Tutorial 04: Using the ARM Simulator

Computer Science Department
CS2208b: Introduction to Computer Organization and Architecture
Winter 2019
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Keil Simulator

- The processor in the PC is not a member of the ARM family.
- It is usually a member of Intel's IA family or an AMD processor.
- However, you can run ARM code on your PC using a simulator program, e.g., Keil.

- The Keil package, called µVision, is very sophisticated and is intended for engineers designing embedded ARM-based systems.
- Consequently, it includes far more facilities than we need.

- The *demonstration version* that you can download for a PC (*about 600 MB*) is limited to assembly-level programs smaller than 32K bytes.
- This restriction will not be a problem at all for students.

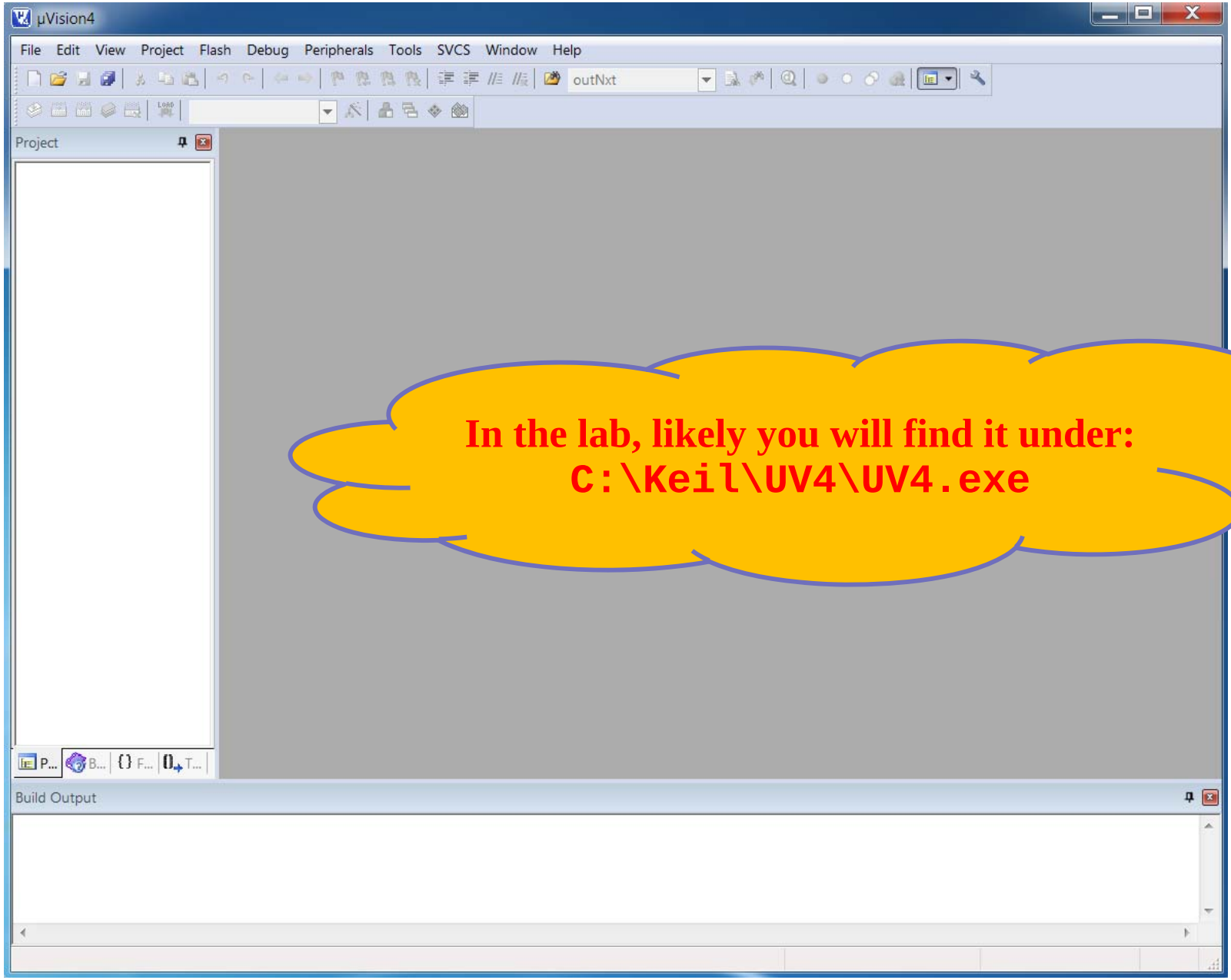
µVision

- Essentially, µVision is an IDE (*integrated development system*) that is *project-based*;
  - Each new program must be part of a project.
    - You begin by creating a project (i.e., a container for all your files) and then
    - You select the target processor you are going to use.
  - You create source files (in our case, these will be assembly language files) and then
  - You build your application (i.e., create code for your chosen processor).

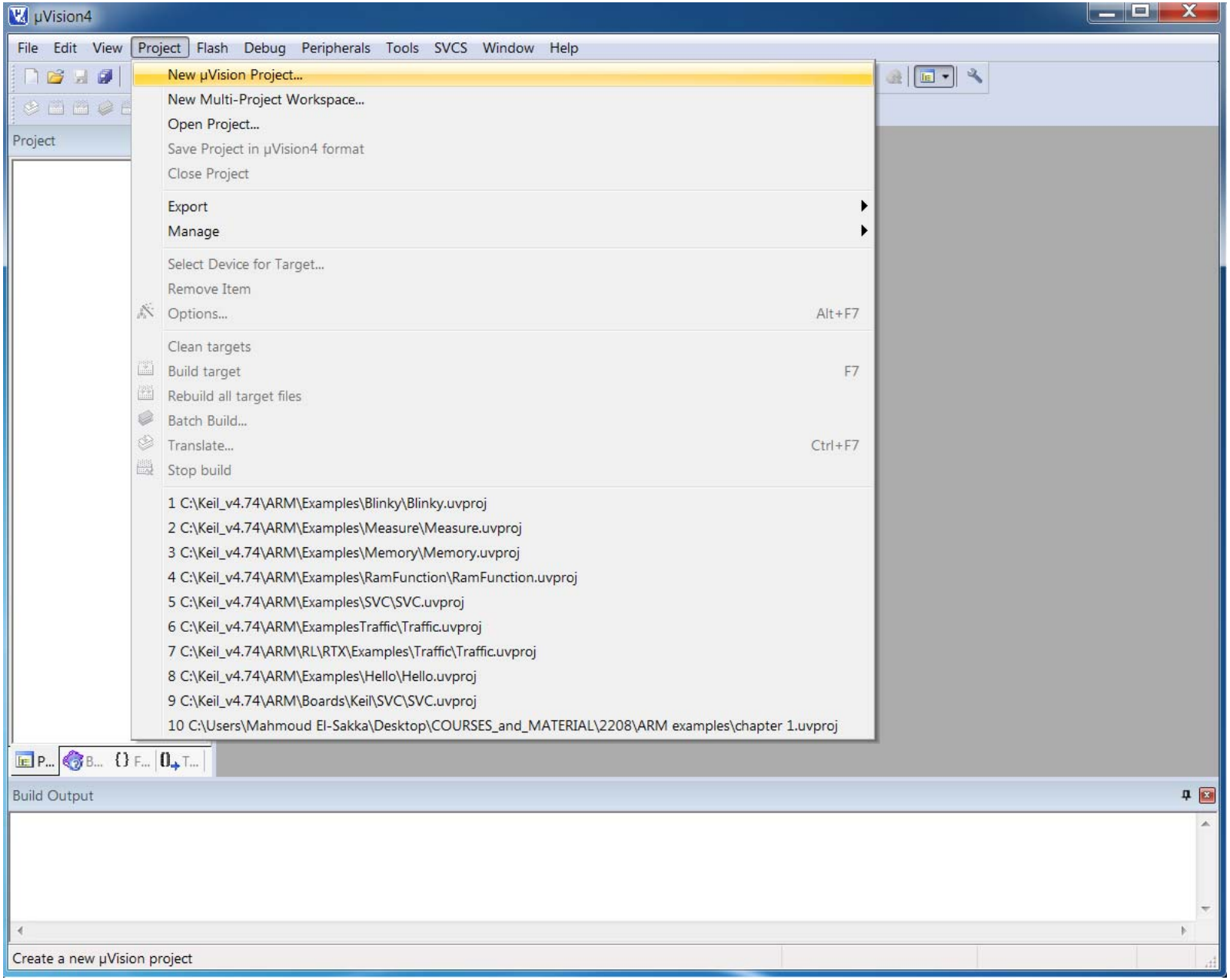
Downloading the Simulator

- The current version of the µVision software is 5.
- However, all illustrations in the book and lectures are based on µVision version 4.
- As there is a big difference between the interface of the two versions, we will stick during this term with version 4.
  - Can be downloaded from *OWL*
    - *It is the Windows Edition*
    - *There is NO Mac edition*

