Yashraj Nikam (yan7)
Dhruv Kedia (dk1157)

Project 3 Report

Implementation:

Data Structures

physicalMemory: This structure stores the base pointer of the physical memory, the physical bitmap and the number of frames in the physical memory.

tlb: This structure holds the TLB entries. It maps a number of virtual pages to their corresponding physical pages. The number is given by 'TLB_ENTRIES', a macro defined in the header file 'my_vm.h'. The mapping is done by declaring 2 arrays. One stores the virtual page number as integer and the other stores the corresponding physical address as void*.

Thread Safe

In order to make the implementation thread safe, spin locks are implemented at the critical sections using atomic test set flags.

API

Helper

mylog2(): Returns the log of the value passed in.

calculateLevels(): Calculates and returns the number of levels of page tables required.

set_bit_at_index():
get_bit_at_index():
get_top_bits():
Bitmap functions used from Project 1:

reset_bit_at_index(): Similar to **set_bit_at_index()**, this function resets the bit to 0 at a given index.

Core

set_physical_mem(): Initializes the physical memory, physical bitmap and the virtual bitmap. It then calculates the number of levels by invoking ‘**calculateLevels()’** and maps the page directories to page tables. It returns the base pointer of the page directory.

translate(): Performs the translation of virtual address to physical address. In case of TLB hit, address is returned without performing the translation and in the case of TLB miss, the translation is computed, added to the TLB and returned to the caller.

get_next_avail(): Uses virtual address bitmap to find the next free pages. It returns the index of the first free page.

t_malloc(): Calculates the number of pages required to accommodate the request. It then invokes **get_next_avail()** and gets the index of the first free virtual page. Then indices of free physical pages are found and temporarily stores them into an array. The mapping of virtual pages to physical pages is done using the array and the virtual address is returned.

t_free(): Checks if the address passed has been allocated or not. If yes, it frees the memory by setting the bit at corresponding index in the physical and virtual index to 0. Returns if no memory is allocated at that address.

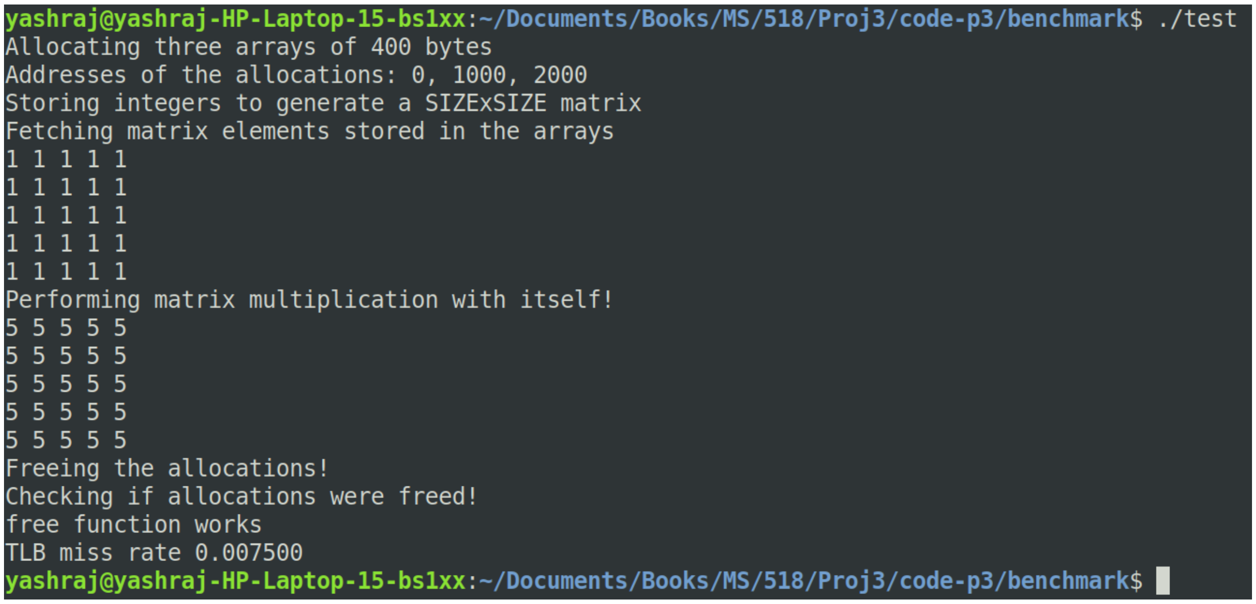
put_value(): Virtual address is translated to physical address. Then val is copied to the physical frame. Multiple pages are handled by virtual bitmap. Virtual bitmap stores the physical frame number so when size exceeds page size, the next page from virtual bitmap is translated and the copying continues.

get_value(): Similar to **put_value()**, just the source and destination changes. That is, contents are copied from the physical frame into val.

add_TLB(): This function adds the physical address corresponding to the virtual address in the TLB. The integer array stores the virtual page number to avoid conflicts.

