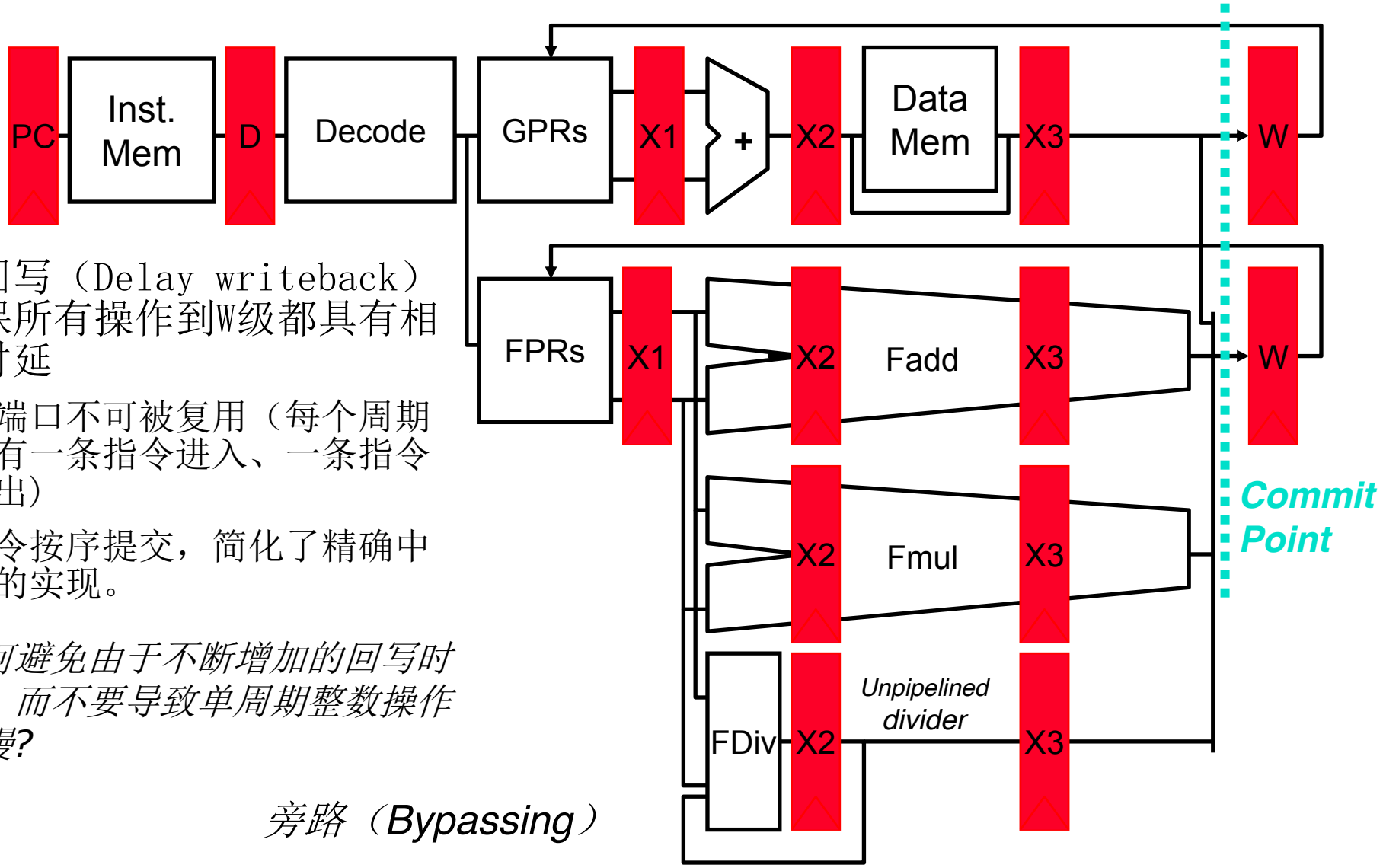
复杂指令流水线



为了追求更高性能，流水线变得更加复杂，这是因为：

- 流水化浮点部件的长时延
- 多功能和存储部件
- 具有可变访问时间的存储系统
- 精确中断

复杂按序指令流水线

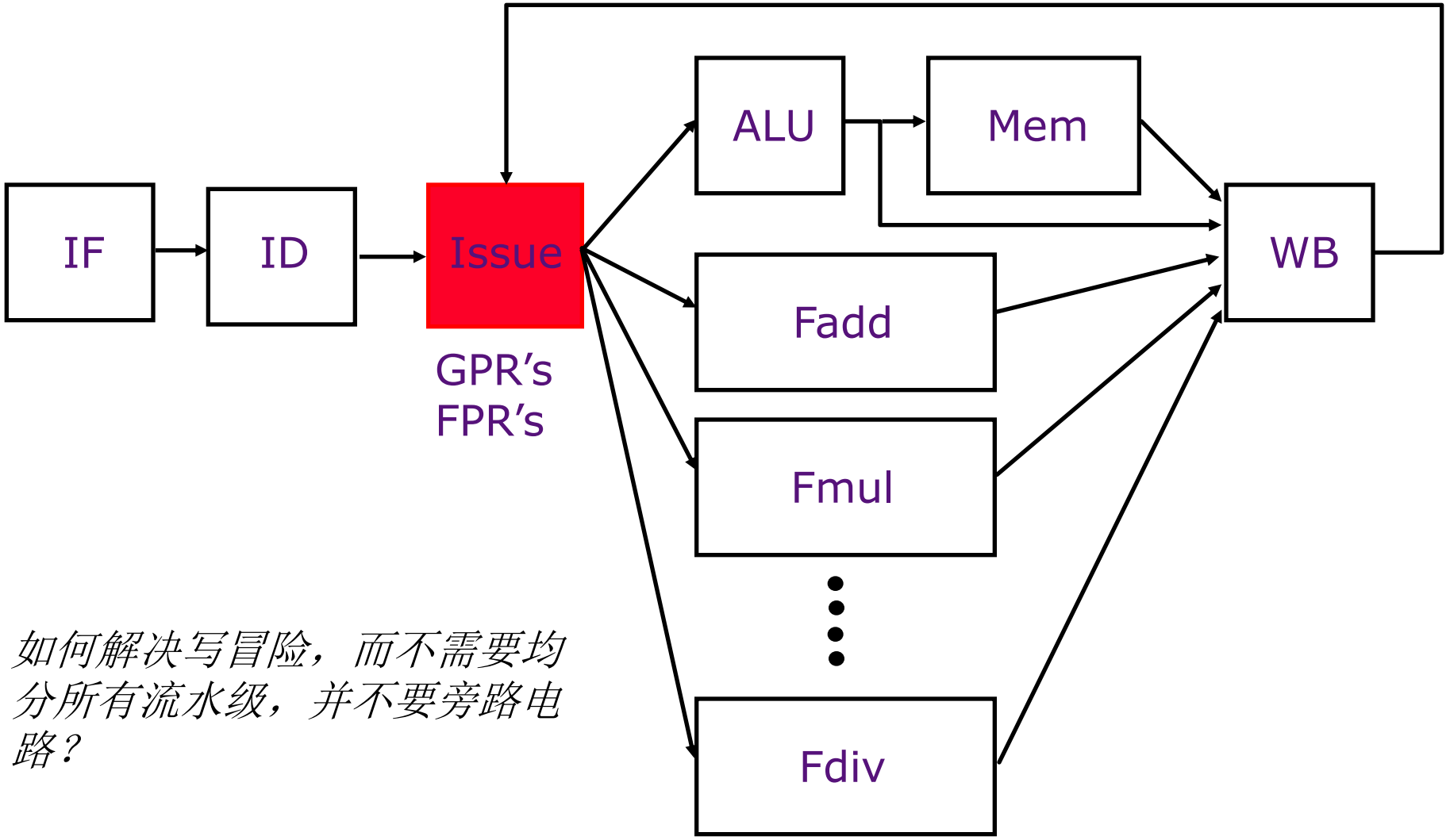


- 延迟回写（Delay writeback）以确保所有操作到W级都具有相同的时延
  - 写端口不可被复用（每个周期只有一条指令进入、一条指令流出）
  - 指令按序提交，简化了精确中断的实现。

如何避免由于不断增加的回写时延，而不要导致单周期整数操作变慢?

旁路（Bypassing）

复杂指令流水线



如何解决写冒险，而不需要均分所有流水级，并不要旁路电路？

何时可以安全地发射一条指令?

假设有一个统一的数据结构跟踪记录在所有功能部件中的所有指令状态

