# Memory Allocation & Function Implementation

Xiang Yu, [xiangyu@cs.stonybrook.edu](mailto:xiangyu@cs.stonybrook.edu)

## Introduction

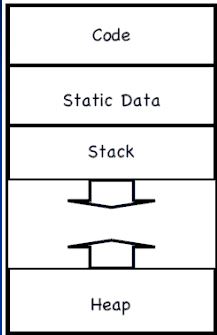
During the procedure of Immediate Code Generation, we need to implement the mechanism of Memory Allocation and Function Implementation, and the latter one can be seen as part of the former.

As we know, icode supports limited registers and memory. Registers are used for intermediate temporaries during operations; while memory holds the most part of all the data used by the program.

Using the classical model, we designed the layout of memory as follows (also you can see it in the figure):

1. Code Region. For storage of generated executive 3-address code, the size of which can be determined during compilation.
2. Static Region. To store global variables and constant variables like constant integers and string. We can also determine its size during compilation.
3. Stack Region. Used for storage of activation records, the implementation of functions. The size of it varies during runtime.
4. Heap Region. We can allocate and free data for variables dynamically during runtime. Thus, its size is not fixed, either. Pointer of its address moves in the opposite direction of Stack Region.

Besides, we need to set guard zones for stack region and heap region, in case that the increasing pointer causes out-of-boundary memory access. All the memory addresses for saving data are aligned with one memory word.



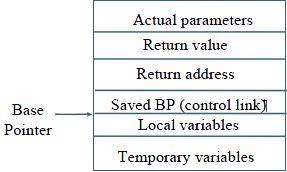
## Details

Data Access supported inside operations:

1. Operations between registers: using 3-address code to directly access registers.
2. Operations related with variables in memory: by absolute and relative memory access to retrieve the data stored in the address corresponded with the variable.

The structure of Activation Records (as described in figure):

The activation records would be responsible for the memory allocation of local variables in the implementation of functions. Control link points to the caller of this callee. SP is used to record the current pointer during execution of sentences in functions. In such a structure, the main procedure and recursive procedures are just activation records in stack region.



Dynamic Memory Allocation in Heap:

The mechanism for dynamic memory allocation in our compiler is letting user to arrange the allocation and deallocation of data manually and explicitly.

If there is enough time for us, we’d like to implement the garbage collector, with a simple version of reference counting.