**Introduction**

The main bottleneck in a CPU is the memory access. There are different memory levels based on speed, size, and cost. SRAM is memory that is faster, but smaller and more expensive. DRAM sits outside of the CPU. It can hold more data, but the access time is much slower. The memory hierarchy system is set up such that the CPU first checks if the data is in the SRAM. If it is not, then it is registered as a miss and has to access the DRAM, which requires more stalls. The goal of the memory system and cache design is to decrease the number of misses as much as possible.

**Design Methodology**

Two main ways to decrease the number of cache misses is to increase the block size and the associativity. On a miss, instead of just transferring the one word that is needed for the instruction, multiple adjacent words will get transferred instead. Since arrays are commonly stored in memory, transferring multiple words at a time will decrease the miss rate when looping through the array. Increasing the number of ways in a cache decreases the number of conflict misses. In a direct mapped cache, if a set is accessed a second time, but with a different tag, then the old data in the cache needs to be replaced with the other data in the main memory. For multiple ways, least recently used replacement policy is used when swapping between cache and main memory. The block that was not used for the longest time is the one that is replaced.

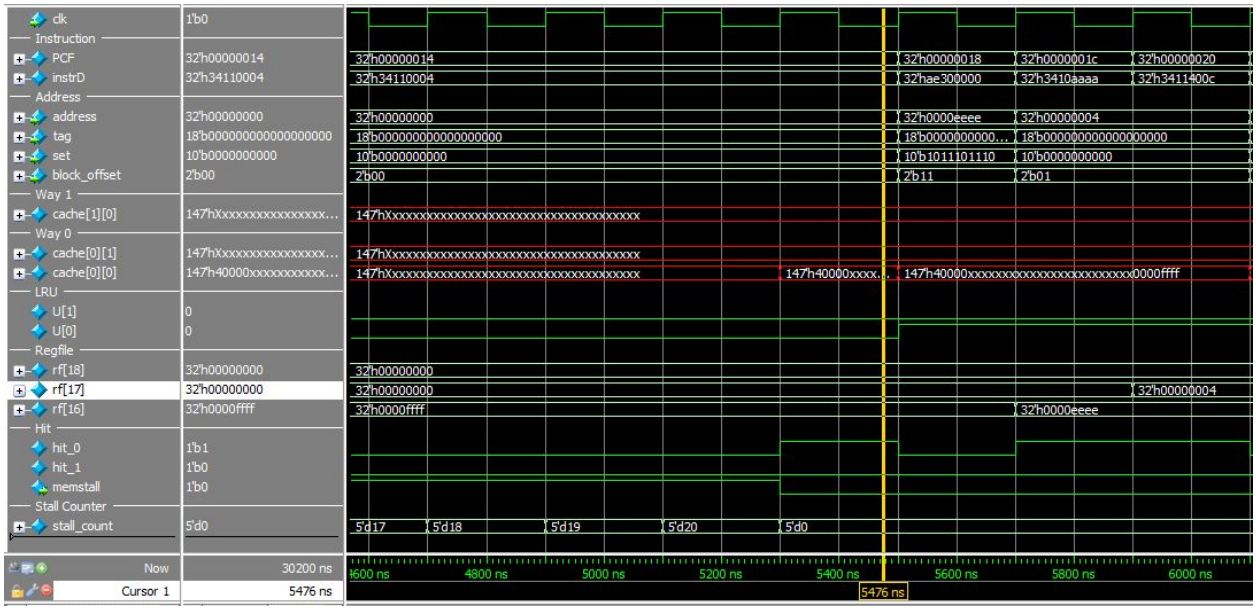
This cache specifically is a two-way set associative 32KB cache with 4-word blocks. Each word is four bytes long, so each set is 32 bytes long. There are 1024 sets in this cache. The memory address is divided starting with two for the byte offset, two for block offset, ten for the set, and the rest for the tag. Given a memory address, the cache will go to the corresponding set and check if the data is valid and if the stored tag is equal to the given tag. On a hit, the read and write operations will function like normal and the LRU bit will change to the other way. On a miss, the data labeled with the LRU bit in the cache will be swapped with corresponding data in main memory on the first cycle. The processor will stall for 20 cycles for the memory to transfer from main memory to the cache., so there is an internal counter in the memory that counts up to

20. On the next cycle, if write is enabled, the write data will be written into only the cache since we are using a write back policy.

