VERSION 0.4V

NOVEMBER 11, 2015

2-WAY SET ASSOCIATIVE CACHE

DATASHEET AND REPORT

DESIGNED BY: PRASAD PANDIT & RADHIKA MANDLEKAR

PORTLAND STATE UNIVERSITY
ECE 585 HOMEWORK 4

2-WAY SET ASSOCIATIVE CACHE

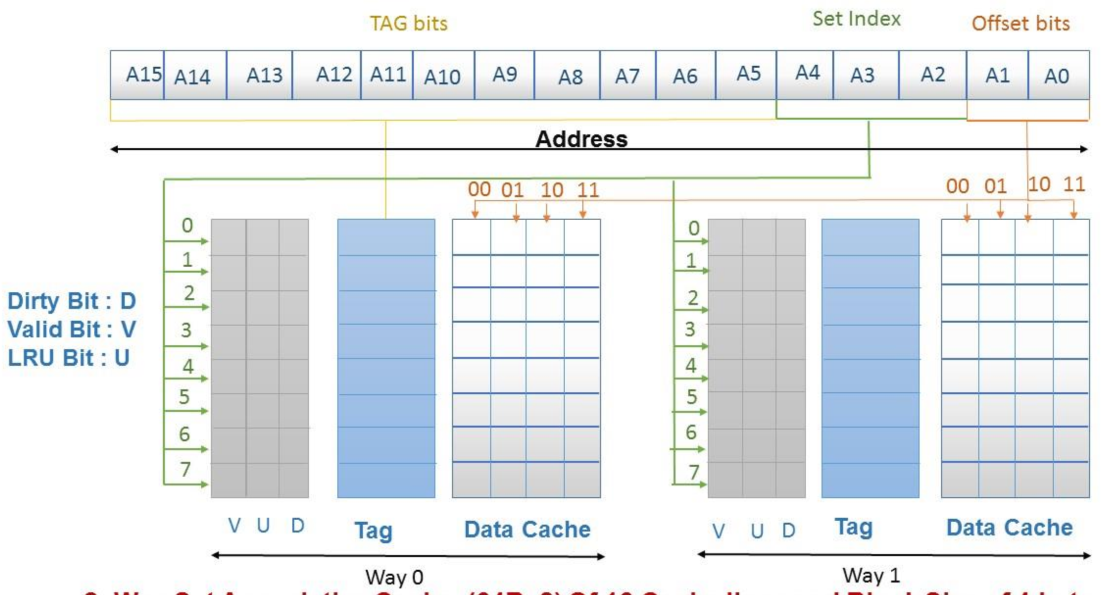
DESCRIPTION:

According to the definition of 2 way set associative cache, each set will contain 2 cache lines. In the problem statement of Homework 4, we have given 16 cache lines. Thus, we need 8 sets in total to cover all the 16 cache lines.

With the cache parameters mentioned in the problem statement, we have assumed each cache line contains 4 Data Bytes.

