实验报告

一、MESI协议概述

MESI协议是一种用于维护多处理器系统中缓存一致性的协议，定义了缓存块的四种状态：

1. Modified (M)：数据已被修改，与主存不一致，仅当前缓存拥有最新数据
2. Exclusive (E)：数据与主存一致，仅当前缓存拥有该数据
3. Shared (S)：数据与主存一致，多个缓存可能共享该数据
4. Invalid (I)：缓存块无效，不包含有效数据

协议通过总线事务协调各缓存状态：

- 读缺失(READ_MISS)：请求其他缓存提供数据
- 写缺失(WRITE_MISS)：通知其他缓存失效
- 置无效(SET_INVALID)：强制其他缓存块失效

请求	源	所寻址缓存块的状态	缓存操作的类型	功能与解释
读命中	处理器	共享或已修改或独占	正常命中	读取本地缓存中的数据
读缺失	处理器	无效	正常缺失	将读缺失放置在总线上
读缺失	处理器	共享	替换	地址冲突缺失：将读缺失放置在总线上
读缺失	处理器	独占	替换	地址冲突缺失：将读缺失放置在总线上
写命中	处理器	独占	正常命中	在本地缓存中写入数据，转换成已修改
读缺失	处理器	已修改	替换	地址冲突缺失：写回块，然后将读缺失放置在总线上
写命中	处理器	已修改	正常命中	在本地缓存中写入数据
写命中	处理器	共享	一致性	将无效操作放置在总线上。这些操作通常称为更新或拥有者缺失，因为它们不能提取数据，只能改变状态
写缺失	处理器	无效	正常缺失	将写缺失放置在总线上
写缺失	处理器	共享	替换	地址冲突缺失：将写缺失放置在总线上
写缺失	处理器	已修改	替换	地址冲突缺失：写回块，然后将写缺失放置在总线上
写缺失	处理器	独占	替换	地址冲突缺失：将写缺失放置在总线上
读缺失	总线	共享	无操作	允许共享缓存或存储器为读缺失提供服务

请求	源	所寻址缓存块的状态	缓存操作的类型	功能与解释
读缺失	总线	已修改	一致性	尝试读取共享数据：将缓存块放置在总线上，写回块，并将状态改为共享
读缺失	总线	独占	一致性	尝试读取共享数据：将缓存块放置在总线上，并将状态改为共享
无效	总线	共享	一致性	尝试写共享块，使该块无效
写缺失	总线	共享	一致性	尝试写共享块，使缓存块无效
写缺失	总线	已修改	一致性	尝试将独占块写到其他位置，写回该缓存块，并在本地缓存中使其状态失效
写缺失	总线	独占	一致性	尝试将独占块写到其他位置，并在本地缓存中使其状态失效

二、代码实现

2路组相联、容量为64B、lru策略的cache，cacheline大小为4B

容量为8KB的主存

基于总线监听的MESI缓存一致性协议

(1) 状态、cache、memory、bus类定义

```
// MESI状态定义（枚举）
enum State {
    MODIFIED,    // M状态 - 数据已修改
    EXCLUSIVE,   // E状态 - 独占且干净
    SHARED,      // S状态 - 共享
    INVALID      // I状态 - 无效
};

// 总线请求类型
enum BusRequest {
    READ_MISS,     // 读缺失
    WRITE_MISS,    // 写缺失
    SET_INVALID    // 置无效
};
```

```
class Cache {
private:
    struct Block;
    struct Set;

    std::vector<Set> cache;
    const int block_size = 4;
    int access_count = 0;
    Bus *bus;
    Memory *memory;

public:
    int processor_id;
    int read_hit_count = 0;
    int read_miss_count = 0;
    int write_hit_count = 0;
    int write_miss_count = 0;

    Cache(int id);
    uint32_t handleBusRequest(BusRequest request, uint16_t address, int
source_id, bool *shared = nullptr);
    bool access(uint16_t address, Operation op, uint16_t write_data = 0,
uint16_t* read_data = nullptr);
    void print_state();
    void setBus(Bus *b) { bus = b; }
    void setMemory(Memory *m) { memory = m; }
};
```

```
class Memory {
private:
    std::vector<uint32_t> data;
    const int size_KB = 8;
    const int block_size = 4;

public:
    Memory();
    uint32_t readOneBlock(uint16_t address);
    void writeOneBlock(uint16_t address, uint32_t value);
    void printState(int start = 0, int end = 10);
};
```

```
class Bus {
private:
    std::vector<Cache *> caches;
    Memory *memory;

public:
    Bus(std::vector<Cache *> &caches, Memory *memory);
    uint32_t broadcast(BusRequest request, uint16_t address, int source_id, bool
*shared = nullptr);
};
```

