

Computer Architecture

08. Exploiting ILP

Jianhua Li

College of Computer and Information
Hefei University of Technology

本章内容概要

4.1 指令级并行的概念

4.2 指令的动态调度

4.3 动态分支预测技术

4.4 多指令流出技术

4.5 循环展开和指令调度

指令级并行的概念

- 处理机利用流水线来使指令重叠并行执行，以达到提高性能的目的。这种指令之间存在的潜在并行性称为指令级并行(**ILP: Instruction-Level Parallelism**)
- 本章讨论：如何通过各种可能的技术，获得更多的指令级并行性。
 - 硬件+软件：必须要硬件技术和软件技术互相配合，才能够最大限度地挖掘出程序中存在的指令级并行。

指令级并行的概念

- 流水线处理机的实际CPI

- 理想流水线的CPI加上各类停顿的时钟周期数：

$$CPI_{\text{流水线}} = CPI_{\text{理想}} + \text{停顿}_{\text{结构冲突}} + \text{停顿}_{\text{数据冲突}} + \text{停顿}_{\text{控制冲突}}$$

- 理想CPI是衡量流水线最高性能的一个指标。

- IPC: Instructions Per Cycle

- 基本程序块 (Basic Block)

- 基本程序块：一段除了入口和出口以外不包含其他分支的线性代码段。
- 据统计：程序平均每5~7条指令就会有一个分支。

指令级并行的概念

- 循环级并行：使一个循环中的不同循环体并行执行。

- 开发循环体中存在的并行性
 - 最常见、最基本
- 是指令级并行研究的重点之一
- 例如，考虑下述语句：

```
for (i=1; i<=500; i++)
```

```
    a[i]=a[i]+s;
```

- 每一次循环都可以与其他的循环重叠并行执行；
- 在每一次循环的内部，却没有任何的并行性。

指令级并行的概念

- 最基本的开发循环级并行的技术
 - 循环展开 (loop unrolling)
 - 采用向量指令和向量数据表示
- 相关与流水线冲突
 - 流水线冲突是指对于具体的流水线来说，相关导致指令流中的下一条指令不能在指定的时钟周期执行。
 - 流水线冲突类型：结构冲突、数据冲突、控制冲突

指令级并行的概念

- **相关与流水线冲突（续）**

- 相关是程序固有的一种属性，它反映了程序中指令之间的相互依赖关系。
- 具体的一次相关是否会导致实际冲突的发生以及该冲突会带来多长的停顿，则是流水线的属性。

- **可以从两个方面来解决相关问题：**

- 保持相关，但避免发生冲突；
- 通过代码变换，消除相关；

- **程序顺序：由源程序确定的在完全串行方式下指令的执行顺序。**

- 由于相关的存在可能影响程序结果，所以必须保持程序顺序。

指令级并行的概念

- 控制相关并不是一个必须严格保持的关键属性。
- 对于正确地执行程序来说，必须保持的最关键的两个属性是：**数据流**和**异常行为**。
 - 保持异常行为是指：无论怎么改变指令的执行顺序，都不能改变程序中异常的发生情况。
 - 即原来程序中是怎么发生的，改变执行顺序后还是怎么发生。
 - 为了提升性能，可以弱化为：指令执行顺序的改变不能导致程序中发生新的异常。
 - 如果我们能做到保持程序的数据相关和控制相关，就能保持程序的数据流和异常行为(从而保证正确)。

指令级并行的概念

- 举例说明

DADDU	R2, R3, R4
BEQZ	R2, L1
LW	R1, 0 (R2)

必须保持对R2的数据相关，否则会影响结果！


L1 :

- 数据流：指数据值从其产生者指令到其消费者指令的实际流动。
 - 分支指令使得数据流具有动态性，因为它使得给定指令的数据可以有多个来源。
 - 仅仅保持数据相关性是不够的，只有再加上保持控制相关，才能够保持程序顺序。

指令级并行的概念

- 举例：

```
DADDU    R1, R2, R3
BEQZ     R4, L1
DSUBU    R1, R5, R6
L1 : ...
OR       R7, R1, R8
```



R1的值依赖于数据流
(之前多条指令)，
具体由分支指令决定！

- 但是，有时不遵守控制相关既不影响异常行为，也不改变数据流。
 - 这时候可以大胆地进行指令调度，把失败分支中的指令调度到分支指令之前。

指令级并行的概念

— 举例：

	DADDU	R1, R2, R3
	BEQZ	R12, Skipnext
	DSUBU	R4, R5, R6
	DADDU	R5, R4, R9
skip:	OR	R7, R8, R9

- 假设 skip后R4不再使用，且DSUBU不产生异常，则可以将DSUBU调度到分支之前。

本章内容概要

4.1 指令级并行的概念

4.2 指令的动态调度

4.3 动态分支预测技术

4.4 多指令流出技术

4.5 循环展开和指令调度

循环展开和指令调度基本方法

- 充分开发指令之间存在的并行性，找出不相关的指令序列，让它们在流水线上重叠并行执行。
- 增加指令间并行性最简单和最常用的方法
 - 开发循环级并行性——循环的不同迭代之间存在的并行性。
 - 在把循环展开后，通过寄存器重命名和指令调度来开发更多的并行性。

