计算机组织与体系结构

第十六讲

计算机科学与技术学院
舒燕君

Recap

- 流水线的实例（**MIPS R4000**）
  - ✓ 整型流水线（深流水、**8**段）
  - ✓ 浮点流水线（多功能非线性流水线）
  - ✓ 流水线性能分析
- 向量处理机 **Cray-I**
  - ✓ 性能指标、基本结构
  - ✓ 链接技术
- 指令级并行
  - ✓ 指令级并行的基本概念
  - ✓ 基本块、循环级并行
  - ✓ 指令调度

调度代码

时钟	未调度	调度后
1	LD F0,0(R1)	LD F0,0(R1)
2	*Stall*	SUBI R1,R1,#8
3	ADDD F4,F0,F2	ADDD F4,F0,F2
4	*Stall*	*Stall*
5	*Stall*	BNEZ R1,Loop
6	SD 0(R1),F4	SD 8(R1),F4
7	SUBI R1,R1,#8	
8	*Stall*	
9	BNEZ R1,Loop	
10	*Stall*	

调度代码结果分析

- 每遍循环6个时钟节拍
- 和未调度代码比较，加速比 10/6=1.7
- 3个有效时钟节拍 (LD, ADDD, SD)
  - 节拍有效比率50%
  - 1拍空转 (占17%)
  - 2拍循环控制 (占33%)

- 如何进一步减少空转和循环控制占用的比率?

循环展开

- 如果将循环展开3次得到4个循环体(假设数组是4的倍数的元素)

```
Loop:   LD      F0,0(R1)
        ADDD    F4,F0,F2
        SD      0(R1),F4
        SUBI    R1,R1,#8
        LD      F0, 0(R1)
        ADDD    F4,F0,F2
        SD      0(R1), F4
        SUBI    R1,R1,#8
        LD      F0, 0(R1)
        ADDD    F4,F0,F2
        SD      0(R1), F4
        SUBI    R1,R1,#8
        LD      F0,0(R1)
        ADDD    F4,F0,F2
        SD      0(R1), F4
        SUBI    R1,R1,#8
        BNEZ    R1,Loop
```

- 编译器直接调整存储器访问的偏移量
- 大量关于F0,F4的名相关，指令难以调度

循环展开

- 如果将循环展开3次得到4个循环体(假设数组是4的倍数的元素)

```
Loop:   LD      F0,0(R1)
        ADDD    F4,F0,F2
        SD      0(R1),F4
        LD      F6,-8(R1)
        ADDD    F8,F6,F2
        SD      -8(R1), F8
        LD      F10,-16(R1)
        ADDD    F12,F10,F2
        SD      -16(R1), F12
        LD      F14,-24(R1)
        ADDD    F16,F14,F2
        SD      -24(R1), F16
        SUBI    R1,R1,#-32
        BNEZ    R1,Loop
```

循环无调度执行

时钟	无调度
1	LD F0,0(R1)
2	*stall*
3	ADDD F4,F0,F2
4	*stall*
5	*stall*
6	SD 0(R1),F4
7	SUBI R1,R1,#8
8	*stall*
9	BNEZ R1,LOOP
10	*stall*

执行时间分析

Loop:	LD	F0,0(R1)	1
	ADDD	F4,F0,F2	2,3
	SD	0(R1),F4	4,5,6
	LD	F6,-8(R1)	7
	ADDD	F8,F6,F2	8,9
	SD	-8(R1), F8	10,11,12
	LD	F10,-16(R1)	13
	ADDD	F12,F10,F2	14,15
	SD	-16(R1), F12	16,17,18
	LD	F14,-24(R1)	19
	ADDD	F16,F14,F2	20,21
	SD	-24(R1), F16	22,23,24
	SUBI	R1,R1,#-32	25
	BNEZ	R1,Loop	26,27
	stall		28

