计算机系统结构实验报告

班 级： 210701

学 号： 21074121

姓 名： 何超然

|  |  |  |  |
| --- | --- | --- | --- |
| **教师签字：** |  | **评阅日期：** |  |

目录

[实验一 流水线中的相关 3](#_Toc153228526)

[一、 实验内容分析 3](#_Toc153228527)

[实验二 循环展开及指令调度 8](#_Toc153228528)

[一、 实验目的 8](#_Toc153228529)

[二、实验内容及结果分析 8](#_Toc153228530)

[实验三 Cache 性能分析 8](#_Toc153228531)

[一、 选用的测试程序及基本配置下的结果 8](#_Toc153228532)

[二、 改变cache容量 8](#_Toc153228533)

[三、改变cache相联度 8](#_Toc153228534)

[四、改变Cache块大小 8](#_Toc153228535)

[五、不同Cache容量下、不同替换算法 8](#_Toc153228536)

[六、不同Cache相联度下、不同替换算法的实验结果与记录 8](#_Toc153228537)

# 实验一 流水线中的相关

## 实验内容分析

1. **执行sum.s,test\_for.s程序，熟练掌握winmips的操作和使用**
2. 运行sum.s并观察，有如下截图

图形用户界面, 应用程序

描述已自动生成

1. 运行test\_for并观察，运行截图如下

图形用户界面

描述已自动生成

1. **用WinMIPS64运行程序test\_for.s，记录相关数据，找出存在资源相关和导致资源相关的部件；记录资源相关引起的暂停时钟周期数，计算占总周期的百分比；论述资源相关对CPU性能的影响；讨论资源相关解决方法；（我是在不采用定向技术下运行的）**
2. 找出存在资源相关和导致资源相关的部件

图表

低可信度描述已自动生成

观察图中raw情况，可知在指令dadd r8,r5,r6与dadd r9,r7,r8有先写后读的数据相关，导致dadd r9,r7,r8指令一直等待在取指阶段，占用ID部件，又会与sw r9,0(r1)指令发生资源相关。再往下，又会与daddi r1,r1,8发生数据相关,要等待sw指令写入r1后才能执行。

1. 记录资源相关引起的暂停时钟周期数，计算占总周期的百分比

文本

描述已自动生成

如上：

总时钟数：**122**

资源相关引起的暂停时钟数：**42**

百分比：**42/122 = 0.34**

1. 论述资源相关对CPU性能的影响；讨论资源相关解决方法

影响：由于资源相关，会导致指令总周期数增大，而指令总数不变的情况下，执行时间 变长了，即CPI增大，性能就下降。

解决方法：可设置多个功能部件，避免资源不够用，导致资源相关。或者改变指令执行 顺序，但不改变功能。

1. **定向与不定向下运行sum.s，test\_for.s并分析差异**
2. 分别在定向，不定向的情况下下运行sum.s,test\_for.s，得到结果如下：