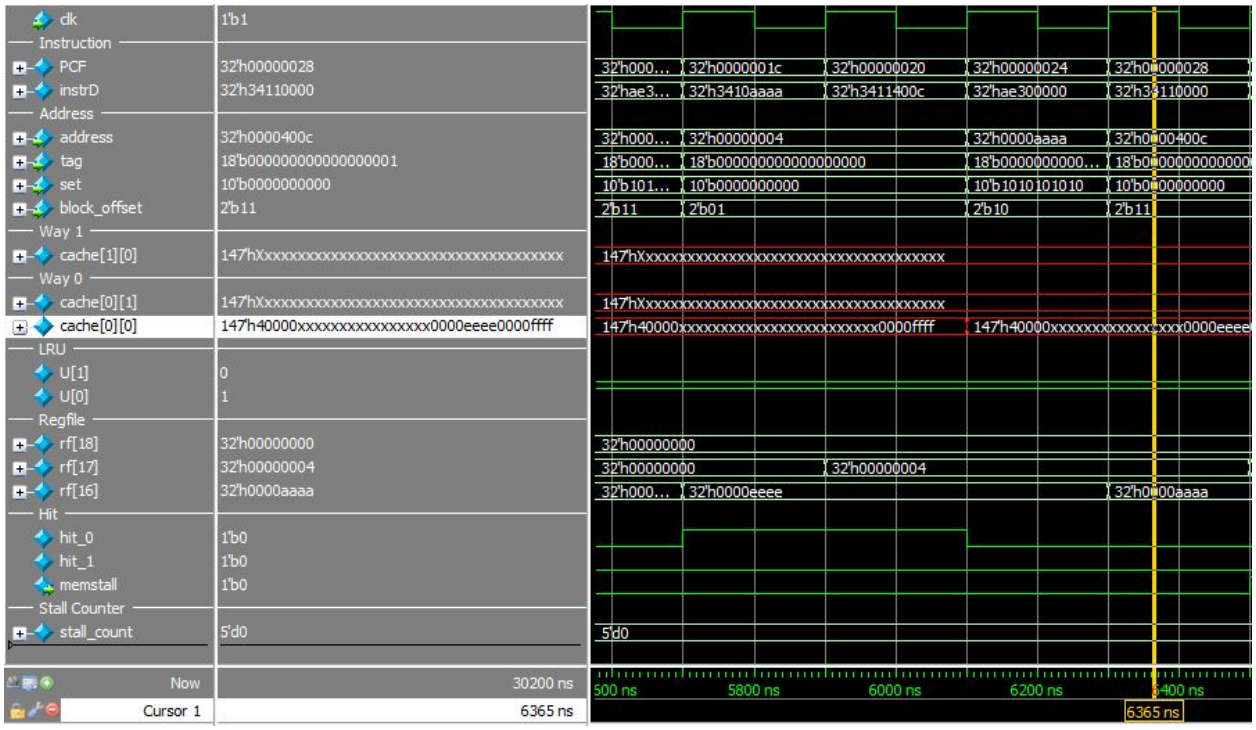
**memfile4.dat**

Initially all LRU and valid bits are 0.

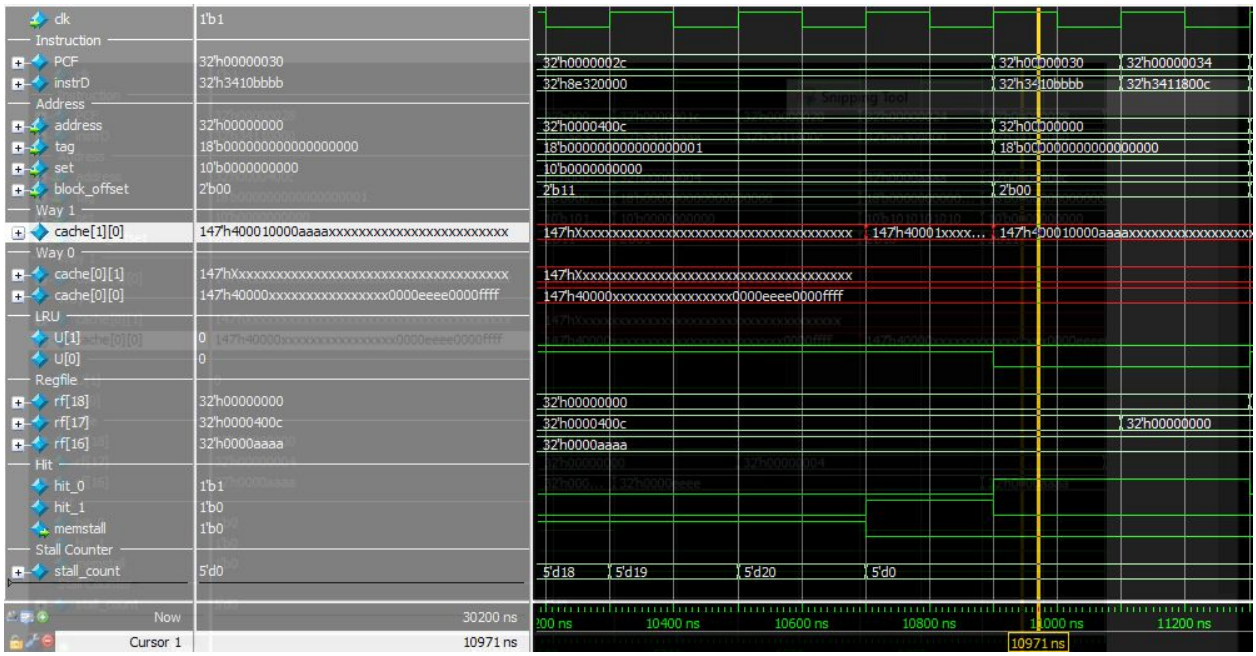
1. Stores 0xFFFF to set 0, way 0, offset 0. Tag = 0. Store miss. Valid set to 1. LRU set to 1.