结果分析

- 循环使用28个时钟节拍
  - 14个空转节拍
    - 每个LD有1个空转个节拍 – 共4拍
    - 每个ADDD有2个空转节拍 - 共8拍
    - SUBI有1个空转节拍 - 共1拍
    - BRANCH有1个空转节拍 - 共1拍
  - 有14个指令流出节拍
- 平均每遍循环7个时钟节拍
- 共计使用9个寄存器

循环展开+指令调度

```
1.  Loop:   LD      F0,0(R1)
2.          LD      F6,-8(R1)
3.          LD      F10,-16(R1)
4.          LD      F14,-24(R1)
5.          ADDD    F4,F0,F2
6.          ADDD    F8,F6,F2
7.          ADDD    F12,F10,F2
8.          ADDD    F16,F14,F2
9.          SD      0(R1),F4
10.         SD      -8(R1),F8
11.         SUBI    R1,R1,#-32
12.         SD      16(R1),F12
13.         BNEZ    R1,R2,Loop
14.         SD      8(R1),F16
```

“循环展开+指令调度”结果分析

- 每遍循环时间下降为**14**个时钟节拍
  - 每个元素平均使用**3.5**个时钟节拍
- 比较
  - 没有循环展开，有指令调度
    - 每个元素**6**拍
  - 有循环展开，没有指令调度
    - 每个元素**7**拍

循环展开和指令调度的总结

- 保证正确性（循环控制和操作数偏移量的修改）
- 注意有效性（找到不同循环体之间的无关性）
- 使用大量不同的寄存器
- 减少循环控制中的测试指令和分支指令
- 注意分析**Load/Store**指令的内存地址
- 注意新的相关性

7.1 指令级并行的概念

7.1.1 循环展开调度的基本方法

7.1.2 相关性

7.1.2 相关性

- 相关与流水线冲突(Dependence and Hazard)
  - 相关：两条指令之间存在某种依赖关系。如果两条指令相关，则它们就有可能不能在流水线中重叠执行或者只能部分重叠执行。
  - 相关有3种类型
    - 数据相关（也称真数据相关、数据流相关）
    - 名相关（包括反相关、输出相关）
    - 控制相关

7.1.2 相关性

1. 数据相关、真数据相关、数据流相关

（Data dependence、Flow Dependence）

- 对于两条指令i（在前）和j（在后），如果下述条件之一成立，则称指令j与指令i数据相关。

  (1)指令j使用指令i产生的结果；

  (2)指令j与指令k数据相关，而指令k又与指令i数据相关。

- 数据相关具有传递性。

数据相关是两条指令存在先写后读的相关链

引起流水线的RAW冲突

例如：下面这一段代码存在数据相关。(数组增值)

```
Loop:  LD      F0, 0 (R1)      // F0为数组元素
       ADDD    F4, F0, F2      // 加上F2中的值
       SD      0 (R1), F4      // 保存结果
       SUBI    R1, R1, #8      // 数组指针递减8个字节
       BNEZ    R1, Loop        // 如果R1≠0，则分支
```

- 如果两条指令之间有真数据相关，那么它们就不能同时执行或是完全重叠执行
- 同时执行这些指令会造成正在流水的处理机检测到这种冲突并插入暂停，从而减少甚至取消指令之间的重叠
- 指令序列中存在的真数据相关反映出产生该指令序列的程序源代码的相关关系

相关性是程序的一个特性，
是否一个相关会导致实际的冲突，是否该冲突会造成暂停，
这是流水线结构的基本特性。

```
Loop:  LD      F0, 0 (R1)      // F0为数组元素
       ADDD    F4, F0, F2      // 加上F2中的值
       SD      0 (R1), F4      // 保存结果
       SUBI    R1, R1, #8      // 数组指针递减8个字节
       BNEZ    R1, Loop        // 如果R1≠0，则分支
```

