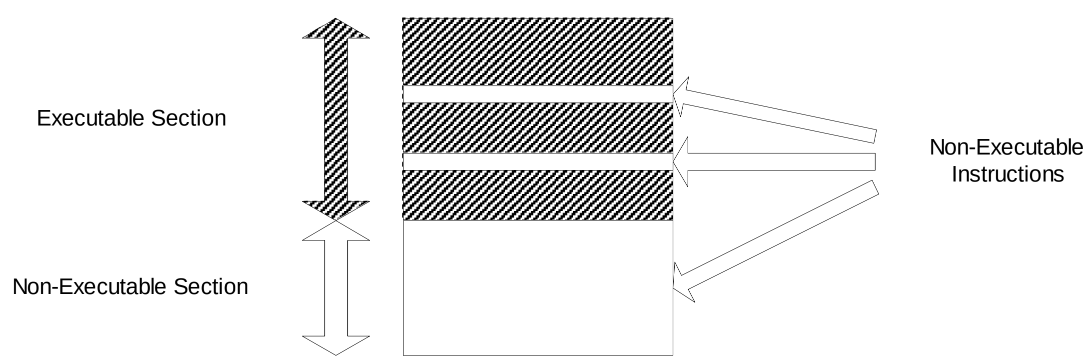
Organizing a Simple Program

We use the term "simple" to refer to an assembler program that is organized using a single control section. Programs that involve multiple control sections are the topic of another article. Every program can be divided into two parts:

An **executable section** that consists of all the instructions in the program which direct the action of the central processing unit. This includes all the instructions in the instruction set. Some examples are **MVC**, **AP**, and **CR**. These instructions move data, perform arithmetic, and compare fields. Also included in the executable section are executable macros like **GET**, **PUT**, **OPEN** and **CLOSE**. These macros generate executable instructions from the instruction set.

A **non-executable section** that consists of all the "directive" statements in the program that control how the assembler processes the source code. These include the **DS** and **DC** directives (used to create variables and storage areas), **EQU** directives (used for register names and target labels), the **LTORG** directive (used to create a literal pool), **USING** and **DROP** directives (used to control register selection by the assembler), and a several "cosmetic" directives like **PRINT**, **EJECT** and **SPACE**.

Physically, the executable section and the non-executable section are intermixed. If you think of the executable section being colored "red", and the non-executable section being colored "blue", then from afar, our "simple program" will appear as a red block on top of a blue block. A closer look will reveal that the red block contains a few scattered lines of blue code. This organization is indicated in the diagram below.



The Executable Section

Next we consider the organization of the executable section. There are several things that must occur upon entry into every program. We will collectively refer to these tasks as the "standard entry section". These tasks are several:

1) Copy the current contents of the registers into a save area which is located in the non-executable part of the program,

2) Establish addressability (the ability to use symbols),

3) Prepare the program for calling other programs.

After the standard entry section the programmer is free to write the code that comprises the body of the program. This section instructs the central processing unit to perform the tasks for which the program is being written.

Finally, coding the “standard exit section” completes the executable section. There are three tasks in this section:

1) Restore the registers to their original state upon entering the program,

2) Set a return code which will be used by the calling program or operating system,

3) Branch back to the calling program or operating system.

A thorough description of the standard entry and exit sections can be found **in Program Linkage**.

Scattered throughout the executable section you will see the following types of non-executable statements:

1) The **CSECT** directive that delimits the beginning of the control section,

2) Cosmetic directives like **PRINT**, **EJECT**, and **SPACE** which are used to control the appearance of the assembler listing of the source code,

3) Comments,

4) **EQU** directives which are used to create symbols and labels, and

5) **USING** and **DROP** directives that control the registers selected by the assembler when creating base/displacement addresses.

The Non-Executable Section

The non-executable section follows the executable section. While the programmer has wide latitude in organizing any assembler program, we will outline a typical layout of a “simple” program’s non-executable section. The typical organization would include the following items in order:

1) DCB’s (data control blocks) that define the files that will be processed by the program,

2) DS and DC directives for creating variables and storage areas used by the executable instructions,

3) The LTORG directive which indicates the position of the literal pool in the program,

4) The END directive which delimits the physical end of the program, and

5) Comments which may be scattered throughout the non-executable section.