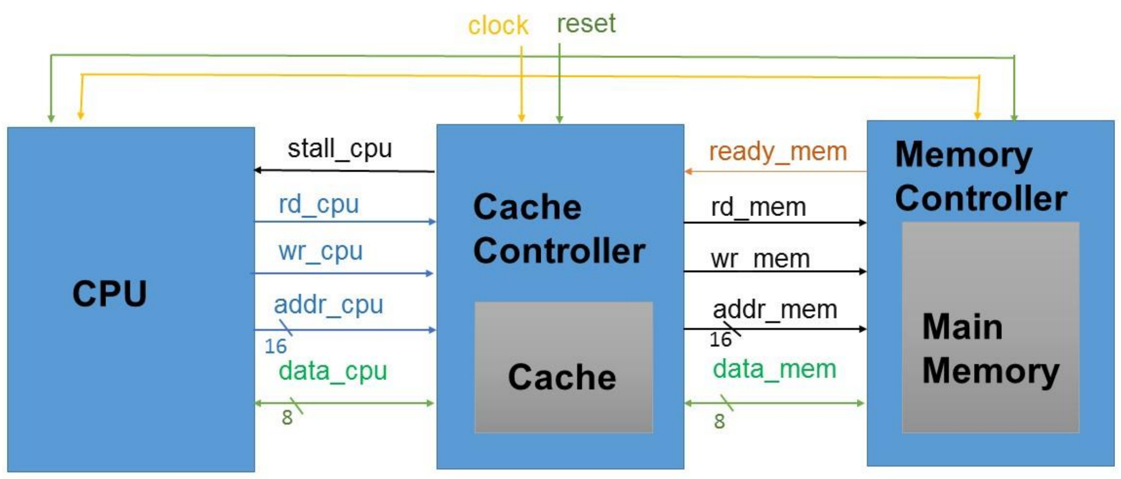
So now, we have number of sets = 8, associativity = 2 and block size = 4. Thus we can calculate the size of cache in following manner.

Cache Size = number of sets * block size * associativity = 8 * 4* 2 = 64 Bytes



2- Way Set Associative Cache (64Bx8) Of 16 Cache lines and Block Size of 4 bytes

BLOCK DIAGRAM OF CACHE CONTROLLER:



Block Diagram Of Cache Controller Interconnections with CPU and Main Memory

DEFAULT PARAMETERS:

Parameter	Value	Unit
Size of Cache	64	Bytes
Associativity	2-way	--
No. of sets	8	--
Blocks/Cache line size	4	Bytes
Address width	16	Bits
Data width	8	Bits
Tag bit width	11	Bits
Set Index bit width	3	Bits
Byte Offset	2	Bits

ASSUMPTIONS

- **Block size = 4** so we can store 4 Bytes of data in each cache line. Not keeping it very small to avoid more miss rate and not keeping it too big as we don't have to write all those bytes in main memory and store new bytes in cache for the single read or write miss.

- **Replacement Policy:** We have chosen **Least Recently Used** Policy as we have a small cache memory which might fill up quickly. Therefore, to load a new block from main memory, we have to replace one of the existing blocks in the cache. So According to the chosen policy, we are replacing the least recently used block/cache line from cache and loading the new data in that line.

  In the Verilog implementation of this cache with LRU policy, we have used a single used bit to identify which of the two cache lines in a set is least recently used.

  For Example, LRU bit =1 indicates that the cache line in way 1 is least recently used and thus, can be evicted from cache if needed and LRU bit = 0 indicates that the cache line in way 0 is least recently used for that set.

  After performing every read and write operation, we have to update the status of least recently used bit.

  For smaller associativity, it is observed that LRU performs the best and for larger associativity, Random Replacement Policy performs the best.

- **Write Back Policy:** According to this policy, the information is written only to the block in the cache and not in the main memory. So we mark it as Dirty to indicate the contents of cache and main memory are not the same or the main memory does not have the updated data.

  The modified cache block or cache line is written to main memory only when it is replaced or needs to be evicted**.**

  Advantage of Write Back Policy: For consecutive writes to same memory location, we just have to update the cache and mark the line as dirty. Therefore we are saving the consecutive main memory write cycles. At the time of eviction only, we will update the main memory with the last updated data in cache. So repeated writes to main memory are preserved and thus saves the longer CPU stalls.

