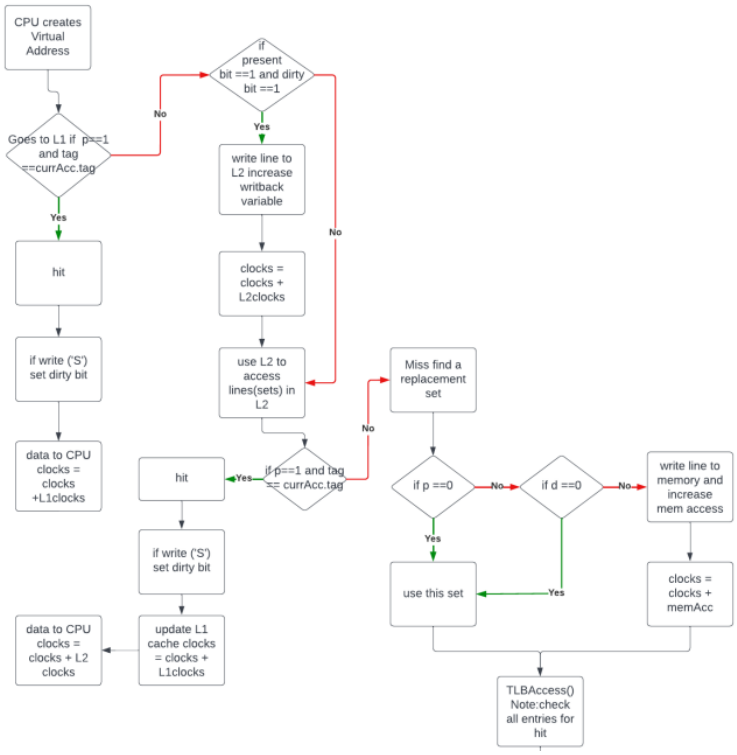
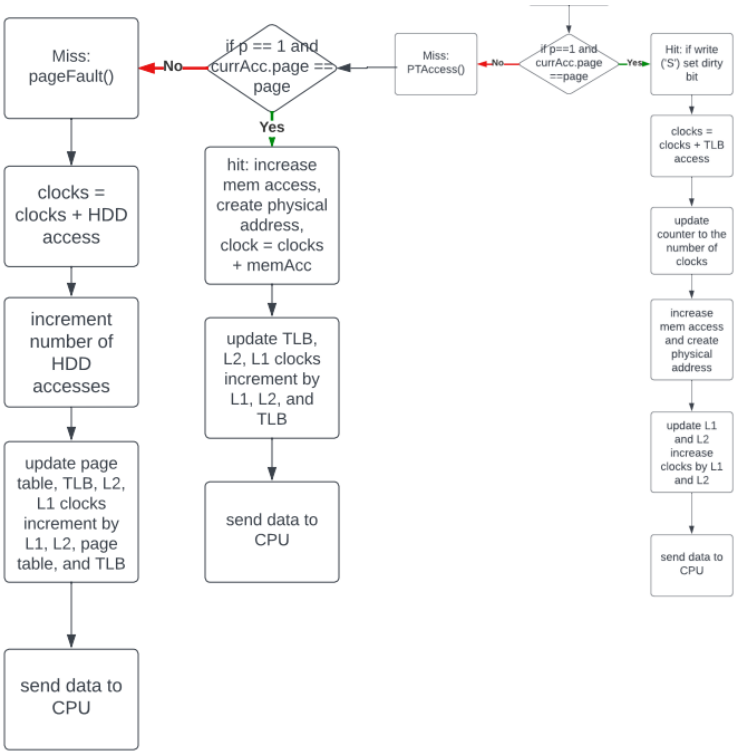
Braylee Cumro

1)

Note: the page fault is simple in this diagram because we do not have page replacement. This homework works with the TLB, ptAccess, and page fault functions to make this simplified model. There is infinite space in homework twelve.

This will also be a separate pdf file called hw12 if that makes it easier to read and follow.

