Assignment 4

Submit Assignment

Due May 4 by 10pm **Points** 75 **Submitting** a text entry box
Available Apr 15 at 8:15am - May 4 at 10pm 20 days

Assignment 4

Memory Allocation

Task

Using C++, create a memory management unit simulator that will implement dynamic storage allocation using paging. The page size will be determined based on a command line parameter, but must be a power of 2 (between 1024 and 32768). You will not actually be spawning processes that consume memory. Rather you will be creating simulated "processes" that each make a series of memory allocations and deallocations. The following are the actions that a process can do:

- Initialize
  - Assign a PID - unique number (start at 1024 and increment up)
  - Allocate some amount of startup memory for the process
    - Text/Code: user specified number (2048 - 16384 bytes)
    - Data/Globals: user specified number (0 - 1024 bytes)
    - Stack: constant (65536 bytes)
- Allocate memory on the heap
  - N chars (N bytes)
  - N shorts (N * 2 bytes)
  - N ints / floats (N * 4 bytes)
  - N longs / doubles (N * 8 bytes)
- Set value for allocated memory
  - Store integer, float, or character values in memory
- Deallocate memory from the heap
  - N chars (N bytes)
  - N shorts (N * 2 bytes)
  - N ints / floats (N * 4 bytes)
  - N longs / doubles (N * 8 bytes)
- Terminate
  - Deallocate all memory associated with the process

Your simulator should continually ask the user to input a command. Your program should interpret the following statements:

- create <text_size> <data_size>
  - Initializes a new process
  - Prints the PID
- allocate <PID> <var_name> <data_type> <number_of_elements>
  - Allocated memory on the heap (how much depends on the data type and the number of elements)
  - Print the virtual memory address
- set <PID> <var_name> <offset> <value_0> <value_1> <value_2> ... <value_N>
  - Set the value for variable <var_name> starting at <offset>
  - Multiple contiguous values can be set with one command
- free <PID> <var_name>
  - Deallocate memory on the heap that is associated with <var_name>
- terminate <PID>
  - Kill the specified process
- print <object>
  - if <object> is "mmu", print the MMU memory table
  - If <object> is "page", print the page table (do not need to print anything for free frames)
  - If <object> is "processes", print a list of PIDs for processes that are still running
  - If <object> is a "<PID>:<var_name>", print the value of the variable for that process
    - If variable has more than 4 elements, just print the first 4 followed by "... [N items]" (where N is the number of elements)
- exit
  - quit program

Your simulated machine will only have 64 MB of RAM (67108864 bytes).

To Earn a C (55/75 pts)

- Implement the 'create', 'allocate', and 'set', 'print', and 'exit' commands
- Each new allocation can be on a new page

To Earn a B or A

- Implement the 'free' and 'terminate' commands **(8 pts)**
- Use first fit algorithm within a page when allocating new data **(8 pts)**
  - allocate new page if no hole is large enough

- Print error message if an allocation would exceed system memory (and don't perform allocation) **(4 pts)**