在发射级分发（dispatch）一条指令之前，需要完成如下检查：

- 所需功能部件是否可用?
- 输入数据是否可用? ⇒ RAW?
- 写目的操作数是否安全? ⇒ WAR? WAW?
- 是否在WB级会出现结构冒险?

硬件策略：指令并行

- 为什么需要硬件在运行时支持?
  - 在编译时有些相关情况不能真正判定
  - 简化编译处理
  - 针对某一机器产生的代码可以在另一机器上有效运行

- 核心思路：允许暂停之后的指令被处理

```
DIVD   F0,F2,F4
ADDD   F10,F0,F8
SUBD   F12,F8,F14
```

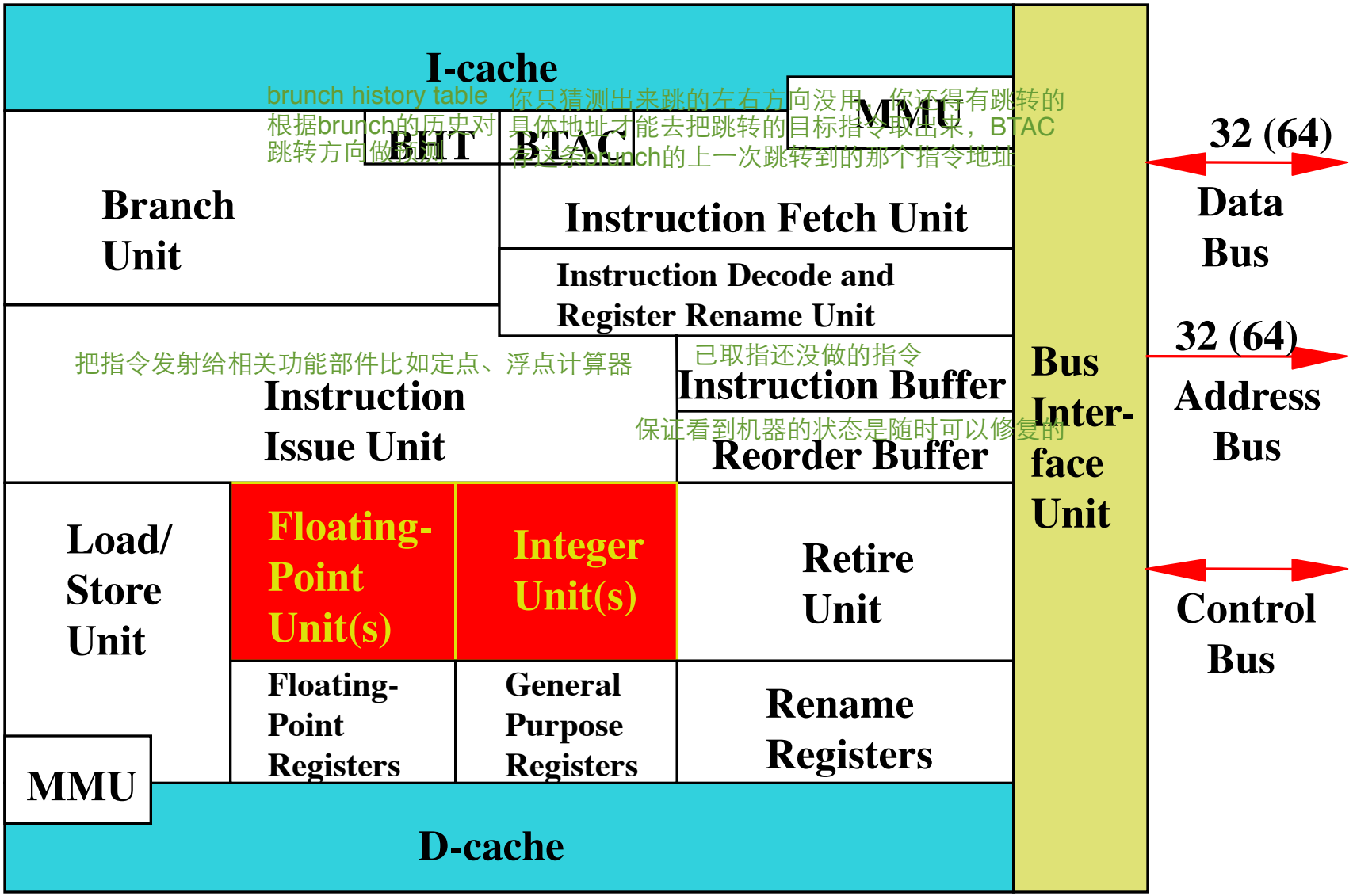
  - 允许乱序(out-of-order)执行 => 乱序完成
  - 在1963年的CDC 6600机器中，ID段检测结构冒险和记分板（Scoreboard）数据

