1. 答：

保留站:指令有序进入保留站,在保留站内等待相关被解决后乱序发射至功能部件进行执行。

重命名寄存器：通过对同一个逻辑寄存器重命名成多个不同的物理寄存器，可以解决 WAW 相关，并且避免伪 RAW 相关。

Reorder buffer：指令不论何时写回寄存器，它在 reoder buffer 中退出的顺序必须**符合程序序**。这保证了乱序执行下，**中断能有精确现场**。

2. 证明：

如果 A(a*i+b)和 A(c*i+d)之间存在相关，则必然存在某个 i1 和 i2，使得 a*i1+b=c*i2+d

这样 d-b=c*i2-a*i1=GCD(c,a)*((c/GCD(c,a))*i2-(a/GCD(c,a))*i1)。

也就是说(d-b)/GCD(c,a)=((c/GCD(c,a))*i2-(a/GCD(c,a))*i1)。

由于上式右半部分显然是整数，因此 GCD(c,a)能够整除(d-b)

3. 答：

在 Tomasulo 算法中：

通过寄存器重命名技术，消除了 WAR 和 WAW 相关。在 tomasulo 算法中，寄存器重命名的功能是通过保留站来实现的。当指令发射到保留站时，它的目标寄存器对应保留站号，消除了 WAR 相关和 WAW 相关。

如果指令的某操作数没有准备好,在它的保留站中的该操作数的位置上标记的是产生该操作数的指令的保留站号。只有当操作数都准备好了，指令才能执行，保证了 RAW 相关。

在记分牌技术中：

在指令发射阶段，如果存在 WAW 相关，则停止发射该指令和任何后继指令。

如果指令的源操作数都准备好了,则计分板会通知相应功能部件读取该操作数并开始执行该指令。这样保证了 RAW 相关。

在指令写回阶段，如果存在 WAR 相关，则停止写回。

4. 答：

所谓精确例外，就是指在处理例外的时候，发生例外指令之前所有的指令都已经执行完了，例外指令后面的所有指令都还没有执行（严格的说是没有产生执行效果，即修改处理机状态）。

实现精确例外处理的办法，就是把后面指令对机器状态的修改延迟到前面指令都已经执行完。具体来讲，在流水线中增加一个叫提交（Commit）的阶段，在这个阶段指令才真正修改机器状态。在执行或者写回阶段，把指令的结果先写到被称为重排序缓存（ReOrder Buffer,

简称 ROB）的临时缓冲器中；在提交的时候，再把 ROB 的内容写回到寄存器或者存储器。一条指令发生了例外，那么对应的 ROB 项，以及之后的指令的 ROB 项就丢弃掉不写回。而这条指令之前的 ROB 项都在 commit 时真正修改寄存器或者存储器。这样就保证了精确中断。

5. 解：

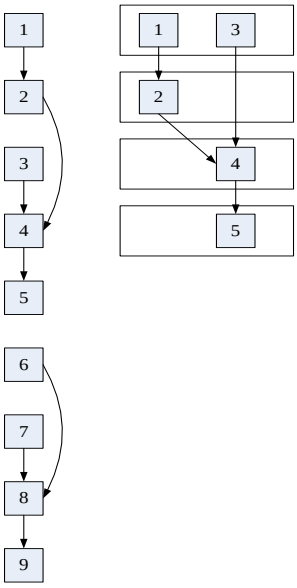
a） 展开两次循环即可消除相关带来的阻塞影响。

```
bar:
        L.D         F2, 0(R1)          ;
        L.D         F3, 8(R1)          ;
        L.D         F6, 0(R2)          ;
        MUL.D       F4, F2, F0         ;
        MUL.D       F5, F3, F0         ;
        L.D         F7, 8(R2)          ;
        DADDUI      R1, R1, #16        ;
        ADD.D       F6, F4, F6         ;
        ADD.D       F7, F4, F7         ;
        DADDUI      R2, R2, #16        ;
        S.D         -16(R2), F6        ;
        DSGTUI      R3, R1, #800       ;
        BEQZ        R3, bar            ;
        S.D         -8(R2), F7         ;
```

