

## **Code Size Determination**

### **Method 1: Use the Programmer's Guide to determine opcode and operand size.**

Each instruction supported by the processor is detailed in the Programmer's Guide for the processor being used. The guide will indicate the general form for the instruction and will show how many bytes are required. The firmware writer can simply examine the data sheet for each assembly instruction used in the code and add up the total number of bytes required.

### **Method 2: Use the listing (.LST) file to view the machine code.**

The firmware writer can simply direct the development tools to generate a listing file when the source file is assembled. The listing file contains all the original assembly source code, as well as the resulting machine code and the addresses at which the machine code will be stored in the memory space. It is very simple to examine each line and see how many bytes were generated by each assembly instruction.

**Note:** Some of the early examples used specifically with the Emily52 simulator make use of the 'illegal' opcode \$A5 (A5H). This is the only opcode left undefined in the 8051/C501 architecture. When the Emily52 simulator encounters this illegal opcode, Emily52 will stop; this provides an easy way to terminate a program that is being simulated. On real hardware, this opcode may be ignored by the processor, since the opcode is undefined. The hardware won't actually stop, since the simple 8051 architecture does not support exceptions, such as 'illegal opcode'. When calculating code size, students should **not** count the illegal opcode \$A5.

## **Timing Calculations**

**Step 1:** Determine how many machine cycles are required for each instruction. The number of machine cycles is listed in the Programmer's Guide.

**Step 2:** Determine how many times each of the instructions is executed during program execution. Some instructions, such as those in loops, may be executed multiple times.

**Step 3:** Multiply the number of times each instruction is executed by the number of machine cycles for each instruction and add all individual results together to get the total cycle count.

**Step 4:** Determine the time required per machine cycle. This depends on the oscillator frequency and the architecture of the processor. In the general 8051/C501, each machine cycle takes 12 oscillator cycles (some students initially think it's 6, but it is actually 12). The oscillator cycle time is just the inverse of the oscillator frequency. We are using an 11.0592 MHz oscillator (this particular frequency makes it easy to use standard RS-232 communication speeds); therefore, each oscillator cycle takes  $1/11059200$  seconds, or roughly 90 nanoseconds. Thus, one machine cycle (of 12 oscillator cycles) takes about 1.085 microseconds.

**Step 5:** Multiply this machine cycle time by the total cycle count in Step 3 to get the total execution time. Note that for rough calculations we can quickly estimate how much time it takes to execute simple assembly functions, since each machine cycle takes about 1 microsecond.