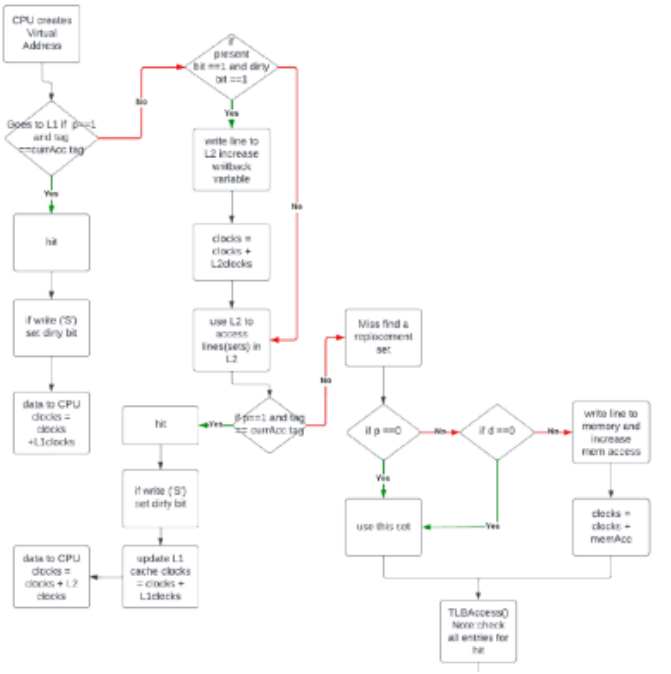


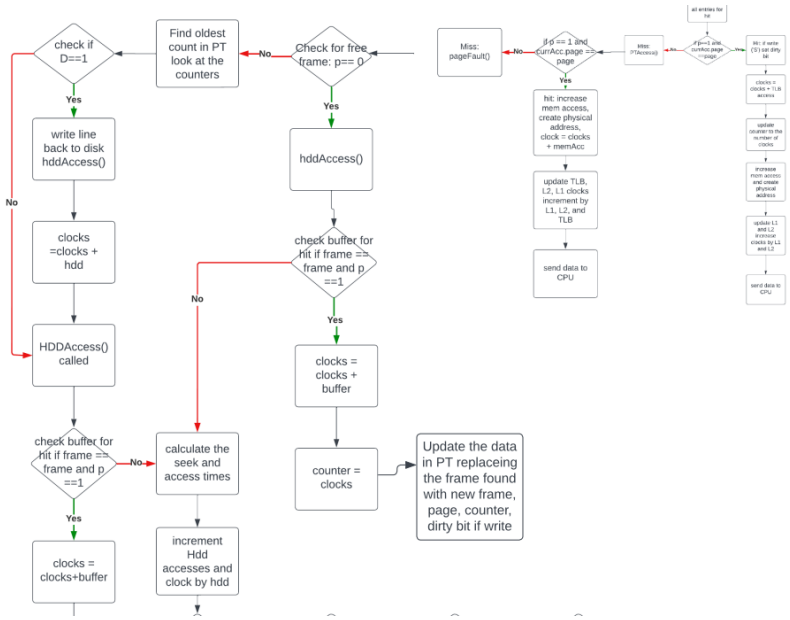


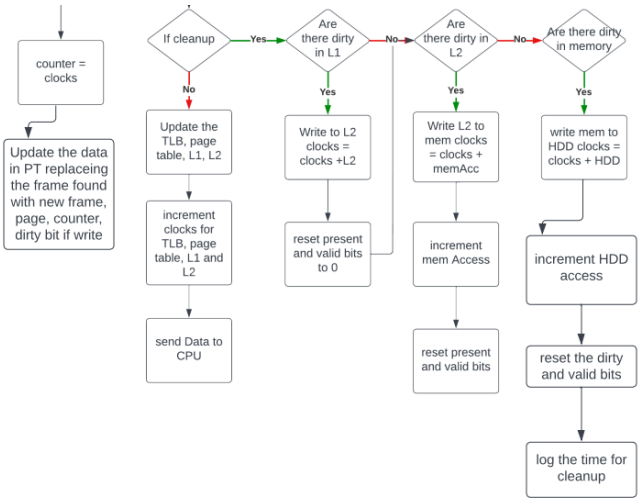
2)

This takes twelve homework assignments and adds in replacement. We are using LRU.