循环展开和指令调度

- 编译器完成这种指令调度的能力受限于两个特性：
 - 程序固有的指令级并行性；
 - 流水线功能部件的执行延迟。
- 本小节中，我们使用的浮点流水线延迟如下：

产生结果的指令	使用结果的指令	延迟（时钟周期数）
浮点计算	另一个浮点计算	3
浮点计算	浮点store（S.D）	2
浮点load（L.D）	浮点计算	1
浮点load（L.D）	浮点store（S.D）	0

循环展开和指令调度

- 假设采用MIPS的5段流水线：
 - 分支的延迟：1个时钟周期。
 - 整数load指令的延迟：1个时钟周期。
 - 整数运算部件是全流水或者重复设置了足够的份数。

循环展开和指令调度

- **例4.6** 对于下面的源代码，转换成MIPS汇编语言，在不进行指令调度和进行指令调度两种情况下，分析其代码一次循环所需的执行时间。

```
for (i=1; i<=1000; i++)
```

```
    x[i] = x[i] + s;
```

解： 把该程序翻译成MIPS汇编语言代码：

假设：

- ① R1的初值是指向第一个元素
- ② 8 (R2) 指向最后一个元素。
- ③ 浮点寄存器F2：用于保存常数s。

循环展开和指令调度

```
Loop: L. D      F0, 0 (R1)
      ADD. D    F4, F0, F2
      S. D      F4, 0 (R1)
      DADDIU    R1, R1, #-8
      BNE       R1, R2, Loop
```

其中：

整数寄存器R1：指向向量中的当前元素。（初值为向量中最高端元素的地址）

循环展开和指令调度

- 不进行指令调度的情况下，程序的实际执行情况：

	指令流出时钟
Loop: L.D F0, 0(R1)	1
(空转)	2
ADD.D F4, F0, F2	3
(空转)	4
(空转)	5
S.D F4, 0(R1)	6
DADDIU R1, R1, # -8	7
(空转)	8
BNE R1, R2, Loop	9
(空转)	10

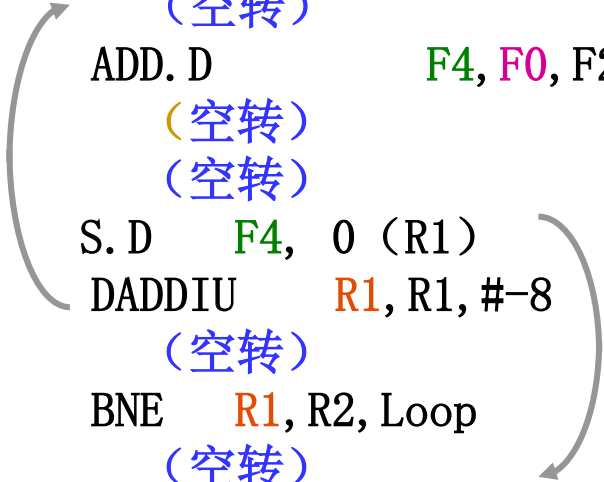
- 每个元素的操作需要10个时钟周期，其中5个是空转周期。

循环展开和指令调度

- 如果进行如下的指令调度：
 - 把DADDIU指令调度到L.D指令和ADD.D指令之间的“空转”拍；
 - 把S.D指令放到了分支指令的延迟槽中；
 - 对存储器地址偏移量进行调整；

循环展开和指令调度

Loop: L. D F0, 0 (R1)
 (空转)
 ADD. D F4, F0, F2
 (空转)
 (空转)
 S. D F4, 0 (R1)
 DADDIU R1, R1, #-8
 (空转)
 BNE R1, R2, Loop
 (空转)



Loop: L. D F0, 0(R1)
 DADDIU R1, R1, #-8
 ADD. D F4, F0, F2
 (空转)
 BNE R1, R2, Loop
 S. D F4, 8(R1)



循环展开和指令调度

指令流出时钟

Loop: L. D	F0, 0(R1)	1
	DADDIU R1, R1, #-8	2
	ADD. D F4, F0, F2	3
	(空转)	4
	BNE R1, Loop	5
	S. D F4, 8(R1)	6

- 一轮循环的操作时间从10个时钟周期减少到6个，其中5个周期是有指令执行的，1个为空转周期。

循环展开和指令调度

- 例子中的问题及解决方案

- 问题：只有L. D、ADD. D和S. D这3条指令是有效操作。
 - 占用3个时钟周期。
 - 而DADDIU、空转和BEN这3个时钟周期都是附加的循环控制开销。
- 解决方案：循环展开技术
 - 把循环体的代码复制多次并按顺序排列，然后相应调整循环的结束条件。
 - 这给编译器进行指令调度带来了更大的空间。

