3. 如果不同的虚拟地址页都映射到不同物理地址页，则下列存储-加载指令对是否可能发生数据依赖？如果可能，说明发生数据依赖的条件；如果不可能，说明理由。当不同的虚拟地址页可以被映射到相同的物理地址页，页大小为 4KB，则结论有什么不同？

1) sd a2,0(a0)
   ld a3,0(a1)
2) sd a2,0(a0)
   ld a3,4(a1)
3) sd a2,0(a0)
   ld a3,4096(a1)

1) 2) 3) 均不会发生数据依赖
∴ 不会映射到同一物理地址页

当不同的虚拟地址页可以被映射到同一物理地址页时，1) 2) 结论依然成立
但∵页大小为 4KB ∴当 0(a0) = 4096(a1) 时
3) 可能发生数据依赖

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

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？
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？
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。

1) 26周期

1 5 6 7 11 16 19 20 21 22 23 24 26

fld ✓

fdivd ✓

fmul.d ✓

fld ✓

fadd.d

fadd.d ✓

fsd

fsd

addi

addi

sub

bnz ✓

2) 23周期

1 5 6 10 16 19 20 21 22 23

fld ✓

fdiv.d ✓

fmul.d ✓

fld ✓

fadd.d

fadd.d

fsd

fsd

addi

addi

sub

bnz ✓

3) 23周期

1 5 6 9 16 19 20 21 22 23

fld ✓

fld ✓

fdiv.d

fmul.d

fadd.d

fsd

fadd.d ✓

addi

fsd

addi

sub

bnz ✓

调度如下

```
fld   f2, 0(a0)
fld   f4, 0(a1)
fdiv.d f8, f0, f2
fmul.d f2, f6, f2
fadd.d f4, f0, f4
fsd    f4, 0(a1)
fadd.d f10, f8, f2
```

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

不难发现

由于一连串数据冲突的存在

即使进行了调度

也无法消除冲突

```
addi  a1,a1,8
fsd   f10,0(a0)
addi  a0,a0,8
sub   x20,x4,a0
bnz   x20,Loop
```

10. 考虑如下的代码片段：

```
Loop: 1 fld      f4,0(a0)
      2 fmul.d   f2,f0,f2
      3 fdiv.d   f8,f4,f2
      4 fld      f4,0(a1)
      5 fadd.d   f6,f0,f4
      6 fsub.d   f8,f8,f6
      7 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。

冲突：

	1	2	3	4	5	6	7
1							
2	-						
3	RAW	RAW					
4	WAW	-	WAR				
5	-	-	-	RAW			
6	-	-	RAW	-	RAW		
7	-	-	-	-	-	RAW	

只需解决34的WAR冲突和14的WAW冲突

只需将f4重命名即可

① 初始：RT

f0	f1	f2	f3	f4	f5	f6	f7	f8
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8

FL：T_9~T_{63}

1~4：将f4映射为T_9，其他照RT

5~7：将f4映射为T_{10}，其他照旧

∴重命名后序列如下：

```
fld      T9, 0(a0)
fmul.d   T2, T0, T2
fdiv.d   T8, T9, T2
fld      T10, 0(a1)
fadd.d   T6, T0, T10
fsub.d   T8, T8, T6
fsd      T8, 0(a1)
```

```
1 fld      f4,0(a0)
2 fmul.d   f2,f0,f2
3 fdiv.d   f8,f4,f2
4 fld      f4,0(a1)
5 fadd.d   f6,f0,f4
6 fsub.d   f8,f8,f6
7 fsd      f8,0(a1)
```

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。

显式重命名
显式重命名通常需要硬件支持，在指令执行阶段中，将指令中的源寄存器和目标寄存器用一个重命名寄存器替换，这个重命名寄存器在处理器内部对应了一个物理寄存器。然后在写回阶段，将重命名寄存器的结果写回物理寄存器中。这样，不同指令中相同的逻辑寄存器就被映射到不同的物理寄存器上，消除了数据冒险。这种显式重命名方式的优点在于其处理器复杂度相对较低，缺点是需要大量的物理寄存器，从而增加了寄存器的开销。

隐式重命名
隐式重命名通过在执行指令的同时进行寄存器重命名，从而避免了指令后重命名的开销。在这种方式下，指令中的源操作数和目标操作数不直接映射到物理寄存器，而是通过指令前面的 ROB（Reorder Buffer）来重命名。ROB 中保存了指令的所有状态，包括指令的操作数和结果，因此可以在执行指令时动态地进行寄存器重命名。当指令执行完后，其结果就被写回到 ROB 中，等到该指令在 ROB 中的序号与其写回的序号相等时，该结果就会被写回到相应的物理寄存器中。由于这种方式下，不需要大量的物理寄存器，因此可以大幅降低寄存器的开销，但需要相应的 ROB 支持。