北京邮电大学计算机系统结构实验报告



实验日期:

2019年5月23日

0. 目录

0.	目录	_ 1
1.	实验目的	_ 2
2.	实验平台	_ 2
3.	实验内容和步骤	_ 2
4.	实验结论	-

1. 实验目的

- (1) 加深对指令调度技术的理解。
- (2) 加深对延迟分支技术的理解。
- (3) 熟练账务用指令调度技术解决流水线中的数据冲突的方法。
- (4) 进一步理解指令调度技术对 CPU 性能的改进。
- (5) 进一步理解延迟分支技术对 CPU 性能的改进。

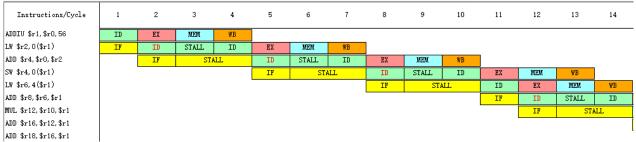
2. 实验平台

指令级和流水线操作级模拟器 MIPSsim

3. 实验内容和步骤

- (1) 启动 MIPSsim。
- (2) 根据实验 2 的相关知识中关于流水线各段操作的描述,进一步理解流水线窗口中各段的功能,掌握各流水线寄存器的含义(双击各段,就可以看到各流水线寄存器中的内容)。
- (3) 选择"配置"→"流水方式"选项,使模拟器工作在流水方式下。
- (4) 用指令调度技术解决流水线中的结构冲突与数据冲突:
 - 1) 启动 MIPSsim。
 - 2) 用 MIPSsim 的"文件"->"载入程序"选项来加载 '5-schedule.s'
 - 3) 关闭定向功能,这是通过"配置"->"定向"选项来实现的。
 - 4) 执行所载入的程序,通过查看统计数据和时钟周期图,找出并记录程序执行过程中各种冲突发生的次数,发生冲突的指令组合以及程序执行的总时钟周期数。

汇总: 汇总: 九行周期总数: 33 ID段执行了15条指令 执行周期总数: 33 硬件配置: 内存容量: 4096 B 加法器个数:1 执行时间(周期数):6 ID 段执行了 15 条指令 乘法器个数:1 除法器个数:1 执行时间(周期数)7 执行时间(周期数)10 停顿 (周期数): 定向机制: 不采用 停顿(周期数): RAW 停顿: 16 占周期总数的百分比: 48.48485% 占周期总数的百分比: 48.48485% RAW停顿: 16 其中: load信顿: 6 占所有RAW信顿的百分比: 37.5% 其中: 浮点停顿: 0 占所有RAW停顿的百分比: 0% ₩A₩停顿: O 占周期总数的百分比:0% 结构停顿: 0 占周期总数的百分比: 0% load 停顿: 6 占所有 RAW 停顿的百分比: 37.5% 控制停顿: 0 占周期总数的百分比:0% 占周期总数的百分比: 3,030303% 白路停顿: 1 停顿周期总数: 17 占周期总数的百分比: 51.51515% 浮点停顿: 0 占所有 RAW 停顿的百分比: 0% 分支指令: 指令条数:0 占指令总数的百分比:0% WAW 停顿: 0 占周期总数的百分比: 0% 其中: 分支成功: 0 占分支指令数的百分比: 0% 分支失败: 0 占分支指令数的百分比: 0% 结构停顿: 0 占周期总数的百分比: 0% load/store指令: 指令条数: 5 占指令总数的百分比: 33.3333% 控制停顿: 0 占周期总数的百分比: 0% 其中: load: 3 占load/store指令数的百分比: 60% 占load/store指令数的百分比: 40% 自陷停顿: 1 占周期总数的百分比: 3.030303% 停顿周期总数: 17 占周期总数的百分比: 51.51515%



发生冲突的指令组合:

