

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
整型运算	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: 1. fld f2,0(a0) |
      11. fdiv.d f8,f0,f2 |
      5. fmul.d f2,f6,f2 |
      4. fld f4,0(a1) |
      3. fadd.d f4,f0,f4 |
      3. fadd.d f10,f8,f2 |
      2. fsd f10,0(a0) |
      2. fsd f4,0(a1) |
      1. addi a0,a0,8 |
      1. addi a1,a1,8 |
      1. sub x20,x4,a0 |
      2. bnz x20,Loop |
  
```

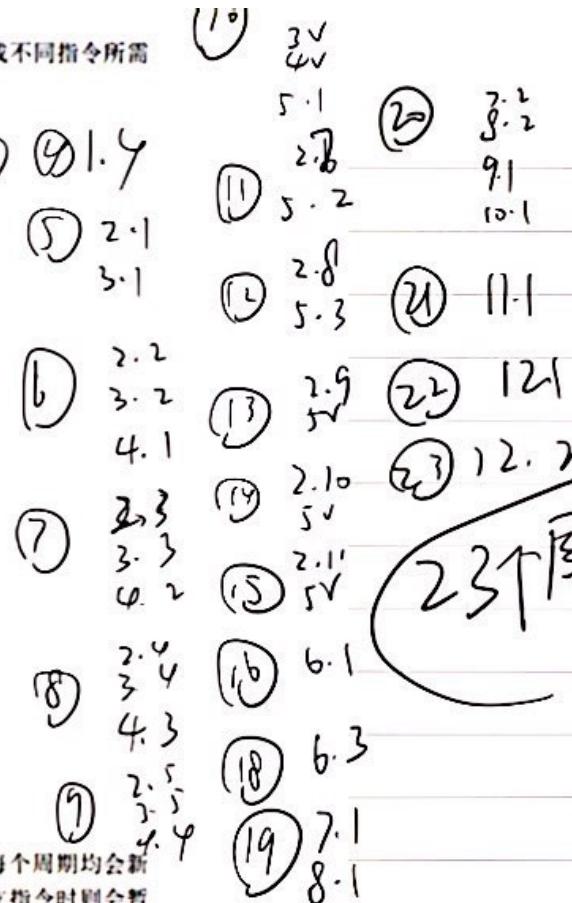
- 1) 假设一条单发射顺序流水线，在没有数据冲突或分支指令时，每个周期均会新发射一条指令（假设运算单元是充足的）。检测到数据冲突或分支指令时则会暂停发射，直到冲突指令执行完毕才会发射新的指令。则上述代码段的一次迭代需要多少个周期执行完成？

25个周期

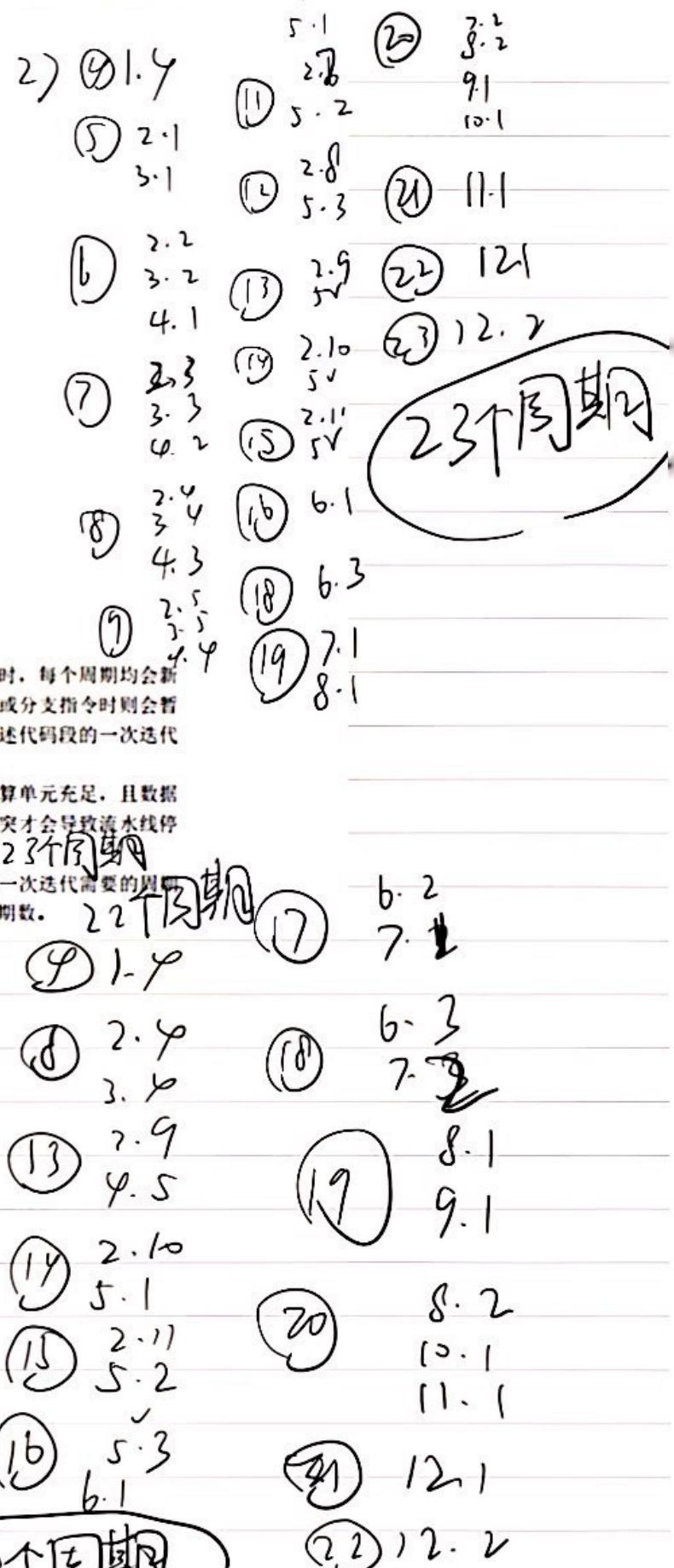
- 2) 假设一条双发射顺序流水线，取指和译码的带宽足够、运算单元充足，且数据在两条流水线之间的传递是无延迟的，因此只有真数据冲突才会导致流水线停顿。则上述代码段的一次迭代需要多少个周期执行完成？

23个周期

- 3) 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。



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



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
```