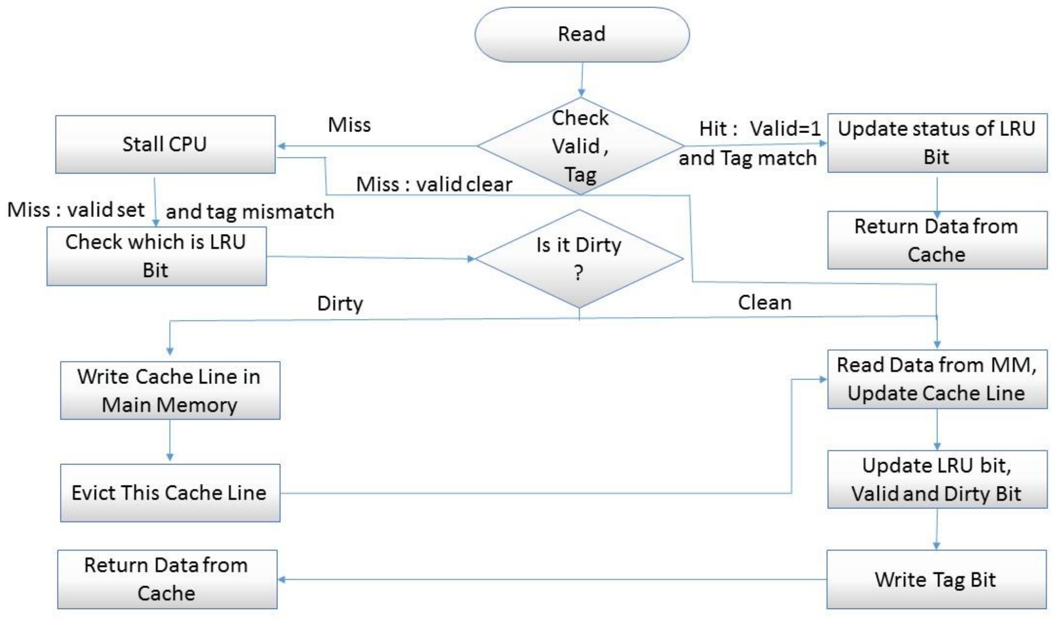
  Disadvantage of Write Back Policy: If a single byte read or write causes a miss in the cache and if the cache is already full, we have to evict least recently used dirty cache line and thus have to write those 4 bytes to main memory to load the cache with new information which is requested by CPU**.**

- **Write Allocate Policy:** For the Write miss, the block is loaded into cache hoping for subsequent writes to the same location, which is now cached.

- Assumed Main Memory reads and writes 4 bytes of data in one clock cycle.
- Memory read signal indicates that the memory is busy or free. If the signal is low it indicates memory is busy doing some operation.
- CPU uses active high Read and Write signals.

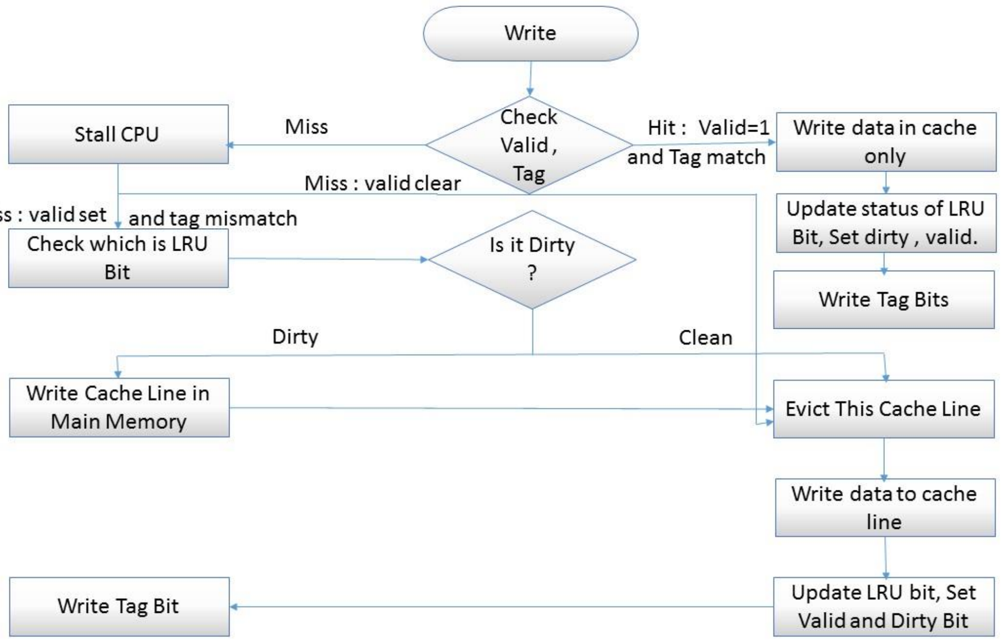
- CPU halts operation and keep address and data lines active when 'stall_cpu' gets and stays high. When it gets low the in next clock cycle CPU reads the data.
- The data bus for both CPU and Main Memory are bidirectional and are controlled using Write CPU and Write Mem signals.
- Memory Uses Active HIGH ready signal to follow data access and write protocol. During memory read, if ready is HIGH then cache keeps address of '00' byte offset on bus. As soon memory receives 'read memory' signal HIGH then it makes 'ready' signal low and when 'ready' it gives out 4 bytes of data from consecutive location. Same with memory write, when memory is ready 4 bytes from cache data block are kept ready and when write memory is made high the memory data bus works as output. Cache writes data in memory in 4 consecutive clock cycle after ready gets low.
- Both memory and CPU uses different read and write signals.
- Data bus of both memory and CPU are bidirectional.
- CPU Data bus is held at previously read data when not in use. It works as input when wr_cpu signal goes high.