通过`Cache::handleBusRequest`和`Bus::broadcast`两个函数配合实现来自总线的请求

通过`Cache::access`处理来自本地的请求

(2) 关键操作流程

```
bool Cache::access(uint16_t address, Operation op, uint16_t write_data,
uint16_t* read_data) {
    access_count++;
    uint16_t offset = address & 0x3;
    uint16_t index = (address >> 2) & 0x7;
    uint16_t tag = (address >> 5) & 0xFF;

    Set &set = cache[index];
    bool hit = false;
    int block_index = -1;

    for (int i = 0; i < 2; i++) {
        if (set.blocks[i].state != INVALID && set.blocks[i].tag == tag) {
            hit = true;
            block_index = i;
            break;
        }
    }

    Block *block_ptr = nullptr;
    if (hit) {
        block_ptr = &set.blocks[block_index];
        block_ptr->lru_counter = access_count;
        if (op == READ) {
            if (block_ptr->state == SHARED || block_ptr->state == EXCLUSIVE ||
block_ptr->state == MODIFIED) {
                if (read_data != nullptr) {
                    *read_data = block_ptr->readTwoBytes(offset);
                }
            }
            read_hit_count++;
        } else {
            if (block_ptr->state == SHARED) {
                block_ptr->state = MODIFIED;
                block_ptr->writeTwoBytes(offset, write_data);
                bus->broadcast(SET_INVALID, address, processor_id);
            } else if (block_ptr->state == EXCLUSIVE || block_ptr->state ==
MODIFIED) {
                block_ptr->state = MODIFIED;
                block_ptr->writeTwoBytes(offset, write_data);
            }
            write_hit_count++;
        }
    } else {
        int lruTarget = 0;
        if (set.blocks[0].state == INVALID || set.blocks[1].state == INVALID) {
            lruTarget = (set.blocks[0].state == INVALID) ? 0 : 1;
        } else {
            lruTarget = (set.blocks[0].lru_counter < set.blocks[1].lru_counter)
? 0 : 1;
        }
        Block& victim = set.blocks[lruTarget];

        if (op == READ) {
            if (victim.state == MODIFIED) {
                memory->writeOneBlock(address, victim.data);
            }
            bool shared = false;
            uint32_t response_data = bus->broadcast(READ_MISS, address,
processor_id, &shared);
            if (shared) {
                victim.state = SHARED;
                victim.tag = tag;
                victim.lru_counter = access_count;
                victim.data = response_data;
            } else {
                victim.state = EXCLUSIVE;
                victim.tag = tag;
                victim.lru_counter = access_count;
                victim.data = memory->readOneBlock(address);
            }
            if (read_data != nullptr) {
                *read_data = victim.readTwoBytes(offset);
            }
            read_miss_count++;
        } else {
            if (victim.state == MODIFIED) {
                memory->writeOneBlock(address, victim.data);
            }
            bus->broadcast(WRITE_MISS, address, processor_id);
            victim.state = MODIFIED;
            victim.tag = tag;
            victim.lru_counter = access_count;
            victim.data = memory->readOneBlock(address);
            victim.writeTwoBytes(offset, write_data);
            write_miss_count++;
        }
    }
    return hit;
}
```

对于本地的读操作（READ）

- 命中时：直接读取数据（S/E/M状态），更新LRU
- 缺失时：
    1. 选择替换块（LRU或INVALID块）
    2. 若替换块为M状态，写回内存
    3. 广播READ_MISS请求
    4. 根据响应确定状态：
        - 其他缓存有数据 → 设为SHARED状态（使用响应数据）
        - 无其他缓存 → 设为EXCLUSIVE状态（从内存读取）

对于本地写操作（WRITE）

- 命中时：
    - SHARED状态：广播SET_INVALID，升级为MODIFIED
    - EXCLUSIVE/MODIFIED：直接升级为MODIFIED
- 缺失时：

1. 选择替换块（M状态需写回）
2. 广播WRITE_MISS（使其他缓存失效）
3. 从内存加载数据块
4. 修改数据并设为MODIFIED状态

```cpp
uint32_t Cache::handleBusRequest(BusRequest request, uint16_t address, int
source_id, bool *shared) {
    if (source_id == processor_id) {return 0;}

    uint16_t index = (address >> 2) & 0x7;
    uint16_t tag = (address >> 5) & 0xFF;
    Set &set = cache[index];

    for (Block &block : set.blocks) {
        if (block.tag == tag) {
            switch (request) {
                case READ_MISS: {
                    if (block.state == SHARED || block.state == EXCLUSIVE) {
                        block.state = SHARED;
                        if (shared != nullptr) {
                            *shared = true;
                        }
                        return block.data;
                    } else if (block.state == MODIFIED) {
                        memory->writeOneBlock(address, block.data);
                        block.state = SHARED;
                        if (shared != nullptr) {
                            *shared = true;
                        }
                        return block.data;
                    }
                    break;
                }
                case WRITE_MISS: {
                    if (block.state == SHARED || block.state == EXCLUSIVE) {
                        block.state = INVALID;
                        return 0;
                    } else if (block.state == MODIFIED) {
                        memory->writeOneBlock(address, block.data);
                        block.state = INVALID;
                        return 0;
                    }
                    break;
                }
                case SET_INVALID: {
                    block.state = INVALID;
                    return 0;
                }
                default: break;
            }
        }
    }
    return 0;
}
```