**图形用户界面, 文本

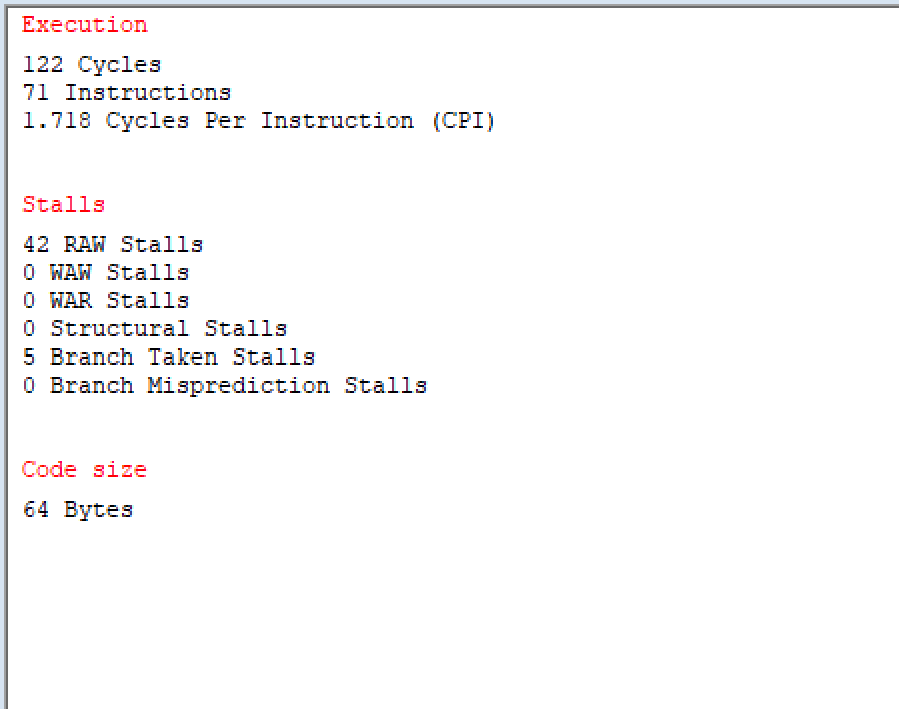
描述已自动生成**

图表 1 不定向运行sum.s

**文本

描述已自动生成**

图表 2 定向运行sum.s



图表 3 不定向运行test\_for.s

图形用户界面, 文本

描述已自动生成

图表 4 定向运行test\_for.s

1. 汇总结果计算有：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sum.s** | | | | | |
|  | 总时钟数 | 数据相关引起的暂停时钟数 | 占的百分比 | cpi | 性能提升倍数 |
| 不采用定向 | 13 | 4 | 0.31 | 2.6 | 1.3 |
| 采用定向 | 10 | 1 | 0.1 | 2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **sum.s** | | | | | |
|  | 总时钟数 | 数据相关引起的暂停时钟数 | 占的百分比 | cpi | 性能提升倍数 |
| 不采用定向 | 122 | 42 | 0.34 | 1.7 | 1.4 |
| 采用定向 | 86 | 6 | 0.07 | 1.2 |

# 实验二 循环展开及指令调度

## 实验目的

1、加深对循环级并行性、指令调度技术、循环展开技术以及寄存器换名技术的理解；

2、熟悉用指令调度技术来解决流水线中的数据相关的方法；

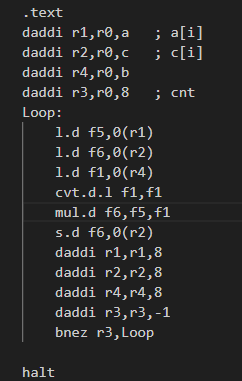
3、了解循环展开、指令调度等技术对 CPU 性能的改进。

## 二、实验内容及结果分析

**1、 用 DLX 汇编语言编写代码文件，记录数据**

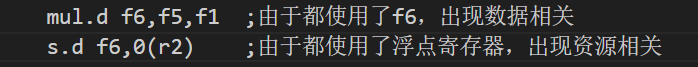
（1）汇编代码实现功能：循环八次的乘法

（2）汇编代码如下：



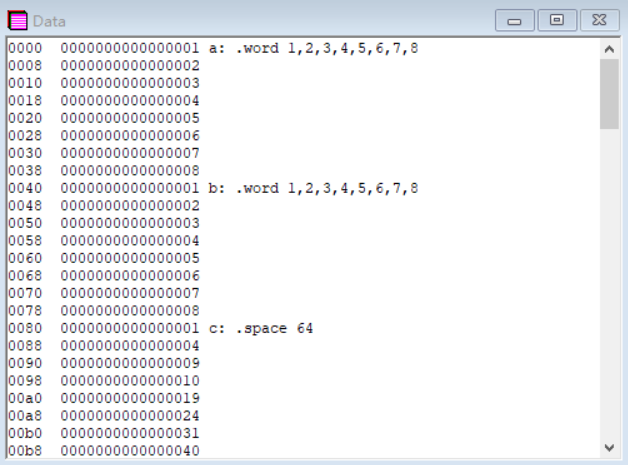
（3）在WinMIPS中运行该代码，记录数据如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 总时钟数 | 数据相关 | 资源相关 | 指令总数 | cpi |
| 168 | 64 | 8 | 93 | 1.806 |

（4）冲突指令以及相关如下**：**



（5）存储器内容如下：



**2、 采用指令调度技术解决流水线中的结构相关与数据相关**

（1）分析及改进思路

由上面的冲突指令分析得到，使用指令将mul.d与s.d指令隔开即可。由于乘法器延迟不一样，为了让s.d在mul.d指令执行完，将数据存入f6并且释放浮点寄存器后执行，需要多个语句来隔开mul.d和s.d，这样做同时解决了数据相关和资源相关问题。要解决l.d指令与mul.d指令的数据相关，只需要用一条指令将他们分隔开。

