Team member

Nandani Patidar (np744)
Sreeram Maddineni (sm2323)

Operating Systems Design | Project - 3 | Fall 2020

User-Level Memory Management Library

1. Detailed logic of how you implemented each virtual memory function.

SetPhysicalMem: This function is called in the myalloc function. Here we use the malloc function to initialize the main memory and page tables. *(Done by **Nandani**)*

add_TLB: This function adds a virtual page address and the corresponding physical page address into the TLB. If the TLB is full it evicts a random virtual to physical page mapping and replaces it with the requested mapping. *(Done by **Sreeram**)*

check_TLB: This function checks whether a virtual page address has a corresponding physical page address in the TLB. If the physical page address is present, then it returns the address otherwise it returns null. *(Done by **Sreeram**)*

print_TLB_missrate: This function prints the number of TLB hits, misses and the TLB miss rate. *(Done by **Sreeram**)*

Translate: This function first checks if the virtual page entry is present in TLB. If the entry is present then the physical page address is taken from TLB, offset is added, and physical address is returned. If the entry is not present, then we look into the page table to get the physical page address. To get the physical address we extract three groups of bits from a 32-bit virtual address. First group of bits helps us to find the address of the inner page table from the outer page table. Second group of bits are used to get the address of the physical page and offset bits are used to get the actual physical address. *(Done by **Sreeram**)*

PageMap: Similar to translate function we extract three groups of bits from a given virtual address. Using these bits, we check whether the given virtual page mapping is already present in the page tables or not. If the mapping is not present, then we enter the virtual and physical page addresses into the page table. We also make an entry for the mapping into the TLB. *(Done by **Sreeram**)*

check_require_avail_pa: In this function we search for the available pages in our memory. This function returns true if memory is available and false when it is not available. *(Done by **Nandani**)*

get_next_avail_pa: This function returns the address of a free physical page from our memory. It returns 'NULL' if no physical page is free. *(Done by **Nandani**)*

get_next_avail_va: This function checks for the availability of free continuous virtual pages and returns the address of the first page. It is called once irrespective of the number of pages required. The number of pages required is passed as a parameter. *(Done by **Nandani**)*

myalloc: This function checks if main memory is initialized or not. If not initialized, then it initializes the main memory. Then it calls other functions (get_next_avail_pa, check_require_avail_pa, etc.) to check that virtual memory and physical memory is present as per user requirement. If both the memories are available, then only it returns a 32-bit virtual address which is the starting address of the allocation. It calls the PageMap function repeatedly to enter the virtual to physical page mapping into the page tables. *(Done by **Nandani**)*

myfree: It frees all the pages and entries in page tables based on the given virtual address and size passed in the function. *(Done by **Nandani**)*

PutVal: This function first calls the translate function to get the corresponding physical address for the virtual address. Then it copies the value passed in the parameter into the physical address. *(Done by **Nandani**)*

GetVal: This function first calls the translate function to get the corresponding physical address for the virtual address. Then it copies the value from the physical address to another variable using the address of the variable. This address is passed as an argument to the GetVal function. *(Done by **Nandani**)*

MatMult: This function takes the starting address of two matrices, multiples them and copies the output to the address of the 'answer' matrix. This 'answer' address is passed as an argument to the function. *(Done by **Sreeram**)*

2. Benchmark output for Part 1 and the observed TLB miss rate / runtime improvements in Part .

Average time taken to access an integer using the GetVal function without the tlb = 2.7 microseconds

Average time taken to access an integer using the GetVal function using the tlb = 0.7 microseconds

TLB miss rate : (No of TLB misses)/(No of TLB misses + No of TLB hits)
Size of each matrix = 30 * 30 *4 = 3600 bytes

Page size = 1024 bytes
TLB misses = 14, TLB hits = 233999, TLB miss rate = 0.000060

Page size = 2048 bytes
TLB misses = 8, TLB hits = 233999, TLB miss rate = 0.000034

Page size = 4096 bytes
TLB misses = 5, TLB hits = 233999, TLB miss rate = 0.000021

3. Support for different page sizes (in multiples of 4K).

Yes, it supports the different page sizes.

4. Possible issues with your code (if any).

Our allocations for the page tables and some other variables are outside our memory manager. Ideally, they should be allocated as part of our memory manager.

5. What were the most difficult parts of solving this project?

The most difficult part was to understand the initial flow of the project before writing the code even though we understood the theory of the topic. Designing the structure of page table was challenging.