循环展开和指令调度

- **例4.7:** 将上述例子中的循环展开4次得到4个循环体，然后对展开后的指令序列在不调度和调度两种情况下，分析代码的性能。假定R1的初值为32的倍数，即循环次数为4的倍数。消除冗余的指令，并且不要重复使用寄存器。

解: 无需在循环体后面增加补偿代码，分配寄存器（不重复使用寄存器）如下：

- F0、F4：用于展开后的第1个循环体
- F2：保存常数s
- F6、F8：第2个循环体
- F10、F12：第3个循环体
- F14、F16：第4个循环体

- 展开后没有调度的代码如下：

指令流出时钟				指令流出时钟			
Loop:	L. D	F0, 0 (R1)	1	ADD. D	F12, F10, F2	15	
	(空转)		2	(空转)		16	
	ADD. D	F4, F0, F2	3	(空转)		17	
	(空转)		4	S. D	F12, -16 (R1)	18	
	(空转)		5	L. D	F14, -24 (R1)	19	
	S. D	F4, 0 (R1)	6	(空转)		20	
	L. D	F6, -8 (R1)	7	ADD. D	F16, F14, F2	21	
	(空转)		8	(空转)		22	
	ADD. D	F8, F6, F2	9	(空转)		23	
	(空转)		10	S. D	F16, -24 (R1)	24	
	(空转)		11	DADDIU	R1, R1, #-32	25	
	S. D	F8, -8 (R1)	12	(空转)		26	
	L. D	F10, -16 (R1)	13	BNE	R1, R2, Loop	27	
	(空转)		14	(空转)		28	

循环展开和指令调度

- 结果分析:

- 这个循环每遍共使用了28个时钟周期。
- 有4个循环体，完成4个元素的操作。平均每个元素

In conclusion, it's not efficient.

- 间：从减少循环控制的开销中获得的
- 在整个展开后的循环中，实际指令只有14条，其他14个周期都是空转。

- 对指令序列进行优化调度，以减少空转周期：

			指令流出时钟
Loop:	L. D	F0, 0(R1)	1
	L. D	F6, -8(R1)	2
	L. D	F10, -16(R1)	3
	L. D	F14, -24(R1)	4
	ADD. D	F4, F0, F2	5
	ADD. D	F8, F6, F2	6
	ADD. D	F12, F10, F2	7
	ADD. D	F16, F14, F2	8
	S. D	F4, 0(R1)	9
	S. D	F8, -8(R1)	10
	DADDIU	R1, R1, #-32	12
	S. D	F12, 16(R1)	11
	BNE	R1, R2, Loop	13
	S. D	F16, 8(R1)	14

循环展开和指令调度

- 结果分析:

- 没有数据相关引起的空转等待。
 - 整个循环仅仅使用了14个时钟周期。
 - 平均每个元素的操作使用 $14/4=3.5$ 个时钟周期。
- 结论: 通过循环展开、寄存器重命名和指令调度, 可以有效地开发出指令级并行。

循环展开和指令调度

- 循环展开和指令调度时要注意以下几个方面：

- 保证正确性。

- 在循环展开和调度过程中尤其要注意两个地方的正确性：
循环控制和操作数偏移量的修改。

- 注意有效性。

- 只有能够找到不同循环体之间的无关性，才能有效地使用循环展开。

- 使用不同的寄存器，否则可能导致新的冲突。

- 注意对存储器数据的相关性分析。

- 注意新的相关性。

- 由于原循环不同次的迭代在展开后都到了同一次循环体中，因此可能带来新的相关性。

本章内容概要

4.1 指令级并行的概念

4.2 指令的动态调度

4.3 动态分支预测技术

4.4 多指令流出技术

4.5 循环展开和指令调度

指令的动态调度

- 静态调度

- 依靠编译器对代码进行静态调度，以减少相关和冲突。
- 它不是在程序执行的过程中，而是在编译期间进行代码调度和优化。
- 通过把相关的指令拉开距离来减少可能产生的停顿。

- 动态调度

- 在程序的执行过程中，依靠专门硬件对代码进行调度，减少数据相关导致的停顿。

指令的动态调度

- **动态调度的优点：**

- 能够处理一些在编译时情况不明的相关（比如涉及到存储器访问的相关），并简化了编译器；
- 能够使本来是面向某一流水线优化编译的代码在其他的流水线（动态调度）上也能高效地执行。