- 核心思路：寄存器换名

```
DIVD   F0,F2,F4          DIVD F0,F2,F4
ADDD   F10,F0,F8         ADDD F10,F0,F8
SUBD   F0,F8,F14         SUBD F100,F8,F14
MULD   F6,F10,F0         MULD F6,F10,F100
```

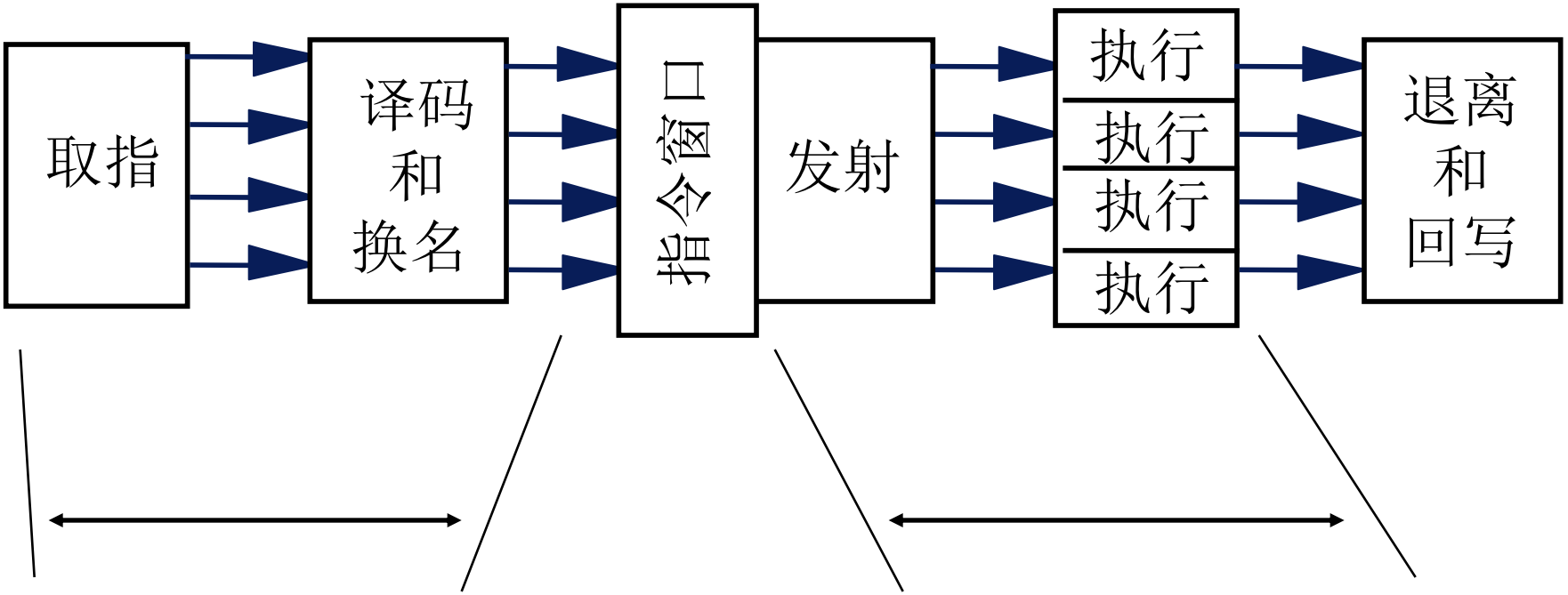
  - 消除WAR和WAW冒险

超标量处理器的内部部件



超标量流水线

顺序执行会带来不必要的hazard，本来没有dep的指令，仅仅因为放在了某些指令的后边就要被延迟执行
我们的理想是，不按照程序写就的指令顺序，而是按照数据可用性的顺序来安排执行，即指令的操作是用数据可用性驱动的
而非以指令顺序流来驱动(PC+1)