（2）代码如下：

文本

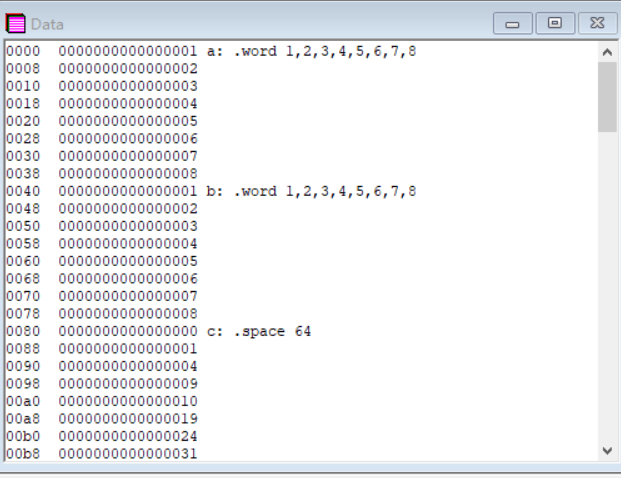
描述已自动生成

使用三条daddi指令来隔开mul.d和s.d指令，使用一条daddi指令来隔开l.d和l.d。

（3）数据记录如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 总时钟数 | 数据相关 | 资源相关 | 指令总数 | cpi | 提升倍数 |
| 修改前 | 168 | 64 | 8 | 93 | 1.806 | 1.313 |
| 修改后 | 128 | 16 | 8 | 93 | 1.376 |

（4）存储器内容如下：



（5）对比分析

通过对比可以看出，指令调度减少了部分的数据相关和资源相关，减少了产生气泡的时钟，提升了cpu的效率。

**3、 采用循环展开、寄存器换名以及指令调度**

（1）分析及改进思路

代码中出现了循环，因此可通过循环展开、寄存器换名的方法使得每一次循环的指令并行度提高。这个操作也许会带来数据相关和资源相关，因此再次使用指令调度的方法来进一步提高cpu的效率。

具体表现为一次循环内做四次乘法，并且由于mul.d与s.d会产生资源相关，用daddi指令将其分割开。

（2）代码如下：

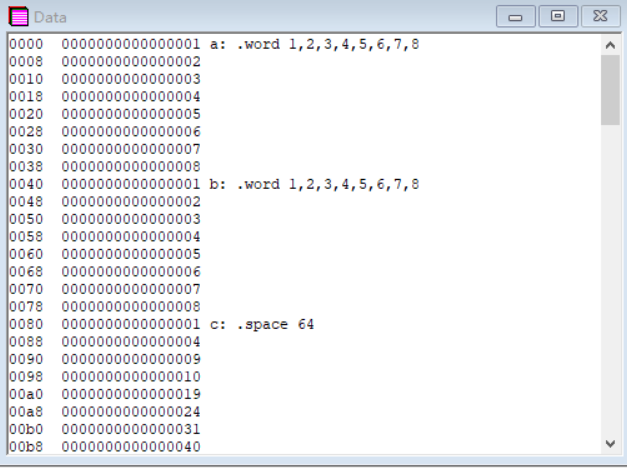
图片包含 文本

描述已自动生成

（3）数据记录如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 总时钟数 | 数据相关 | 资源相关 | 指令总数 | cpi | 提升倍数 |
| 修改前 | 168 | 64 | 8 | 93 | 1.806 | 1.498 |
| 修改后 | 76 | 0 | 8 | 63 | 1.206 |