ADDIU \$r1,\$r0,A LW \$r2,0(\$r1) **ADD** \$r4,\$r0,\$r2 SW \$r4,0(\$r1) \$r6,4(\$r1) LW ADD \$r8,\$r6,\$r1 \$r12,\$r10,\$r1 MUL **ADD** \$r16,\$r12,\$r1 **ADD** \$r18,\$r16,\$r1 \$r18,16(\$r1) SW r20,8(r1)LW \$r22,\$r20,\$r14 MUL \$r24,\$r26,\$r14 MUL

相邻两条指令之间均存在冲突

- 5) 自己采用调度技术对程序进行指令调度,消除冲突(自己修改源程序)。将调度(修
- 改)后的程序重新命名为 5-aferschedule.s。(注意:调度方法灵活多样,在保证程序正确

性的前提下自己随意调度,尽量减少冲突即可,不要求要达到最优。)

■ 优化后的代码:

```
.text
main:
ADDIU $r1,$r0,A
LW
      $r2,0($r1)
MUL
     $r12,$r10,$r1
ADD
      $r4,$r0,$r2
ADD
    $r<mark>16,$r12,$r1</mark>
SW
      $r4,0($r1)
ADD
     $r18,$r16,$r1
      $r6,4($r1)
LW
      $r20,8($r1)
LW
ADD
    $r8,$r6,$r1
SW
     $r18,16($r1)
MUL
     $r22,$r20,$r14
      $r24,$r26,$r14
MUL
TEQ $r0,$r0
.data
A:
.word 4,6,8
```

6) 载入 afer-schedule.s, 执行该程序, 观察程序在流水线中的执行情况, 记录程序执行的总时钟周期数。

■ 执行结果:

汇总:

执行周期总数: 22 ID 段执行了 15 条指令

停顿 (周期数):

RAW 停顿: 5 占周期总数的百分比: 22.72727%

其中:

 load 停顿: 2
 占所有 RAW 停顿的百分比: 40%

 浮点停顿: 0
 占所有 RAW 停顿的百分比: 0%

WAW 停顿: 0 占周期总数的百分比: 0%

结构停顿: 0占周期总数的百分比: 0%控制停顿: 0占周期总数的百分比: 0%

自陷停顿: 1 占周期总数的百分比: 4.545455% 停顿周期总数: 6 占周期总数的百分比: 27.27273% 7) 比较调度前和调度后的性能,论述指令调度对提高 CPU 性能的作用。 调度前: 调度后:

汇总: 执行周期总数: 33 ID段执行了15条指令 内存容量: 4096 B 加法器个数: 1 执行时间(周期数):6 乘法器个数:1 执行时间(周期数)7 除法器个数:1 执行时间(周期数)10 定向机制: 不采用 停顿(周期数): 占周期总数的百分比: 48.48485% RAW停顿: 16 load傳輸: 6 占所有RAW停顿的百分比: 37.5% 浮点停顿: 0 占所有RAW停顿的百分比: 0% WAW停顿: 0 结构停顿: 0 占周期总数的百分比:0% 占周期总数的百分比: 0% 占周期总数的百分比:0% 占周期总数的百分比: 3.030303% 白陷停顿: 1 停顿周期总数:17 占周期总数的百分比: 51.51515% 分支指令: 指令条数:0 占指令总数的百分比:0% 其中: . 分支成功: 0 占分支指令数的百分比: 0% 分支失败: 0 占分支指令数的百分比:0% load/store指令: 占指令总数的百分比: 33.3333% 指令条数:5 其中: load: 3 占load/store指令数的百分比: 60% 占load/store指令数的百分比: 40%

汇总: .心· 执行周期总数:22 ID段执行了15条指令 硬件配置: 内存容量: 4096 B 加法器个数: 1 乘法器个数: 1 执行时间(周期数): 6 执行时间(周期数)7 执行时间(周期数)10 除法哭个数:1 定向机制: 不采用 停顿(周期数): 占周期总数的百分比: 22.72727% RAW停顿:5 其中: load停顿: 2 占所有RAW停顿的百分比: 40% 占所有RAW停顿的百分比: 0% 占周期总数的百分比: 0% 浮点停顿: 0 WAW停顿: 0 结构停顿: 0 占周期总数的百分比:0% 占周期总数的百分比: 0% 控制停顿: 0 占周期总数的百分比: 4.545455% 停顿周期总数:6 占周期总数的百分比: 27.27273% 分支指令: 指令条数:0 其中: 占指令总数的百分比:0% +中: 分支成功: 0 分支失败: 0 占分支指令数的百分比: 0% 占分支指令数的百分比:0% load/store指令: 占指令总数的百分比: 33,33333% 指令条数:5 load: 3 占load/store指令数的百分比: 60% 占load/store指令数的百分比: 40% store: 2

