Check if theIndex is correct

FOR FA, index is not always 0! Finding out why may solve my issues Only for debug.txt I think (no width being pulled their to adjust block\_count)

For LRU:

replace whichever piece has the maximum value. set that pieces counter to 0. increment the rest. Do I do this for read as well? Probably

Their DM, 2W, 4W, FA has no effect on the hit rate. This makes no sense…

2W: build the cache.

Then change write method.

Then change read method

For a 2 way cache, we want our width to be two and our block count to be halved (if 2W, width = 2,

elif 4W

FA)

block count = block count/width

Breaking when I up the size, “Index out of range”

Start by running through different sizes for direct mapped cache. Next, make the 2W, 4W, and FA. Finally, track bytes written and add in WB vs WT

Now writes and reads are hitting when the tag is the same, but index is different

test2: hitting on first read for some reason (valid bit should be 0, so hit should be impossible) Forgot to remove test line

Write: Miss if data not in cache. If not in cache, write it to cache?

WB and WT won’t change hit rate. However, you do need to track:

total bytes transferred from cache to memory

total byte transferred from memory to cache

both of which will change

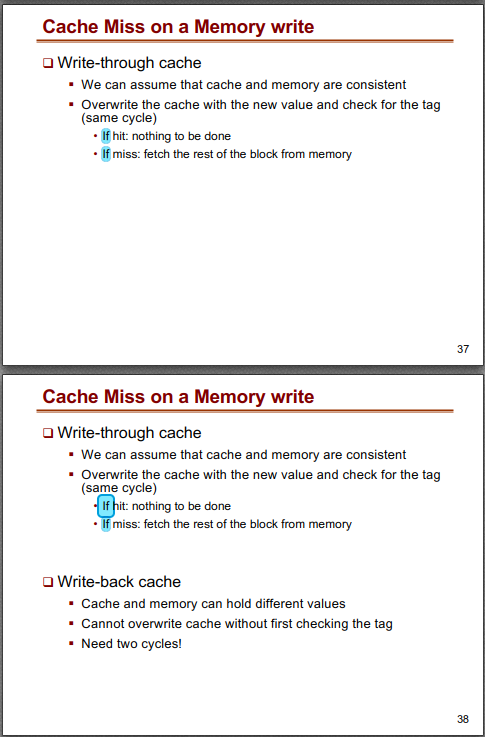
parse file: count number of lines. Set loop to number of lines.

if read

else if write

else:

do nothing



0x04000000 should be index 0, tag 65536

In main:

ParseFile()

{

RWvar;

address;

if “RWvar == read”

ReadFunction(numReads, numHits, address) TODO check if numHits or numMisses

else if “RWvar == write”

writeFunction(address) TODO track hits and misses here?

}

Have a function that builds and returns a CacheMatrix

Have a write Address function, which calculates where to store an element, set the tag, valid bit~~, and dirty bit~~.

Figure out what a dirty bit is and why it’s needed Not needed, that’s for virtual memory You actually do need a dirty bit, it’s used in the write back policy

Have a read address function, which takes in an address, calculates what the index and tag should be, checks the tag at that index. (If not direct mapped, checks multiple tags.) If valid bit is 0 or tag(s) doesn’t match, miss. Store total number of misses, also increment every read. Can probably pass in totalReads variable or something and just increment at the top of read method.

First, figure out how to simulate a cache of one type.

Then, make a class/method which will build the cache based on the 4 relevant variables.

Then, simulate reads/writes on the cache. Recall we don’t actually care what the data is

Finally, read in the file and interpret “read” for read method and “write” for write method, and interpret hex address.

Then we need to track certain things, plot those things, etc.