- **动态调度的缺点：**

- 以硬件复杂性的显著增加为代价

指令的动态调度

- 到目前为止我们所使用流水线的最大的局限性

- 指令必须按序流出和执行
- 考虑下面一段代码：

DIV.D F4, F0, F2

SUB.D F10, F4, F6

ADD.D F12, F6, F14

- SUB.D指令与DIV.D指令关于F4相关，导致流水线停顿；
- ADD.D指令与前面的指令都没有关系，但其处理也因此受阻；

指令的动态调度

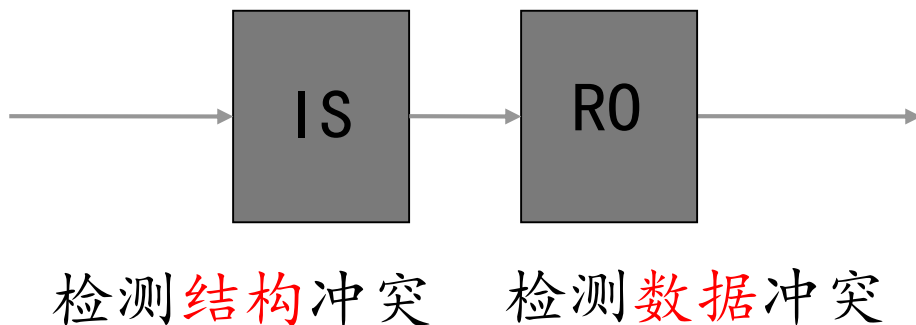
- 在前面的基本流水线中：



- 一旦一条指令受阻，其后的指令都将停顿，这种处理策略会限制流水线的性能。

指令的动态调度

- 为了能够乱序执行，可以将5段流水线的译码阶段再分为两个阶段：
 - 流出 (Issue, IS) : 指令译码，检查是否存在结构冲突。(in-order issue)
 - 读操作数 (Read Operands, RO) : 等待数据冲突消失，然后读操作数。



指令的动态调度

- 在前述5段流水线中，是不会发生WAR冲突和WAW冲突的。但乱序执行就使得它们可能会发生。

– 例如，考虑下面的代码

	DIV. D	F10, F0, F2	} 存在输出相关
存在反相关 {	SUB. D	F10, F4, F6	
	ADD. D	F6, F8, F14	

- Tomasulo算法可以通过使用寄存器重命名来消除上面的相关，从而提升指令执行效率。
 - 后续会详细介绍该算法

指令的动态调度

- I/O device request
- Invoking an operating system service from a user program
- Tracing instruction execution
- Breakpoint (programmer-requested interrupt)
- Integer arithmetic overflow
- FP arithmetic anomaly
- Page fault (not in main memory)
- Misaligned memory accesses (if alignment is required)
- Memory protection violation
- Using an undefined or unimplemented instruction
- Hardware malfunctions
- Power failure

指令的动态调度

- 动态调度让指令乱序完成，会大大增加异常处理的难度。
- 动态调度需要保持正确的异常行为：
 - 对于一条会产生异常的指令来说，只有当处理机确切地知道该指令将被执行后，才允许它产生异常。
 - 即使保持了正确的异常行为，动态调度处理机仍可能发生不精确异常。

指令的动态调度

- 不精确异常：
 - 当执行指令 i 导致发生异常时，处理机的现场（状态）与严格按程序顺序执行时指令 i 的现场不同。
 - 当指令 i 发生异常时：
 - 流水线可能已经执行完按程序顺序是位于指令 i 之后的指令；
 - 流水线可能还没完成按程序顺序是指令 i 之前的指令。
 - 不精确异常使得在异常处理后难以接着继续执行程序。
- 精确异常：
 - 如果发生异常时，处理机的现场跟严格按程序顺序执行时指令 i 的现场相同。