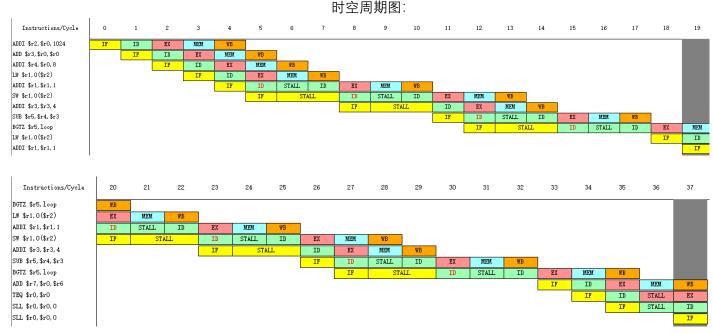
调度后,执行周期总数减少了11,总效率提高了33.3% 通过指令调度,消除部分的数据冲突,大幅度减少了RAW停顿次数,提高了CPU的性能

- (5) 用延迟分支技术减少分支指令对性能的影响:
 - 1) 在 MIPSsim 中载入 5-branch.s 样例程序

.text main: ADDI \$r2,\$r0,1024 **ADD** \$r3,\$r0,\$r0 ADDI \$r4,\$r0,8 loop: \$r1,0(\$r2) LW ADDI \$r1,\$r1,1 SW \$r1,0(\$r2) ADDI \$r3,\$r3,4 **SUB** \$r5,\$r4,\$r3 BGTZ \$r5,loop \$r7,\$r0,\$r6 ADD \$r0,\$r0 TEQ

2) 关闭延迟分支功能。这是通过在"配置"->"延迟槽"选项来实现的。

3) 执行该程序,观察并记录发生分支延迟的时刻,记录该程序执行的总时钟周期数。



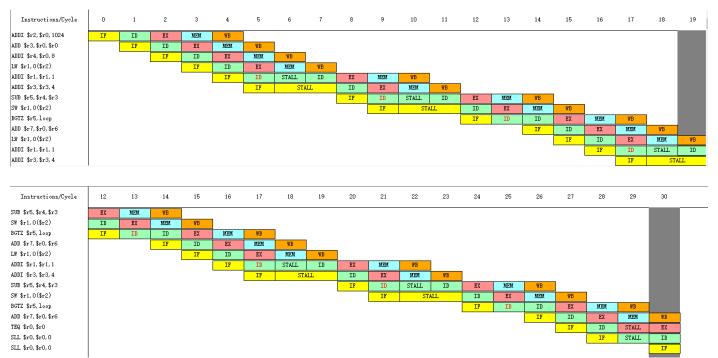
发生分支延迟的时刻: 13, 28

总的周期数: 38

4) 假设延迟槽为一个,自己对 5-branch.s 程序进行指令调度(自己修改源程序),将 调度后的程序重新命名为 5-delayed-branch.s。

进行指令调度 .text main: ADDI \$r2,\$r0,1024 \$r3,\$r0,\$r0 ADD ADDI \$r4,\$r0,8 loop: \$r1,0(\$r2) LW ADDI \$r1,\$r1,1 ADDI \$r3,\$r3,4 **SUB** \$r5,\$r4,\$r3 \$r1,0(\$r2) SW BGTZ \$r5,loop \$r7,\$r0,\$r6 ADD \$r0,\$r0 TEQ

