Register indexed addressing - lookup tables

Data tables can be used to implement conversion functions that are not easily expressed by an equation or set of logical instructions. For example, to find the truncated integer square root of a 4 bit number, it is easier to look it up in a table than to compute it in assembly language.

4-bit number, A	square root (A)
0	0
1	1
2	1
3	1
4	2
5	2
6	2
7	2
8	2
9	3
a	3
b	3
c	3
d	3
e	3
f	3

We can use the "Define Byte" assembler directive to define the table.

DB 119 [*label:*] DB *expression* [, *expr* ...] Generate a list of byte values.

```
squares:    db  0,1,1,1,2,2,2,2,2,3,3,3,3,3,3,3
```

"squares" is the label of the first address in the table.

Exercises:

Using DPTR to hold base address

1. Write a program that finds the integer square root of the A register. Place the program code at 2000h (don't forget the jump instruction at the reset location) and the data table at 20FFh using the cseg assembler directive. You must assume that

the A register value will be between 0 and 15. Let the program end with a "nop" instruction.
2. Save your file as **table1.asm** and assemble, build, and download the file.
3. View the 8051 registers and code memory. Check locations 20FF-210E to make sure your table is there.
4. Click on the ACC register value in the debug window and set the ACC to 05. (Wow, did you realize you could do that?!)
5. Step through the program and see if you end up with "2" in the ACC when you are done.

Additional Challenge: (if you finish early) Using PC to hold base address

1. Save your program under a new name: **table2.asm**. Note that in this case the table will not be in the location 20FF as it will not be possible to reach it there.
2. Modify the table2.asm program so that you use PC as the base address for the table. Let the program end with a "nop" instruction:
   nop ; takes up one byte in program memory

3. Here is the example from the lecture that might help. You may need an "increment A" instruction: `inc a`

```
Addr     cseg at 0x1000h
1000     mov a, #5
1002     movc a, @a + PC ;a <-- M[1008]
1003     nop
```

After assembling, building and downloading the program, set the A register in the debug window to be 9. Step through the program and see if you end up with a 3 in the A register. If not, try answering the following questions to debug your program:

What is the value of PC when you get to the nop instruction?

Where does your table start in memory?

What do you have to add to the base address of the table to access the correct memory location?