- 按序将指令递交到乱序执行的内核！

CDC 6600 *Seymour Cray, 1963*

- A fast pipelined machine with 60-bit words
  - 128 Kword main memory capacity, 32 banks
- Ten functional units (parallel, unpipelined)
  - Floating Point: adder, 2 multipliers, divider
  - Integer: adder, 2 incrementers, ...
- Hardwired control (no microcoding)
- *Scoreboard* for dynamic scheduling of instructions
- Ten Peripheral Processors for Input/Output
  - a fast multi-threaded 12-bit integer ALU
- Very fast clock, 10 MHz (FP add in 4 clocks)
- >400,000 transistors, 750 sq. ft., 5 tons, 150 kW, novel freon-based technology for cooling
- Fastest machine in world for 5 years (until 7600)
  - over 100 sold ($7-10M each)

CDC 6600: A Load/Store Architecture

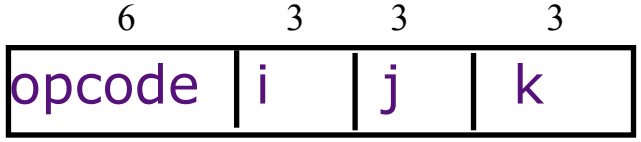
- Separate instructions to manipulate three types of reg.

  8x60-bit data registers (X)

  8 18-bit address registers (A)

  8 18-bit index registers (B)

- All arithmetic and logic instructions are reg-to-reg



Ri ← (Rj) op (Rk)

- Only Load and Store instructions refer to memory!

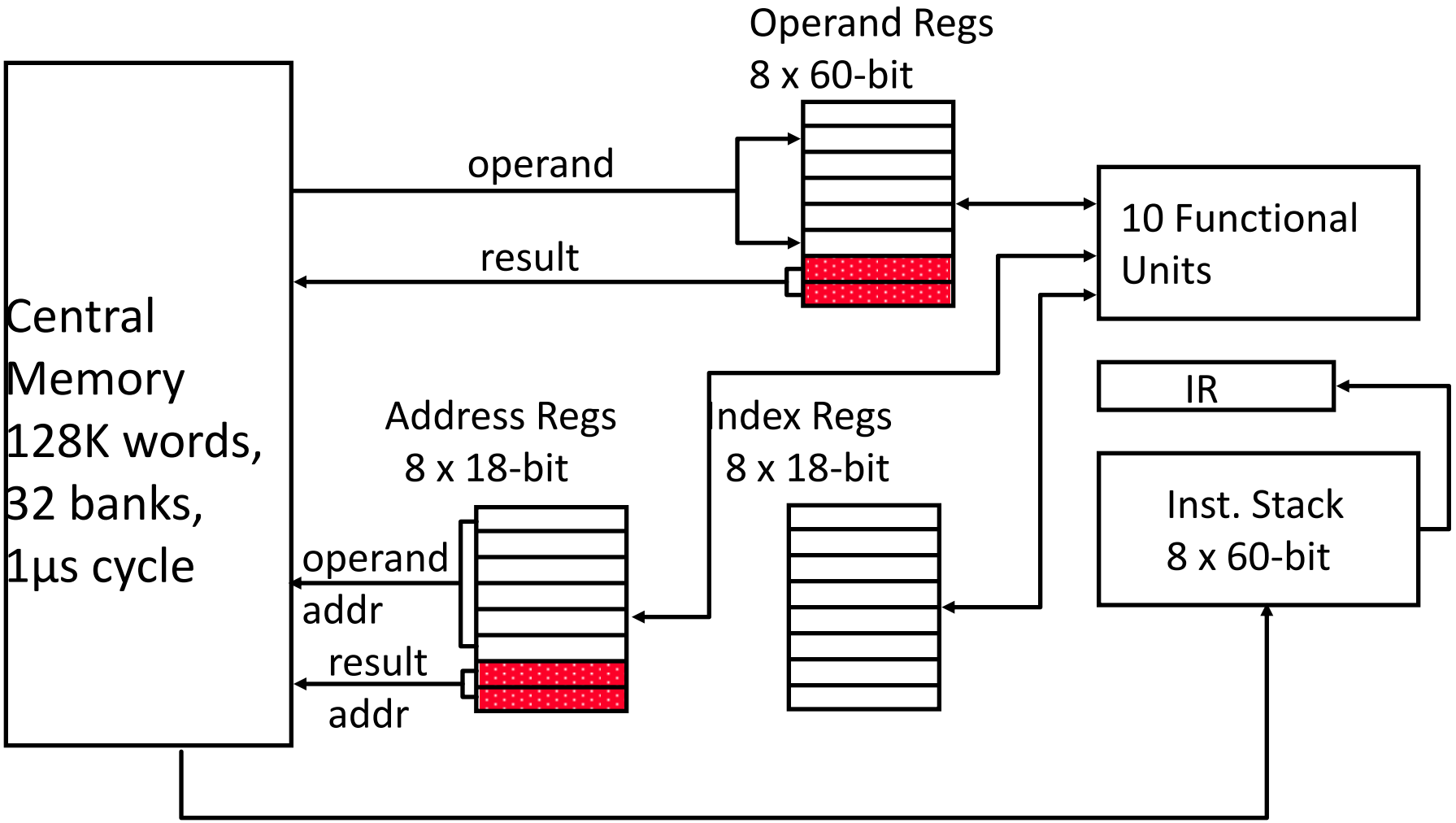
Ri ← M[(Rj) + disp]

Touching address registers 1 to 5 initiates a load

6 to 7 initiates a store

- very useful for vector operations

CDC 6600: Datapath



CDC6600 ISA designed to simplify high-performance implementation

- Use of three-address, register-register ALU instructions simplifies pipelined implementation
  - Only 3-bit register specifier fields checked for dependencies
  - No implicit dependencies between inputs and outputs
- Decoupling setting of address register (Ar) from retrieving value from data register (Xr) simplifies providing multiple outstanding memory accesses
  - Software can schedule load of address register before use of value
  - Can interleave independent instructions inbetween
- CDC6600 has multiple parallel but unpipelined functional units
  - E.g., 2 separate multipliers
- Follow-on machine CDC7600 used pipelined functional units
  - Foreshadows later RISC designs