记分板调度算法

- 例：数据先读后写（WAR）相关引起的阻塞代码序列：

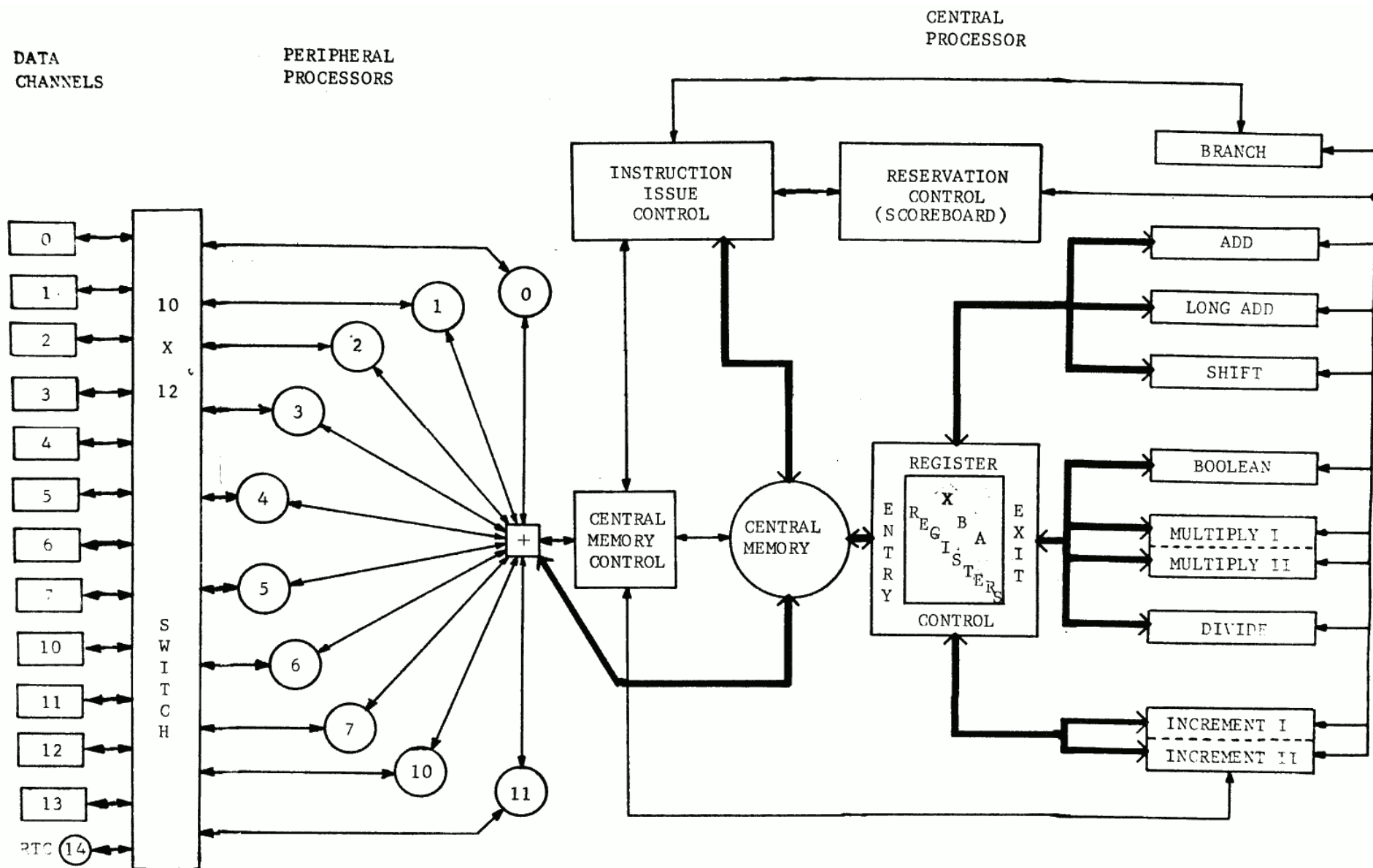
```
DIVD    F0 , F2 , F4  
ADDD    F10 , F0 , F8  
SUBD    F8 , F8 , F14
```

