Devin Macalalad dtm97

David Gasperini dlg195

1)

set\_physical\_mem: mallocs using MEMSIZE to allocate for physical mem, which is just a char (byte) array. Uses calloc to allocate and zero out bitmaps for physical and virtual pages. The global list of TLB structures are also allocated and set to default values.

add\_TLB: right shifts the virtual address given by pgbits(offset bits) to be able to index to proper TLB structure given virtual address. If the TLB at the indexed slot is occupied, the occupying page is removed via remove\_TLB. The valid is then set to true, the virtual address and physical address are then set to the passed arguments.

remove\_TLB: right shifts the virtual address given by pgbits(offset bits) to be able to index to proper TLB structure given virtual address. Sets the validity to false, and both pointers to NULL.

check\_TLB: TLB works by having an address map to a TLB structure by using a modulo on the virtual address passed in with the total number of TLB structs (index=va%TLB\_ENTRIES). check\_TLB right shifts the virtual address given by pgbits(offset bits) to be able to index to proper TLB structure given virtual address. If the valid member is set to false, the miss counter is incremented and NULL is returned. Otherwise, the physical address in the TLB is returned. The counter of total lookups is incremented regardless.

print\_TLB\_missrate:

translate:

page\_map: Indexes to specific page directory entry using top bits. If the page table of the page directory entry is NULL, a page table is allocated using malloc, with each physical address being set to NULL. Then a specific page table entry is indexed to using middle bits. Finally, the physical address of that page table entry is set to what was passed.

get\_next\_avail:

a\_malloc:

a\_free:

put\_value: number of pages to hold the passed size is calculated. A loop iterated through this number, each time finding the physical address mapped to the virtual address + the current page via translate. Then a memcpy is used to copy the data from val + the current page to the physical address. If the current page is the last needed, size%PGSIZE amount of data is copied instead of a whole page.

get\_value: same as put\_value except the current page of the physical address is being copied into val + the current page

mat\_mult: Two nested loops count through the rows and columns, while a third inner loop is used to be able to address each matrix separately to perform a correct multiplication. Get\_value is used to obtain the values from the two matrices. They are then multiplied together and added to the running sum for the current position using put\_value.

2)

Benchmark Output:

Allocating three arrays of 400 bytes

Addresses of the allocations: 1, 1001, 2001

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!

Checking if allocations were freed!

free function works

TLB miss rate 0.005964

3) The code supports page sizes in multiples of 4K.

4)

Possible issues:

* Anywhere malloc/calloc is used, a failure will result in an error of the over all program