FLOW DIAGRAM FOR READ OPERATION:



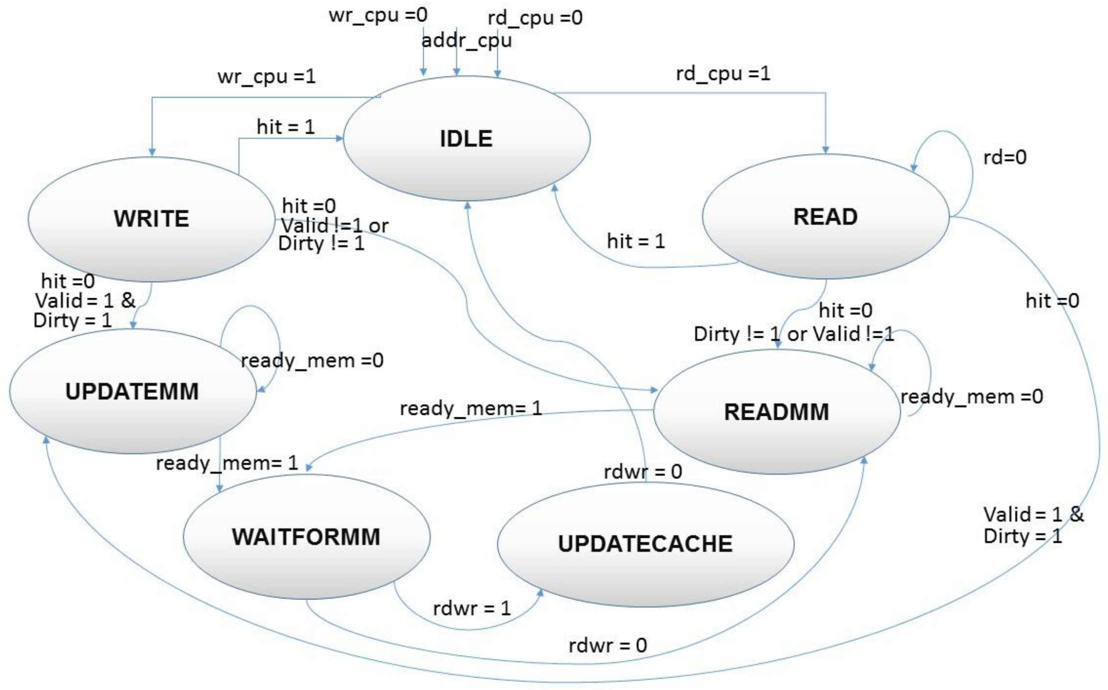
Flow Chart For Read Operation From Cache