- 指令乱序执行时就会出现先读后写相关。
- 记分板调度算法的目标：
 - 在资源充足时，尽可能早地执行没有数据阻塞的指令，达到每个时钟周期执行一条指令。

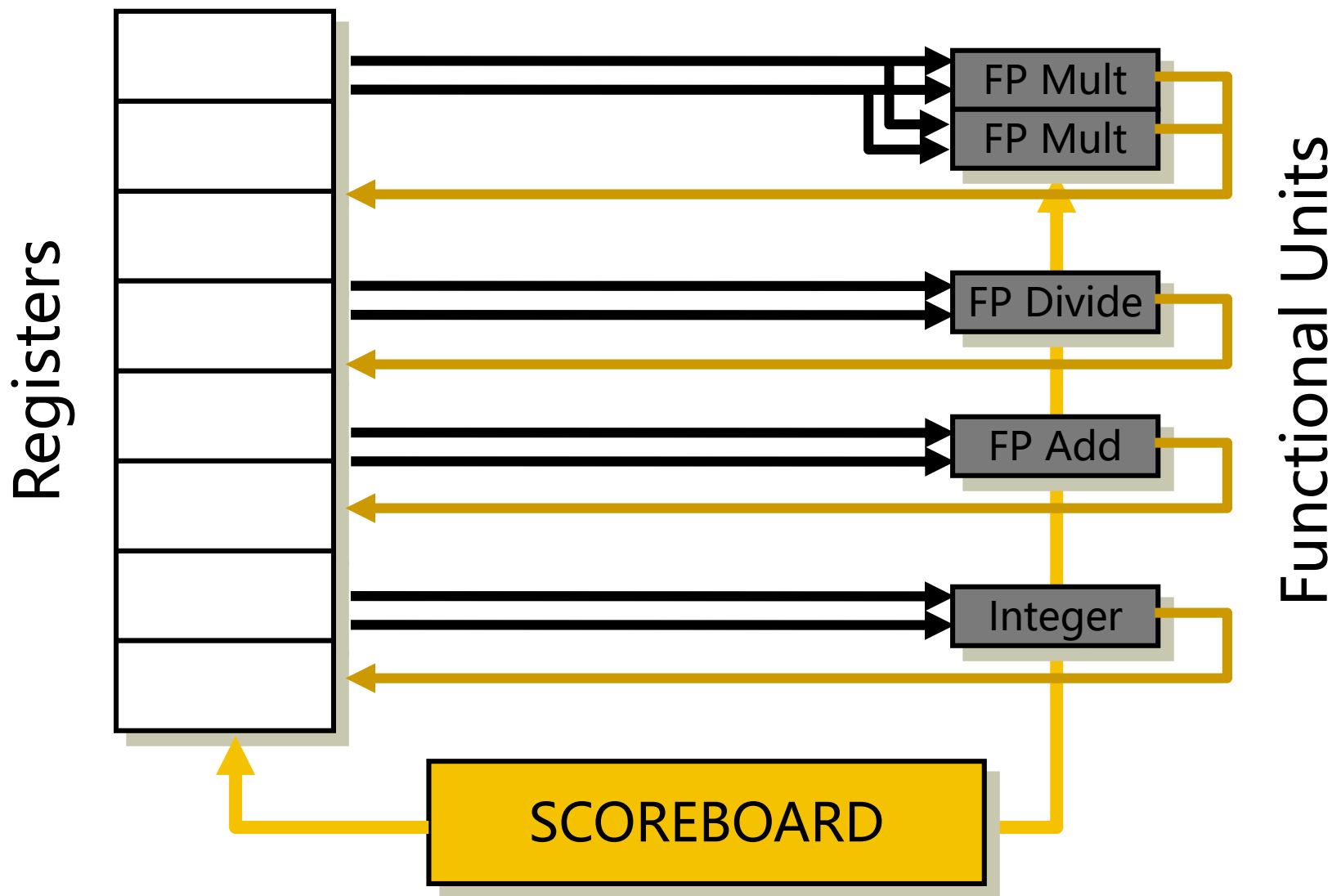
CDC 6600



CDC 6600 结构简图



记分板体系结构



记分板的含义

- 指令乱序执行 => 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
 - 只在读操作数段才读取寄存器
- 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; (消除瓶颈)
- 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

1. Issue—decode instructions & check for structural hazards.

If ❶ a functional unit for the instruction is **free** and ❷ 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 (**in-order issue**).

2. Read operands—wait until no data hazards, then read operands.

❸ 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**.

记分板控制的四级

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.

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.

④ If **WAR**, then it stalls the instruction.

Example:

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

CDC 6600 scoreboard would stall SUBD until ADDD reads operands

记分板需要记录的信息

1. Instruction status

- which of 4 steps the instruction is in

2. Functional unit status—Indicates the state of the functional unit (FU).

Busy — Indicates whether the unit is busy or not

Op — Operation to perform in the unit (e.g., + or −)

Fi — Destination register

Fj, Fk — Source-register numbers

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

Rj, Rk— Flags indicating when Fj, Fk are ready

3. Register result status

- Indicates which functional unit will write each register, if one exists.
Blank when no pending instructions will write that register

记分板示例 第0周期

<u>Instruction status</u>				<i>Read</i>	<i>Exec</i>	<i>Write</i>	
Instruction	<i>j</i>	<i>k</i>		<i>Issue</i>	<i>opera</i>	<i>compl</i>	<i>Result</i>
LD	F6	34+	R2				
LD	F2	45+	R3				
MUL	F0	F2	F4				
SUB	F8	F6	F2				
DIV	F10	F0	F6				
ADD	F6	F8	F2				

<u>Functional unit status</u>			<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU for</i>	<i>FU for</i>	<i>Fj?</i>	<i>Fk?</i>	
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

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

Execution time:

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

记分板示例 第1周期

Instruction status: Read Execu Write
Instruction *j* *k* Issue operat compl Result

LD	F6	34+	R2	1
LD	F2	45+	R3	
MUL	F0	F2	F4	
SUB	F8	F6	F2	
DIV	F10	F0	F6	
ADD	F6	F8	F2	

Functional unit status *dest* *S1* *S2* *FU for* *FU for* *Fj?* *Fk?*

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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				Integer					

记分板示例 第2个周期

Instruction status Read Execu Write
Instruction *j* *k* Issue operat compl Result

LD F6 34+ R2

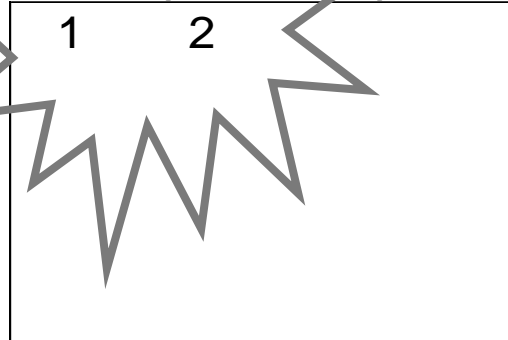
LD F2 45+ R3

MUL F0 F2 F4

SUB F8 F6 F2

DIV F10 F0 F6

ADD F6 F8 F2



• Issue 2nd LD?