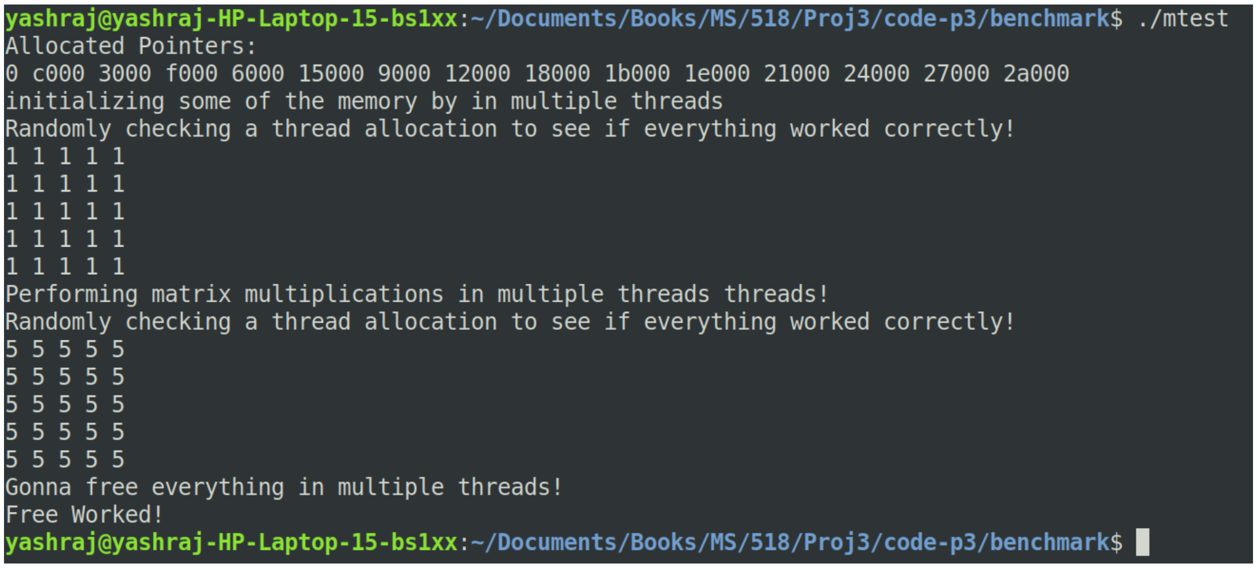
check_TLB(): Checks if translation of virtual address passed in exists in the TLB.

