

18.9.

1)

fld f2, 0(a0) RAW 4 1 6

fdiv.d f8, f0, f2 11 2 18

fmul.d f2, f6, f2 5 7 13 19

fld f4, 0(a1) RAW 4 8 20

fadd.d f4, f0, f4 3 9 24

fadd.d f4, f10, f8, f2 RAW 3 21 25

fsd f10, 0(a0) 2 22 28

fsd f4, 0(a1) 2 26 29

addi a0, a0, a8 1 27 30

addi a1, a1, 8 RAW 1 28 31

sub x20, x4, a0 RAW 1 29 32

bnz x20, Loop RAW 2 31 35

需要35个周期

2)

双发射

发射 写回结束

1 6

 $\frac{1}{2} \times 1 + 18 = 19$

7 19

 $\frac{1}{2} \times 1 + 20 = 20$

8 24

 $\frac{1}{2} \times 1 + 25 = 25$

21 28

 $\frac{1}{2} \times 1 + 29 = 29$

26 30

26 31

27 32

27 35

需要35个周期



扫描全能王 创建

37					
	序号	ex	发射	写回结束	
fld f2, 0(a0)	1	4		6	
fld f4, 0(a1)	4	4		7	
fmul.d f2, f6, f2	3	5	2	12	
fdiv.d f8, f0, f2	2	11	2	18	
fadd.d f4, f0, f4	5	13	7	19	
fadd.d f10, f8, f2	6	3	7	22	
fsd f4, 0(a0)	8	2	19	23	
fsd f10, 0(a1)	7	2	19	25	
addi a0, a0, a8	9	1	23	26	
addi a1, a1, 8	10	1	23	27	
sub x20, x4, a0	11	1	24	28	
bnez x20, Loop	12	2	24	31	

需要 31 个周期

10.

Loop: fld f4, 0(a0) 默认初始
 fmul.d f2, f0, f2 $f_x \rightarrow T_x$
 fdiv.d f8, f4, f2
 fld f4, 0(a1)
 fadd.d f6, f0, f4
 fsub.d f8, f8, f6
 fsd f8, 0(a1)

重命名后

Loop: fld T9, 0(a0) $f4 \rightarrow T9$
 fmul.d T10, T0, T2 $f2 \rightarrow T10$
 fdiv.d T11, T9, T10 $f8 \rightarrow T11$
 fld T12, 0(a1) $f4 \rightarrow T12$
 fadd.d T13, T0, T12 $f6 \rightarrow T13$
 fsub.d T14, T11, T13 $f8 \rightarrow T14$
 fsd T14, 0(a1)



11.

区别:

显式重命名: 安排了比ISA寄存器数量更多的物理寄存器, 既包括已经提交的寄存器又包括尚未处于“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: ISA寄存器数量与物理寄存器相同, 仅保存已提交的指令的值, 不包括处于“推测”状态的值, 这些交由ROB保存。指令提交时, ROB将值提交给ISA寄存器堆

优缺点:

隐式重命名不需要free-List来记录物理寄存器

状态, 指令被写入ROB即完成重命名,

需要的物理寄存器数目更少

隐式重命名: 每个操作数在其生命周期中需要保存在ROB和ARF两个位置, 读取数据的复杂度较高, 功耗更高

可能的实现方式:

显式:

map-table 记录ISA与物理寄存器的映射关系

free-List 记录物理寄存器的空闲状态

busy-table 记录寄存器是否可读

流程: 索引map-table获取源操作数逻辑寄存器对应的物理寄存器, 由free-List分配一个空闲的物理寄存器作为指令的目的寄存器, 由busy-table判断是否可读, 如果可读则发射指令

隐式:

尚未提交 ROB - 正在执行 即将写入 ARF - 已经提交 (的值保存)

映射表, 记录操作数在ROB中的位置

且记录对应寄存器的最新值保存在ROB还是ARF中
指令被写入ROB即完成重命名