（4）存储器内容如下：



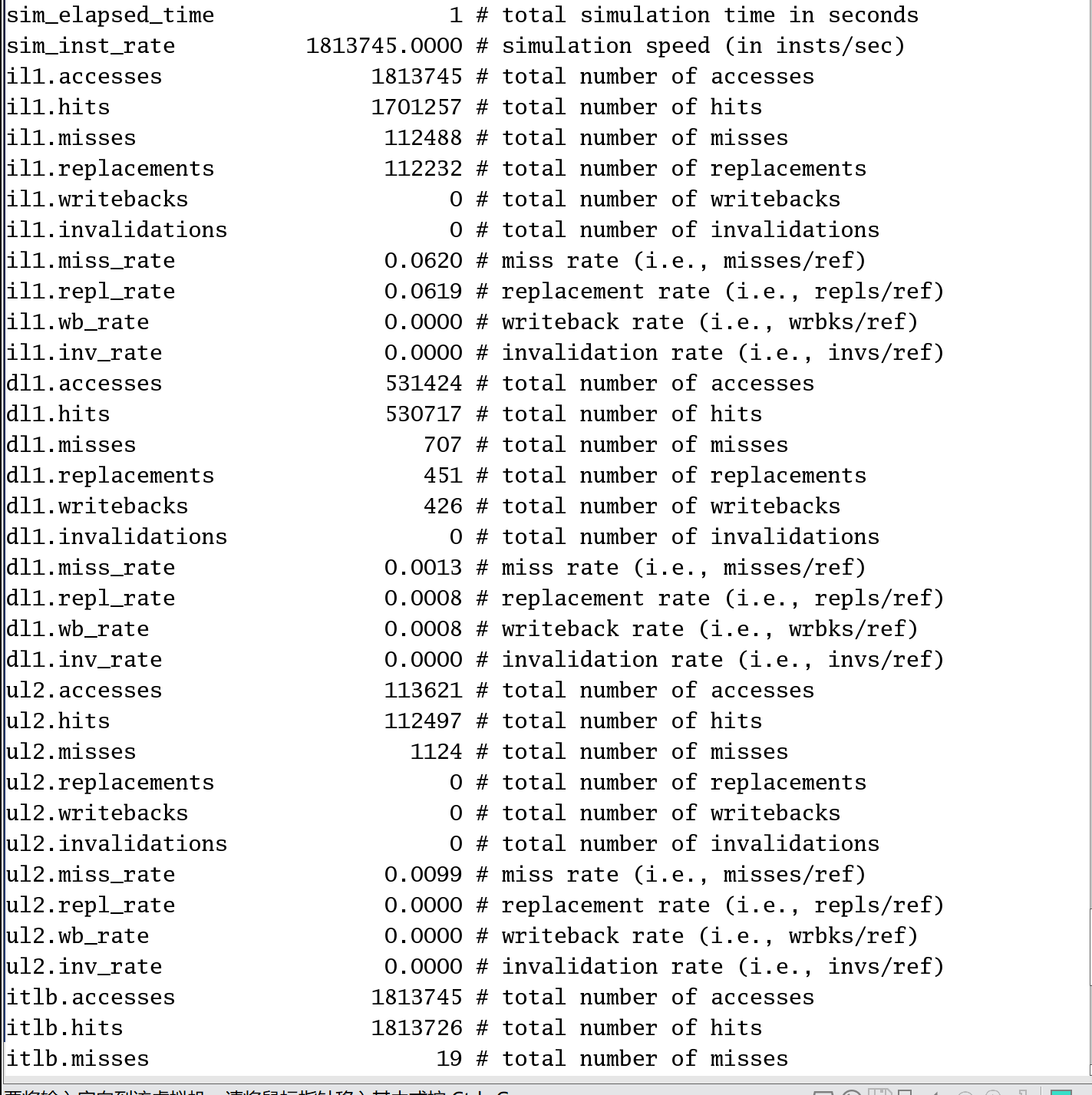
（5）对比分析

通过对比可以看出，循环展开进一步减少了cpu空转的次数，比起只使用指令调度来讲，cpu效率得到了进一步的提升。

# 实验三 Cache 性能分析

## 选用的测试程序及基本配置下的结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 基本配置 | | | | | |
| 程序 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| test-math | 57466 | 804 | 548 | 256 | 0.014 |
| test-printf | 531424 | 707 | 451 | 256 | 0.0013 |
| test-fmath | 16639 | 578 | 322 | 256 | 0.0347 |