- 如果分支检测移到了ID流水段，相关会造成1次暂停
- 如果分支检测仍在EX流水段，这个相关就不会引起暂停

- 数据相关性会限制可以开发的指令级并行性
- 解决方法
  - 硬件：采用互锁机制，一旦检测存在数据相关，则停止本指令的执行，并插入空转周期
  - 软件：使用编译器，在相关出插入空操作（空转周期），保证当前指令不会错误的使用一个尚未产生的数据。
- 数据相关的检测
  - 当数据的流动是经过寄存器时，相关的检测比较直观和容易
  - 当数据的流动是经过存储器时，检测比较复杂。
    - 相同形式的地址其有效地址未必相同；
    - 形式不同的地址其有效地址却可能相同。

7.1.2 相关性

2. 名相关（**Name dependence**）
   - 名：指令所访问的寄存器或存储器单元的名称。
   - 如果两条指令使用相同的名，但是它们之间并没有数据流动，则称这两条指令存在名相关。

– 指令j与指令i之间的名相关有两种：

（1）反相关（Anti dependence）：

如果指令j写的名与指令i读的名相同（引发流水线的先读后写WAR冲突），则称指令i和j发生了反相关。

指令j写的名＝指令i读的名

必须保证指令的原来顺序，以确保i能读到正确的值。

（2）输出相关（Output dependence）：

如果指令j和指令i写相同的名（引发流水线的写后写WAW冲突），则称指令i和j发生了输出相关。

指令j写的名＝指令i写的名

操作顺序必须得到保证，以使最后写入的值得以保存。

2. 名相关

- 两条指令之间并没有数据的传送。
- 如果一条指令中的名改变了，并不影响另外一条指令的执行。
- 换名技术
  - 通过改变指令中操作数的名来消除名相关。
  - 对于寄存器操作数进行换名称为寄存器换名（重命名，Renaming）。
  - 既可以用编译器静态实现，也可以用硬件动态完成。

循环展开中的换名

- 如果将循环展开3次得到4个循环体(假设数组是4的倍数的元素)

```
Loop:   LD    F0,0(R1)
        ADDD  F4,F0,F2
        SD    0(R1),F4
        LD    F6,-8(R1)
        ADDD  F8,F6,F2
        SD    -8(R1), F8
        LD    F10,-16(R1)
        ADDD  F12,F10,F2
        SD    -16(R1), F12
        LD    F14,-24(R1)
        ADDD  F16,F14,F2
        SD    -24(R1), F16
        SUBI   R1,R1,#-32
        BNEZ  R1,Loop
```

– 再考虑以下代码：

```
DIVD    F0，F2，F4
MULD    F6，F0，F8
SD      0（R1），F6
SUBD    F8，F10，F14
ADDD    F6，  F10，F8
```

输出相关（F6）导致WAW冲突

反相关（F8）导致WAR冲突

第二、三条指令中的F6是真数据相关
第四、五条指令中的F8是真数据相关

– 消除名相关

- 引入两个临时寄存器S和T
- 把这段代码改写为：

```
DIVD    F0，F2，F4
MULD    S，F0，F8
SD      0（R1），S
SUBD    T，F10，F14
ADDD    F6，F10，T
```

两个F6都换名为S

两个F8都换名为T

Dependence Types

Flow dependence

$r_3 \leftarrow r_1 \text{ op } r_2$

$r_5 \leftarrow r_3 \text{ op } r_4$

Read-after-Write (RAW)

Anti dependence

$r_3 \leftarrow r_1 \text{ op } r_2$

$r_1 \leftarrow r_4 \text{ op } r_5$

Write-after-Read (WAR)

Output-dependence

$r_3 \leftarrow r_1 \text{ op } r_2$

$r_5 \leftarrow r_3 \text{ op } r_4$

$r_3 \leftarrow r_6 \text{ op } r_7$

Write-after-Write (WAW)