5) 载入 5-delayed-branch.s, 打开延迟分支功能, 执行该程序, 观察其时钟周期图, 记录程序执行的总时钟周期数。



指令调度优化并使用延迟槽后的执行总周期数: 31

4. 实验结论

未使用延迟槽: 11 13 17 ADDI \$r2,\$r0,1024 ADD \$r3,\$r0,\$r0 ADDI \$r4,\$r0,8 LW \$r1,0(\$r2) ADDI \$r1,\$r1,1 SW \$r1,0(\$r2) ADDI \$r3,\$r3,4 SUB \$r5, \$r4, \$r3 BGTZ \$r5.loop LW \$r1,0(\$r2) ADDI \$r1,\$r1,1 Instructions/Cycle 20 21 22 23 24 27 30 31 32 33 34 35 37 BGTZ \$r5,loop LW \$r1,0(\$r2) ADDI \$r1,\$r1,1 STALL SW \$r1,0(\$r2) ADDI \$r3,\$r3,4 MEM SUB \$r5, \$r4, \$r3 BGTZ \$r5,loop ADD \$r7, \$r0, \$r6 TEQ \$r0,\$r0 EX SLL \$r0, \$r0, 0 ID SLL \$r0, \$r0, 0

使月	用延迟	₹槽:																		
Instructions/Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	1
ADDI \$r2,\$r0,1024	IF	ID	EX	MEM	WB															
ADD \$r3,\$r0,\$r0		IF	ID	EX	MEM	WB														
ADDI \$r4,\$r0,8			IF	ID	EX	MEM	WB													
LW \$r1,0(\$r2)				IF	ID	EX	MEM	WB												
ADDI \$r1,\$r1,1					IF	ID	STALL	ID	EX	MEM	WB									
WDDI \$r3,\$r3,4						IF	STA	ALL	ID	EX	MEM	WB	1							
SUB \$r5,\$r4,\$r3									IF	ID	STALL	ID	EX	MEM	WB					
SW \$r1,0(\$r2)										IF	STA	LL	ID	EX	MEM	WB				
BGTZ \$r5,loop													IF	ID	ID	EX	MEM	WB		
ADD \$r7, \$r0, \$r6														•	IF	ID	EX	MEM	WB	
LW \$r1,0(\$r2)																IF	ID	EX	MEM	,
DDI \$r1,\$r1,1																	IF	ID	STALL	
ADDI \$r3,\$r3,4																		IF	ST	LL
Instructions/Cycle	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
SUB \$r5, \$r4, \$r3	EX	MEM	WB																	
SW \$r1,0(\$r2)	ID	EX	MEM	WB	1															
BGTZ \$r5,loop	IF	ID	ID	EX	MEM	WB	1													
ADD \$r7, \$r0, \$r6			IF	ID	EX	MEM	WB	1												
LW \$r1,0(\$r2)				IF	ID	EX	MEM	WB	1											
ADDI \$r1,\$r1,1					IF	ID	STALL	ID	EX	MEM	WB	1								
ADDI \$r3,\$r3,4						IF	ST	ALL	ID	EX	MEM	WB								
									IF	ID	STALL	ID	EX	MEM	WB					
SUB \$r5,\$r4,\$r3										IF	ST.	ATT	ID	EX	MEM	WB				
										Tr	51.									
SW \$r1,0(\$r2)										- 11	51.	ria.	IF	ID	ID	EX	MEM	WB		
SW \$r1,0(\$r2) BGTZ \$r5,loop										- 11	51.	, can					MEM EX	WB	WB	
W \$r1,0(\$r2) GTZ \$r5,loop DD \$r7,\$r0,\$r6										Ir	31.				ID	EX			WB EX	
SUB \$r5, \$r4, \$r3 SW \$r1, O(\$r2) SGTZ \$r5, loop UD \$r7, \$r0, \$r6 SEQ \$r0, \$r0 SLL \$r0, \$r0, 0										11	31.	The state of the s			ID	EX	EX	MEM		

两者差别在于使用了延迟槽后,运行到跳转指令时,分支指令 BGTZ 不会停顿等待,减少了分支延迟,提高了 CPU 的执行效率。