Operating Systems 198:416
Prof. Srinivas Narayana
AC1771 Advith Chegu
TSC95 Tajvir Chahal

1. Virtual memory function implementation explanation

void set_physical_mem():

The set_physical_mem function initializes the virtual memory system. It allocates physical memory using malloc, sets up the TLB entries, and initializes bitmaps to track the allocation status of physical and virtual pages. The function also calculates offsets and sizes for page directory and table entries, providing the groundwork for the entire virtual memory system.

pte_t translate(pde_t pgdir, void *va):

The translate function performs virtual-to-physical address translation. It first checks the TLB for a translation. If a translation is found, it directly returns the corresponding physical address. In the case of a TLB miss, the function looks up the page table, updates the TLB with the new translation, and returns the physical address.

int page_map(pde_t pgdir, unsigned long va, pte_t pa)

The page_map function establishes a mapping between a virtual page and a physical page. It checks for the existence of page directory and table entries, initializing them if necessary. After updating the TLB with the new translation, the function returns 0 to indicate a successful mapping.

void *t_malloc(unsigned int num_bytes)

Allocates virtual memory for a specified number of bytes. It checks whether physical memory is initialized and initializes it if necessary. The function then finds the next available virtual address, marks the corresponding virtual pages as allocated, and updates the TLB with the new translations.

int put_value(void *va, void *val, int size)

Copies data from a source buffer to the virtual memory pages indicated by the given virtual address. The function uses translate() to find the corresponding physical pages, and it handles cases where the data size exceeds a single page.

void get_value(void *va, void *val, int size)

Retrieves data from virtual memory pages indicated by the given virtual address and copies it to a destination buffer. Similar to put_value(), it uses translate() to find the corresponding physical pages, accommodating cases where the data size spans multiple pages.

void mat_mult(void *mat1, void *mat2, int size, void *answer):

Performs matrix multiplication using virtual memory. The function uses get_value() and put_value() to load and store matrix elements, facilitating the multiplication process. It works with the virtual memory system to handle large matrices efficiently.

void t_free(void *va, int size):

The t_free() function frees memory pages starting from the given virtual address. It updates the physical and virtual bitmaps to mark pages as unallocated, allowing for future use. The function also removes corresponding translations from the TLB to maintain consistency.

Int add_TLB(void *va, void *pa)`:

The add_TLB() function adds a virtual-to-physical page translation to the TLB. It calculates the TLB entry index based on the virtual address and updates the TLB with the new translation. If an entry is already in use, the function overwrites the existing translation.

Pte_t check_TLB(void *va):

The check_TLB function performs a TLB lookup to check for the presence of a valid translation for the given virtual address. It calculates the TLB entry index, checks if the entry is valid, and returns the corresponding physical page address if found. If no valid entry is found, it returns NULL.

Void print_TLB_missrate()`:

The print_TLB_missrate function calculates and prints the TLB miss rate. It analyzes the number of TLB hits and misses to provide insights into the efficiency of the TLB. The output includes hit and miss counts, allowing for performance evaluation and optimization.

long get_next_page()

The get_next_page function is responsible for identifying the next available physical page in the system's memory. It iterates through the physical bitmap, which tracks the allocation status of each page, to find the first unallocated page. Once an unallocated page is identified, the function returns its address, marking it as allocated for subsequent use. This function plays a crucial role in memory allocation, ensuring that physical pages are efficiently utilized and preventing memory exhaustion.

unsigned long get_next_avail(int num_pages)

The get_next_avail function is designed to find the next available block of virtual memory pages. It iterates through the virtual address bitmap, checking for a contiguous sequence of unallocated pages that can accommodate the specified number of pages (num_pages). When such a block is found, the function returns the starting virtual address of this free memory region. This function is essential for allocating virtual memory efficiently, ensuring that consecutive pages are allocated together to support the requested memory size. It is a key component in managing the virtual address space during memory allocation operations.

2. Benchmarking

Results for single threaded test:

```
hits: 151.000000
misses: 252.000000
TLB miss rate 0.625310
```

Results for multithreaded with 15 threads

```
hits: 590.000000
misses: 1225.000000
TLB miss rate 0.674931
```

Results for multithreaded with 150 threads

```
hits: 288.000000
misses: 17412.000000
TLB miss rate 0.983729
```

3. Support for different pages

The program does support different page sizes.

4. Possible Issues with our code

There are no major issues preventing the program from compiling or running correctly. There could potentially be improvements in performance such as the hit rate of TLB by changing the method of picking an entry to evict.

5. Extra Credit

Did not complete the extra credit

6. Resources

- Course Slides
- C reference manual
    - https://devdocs.io/c/
- Geeks4Geeks Explanation of 2 level paging and TLB
    - https://www.geeksforgeeks.org/two-level-paging-and-multi-level-paging-in-os/
    - https://www.geeksforgeeks.org/translation-lookaside-buffer-tlb-in-paging/