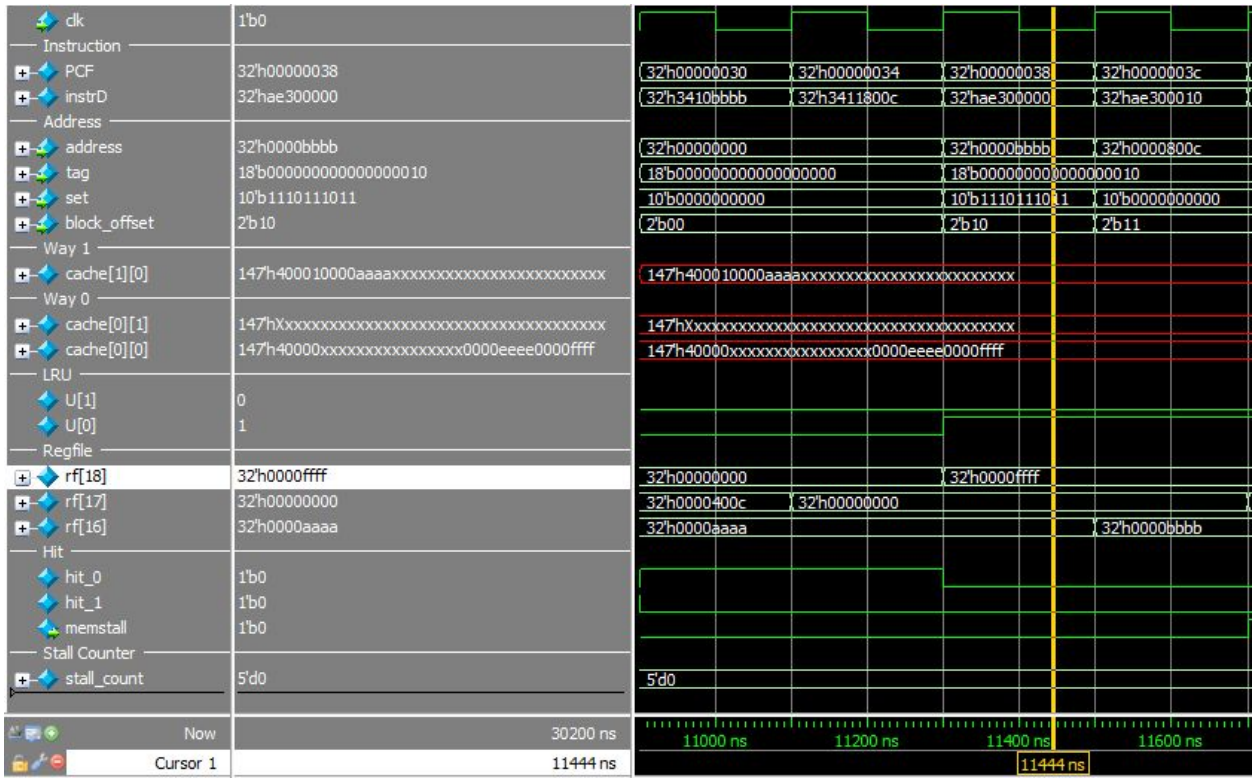

2. Stores 0xEEEE to set 0, way 0, offset 1. Tag = 0. Store hit in way 0. LRU set to 1.



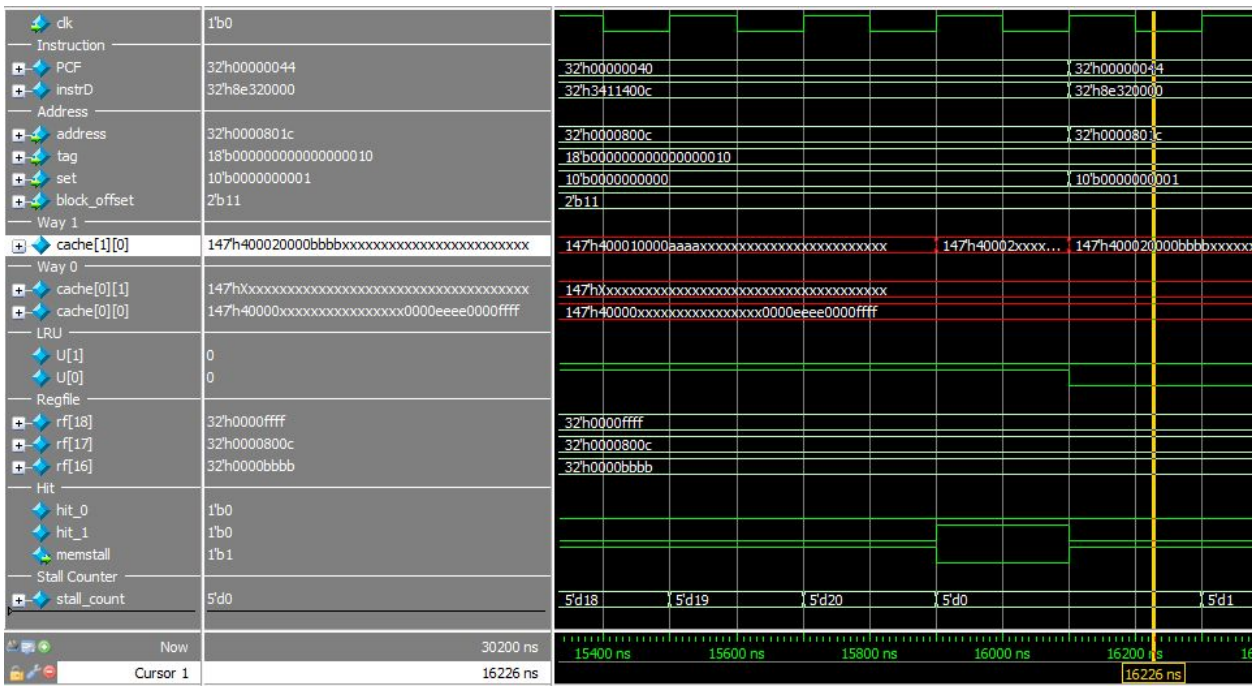

3. Stores 0xAAAA to set 0, way 1, offset 3. Tag = 1. Store miss. LRU set to 0.



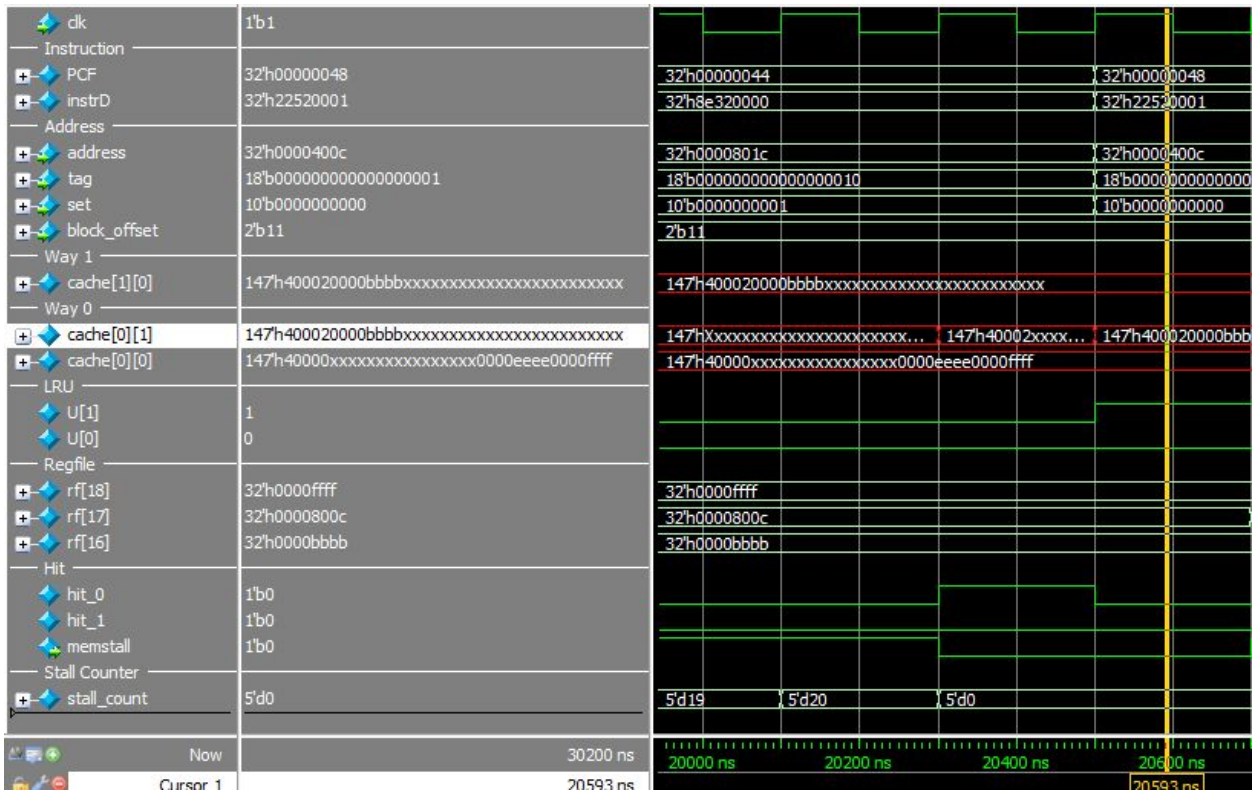