FLOW DIAGRAM FOR WRITE OPERATION:



Flow Chart For Write Operation

STATE DIAGRAM FOR CACHE CONTROLLER FSM:

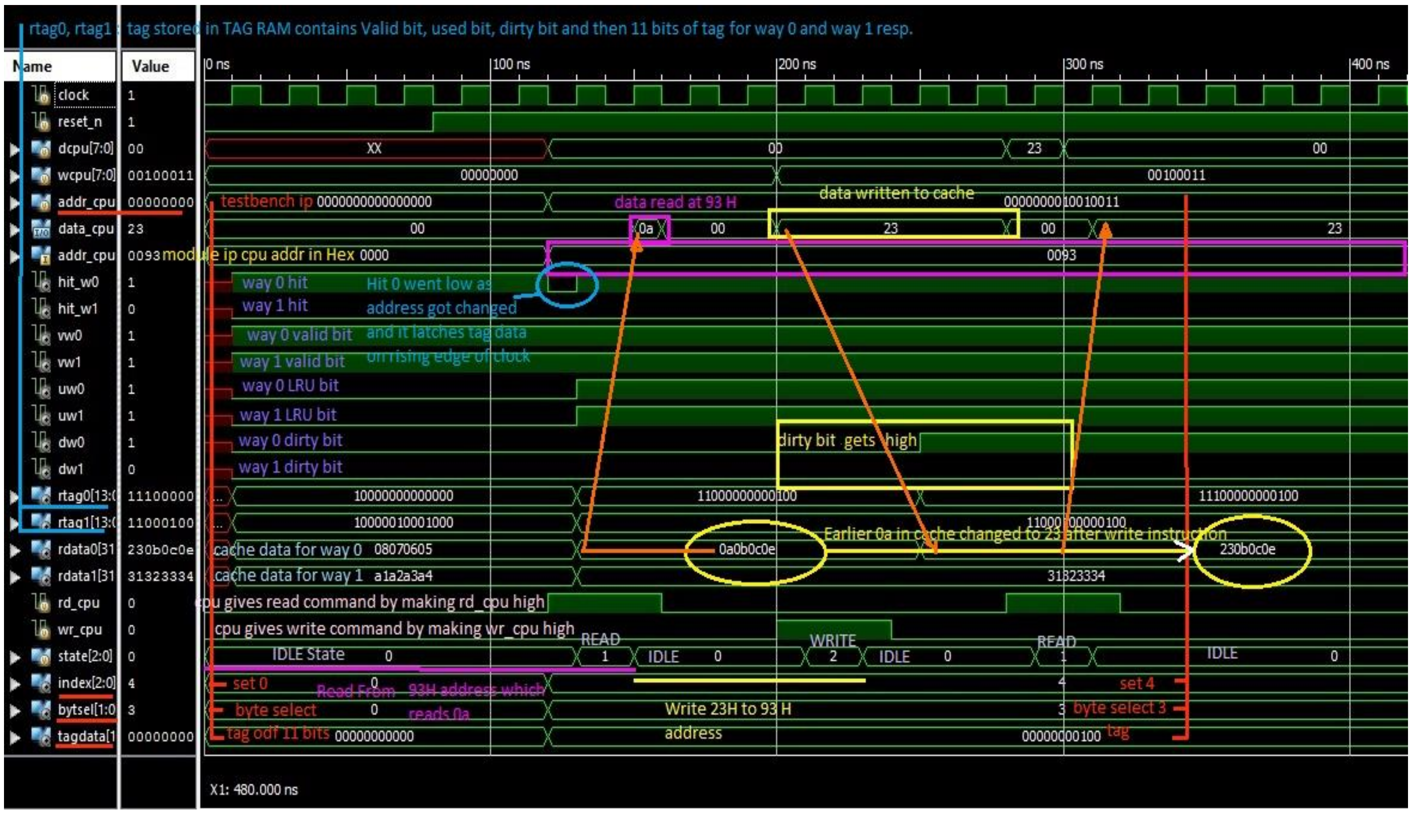


NOTE: Detailed Signal and state description is given in the verilog code.