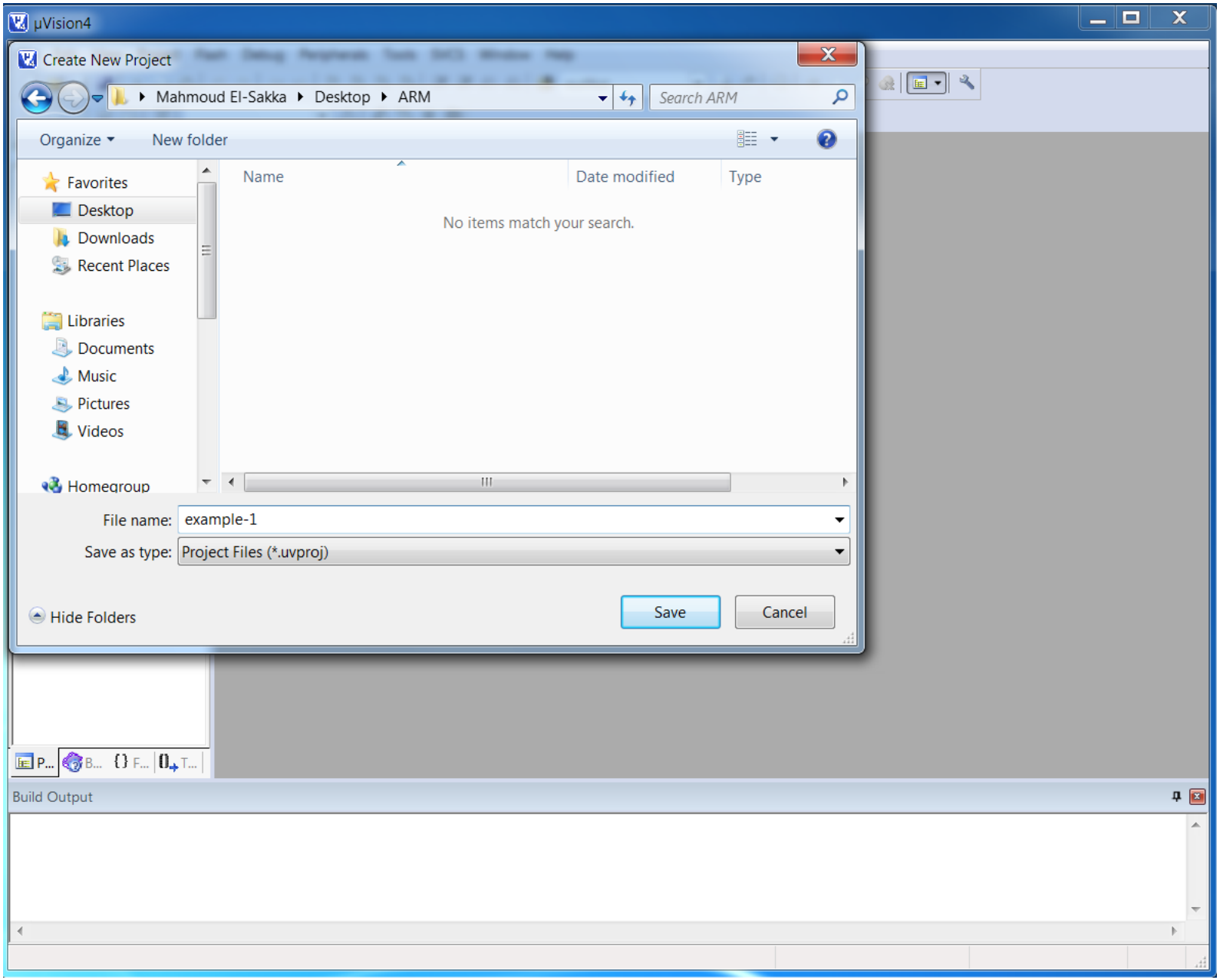
Using the Simulator—*step-by-step*



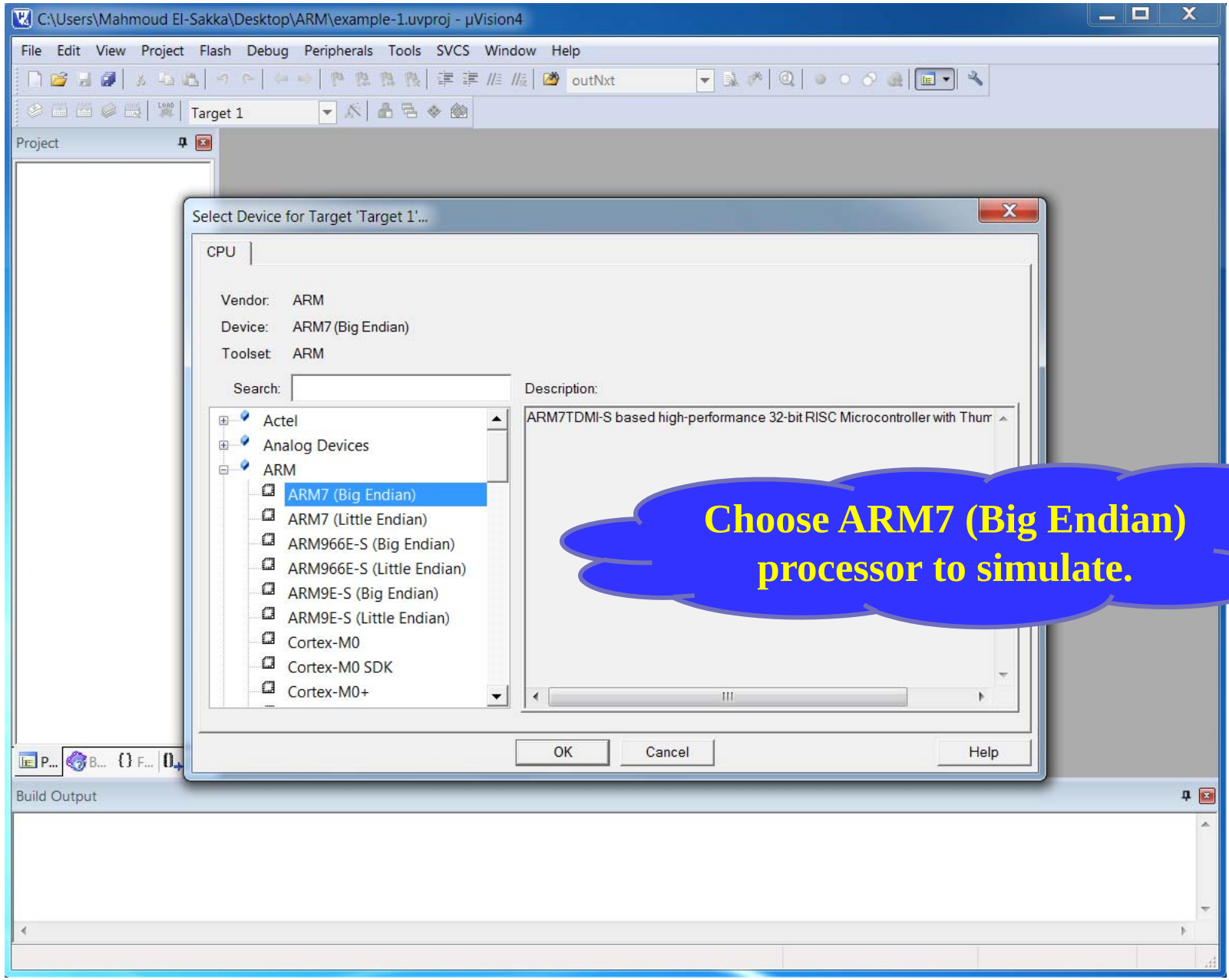
Using the Simulator—*step-by-step*



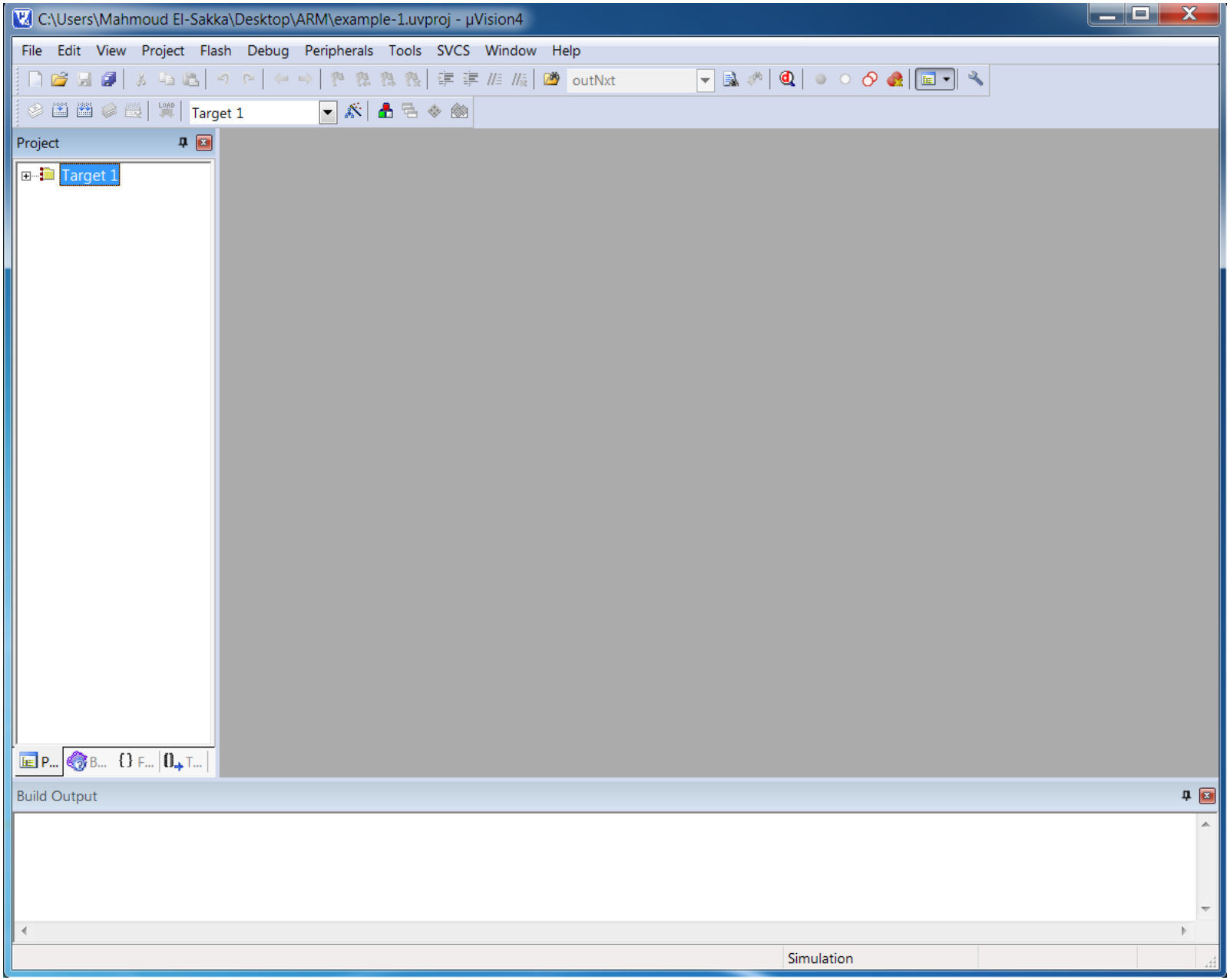
Using the Simulator—*step-by-step*



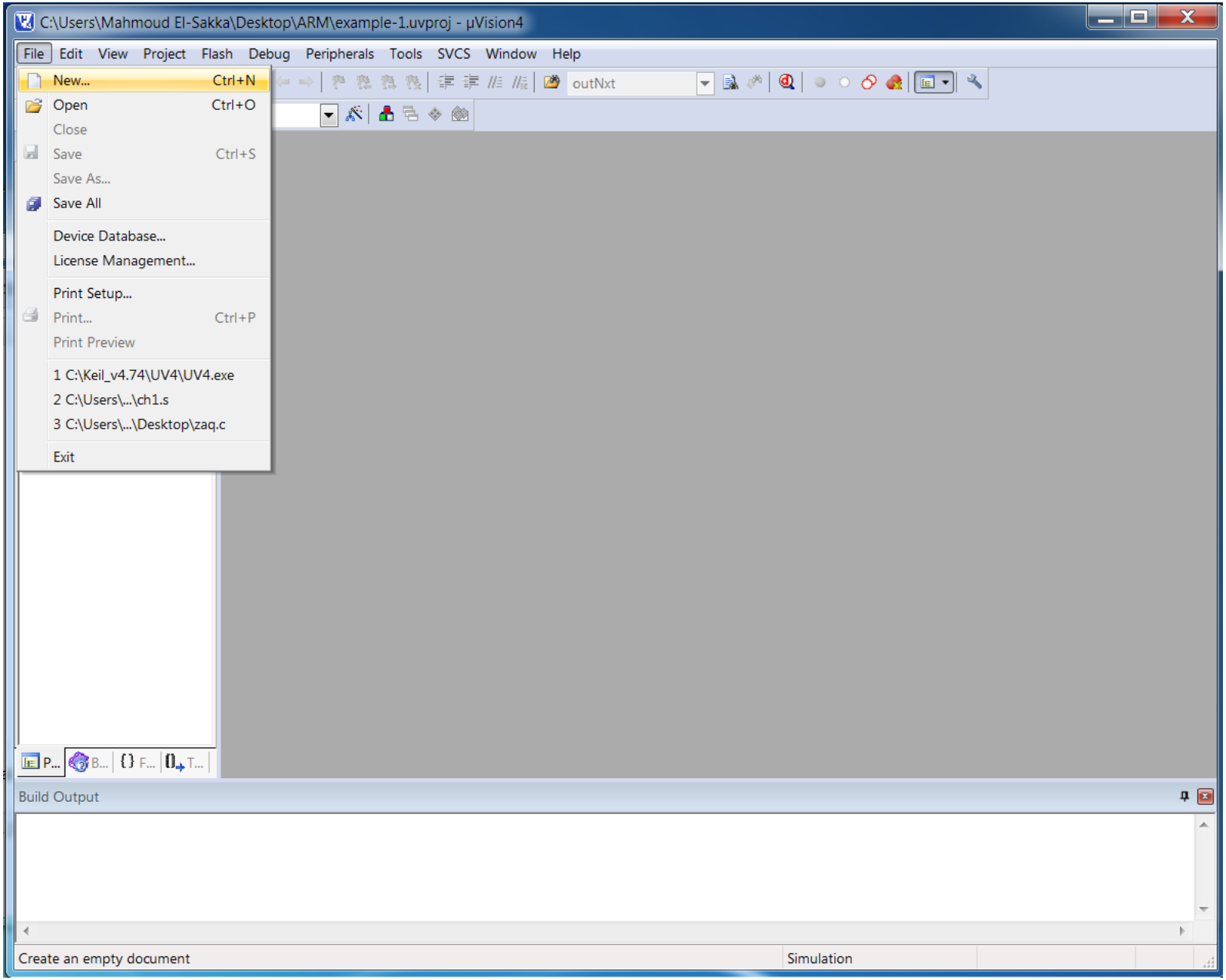
Using the Simulator—*step-by-step*



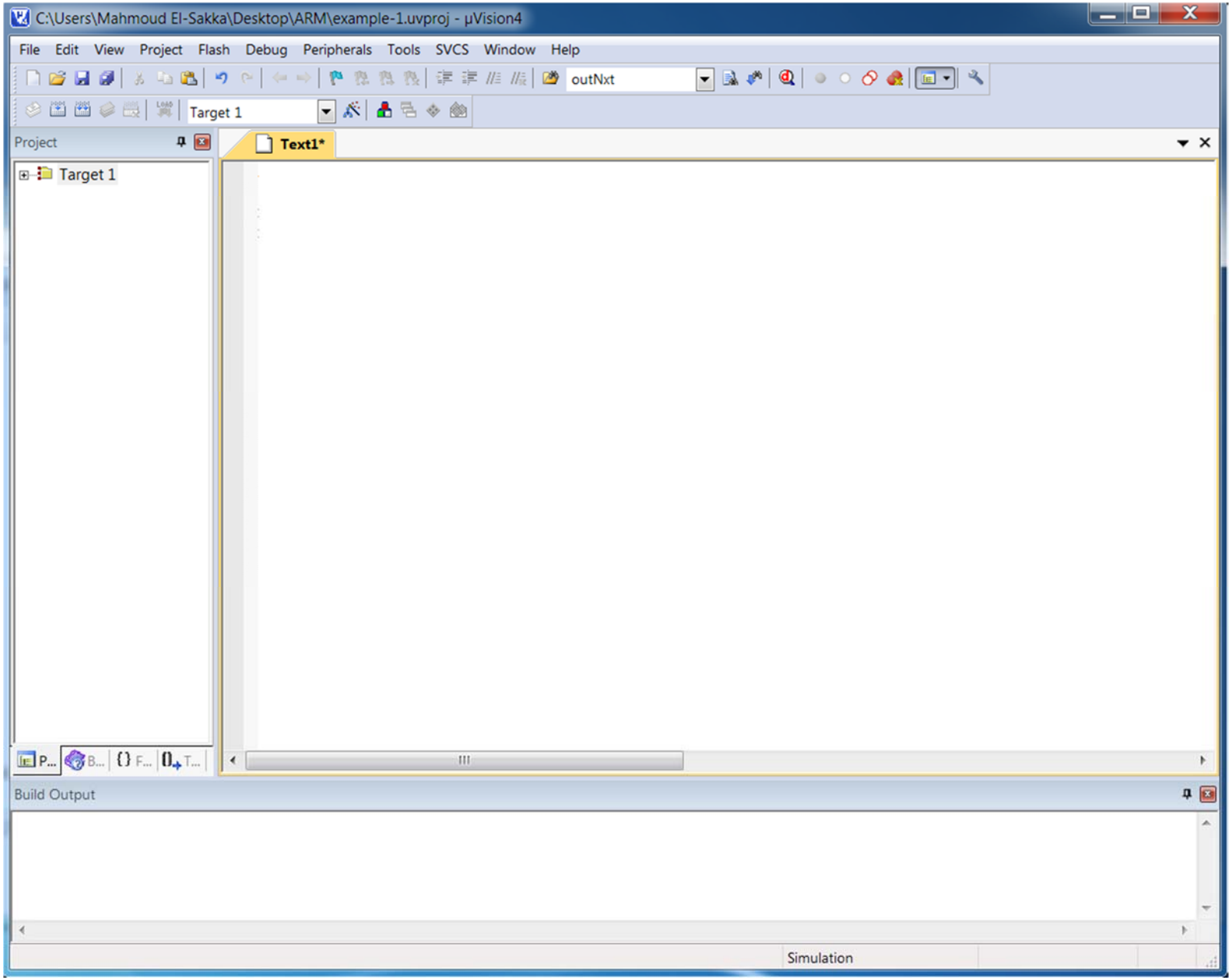
Using the Simulator—*step-by-step*



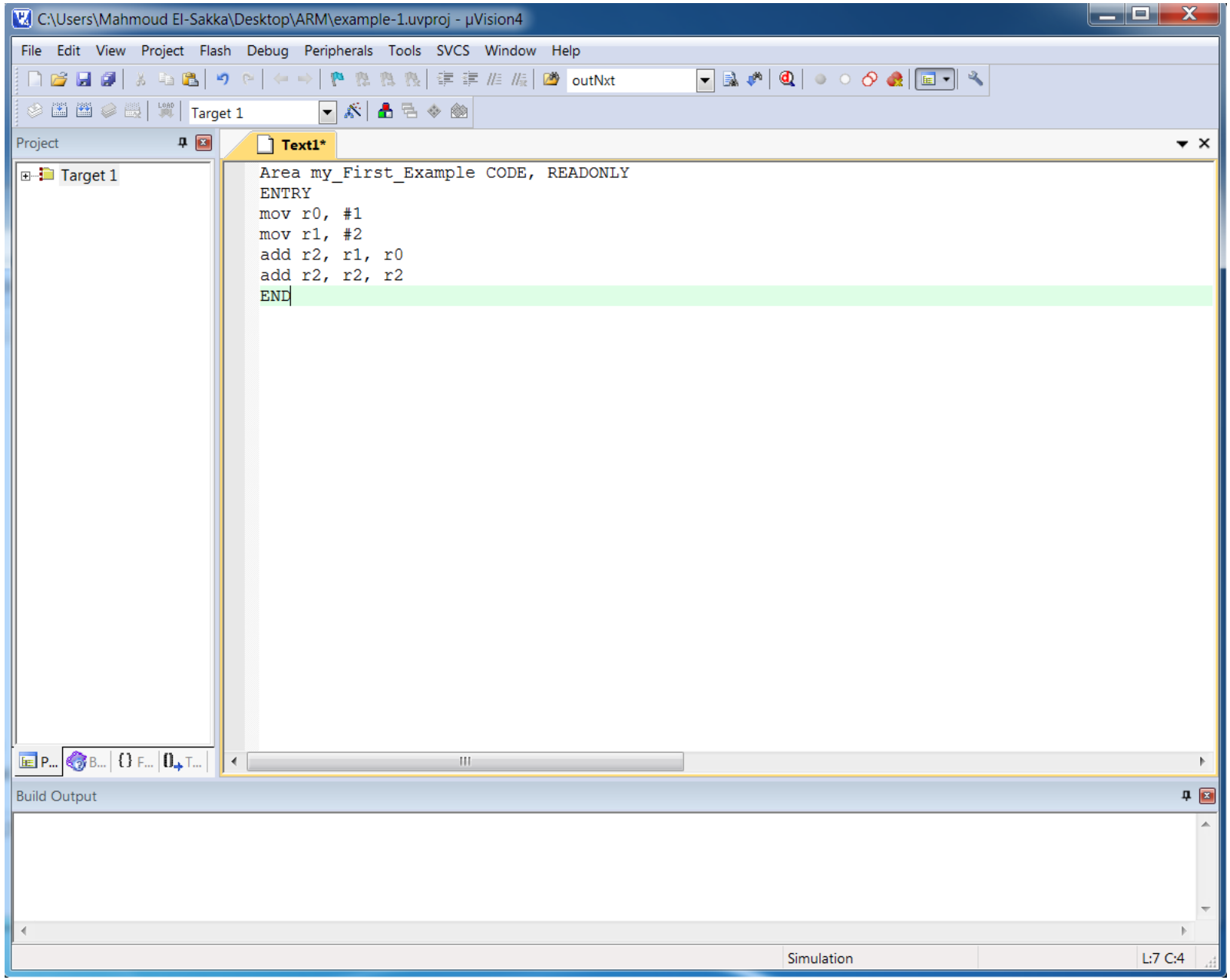
Using the Simulator—*step-by-step*



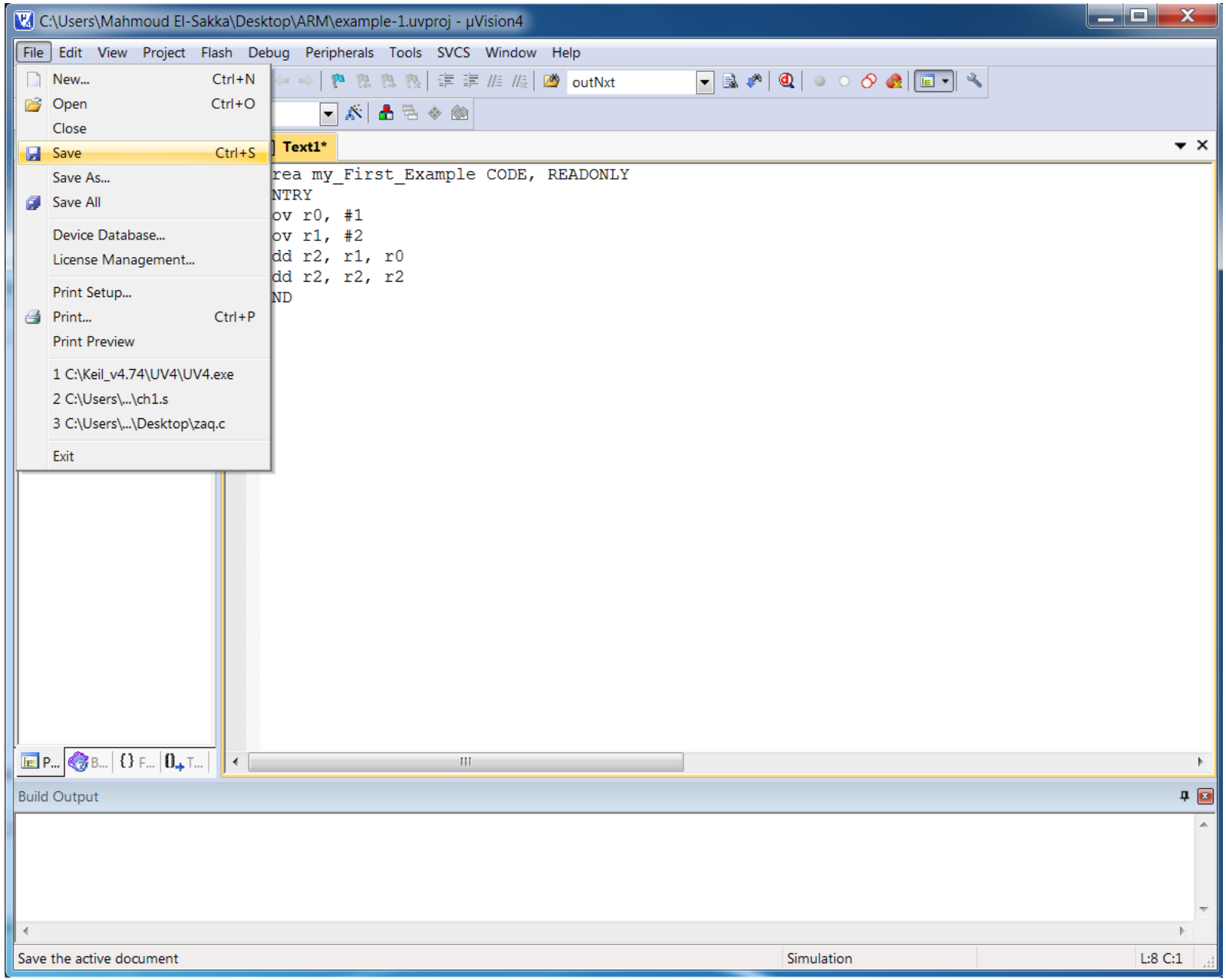
Using the Simulator—*step-by-step*



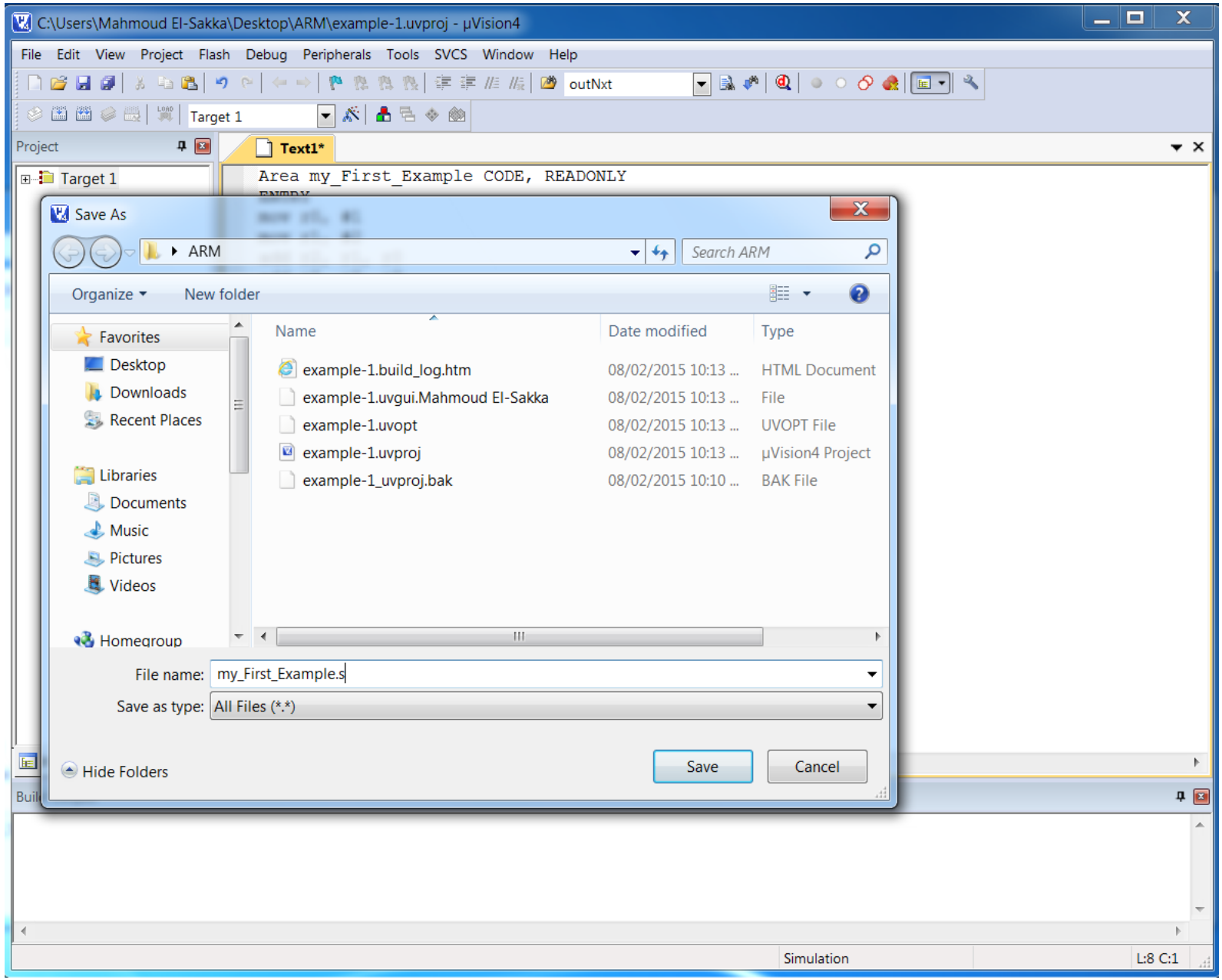
Using the Simulator—*step-by-step*



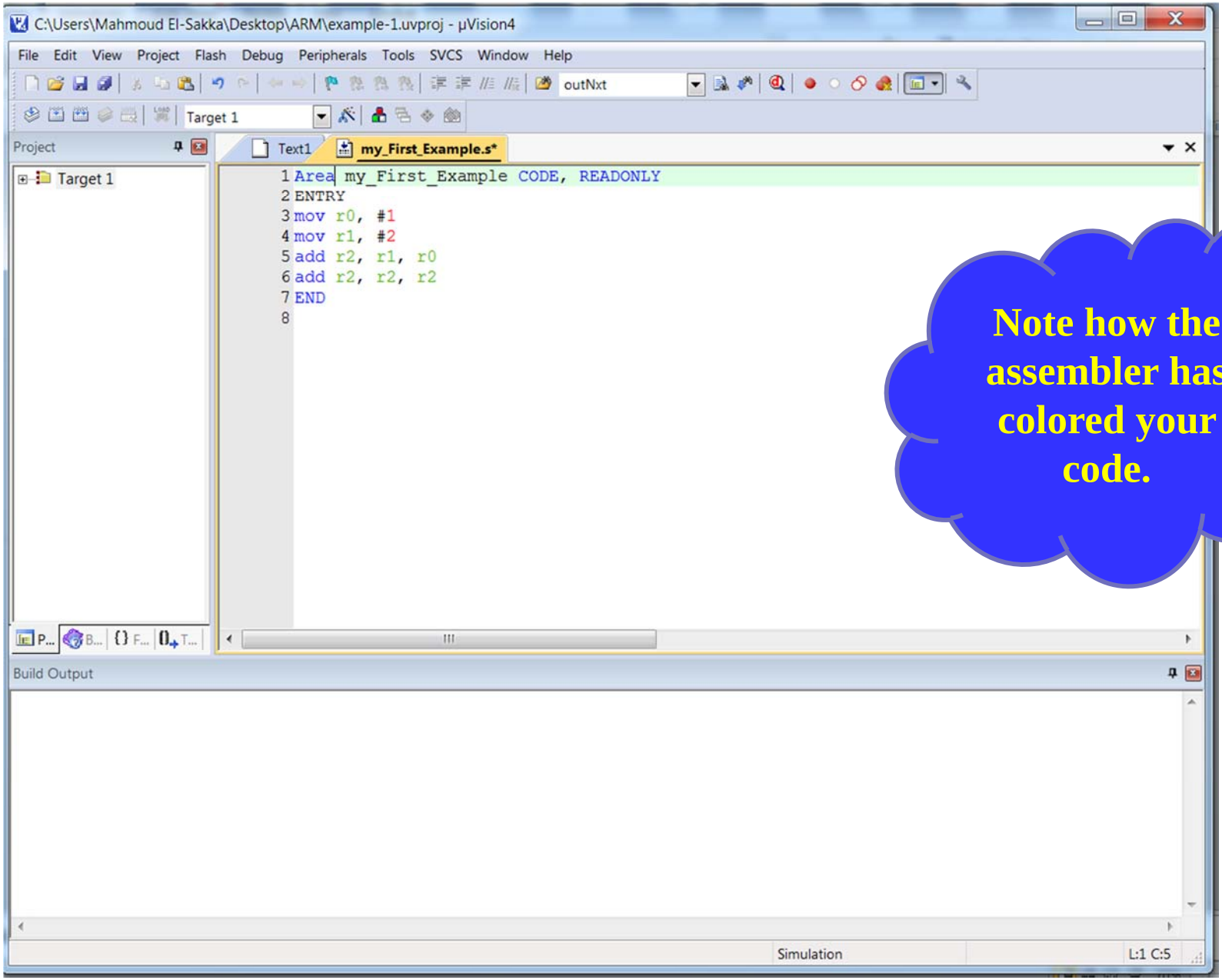
Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*

