CS232 Operating Systems
Assignment 3: Stack and Heap Memory Management

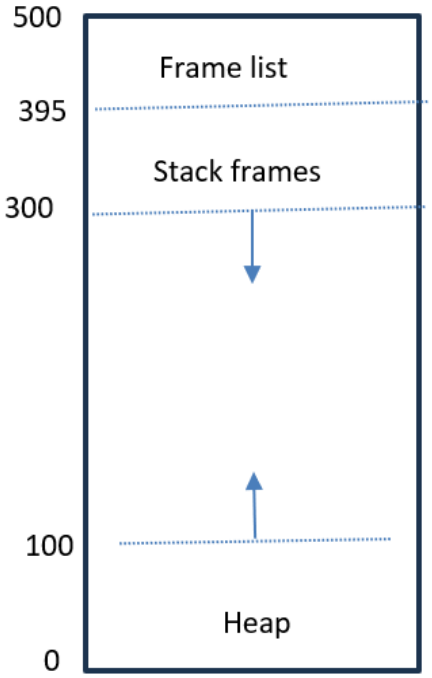
CS Program
Habib University

Fall 2023
Due Date: 13 November 2023 @ 11:59PM

1 Introduction

This assignment simulates a simple stack and heap memory system.

1. The whole memory allocated is 500 Bytes in size.
2. The stack starts at location 500 and grows towards lower addresses.
3. The heap starts at location 0 and grows towards higher addresses.
4. The stack can have a maximum of 5 frames. One frame per function.
5. The stack frame stores local variables, function return addresses and pointers.
6. The frame pointer points to the start address of the frame currently in execution.
7. The stack can grow up to a maximum of 300 Bytes.
8. The heap can grow up to a maximum of 300 Bytes.



2 Stack Memory Layout

The stack memory layout is as follows: Initially 100 bytes, the stack can grow up to a maximum of 300 bytes. The stack is further divided into stack frames and frame status (fstatus).

The frame status is 21 bytes (each) in size and contains metadata about the stack frames as indicated by the data structure in the accompanying memorysystem.c. The frame status stores the frame number (4 bytes), function name (8 bytes), function address (4 bytes), frame address (4 bytes), and a boolean value (1 byte) indicating whether that particular frame is top of stack or not.

Just after the frame status list (in the stack), we have the stack frames (maximum of 5 frames). Each frame can store integers (4 bytes each), doubles (8 bytes each), character (1 byte each), and data pointers (4 bytes each). The minimum size of the stack frame is 10 bytes and the maximum size of stack frame is 80 bytes.

3 Heap Memory Layout

The heap memory layout is as follows: The heap can grow up to a maximum of 300 bytes. For each heap allocation, the system allocates the requested number of bytes plus 8 bytes (for storing the size of allocated region and random generated magic number).

The system maintains a free list which shows the memory segments that are currently free and not allocated.

4 Required Tasks

Your program should be able to process the following commands:

4.1 Create a frame

syntax: CF functionname functionaddress

This command should create a frame on stack. The functionname is a maximum of 8 characters. The functionaddress is an assumed address of the function in program code.

If there's not enough space on stack, it should output an error saying "stack overflow, not enough memory available for new function". If all the maximum number of frames have reached, it should output an error saying "cannot create another frame, maximum number of frames have reached".

If a function with the given name already exists, it should give an error "function already exists".In case of no errors, it should create a frame on stack and create an entry in sframe.

4.2 Delete a Function

syntax: DF

This command deletes the function on top of the stack.
If no function exists on stack, it should output an error message saying "stack is empty".

4.3 Create integer local variable

syntax: CI integername integervalue

This command creates an integer of size 4 bytes on the current frame. If the frame is full, it should output an error message saying "the frame is full, cannot create more data on it".

4.4 Create double local variable

syntax: CD doublename doublevalue

This command creates a double of size of 8 bytes on the current frame. If the frame is full, it should output an error message saying "the frame is full, cannot create more data on it".

4.5 Create character local variable

syntax: CC charactername charactervalue

This command creates a character of size of 1 byte on the current frame. If the frame is full, it should output an error message saying "the frame is full, cannot create more data on it".

4.6 Create character buffer on heap

syntax: CH buffername size

This command allocates a buffer of bytes size plus 4 bytes on heap. It also creates a local pointer on stack and stores the starting address of the allocated region. The buffer is filled with random characters. If the heap is full, it should output an error message saying "the heap is full, cannot create more data".

4.7 Deallocate a buffer on heap

syntax: DH buffername

This command de-allocates a buffer on stack. A total of buffer size plus 4 bytes are deallocated. The data in the deallocated region is replaced with zeros. If the buffer was already de-allocated or the pointer is invalid, output an error message saying "the pointer is NULL or already de-allocated".

4.8 Show memory image

syntax: SM
This command should output the stack and heap snapshots.

5. Input

Your program should give an interactive (shell like) environment, where the user can interact with the system and run the above commands. It should read the commands and execute them. You can assume that the input will always be in the correct format.
After executing every command, the program should update the state of the heap and stack memory in an in-memory data structure.
At the start of your program, it should create a character data structure (of size 500 bytes) that will hold stack and heap data and metadata. Format the allocated data structure with "spaces" for better readability.

6. Submission and Rubric

6.1 Submission

You will submit the following:

1. memorysystem.c containing your complete C code
2. makefile for compiling and running of your C code. It should contain compile, build and clean targets.
3. PDF report on the data structures and algorithms you have used in your implementation.

Compress all of these files into a zip file and rename it with your CS registration number (CSxxyyyy.zip). Submit the zip file on the Assignment 1 submission module on LMS.

6.2 Rubric

The details are given in the following. Note that you might also be called for a viva at the instructor's discretion.

1. **Marks**

   commands work as specified: - 80 marks

   makefile: - 10 marks

   submission (code legible, commented, PDF correctly formatted): - 10 marks

2. **Penalties**

   code doesn't compile: -100 marks

   code has warnings (compile with -Wall): -20 marks

   code has memory leaks: -30 marks

   makefile without required targets: -10*number of missing targets

   program crashes: -30 marks

   late submission: -20 marks for missing deadline + -10*num days Mark

obtained = max (marks+penalties, 0)

7. Using chatGPT or other AI software

You are not allowed to use any AI software to obtain the code for this assignment. Appropriate tool will be used to evaluate your submission for AI tool usage. If you are found using such a tool, you will be given a straight 0 and an Academic Conduct will be filed against you for academic dishonesty.

8. Plagiarism Policy

We have zero tolerance for plagiarism. Every submission will be screened using a plagiarism detection software. If there is any evidence of plagiarism, the case will be reported to the Office of Academic Conduct and all offenders will get a 0. This is applicable even for cases when the code is copied or a significant amount has been obtained from an online repository on open-source platforms like bitbucket or github without proper attribution. In case you are taking any material from online sources, we expect that a proper credit/reference is given to the source.