

作业9:

9. ~~严格顺序时数据冲突只考虑RAW:~~

1). 如图:

共21个周期

2). 共20个周期

3). fld

fdiv.d

fmul.d

fld

fadd.d

fsd

fsd

addi

共18个周期

addi

sub

fadd.d

bnz

第一小问图：

[illegible]

第二小问图：

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
fld	Y	Y	Y	Y(f2)																		
fdiv.d					Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y(f8)							
fmul.d					Y	Y	Y	Y	Y(f2)													
fld					Y	Y	Y	Y(f4)														
fadd.d					Y	Y	Y(f4)															
fadd.d																Y	Y	Y(f10)				
fsd																Y	Y(f10)					
fsd																	Y	Y(f4)				
addi																	Y(a0)					
addi																		Y(a1)				
sub																		Y(x20)				
bnz																			Y	Y		

第三小问图：

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
fld	Y			Y(f2)																			
fdiv.d		Y	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y(f8)								
fmul.d				Y	Y	Y	Y	Y	Y(f2)														
fld					Y	Y	Y	Y	Y(f4)														
fadd.d					Y	Y	Y	Y(f4)															
fsub.d						Y	Y	Y(f10)															
fdiv.d						Y	Y	Y(f4)															
addi							Y(a0)																
addi							Y(a1)																
sub								Y(x20)															
fadd.d															Y	Y	Y(f10)						
bnz																Y	Y						

10 ⑩ fld: $D = \{a_0\}$, $R = \{f_{14}\}$
 ⑪ fmul.d: $D = \{f_2, f_0\}$, $R = \{f_2\}$
 ⑬ fdiv.d: $D = \{f_2, f_4\}$, $R = \{f_8\}$
 ⑭ fld: $D = \{a_1\}$, $R = \{f_{14}\}$
 ⑮ fadd.d: $D = \{f_4, f_0\}$, $R = \{f_6\}$
 ⑯ fsub.d: $D = \{f_6, f_8\}$, $R = \{f_8\}$
 ⑰ fsd: $D = \{f_8, a_1\}$, $R = \{f_{14}\}$

WAW: ⑩-⑭, ⑬-⑯, ~~⑮-⑰~~, ~~⑮-⑰~~

WAR: ~~⑮-⑰~~,

RAW: ⑩-⑬, ⑩-⑮, ⑮-⑰, ⑮-⑰, ⑮-⑰,
 ⑭-⑮, ⑮-⑰, ⑰-⑰

改为:

fld T_4 , 0(a₀)

fmul.d T_2 , T_0 , T_2

fdiv.d T_8 , T_4 , T_2

fld T_9 , 0(a₁)

fadd.d T_6 , T_0 , T_9

fsub.d T_8 , T_8 , T_6

fsd T_{10} , 0(a₁)

第九次作业

区别：显示重命名的 ROB 不记录结果，而是将数据都存在物理寄存器中，通过为逻辑寄存器分配空闲物理寄存器来实现。

隐式重命名需要两个表格（ROB 和 ARF）保存记过，而不是将结果写在实际的物理寄存器中。逻辑寄存器和物理寄存器的数量相同。

优缺点：两个对比发现隐式重命名读取数据的复杂度较高，功耗更高，但是需要的物理寄存器数量更少。

实现方式：

用“重命名映射表”实现逻辑寄存器和物理寄存器的对应关系，寻找空闲物理寄存器写入数据消除 WAW, WAR。

1. 重排序缓冲

用 ROB 表项编号建立与逻辑寄存器的联系。通过分配空闲的物理寄存器并为其编号，再将编号与逻辑寄存器对应起来，为每一条指令分配一个表项和一个编号，指明写、读数据的位置。

缺点：1) 如果进入表格的指令不需要寄存器，但是实际给该指令分配了寄存器，则会浪费一个物理寄存器。

2) 由于源数据可能在逻辑寄存器堆（逻辑）中或者是 ROB 表（物理）中，所以对一条指令需要 ROB 有两个读端口，对芯片的面积与延时造成影响。

2. 扩展逻辑寄存器（类似于 1）

在逻辑寄存器堆、ROB 之外额外设计一组物理寄存器。当有空闲物理寄存器的时候，发射的指令中的寄存器会被对应一个物理寄存器，并构建一个映射，指令执行的结果被存在物理寄存器中，最后提交时再把物理寄存器的值写入逻辑寄存器堆。

3. 使用统一的物理寄存器重命名（上述方式当一项数据被存在物理寄存器中时，对应的逻辑寄存器一直空闲，造成空间浪费）

不区分逻辑和物理寄存器。逻辑寄存器映射表和空闲物理寄存器可以由任何一个寄存器构成，根据动态指定的映射表确定逻辑寄存器和需要使用的物理寄存器。

优点：可以减少数据的写入次数，只要一次。

一个源数据只存在于一个地方

←