PART1

Masks are established such that they will be equivalent to 0b11111111110000000000 and 0b00000000001111111111 so that proper addresses are taken.

search_tlb(): all entries of tlb are iterated and their physical addresses are returned if the virtual address corresponds to the regarding physical address.

add_to_tlb(): treating tlb as a circular array and insert a given physical and logical address to the next entry of tlb array. When array is finished turns back to the start of the array

Page Fault: free page in the backing memory is allocated to the physical address with the usage of correct pointers and intervals between those pointers. Then the page table is updated to map the virtual memory to the new physical memory.i

Part2

Extended the atoi(argv) argument to read command line argument -p 1 and -p 0.

FIFO:

Treated free_page index as a circular array index, when index exceed page number, it returns to the head of the array and continues to erase and allocate to the next element as new addresses come. That way the oldest element in the table is automatically erased.

LRU:

Tried to create a counter array for each pagetable entry. At every cycle the counter increments by one. When new physical memory needs to be allocated the search_LRU_counter function will iterate over all the counter and find the oldest element which hasnt been updated. If an element gets updated the counter corresponding to it is initialized to 0.

We tried to implement this but kept getting an segmentation fault, we hope to get a good partial point :((

Ecem Kuloğlu 71957

Ege Uğur Amasya 71484