This will be a separate file called hw13 a pdf in this zip file, so it is easier to read.

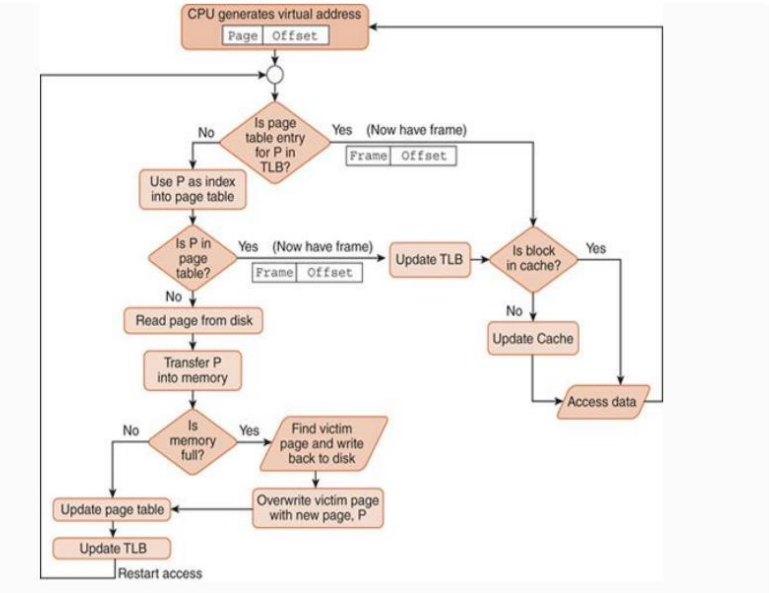

CPU creates Virtual Address
Goes to L1 if p==1 and tag ==currAcc tag
No
Yes
if present bit ==1 and dirty bit ==1
Yes
No
write line to L2 increase writeback variable
clocks = clocks + L2clocks
use L2 to access lines(sets) in L2
hit
if write ('S') set dirty bit
data to CPU clocks = clocks +L1clocks
if p==1 and tag == currAcc tag
No
Yes
hit
if write ('S') set dirty bit
update L1 cache clocks = clocks + L1clocks
data to CPU clocks = clocks + L2 clocks
Miss find a replacement set
if p ==0
No
Yes
if d ==0
No
write line to memory and increase mem access
use this ext
Yes
clocks = clocks + memAcc
TLBAccess() None check all entries for hit

