In the name of God

Computer Architecture Final Project – Fall 2023

Deadline: Jan 30th

Parallel Instruction Execution with MSI Cache Protocol

This project aims to design and simulate a simplified processor architecture implementing parallel instruction execution across stages while integrating the MSI cache coherence protocol to manage cache states in a single processor environment.

Consider these points while implementing:

1. Processor Model Design:

- Develop a simplified processor architecture with pipelined stages: Fetch, Decode, Execute, Memory, Write-back.
- Enable parallel execution of instructions across stages to maximize throughput.
 (report system throughput at last)
- Printing the activities happening in each stage of the CPU and detailing the status of instructions in each clock cycle is necessary. Track the status of each instruction being processed:
 - Fetching: Indicate the instruction being fetched and its address.
 - Decoding: Report the decoded instruction and its operands.
 - Execution: Display the execution process, including arithmetic or logical operations.
 - Memory Access: If applicable, note memory access details (read/write).
 - Write-back: Log the write-back process and register updates.

2. MSI Cache Integration:

This is a basic level one 2-way set-associative cache coherence protocol used in multiprocessor system. The letters of protocol name identify possible states in which a cache can be. So, for MSI each block can have one of the following possible states:

Modified

The block has been modified in cache, i.e., the data in the cache is inconsistent with the backing store (memory). So, a cache with a block in "M" state has responsibility to write the block to backing store when it is evicted.

Shared

This block is not modified and is present in at least one cache. The cache can evict the data without writing it to backing store.

Invalid

This block is invalid and must be fetched from memory or from another cache if it is to be stored in this cache.

3. Synchronization and Communication:

- Establish synchronization mechanisms between pipeline stages.
- Implement communication channels for coherence operations using the MSI protocol. Reporting the cache status, including the values of its inner flags, in each clock cycle is necessary.
- o Update the cache status based on the ongoing operations. For instance:
 - If a cache hit occurs, update the status flags accordingly (e.g., set valid bit, update usage information).
 - On a cache miss, update flags (e.g., update replacement policies, mark blocks as invalid or load new data).

Bonus:

Extend the single-processor project to a multi-processor (2) environment to showcase the benefits and challenges of parallelism and cache coherence in a distributed computing setting. Implement mechanisms for synchronization and data exchange between processor cores.