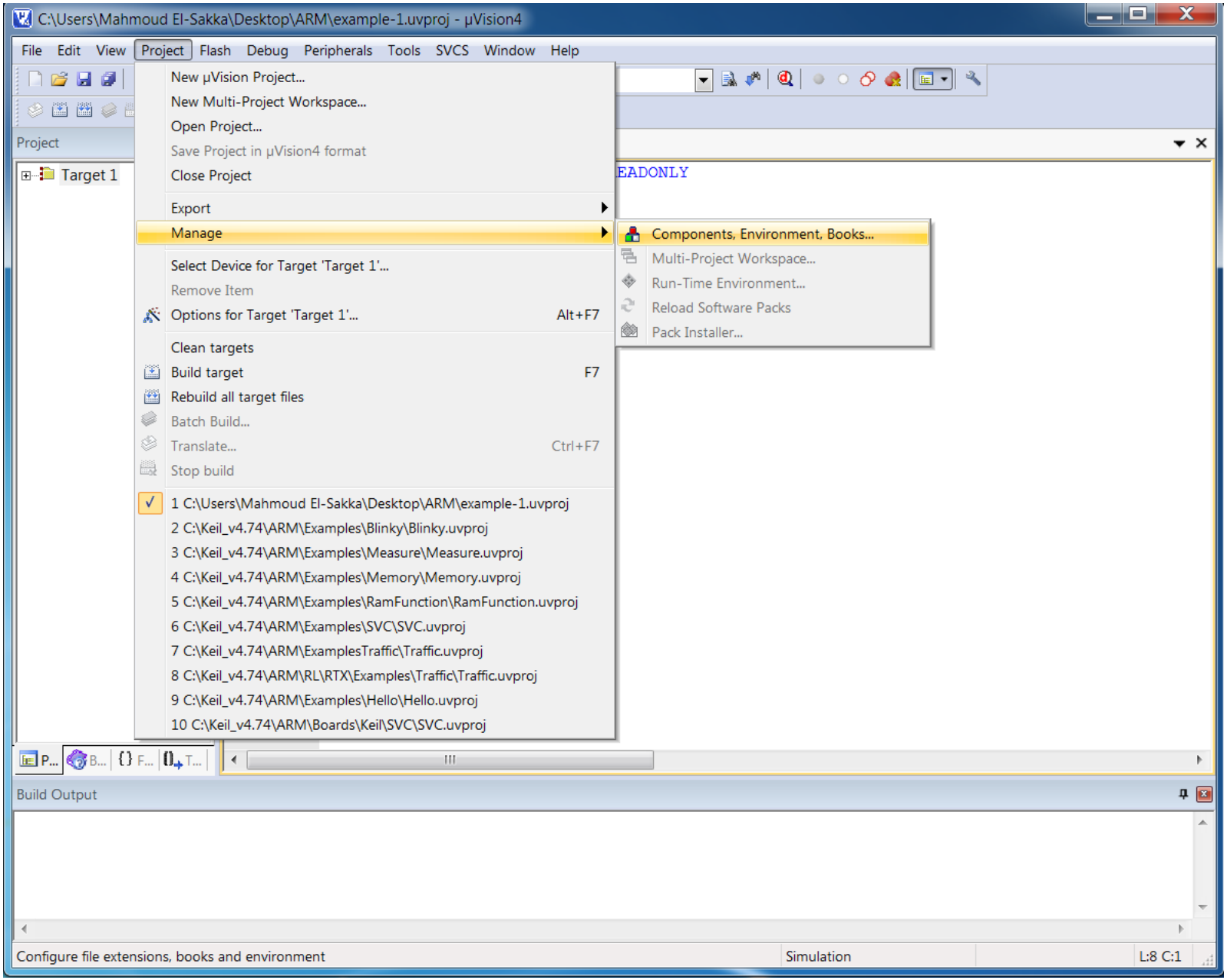


Using the Simulator—*step-by-step*

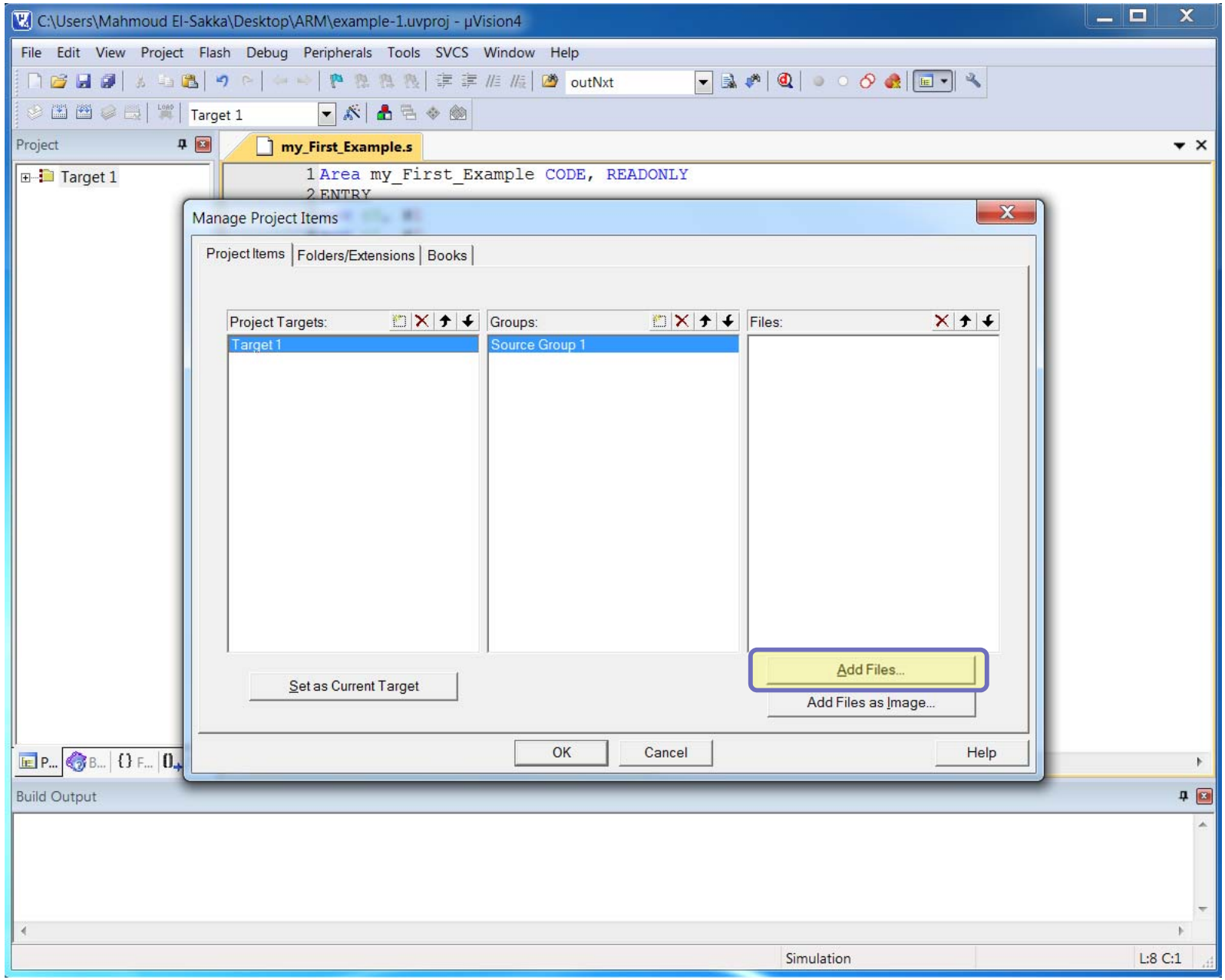


Note how the assembler has colored your code.