7.1.2 相关性

3. 控制相关
   - 由分支指令引起的相关。
     - 为了保证程序应有的执行顺序，必须严格按控制关确定的顺序执行。
   - 典型的程序结构是“if-then”结构。
   - 例：

```
if p1 {
        S1;
    };
S;

if p2 {
        S2;
    };
```

– 控制相关有以下两个原则：

（1）与控制相关的指令不能移到分支指令之前，即控制有关的指令不能调度到分支指令的控制范围之外。

- 对于上述的例子，then 部分中的指令不能移到if语句之前

（2）与控制无关的指令不能移到分支指令之后，即控制无关的指令不能调度到分支指令的控制范围以内。

- 对于上述的例子，不能把S移到if语句的then 部分中。

```
if p1 {
        S1;
    };

S;

if p2 {
        S2;
    };
```

第七章 指令级并行

7.1　　指令级并行的概念

<u>7.2　　指令的动态调度</u>

7.3　　控制相关的动态解决技术

7.4　　多指令流出技术

7.2 指令的动态调度

7.2.1 动态调度的原理

7.2.2 动态调度算法之一：记分牌

7.2.3 动态调度算法之二：Tomasulo算法

7.2 指令的动态调度

- 编译器本质上通过对每个循环迭代中寄存器重命名来展开循环。
- 硬件也可通过寄存器重命名和乱序执行（**Out-of-Order， OoO**）来获得同样的效果。
- 动态调度
  - 记分牌
  - **Tomasulo’s**算法

冲突的检测和调度

- 如果存在数据相关，硬件检测机制会做如下的事情直到相关消除动态调度
  - ✓ 暂停指令
  - ✓ 停止取指令和发射指令
- 静态调度（开始于**60s**，流行于**80s**）
  - ✓ 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。

动态调度

- 动态调度的目的
  - 在程序执行的时候，解决 **WAW**、**WAR**、**RAW**带来的流水线冲突
- 优点：
  - 处理在编译的时候未知的相关，简化编译器
  - 在不同的流水线上都能有效的运行
- 缺点：
  - 很大地增加了硬件的复杂性
- 两个动态调度技术
  - 记分牌
  - **Tomasulo**算法

7.2.1 动态调度的原理

- 到目前为止我们所使用流水线的最大的局限性:
  - 指令是按序流出和按序执行的
  - 考虑下面一段代码:

    DIVD F4，F0，F2

    ADDD F10，F4，F6

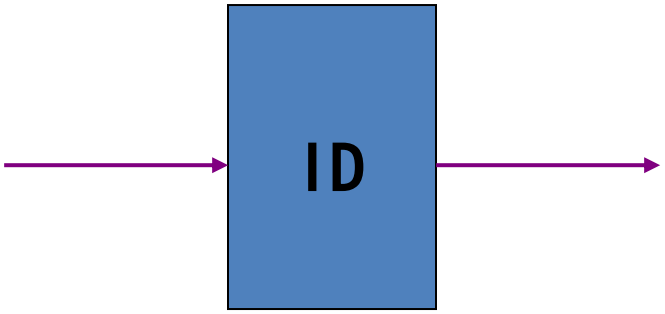
    SUBD F12，F6，F14

ADDD指令与**DIVD**指令关于**F4**相关，导致流水线停顿。

SUBD指令与流水线中的任何指令都没有关系，但也因此受阻。

7.2.1 动态调度的原理

在前面的基本流水线中：



检测结构冲突

检测数据冲突

一旦一条指令受阻，其后的指令都将停顿。

- 为了使上述指令序列中的**SUBD**指令能继续执行下去，必须把指令流出的工作拆分为两步：
  - 检测结构冲突
  - 等待数据冲突消失

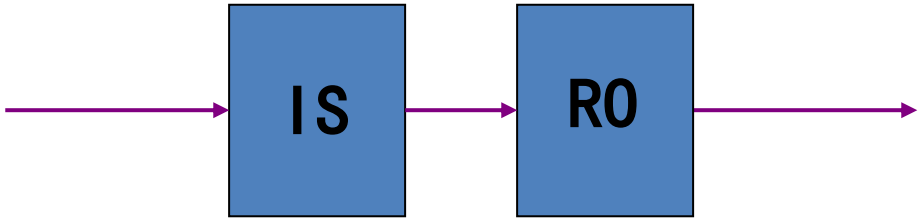
只要检测到没有结构冲突，就可以让指令发射，并且流出后的指令一旦其操作数就绪就可以立即执行。

