HILEVEL Technology, Inc.

Q'n Apps #E7

Q: How can I loop through a short set of code 10,000 times?

Using the Program Mode to repeat sequences of vectors

First, you must employ Program Mode (invoked by selecting "Programmedò in the ñSequencing" box under **Run Setup**). Your objective is to set a counter to ***n***, which is the number of times you wish to repeat a sequence of vectors, and then to execute these test vectors ***n*** times. Our examples use "Hex" as the radix selected for address from the **View** drawer.

Begin by filling the entire program space in the Vectors window with NOOP's so that the vectors will simply execute in a sequential fashion. Pull down the **Vector** drawer and select **Program Fill + NOOP + Fill It**. Did you notice how quickly the HiLevel HRISC processor filled your vector program?

Next go to the vector that is the starting point of the repetitive vectors. Three vectors prior to that, load a count value using the LOAD and LOADX commands (LOAD takes care of the three least significant hex digits, while LOADX accesses the most significant hex digit -- forming a 64K repeat counter since the most significant digits must be three or less). Then for the two last vectors in the loop use the instructions DEC (decrement) and CJMP [ADDRESS] (conditional jump), where ADDRESS is the hex address of the first vector in the loop that follows the LOAD process. Let us illustrate this with an example.

Consider the following vector segment:

```
23A NOOP H 00 1 AXXX 0 0 10 00A1 1
23B NOOP L 11 1 XXXX 0 0 10 BF00 1
23C NOOP H 10 1 AXXX 0 0 10 BF00 1
23D NOOP H 00 0 BXXX 0 0 10 BF00 1
23E NOOP L 00 1 AXXX 0 0 10 BF00 1
23F NOOP L 00 0 AXXX 0 0 10 BF00 1
240 NOOP L 00 1 AXXX 0 0 10 BF00 1
241 NOOP L 00 0 AXXX 0 0 10 BF00 1
242 NOOP L 00 1 AXXX 0 0 10 BF00 1
243 NOOP L 00 0 AXXX 0 0 10 BF00 1
244 NOOP L 00 1 AXXX 0 0 10 BF00 1
245 NOOP L 00 0 AXXX 0 0 10 BF00 1
246 NOOP L 00 1 AXXX 0 0 10 BF00 1
247 NOOP L 00 0 AXXX 0 0 10 BF00 1
248 NOOP L 00 1 AXXX 0 0 10 BF00 1
249 NOOP L 00 0 AXXX 0 0 10 BF00 1
24A NOOP L 00 1 AXXX 0 0 10 BF00 1
24B NOOP L 00 0 AXXX 0 0 10 BF00 1
24C NOOP L 00 1 AXXX 0 0 10 BF00 1
24D NOOP L 00 0 AXXX 0 0 10 BF00 1
24E NOOP H 11 1 AXXX 0 0 10 BF00 1
24F NOOP H 00 0 XXXX 1 1 11 0000 1
```

Vector example for Repeat looping

Here, vectors 23E and 23F are repeated eight times. While this normally may not be worth the expansion effort, it certainly would be if the vectors were repeated a hundred thousand times or more, and the mechanism for doing the expansion is the same. The following shows the end result:

```
23B LOADX 0  L 11 1 XXXX 0 0 10 BF00 1
23C LOAD  8  H 10 1 AXXX 0 0 10 BF00 1
23D NOOP     H 00 0 BXXX 0 0 10 BF00 1
23E DEC      L 00 1 AXXX 0 0 10 BF00 1
23F CJMP 23E L 00 0 AXXX 0 0 10 BF00 1
24E NOOP     H 11 1 AXXX 0 0 10 BF00 1
```

Vector example: eight repeats

The savings here is only 14 vectors. However, if the vectors were repeated 10,000 times, the savings would be 19,998

vectors and the pattern generator program would look like this:

```
23B LOADX 2  L 11 1 XXXX 0 0 10 BF00 1
23C LOAD 710 H 10 1 AXXX 0 0 10 BF00 1
23D NOOP     H 00 0 BXXX 0 0 10 BF00 1
23E DEC      L 00 1 AXXX 0 0 10 BF00 1
23F CJMP 23E L 00 0 AXXX 0 0 10 BF00 1
24E NOOP     H 11 1 AXXX 0 0 10 BF00 1
```

Vector example: 10,000 repeats

If you use the $L command in CaeLink you can compress redundant vectors automatically during the translation process. The CaeLink translator will provide a PRG file along with the translated vector file, containing all of the vector loops needed for execution.

More information about the Programmed Mode can be found in the ETS2k User Manual.

Also See:
Q'nApp #E8: Patgen pattern matching
Q'nApp #E32: Symbolic Addressing Mode
Q'nApp #E33: MASK and UNMSK
Q'nApp #E39: "PRG" file attachment to SET file
Q'nApp #E44: Vector files without path