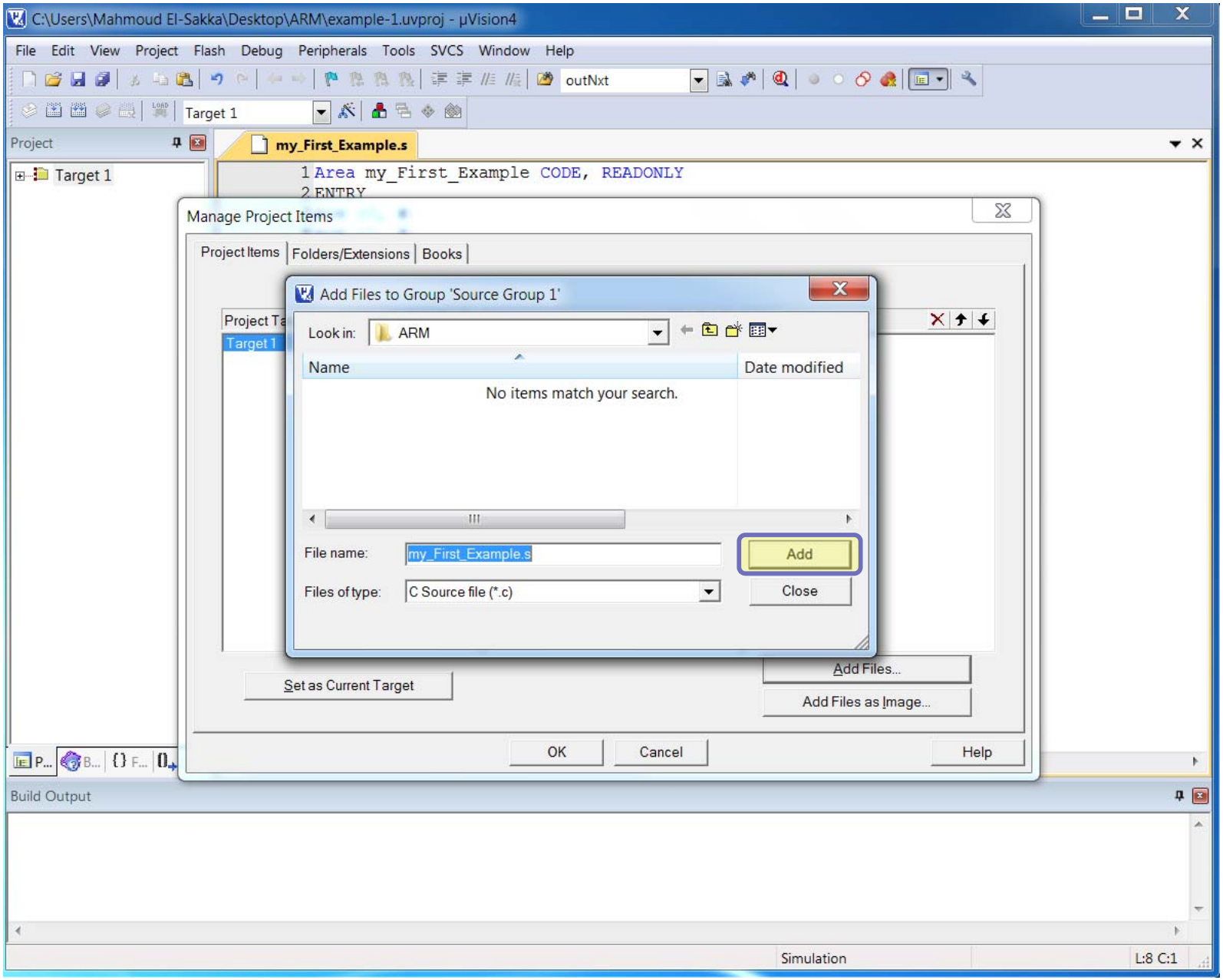
Using the Simulator—*step-by-step*



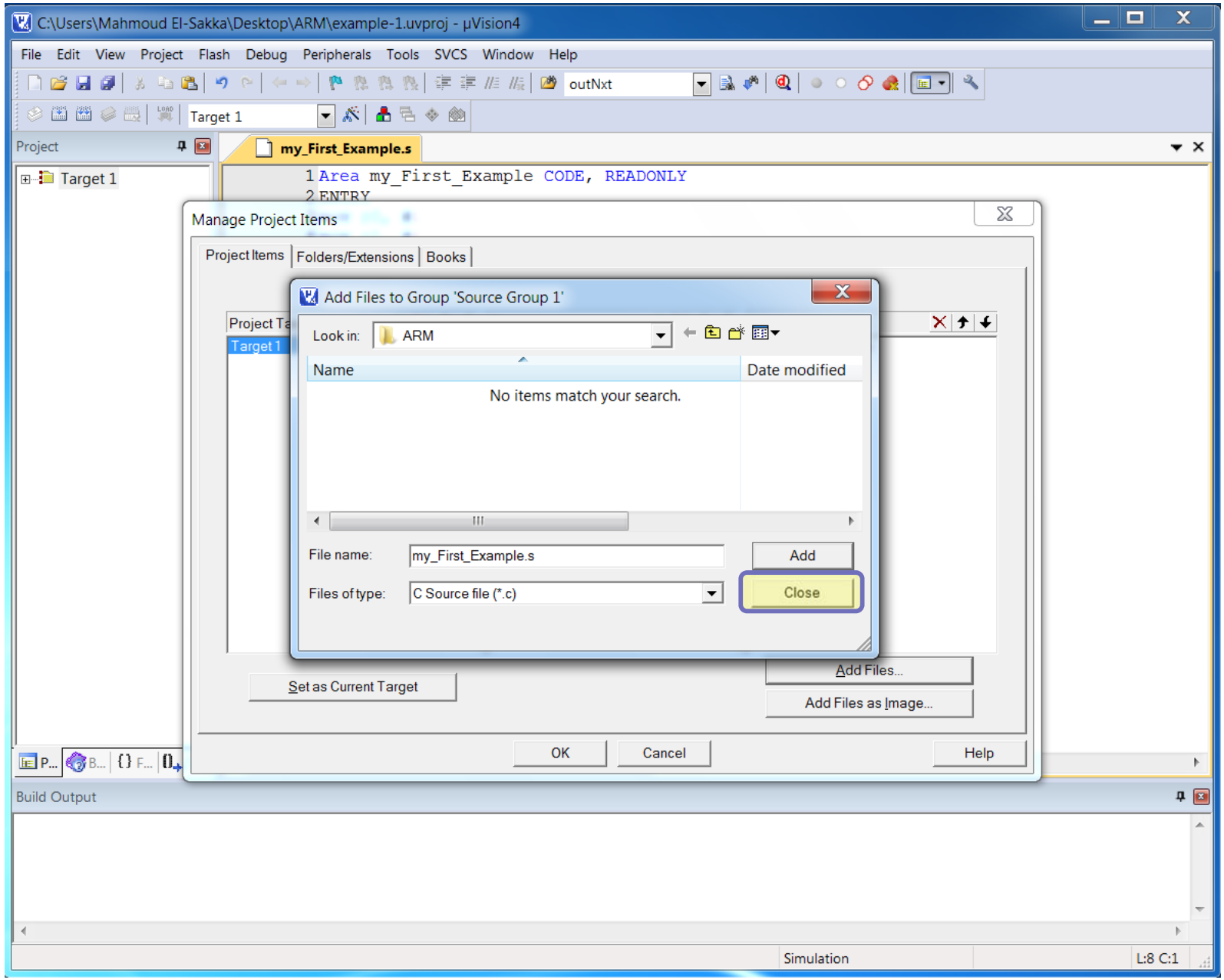
Using the Simulator—*step-by-step*



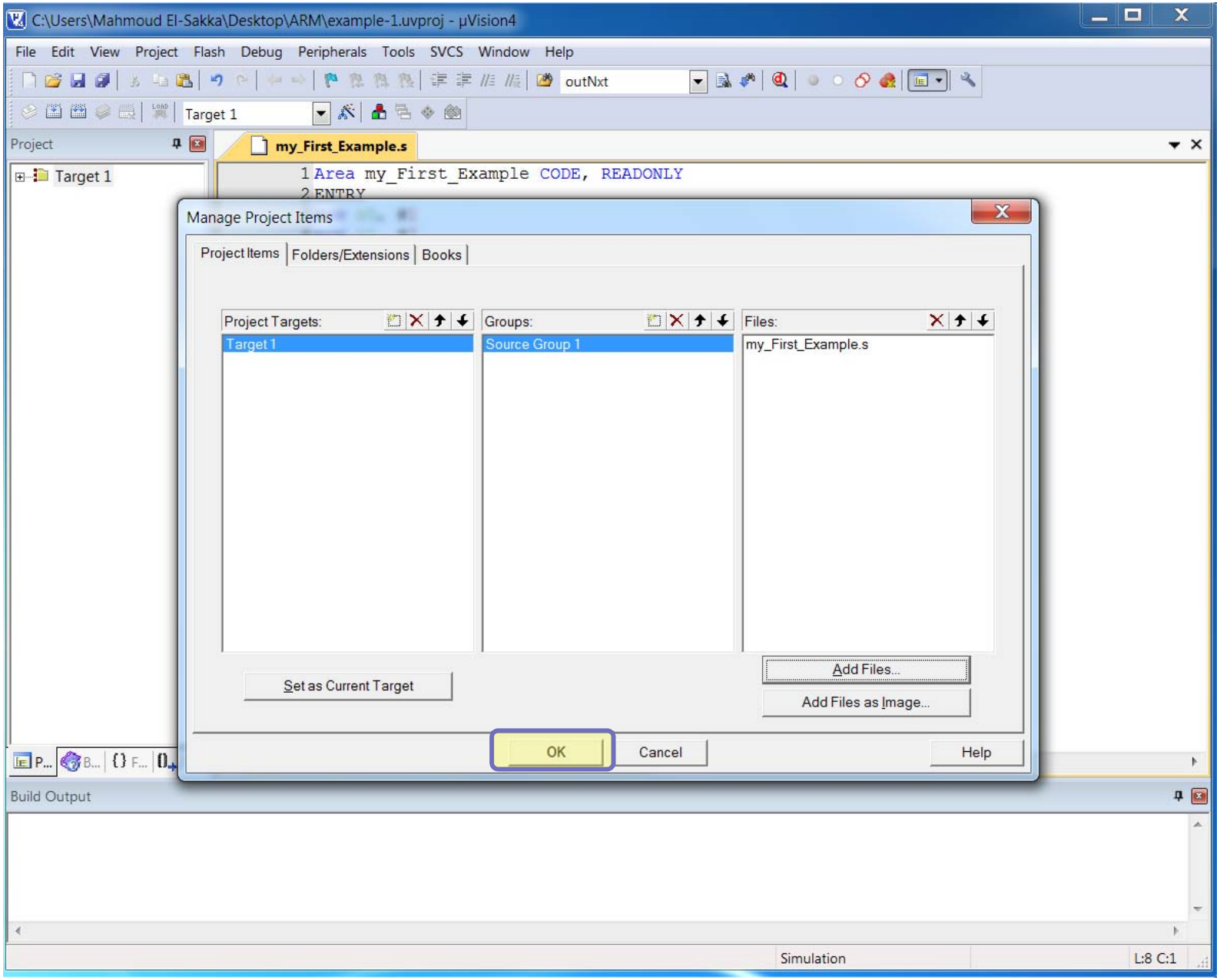
Using the Simulator—*step-by-step*



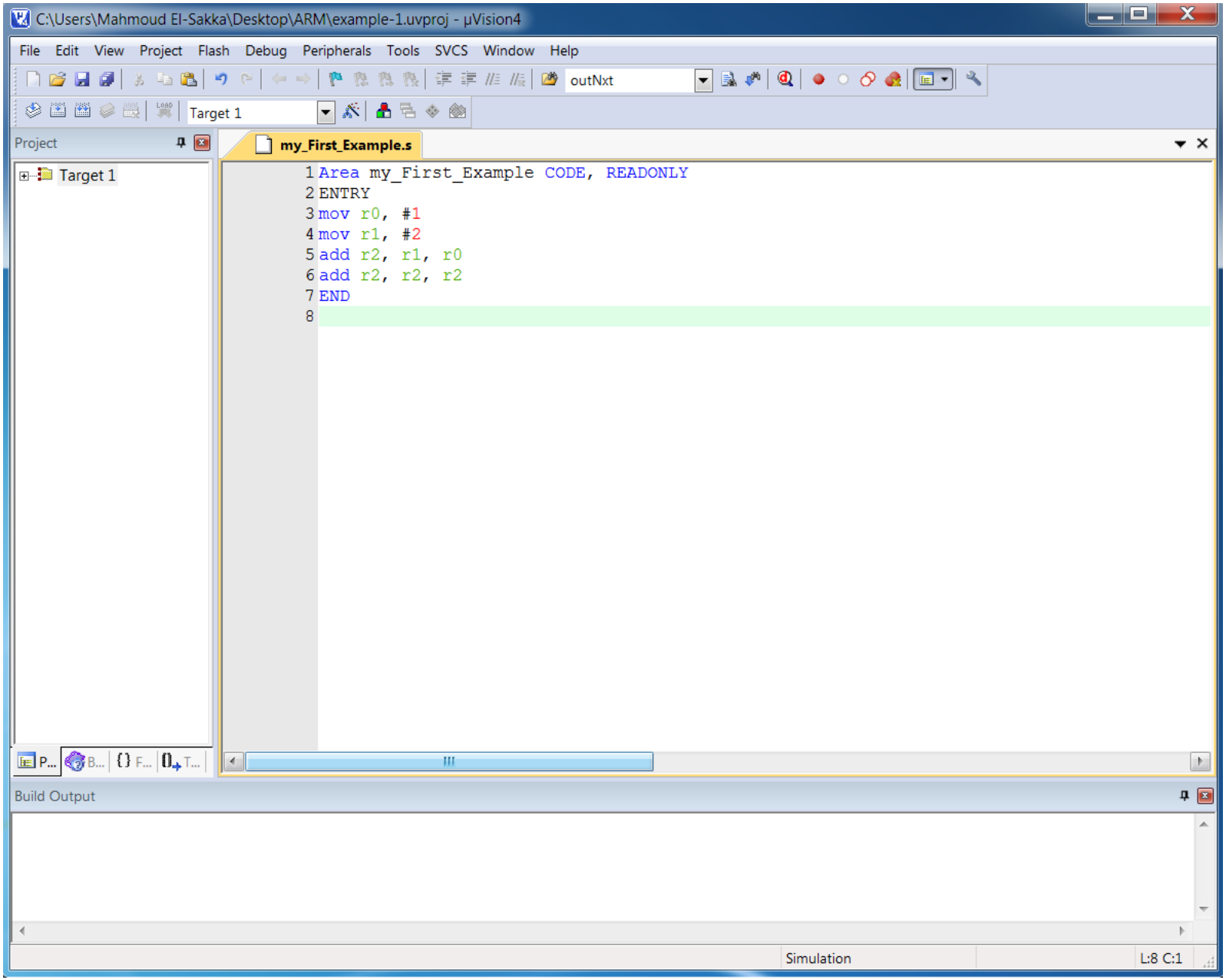
Using the Simulator—*step-by-step*



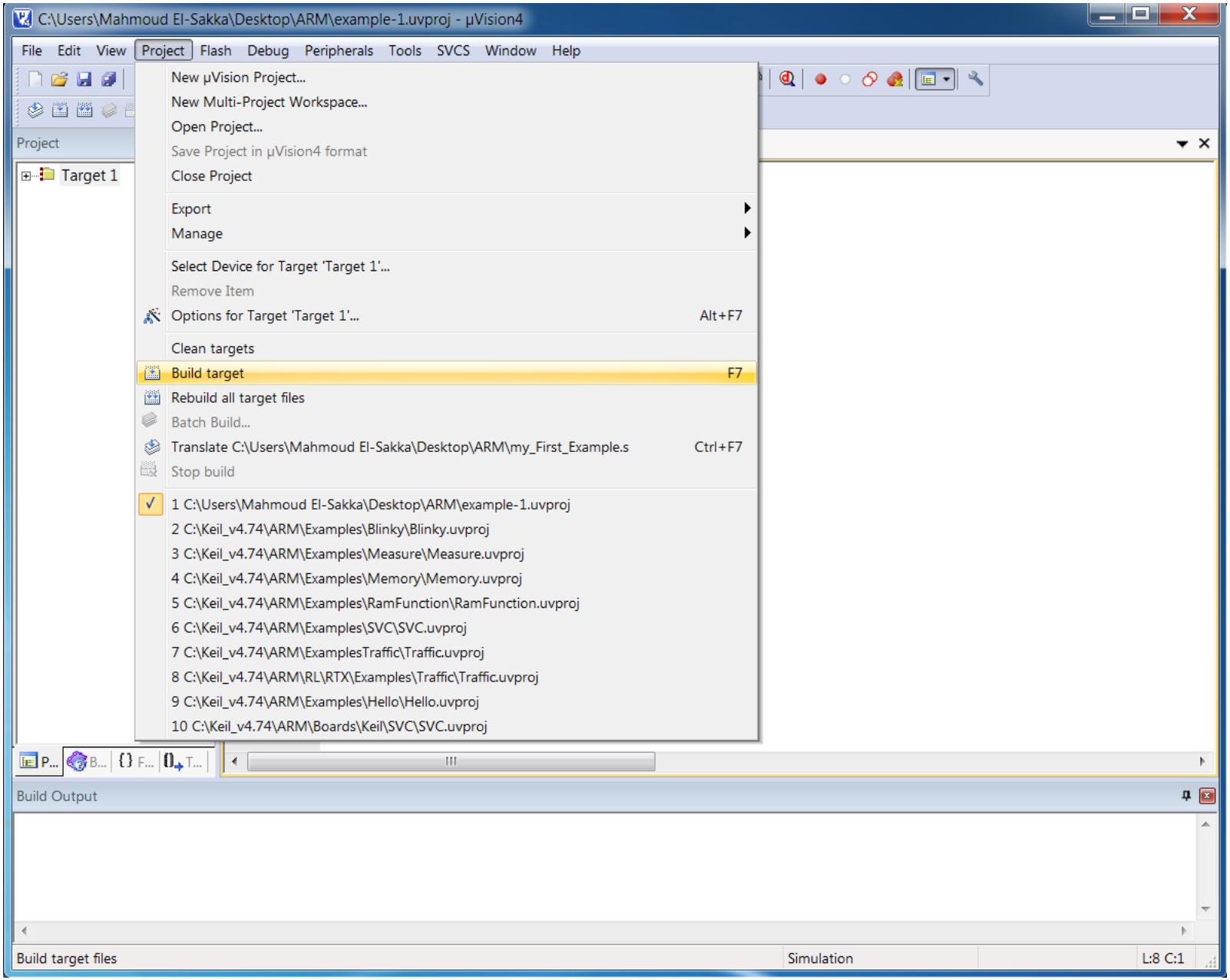
Using the Simulator—*step-by-step*



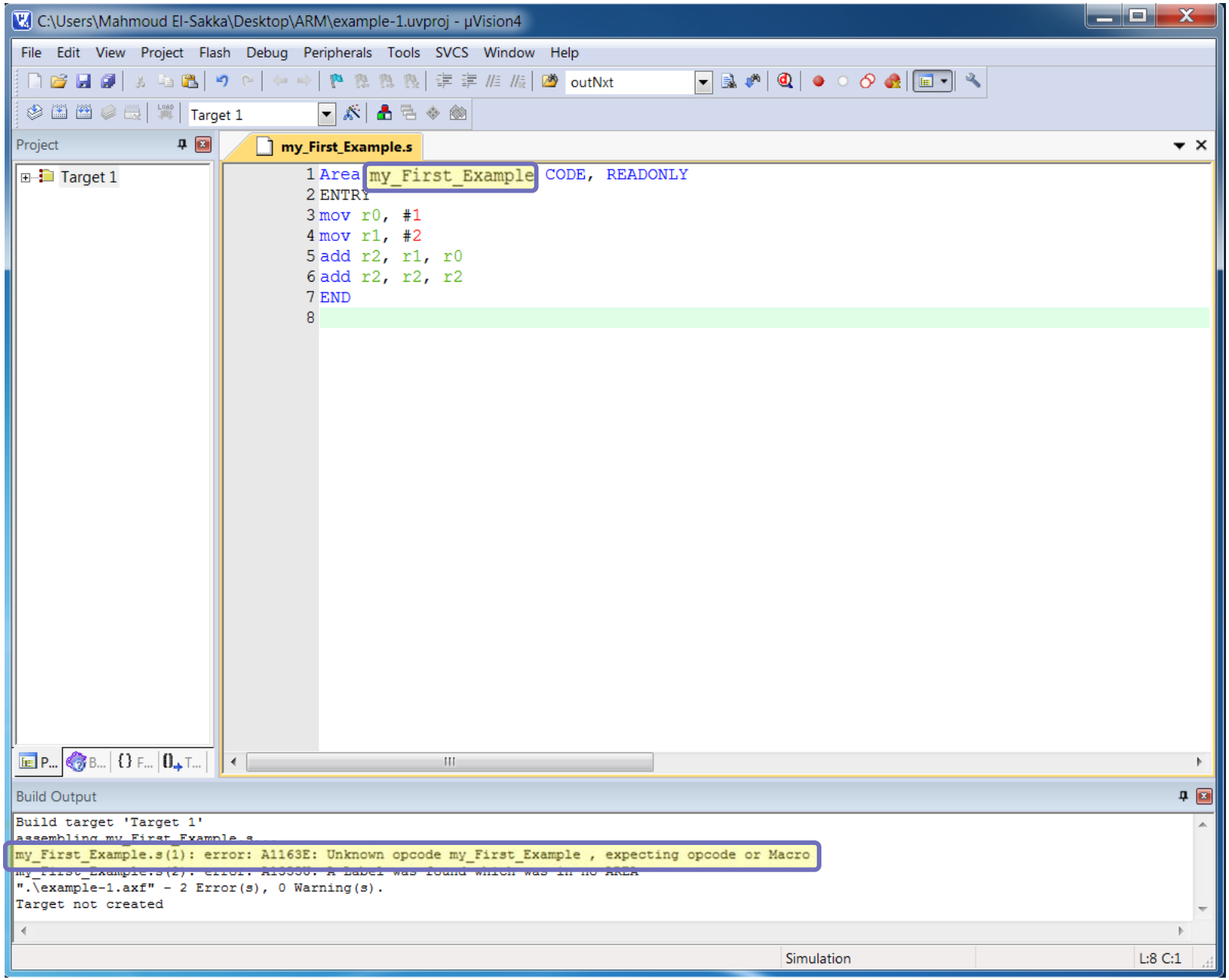
Using the Simulator—*step-by-step*



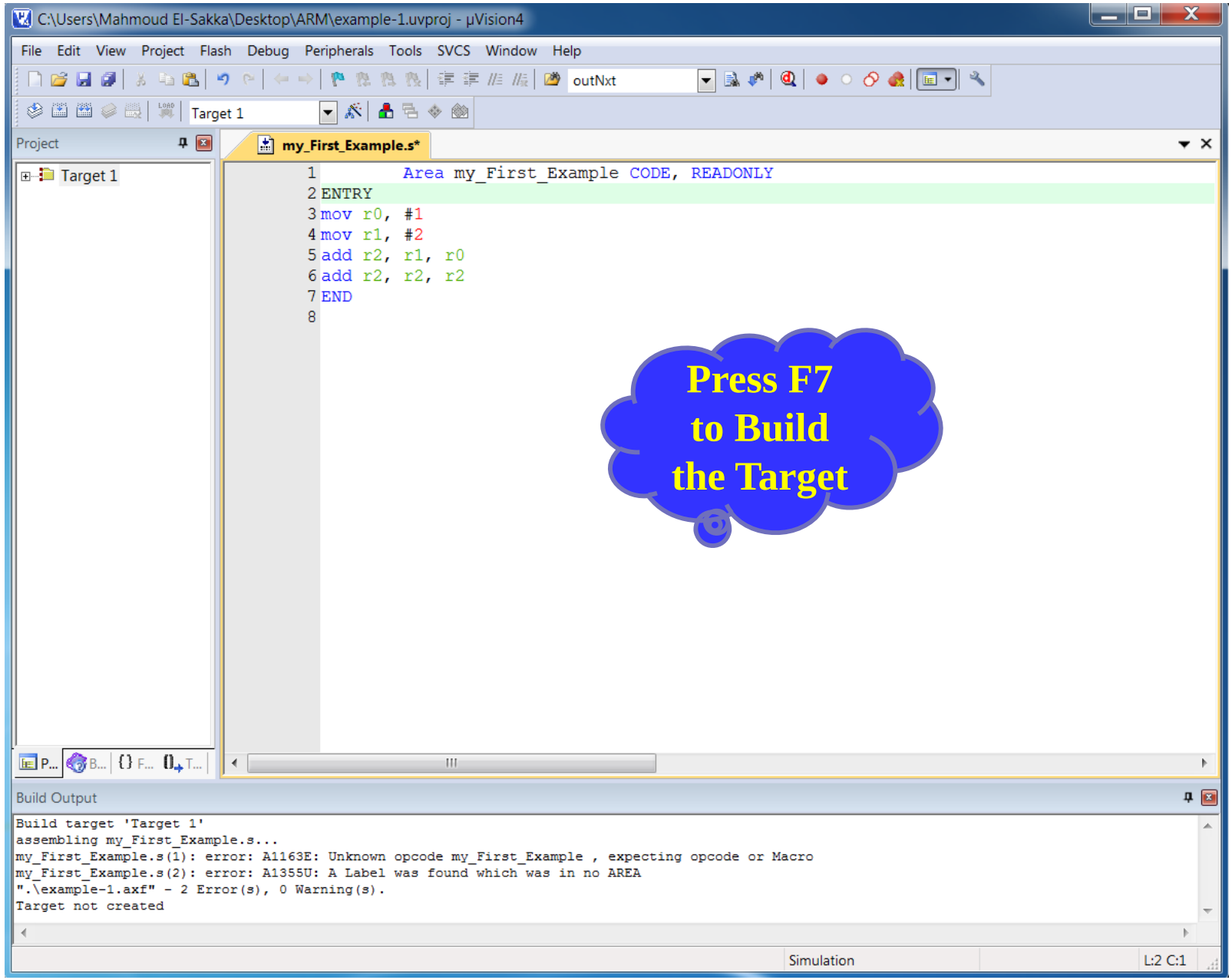
Using the Simulator—*step-by-step*



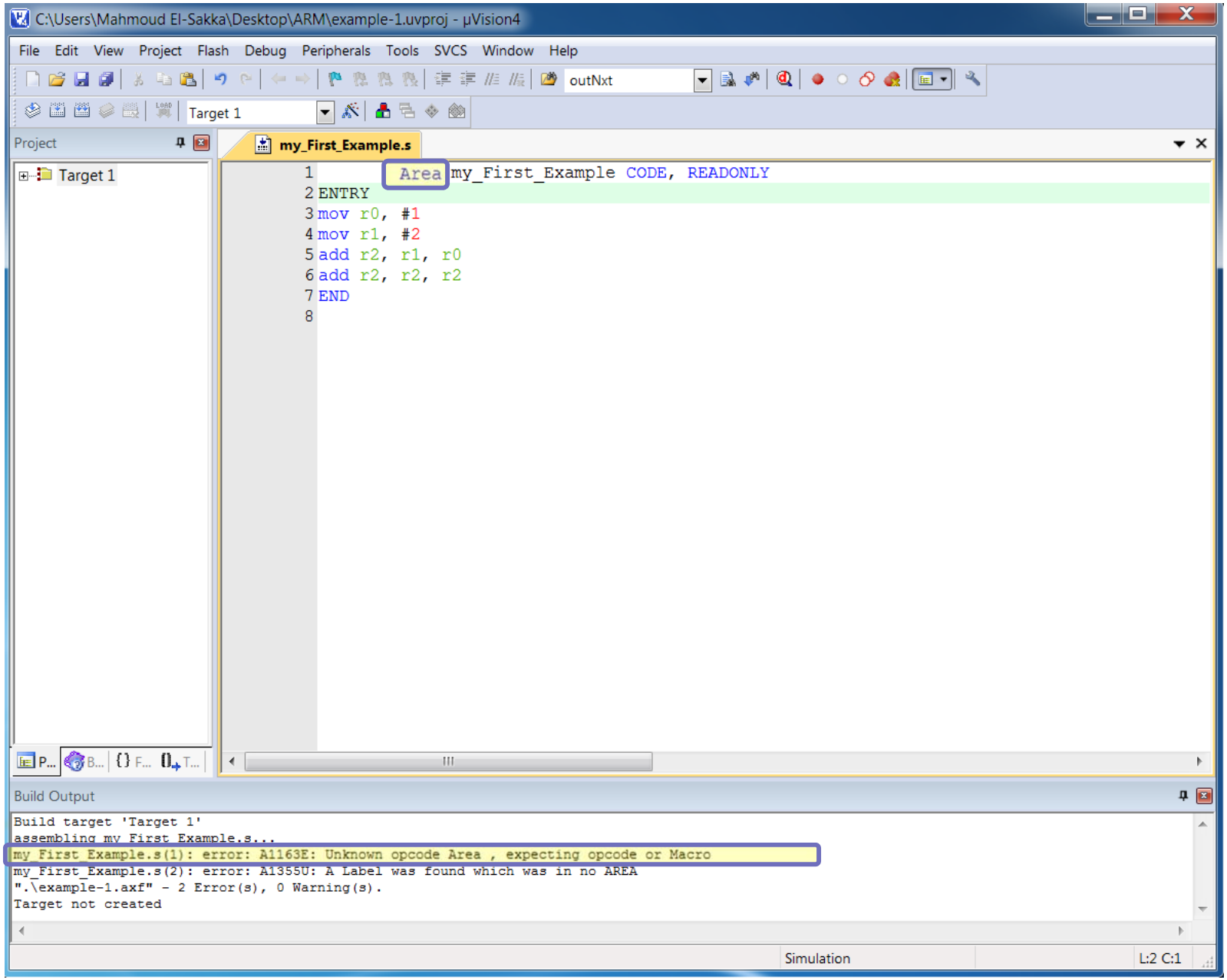
Using the Simulator—*step-by-step*



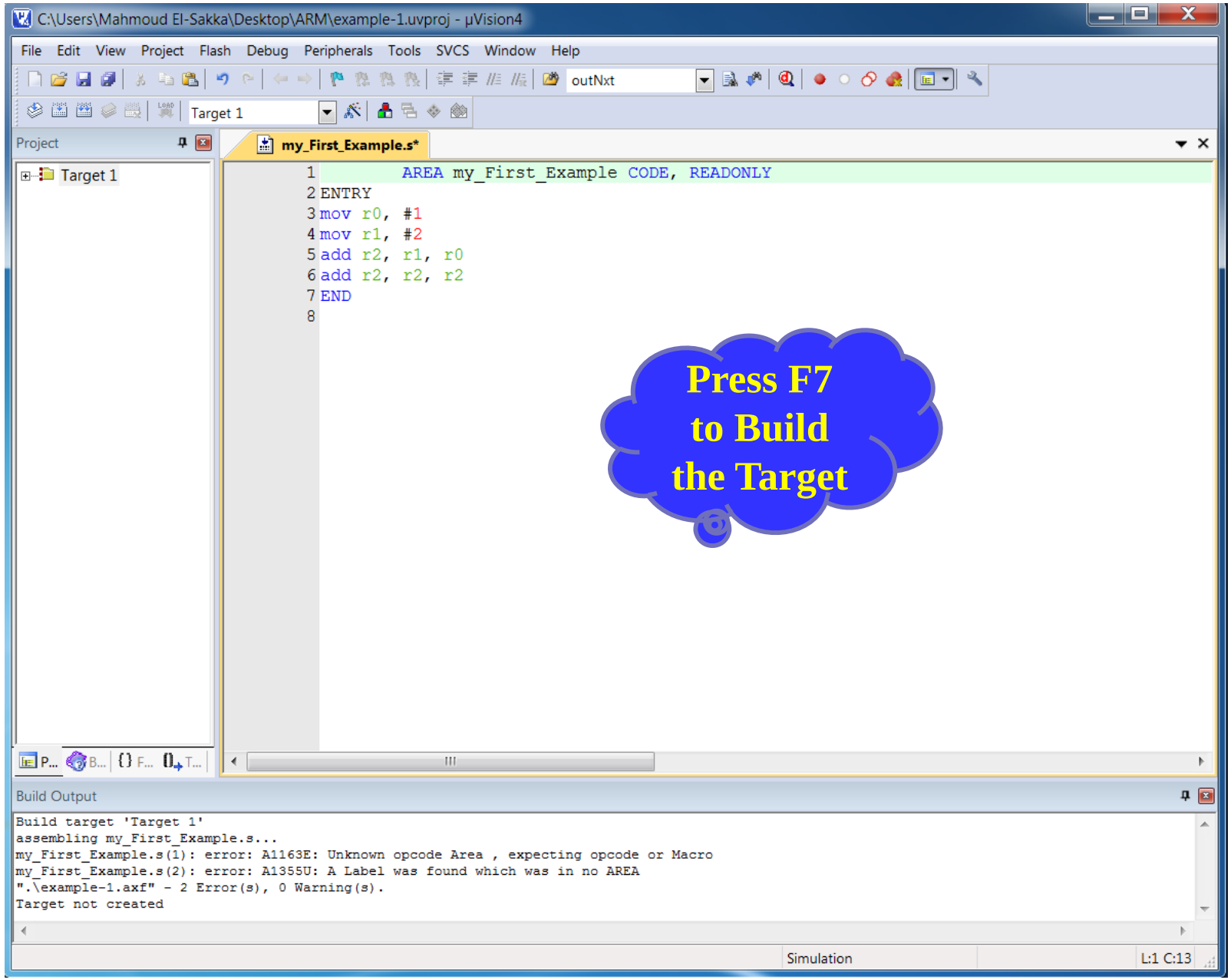
Using the Simulator—*step-by-step*



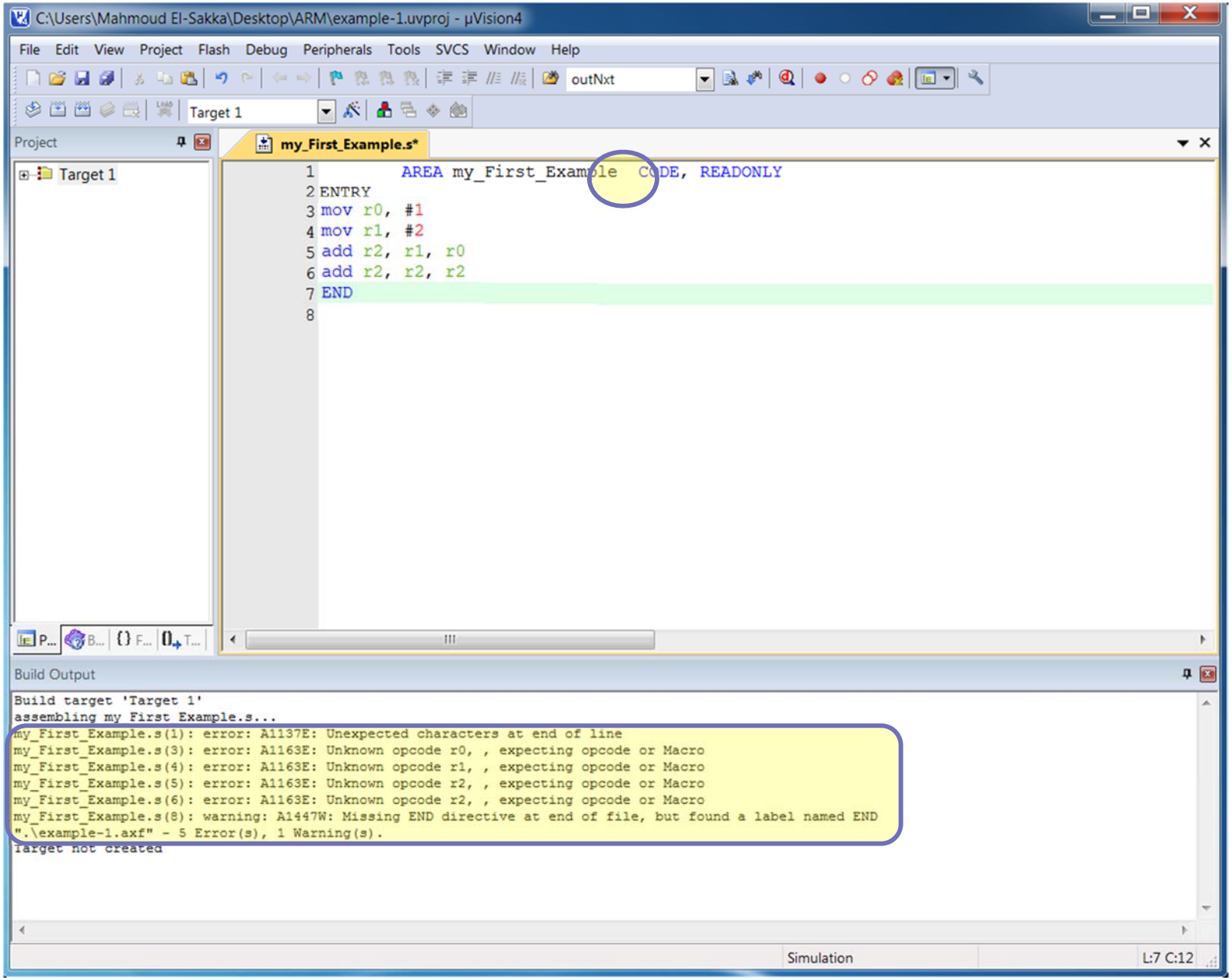
Using the Simulator—*step-by-step*



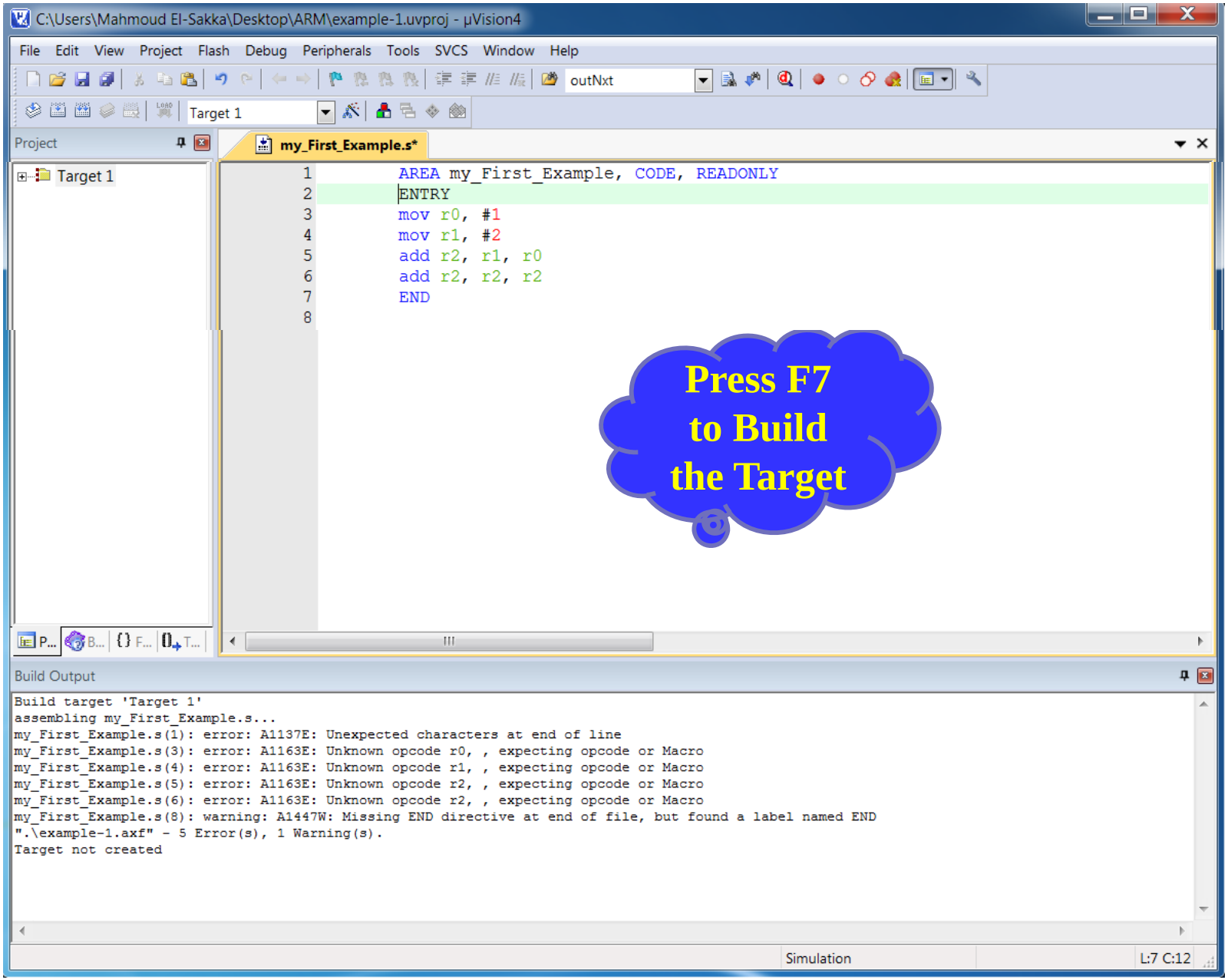
Using the Simulator—*step-by-step*



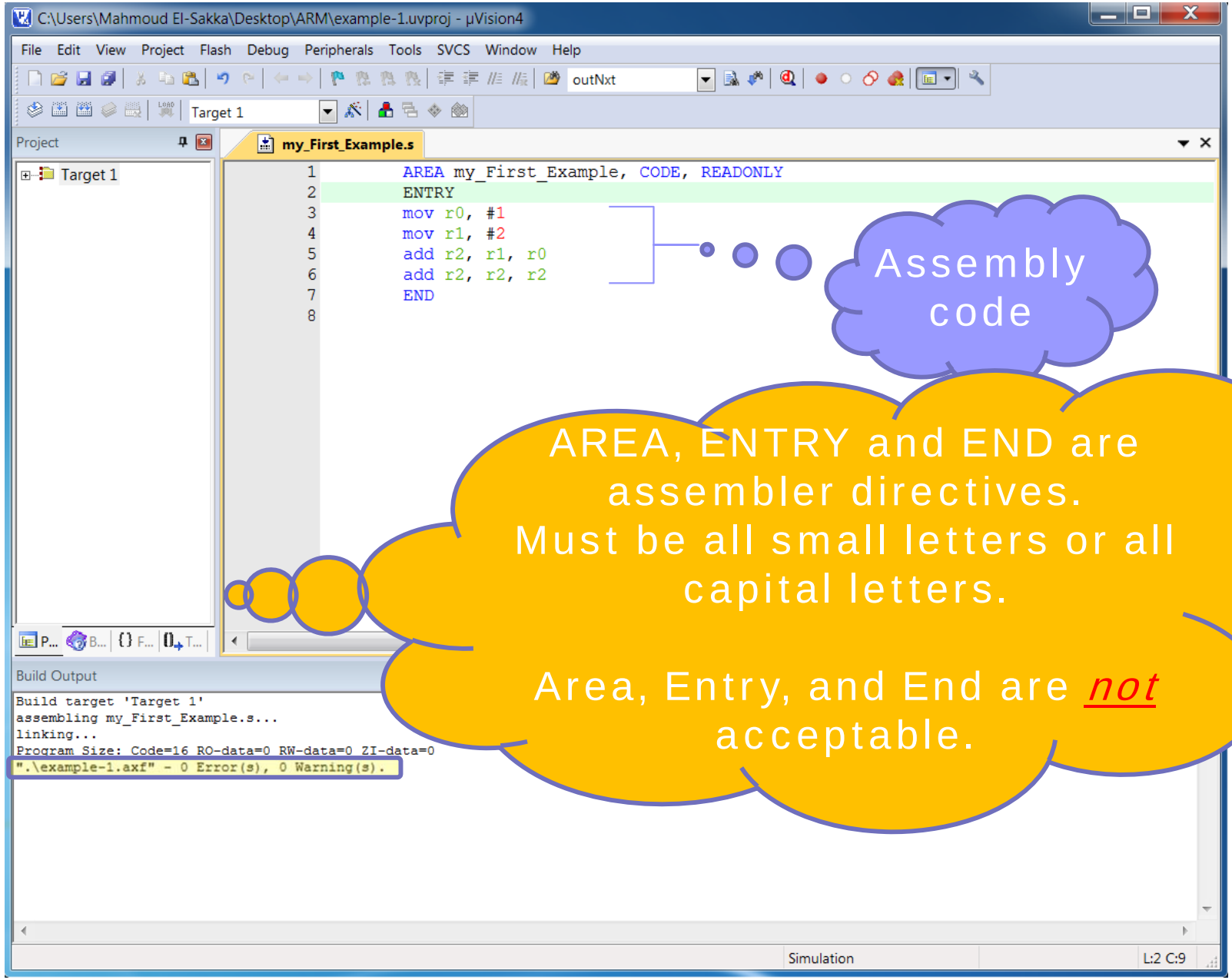
Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*



Comments on Assembly Programs Layout