图表 5 使用模拟器获取数据操作时的一个样例截图

## 二、 改变cache容量

（1）初始设置及数据记录

改变Cache容量这里是通过增加组数改变，按照要求逐倍递乘，其它的都不变。设置块大小32，组内块数1，替换算法为LRU，数据均记录如下。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-math 改变cache容量 | | | | | |
| 参数 | 访问次数 | 总失效次数 miss | 容量失效与冲突失效次数 replacement | 强制性失效次数miss-replacement | 失效率  miss\_rate |
| 8:32:1:l | 57466 | 12010 | 12002 | 8 | 0.2090 |
| 16:32:1:l | 57466 | 7708 | 7692 | 16 | 0.1341 |
| 32:32:1:l | 57466 | 4102 | 4070 | 32 | 0.0714 |
| 64:32:1:l | 57466 | 2030 | 1966 | 64 | 0.0353 |
| 512:32:1:l | 57466 | 560 | 138 | 422 | 0.0097 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-printf 改变cache容量 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 8:32:1:l | 531424 | 91328 | 91320 | 8 | 0.1719 |
| 16:32:1:l | 531424 | 45042 | 45026 | 16 | 0.0848 |
| 32:32:1:l | 531424 | 17618 | 17586 | 32 | 0.0332 |
| 64:32:1:l | 531424 | 9797 | 9733 | 64 | 0.0184 |
| 512:32:1:l | 531424 | 672 | 234 | 438 | 0.0013 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-fmath 改变cache容量 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 16:32:1:l | 16639 | 1812 | 1796 | 16 | 0.1089 |
| 32:32:1:l | 16639 | 1205 | 1173 | 32 | 0.0724 |
| 64:32:1:l | 16639 | 780 | 716 | 64 | 0.0469 |
| 128:32:1:l | 16639 | 644 | 516 | 128 | 0.0387 |
| 1024:32:1:l | 16639 | 469 | 1 | 468 | 0.0282 |

(2）分析Cache容量对Cache性能的影响

由上表的数据可看出：

## 随着Cache容量的增加，总的Cache失效次数减少，失效率也下降。

## 同时，随着Cache容量的增加，容量失效和冲突失效逐渐减少，而强制性失效次数则持续增加。

## 不同程序之间失效率的变化程度存在差异。

## 随着失效率逐渐减小，减小的速度也逐渐减缓。

## 三、改变cache相联度

(1) 初始设置及数据记录

改变Cache相联度这里是通过增加组内块数改变，按照要求逐倍递乘，同时为了保证容量不变，组数要应改变。设置块大小16，替换算法为LRU，数据均记录在下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-math 改变cache相联度 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:16:1:l | 57466 | 1034 | 204 | 830 | 0.018 |
| 512:16:2:l | 57466 | 1029 | 180 | 849 | 0.0179 |
| 256:16:4:l | 57466 | 1018 | 79 | 939 | 0.0177 |
| 128:16:8:l | 57466 | 1015 | 44 | 971 | 0.0177 |
| 16:16:64:l | 57466 | 1014 | 13 | 1001 | 0.0176 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-printf 改变cache相联度 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:16:1:l | 531424 | 1176 | 317 | 859 | 0.0022 |
| 512:16:2:l | 531424 | 1062 | 128 | 934 | 0.0020 |
| 256:16:4:l | 531424 | 1067 | 98 | 969 | 0.0020 |
| 128:16:8:l | 531424 | 1063 | 71 | 992 | 0.0020 |
| 16:16:64:l | 531424 | 1066 | 47 | 1019 | 0.0020 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-fmath 改变cache相联度 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:16:1:l | 16639 | 930 | 127 | 803 | 0.0559 |
| 512:16:2:l | 16639 | 930 | 119 | 811 | 0.0559 |
| 256:16:4:l | 16639 | 905 | 32 | 873 | 0.0544 |
| 128:16:8:l | 16639 | 901 | 10 | 891 | 0.0541 |
| 16:16:64:l | 16639 | 889 | 0 | 889 | 0.0540 |

