**《计算机原理与体系结构》实验报告**

**学号：16300240023 姓名：杨芗琳 学院：计算机科学技术学院 完成时间：2018.5.1**

1. 实验名称

延迟分支和动态调度

1. 实验目的
2. 熟悉延迟分支
3. 熟悉延迟分支时代码的移动
4. 熟悉循环展开处理
5. 熟悉记分牌算法和Tomasulo算法
6. 熟悉分支性能
7. 实验过程

3.1 Use PCSPIM to observe the Delayed Branches behavior.

1.在原来的代码中，每一次循环之间不能并行执行，速度较慢，并且每次循环都要判断，速度很慢。

2.将循环展开之后，位于分支判断条件之后的



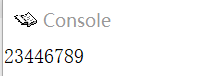
，独立于分支指令。当存在分支延迟时，无论分支是否成功，都会执行，填补了一个时钟周期，加快了速度。

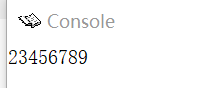
3.思路：设置一个数组（1，2，3，4，5，6，7，8），令$f2=1，当存在分支延迟时，

|  |
| --- |
| s.d $f16,-8($t0) |

一定会被执行，所以最后数组变为（2，3，4，5，6，7，8，9）；当不存在分支延迟时，第四个元素不会被修改，故数组为（2，3，4，4，6，7，8，9）

输出截图：





3.3.1 Why the following codes (code 3.1 and Code 3.2) are equivalent?

3.2代码用位运算取代了3.1代码中的分支

|  |
| --- |
| if(data[c]>=128)  sum+=data[c]; |

|  |
| --- |
| int t=(data[c]-128)>>31;  sum+=~t&data[c]; |

data[c]是0~255之间的整数，3.1代码是将大于128的整数加到sum中；而3，2代码中，（data[c]-128）右移31位，如果原来data[c]是大于128的数，则t=0，否则t=1；所以当data[c]>=128时，~t=1，~t&data[c]得到的就是data[c]的值；data[c]<128时，~t=0,故~t&data[c]=0；综上所述，3.1和3.2代码的效果是相同的。

3.3.2 \* Write or add C++ code to compare performance of different schemes

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Schemes | Scheme name | Compiler and option | 1/s | 2/s | 3/s | 4/s | 5/s | Avg time/s | rank |
| 1 | Random | C++ Visual Studio 2017 –x64 release | 21.058 | 21.056 | 21.205 | 21.067 | 21.023 | 21.082 | 4 |
| 2 | Sorted | 7.364 | 7.509 | 7.448 | 7.401 | 7.396 | 7.424 | 1 |
| 3 | Random-hacked | 8.198 | 8.093 | 8.094 | 8.149 | 8.211 | 8.149 | 3 |
| 4 | sorted-hacked | 8.229 | 8.044 | 8.046 | 8.127 | 8.071 | 8.103 | 2 |