Output for Part 1:



```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./test
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!
Checking if allocations were freed!
free function works
TLB miss rate 0.007500
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

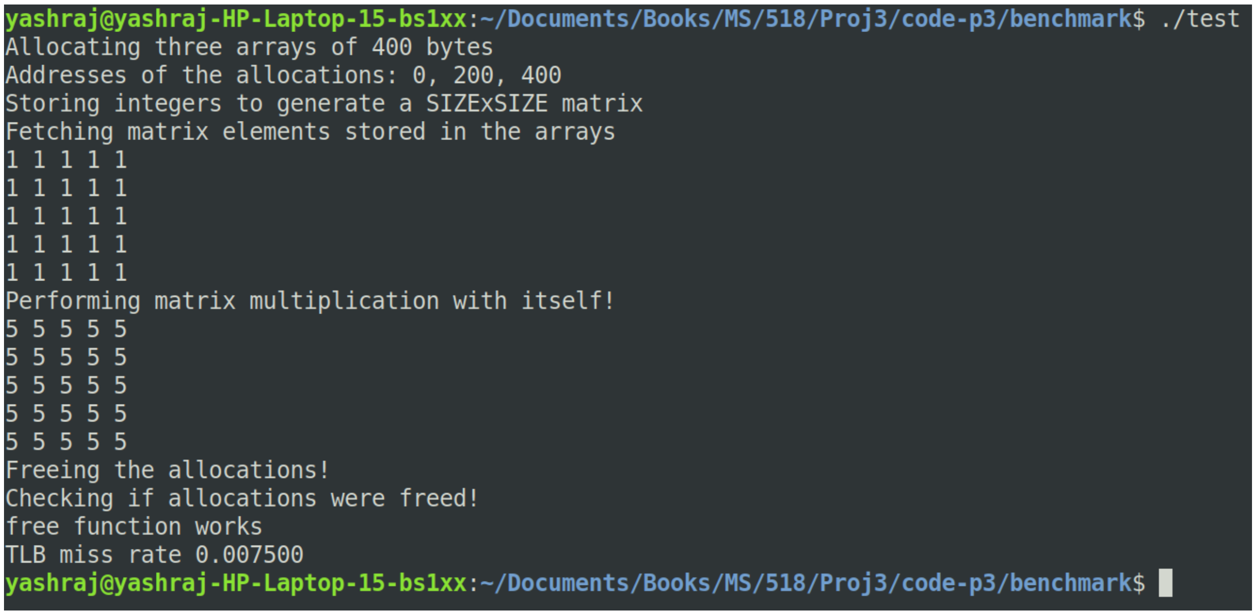
As seen in the output, the matrix multiplication result is as expected. The TLB miss rate is 0.007500.



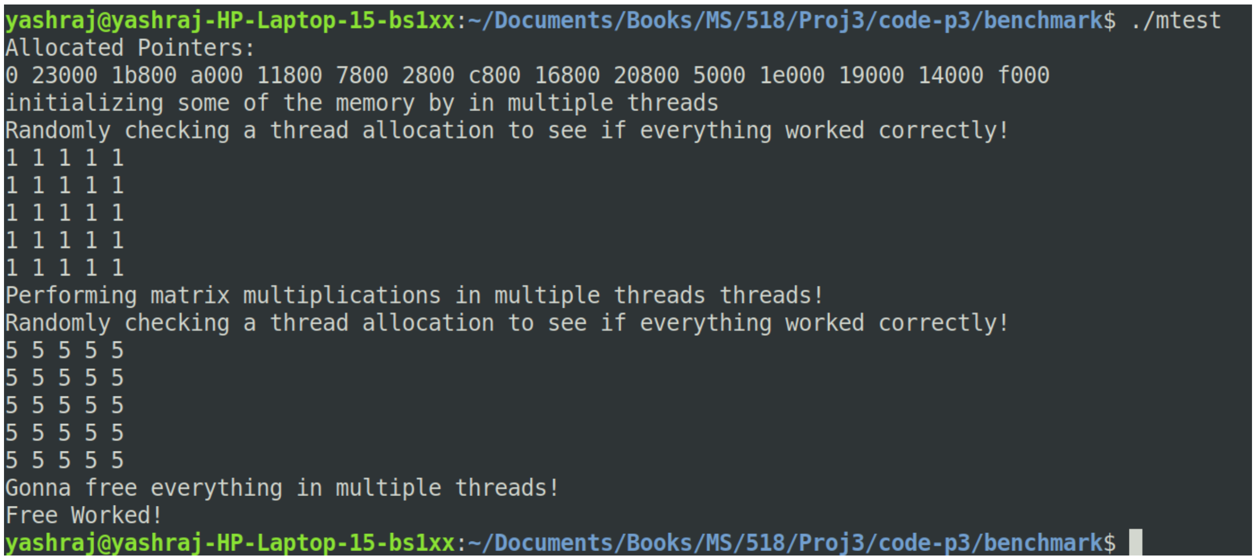
```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
Allocated Pointers:
0 c000 3000 f000 6000 15000 9000 12000 18000 1b000 1e000 21000 24000 27000 2a000
initializing some of the memory by in multiple threads
Randomly checking a thread allocation to see if everything worked correctly!
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 multiplications in multiple threads threads!
Randomly checking a thread allocation to see if everything worked correctly!
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
Gonna free everything in multiple threads!
Free Worked!
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

Support for different page sizes(test and multi_test):

512 bytes:

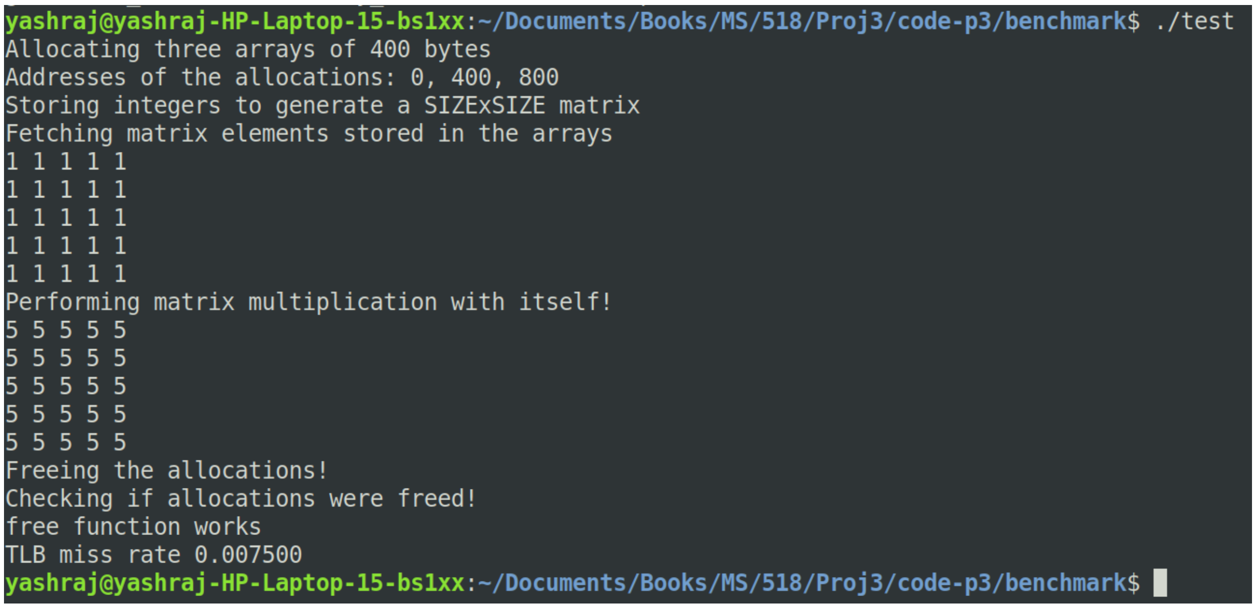


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./test
Allocating three arrays of 400 bytes
Addresses of the allocations: 0, 200, 400
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 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 0.007500
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

