Kunal Jangam ksj48

Implementation

Both the makefiles were compiled with the -lm flag since log2 and pow were used for some calculations. I created a bitmap for the l1 table and for physical memory, and an array of bitmaps for the l2 tables. The page table directory was an unsigned int ** that points to all the l2 tables that are in use. The first time a_malloc is called, set_physical_mem is called and all the variables are initialized and malloced with the correct number of bytes.
Translate calls check_TLB. If a page is returned, it returns that page. Otherwise, the inner and outer vpn bits are read and the bitmaps are checked to see if an entry exists. If it does, the entry and vpn are added to the tlb and the entry is returned.
Page_map reads the inner and outer vpn bits and checks the bitmaps if an entry exists. If one does, return failure. Otherwise, add in an entry, set the bit in the bitmaps, and return success.
Get_next_avail goes through the physical memory bitmap to find an unused page. When found, the bit in the bitmap is set and the page returned.
Get_vpages is a helper function that returns the starting virtual address to a block of contiguous virtual memory that has enough unused pages (empty entries) to fit in the number of pages required.
A_malloc calculates the number of pages to allocate, calls get_vpages for the starting virtual address, then for each page calls get_next_avail, page_map, and add_TLB, and increments the vpn bits of the virtual address by one for the next page. I did not count adding to the tlb here as a tlb miss since no vpn to ppn translation was attempted.
A_free calculated how many pages to free, then for each page, found the physical page, set the bit in the bitmaps to 0, removed the translation from the tlb, and incremented the vpn bits in the virtual address by one to get to the next page.
Put_value and get_value found the number of pages to copy, and for each page found the physical page using translate. Put_value copied the data from val to the physical page and get_value copied the data from the physical page to val. The index to put the data into or take the data from val was also kept track of.
Matrix multiplication did 3 get_values to get mat1[i][k], mat2[k][j], and answer[i][j]. Val3 was set to answer[i][j]+mat1[i][k]*mat2[j][k] and, through put_value, answer[i][j] was set to val3.
add_TLB took the vpn bits of va, and attempted to find an empty spot in the tlb(valid==0) to put the new translation. If one was not found, a tlb entry was replaced. Eviction was FIFO.

check_TLB incremented tlbchecks, and took the vpn bits of va, then attempts to find a valid tlb entry that has the vpn to ppn translation and return the ppn. If no such entry was found, tlbmiss was incremented.
print_TLB_missrate prints tlbmiss/tlbchecks.

Benchmark results

PGSIZE 4k, TLB size 512, SIZE 5, ARRAY_SIZE 400
Allocating three arrays of 400 bytes
Addresses of the allocations: 0, 1000, 2000
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
Performing matrix multiplication with itself!
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
Freeing the allocations!
TLB miss rate 0.000000
Checking if allocations were freed!
free function works

PGSIZE 8k, TLB size 64, SIZE 3, ARRAY_SIZE 200
Allocating three arrays of 200 bytes
Addresses of the allocations: 0, 2000, 4000
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1
1 1 1
1 1 1
Performing matrix multiplication with itself!

3 3 3
3 3 3
3 3 3
Freeing the allocations!
TLB miss rate 0.000000
Checking if allocations were freed!
free function works

PGSIZE 64k, TLB size 1, SIZE 10, ARRAY_SIZE 800
Allocating three arrays of 800 bytes
Addresses of the allocations: 0, 10000, 20000
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
Performing matrix multiplication with itself!
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
Freeing the allocations!
TLB miss rate 0.755718
Checking if allocations were freed!
free function works

Page sizes of 4k, 8k, 16k, 32k, and 64k were tested to work properly.