STATE DESCRIPTION:

State	State description
IDLE	Address is latched. For write Operation, Input data is latched. All registers are by default contain zero.
READ	Check for hit or miss. If hit, read data and update LRU status. If miss transit to other states.
WRITE	Check for hit or miss. If Hit, write data to cache and set dirty bit else transit to other states.
READMM	If memory is ready, address given to MM to read from and wait till read operation is complete else wait for memory to become ready.
WAITFORMM	If memory has completed the read or write operation then transit to other states else keep waiting.
UPDATEMM	Write data in MM depending on LRU bit address
UPDATECAHE	Write cache tags as well as data in the cache

TIMING DIAGRAM FOR SIMPLE READ AND WRITE OPERATION:



DESCRIPTION:

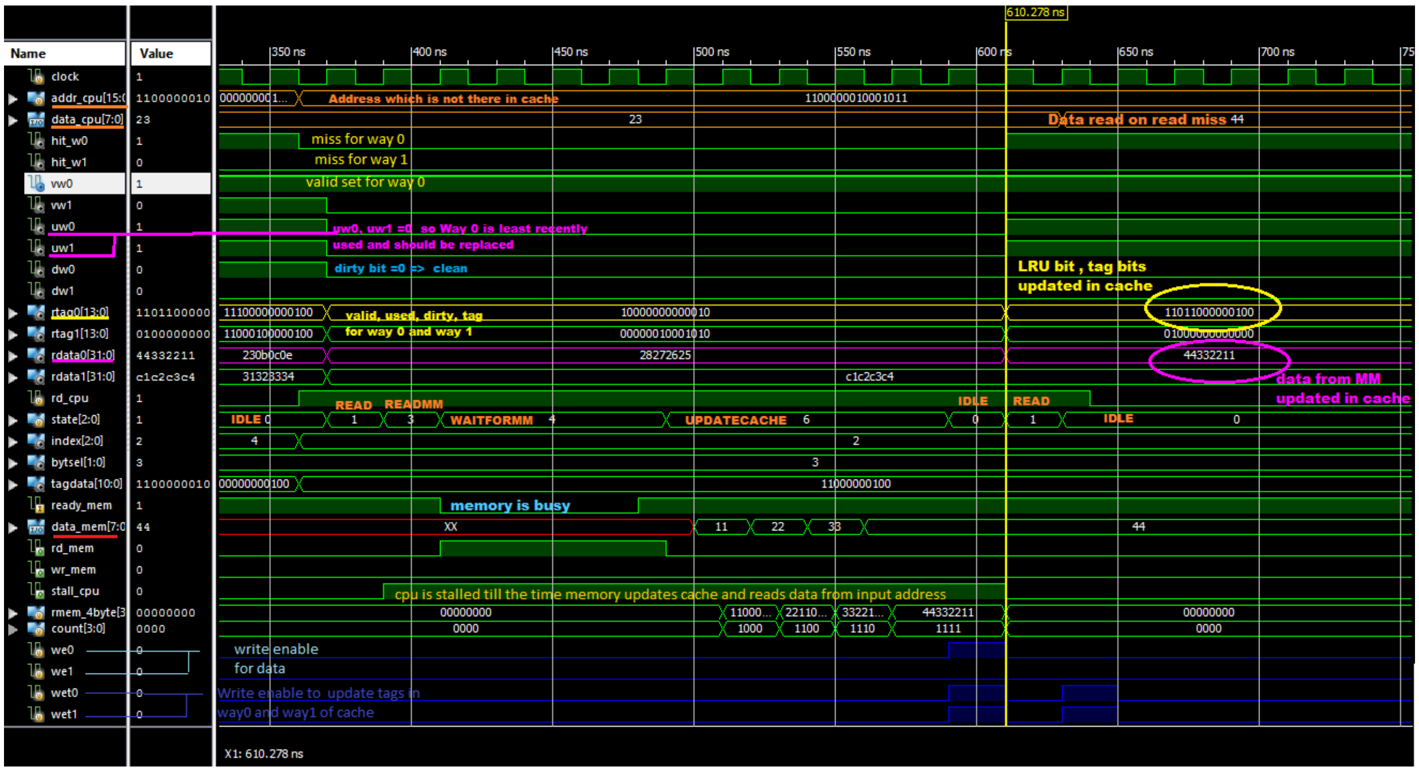
We have already stored tag and data for 93 H location which contains 0a0b0c0e H at offsets 3,2,1,0 respectively.