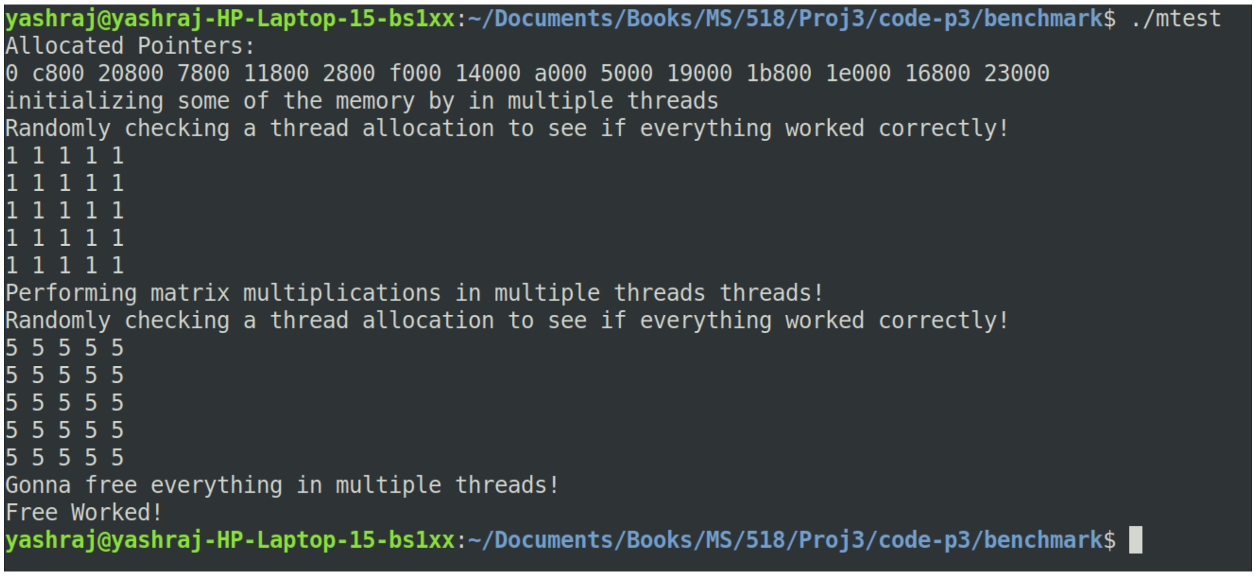


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
Allocated Pointers:
0 23000 1b800 a000 11800 7800 2800 c800 16800 20800 5000 1e000 19000 14000 f000
initializing some of the memory by in multiple threads
Randomly checking a thread allocation to see if everything worked correctly!
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 multiplications in multiple threads threads!
Randomly checking a thread allocation to see if everything worked correctly!
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
Gonna free everything in multiple threads!
Free Worked!
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

1kB:

