ECED 3401 A2 Cache design

Problem Statement

The aim of the project is to design, implement, and test a cache memory emulator for the XM23 emulator to demonstrate the effects of different cache organizations and two cache replacement policies. The cache emulator will enable us to reduce CPU memory-access times by storing data and instructions in the cache, thereby increasing the overall speed of the machine.

Algorithm

Cache Organization

The cache memory is made up of 32 lines, each containing the address of the contents of the primary memory location and the contents of the location itself. We will emulate two types of cache organization:

1. **Associative Cache:** In this organization, the cache circuitry inspects each cache line for the target address. There is no relationship between the target address and its location in the cache memory.
2. **Direct Mapping:** In this organization, the target address is mapped directly into a cache line. There is a direct relationship between the target address and its location in the cache memory.

Cache Replacement Policies

We will implement two replacement policies:

1. **Least Recently Used (LRU):** In an associative cache, this policy replaces the least recently used line in the cache when a cache miss occurs.
2. **Write Back (WB) vs Write Through (WT):** When a write operation results in a hit, the cache needs to be updated. In WB, the contents are written back to primary memory. In WT, the value written to the cache is written to primary memory in parallel, ensuring both the cache and primary memory are consistent.

Data Dictionary

Here are the major data structures used:

1. **cache_line (Structure)**: Represents a line of cache with the following fields:
    - address (unsigned short): The primary memory address of the data.
    - contents (unsigned short): The actual data stored.
    - dirty (int): Bit to signify if the cache line has been modified.
    - age (int): Used for the LRU (Least Recently Used) cache eviction policy.
    - used (int): Indicates if the cache line is in use.
2. **MappingType (Enum)**: Specifies the cache memory mapping technique.

- DIRECT: Direct Mapping.
- ASSOCIATIVE: Associative Mapping.

3. **WritePolicy (Enum)**: Specifies the cache memory write policy.
    - WRITE_BACK: Cache line is updated first, primary memory is updated later.
    - WRITE_THROUGH: Cache line and primary memory are updated simultaneously.
4. **cache_mem (Array)**: The cache memory, an array of cache_line structures.
5. **mapping_type (MappingType variable)**: Global variable that defines the mapping policy used.
6. **write_policy (WritePolicy variable)**: Global variable that defines the write policy used.
7. **hit_count (int variable)**: Global variable that counts the number of cache hits.
8. **miss_count (int variable)**: Global variable that counts the number of cache misses.
9. **Functions**:
    - cache: Entry point to the cache, takes memory address register, memory buffer register, read/write flag, and byte/word flag.
    - cache_associative: Implements the cache operation for associative mapping using LRU for cache line replacement.
    - cache_direct: Implements the cache operation for direct mapping.
    - update_usage: Updates the age of cache lines on cache hit.
    - print_cache: Prints the current cache contents and the hit and miss counts.