对于总线的读缺失（READ_MISS）

当前状态	动作
SHARED	保持SHARED状态，返回块数据
EXCLUSIVE	转为SHARED状态，返回块数据
MODIFIED	将数据写回内存，转为SHARED状态，返回块数据
INVALID	无操作

提供最新数据副本，确保读请求方可以进入SHARED状态，维持多缓存一致性。

对于总线的写缺失（write_miss）

当前状态	动作
SHARED	立即失效（INVALID）
EXCLUSIVE	立即失效（INVALID）
MODIFIED	将数据写回内存，设为INVALID状态
INVALID	无操作

对于总线的无效（SET_INVALID）

当前缓存块设置为INVALID。强制清除其他缓存的副本，确保写操作独占性（MODIFIED）。

三、测试结果

T - tag, S - state, D - data, L - LRU_tag

Normal.txt

```
P0 W 0x1000 0x27
         Cache State (Processor 0):
Set 0:  [T:0x80 S:M D:0x00000027 L:1]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P2 R 0x1000 0xffff
Cache State (Processor 2):
Set 0:  [T:0x80 S:S D:0x00000027 L:1]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]
```

```
P3 R 0x1000 0xffff
Cache State (Processor 3):
Set 0:  [T:0x80 S:S D:0x00000027 L:1]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P0 W 0x1000 0x36
Cache State (Processor 0):
Set 0:  [T:0x80 S:M D:0x00000036 L:2]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P3 W 0x1000 0x28
Cache State (Processor 3):
Set 0:  [T:0x80 S:M D:0x00000028 L:2]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P1 R 0x1000 0xffff
Cache State (Processor 1):
Set 0:  [T:0x80 S:S D:0x00000028 L:1]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P2 W 0x1000 0x22
Cache State (Processor 2):
Set 0:  [T:0x80 S:M D:0x00000022 L:2]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]
```

false+sharing.txt

```
P0 W 0x1000 0x100
Cache State (Processor 0):
Set 0:  [T:0x80 S:M D:0x00000100 L:1]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P1 W 0x1004 0x200
Cache State (Processor 1):
Set 0:  [INVALID]                       [INVALID]
Set 1:  [T:0x80 S:M D:0x00000200 L:1]   [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P2 W 0x1008 0x300
Cache State (Processor 2):
Set 0:  [INVALID]                       [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [T:0x80 S:M D:0x00000300 L:1]   [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]

P3 W 0x100c 0x400
Cache State (Processor 3):
Set 0:  [INVALID]                       [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [T:0x80 S:M D:0x00000400 L:1]   [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]
```

random.txt

```
P0 W 0x1000 0x100
Cache State (Processor 0):
Set 0:  [T:0x80 S:M D:0x00000100 L:1]   [INVALID]
Set 1:  [INVALID]                       [INVALID]
Set 2:  [INVALID]                       [INVALID]
Set 3:  [INVALID]                       [INVALID]
Set 4:  [INVALID]                       [INVALID]
Set 5:  [INVALID]                       [INVALID]
Set 6:  [INVALID]                       [INVALID]
Set 7:  [INVALID]                       [INVALID]
```

```
P1 W 0x1004 0x200
Cache State (Processor 1):
Set 0:  [INVALID]                         [INVALID]
Set 1:  [T:0x80 S:M D:0x00000200 L:1]    [INVALID]
Set 2:  [INVALID]                         [INVALID]
Set 3:  [INVALID]                         [INVALID]
Set 4:  [INVALID]                         [INVALID]
Set 5:  [INVALID]                         [INVALID]
Set 6:  [INVALID]                         [INVALID]
Set 7:  [INVALID]                         [INVALID]

P2 W 0x1008 0x300
Cache State (Processor 2):
Set 0:  [INVALID]                         [INVALID]
Set 1:  [INVALID]                         [INVALID]
Set 2:  [T:0x80 S:M D:0x00000300 L:1]    [INVALID]
Set 3:  [INVALID]                         [INVALID]
Set 4:  [INVALID]                         [INVALID]
Set 5:  [INVALID]                         [INVALID]
Set 6:  [INVALID]                         [INVALID]
Set 7:  [INVALID]                         [INVALID]

P3 W 0x100c 0x400
Cache State (Processor 3):
Set 0:  [INVALID]                         [INVALID]
Set 1:  [INVALID]                         [INVALID]
Set 2:  [INVALID]                         [INVALID]
Set 3:  [T:0x80 S:M D:0x00000400 L:1]    [INVALID]
Set 4:  [INVALID]                         [INVALID]
Set 5:  [INVALID]                         [INVALID]
Set 6:  [INVALID]                         [INVALID]
Set 7:  [INVALID]                         [INVALID]
```