CDC6600: Vector Addition

```
        B0      - n
loop:   JZE   B0, exit
        A0      B0 + a0         load X0
        A1      B0 + b0         load X1
        X6      X0 + X1
        A6      B0 + c0         store X6
        B0      B0 + 1
        jump loop
```

Ai = address register

Bi = index register

Xi = data register

CDC6600 Scoreboard

- Instructions dispatched in-order to functional units provided no structural hazard or WAW
  - Stall on structural hazard, no functional units available
  - Only one pending write to any register
- Instructions wait for input operands (RAW hazards) before execution
  - Can execute out-of-order
- Instructions wait for output register to be read by preceding instructions (WAR)
  - Result held in functional unit until register free

IBM Memo on CDC6600

Thomas Watson Jr., IBM CEO, August 1963:

> *"Last week, Control Data ... announced the 6600 system. I understand that in the laboratory developing the system there are only 34 people including the janitor. Of these, 14 are engineers and 4 are programmers... Contrasting this modest effort with our vast development activities, I fail to understand why we have lost our industry leadership position by letting someone else offer the world's most powerful computer."*

To which Cray replied: *"It seems like Mr. Watson has answered his own question."*

支持按序发射指令的记分板技术
Scoreboard for In-order Issues

把寄存器和每条指令所处的状态记录在一块板子上

Busy[FU#]：a bit-vector to indicate FU's availability.
(FU = Int, Add, Mult, Div)

These bits are hardwired to FU's.

write pending

WP[reg#]：a bit-vector to record the registers for which writes are pending.

These bits are set to true by the Issue stage and set to false by the WB stage

我现在要读一个数，我想知道这个数是否真的已经写进去了，我就去读这个操作数的wp，如果还没写完，我就等着

Issue checks the instruction (opcode dest src1 src2) against the scoreboard (Busy & WP) to dispatch

FU available?	Busy[FU#]
RAW?	WP[src1] or WP[src2]
WAR?	*cannot arise*
WAW?	WP[dest]

硬件策略：指令并行（续一）

- 乱序执行 分解 ID段：
  1. Issue—decode instructions, check for structural hazards 对指令译码，看有没有structural hazard，如果没有就发到下一个阶段
  2. Read operands—wait until no data hazards, then read operands 看有没有RAWhazard，如果没有就发到下一阶段执行，如果有就在这里挡住

- 只要指令同时满足上述两个条件，记分板就允许该指令执行，而无需等待前面的指令完成

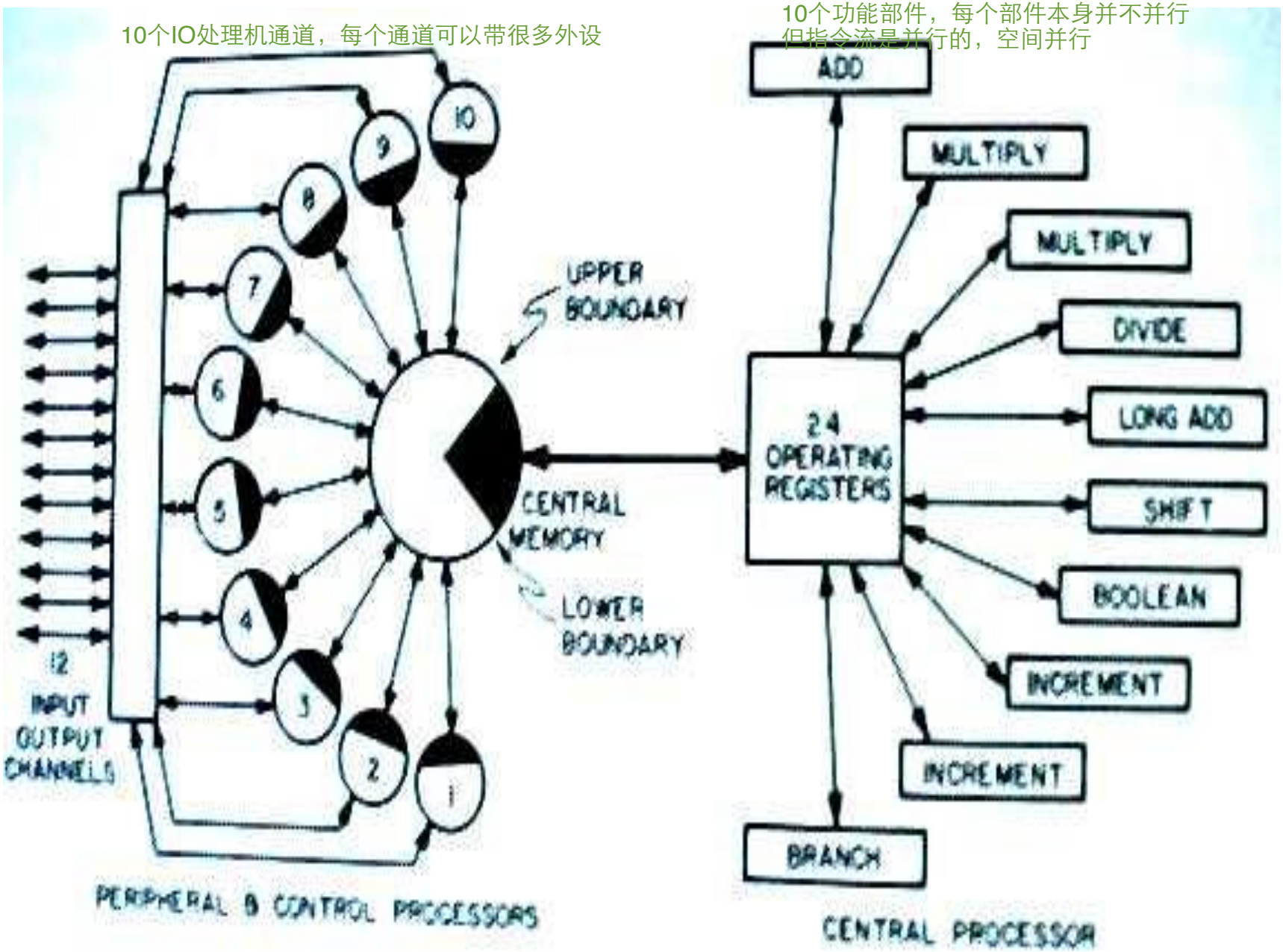
- CDC 6600: 第一代并不关心是否按序提交，第一代是乱序执行，乱序提交的，精确中断什么的不管
  - 按序发射
  - 乱序执行
  - 乱序提交（commit） （ 也就是完成[completion]）