重命名

```
f1d  l9, 0(a0)
fmul.d T10, T11, T10
fdiv.d T12, T9, T10
fld  T13, 0(a1)
fadd.d T14, T11, T11
```

```
fsub.d T15, T12, T12
fsd  T16, da1)
```

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。

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：引入两种硬件：空闲列表_{PL}（用于维护物理寄存器的空闲状态信息），指示了当前物理寄存器堆中有哪些寄存器可用；重命名列表_{RL}（维护物理寄存器和ISA 寄存器之间的映射关系）。
- 优点：不需要在重排序缓冲区中创建大量存储临时值空位，提供更多物理寄存器；缺点：FL受稳定性限制，显式使用旧模式称的SQL查询将中断实现方式：查找 FL 并选择一个空闲的物理寄存器，将其和该指令要写入的ISA 寄存器进行绑定；指令源操作数查找 RT 以确定是否需要从某个被映射的寄存器取值；指令执行结束，结果写入对应物理寄存器。
- ② 隐式重命名：该方案物理寄存器数量与 ISA 规定保持一致，其中存放已经最终写回的指令结果。
- 优点：需要的物理寄存器个数少；缺点：读取数据的频率高，功耗较高。
- 实现方式：RDB 保存正在执行，尚未提交的指令结果，ARF 保存正在指令中即将写入寄存器中的值，不需要 free-list 来记录物理寄存器状态，指令被写入 RDB 即完成重命名。