First Read Operation: reads 0a H from 93 H as byte select is 3 (cache Hit)

First Write Operation: Write 23 H at 93H at byte select 3 (cache Hit)

Second Read Operation: reads 23H from 93H as byte select is 3 and 0a H is modified to 23 H in Write operation.

TIMING DIAGRAM FOR READ MISS (DIRTY BIT = 0) AND REPLACEMENT POLICY:



DESCRIPTION:

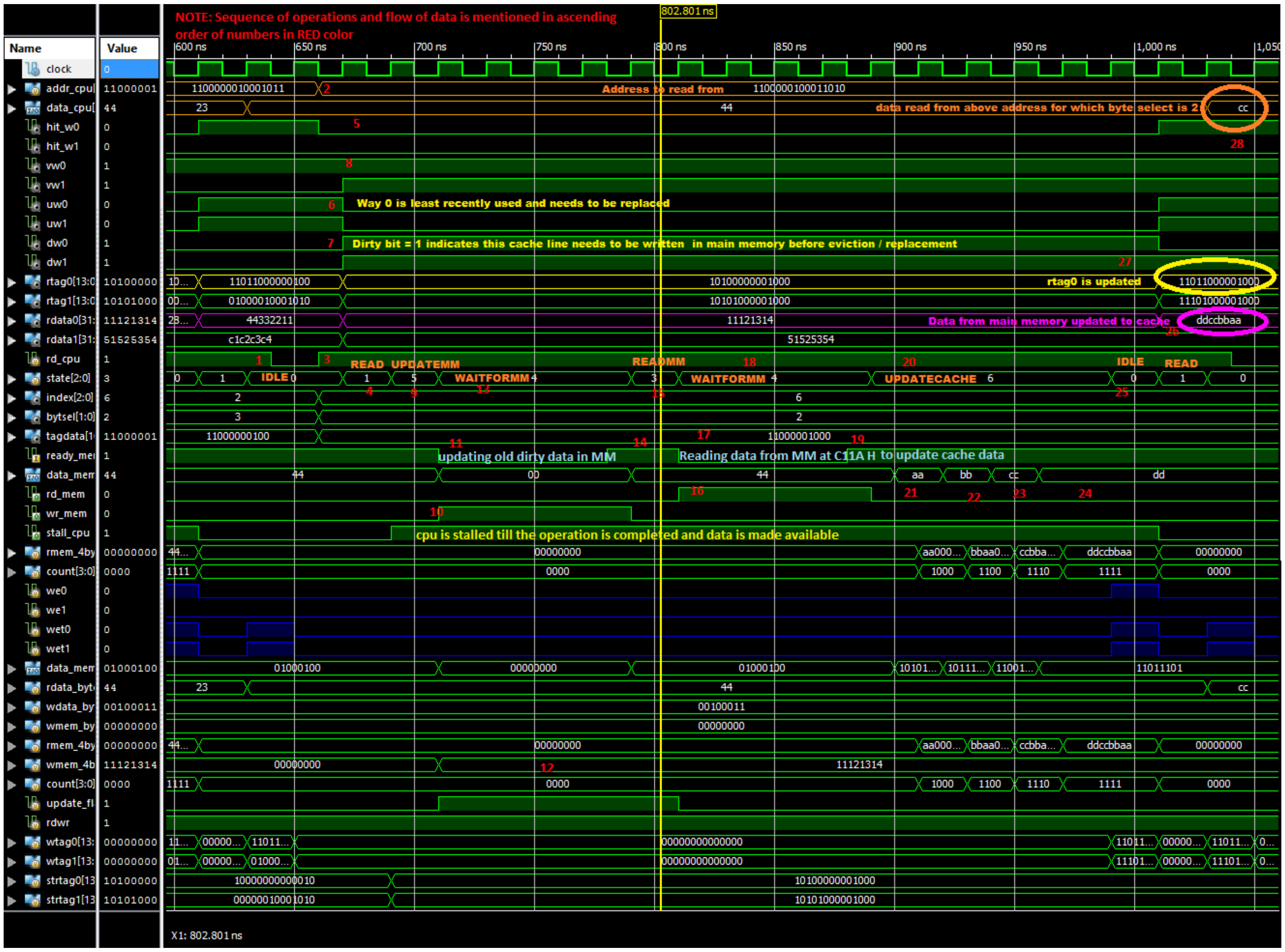
It is one of the examples of Least Resently Used – Replacement Policy.