Example

```
cmd$ ./memsim 8192
Welcome to the Memory Allocation Simulator! Using a page size of 8192 bytes.
Commands:
  * create <text_size> <data_size> (initializes a new process)
  * allocate <PID> <var_name> <data_type> <number_of_elements> (allocated memory on the heap)
  * set <PID> <var_name> <offset> <value_0> <value_1> <value_2> ... <value_N> (set the value for a variable)
  * free <PID> <var_name> (deallocate memory on the heap that is associated with <var_name>)
  * terminate <PID> (kill the specified process)
  * print <object> (prints data)
    * If <object> is "mmu", print the MMU memory table
    * if <object> is "page", print the page table
    * if <object> is "processes", print a list of PIDs for processes that are still running
    * if <object> is a "<PID>:<var_name>", print the value of the variable for that process

> create 5992 564
1024
> create 14788 296
1025
> allocate 1024 point_x int 1
72092
> allocate 1024 point_y int 1
72096
> allocate 1025 temperature double 1
80620
> allocate 1024 name char 256
72100
> allocate 1024 time long 2
72356
> allocate 1024 data int 2000
72372
> allocate 1025 pressure double 1
80628
> set 1024 point_x 0 100
> set 1024 point_y 0 200
> set 1024 name 0 l o c a t i o n
> set 1024 time 0 91246723975
> set 1024 time 1 91246724068
> set 1025 temperature 0 98.6
> set 1025 pressure 0 36.2
> print processes
1024
1025
> print page
 PID  | Page Number | Frame Number
------+-------------+--------------
 1024 |           0 |            0
 1024 |           1 |            1
 1024 |           2 |            2
 1024 |           3 |            3
 1024 |           4 |            4
 1024 |           5 |            5
 1024 |           6 |            6
 1024 |           7 |            7
 1024 |           8 |            8
 1024 |           9 |           19
 1025 |           0 |            9
 1025 |           1 |           10
 1025 |           2 |           11
 1025 |           3 |           12
 1025 |           4 |           13
 1025 |           5 |           14
 1025 |           6 |           15
 1025 |           7 |           16
 1025 |           8 |           17
 1025 |           9 |           18
> print mmu
 PID  | Variable Name | Virtual Addr | Size
------+---------------+--------------+------------
 1024 | <TEXT>        |   0x00000000 |       5992
 1024 | <GLOBALS>     |   0x00001768 |        564
 1024 | <STACK>       |   0x0000199C |      65536
 1024 | point_x       |   0x0001199C |          4
 1024 | point_y       |   0x000119A0 |          4
 1024 | name          |   0x000119A4 |        256
 1024 | time          |   0x00011AA4 |         16
 1024 | data          |   0x00011AB4 |       8000
 1025 | <TEXT>        |   0x00000000 |      14788
 1025 | <GLOBALS>     |   0x000039C4 |        296
 1025 | <STACK>       |   0x00003AEC |      65536
 1025 | temperature   |   0x00013AEC |          8
 1025 | pressure      |   0x00013AF4 |          8
> print 1024 name
l, o, c, a, ... [256 items]
> print 1024 time
91246723975, 91246724068
> free 1024 data
> print page
 PID  | Page Number | Frame Number
------+-------------+-------------
 1024 |           0 |            0
 1024 |           1 |            1
 1024 |           2 |            2
 1024 |           3 |            3
 1024 |           4 |            4
 1024 |           5 |            5
 1024 |           6 |            6
 1024 |           7 |            7
 1024 |           8 |            8
 1025 |           0 |            9
 1025 |           1 |           10
 1025 |           2 |           11
 1025 |           3 |           12
 1025 |           4 |           13
 1025 |           5 |           14
 1025 |           6 |           15
 1025 |           7 |           16
 1025 |           8 |           17
 1025 |           9 |           18
> terminate 1025
> print mmu
 PID  | Variable Name | Virtual Addr | Size
------+---------------+--------------+------------
 1024 | <TEXT>        |   0x00000000 | 5992
 1024 | <GLOBALS>     |   0x00001768 | 564
 1024 | <STACK>       |   0x0000199C | 65536
 1024 | point_x       |   0x0001199C | 4
 1024 | point_y       |   0x000119A0 | 4
 1024 | name          |   0x000119A4 | 256
 1024 | time          |   0x00011AA4 | 16
> exit
```

Note: lines starting with ">" are user input. All other lines are output from the program.

Instructions

You will be working in teams of 2 students to complete this project. You are allowed to work together on all aspects of the project or to divide tasks equally among the members. You are NOT however allowed to collaborate with students from other teams. The only exception to this is posting general questions/answers to the discussion board on Canvas.

Starter code is provided in **assignment04.zip**, which is available on Canvas.

Code should be saved in a repository on GitHub with all team members as collaborators.

Submission

Each group member should independently submit the following in Canvas:

- Project's GitHub URL
- Compilation and running instructions
- List of what each team member contributed to the project
- Expected grade (number of points earned)

Deadline

This assignment is due Monday, May 4 at 10:00pm.

Teams

Team A	**Team B**	**Team C**	**Team D**
• Alex • John	• Bernadette • Brandon	• Josh • Nick P.	• Yuki • Devin
Team E	**Team F**	**Team G**	**Team H**
• Tyler • Matt	• Nick H. • Abby	• Paige • Keith	• Jesse • Chase
Team I			
• Jack			