CDC 6600
logic gates

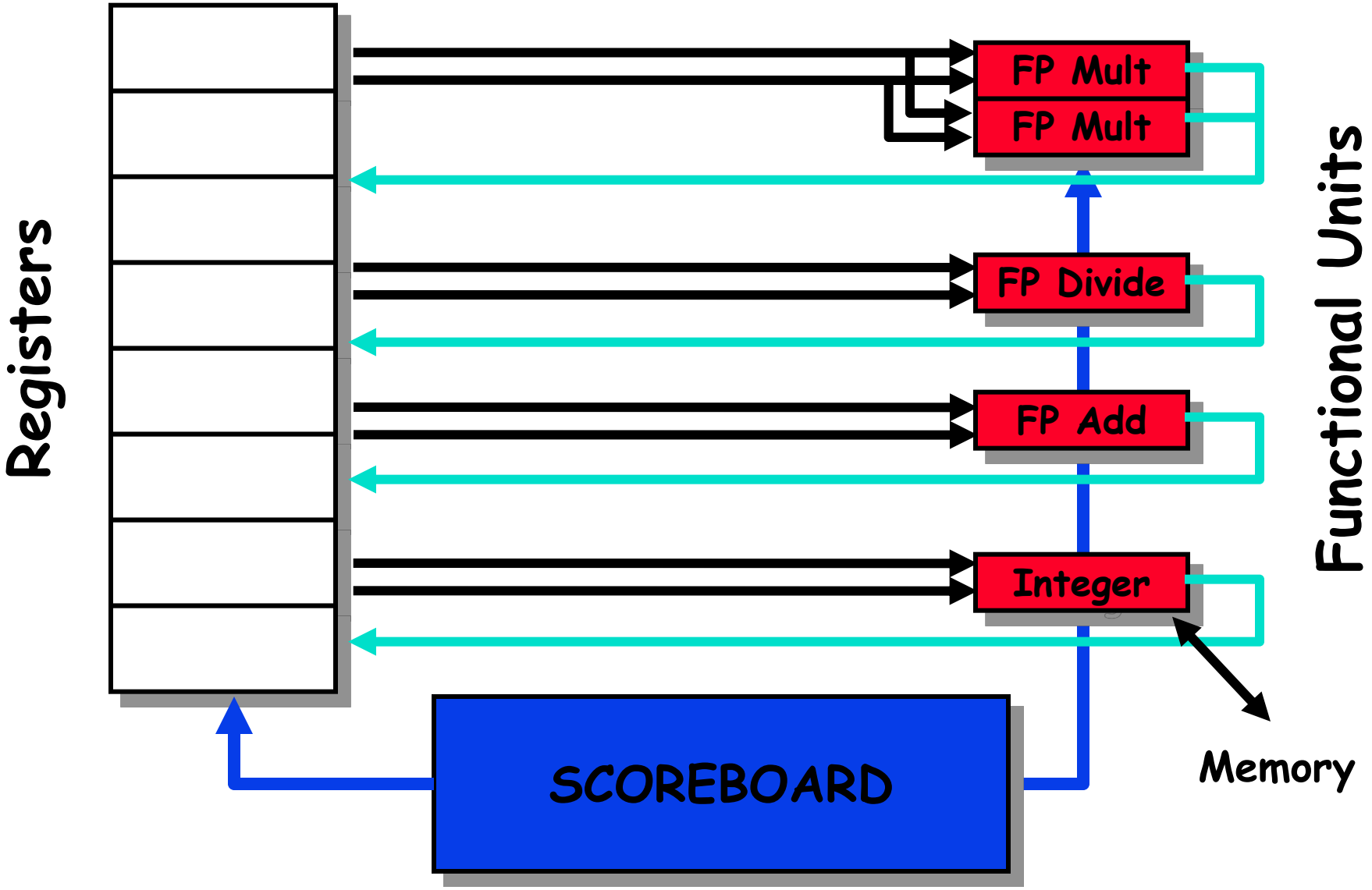
CDC 6600 结构简图

10个IO处理机通道，每个通道可以带很多外设

10个功能部件，每个部件本身并不并行
但指令流是并行的，空间并行



记分板体系结构


Registers
FP Mult
FP Mult
FP Divide
FP Add
Integer
Functional Units
SCOREBOARD
Memory

记分板的含义

- 乱序完成 => WAR, WAW冒险?
- 对WAR的解决方案
  - 排队等待操作以及它们操作数的拷贝
  - 只在读操作段才读取寄存器
- 对WAW的解决方案，必须检测冒险：暂停等待到其他指令完成
- 在执行阶段可能有多个指令 => 设置多个执行部件或者流水化执行部件
- 记分板跟踪相关、状态或操作
- 记分板用四个流水段代替ID、EX、WB三段

记分板控制的四级

当时还没解决control hazard，遇到跳转就停止等待

1. **Issue**—decode instructions & check for structural hazards (ID1)

对指令译码，然后看看有没有structural hazard，如果没有就发到下一阶段

If a functional unit for the instruction is free and no other active instruction has the same destination register (WAW), the scoreboard issues the instruction to the functional unit and updates its internal data structure. If a structural or WAW hazard exists, then the instruction issue stalls, and no further instructions will issue until these hazards are cleared.

