

T9

	1)	2)	3)
fld	$f_2, 0(a_0)$ 1~4	1~4	调整为 fld $f_2, 0(a_0)$ 1~4
fdiv.d	f_8, f_0, f_2 5~15	5~15	fld $f_4, 0(a_1)$ 1~4
fmul.d	f_2, f_6, f_2 16~20	5~9	fdiv.d f_8, f_0, f_2 5~15
fld	$f_4, 0(a_1)$ 17~20	6~9	fadd.d f_4, f_0, f_4 5~7
fadd.d	f_4, f_0, f_4 21~24	10~12	fmul.d f_2, f_6, f_2 6~10
fadd.d	f_8, f_8, f_2 25~28	16~18	fsub $f_4, 0(a_1)$ 8~9
fsub	$f_{10}, 0(a_0)$ 29~30	19~20	addi $a_1, a_1, 8$ 9~17
fsub	$f_4, 0(a_1)$ 31~32	19~20	fadd.d f_{10}, f_8, f_2 16~18
addi	$a_0, a_0, 8$ 32	20	fsub $f_{10}, 0(a_0)$ 19~20
addi	$a_1, a_1, 8$ 33	20	addi $a_0, a_0, 8$ 19
sub	x_{20}, x_4, a_0 34	21	sub x_{20}, x_4, a_0 20 这
bnz	x_{20}, Loop 35~36	22~23	bnz x_{20}, Loop 21~22 用
4内存加载	共36个周期	共22个周期	共22个周期

T10

```
Loop: fld T9, 0(a0)
      fmul.d T10, T0, T2
      fdiv.d T11, T9, T10
      fld T12, 0(a1)
      fadd.d T13, T0, T12
      fsub.d T14, T11, T13
      fsd T14, 0(a1)
      T11.
```

区别:

显式重命名安排了比指令集要求的寄存器数量更多的物理寄存器。在这些物理寄存器中不仅包括已经提交的寄存器，也包括尚未提交处于“推测”状态的寄存器。在重命名的同时，每个指令的寄存器指示符指向所使用的物理寄存器。

隐式重命名中物理寄存器的数量和LSA寄存器数量相同，LSA寄存器只保存已经提交的指令的值，而不包括处于“推测”状态的值，这部分内容交由ROB保存。指令提交时，ROB将值提交给LSA寄存器堆。

优缺点:

显式重命名优点在于更易理解和维护，因为它直接表明了寄存器的新含义。读取数据复杂度低，功耗低。

隐式寄存器化点是可以减少代码中指令数和访存次数，因为它不需要将值存储在内存中，并在寄存器之间传递。缺点是导致代码更难理解和维护，功耗较高。

可能实现的方式

显式重命名:引入空闲列表维护物理寄存器空闲状态信息;引入重命名列表维护物理寄存器和ISA寄存器的映射关系。当指令译码后,处理器查找FL并选择一个空闲的物理寄存器,将其和指令要写入的目的ISA寄存器进行绑定并记录在RT中。同时指令的源操作数也需要查找RT以确定是否需要从某个被映射的物理寄存器中取出对应值。当指令执行阶段结束后,结果会被写入对应的物理寄存器。

隐式重命名:需一个额外的表项来记录寄存器的最新值是否已写回ARF中还是暂存在重排序缓冲区中。为此重排序缓冲区一般需支持前缀,以将处于推测状态的最新值转给其他指令作为源操作数。