(2）分析Cache相联度对Cache性能的影响

由上表的数据可看出：

## Cache相联度的增加对失效率的影响较小，表现为总失效次数和失效率略微降低，但变化并不显著。

## 随着Cache容量的增加，容量失效和冲突失效逐渐减少，与此同时，强制性失效次数不断增加。

## 在不同的程序中，虽然失效率存在差异，但变化幅度较小。

## 随着失效率逐渐减小，减小的速度也逐渐减缓。

## 四、改变Cache块大小

（1）初始设置及数据记录

改变Cache块大小这里是通过改变第二个参数，按照要求逐倍递乘，同时为了保证容量不变，组数要应对改变。设置组内块数为1，替换算法为LRU，数据均记录在下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-math 改变cache块大小 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:8:1:l | 57466 | 2160 | 1136 | 1024 | 0.0376 |
| 512:16:1:l | 57466 | 1201 | 689 | 512 | 0.0209 |
| 256:32:1:l | 57466 | 804 | 548 | 256 | 0.0141 |
| 128:64:1:l | 57466 | 623 | 495 | 128 | 0.0108 |
| 16:512:1:l | 1 | 1 | 0 | 1 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-printf 改变cache块大小 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:8:1:l | 531424 | 2192 | 1168 | 1024 | 0.0041 |
| 512:16:1:l | 531424 | 1197 | 685 | 512 | 0.0023 |
| 256:32:1:l | 531424 | 707 | 451 | 256 | 0.0013 |
| 128:64:1:l | 531424 | 767 | 639 | 128 | 0.0014 |
| 16:512:1:l | 1 | 1 | 0 | 1 | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-fmath 改变cache块大小 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:8:1:l | 16639 | 1844 | 820 | 1024 | 0.1108 |
| 512:16:1:l | 16639 | 987 | 475 | 512 | 0.0593 |
| 256:32:1:l | 16639 | 578 | 322 | 256 | 0.0347 |
| 128:64:1:l | 16639 | 394 | 266 | 128 | 0.0237 |
| 16:512:1:l | 1 | 1 | 0 | 1 | 1 |

（2）分析Cache块大小对Cache性能的影响

由上表的数据可看出：

## 随着Cache块大小的增加，Cache总失效次数减少，导致失效率降低。

## 随着Cache块大小的增加，容量失效、冲突失效和强制性失效次数均减少，这是因为增大的块大小更容易命中。

## 当块大小足够大时，出现总访问次数为1的情况，这可能表明块大小过大，导致程序执行出现问题。

## 随着失效率逐渐减小，减小速度逐渐减缓，表现为边际效应。

## 五、不同Cache容量下、不同替换算法

（1）初始设置及数据记录

改变的方法跟上面一样，设置组内块数为2，块大小32，逐倍递增组数，替换算法交替运行，数据均记录在下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-math 改变cache容量、替换算法 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 16:32:2:l | 57466 | 2838 | 2806 | 32 | 0.0494 |
| 16:32:2:r | 57466 | 3137 | 3105 | 32 | 0.0546 |
| 32:32:2:l | 57466 | 1394 | 1330 | 64 | 0.0243 |
| 32:32:2:r | 57466 | 1427 | 1363 | 64 | 0.0248 |
| 64:32:2:l | 57466 | 839 | 711 | 128 | 0.0146 |
| 64:32:2:r | 57466 | 882 | 754 | 128 | 0.0153 |
| 128:32:2:l | 57466 | 668 | 412 | 256 | 0.0116 |
| 128:32:2:r | 57466 | 693 | 454 | 239 | 0.0121 |
| 1024:32:2:l | 57466 | 541 | 0 | 541 | 0.0094 |
| 1024:32:2:r | 57466 | 541 | 7 | 534 | 0.0094 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-printf 改变cache容量、替换算法 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 16:32:2:l | 531424 | 12615 | 12583 | 32 | 0.0237 |
| 16:32:2:r | 531424 | 15999 | 15967 | 32 | 0.0301 |
| 32:32:2:l | 531424 | 2901 | 2837 | 64 | 0.0055 |
| 32:32:2:r | 531424 | 3408 | 3344 | 64 | 0.0064 |
| 64:32:2:l | 531424 | 812 | 684 | 128 | 0.0015 |
| 64:32:2:r | 531424 | 953 | 825 | 128 | 0.0018 |
| 128:32:2:l | 531424 | 597 | 341 | 256 | 0.0011 |
| 128:32:2:r | 531424 | 628 | 390 | 238 | 0.0012 |
| 1024:32:2:l | 531424 | 559 | 0 | 559 | 0.0011 |
| 1024:32:2:r | 531424 | 562 | 14 | 548 | 0.0011 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-fmath 改变cache容量、替换算法 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 16:32:2:l | 16639 | 932 | 900 | 32 | 0.056 |
| 16:32:2:r | 16639 | 969 | 937 | 32 | 0.0582 |
| 32:32:2:l | 16639 | 583 | 519 | 64 | 0.035 |
| 32:32:2:r | 16639 | 631 | 567 | 64 | 0.0379 |
| 64:32:2:l | 16639 | 511 | 383 | 128 | 0.0307 |
| 64:32:2:r | 16639 | 524 | 396 | 128 | 0.0315 |
| 128:32:2:l | 16639 | 501 | 245 | 256 | 0.0301 |
| 128:32:2:r | 16639 | 506 | 273 | 233 | 0.0304 |
| 1024:32:2:l | 16639 | 469 | 0 | 469 | 0.0282 |
| 1024:32:2:r | 16639 | 469 | 0 | 469 | 0.0282 |