This timing diagram describes a scenario to read from loacation C08B H which is not preasent in cache. (READ MISS as the tag is not present)

As Least Recently used bit indicates data in way 0 is least recently used. So it is replaced with new data ie 44332211 H and tag which is rtag0 = 1101100000100

Data at C08B H in main memory is 44332211 H As the byte select is 3. so data read is 44H.

TIMING DIAGRAM FOR READ MISS WITH REPLACEMENT POLICY AND EVICTION:


NOTE: Sequence of operations and flow of data is mentioned in ascending order of numbers in RED color
802.801 ns
Name
Value
clock
addr_cpu
data_cpu
hit_w0
hit_w1
vw0
vw1
uw0
uw1
dw0
dw1
rtag0[13:0]
rtag1[13:0]
rdata0[31:0]
rdata1[31:0]
rd_cpu
state[2:0]
index[2:0]
bytsel[1:0]
tagdata[1
ready_mem
data_mem
rd_mem
wr_mem
stall_cpu
rmem_4by
count[3:0]
we0
we1
wet0
wet1
data_mem
rdata_byt
wdata_by
wmem_by
rmem_4by
wmem_4b
count[3:0]
update_fl
rdwr
wtag0[13:
wtag1[13:
strtag0[13
strtag1[13
Address to read from
data read from above address for which byte select is 2
Way 0 is least recently used and needs to be replaced
Dirty bit = 1 indicates this cache line needs to be written in main memory before eviction / replacement
rtag0 is updated
Data from main memory updated to cache
READ
UPDATEMM
WAITFORMM
READMM
WAITFORMM
UPDATECACHE
IDLE
updating old dirty data in MM
Reading data from MM at C11A H to update cache data
cpu is stalled till the operation is completed and data is made available
X1: 802.801 ns

DESCRIPTION :

CPU gave address C11A H to read the data. This address is not present in cache.

Set 6 byte select 2 dirty bit 1 valid 1 LRU bit 0 => way 0 of set 6 is least recently used and is dirty.

So data at that location needs to be written in main memory before eviction and replacement and then the new data in main memory at location C11A should be updated to cache along with corresponding tag, dirty and LRU bits.

Data in MM at C11A H is ddccbbaa H is updated in set 6 way 0 in cache and second byte ie cc is read from cache and provided as output to CPU.

In the next cycle if data is to be read from same location will result in Hit.

NOTE : Please check separate images for better readability.