Functional unit status

Time Name

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

Register result status

Clock

2

F0 *F2* *F4* *F6* *F8* *F10* *F12* ... *F30*

FU

Integer

记分板示例 第3周期

Instruction status

Instruction	j	k
LD	F6	34+ R2
LD	F2	45+ R3
MULTDF0	F2	F4
SUBD	F8	F6 F2
DIVD	F10	F0 F6
ADDD	F6	F8 F2

Functional unit status

Time	Name	Busy	Op	dest	S1	S2	FU for j	FU for k	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	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
3	FU Integer								

记分板示例 第4周期

<u>Instruction status</u>				<i>Read</i>	<i>Execu</i>	<i>Write</i>
Instruction	j	k		<i>Issue operat</i>	<i>compl</i>	<i>Result</i>
LD F6 34+ R2				1	2	3
LD F2 45+ R3						
MUL F0 F2 F4						
SUB F8 F6 F2						
DIV F10 F0 F6						
ADD F6 F8 F2						

<u>Functional unit status</u>		<i>dest</i>	<i>.S1</i>	<i>.S2</i>	<i>FU for Fi?</i>	<i>FU for Fj?</i>	<i>Fk?</i>
<i>Time Name</i>	<i>Busy Op</i>	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>
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6周期

<u>Instruction status</u>				<i>Read</i>	<i>Execu</i>	<i>Write</i>
Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>operat</i>	<i>compl</i>	<i>Result</i>
LD	F6	34+	R2	1	2	3
LD	F2	45+	R3	5	6	
MUL	F0	F2	F4	6		
SUB	F8	F6	F2			
DIV	F10	F0	F6			
ADD	F6	F8	F2			

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

Functional unit status			dest	S1	S2	FU for	FU for	Fj?	Fk?
Time	Name	Busy Op	Fi	Fj	Fk	Qj	Qk	Rj	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		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
Clock										
6	<i>FU</i>	Mult	Integer							

记分板示例 第7周期

Instruction status				Read	Execu	Write				
Instruction	<i>j</i>	<i>k</i>		Issue	operat	compl	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						
DIV	F10	F0	F6							
ADD	F6	F8	F2							
Functional unit status				dest	<i>S1</i>	<i>S2</i>	<i>FU for</i>	<i>FU for</i>	<i>Fj?</i>	<i>Fk?</i>
Time	Name	Busy	Op	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	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		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
7	<i>FU</i>	Mult1	Integer			Add				

- Read multiply operands?

记分板示例 第8a周期(前半周期)

Instruction status

Instruction *j* *k* *Read* *Execu* *Write*
Issue *operat* *compl* *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			
DIV	F10	F0	F6	8			
ADD	F6	F8	F2				

Functional unit status

Time *Name* *Busy* *Op* *dest* *S1* *S2* *FU for* *FU for* *Fj?* *Fk?*
Fi *Fj* *Fk* *Qj* *Qk* *Rj* *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周期(后半周期)

Instruction status Read Execu Write
Instruction j k Issue opera compl 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			
DIV	F10	F0	F6	8			
ADD	F6	F8	F2				

Functional unit status dest S1 S2 FU for FU for Fj? Fk?
Time Name Busy Op Fi Fj Fk Qj Qk Rj 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9周期

Instruction status				Read	Execu	Write
Instruction	j	k		Issue	operat	compl 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	
DIV F10 F0 F6				8		
ADD F6 F8 F2						

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

• Read operands for MULT & SUBD? Issue ADDD?

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			

记分板示例 第10周期

Instruction status

Instruction *j* *k* *Issue* *operat* *compl* *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		
DIV	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* *S1* *S2* *FU for* *FU for* *Fj?* *Fk?*
Fi *Fj* *Fk* *Qj* *Qk* *Rj* *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11周期

Instruction status				Read	Execu	Write
Instruction	j	k	Issue	operat	compl	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	
DIV F10 F0 F6			8			
ADD F6 F8 F2						

ADD: 2 cycles

Mult: 10 cycles

Divd: 40 cycles

<u>Functional unit status</u>			<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU for</i>	<i>FU for</i>	<i>Fj?</i>	<i>Fk?</i>	
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	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12周期

Instruction status

Instruction *j* *k* *Issue* *operat* *compl* *Result*

LD F6 34+ R2

LD F2 45+ R3

MUL F0 F2 F4

SUB F8 F6 F2

DIV F10 F0 F6

ADD F6 F8 F2

Read *Execu* *Write*

1	2	3	4
5	6	7	8
6	9		
7	9	11	12
8			

• Read operands for DIVD?

Functional unit status

Time *Name*

Busy *Op* *dest* *S1* *S2* *FU for* *FU for* *Fj?* *Fk?*
Fi *Fj* *Fk* *Qj* *Qk* *Rj* *Rk*

Integer

7 Mult1

Mult2

Add

Divide

No								
Yes	Mult	F0	F2	F4			No	No
No								
No								
Yes	Div	F10	F0	F6	Mult1		No	Yes

Register result status

Clock

F0 F2 F4 F6 F8 F10 F12 ... F30

12

FU

Mult1

Divide

记分板示例 第13周期

Instruction status				Read	Execu	Write
Instruction	j	k		Issue	operat	compl 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
DIV F10 F0 F6				8		
ADD F6 F8 F2				13		

<u>Functional unit status</u>			<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU for</i>	<i>FU for</i>	<i>Fj?</i>	<i>Fk?</i>	
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	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14周期