2. **Read operands**—wait until no data hazards, then read operands (ID2)

解决RAW hazard，就去查两个source的WP上标志的是不是这俩都是空的没有别的指令要写

A source operand is available if no earlier issued active instruction is going to write it, or if the register containing the operand is being written by a currently active functional unit. When the source operands are available, the scoreboard tells the functional unit to proceed to read the operands from the registers and begin execution. The scoreboard resolves RAW hazards dynamically in this step, and instructions may be sent into execution out of order.

1、2保证了我可以用data driven来做?

记分板控制的四级（续一）

3. Execution—operate on operands (EX)

The functional unit begins execution upon receiving operands. When the result is ready, it notifies the scoreboard that it has completed execution.

这个过程只管执行，啥心也不用操

4. Write result—finish execution (WB)

Once the scoreboard is aware that the functional unit has completed execution, the scoreboard checks for WAR hazards. If none, it writes results. If WAR, then it stalls the instruction.

Example:

```
DIVD    F0,F2,F4
ADDD    F10,F0,F8
SUBD    F8,F8,F14
```

指令i有两个source，有可能a ready了，b还没有ready，a就得放到寄存器里头，等b好了才一起被读走，这时如果指令j的EX阶段完毕想要WB到寄存器a，但因为b没有ready因此a也没有被读走，j就得被stall，这样就产生了WAR hazard

CDC 6600 scoreboard would stall SUBD until ADDD reads operands

记分板的三个主要组成部分

1. **Instruction status**—which of 4 steps the instruction is in

2. **Functional unit status**—Indicates the state of the functional unit (FU). 9 fields for each functional unit

   Busy—Indicates whether the unit is busy or not

   Op—Operation to perform in the unit (e.g., + or –)
   比如加法器和减法器是一个功能部件，但做加法还是做减法需要有控制符来确定

   Fi—Destination register
   要描述目标寄存器和源寄存器是什么
   把该部件当前的寄存器使用情况描述清楚

   Fj, Fk—Source-register numbers
   若源数据产生好了，就用Fj/Fk来记录源数据在哪个寄存器中
   若原还没产生好，就用Qj、Qi来记录源数据将会从哪个部件产生

   Qj, Qk—Functional units producing source registers Fj, Fk

   Rj, Rk—Flags indicating when Fj, Fk are ready
   Rj，Rk，Qj，Qk都是布尔变量

这个就是WP标志位，标记是否有FU在这个reg的write pending上，一个reg的write pending在同一时间只能有一个，因此只需一位标志即可

3. **Register result status**—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions will write that register

记分板流水线控制的细节

bookkeeping就是对计分板的改变

Instruction status	Wait until	Bookkeeping
Issue	Not busy (FU) and not result(D)	Busy(FU)← yes; Op(FU)← op; Fi(FU)← `D'; Fj(FU)← `S1'; Fk(FU)← `S2'; Qj← Result('S1'); Qk← Result(`S2'); Rj← not Qj; Rk← not Qk; Result('D')← FU;
Read operands	Rj and Rk	Rj← No; Rk← No
Execution complete	Functional unit done	这一阶段不判断什么相关，也不做book keeping，就执行
Write result	∀f((Fj(f)≠Fi(FU) or Rj(f)=No) & (Fk(f) ≠Fi(FU) or Rk(f)=No))	∀f(if Qj(f)=FU then Rj(f)← Yes); ∀f(if Qk(f)=FU then Rj(f)← Yes); Result(Fi(FU))← 0; Busy(FU)← No

若S1有写pending，则说明S1还没ready，否则就是S1已ready

Rj和Rk都ready后，就把数读出来。然后再做完这一步的bookkeeping，就进入下一阶段

左栏的条件满足后就可以写数据了
写完数据进行右栏的操作

记分板示例

Instruction status

Instruction		j	k	Issue	Read operands	Execution complete	Write Result
LD	F6	34+	R2				
LD	F2	45+	R3				
MUL	F0	F2	F4				
SUB	F8	F6	F2				
DIVD	F10	F0	F6				
ADD	F6	F8	F2				

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
FU									

记分板示例第一个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1			
LD F2	45+	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIVI F10	F0	F6				
ADD F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU foı Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
1	FU				Integer					

记分板示例第二个周期

Instruction status

Instruction	j	k	Issue	Read operand	Execution complete	Write Result
LD F6	34+	R2	1	2		
LD F2	45+	R3				
MULT F0	F2	F4				
SUBI F8	F6	F2				
DIVD F10	F0	F6				
ADDI F6	F8	F2				

- **Issue 2nd LD?**

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6	立即数	R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
2	FU				Integer					

记分板示例第三个周期

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
LD F6	34+	R2	1	2	3	
LD F2	45+	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIVD F10	F0	F6				
ADD F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F6		R2				No
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
3	FU	Integer								

- **Issue MULT?**

记分板示例第四个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3				
MUL F0	F2	F4				
SUB F8	F6	F2				
DIVI F10	F0	F6				
ADD F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
4	FU									

记分板示例第五个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5			
MUL F0	F2	F4				
SUB F8	F6	F2				
DIVI F10	F0	F6				
ADD F6	F8	F2				

我做的是in order的issue，因此第二个LD不做我都不去看MUL

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
5	FU		Integer							

记分板示例第六个周期

Instruction status

Instruction	j	k	Issue	Read operand	Execution complete	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6		
MUL F0	F2	F4	6			
SUB F8	F6	F2				
DIVD F10	F0	F6				
ADD F6	F8	F2				

