

9. 考虑一个顺序流水线，忽略前端的取指和译码，处理器从发射到执行完成不同指令所需要的总周期数如下表所示。

指令类型	总周期数
内存加载	4
内存存储	2
整数运算	1
分支	2
浮点加法	3
浮点乘法	5
浮点除法	11

考虑如下的指令序列：

```

Loop:    fld    f2.0(a0)
        fdiv.d f8,f0,f2
        fmul.d f2,f6,f2
        fld    f4.0(a1)
        fadd.d f4,f0,f4
        fadd.d f10,f8,f2
        fsd   f1.0(a0)
        fsd   f4.0(a1)
        addi  a0,a0,8
        addi  a1,a1,8
        sub   x20,x4,a0
        bnez x20,Loop
    
```

- 假设一条单发射顺序流水线，在没有数据冲突或分支指令时，每个周期均会新发射一条指令（假定运算单元是充足的）。检测到数据冲突或分支指令时则会暂停发射，直到冲突指令执行完毕才会发射新的指令。则上述代码段的一次迭代需要多少个周期执行完成？
- 假设一条双发射顺序流水线，取指和译码的带宽足够、运算单元充足，且数据在两条流水线之间的传递是无延迟的，因此只有直数据冲突会导致流水线停顿。则上述代码段的一次迭代需要多少个周期执行完成？
- 调整指令的排列顺序，使得其在上述双发射流水线中完成一次迭代需要的周期数量减少。给出调整后的指令序列及一次迭代所需要的周期数。

(1)

1	fld	f2.0(a0)	RAW
2	fdiv.d	f8,f0,f2	RAW
3	fmul.d	f2,f6,f2	RAW
4	fld	f4.0(a1)	RAW
5	fadd.d	f4,f0,f4	RAW
6	fadd.d	f10,f8,f2	RAW
7	fsd	f1.0(a0)	RAW
8	fsd	f4.0(a1)	RAW
9	addi	a0,a0,8	RAW
10	addi	a1,a1,8	RAW
11	sub	x20,x4,a0	RAW
12	bnez	x20,Loop	RAW

指令下的括号内数字表示指令完成的周期

1 ... 5 -- 15 16 17 -- 21 22 -- 25 26 27 28 29 30 31
 (4) fdiv (15)

fmul (20) fdiv (25) fadd (29) fadd (30)

7 (21) addi (21) addi (22) sub (28) sub (29) bnez (31)

∴一个迭代需31个周期

(2)

1	fld	f2.0(a0)	
2	fdiv.d	f8,f0,f2	
3	fmul.d	f2,f6,f2	
4	fld	f4.0(a1)	RAW
5	fadd.d	f4,f0,f4	RAW
6	fadd.d	f10,f8,f2	RAW
7	fsd	f1.0(a0)	
8	fsd	f4.0(a1)	
9	addi	a0,a0,8	
10	addi	a1,a1,8	
11	sub	x20,x4,a0	
12	bnez	x20,Loop	RAW

1 ... 5 . 6 . . 10 . . 15 . 16 . . 18 . 19 . 20 . 21 . 22
 (4) fdiv (15) fdiv (18) fmul (9) fadd (12) fadd (18) fadd (20) fadd (21) sub (22) bnez (23)

∴一个迭代需23个周期

(3)

1	fld	f2.0(a0)	
2	fdiv.d	f8,f0,f2	
3	fmul.d	f2,f6,f2	
4	fld	f4.0(a1)	
5	fadd.d	f4,f0,f4	
6	fadd.d	f10,f8,f2	
7	fsd	f1.0(a0)	
8	fsd	f4.0(a1)	
9	addi	a0,a0,8	
10	addi	a1,a1,8	
11	sub	x20,x4,a0	
12	bnez	x20,Loop	

红色下划线指令为程序执行中的关键路径
 由于其数据依赖性，为保持程序原有语义不变
 其顺序需不变
 ⇒无法减少一次迭代所需周期数。

10. 考虑如下的代码片段：

```

Loop:    fld    f4.0(a0)
        fmul.d f2,f0,f2
        fdiv.d f8,f4,f2
        fld    f4.0(a1)
        fadd.d f6,f0,f4
        fsub.d f8,f8,f6
    
```

fsd f8.0(a1)

现将其进行简单的寄存器重命名，假定有 T0-T63 的临时寄存器池，且 T9 开始的寄存器可用于重命名。写出重命名后的指令序列。

fbl f4.0(a0) $T_9 - f_4$
 fmul.d T_9, f_0, f_2 $T_{10} - f_4$
 fdiv.d f_8, f_4, f_2
 fld $T_{10}, 0(a_1)$
 fadd.d f_6, f_0, T_{10}
 fsub.d f_8, f_8, f_6
 fsd $f_8, 0(a_1)$

11. 查阅资料，简述显式重命名和隐式重命名的区别、优缺点以及可能的实现方式。

(1) 显式重命名：将程序中的ISA寄存器映射到物理寄存器文件(PRF)中的实际寄存器。

同-ISA寄存器可被映射到不同的物理寄存器上，从而允许多个指令同时更改该寄存器的值

由于ISA寄存器和物理寄存器之间的映射在指令执行过程中是可见的，因此称为显式重命名

优点：重命名避免了数据相关性，允许乱序执行指令，提高了指令并行性，从而提高了吞吐量和性能；

缺点：需要额外的硬件支持，增加了成本和功耗；重命名器可能会引入新的延迟。

可能的实现方式：使用物理寄存器文件或基于标记的重命名器(Tagged Register Renaming)来实现

(2) 隐式重命名：ISA寄存器不显式地映射到物理寄存器。相反，每个指令都有一个重命名表来跟踪它所需的源和目标寄存器。指令执行时，操作数从重命名表中获取，结果写入重命名表中

优点：相比于显式重命名，隐式重命名实现更简单，成本更低；可以更好地支持动态指令集、超标量执行和多线程处理器

缺点：需要维护每个指令的重命名表，可能会增加复杂度和延迟；较难在硬件上实现乱序执行。

可能的实现方式：使用统一的重命名表(Unified Rename Table)或分离的源操作数表和目标操作数表

来实现