Instruction status				Read	Execu	Write				
Instruction	j	k		Issue	operat	compl	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			
DIV	F10	F0	F6	8						
ADD	F6	F8	F2	13	14					
Functional unit status				dest	S1	S2	FU for	FU for	Fj?	Fk?
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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15周期

Instruction status				Read Execut Write			
Instruction	j	k		Issue	operat	compl	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
DIV F10 F0 F6				8			
ADD F6 F8 F2				13	14		

Functional unit status			dest	S1	S2	FU for	FU for	Fj?	Fk?	
Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	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		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock										
15	FU	Mult1			Add		Divide			

记分板示例 第16周期

Instruction status				Read Execu Write			
Instruction	<i>j</i>	<i>k</i>		Issue	operat	compl	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
DIV	F10	F0	F6	8			
ADD	F6	F8	F2	13	14	16	

Functional unit status			<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU for</i>	<i>FU for</i>	<i>Fj?</i>	<i>Fk?</i>	
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	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		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
Clock										
16	<i>FU</i>	Mult1			Add		Divide			

记分板示例 第17周期

Instruction status

Instruction *j* *k* *Read* *Execu* *Write*
Issue *operat* *compl* *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
DIV	F10	F0	F6	8			
ADD	F6	F8	F2	13	14	16	

• Write result of ADDD?

Functional unit status

Time *Name* *Busy* *Op* *dest* *S1* *S2* *FU for* *FU for* *Fj?* *Fk?*
Fi *Fj* *Fk* *Qj* *Qk* *Rj* *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	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
17	<i>FU</i>	Mult1		Add		Divide			

记分板示例 第18周期

Instruction status *Read Execu Write*
Instruction j k Issue operat compl 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
DIV	F10	F0	F6	8			
ADD	F6	F8	F2	13	14	16	

Functional unit status *dest S1 S2 FU for FU for Fj? Fk?*
Time Name Busy Op Fi Fj Fk Qj Qk Rj 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19周期

Instruction status				Read Execu Write			
Instruction	<i>j</i>	<i>k</i>		Issue	operat	compl	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
DIV	F10	F0	F6	8			
ADD	F6	F8	F2	13	14	16	

Functional unit status			<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU for</i>	<i>FU for</i>	<i>Fj?</i>	<i>Fk?</i>	
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	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		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
Clock										
19	<i>FU</i>	Mult1			Add		Divide			

记分板示例 第20周期

Instruction status				Read Execut Write			
Instruction	<i>j</i>	<i>k</i>		Issue	operat	compl	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
DIV	F10	F0	F6	8			
ADD	F6	F8	F2	13	14	16	

Functional unit status			<i>dest</i>	<i>S1</i>	<i>S2</i>	<i>FU for</i>	<i>FU for</i>	<i>Fj?</i>	<i>Fk?</i>	
<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	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		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
Clock										
20	FU				Add		Divide			

记分板示例 第21周期

Instruction status

Instruction *j* *k* *Issue* *operat* *compl* *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
DIV	F10	F0	F6	8	21		
ADD	F6	F8	F2	13	14	16	

Functional unit status

Time *Name* *Busy* *Op* *dest* *S1* *S2* *FU for* *FU for* *Fj?* *Fk?*
Fi *Fj* *Fk* *Qj* *Qk* *Rj* *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22周期

<u>Instruction status</u>				<i>Read</i>	<i>Execu</i>	<i>Write</i>
Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>operat</i>	<i>compl</i>	<i>Result</i>
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
DIV	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			dest	S1	S2	FU for	FU for	Fj?	Fk?
Time	Name	Busy Op	Fi	Fj	Fk	Qj	Qk	Rj	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	<div style="display: flex; align-items: center;"> <div style="border-right: 1px solid black; padding-right: 5px;">FU</div> <div style="padding-left: 5px; color: green;">Divide</div> </div>								

记分板示例 第61周期

Instruction status				<i>Read Execu Write</i>			
Instruction	<i>j</i>	<i>k</i>	<i>Issue</i>	<i>operat</i>	<i>compl</i>	<i>Result</i>	
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
DIV	F10	F0	F6	8	21	61	
ADD	F6	F8	F2	13	14	16	22

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

Register result status		<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	<i>...</i>	<i>F30</i>
Clock	<i>FU</i>	Divide								
61										

记分板示例 第62周期

Instruction status

Instruction *j* *k* *Issue* *operat* *compl* *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
DIV	F10	F0	F6	8	21	61	62
ADD	F6	F8	F2	13	14	16	22

Functional unit status

Time *Name* *Busy* *Op* *dest* *S1* *S2* *FU for* *FU for* *Fj?* *Fk?*
Fi *Fj* *Fk* *Qj* *Qk* *Rj* *Rk*

Integer	No
Mult1	No
Mult2	No
Add	No
0 Divide	No

Register result status

Clock	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
62	<i>FU</i>								

CDC 6600 的记分板

- 6600记分板的局限性：
 - 没有前递/旁路(forwarding)硬件；
 - 指令调度局限于基本块内(指令窗口小)；
 - 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；
 - 防止WAW冒险；