刘思远 1100012713

黄亚蒙 1100012786

基于LRU替换策略的二路组相联Cache流水访问实现

Cache行为模拟作业报告

题目描述

在存储层次结构一节同学们学习了高速缓存 Cache 的相关知识。Cache 以局部性原理为其理论基础，一般通过行选择、Tag 匹配、读写数据三个步骤进行数据的读取与写入操作。本次大作业就是编写程序来实现对 Cache 的模拟，并在模拟中体现课堂中所学的Cache 的基本原理、组织结构、工作过程。

Cache原理

- 二路组相联Cache：Cache被逻辑分为两路，每路内的Cache是直接映射Cache。这样做的原因是，对于Cache的访问，可以先进行行选，然后并行检查两路中选出的两行，进行Tag的匹配，这样降低了冲突失效，并提高了访问Cache的效率。
- Cache替换策略（使用LRU）：由于二路组相联的设置，Cache对于同一个索引可以存在两路直接映射的位置。这样，当发生容量冲突的时候，需要替换其中的一行。LRU为最近最少使用，我们设计的Cache通过不断记录Cache的访问情况，启动基于LRU替换策略的选择器来进行Cache的替换。
- Cache的访问：Cache的访问我们实现了三种，分别是读、写、清除Cache。访问需要记录访问，为LRU替换提供条件。
  - 写：采用写返回的策略，先将内容写在Cache中，随后换出的时候再写回内存。
  - 清除Cache：将所有有效和“脏”的Cache块写回内存。

- 不同粒度的读写：支持设置不同粒度（8bit，16bit，32bit）的读写，在读写的时候，按要求控制读写长度即可。
- 读操作的扩展：提供符号扩展和无符号扩展的服务。比如，按8bit粒度读的时候，可以按符号扩展或者无符号扩展，得到相应的32bit数据。

程序设计

- Cache模拟
  - 代码结构

Package	功能
pku.ca.driver	**Cache最高层抽象**：包括启动Cache及一个时钟周期Cache的行为
pku.ca.cache	**Cache实现层抽象**：包括Cache的行选，Tag匹配，读写及清除操作
pku.ca.memory	**内存的模拟**：包括内存的读和写
pku.ca.addrparser	**访问地址解析器**：由一个16位的虚拟地址，解析成Tag，Index，Offset
pku.ca.replacement	**LRU替换选择器**：由Cache的访问情况，得到当前状态下替换的路号。
pku.ca.util	**请求/回应的抽象**：封装了请求和回应数据

  - Cache访问：三级流水

Cache请求的抽象：*pku.ca.util.Request.java*

成员变量	作用
int op	请求的操作：0表示读，1表示写，2表示清除Cache
int address	请求的地址：4位Tag，7位Index，5位Offset
byte []data	请求的数据：如果写的话，该变量有效，按字节存放写数据
int particle	请求的粒度：分别用8，16，32表示8bit，16bit，32bit的粒度
int ext	请求的扩展：0表示无符号扩展，1表示有符号扩展

- 行选：面对二路组相联Cache，得到当前访问的Cache行号，从而定位两个来自不同路的Cache行。
- Tag匹配：对选出的两个Cache行进行Tag匹配，如果有Tag匹配，说明Cache命中，如果没有Tag匹配，说明Cache失效，需要换入一行。

Cache回应的抽象：*pku.ca.util.Response.java*

成员变量	作用
int op	结束Cache访问的操作：0表示读，1表示写，2表示清除Cache
int address	结束Cache访问的地址：4位Tag，7位Index，5位Offset
int data	回应的数据：如果读的话，该变量有效，按字节存放写数据
int particle	结束Cache访问的粒度：分别用8，16，32表示8bit，16bit，32bit的粒度
int ext	结束Cache访问的扩展：0表示无符号扩展，1表示有符号扩展

- 操作执行：读、写、清除Cache，经过Tag匹配的操作，除了清除Cache操作，操作要访问的数据已经加载到了Cache中
  - 读：根据Request，按要求读出Cache的数据，并且构建Response返回
  - 写：根据Response，按要求写入Cache数据，并且构建Response返回
  - 清除Cache：将Cache中有效并且是“脏”块的数据写回Memory

