Dynamic Memory Allocation

Last updated 1/10/19

Dynamic Memory Allocation

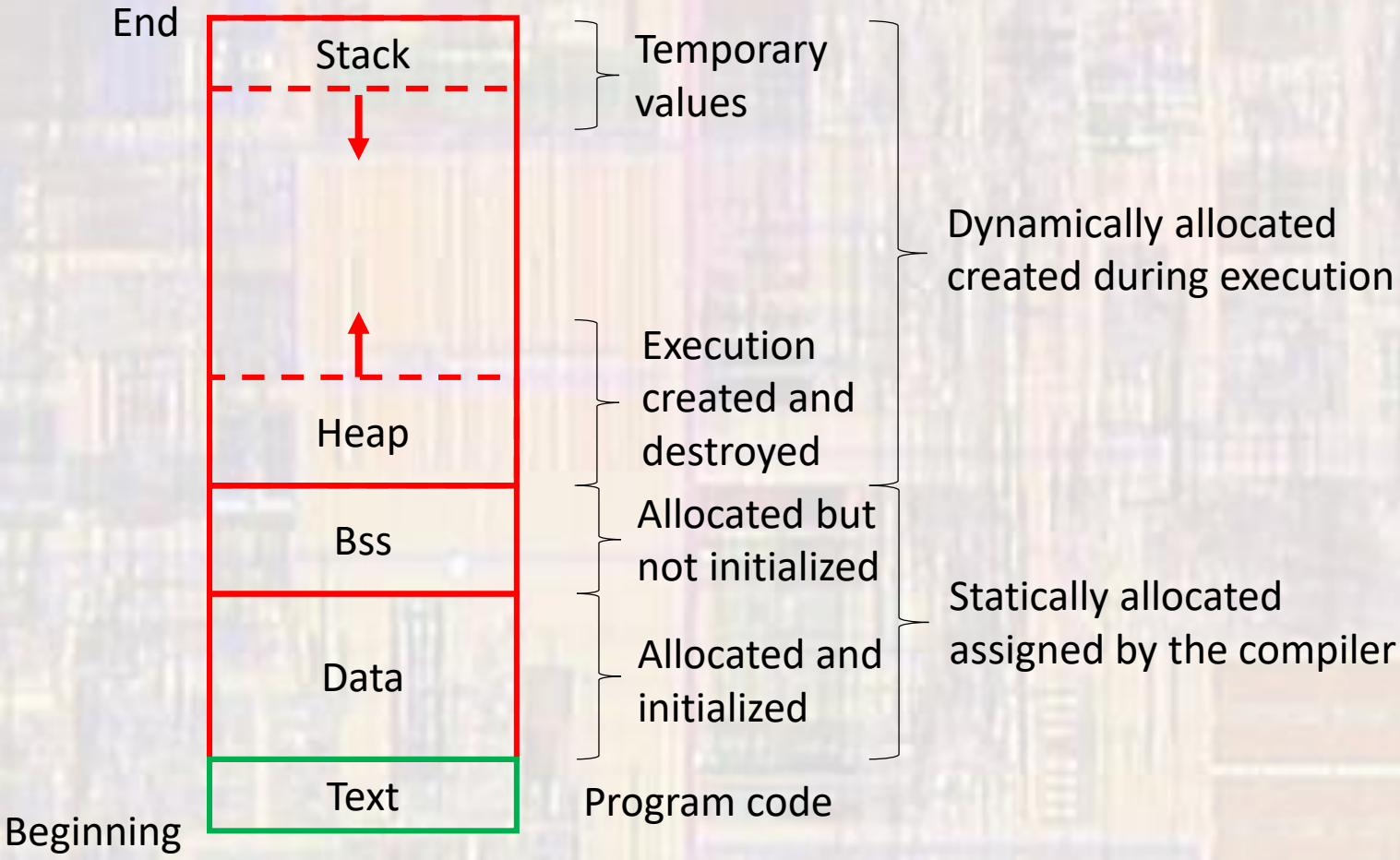
- Stack
  - A section of Data memory
  - Used to hold all temporary variables whose size is known at compile time
    - Return address for a function
    - Copies of parameters passed into a function
    - Temporary variables used in a function
      - Counters, …
      - An array with 26 elements inside a function
      - Note – main is just another function

Dynamic Memory Allocation

- Heap
  - Section of Data memory
  - Dynamic memory
    - Created and destroyed by the program
  - Persists until you de-allocate it
  - Typically dependent on run time information
    - The heap is used to hold all variables whose size are not known at compile time
      - Store a list of numbers from the user, where the # of inputs is not known ahead of time
  - Can be accessed throughout the program and it's functions

Dynamic Memory Allocation

- Data Memory
  - Stack and heap grow towards each other



Dynamic Memory Allocation

- Allocating dynamic memory
  - Use the new operator
    - Assign a chunk of memory in the heap
    - Operand is a “type” and optionally a number of elements
    - Evaluates to a pointer – pointing to the beginning of the chunk of memory

```
int * foo_ptr;
foo_ptr = new int;

float * boo_ptr;
boo_ptr = new float[i];

char * soo_ptr = new char[i];
```

Allocates a number of memory elements (not an array)

i would typically be a run time value

Dynamic Memory Allocation

- Cleaning up dynamic memory
  - Dynamic memory allocated during program execution persists until either
    - The end of the run
    - The memory is de-allocated
  - Failure to clean up no longer needed allocated memory can cause the program to run out of memory over time
    - Called a memory leak

```
delete foo_ptr;

delete[] boo_ptr;

delete[] soo_ptr;
```

Dynamic Memory Allocation

- Allocating dynamic memory
  - What happens if there is no memory left to allocate
    - System “throws” an exception
    - 2 approaches to deal with the exception
      1. Prevent the system from throwing it ✓
      2. Setup an exception handler

Dynamic Memory Allocation

- Allocating dynamic memory
  - Preventing the system from throwing an exception
    - Tell the system not to throw an exception
    - (nothrow) added to the new operator

```
int * foo_ptr;
foo_ptr = new (nothrow) int;

float * boo_ptr;
boo_ptr = new (nothrow) float[i];

char * soo_ptr = new (nothrow) char[i];
```

i would typically be a run time value

Dynamic Memory Allocation

- Allocating dynamic memory
  - Preventing the system from throwing an exception
    - But what if we run out of memory?
      - If it cannot allocate the memory it will not create the pointer
        - The pointer variable's value will be null "0"
      - We can test for this and exit cleanly
        - Requires inclusion of <cstlib>

```
float * boo_ptr;
boo_ptr = new (nothrow) float[i];
if(boo_ptr == 0){
   cout << "Error: Could not create dynamic memory.\n";
   exit(EXIT_FAILURE);
}
```

requires <cstdlib> be included