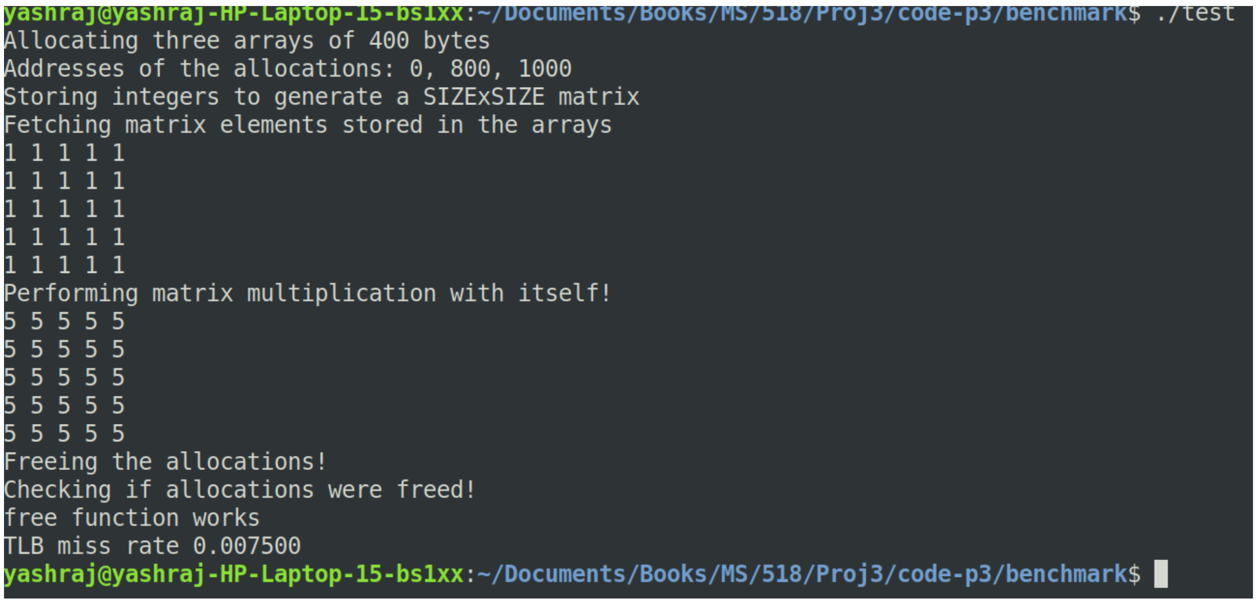


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./test
Allocating three arrays of 400 bytes
Addresses of the allocations: 0, 400, 800
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 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 0.007500
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

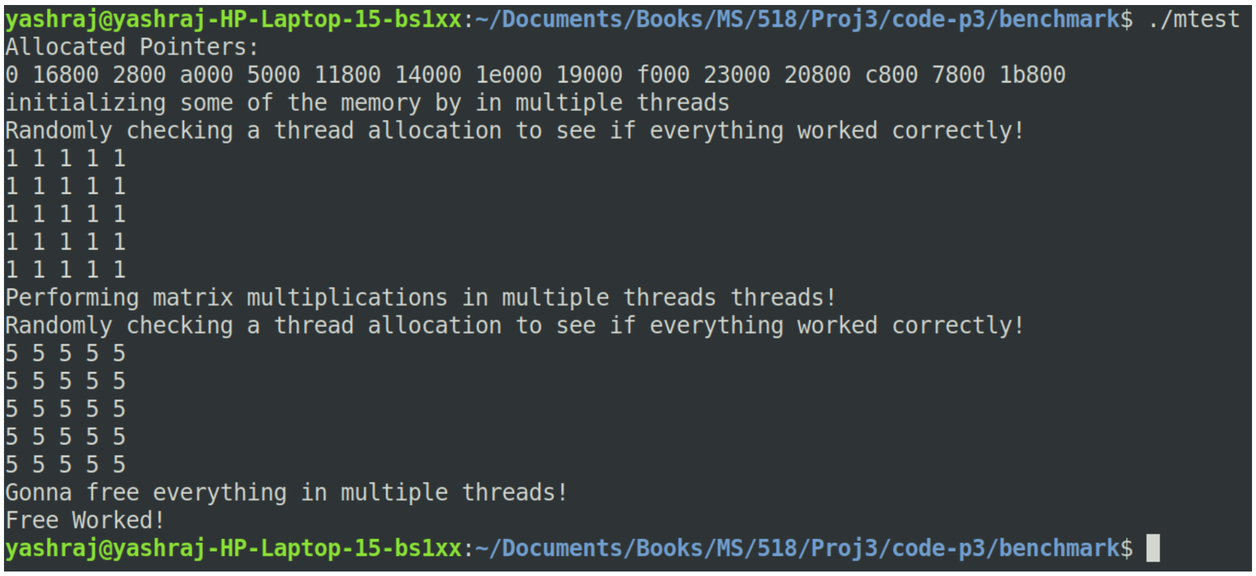


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
Allocated Pointers:
0 c800 20800 7800 11800 2800 f000 14000 a000 5000 19000 1b800 1e000 16800 23000
initializing some of the memory by in multiple threads
Randomly checking a thread allocation to see if everything worked correctly!
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 multiplications in multiple threads threads!
Randomly checking a thread allocation to see if everything worked correctly!
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
Gonna free everything in multiple threads!
Free Worked!
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

2kB:

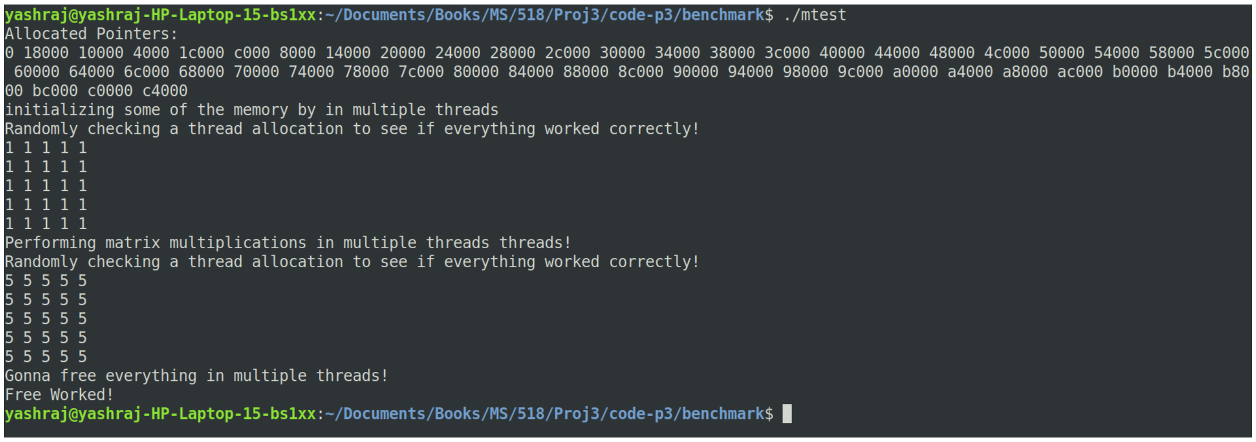


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./test
Allocating three arrays of 400 bytes
Addresses of the allocations: 0, 800, 1000
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 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 0.007500
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