14 拍两个，也就是 7 拍 1 个。

b） 首先分析原有循环中数据流图，将运算部分的流图进行流水切分。

```
1     L.D       F2, 0(R1)
2     MUL.D     F4, F2, F0
3     L.D       F6, 0(R2)
4     ADD.D     F6, F4, F6
5     S.D       0(R2), F6
6     DADDUI    R1, R1, #8
7     DADDUI    R2, R2, #8
8     DSGTUI    R3, R1, #800
9     BEQZ      R3, bar
```



软件流水后主题循环代码如下：

```
S.D       -24(R2), F6       ;第 i-3 次循环的 5
ADD.D     F6, F4, F8        ;第 i-2 次循环的 4
MUL.D     F4, F2, F0        ;第 i-1 次循环的 2
L.D       F2, 0(R1)         ;
L.D       F8, -8(R2)        ;
DADDUI    R1, R1, #8        ;
DSGTUI    R3, R1, #800      ;
BEQZ      R3, bar           ;
DADDUI    R2, R2, #8        ;
```

9 拍处理 1 个元素

注： 假如 SD 的偏移为 A，LD(R1)的偏移为 B，LD(R2)的偏移为 C，循环退出条件为 D，有 A+16 = C ， D+B=800，即有（-24，0，-8， 800）或者（-16，0， 0， 800）

全部代码：

装入		循环		排空	
L.D	F2, 0(R1)	S.D	-Y(R2), F6	S.D	-Y(R2),F6
L.D	F8, 0(R2)	ADD.D	F6, F4, F8	ADD.D	F6, F4, F8
MUL.D	F4, F2, F0	MUL.D	F4, F2, F0	S.D	8-Y(R2),F6
ADD.D	F6, F4, F8	L.D	F2, 24-X(R1)	MUL.D	F4, F2, F0
L.D	F2, 8(R1)	L.D	F8, 16-Y(R2)	L.D	F8, 16-Y(R2)
L.D	F8, 8(R2)	DADDUI	R1, R1, #8	ADD.D	F6, F4, F8
MUL.D	F4, F2, F0	DSGTUI	R3, R1, 800-24+X	S.D	16-Y(R2),F6
L.D	F2, 16(R1)	BEQZ	R3, bar		
DADDUI	R1, R1, X	DADDUI	R2, R2, #8		
DADDIU	R2, R2, Y				

6. 解：

指令执行的状态表：(数字表示该操作在第几个时钟周期发生)

指令	发射	执行	写回	提交
DIV F0, F1, F2	1	2	4	5
MUL F3, F0, F2	2	4	6	7
ADD F0, F1, F2	3	4	5	8
MUL F3, F0, F2	4	5	7	9

下面给出的例子，所有寄存器均初始为 1.0 值。

Cycle 1

3		4.0		3				resbus
2		3.0		2				
1		2.0		1				
0	0	1.0		0	div	F0		

Cycle 2

3	1	4.0		3				resbus
2		3.0		2				
1		2.0		1	mul	F3		
0	0	1.0		0	div	F0		

Cycle 3

3	1	4.0		3				resbus
2		3.0		2	add	F0		
1		2.0		1	mul	F3		
0	2	1.0		0	div	F0		

Cycle 4

3	3	4.0		3	mul	F3		resbus
2		3.0		2	add	F0		(0, 0.67)
1		2.0		1	mul	F3		
0	2	1.0		0	div	F0	0.67	

Cycle 5

3	3	4.0		3	mul	F3		resbus
2		3.0		2	add	F0	5.0	(2, 5.0)
1		2.0		1	mul	F3		
0	2	0.67		0				

Cycle 6

3	3	4.0		3	mul	F3		resbus
2		3.0		2	add	F0	5.0	(1, 2.0)
1		2.0		1	mul	F3	2.0	
0	2	0.67		0				

7. 解：

a)

```
mul f0,f0,f0;
```

add f1,f1,f1;

上述指令序列会同时写回，导致停顿。

b)至少需要 3 条结果总线。

注：如果访存定点部件为全流水且延迟固定为 2 或者 3，则只需要 2 条结果总线。