- Cache的抽象结构：由代码的模块化和层级设计，我将Cache的实现分成了几级
  - 最底层Cache实现：实现对Cache Entry的操作——*pku.ca.cache.Cache.java*

成员方法	作用
boolean examValid(int numofgroup, int addrindex)	检查某一个Cache Entry的有效性
boolean examTag(int addrtag, int group, int index)	检查某一个Cache Entry的Tag是否与*addrtag*匹配
boolean getEntry(int addrindex, int numofgroup, CacheEntry entry)	返回一个请求的Cache Entry到*entry*中
boolean readEntryBytedata(int offset, int group, int index, byte []bytedata)	读相应的Entry data到*bytedata*中
boolean writeEntrybyte(int numofgroup, int addrindex, int offset, byte []writedata, int len)	将writedata中的数据写入相应的位置

  - 封装Cache访问操作的实现：将Cache的访问，封装成了上述的三部分，供后面抽象调用——*pku.ca.cache.CacheManager.java*

成员方法	作用
void cacheGroupSelect(Request request)	行选
void cacheBlockSelect()	Tag匹配确定某一路
boolean operateCache(Response response)	Cache操作
boolean readHandler(**Request request,** **Response response)**	读操作的实现：利用pku.ca.*cache.Cache.java*中的读抽象
boolean writeHandler(Request request)	写操作的实现：利用pku.ca.*cache.Cache.java*中写的抽象
void clearCacheAll()	清除Cache的实现
int missHandler(int address)	失效情况的实现：包括回写和返回写回路号
int extent(byte []bdata, int ext)	无符号和符号扩展的实现

- 最高级时钟周期Cache访问操作封装：Cache的初始化，及以时钟周期为单位的Cache操作——*pku.ca.driver.Driver.java*

成员方法	作用
void start(String filePath)	按指令文件*filePath*启动Cache
void oneTick()	一个时钟周期Cache行为的封装，其中包含*pku.ca.cache.CacheManeger.java* *cacheGroupSelect*， *cacheBlockSelect*， *operateCache* Cache流水便是不断执行该函数得到实现的

- Memory模拟
  - 代码结构：pku.ca.memory.Memory.java

成员	作用
byte []memorybyte	memory模拟的数据，以byte组织，顺序访问
boolean getBytes(int addr, int len, byte []retbyte)	memory读的模拟：从指定地址*addr*，读取*len*字节的数据到*retbyte*里
boolean setBytes(int addr, int len, byte []indata)	memory写的模拟：在指定地址*addr*，写入*len*字节数据*retbyte*

测试

测试数据：

附件: instr1.txt, instr2.txt, instr3.txt

instr1.txt	instr2.txt	instr3.txt
1. 义务失效，填充第一路 2. 清除 3. 失效，填充第一路 4. 命中，第一路，进行了符号扩展 5. 失效，填充第二路 6. 命中，第一路 7. 失效，替换第二路 8. 清除	1. 义务失效，填充第一路，8bit 2. 命中，第二路，16bit 3. 命中，第一路，32bit 4. 清除 5. 义务失效：填充第一路	1. 义务失效，填充并写第一路 0000 2. 失效，填充并写第二路 4000 3. 失效，替换并写第一路 8000 4. 失效，填充并读第二路 0000 5. 失效，填充并读第一路 4000 6. 失效，填充并读第二路 8000 7. 清除 8. 失效，填充并读第一路 8000

指令结构：

op code	address	data	particle	extend
访问操作 0：读 1：写 2：清除Cache	访问地址： 16位地址空间	写入数据： 32位写入数据	粒度 8：8bit 16：16bit 32：32bit	扩展： 0：无符号扩展 1：符号扩展

测试结果

首先，由于测试数据根据按照各种Cache的逻辑通路设计，达到了测试各种Cache访问及Cache流水访问的情况，其次，测试最终取得了正确的结果

用户使用手册

（一）启动程序：建立Eclipse工程，运行工程，启动程序界面

1.输入文件名：绝对路径，如instr1.txt

2.点击Start按钮，启动Cache

3.不断点击Next Cycle可以看到Cache流水访问的过程

附件

1. Cache流程图：CacheGraph
2. 代码包：

   2.1.Cache实现代码

   2.2.测试数据