乱序执行

- 指令的执行顺序与程序顺序不相同
- 指令的完成也是乱序完成的
  - 即指令的完成顺序与程序顺序不相同。

为了支持乱序执行，我们将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。（Out-of-Order execution）



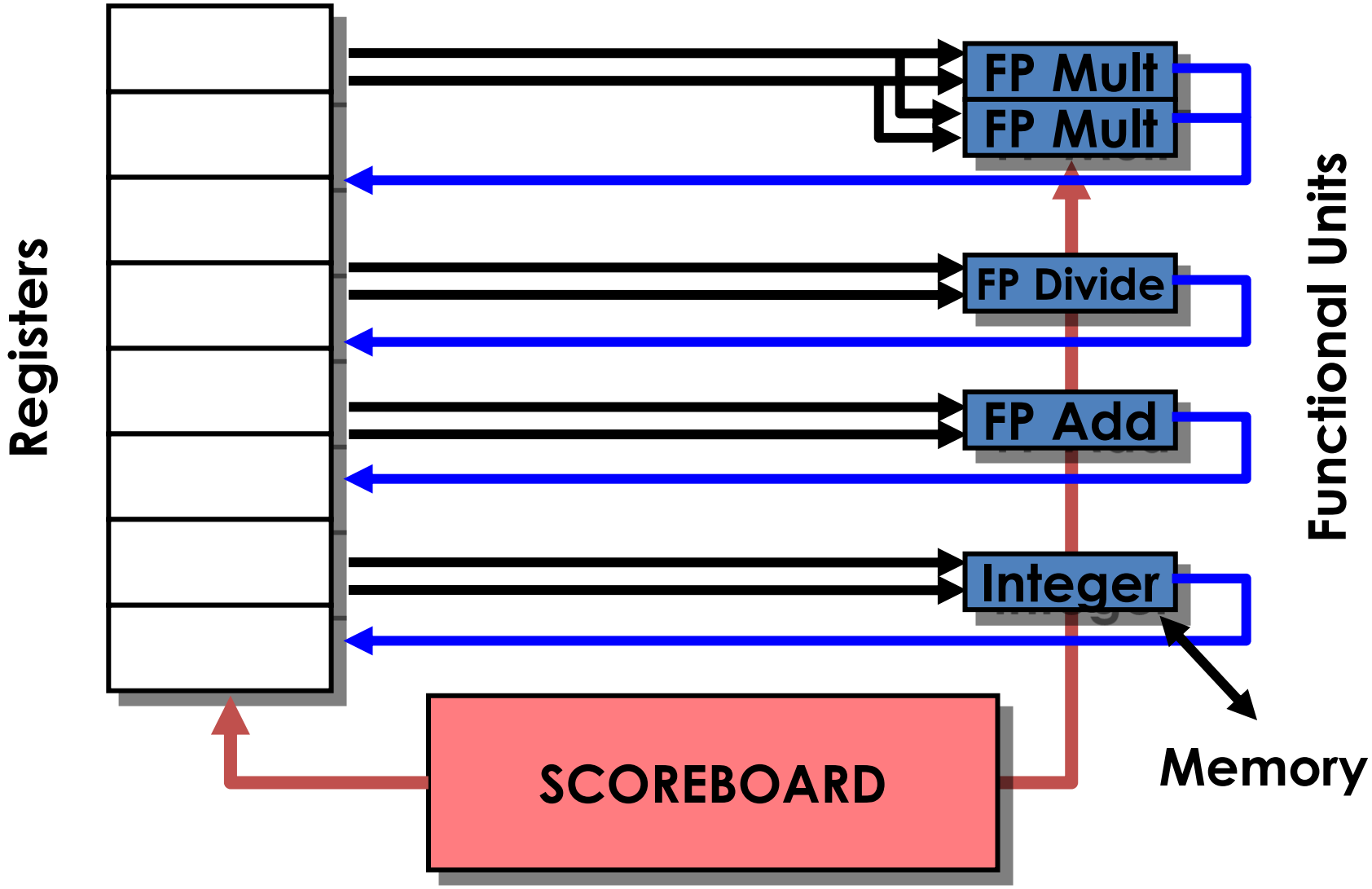
检测结构冲突 检测数据冲突

7.2.2 动态调度算法之一：记分牌

- 记分牌在1964年被Cray用于CDC 6600
- 记分牌允许指令乱序执行，前提是:
  - 充足的资源
  - 指令可以乱序执行
- 基本原理
  - 每条指令均经过记分牌，记录各指令间数据相关的信息，进行相关检测，控制指令的流出和执行
  - 如果记分牌判断出一条指令不能立即执行，它就检测硬件的变化从而决定何时能够执行
  - 集中控制寄存器与处理单元的数据传送，检测或消除数据相关性，加快程序执行速度。

CDC6600的照片

具有记分牌的MIPS处理器基本结构



记分牌执行过程

（1）流出（**Issue**）

- 本指令所需的功能部件有空闲
- 正在执行指令使用的目的寄存器与本指令不同
  - ✓ 保证没有**WAW**相关
- 如果存在结构冲突或**WAW**相关，本指令就不会流出，后面的指令也不会流出，直至阻塞消失

（2）读操作数（**Read operands**）

- 前面已流出的还在运行的指令不对本指令的源操作数寄存器进行写操作
- 一个正在工作的功能部件已经完成了对这个寄存器的写操作
  - ✓ 动态解决**RAW**相关

前面两步完成了原来ID段的功能

记分牌执行过程

（3）执行（Execution）

- 开始于取到操作数后
- 当结果产生后，修改记分牌
- FP流水部件会占用多个周期

（4）写结果（Write result）：检查WAR相关

出现以下情况时，不允许指令写结果：

- 前面的某条指令（已按顺序流出）还没有读取操作数