4. Loads set 0, tag = 0, offset  0 to $18. Load hit in way 0. LRU set to 1.



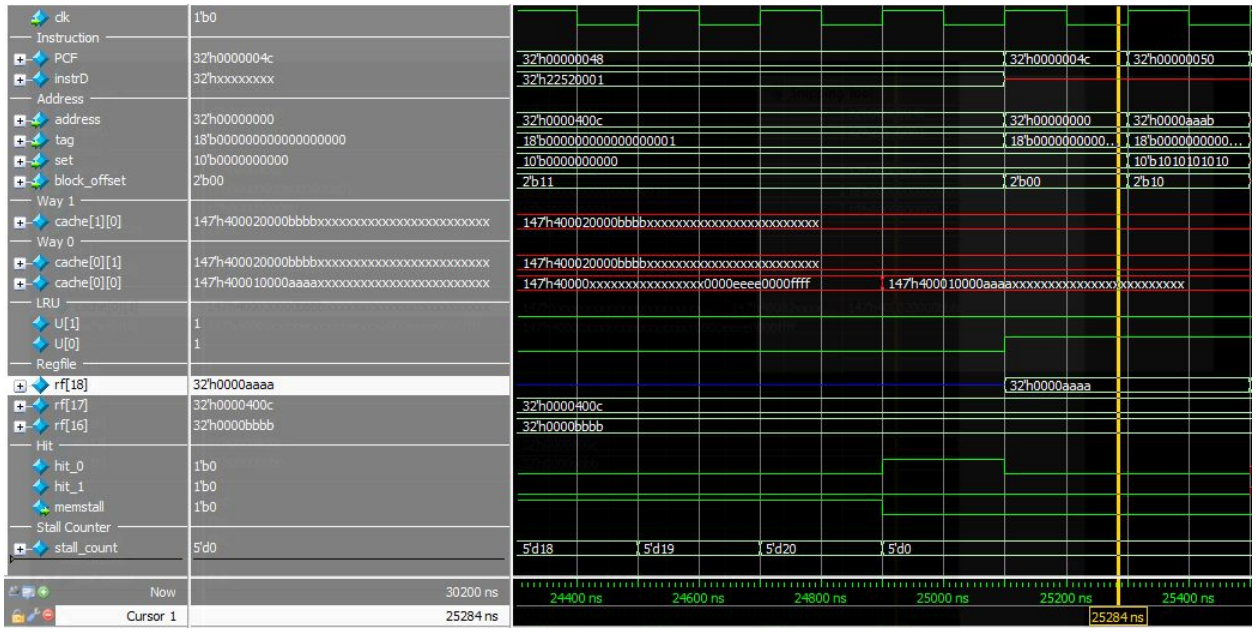

5. Stores 0xBBBB to set 0, way 1, offset 3. Tag = 2. Store miss. LRU set to 0.



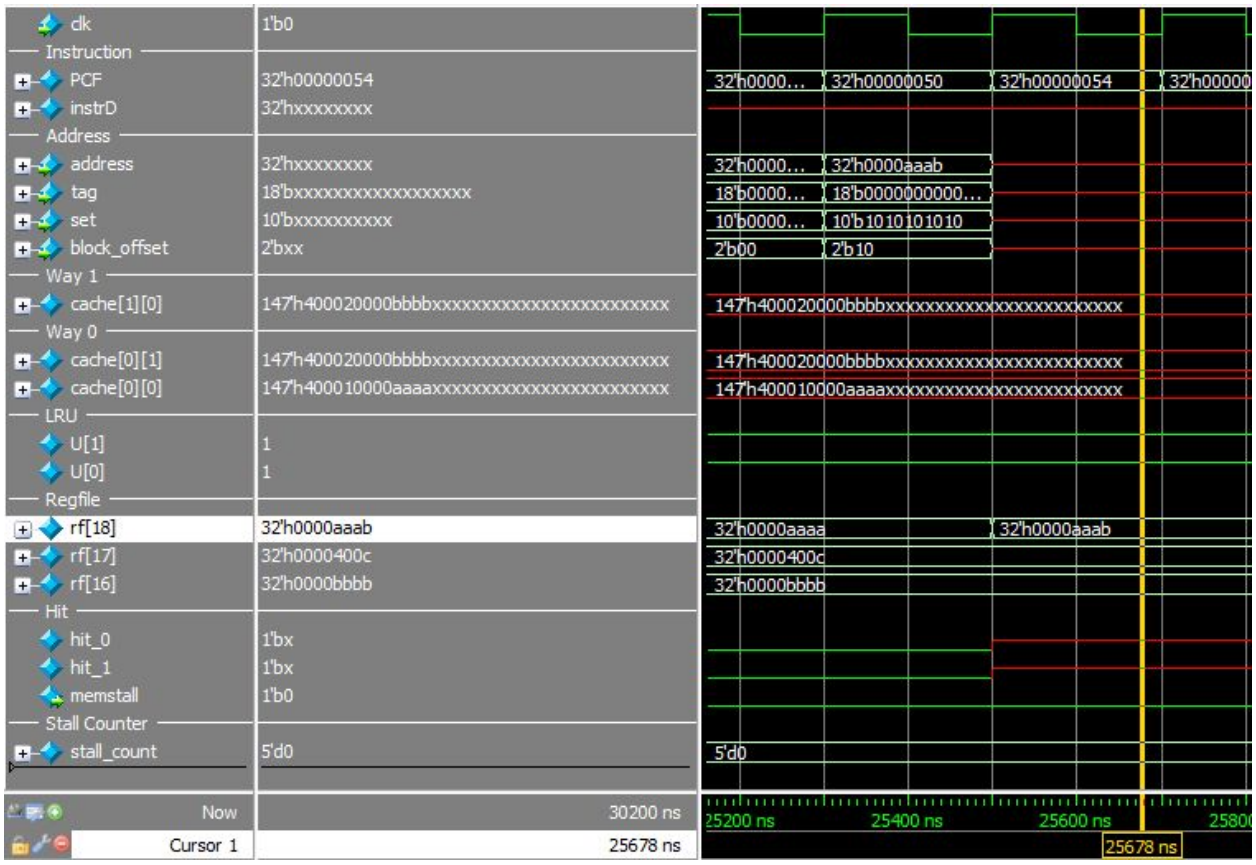
6. Stores 0xBBBB to set 1, way 0, offset 3. Tag = 2. Store miss. LRU set to 1.




7. Loads set 0, tag = 1, offset 3 to $18. Load miss. LRU set to 1.



8. Adds 1 to $18




**Conclusion**

The main goal in a memory system is to decrease the number of stalls a processor needs to take when accessing the memory. A hierarchy of memory is used to first access the fastest memory then go down the hierarchy if there was a miss. Increasing the number of words that are transferred from main memory to cache and keeping the most recently used data in cache usually decreases the miss rate depending on the program.