- When writing an assembly language program, column one is reserved for a user-defined name that allows us to refer to that line (more specifically, it corresponds to the address of that line in the program when it's been assembled into machine code).
  - In our example, we did not use any label.

- Anywhere after column one, we can write an instruction.

{Label} Op-code **operand1**, operand2, operand3 {;comment}

Comments on Assembly Programs Layout

- When writing assembly instructions, note that:
  - there must be at least one space following the mnemonic (operation code)
  - Parameters must be separated by commas
  - Spaces after a comma are optional; for example, we can write

```
ADD r2,r1,r0
```

or

```
ADD r2  ,r1,    r0
```

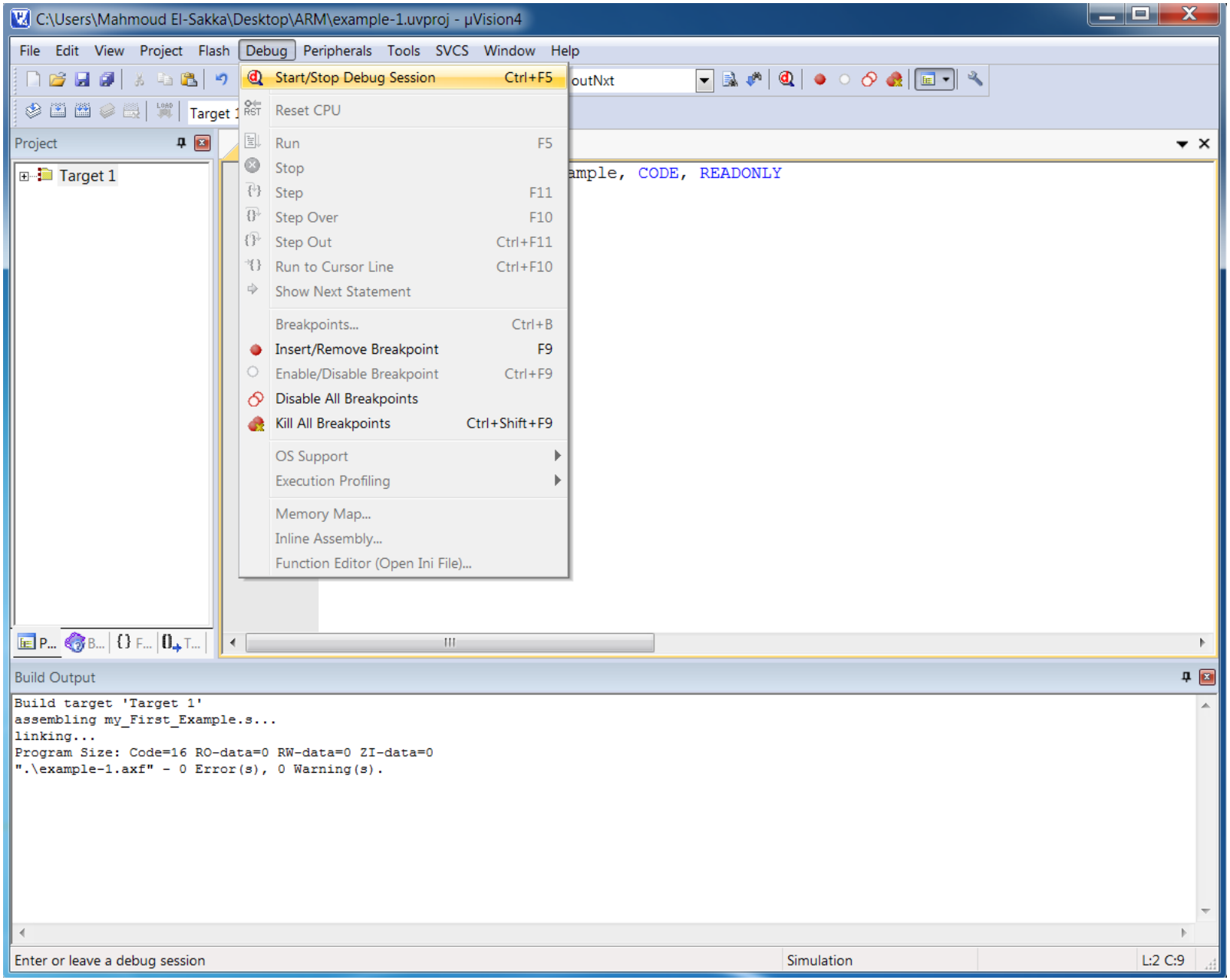
  - Operation code must be all capital letters or all small letters

```
ADD r2,r1,r0 ; Ok
add r2,r1,r0 ; Ok
Add r2,r1,r0 ; Not Ok
```

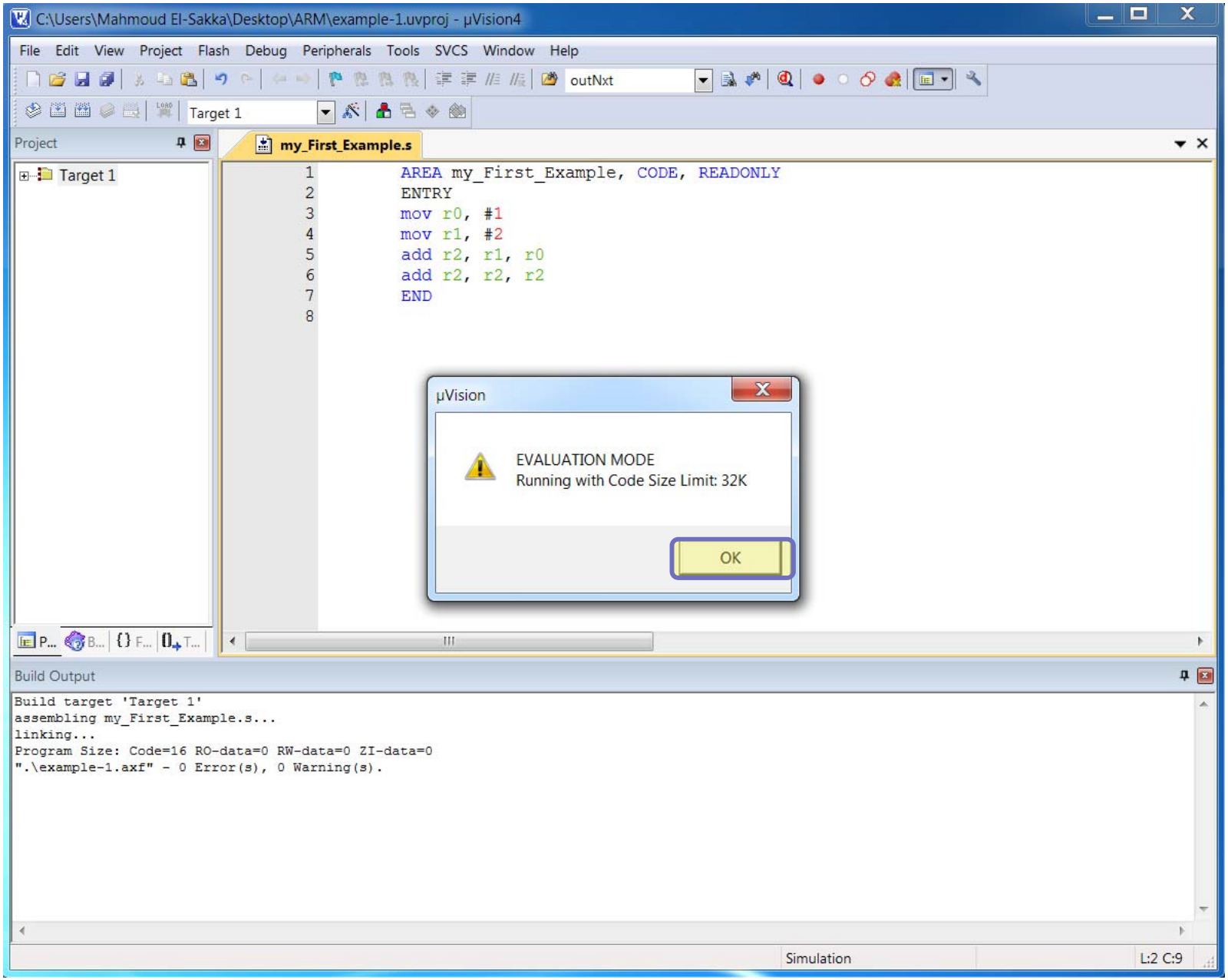
Comments on Assembly Programs Layout

- Finally, we can append a comment to the right.
  - The assembler we are using requires a semicolon to separate a comment from the code.
- Although we don't have to write a program in columns as we've done above, it makes the program easier to read.