- 而且其中某个源操作数寄存器与本指令的目的寄存器相同

对应MIPS的EX段和WB段

记分牌结构

- 指令状态表：记录正在执行的各条指令进入到哪一个阶段
- 功能部件的状态表：
  - *Busy*：指示功能部件是否工作
  - *Op*：功能部件当前执行的操作
  - *Fi*：目的寄存器编号
  - *Fj*，*Fk*：源寄存器编号
  - *Qj*，*Qk*：向源寄存器*Fj*，*Fk*中写结果的功能部件
  - *Rj*，*Rk*：标示*Fj*，*Fk*是否就绪，是否已经被使用
- 结果寄存器状态表：记录写入寄存器的功能部件编号。

计分牌实例

Instruction status:

Instruction	*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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									

记分牌流水线控制

指令状态	进入条件	记分牌内容
流出	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;
读操作数	Rj and Rk	Rj← No; Rk← No Qj← 0; Qk←0;
执行	功能部件操作	
写结果	∀f((Fj(f)≠Fi(FU) or Rj(f)=No) and (Fk(f) ≠Fi(FU) or Rk(f)=No))	∀f(if Qj(f)=FU then Rj(f)← Yes); ∀f(if Qk(f)=FU then Rk(f)← Yes); Result(Fi(FU))← 0; Busy(FU)← No

数据结构：程序实例

- 指令序列

```
LD      F6, 34(R2)
LD      F2, 45(R3)
MULTD   F0, F2, F4
SUBD    F8, F6, F2
DIVD    F10, F0, F6
ADDD    F6, F8, F2
```