（2）分析Cache块大小对Cache性能的影响

由上表的数据可看出：

## 在Cache的其它数据相同时，与随机替换算法相比，LRU替换算法的总失效次数较低，导致失效率下降，尽管降低幅度并不明显。

## 随着Cache容量的不断增大，LRU和随机替换算法之间的差距逐渐减小。这表明在足够大的容量下，替换算法的影响逐渐减弱。

## 两者在非强制失效次数上存在差异，而在强制失效次数上差异不大。

## 六、不同Cache相联度下、不同替换算法的实验结果与记录

（1）初始设置及数据记录

改变的方法跟上面一样，块大小为8，逐倍递组内块数，替换算法交替运行，数据均记录在下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-math 改变cache相联度、替换算法 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:8:1:l | 57466 | 2160 | 1136 | 1024 | 0.0376 |
| 1024:8:1:r | 57466 | 2160 | 1136 | 1024 | 0.0376 |
| 512:8:2:l | 57466 | 2026 | 1002 | 1024 | 0.0353 |
| 512:8:2:r | 57466 | 2063 | 1124 | 939 | 0.0359 |
| 256:8:4:l | 57466 | 2015 | 991 | 1024 | 0.0351 |
| 256:8:4:r | 57466 | 2044 | 1129 | 915 | 0.0356 |
| 128:8:8:l | 57466 | 2014 | 990 | 1024 | 0.0350 |
| 128:8:8:r | 57466 | 2046 | 1152 | 894 | 0.0356 |
| 16:8:64:l | 57466 | 2014 | 990 | 1024 | 0.0350 |
| 16:8:64:r | 57466 | 2064 | 1166 | 898 | 0.0359 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-printf 改变cache相联度、替换算法 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:8:1:l | 531424 | 2192 | 1168 | 1024 | 0.0041 |
| 1024:8:1:r | 531424 | 2192 | 1168 | 1024 | 0.0041 |
| 512:8:2:l | 531424 | 2115 | 1091 | 1024 | 0.0040 |
| 512:8:2:r | 531424 | 2169 | 1218 | 951 | 0.0041 |
| 256:8:4:l | 531424 | 2110 | 1086 | 1024 | 0.0040 |
| 256:8:4:r | 531424 | 2160 | 1224 | 1036 | 0.0041 |
| 128:8:8:l | 531424 | 2110 | 1086 | 1024 | 0.0040 |
| 128:8:8:r | 531424 | 2170 | 1252 | 918 | 0.0041 |
| 16:8:64:l | 531424 | 2109 | 1085 | 1024 | 0.0040 |
| 16:8:64:r | 531424 | 2159 | 1270 | 889 | 0.0041 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| test-fmath 改变cache相联度、替换算法 | | | | | |
| 参数 | 访问次数 | 总失效次数 | 容量失效与冲突失效次数 | 强制性失效次数 | 失效率 |
| 1024:8:1:l | 16639 | 1844 | 820 | 1024 | 0.1108 |
| 1024:8:1:r | 16639 | 1844 | 820 | 1024 | 0.1108 |
| 512:8:2:l | 16639 | 1807 | 783 | 1024 | 0.1086 |
| 512:8:2:r | 16639 | 1817 | 889 | 928 | 0.1092 |
| 256:8:4:l | 16639 | 1805 | 781 | 1024 | 0.1085 |
| 256:8:4:r | 16639 | 1807 | 918 | 889 | 0.1086 |
| 128:8:8:l | 16639 | 1805 | 781 | 1024 | 0.1085 |
| 128:8:8:r | 16639 | 1803 | 941 | 862 | 0.1084 |
| 16:8:64:l | 16639 | 1805 | 781 | 1024 | 0.1085 |
| 16:8:64:r | 16639 | 1807 | 951 | 856 | 0.1086 |

（2）分析Cache块大小对Cache性能的影响

由上表的数据可看出：

## 在Cache的其它数据相同的情况下，LRU与随机替换算法相比，总失效次数和失效率的差距不大。

## 随着Cache块大小的增大，LRU与随机替换算法之间的差距变化不大。这表明块大小对替换算法选择的影响较小。