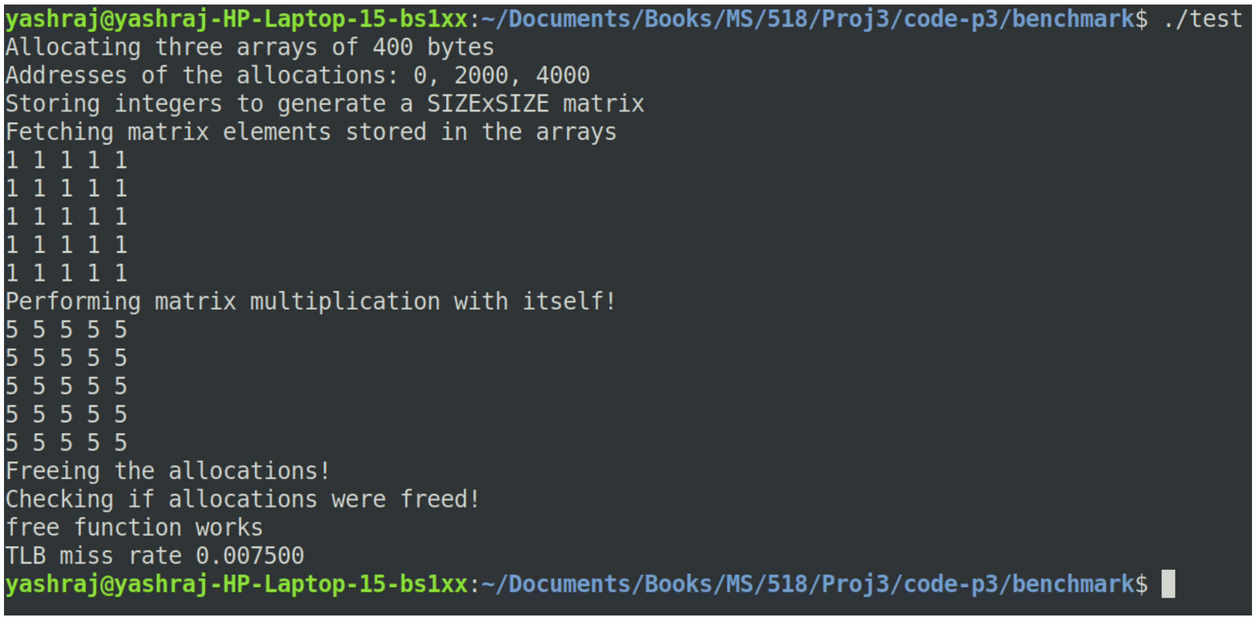


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
Allocated Pointers:
0 16800 2800 a000 5000 11800 14000 1e000 19000 f000 23000 20800 c800 7800 1b800
initializing some of the memory by in multiple threads
Randomly checking a thread allocation to see if everything worked correctly!
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 multiplications in multiple threads threads!
Randomly checking a thread allocation to see if everything worked correctly!
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
Gonna free everything in multiple threads!
Free Worked!
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

8kB:

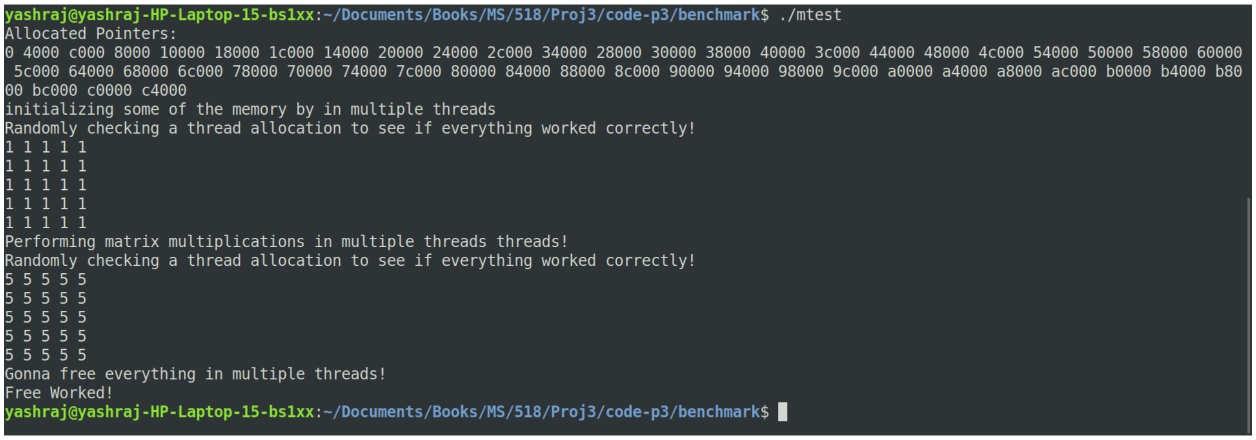


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
Allocated Pointers:
0 18000 10000 4000 1c000 c000 8000 14000 20000 24000 28000 2c000 30000 34000 38000 3c000 40000 44000 48000 4c000 50000 54000 58000 5c000
 60000 64000 6c000 68000 70000 74000 78000 7c000 80000 84000 88000 8c000 90000 94000 98000 9c000 a0000 a4000 a8000 ac000 b0000 b4000 b80
00 bc000 c0000 c4000
initializing some of the memory by in multiple threads
Randomly checking a thread allocation to see if everything worked correctly!
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 multiplications in multiple threads threads!
Randomly checking a thread allocation to see if everything worked correctly!
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
Gonna free everything in multiple threads!
Free Worked!
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

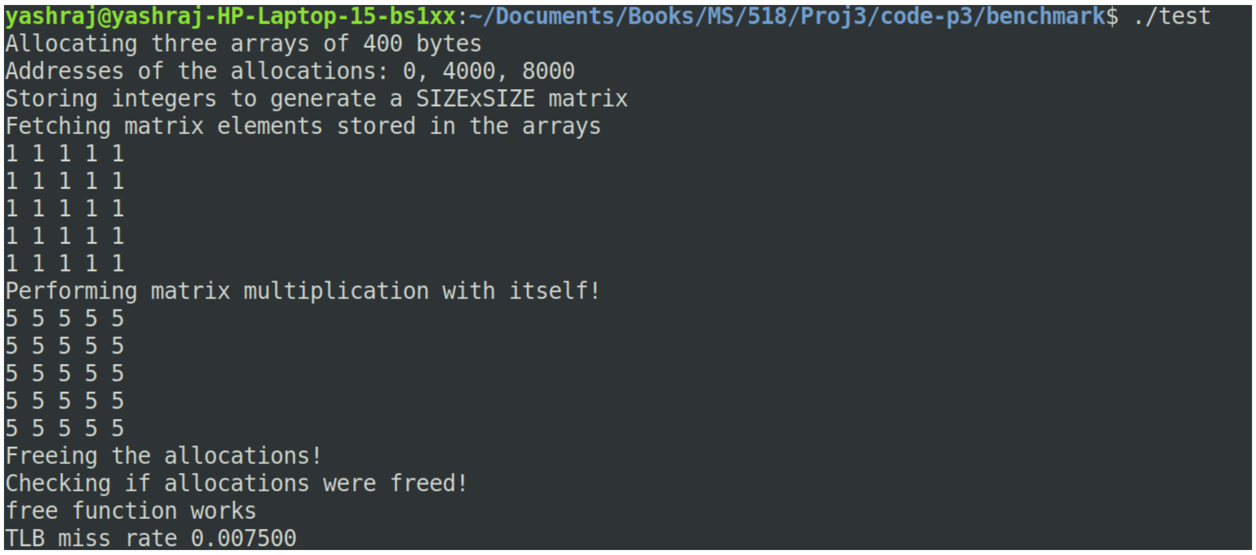


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./test
Allocating three arrays of 400 bytes
Addresses of the allocations: 0, 2000, 4000
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
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 0.007500
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

16kB:

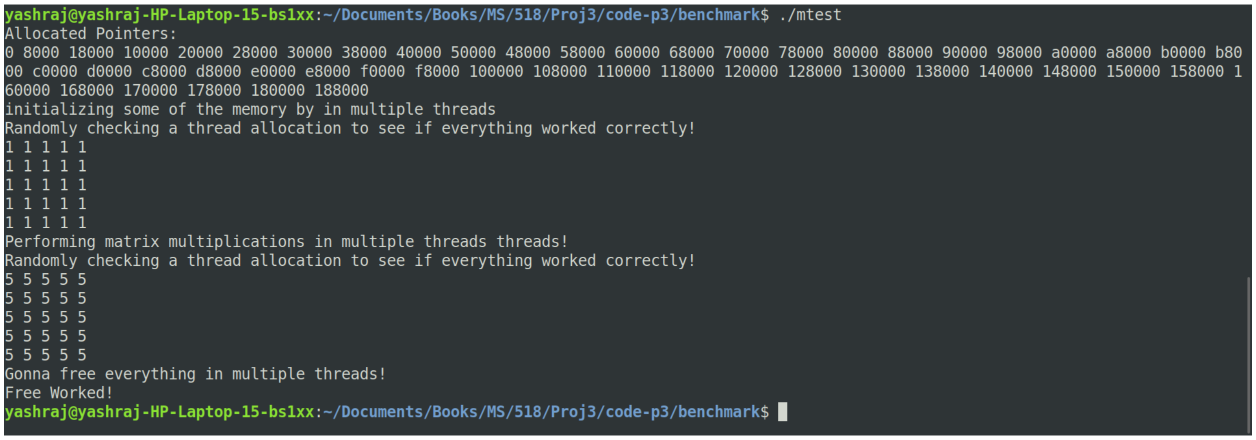


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
Allocated Pointers:
0 4000 c000 8000 10000 18000 1c000 14000 20000 24000 2c000 34000 28000 30000 38000 40000 3c000 44000 48000 4c000 54000 50000 58000 60000
 5c000 64000 68000 6c000 78000 70000 74000 7c000 80000 84000 88000 8c000 90000 94000 98000 9c000 a0000 a4000 a8000 ac000 b0000 b4000 b80
00 bc000 c0000 c4000
initializing some of the memory by in multiple threads
Randomly checking a thread allocation to see if everything worked correctly!
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 multiplications in multiple threads threads!
Randomly checking a thread allocation to see if everything worked correctly!
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
Gonna free everything in multiple threads!
Free Worked!
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$
```

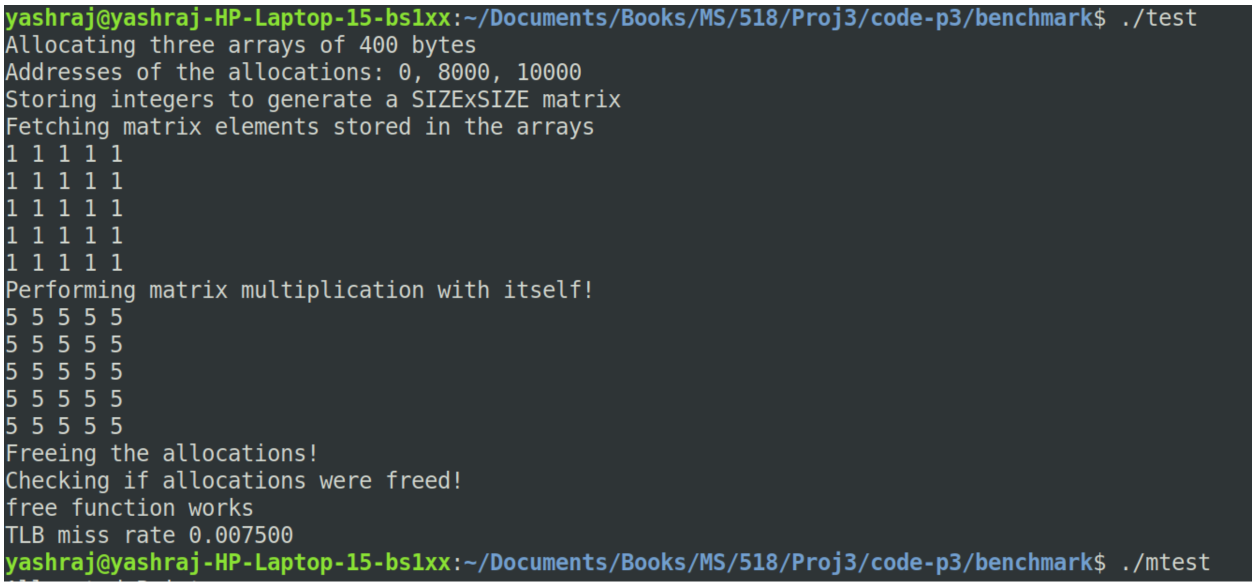


```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./test
Allocating three arrays of 400 bytes
Addresses of the allocations: 0, 4000, 8000
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
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 0.007500
```

32kB:



```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
Allocated Pointers:
0 8000 18000 10000 20000 28000 30000 38000 40000 50000 48000 58000 60000 68000 70000 78000 80000 88000 90000 98000 a0000 a8000 b0000 b80
00 c0000 d0000 c8000 d8000 e0000 e8000 f0000 f8000 100000 108000 110000 118000 120000 128000 130000 138000 140000 148000 150000 158000 1
60000 168000 170000 178000 180000 188000
initializing some of the memory by in multiple threads
Randomly checking a thread allocation to see if everything worked correctly!
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 multiplications in multiple threads threads!
Randomly checking a thread allocation to see if everything worked correctly!
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
Gonna free everything in multiple threads!
Free Worked!
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ 
```



```
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./test
Allocating three arrays of 400 bytes
Addresses of the allocations: 0, 8000, 10000
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 0.007500
yashraj@yashraj-HP-Laptop-15-bs1xx:~/Documents/Books/MS/518/Proj3/code-p3/benchmark$ ./mtest
```

Extra Credit:

1st Part: The number of levels are calculated dynamically by dividing the number of bits required for the total number of pages at the last level divided by the number of bits required to represent the number of entries in one page i.e. **calculateLevels()** by taking the ceil of (log of vpn) / (log of page table entries).