➢ 浮点部件执行时间延迟
  - 加（减）法2拍
  - 乘法 10 拍
  - 除法 40 拍

- 指令序列

```
LD      F6, 34(R2)
LD      F2, 45(R3)
MULTD   F0, F2, F4
SUBD    F8, F6, F2
DIVD    F10, F0, F6
ADDD    F6, F8, F2
```

- 代码段存在以下相关性：

（1）先写后读RAW相关：第二条LD到MULTD和SUBD之间， MULTD到DIVD之间，SUBD到ADDD之间；

（2）先读后写WAR相关：DIVD和ADDD之间，SUBD和ADDD之间；

（3）写后写WAW相关：第一条LD和ADDD之间；

（4）结构冲突：ADDD和SUBD指令关于浮点加法部件。

计分牌实例: Cycle 1

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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				

Functional unit status:

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

Register result status:

Clock
1

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

计分牌实例: Cycle 2

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2		
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

- Issue 2nd LD?

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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*
2	*FU*				Integer					

计分牌实例: Cycle 3

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	
LD	F2	45+	R3				
MULTD	F0	F2	F4				
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

- Issue MULT?

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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								

计分牌实例: Cycle 4

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
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*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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*				Integer					

计分牌实例: Cycle 5

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5			
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*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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							

计分牌实例: Cycle 6

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6		
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2				
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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							

计分牌实例: Cycle 7

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6				
ADDD	F6	F8	F2				

- Read multiply operands?

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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				

计分牌实例: Cycle 8a (前半拍)

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 8b (后半拍)

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6			
SUBD	F8	F6	F2	7			
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 9

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9		
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

- Issue ADDD?

Functional unit status:

Note Remaining

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 10

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9		
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 11

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 12

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2				

- Read operands for DIVD?

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 13

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13			

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 14

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14		

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 15

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14		

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 16

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 17

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

- Why not write result of ADD?

WAR Hazard!

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 18

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*Write Result*
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULTD	F0	F2	F4	6	9		
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 19

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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	
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 20

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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			
ADDD	F6	F8	F2	13	14	16	

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 21

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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		
ADDD	F6	F8	F2	13	14	16	

- WAR Hazard is now gone...

Functional unit status:

Time Name	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 22

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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		
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

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

Register result status:

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

计分牌实例: Cycle 61（跳过一些节拍）

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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	
ADDD	F6	F8	F2	13	14	16	22

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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			

计分牌实例: Cycle 62

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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*
62	*FU*									

计分牌最终状态

Instruction status:

Instruction		*j*	*k*	*Issue*	*Read Oper*	*Exec Comp*	*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

- In-order issue
- out-of-order execute
- out-of-order WR

Functional unit status:

Time	*Name*	*Busy*	*Op*	*dest Fi*	*S1 Fj*	*S2 Fk*	*FU Qj*	*FU 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*
62	*FU*									

开销和性能提升

- **CDC6600**

性能提升	编码方式
1.7	**FORTRAN**
2.5	手工汇编

- **Condition（CDC6600记分牌）**
  - 无软件流水调度
  - 无主存储器
  - 无**Cache**
- **CDC6600**上的记分牌的逻辑电路相当于一个功能部件，器件的耗费是非常低的，但增加了大量的连线。

- 记分牌的性能受限于以下几个方面：
  - 程序代码中可开发的并行性，即是否存在可以并行执行的不相关的指令。
  - 记分牌的容量。
    - 记分牌的容量决定了流水线能在多大范围内寻找不相关指令。流水线中可以同时容纳的指令数量称为指令窗口。
  - 功能部件的数目和种类。
    - 功能部件的总数决定了结构冲突的严重程度。
  - 反相关和输出相关。
    - 它们引起计分牌中更多的**WAR**和**WAW**冲突 。