

4/21 第三章

3. (1) 不存在数据依赖。因为存储和加载操作使用不同的虚拟和物理地址。
 (2) 存在。因为存储和加载操作使用相同的虚拟地址和物理地址中的不同偏移量。
 (3) 不存在。因为存储和加载操作使用不同的虚拟地址和物理地址。
 (1) ⁽²⁾ ~~存在~~ 存在数据依赖, (3) 不存在。因为 (3) 中地址间隔超过 4KB, 而 (1)、(2) 未超过并被映射到相同物理地址页。

9. (1) ~~与 fdiv 数据依赖~~

fld f2, 0(a0)	4	①: fld, 与 fdiv 数据依赖 f2 冲突
fdiv.d f8, f0, f2	11	⑤: fld 完成, fdiv 进入, ③ 完成
fmul.d f2, f8, f2	5	⑥: fmul.d 进入, ④ 完成
fld f4, 0(a0)	4	⑦: fld 进入, ⑥ 完成
fadd.d f4, f0, f4	3	⑧: fadd.d fadd.d 进入, ⑦ 完成
fadd.d f10, f8, f2	3	⑨: fadd.d f10, f8, f2 进入, ⑧ 完成
fsd f10, 0(a0)	2	⑩: fsd f10, 0(a0) 进入, ⑨ 完成
fsd f4, 0(a1)	2	⑪: fsd f4, 0(a1) 进入, ⑩ 完成
addi a0, a0, 8	1	⑫: addi a0, a0, 8 进入, 完成
addi a1, a1, 8	1	⑬: addi a1, a1, 8 进入, 完成
sub x20, x4, a0	1	⑭: sub x20, x4, a0 进入, 完成
bne x20, Loop	2	⑮: bne x20, Loop 进入, ⑭ 完成

∴ 需要 ³¹ 个周期。

(2) ~~fld f2, 0(a0)~~, ~~fldiv.d f8, f4, f2~~, ~~fmul.d f2, f6, f2~~, ~~fadd.d f10, f8, f2~~, ~~addi a1, a1, 8~~, ~~fsub.d f11, f8, f6~~, ~~fsub.d f12, 0(a1)~~

①: fld, fldiv.d 进入, 冲突, ④ 完成冲突, fld ④ 完成, fldiv.d ⑥ 完成

⑤: fmul.d, fld 进入, ⑥ 完成冲突, ⑦ 完成 fmul.d ⑧ 完成, fld ⑧ 完成

⑩: fadd.d 和 fadd.d 进入, ⑩ 完成冲突, fadd.d ⑩ 完成, ⑪ 完成

⑭: fsub.d 和 fsub.d 进入, ⑭ 完成冲突, fsub.d ⑭ 完成, ⑮ 完成

⑰: addi 和 addi 进入, ⑰ 完成冲突, addi ⑰ 完成, ⑱ 完成

⑲: sub 和 bnez 进入, ⑲ 完成冲突, sub ⑲ 完成, bnez ⑲ 完成

∴ 需要 29 个周期

(3) fld f2, 0(a0)

fld f4, 0(a1)

fldiv.d f8, f4, f2

fadd.d f10, f0, f4

fmul.d f2, f6, f2

fsub.d f4, 0(a1)

fadd.d f10, f8, f2

addi a1, a1, 8

fsub.d f10, 0(a0)

addi a0, a0, 8

sub x20, x4, a0

bnez x20, Loop

①: fld 和 fld 进入, ④ 完成

⑤: fldiv.d 和 fadd.d 进入, ⑤ 开始, fldiv.d ⑨ 完成, fadd.d ⑦ 完成

⑩: fmul.d 和 fsub.d 进入, fmul.d ⑩ 完成, fsub.d ⑪ 完成

⑭: fadd.d 和 addi 进入, fadd.d ⑭ 完成, addi ⑭ 完成

⑰: fsub.d 和 addi 进入, fsub.d ⑰ 完成, addi ⑰ 完成

⑲: sub 和 bnez 进入, sub ⑲ 完成, bnez ⑲ 完成

10. Loop: fld f4, 0(a0)

fmul.d T9, f0, f2

fldiv.d f8, f4, f2

fld T10, 0(a1)

fadd.d f6, f0, T10

fsub.d T11, f8, f6

fsub.d T12, 0(a1)

11. 显式重命名 ~~提交的数据~~ 在编译或运行时将虚拟寄存器映射到物理寄存器, 优点是可以高效实现寄存器分配, 缺点是需要 在程序中插入额外指令, 增加代码量;

隐式重命名 在处理器内部自动将虚拟寄存器映射到物理寄存器, 优点是不用在程序中插入额外指令, 可以有效减少代码量和编译时间, 缺点是需要 额外的硬件支持。

实现方式: 显式重命名可以在程序中插入专门的重命名指令来实现, 例如 Intel x86 架构中的 MOV 指令。
隐式重命名在处理器中, 指令首先被解码并分配物理寄存器, 将其重命名为表中的条目, 例如 Intel x86 架构中的 Pentium Pro 处理器。