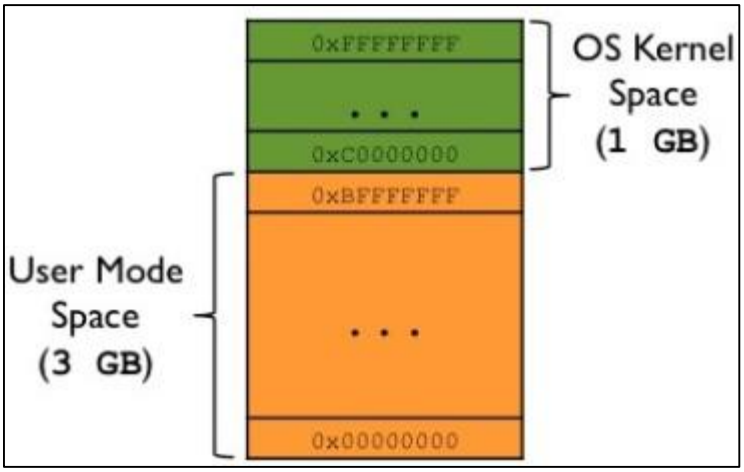
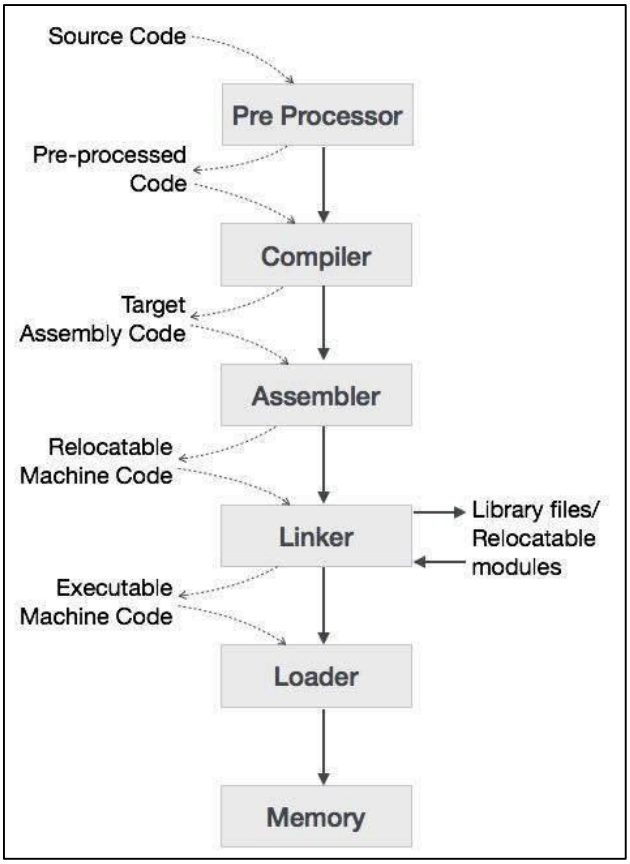
Major Sections of Computer Memory

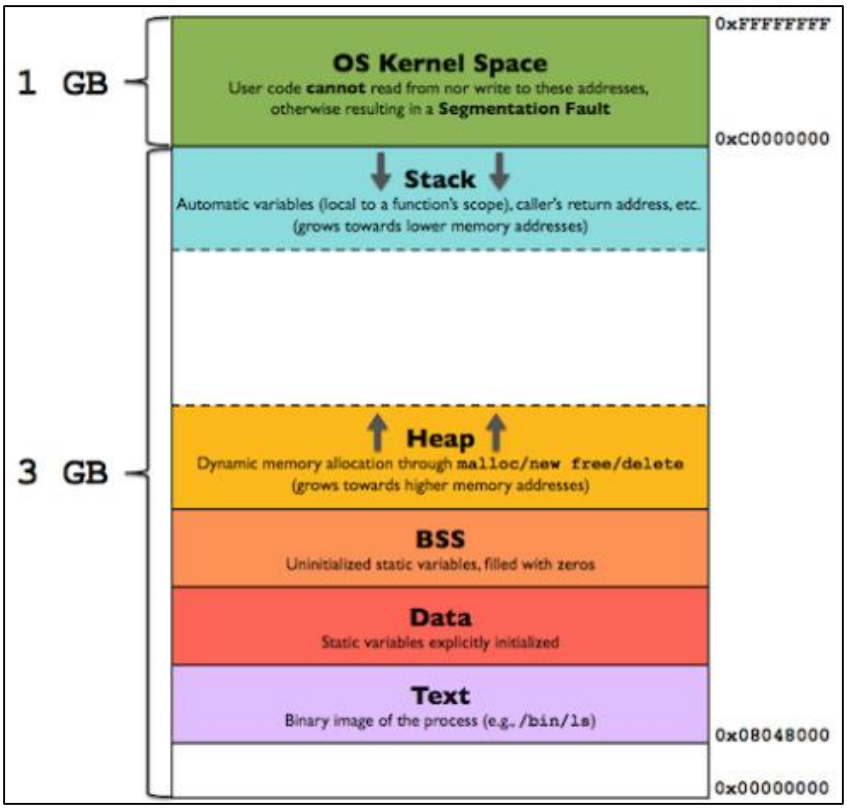


Language Processing System:



MEMORY REPRESENTATION OF A C PROGRAM:

A typical memory representation of a C program consists of the following sections.



1. Text/Code segment
2. Data Segment
    a. Initialized data segment
    b. Uninitialized data segment
3. Stack segment
4. Heap segment

1. Text Segment:

- Text/Code segment
- contains executable instructions
- May be placed below the heap or stack in order to prevent heaps and stack overflows from overwriting it

- Often read-only, to prevent a program from accidentally modifying its instructions

2. Data Segment:

a) Initialized data segment

- Contains the global variables and static variables that are initialized by the programmer
- is not read-only, since the values of the variables can be altered at run time

b) Un-initialized/BSS Segment:

- "Block start by Symbol"
- Contains all global and static variables is initialized by the OS kernel to arithmetic 0 before the program starts executing.

3. Stack:

- The stack area traditionally is adjoined the heap area
- grows in the opposite direction; when the stack pointer met the heap pointer, free memory was exhausted.
- The stack area contains the program stack, a LIFO structure,
- Typically located in the higher parts of memory
- A "stack pointer" register tracks the top of the stack; it is adjusted each time a value is "pushed" onto the stack.
- The set of values pushed for one function call is termed a "stack frame"; A stack frame consists at minimum of a return address.
- Stack, where automatic variables are stored, along with information that is saved each time a function is called.
- The newly called function then allocates room on the stack for its automatic and temporary variables. This is how recursive functions in C can work.
- Each time a recursive function calls itself, a new stack frame is used, so one set of variables doesn't interfere with the variables from another instance of the function.

4. Heap:

- Heap is the segment where dynamic memory allocation usually takes place.
- The heap area begins at the end of the BSS segment and grows to larger addresses from there.
- The Heap area is manually managed by malloc, calloc, and free in c language.