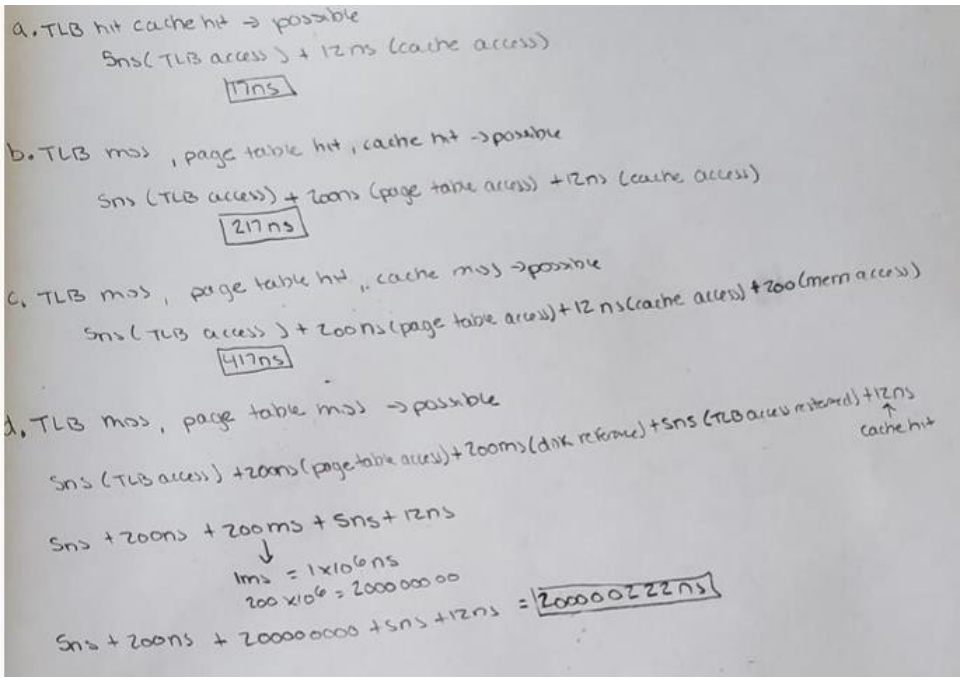

check if D==1
Yes
No
write line back to disk hddAccess()
clocks =clocks + hdd
HDDAccess() called
check buffer for hit if frame == frame and p ==1
No
Yes
clocks = clocks+buffer
calculate the seek and access times
increment Hdd accesses and clock by hdd
Find oldest count in PT look at the counters
No
Check for free frame: p== 0
Yes
hddAccess()
check buffer for hit if frame == frame and p ==1
No
Yes
clocks = clocks + buffer
counter = clocks
Update the data in PT replaceing the frame found with new frame, page, counter, dirty bit if write
Miss: pageFault()
No
if p == 1 and currAcc.page == page
Yes
hit: increase mem access, create physical address, clock = clocks + memAcc
update TLB, L2, L1 clocks increment by L1, L2, and TLB
send data to CPU



3)

I used the diagram from the textbook for reference shown below. It was on page 936.





I got that all scenarios were possible. An impossible one would be a TLB miss, page table miss, cache hit because the cache value would be old if it was not present in the page table.

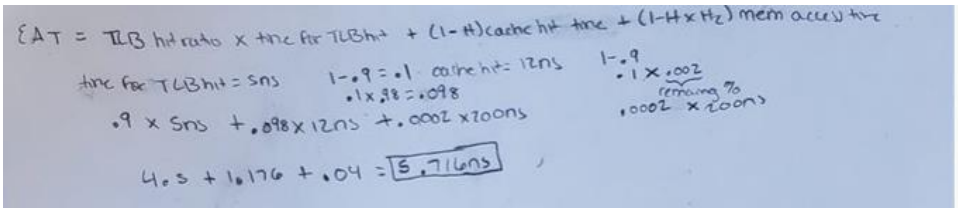
For a TLB hit and cache hit I got it took 17ns.

For a TLB miss, page table hit, cache hit I got 217ns

For a TLB miss, page table hit, cache miss I got 417ns

For a TLB miss and a page table miss I got 200000222ns

For the Effective access time (EAT):



I got 5.716 ns

I got this by multiplying the TLB hit ratio by the time for a TLB hit. Then I added to that 1-TLB hit ratio times the hit ratio for the cache and multiplied that by the time for a cache hit. The final bit was to take the remaining percentage (1 – hit ratio TLB - (1-TLB hit ratio – cache hit ratio)) which gave me the percentage of a hit for the memory access. This was multiplied by the time a memory access takes.

4) A TLB miss does not always indicate that a page is missing from the memory. The page can still be in the page table even if it is not in the TLB and thus still be in the memory. This can happen because the TLB is a cache for the page table, and in being such it often has a smaller set of pages for faster lookup than going into the page table. Since the page table is typically a smaller structure that works like a cache for the page table, there can be a miss in the TLB but a hit in the page table and in that case the page was not missing from memory only from the page table.

5)How many pages are there in virtual memory?

There is 2^7 or 128 pages in the virtual memory.

I got this number by taking the bytes of virtual memory 2^12 and dividing it by the page size 2^5.

How many page frames are there in the main memory?

There is 2^5 or 32 page frames in the main memory.

I got this by taking the size of the main memory 2^10 and dividing it by the page size 2^5.

How many entries are in the page table for a process that uses all available virtual memory?

There are 32 entries in a page table that uses all the virtual memory.