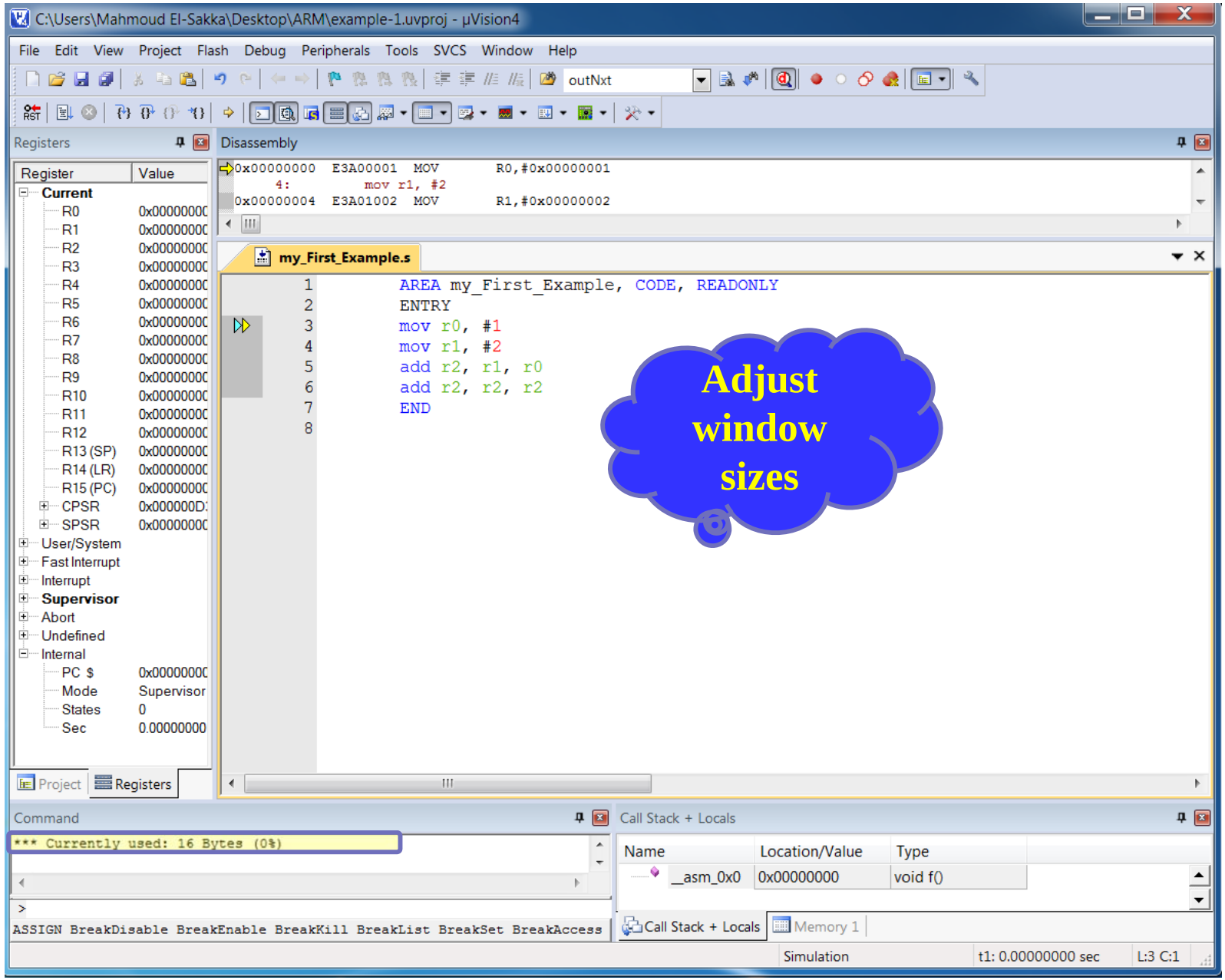
Using the Simulator—*step-by-step*



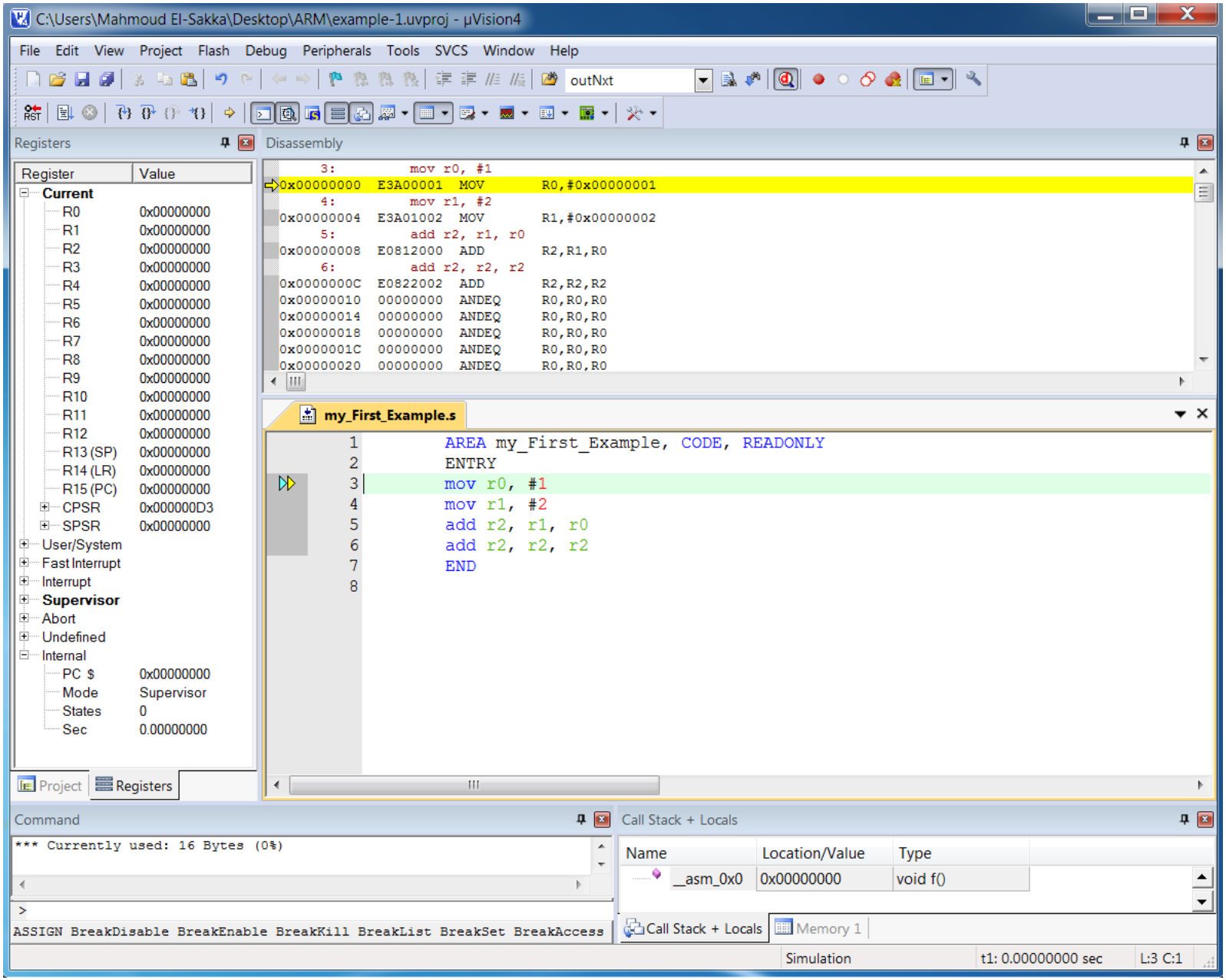
Using the Simulator—*step-by-step*



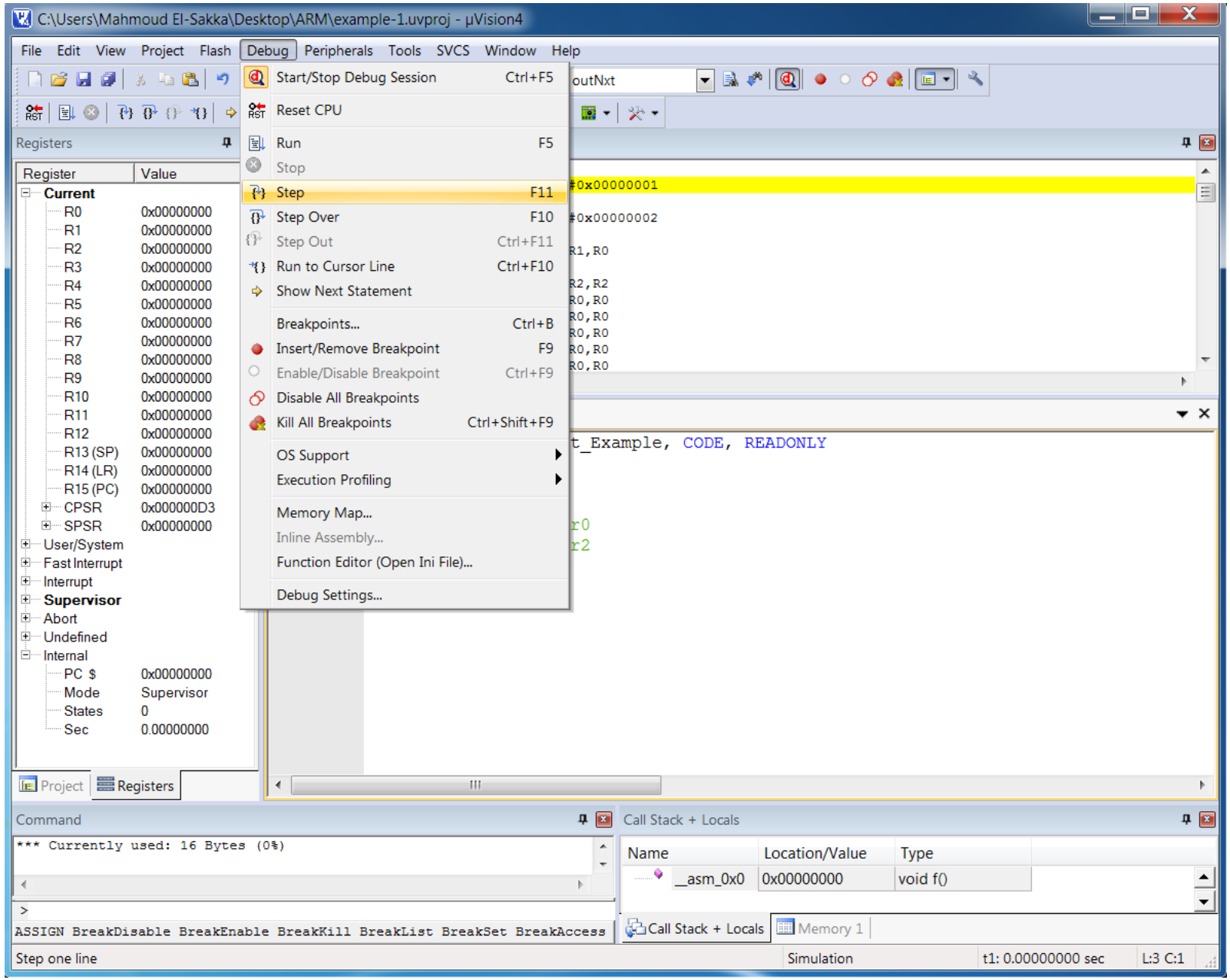
Using the Simulator—*step-by-step*



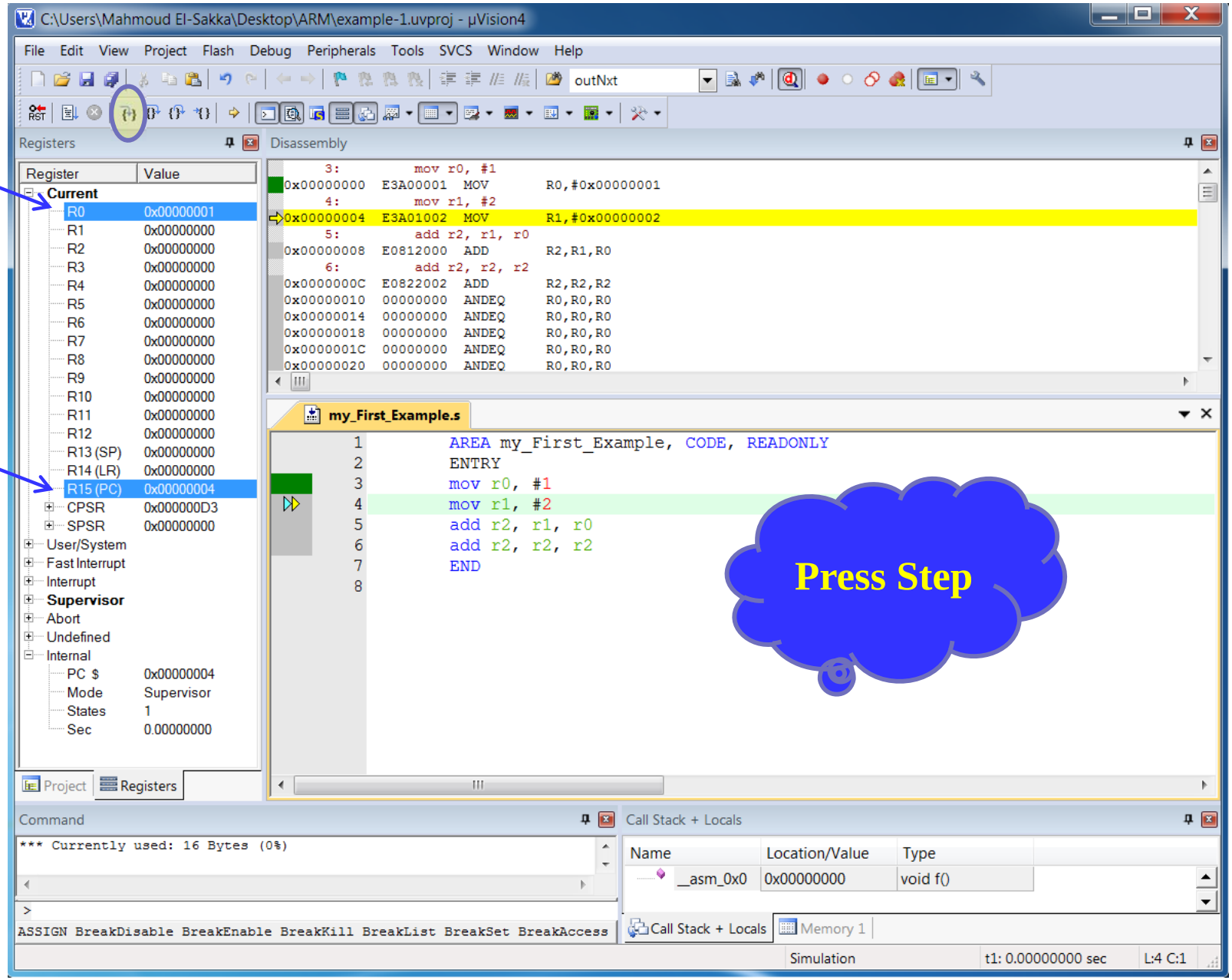
Using the Simulator—*step-by-step*



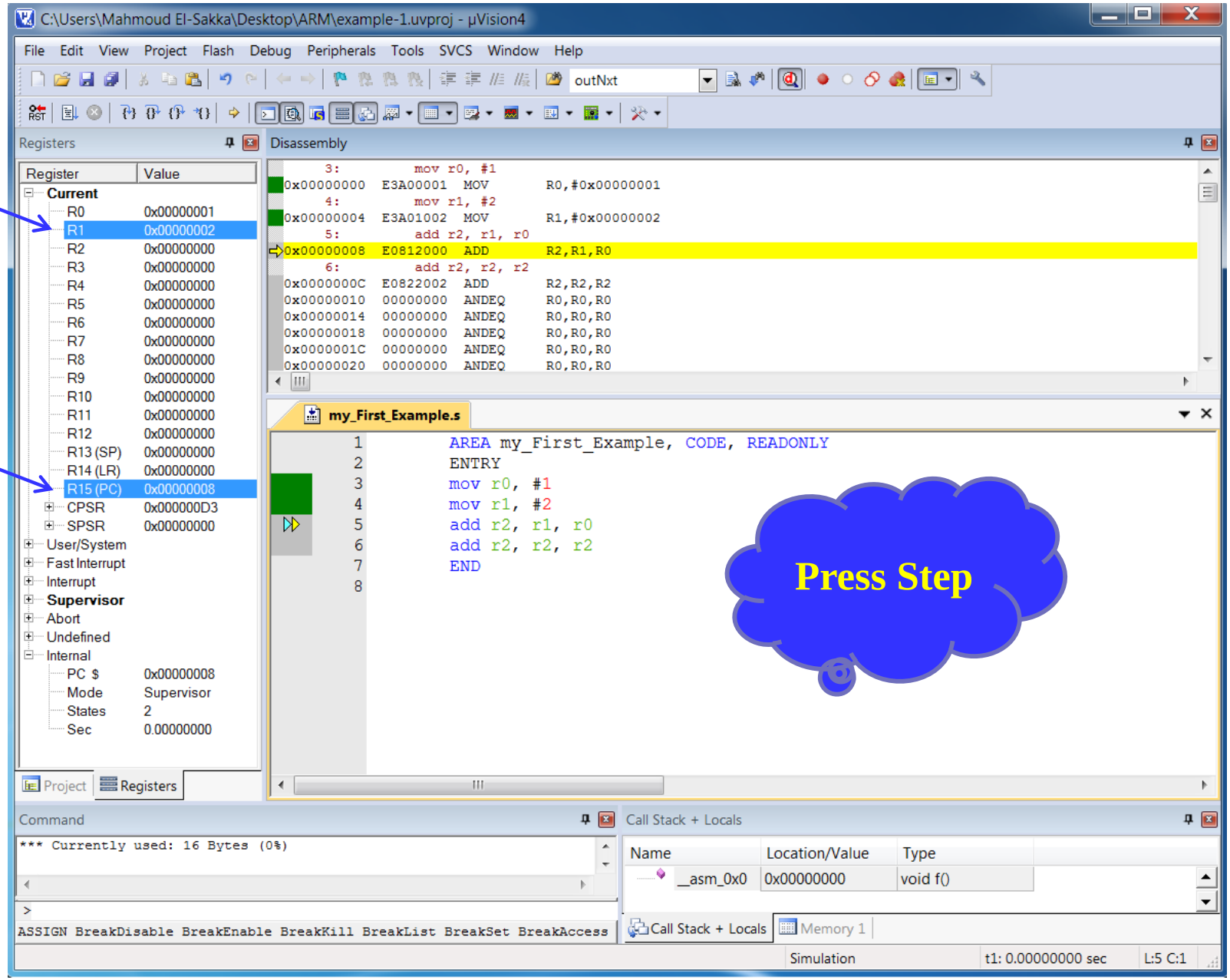
Using the Simulator—*step-by-step*



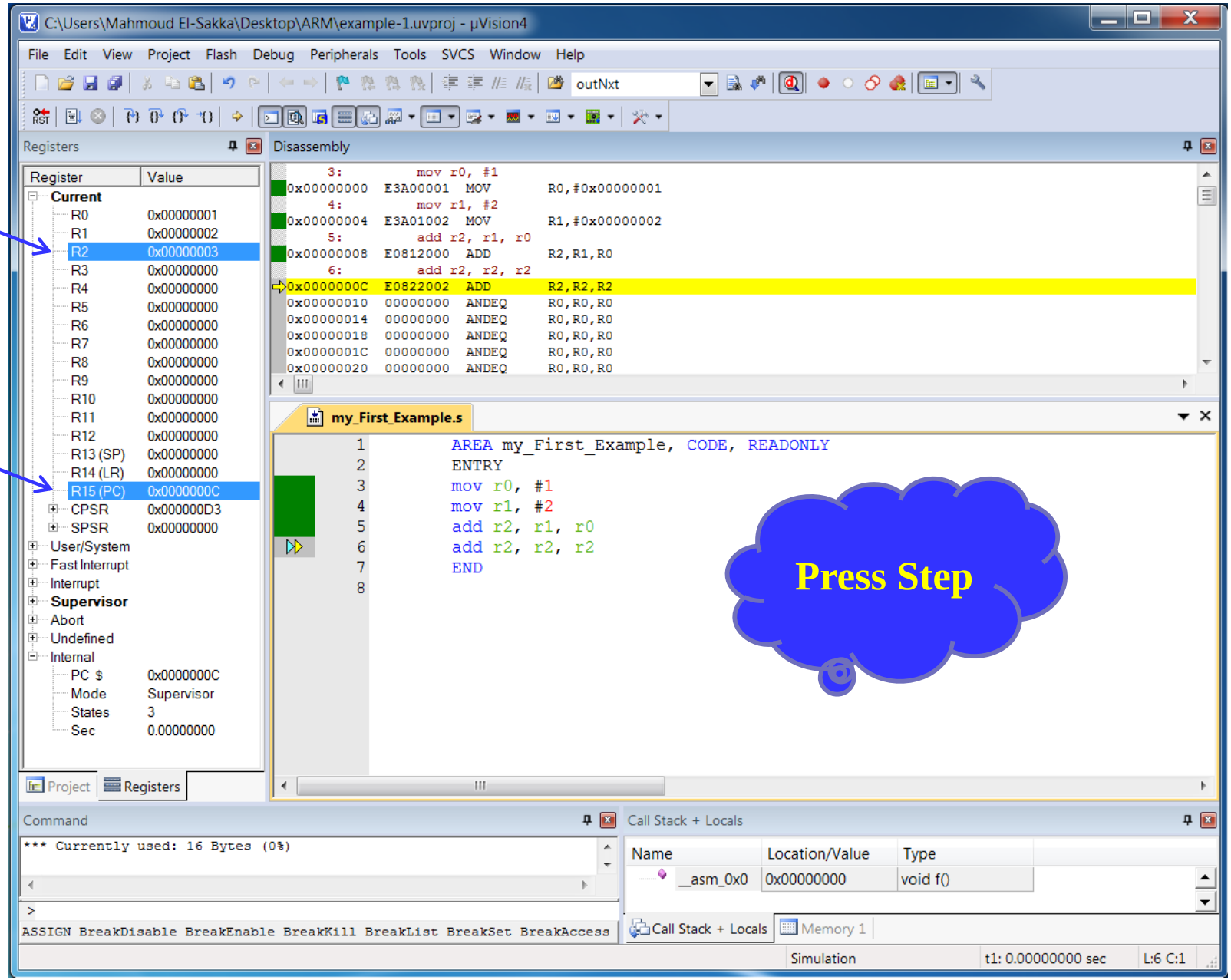
Using the Simulator—*step-by-step*



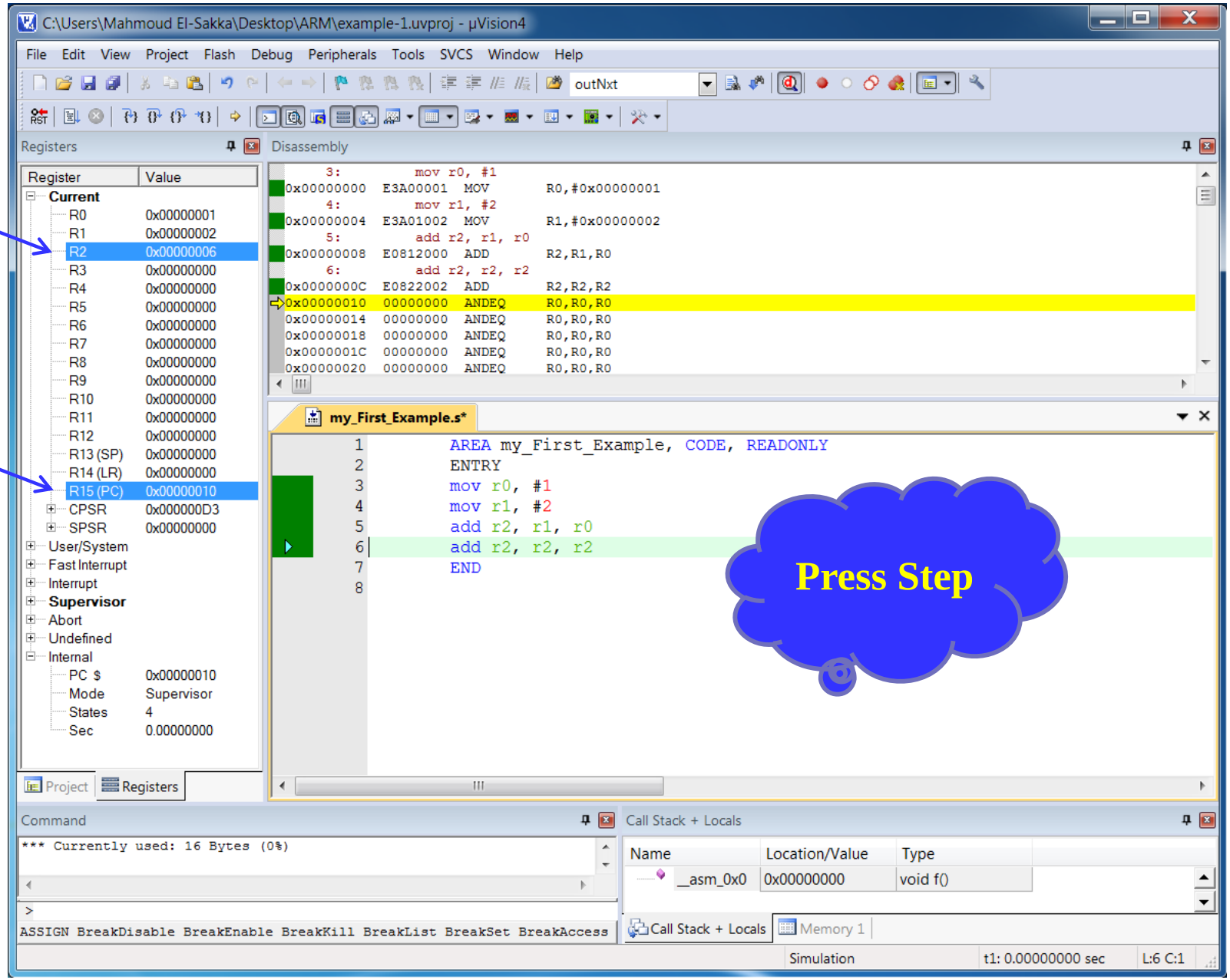
