Now enter the contents of the Label, Mnemonic, and Remark columns. Because our *Editor* is in BASIC, your text input will be slower than with a full-blown word processor. The Label column is empty in line 001 of our sample program, so press the proper key or key combination (see your Control Capsule) to tab into the Mnemonic field. (We will explain labels below.) Now type in the first instruction: LDA# 3. You must enter the text *exactly* as it appears in the listing, or the *NanoAssembler* program will not interpret the code properly. Make sure there is no space between the A and the #. You must, however, place a space between the # and the 3.

This spacing is critical because the Mnemonic field actually consists of two sub-fields; and the space acts as a separator for these sub-fields. The left sub-field is the "op-code," or instruction field, which defines the actual instruction. In line 001, the op-code is LDA#. The right field contains the "operand." The operand is either a two-nibble address or a single-nibble quantity to be loaded or stored in a register or memory location. It defines the number that the op-code is to operate on. In line 001, the number 3 (%0011) is the operand.

After you have entered the first instruction, you may tab into the Remark field. On a program as short as this one, however, you may choose to save time by omitting the remarks. Continue entering lines 002, 003, and 004 in a similar fashion.

Once you've entered part or all of the program into memory using the Add command, you can use the other editing commands. Each of these commands prompts you for a particular line number. E lets you Edit an already-existing line in memory. D allows you to Delete a line, and I lets you Insert a line. The L command lets you List up to 10 lines of a program to inspect what is in memory. If the program extends more than 10 lines beyond the beginning line number that you specify, you have the option to either continue listing more lines or quit and return to the command line.

Labels As Labor Savers

In line 005 (HERE JMP HERE), you encounter an important assembly-language tool—the "label." In the *NanoAssembler*, we define a label as a group of up to 6 alpha-numeric characters, beginning with a letter—in our example, the word HERE. Assembler programs use labels in place of numeric quantities. In this case, HERE represents the address to be JuMPed to. One major advantage of labels is that you do not have to know the actual numeric addresses used in a program. Instead, the assembler uses the labels to assign the correct address to a particular instruction for you.

Before continuing, let's clear up an area that sometimes confuses a beginner at assembly language: the difference between line numbers of a source file and addresses of an object file. Each line in a source file contains only one op-code. But when you assemble the source file into object code, the op-code may require as many as three addresses (see Figure 1 for the number of nibbles each instruction requires). Thus, a source file's line numbers and the actual addresses of the object code almost always differ. When the *Assembler* prints out its listing, the addresses and codes are located on the line just below the source code, representing the order of events during assembly.

By inspecting the two left-hand columns of Sample Program 1, you can see that the address to be JuMPed to is 6. You know this only because we have already assembled (or translated) the source code on the right into the object code on the left. If we hadn't provided the machine code, however, you would have to assemble all of the instructions to discover what address you wanted to JuMP to. The use of labels saves you from this tedious task and is one of the primary advantages of assemblers.

When you finish entering line 005 and press [RETURN] or [ENTER], a prompt tells you to enter line 6. This program has no line 006, so press the [ESCAPE] key for your machine (see your Control Capsule), and the program returns you to the command line. Now you can use the List command to see if you have entered everything correctly. If you find any errors, you can Edit the line or lines that they occur in. If you change a line, then decide that you don't want those changes, you can press the [ESCAPE] key instead of [RETURN] or [ENTER] to revert back to the original version of the line. This option is also available if you select Insert, but change your mind before finally entering the line.

From Editor To Assembler

After you are sure that you've correctly entered the program, save it to disk (or tape on Atari, C-64, or TI). To save your file, select option (2) Files. Then select the appropriate menu options, and enter the file name. If your operating system does not normally support extensions to file names (all but Atari and IBM), the name must be at least two characters shorter than a normal legal file name. The program will automatically append a .S (_S on the TI), for Source, so that you can use the same name for both source and object files without any confusion. If you have a printer, you may also wish to get a hardcopy of your program. This is helpful when you are tracking down errors during assembly. To use

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
ESC	Escape
Edit Mode:	
BACKSPACE	Backspace
CONTROL D	Erase line
TAB	Tab.
←	Cursor left
→	Cursor right
RETURN	Enter Line

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
ESC	Escape
Edit Mode:	
DELETE	Backspace
SHIFT DELETE	Erase line
TAB	Tab
CONTROL ←	Cursor left
CONTROL →	Cursor right
RETURN	Enter Line

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
F1	Escape
Edit Mode:	
DEL	Backspace
F3	Erase line
F5	Tab
CRSR ←	Cursor left
CRSR →	Cursor right
RETURN	Enter line

CONTROL CAPSULE

NanoEditor

KEY	FUNCTION
ESCAPE	Escape
Edit Mode:	
BACKSPACE	Backspace
DELETE	Delete character
TAB	Tab
←	Cursor left.
→	Cursor right
ENTER	Enter line

CONTROL CAPSULE

NanoEdtior

KEY	FUNCTION
FTCN 9	Escape
Edit Mode:	
FCTN 1	Delete
FCTN 3	Erase line
FCTN 7	Tab.
FCTN S	Cursor left
FCTN D	Cursor right
ENTER	Enter line