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
6	FU	Mult1	Integer							

记分板示例第七个周期

Instruction status

Instruction	j	k	Issue	Read operands	Execution complete	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	
MUL F0	F2	F4	6			
SUB F8	F6	F2	7			
DIVD F10	F0	F6				
ADD F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				No
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
7	FU	Mult1	Integer			Add				

- **Read multiply operands?**

记分板示例第8a个周期(前半个周期)

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	
MUL F0	F2	F4	6			
SUB F8	F6	F2	7			
DIVI F10	F0	F6	8			
ADD F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	Yes	Load	F2		R3				No
	Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Integer	Yes	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Mult1	Integer			Add	Divide			

记分板示例第8b个周期(后半个周期)

Instruction status

Instruction	j	k	Issue	Read operand	Execution complete	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6			
SUB F8	F6	F2	7			
DIVD F10	F0	F6	8			
ADD F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
8	FU	Mult1				Add	Divide			

记分板示例第九个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9		
DIVI F10	F0	F6	8			
ADD F6	F8	F2				

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

Functional unit status

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU foı Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
10	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
2	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
9	FU	Mult1				Add	Divide			

- **Read operands for MULT & SUBD? Issue ADDD?**

记分板示例第十个周期

Instruction status

Instruction	j	k	Issue	Read operand	Execution complete	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9		
DIVD F10	F0	F6	8			
ADD F6	F8	F2				

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
9	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
1	Add	Yes	Sub	F8	F6	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
10	FU	Mult1				Add	Divide			

记分板示例第十一个周期

Instruction status

Instruction	j	k	Issue	Read operand	Execution complete	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	
DIVD F10	F0	F6	8			
ADD F6	F8	F2				

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
8	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
0	Add	Yes	Sub	F8	F6	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
11	FU	Mult1				Add	Divide			

记分板示例第十二个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8			
ADD F6	F8	F2				

Functional unit status

Tim	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU fo Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
7	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	No								
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
12	FU	Mult1					Divide			

- **Read operands for DIVD?**

记分板示例第十三个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8			
ADD F6	F8	F2	13			

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
6	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
13	FU	Mult1			Add		Divide			

记分板示例第十四个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8			
ADD F6	F8	F2	13	14		

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
5	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
2	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
14	FU	Mult1			Add		Divide			

记分板示例第十五个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8			
ADD F6	F8	F2	13	14		

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
4	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
1	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
15	FU	Mult1			Add		Divide			

记分板示例第十六个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
3	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
0	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
16	FU	Mult1			Add		Divide			

记分板示例第十七个周期

Instruction status

Instruction	j	k	Issue	Read operand	Execution complete	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9		
SUB F8	F6	F2	7	9	11	12
DIVD F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

WAR Hazard!

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
2	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
17	FU	Mult1			Add		Divide			

- **Write result of ADDD?**

记分板示例第十八个周期

Instruction status

Instruction		j	k	Issue	Read operand	Execution complete	Write Result
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MUL	F0	F2	F4	6	9		
SUB	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADD	F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
1	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
18	FU	Mult1			Add		Divide			

记分板示例第十九个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
0	Mult1	Yes	Mult	F0	F2	F4			No	No
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
19	FU	Mult1			Add		Divide			

记分板示例第二十个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8			
ADD F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			No	No
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
20	FU				Add		Divide			

记分板示例第二十一个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8	21		
ADD F6	F8	F2	13	14	16	

Functional unit status

Time Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
Integer	No								
Mult1	No								
Mult2	No								
Add	Yes	Add	F6	F8	F2			No	No
Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
21	FU				Add		Divide			

记分板示例第二十二个周期

Instruction status

Instructio	j	k	Issue	Read operan	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8	21		
ADD F6	F8	F2	13	14	16	22

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
40	Divide	Yes	Div	F10	F0	F6			No	No

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
22	FU						Divide			

记分板示例第六十一个周期

Instruction status

Instructio	j	k	Issue	Read operat	Execu compl	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIVI F10	F0	F6	8	21	61	
ADD F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for Qj	FU for Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	Yes	Div	F10	F0	F6			No	No

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
61	FU						Divide			

记分板示例第六十二个周期

Instruction status

Instruction	j	k	Issue	Read operand	Execution complete	Write Result
LD F6	34+	R2	1	2	3	4
LD F2	45+	R3	5	6	7	8
MUL F0	F2	F4	6	9	19	20
SUB F8	F6	F2	7	9	11	12
DIVD F10	F0	F6	8	21	61	62
ADD F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	No								

Register result status

Clock		F0	F2	F4	F6	F8	F10	F12	...	F30
62	FU									

CDC 6600 的记分板

- 来自编译的加速比1.7；手编代码的加速比2.5，但是由于存储速度慢（没有Cache）限制了加速比的提高

- 6600记分板的局限性：
  - 没有前递硬件　类似于“数据旁路”
  - 指令调度局限于基本块内(指令窗口小)　只处理了data dep和name dep，control dep就没有碰，这就只能在basic block里头做
  - 功能部件少（结构冒险），特别是integer/load store部件
  - 存在结构冒险，就暂停发射指令
  - 等待到WAR冒险解决　在WB阶段，遇到WAR就stall了，这不好
  - 防止WAW冒险

上来第一步就要看WAW，一遇到WAW就干脆暂停issue了，CDC6600就8个寄存器，name hazard到处都是，WAW太多

本讲小结

- 软件或硬件的指令级并行（ILP）
- 循环级并行最容易判定
- 软件并行性取决于程序，如果硬件不能支持就出现冒险
- 软件相关性/编译器复杂性决定编译中是否能展开循环
  - 存储器相关是最难判定的
- 硬件开采ILP
  - 在编译时有些相关情况不能真正判定
  - 针对某一机器产生的代码可以在另一机器上有效运行
- 记分板的核心思想：允许暂停之后的指令提前处理(译码 => 发射指令 & 读取操作数)
  - 允许乱序执行 => 乱序完成
  - ID段检测所有的结构冒险