Using the Simulator—*step-by-step*



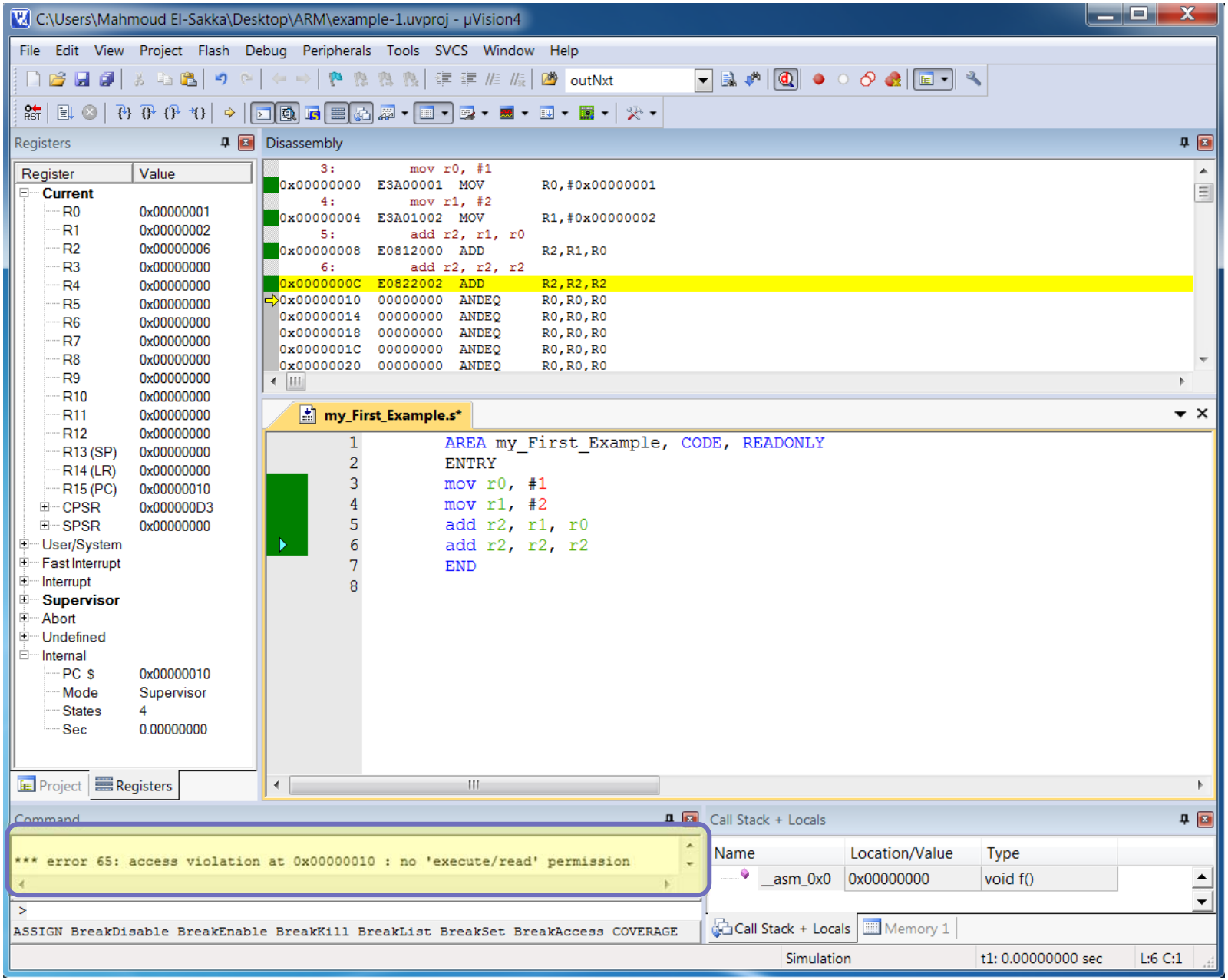
Using the Simulator—*step-by-step*



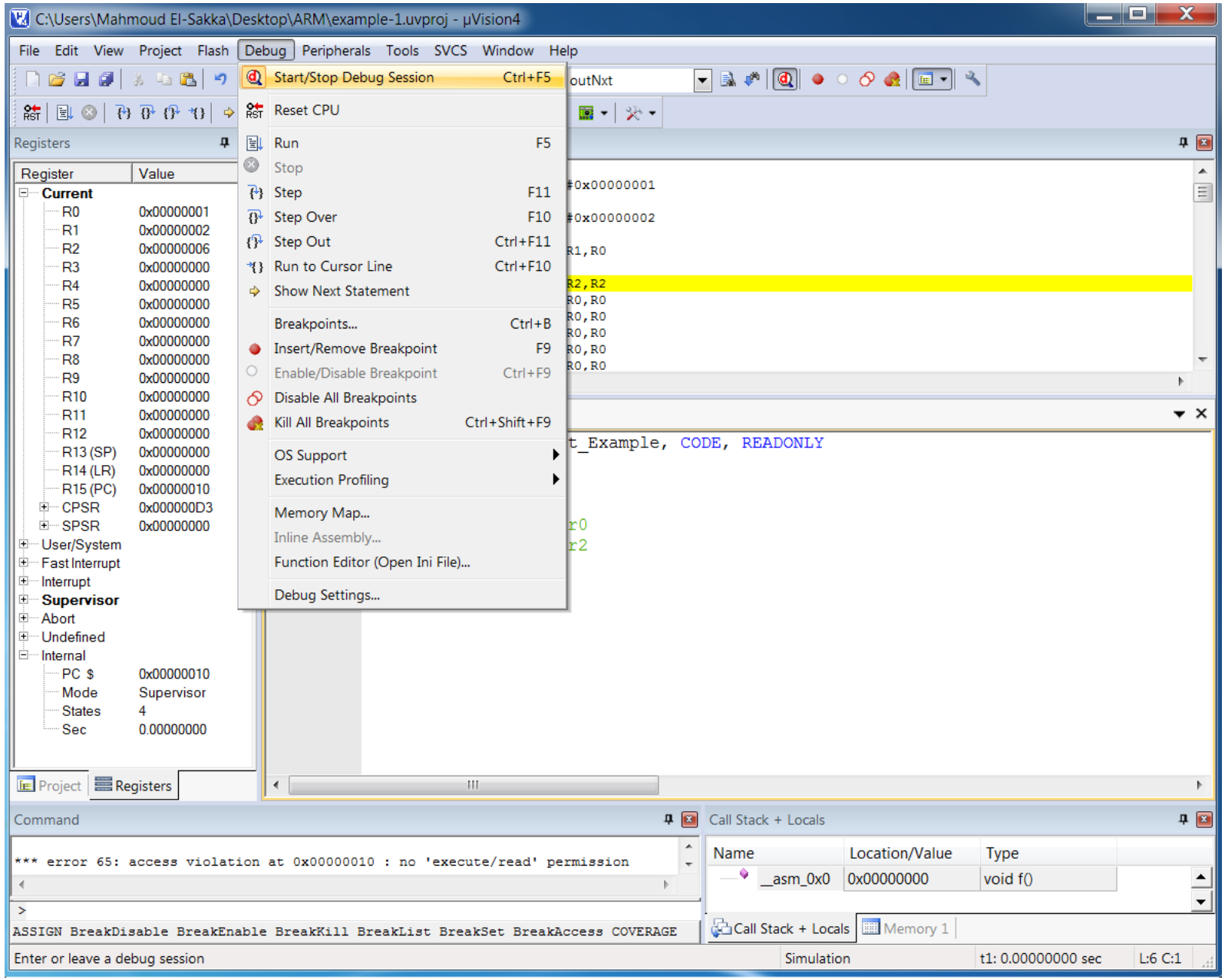
Using the Simulator—*step-by-step*



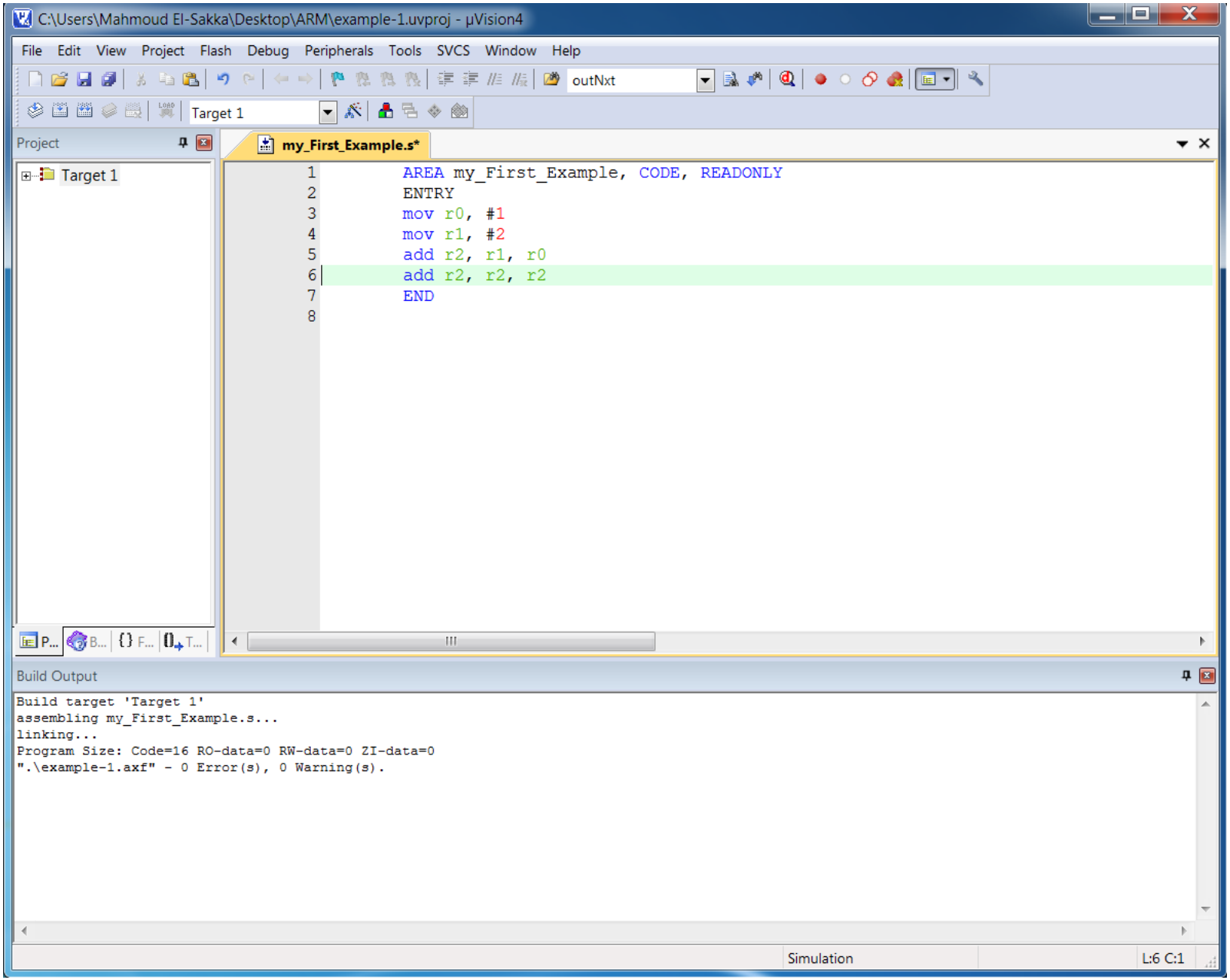
Using the Simulator—*step-by-step*



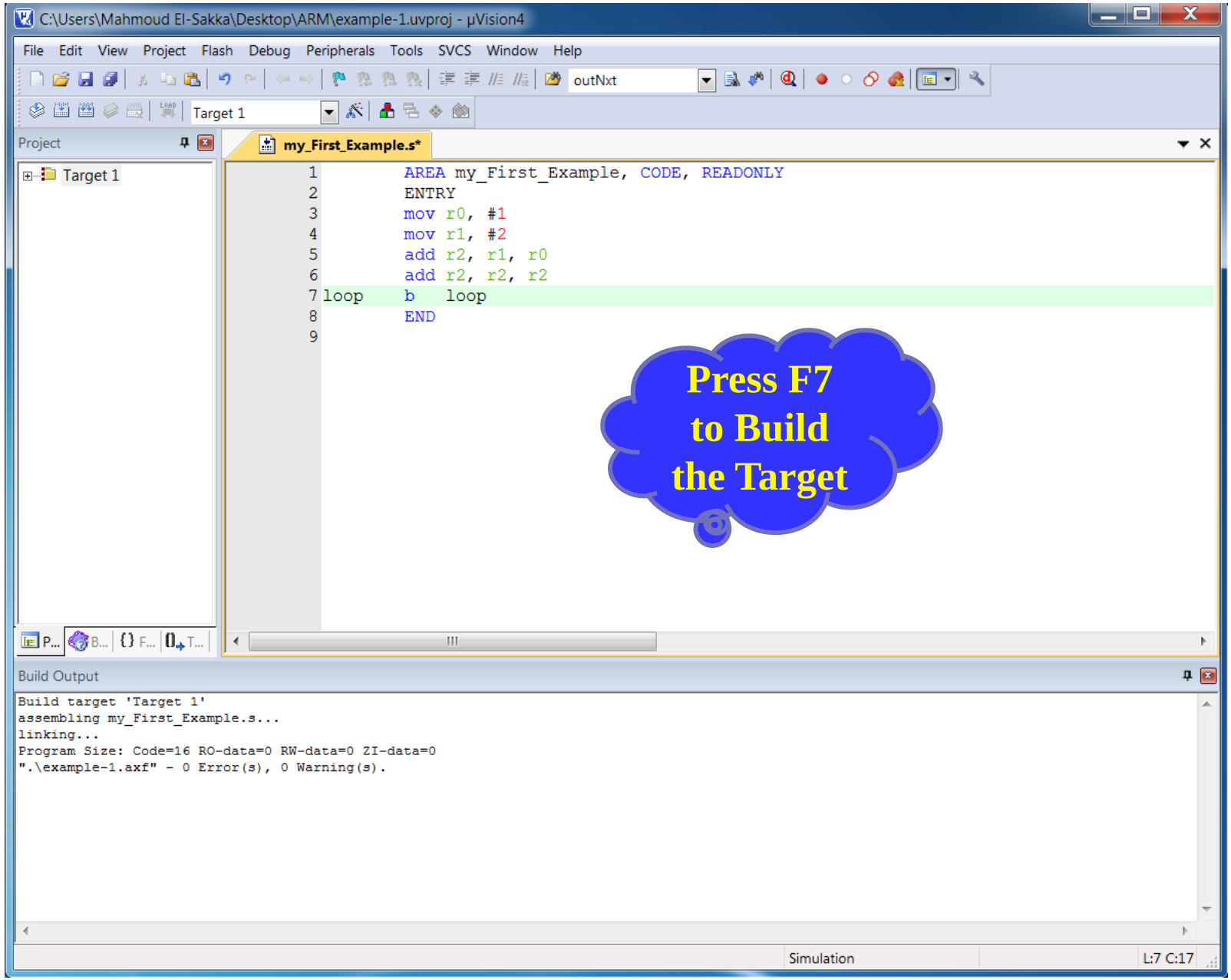
Using the Simulator—*step-by-step*



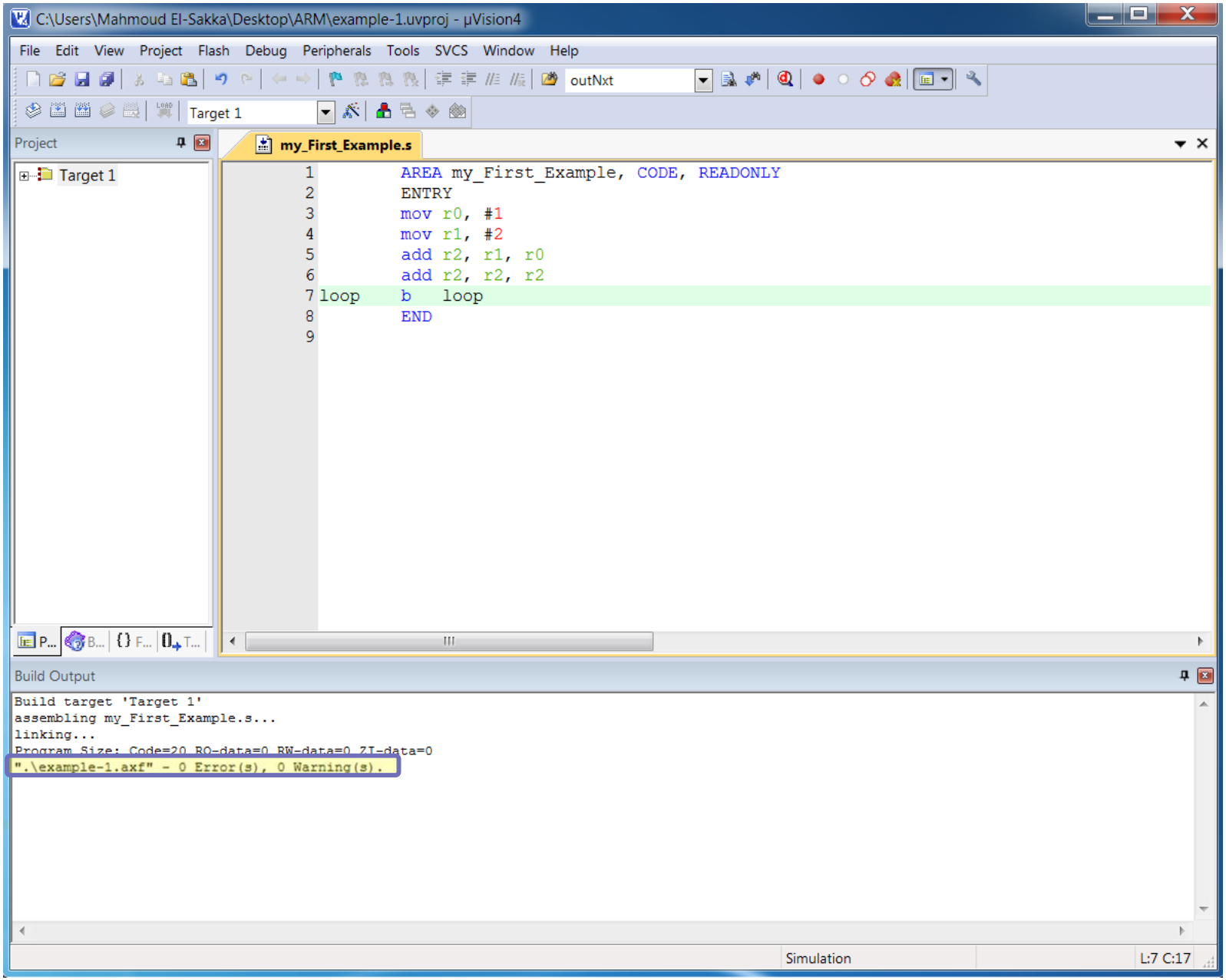
Using the Simulator—*step-by-step*



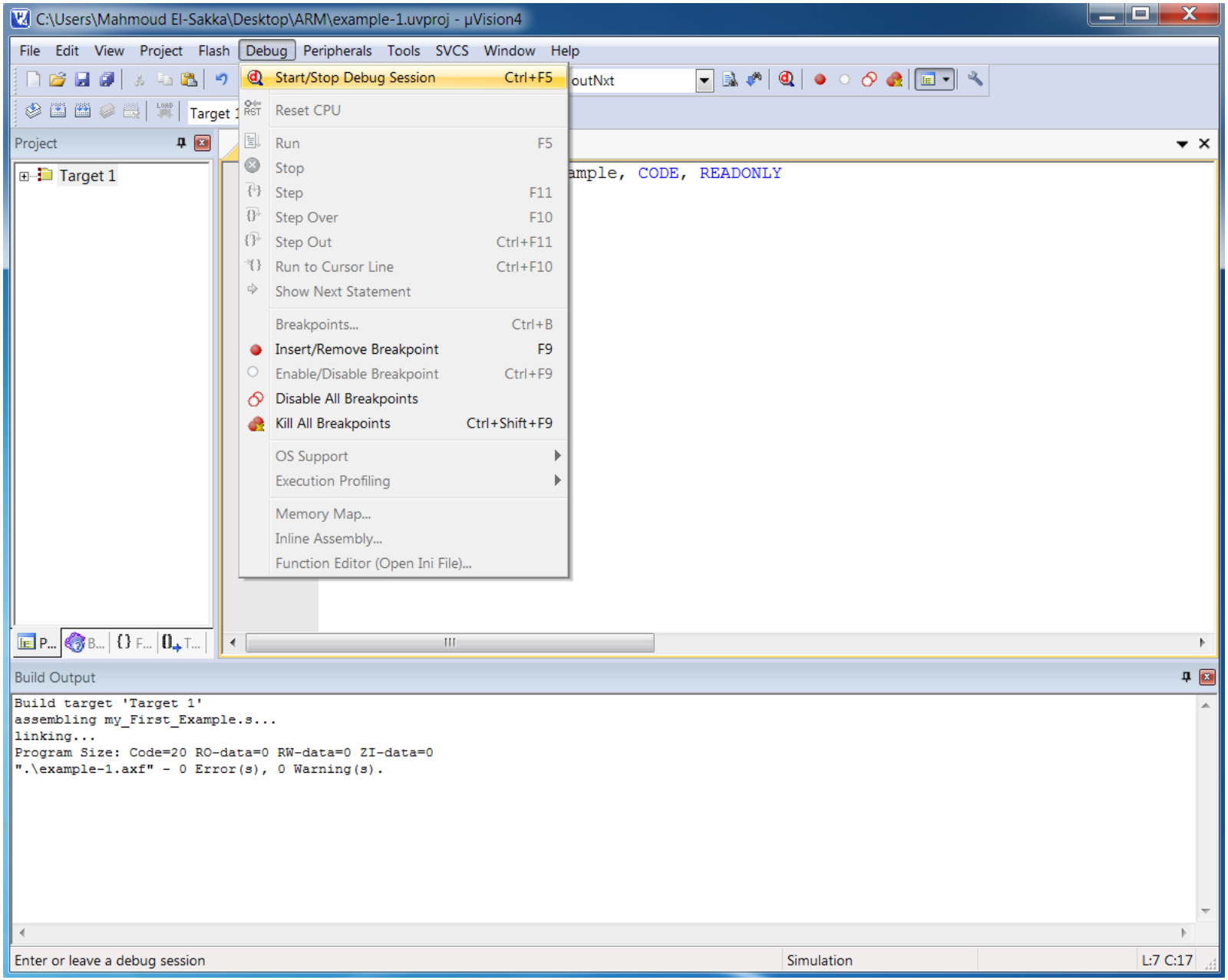
Using the Simulator—*step-by-step*



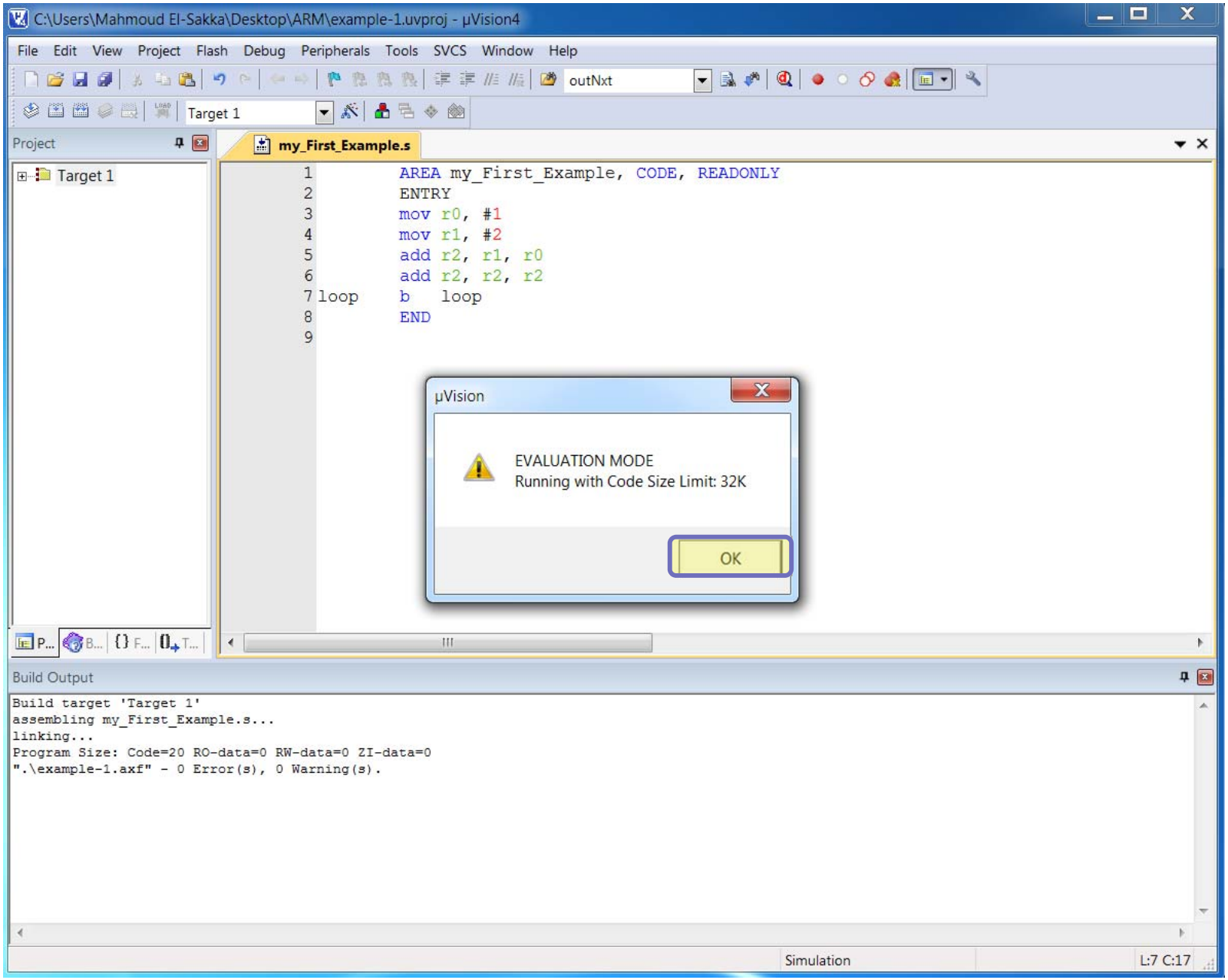
