**Cache Controller Design**

**Daniel Komac**

**CPE 186 Spring 2016**

**Professor Vadhva**

**April 14, 2016**

**Table of Contents:**

1. Main Memory
2. Cache
3. Cache sets 2K bytes each
4. 4 way set associative
5. Block Size 64LW(526bytes)
6. 4 Lines per block
7. Line size 16LW(64 bytes)
8. LRU policy
9. 8-bit and 16-bit non-cacheable
10. I/O transfers are non-cacheable
11. Read cache miss LW acces
    1. LW 0 – LW12
       1. Transfer will start from the address accessed till the end of the line boundary and then wrap around to the address
       2. The first data is transferred to the CPU at the same time to the cache.
    2. LW 13 – 14 non-cacheable, however the requested data is transferred to the CPU
    3. LW 15
       1. Non-cacheable, data transferred to the CPU
       2. After CPU receives this LW, entire next consecutive line is transferred to the cache.
12. Cache Write
    1. Cache Write miss is non-cacheable
    2. Cache write hit- write back policy
13. Bus Snooping circuits – MESI protocol
14. Memory pages 100 to 200 are non-cacheable
15. Flush circuitry to invalidate the entire Cache
16. **Main Memory**

The Main memory is the area in which all data is being held. Our permitted memory size is 4 Mbytes which in Long word translate to 1 M LW (Mega Long Word). Now we can splice this memory into 2000 pages which is considering the caches. Now this memory is 4 way set associative so 4 sets must be incorporated which means each set will handle 500bytes in this page. That means that there are 4 rows, each a set with 500 bytes of information.



Main memory is also connected to the processors, four processors total, each with an equivalent cache size of 2K bytes(512 LW)



**2. Cache Size**

The cache is designed in the same manner, several pages per cache this is approximately 8 Kbytes of data. Each cache is designed to be 2 Kbytes of data because of 4 way set associative rule. The pages of caches hold 64 LW (256 bytes) of info. The cache is grouped in with processor and is used to make memory access much more available.

Both connected to each other to allow quick transfer of data.

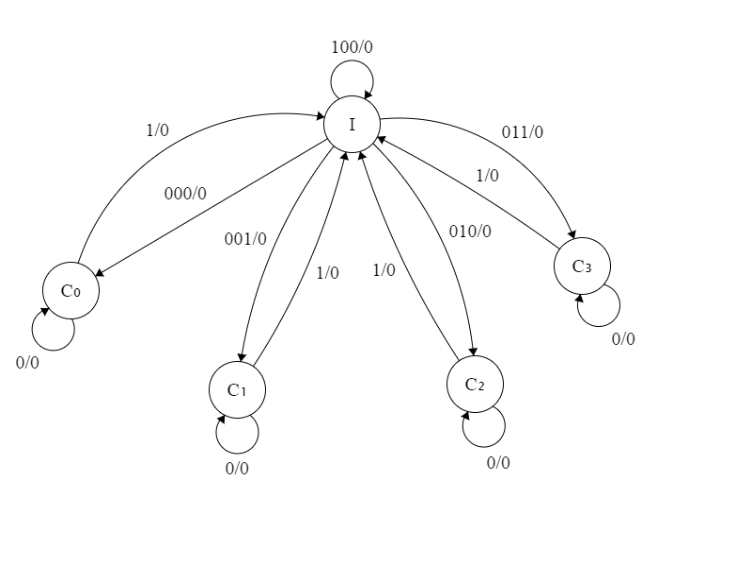


Cache is able to be very useful at the expense of being expensive. Cache is also granted according to who needs access to the bus.

**3. 4 Way set Associative:**

Four way set association is the process in which the memory of the cache is split amongst four specified sets. These sets will give the cache the ability to take a fourth of the page and assign each individual cache a segment. The memory segments are split into 4 lines, the Block size correlates to how each cache will work. These splits allow for the separate storage of memory but storing memory like this requires a comparator which will check each set to make sure there aren’t duplicate information.



The cache being split into different segments allows for separation of materials and avoiding redundant information in specific sets. Having this set up uses more resources but allows for an efficient means of storing info. The cache line that the bus/ cache controller uses contains a tag along with a verification giving the bus the access to the specified set which would be tagged. The tag allows a value to be hold of the specific set allowing for read write capabilities and other capabilities.

**4. 2Kbytes Set Cache**

Each set being 2k allows for Set cache to operate as an entire entity of cache. The total Cache size is 8Kbytes which means that all 4 Caches can split this page of Cache equally and retain all the same capabilities. When cache is segmented into sets there are additional resources needed and precautions taken.

Each cache is split into this amount of memory because it is one fourth of the cache, being able to split data into smaller pages, a 4 way comparator is still needed and the more caches we can implement the closer the system will become to fully associative.

Each 2k slice of Cache will contain a unique tag so they can be uniquely addressed and they will have the same amount of memory as other caches as so the memory can readily be used for easy transactions from cache to cache, or sharing of data between any other processes that need access to the cache.

The reasoning behind split memory is for easier management at the expense of more materials, the lookup penalties do increase simply because there are more entries that have to be managed in order to assure that the right information is being attained. 

Allowing for unique memory addresses gives more secure uses of holding data. Especially since cache is rapidly updated over and over as processes happen.

**5. Block Size**

The block size for the cache will be 64 LW which is roughly 256 bytes. The long word consists of 32 bits and uses all available bits for information relevant for the cache being operated, I.E. tags, data, and relative address. In this cache controller it will need a set field, 2 bits, an address, and data. All able to be organized with in the lines of each section of the cache.



**6. 4 Lines per Block:**

Within each block of memory that is being held in the cache it must be segmented into 4 lines for 4 way set associativity. We do this by assigning special bit amounts to each unique line. Also needing comparators to check to make sure there are no information that is being shared or is going to throw an invalid flag within the MESI protocol.

This allows us to identify chucks with bit sets, as labeled on the side of each line. These lines will contain unique information and allow for quick transportation between data at the cost of some extra materials. It is also worth noting that the more associativity cache blocks gain the closer it becomes to fully associative.

**7. Line Size 16 LW:**

Each address line within the line will hold 4 bytes which is equivalent to 16 bits. All the information will be stored with a generic address, i.e. line address, set number or unique id, and data. These are usually held in the order of the upper bits being the address while the lower bits hold the data. This means that we’ll be able to hold information in a four way associative cache diagram.