```
100 INPUT T%
110 MACHINE 200
120 CODE"INC":CODEL(T%):CODE"BNE,@130,INC":CODEL(T%+1)
130 CODE"RTS"
200 PRINT T%
```

Let's see how it works. On line 120, INC adds one to the number at the memory location specified by the following two bytes. CODEL generates the two byte address of the lower order 8 bits of T%. Thus, the lower order part of T% is increased by one. If the result is non zero, we are done and BNE causes a branch to line 130. If INC causes a zero, we have a carry and must add one to the high order part of T%. The CODEL(T%+1) generates the two byte code for one plus the address of T% (for the high order part of T%). In another example, the program stores zero in all elements of the array A%:

```
100 DIM A%(5)
110 T%=ADR(A%(0%))
120 MACHINE 200
130 CODE"LDYIM,!11,LDA":CODEL(T%):CODE"STAZ,E0,LDA"
140 CODEL(T%+1):CODE"STAZ,E1,LDAIM,0"
150 CODE"STAIY,E0,DEY,BPL,@150,RTS"
200 END
```

At line 110 we set T% equal to the address of A%(0%), which is the first element of the array. There are six elements in the array and therefore, 12 locations that we must zero. Remember, each integer takes up two bytes of memory. At line 120 we first load the Y register with 11. Note the ! symbol in front of the 11. Normally, the CODE command expects assembly language mnemonics or hex numbers. Any number preceded by !, however, is assumed to be a decimal number.

Next we load the low order part of T% (the low order part of the address of A% (0%)), and then store it in the zero page location E0. Note that any mnemonic ending in a 'Z' refers to a zero page location and needs only one hex byte following it to specify the location. Next we get the high order byte of T% and store this in zero page location E1. Finally on line 140, we load the accumulator with 0.

At line 150, STAIY is a store indirect command. The E0 immediately following it tells us the address where we are to store the accumulator; the address is the value in the Y register plus the number stored in E0 (low order part) and E1. In line 140 we stored T%, and thus the address of A%(0%) in E0 and E1. The Y register starts at 11 and therefore, we will store zero in the address of A%(0%)+11. The loop at 150 stores zero in all 12 bytes of the array A%. Note that an assembly language subroutine may use only the zero page locations from D4 to FF.

WARNING: If you make a mistake in assembly language code, the system might hangup. Even BREAK may not get you out of it. Therefore, be sure to save your program before running it.