Programming Assignment 3 - Cache
Design Document

Group 55 - Maxwell Bruce, Samuel Woolledge, Jiapei Kuang

General idea:

To have an instruction cache (I-cache) and a data cache (D-cache) to the pipelined RISC-V CPU.

Cache- overview:

All the implementations for cache are in the cache.* files with the direct mapped cache and extra-credit (two-set associative cache) implemented. There are four functions specified, access_read, evict_read, access_write, and evict_write.

Cache - structure:

There are four defined struct for cache row. Each row contains dirty bit, tags, valid bit, and cache data.

Cache table contains cache rows and some other parameters regarding the cache such as length of tags, index, and number of blocks. In addition, there are variables for defining a two-way associative write back cache.

Cache - direct mapped cache:

First, we will construct a cache table with each cache row assigned.

There are two functions for read and write access: For read_access, if there is a valid bit and the tags are matching, there is a hit and it will return data of the cache.

For write_access, if there is a valid bit and the tags are matching, it is a hit and it will access the data. And for both functions, if there is a cache miss, then the function will call evict functions.

In evict functions, for evict_read, it will input new data into the tag and valid bits. And for evict_write, it will input new data and input an additional data, dirty bit.

Cache - two-way set associative write-back cache:

First, we will construct a cache table with each cache row assigned. And for two-way set associative cache, we will twice the number of blocks.

The access functions work similarly as the direct mapped cache. For read_access, if the function notices it is a two-way set associative cache, and if there is a valid bit and the tags are matching, it is a hit and the data will be returned. For write_access, if the function notices it is a two-way set associative cache, and if there is

a valid bit and the tags are matching, it is a hit and it will set the data. And if the cache miss, both function will call evict_write or read.

For evict_read and evict_write, if the functions notice that the cache is a two-way set associative cache, they will choose specific index to be evicted. And then put in new data.

Cache - write back:

If write_access function indicates a writeback occur, then the evict functions will handle it. If there is a write back and a dirty bit, and if there is a hit, they will clear the dirty bit, if not, they will evict the index data (for read) or return 0 (for write) .