Using the Simulator—*step-by-step*



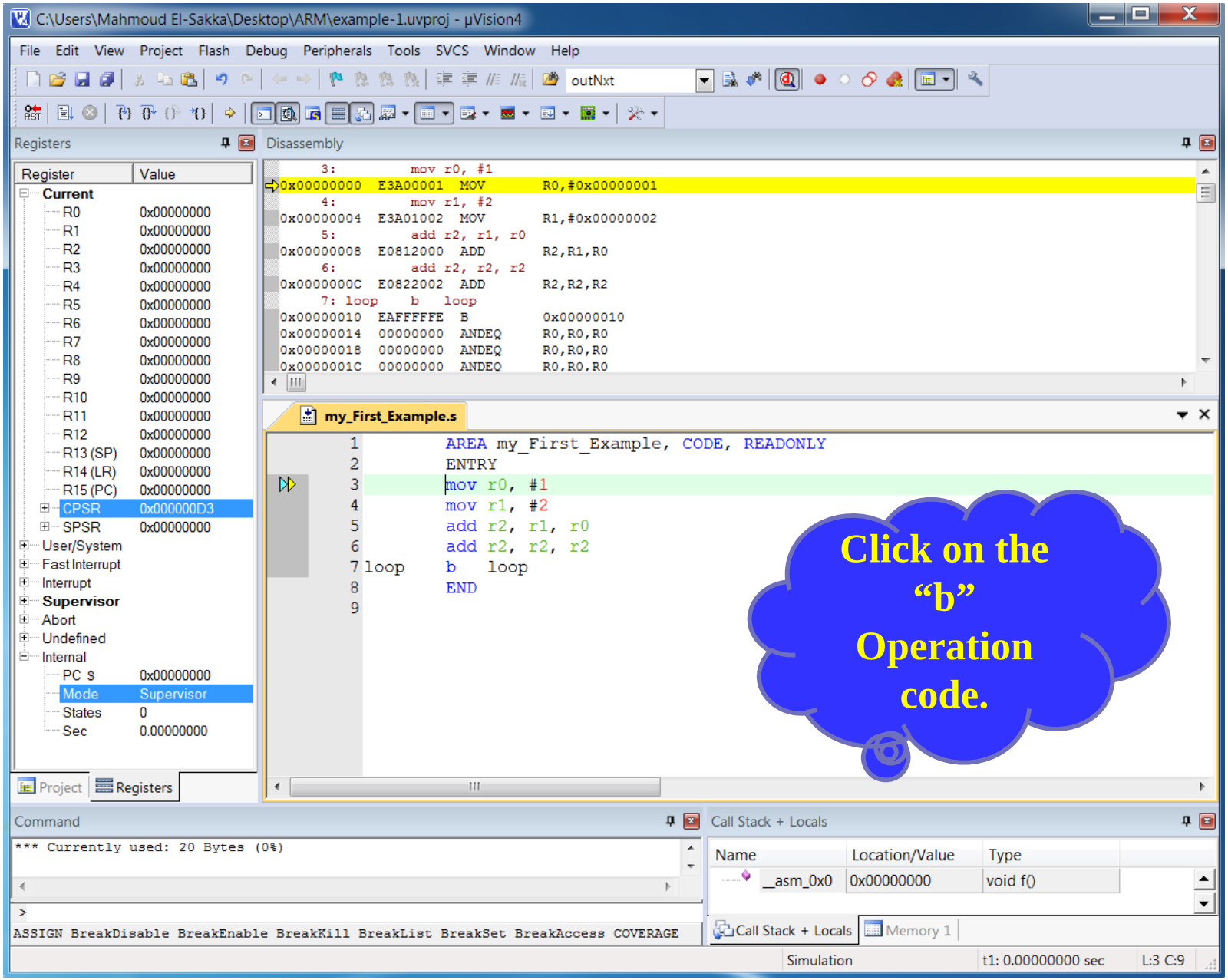
Using the Simulator—*step-by-step*



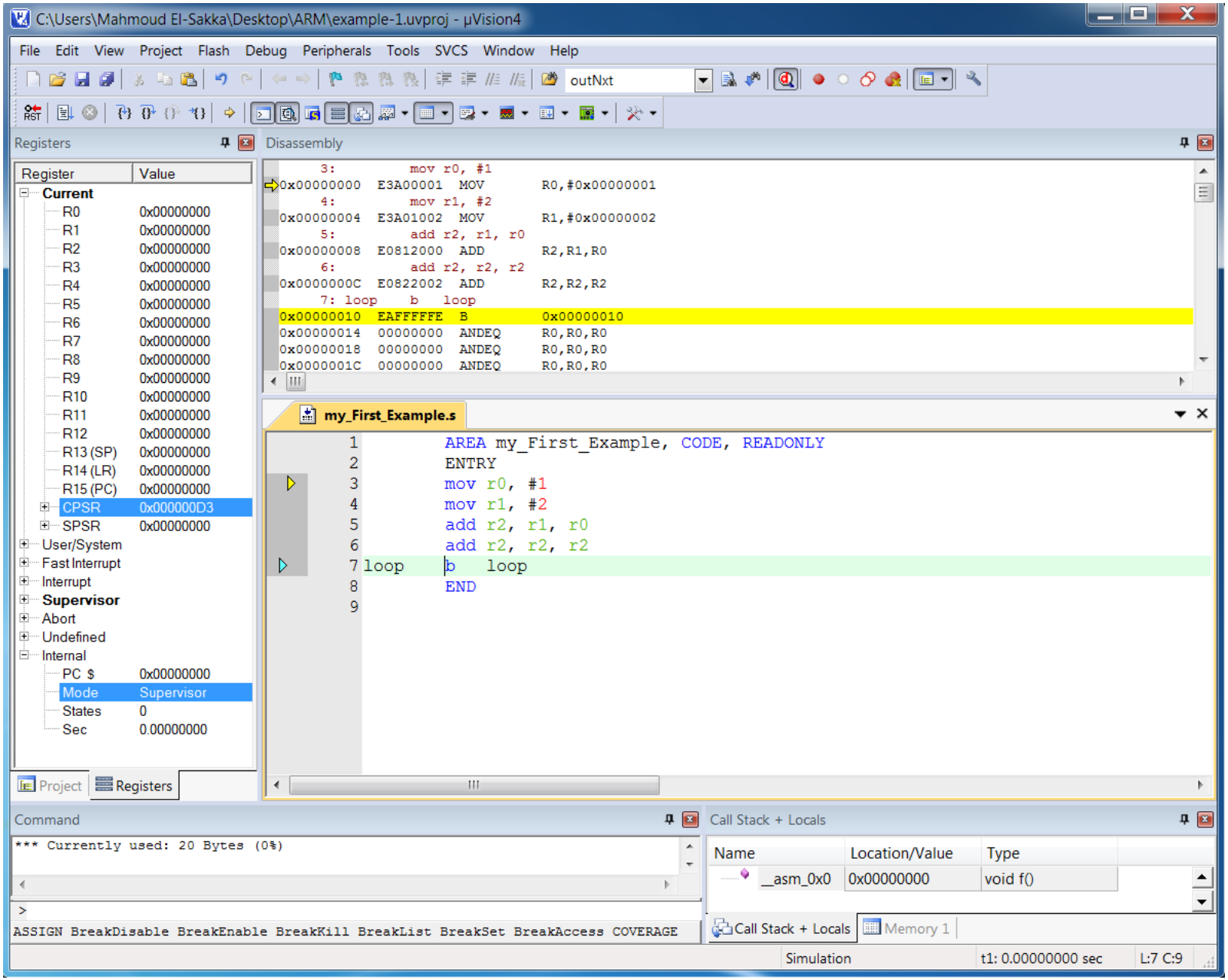
Using the Simulator—*step-by-step*



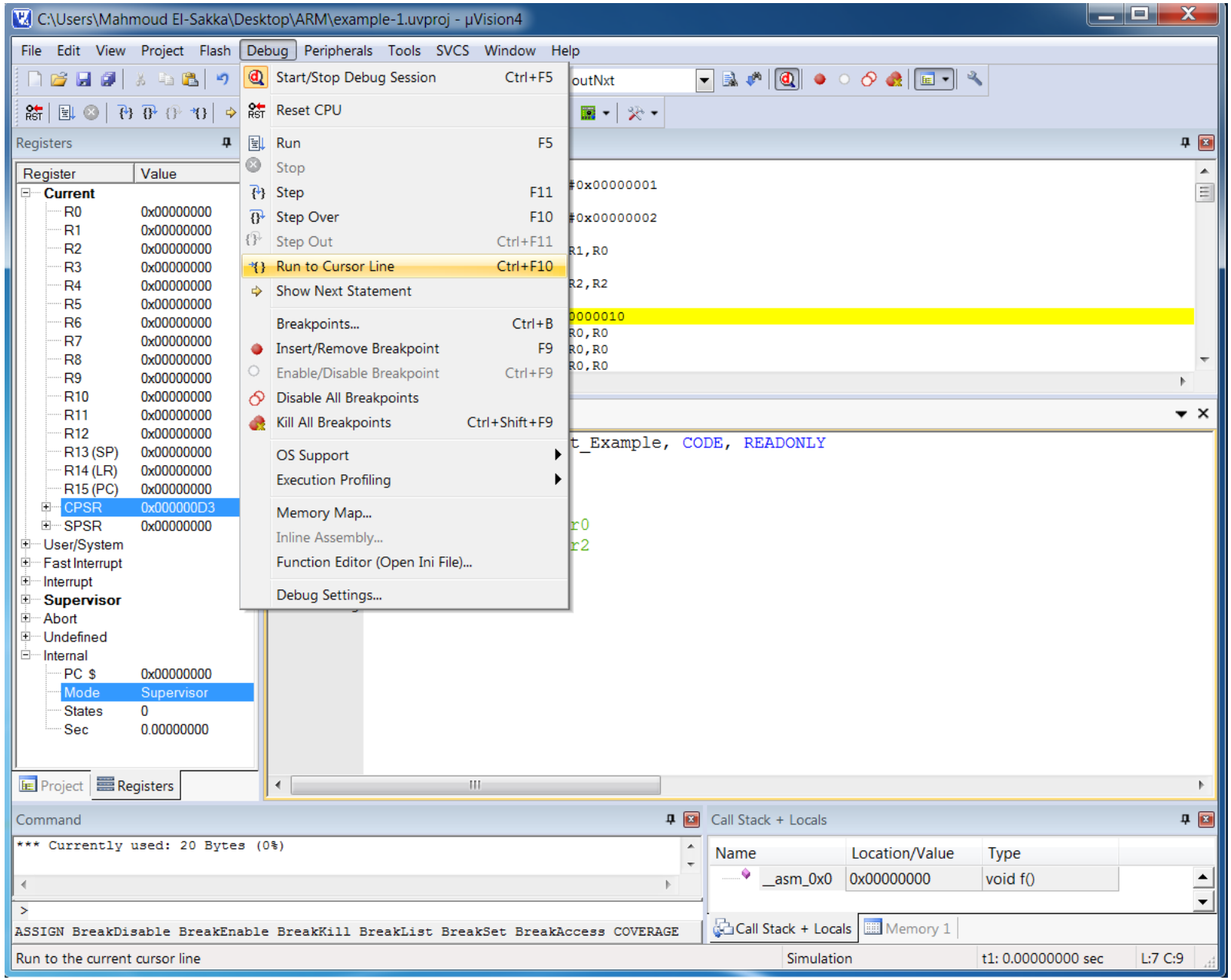
Using the Simulator—*step-by-step*



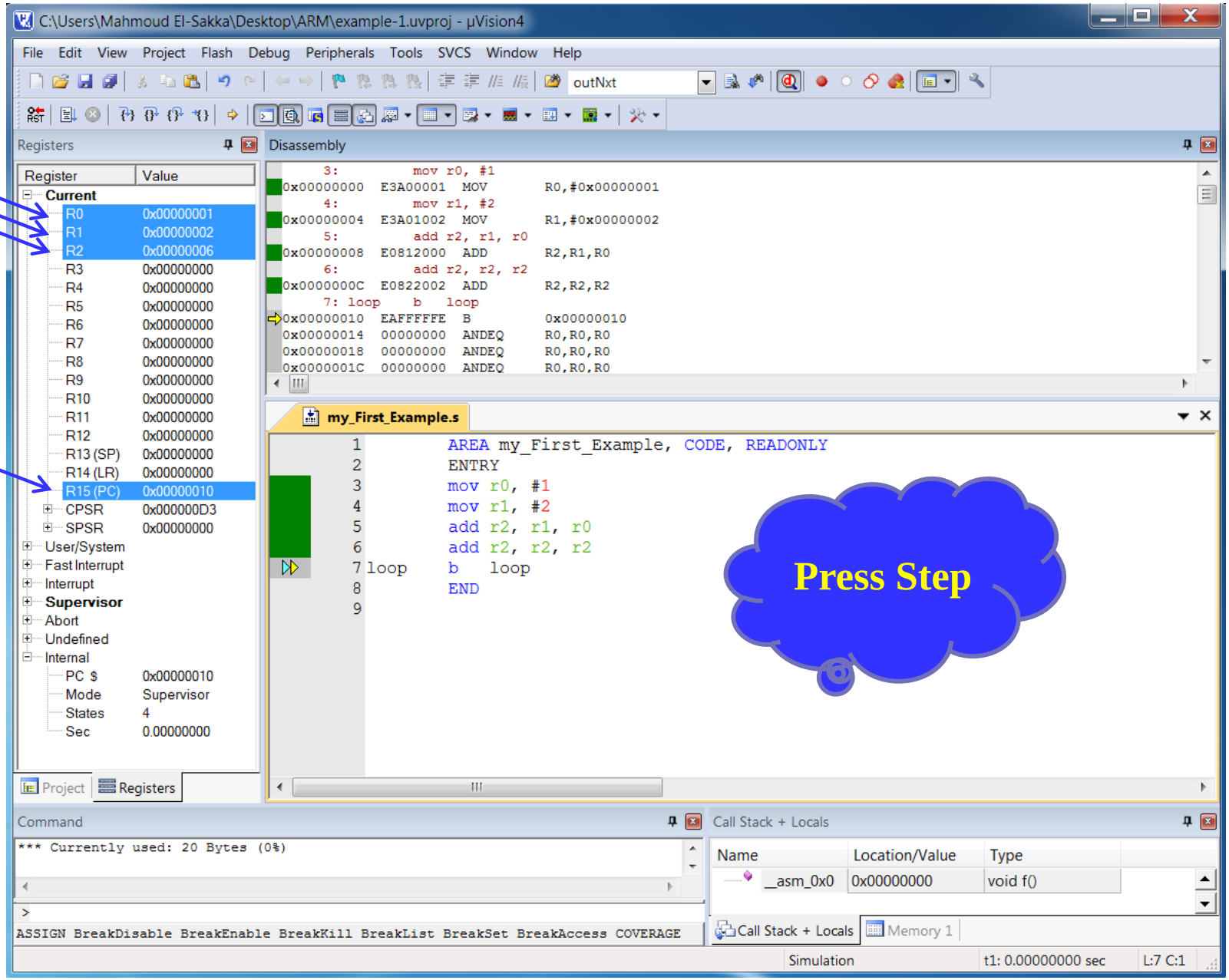
Using the Simulator—*step-by-step*



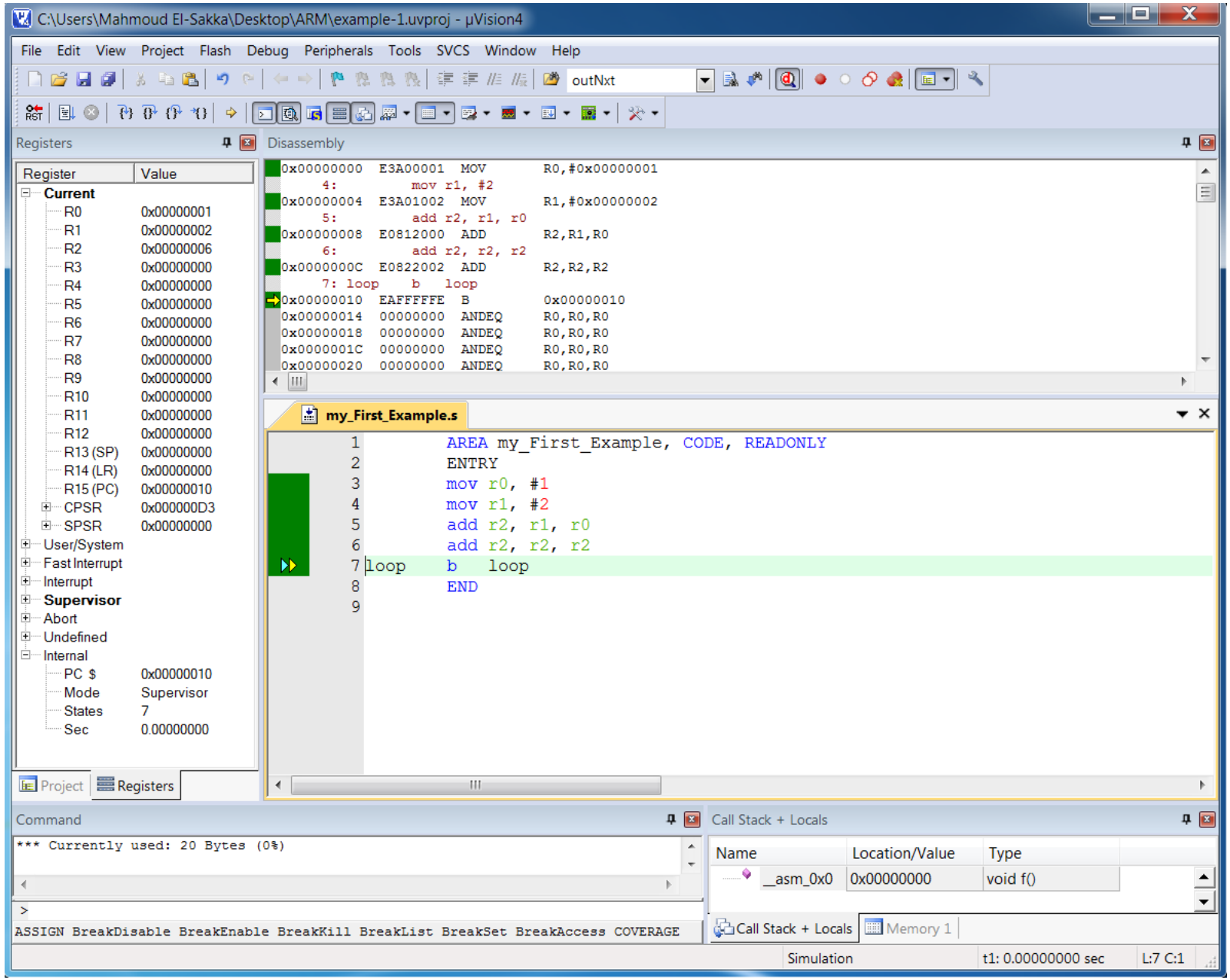
Using the Simulator—*step-by-step*



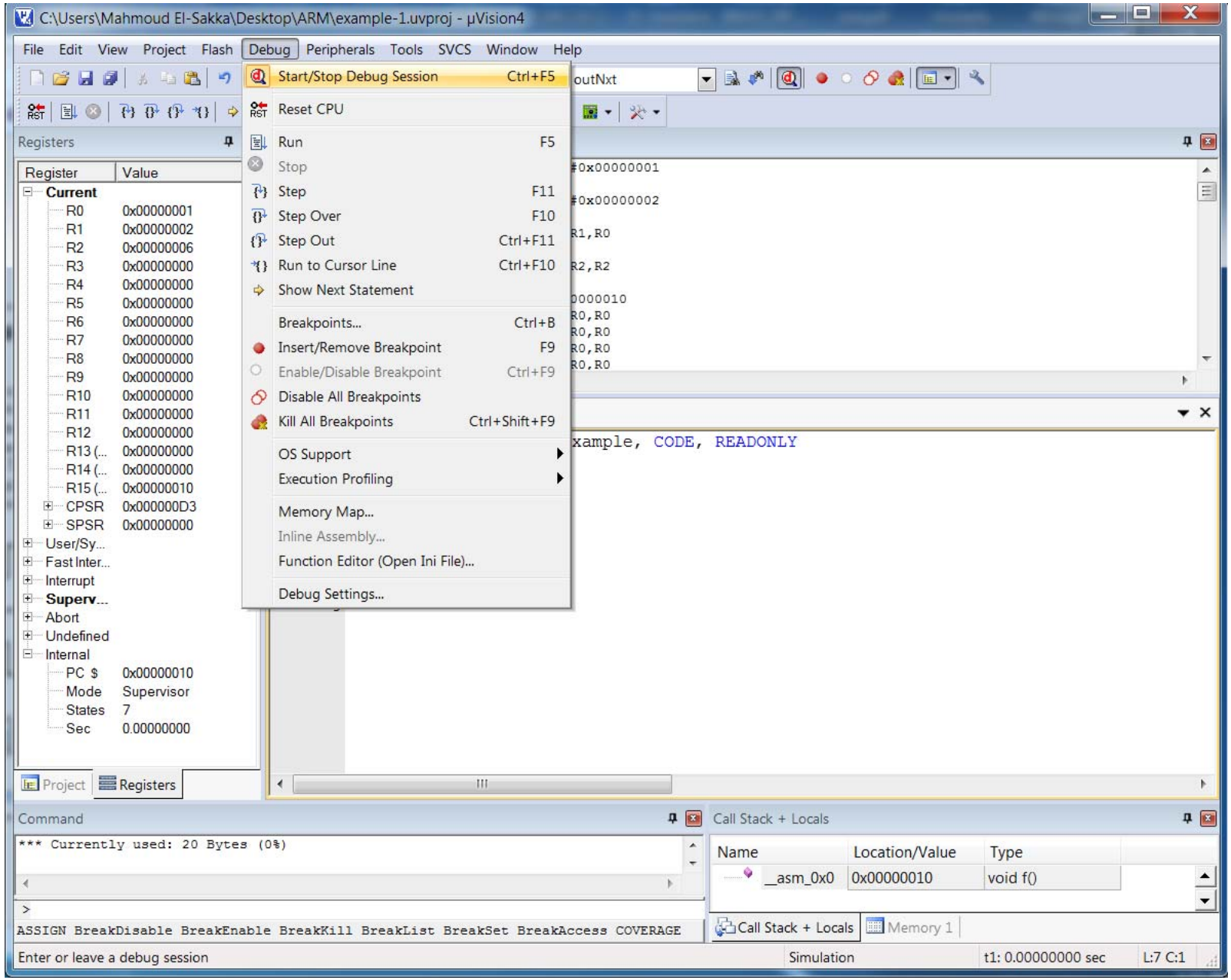
Using the Simulator—*step-by-step*



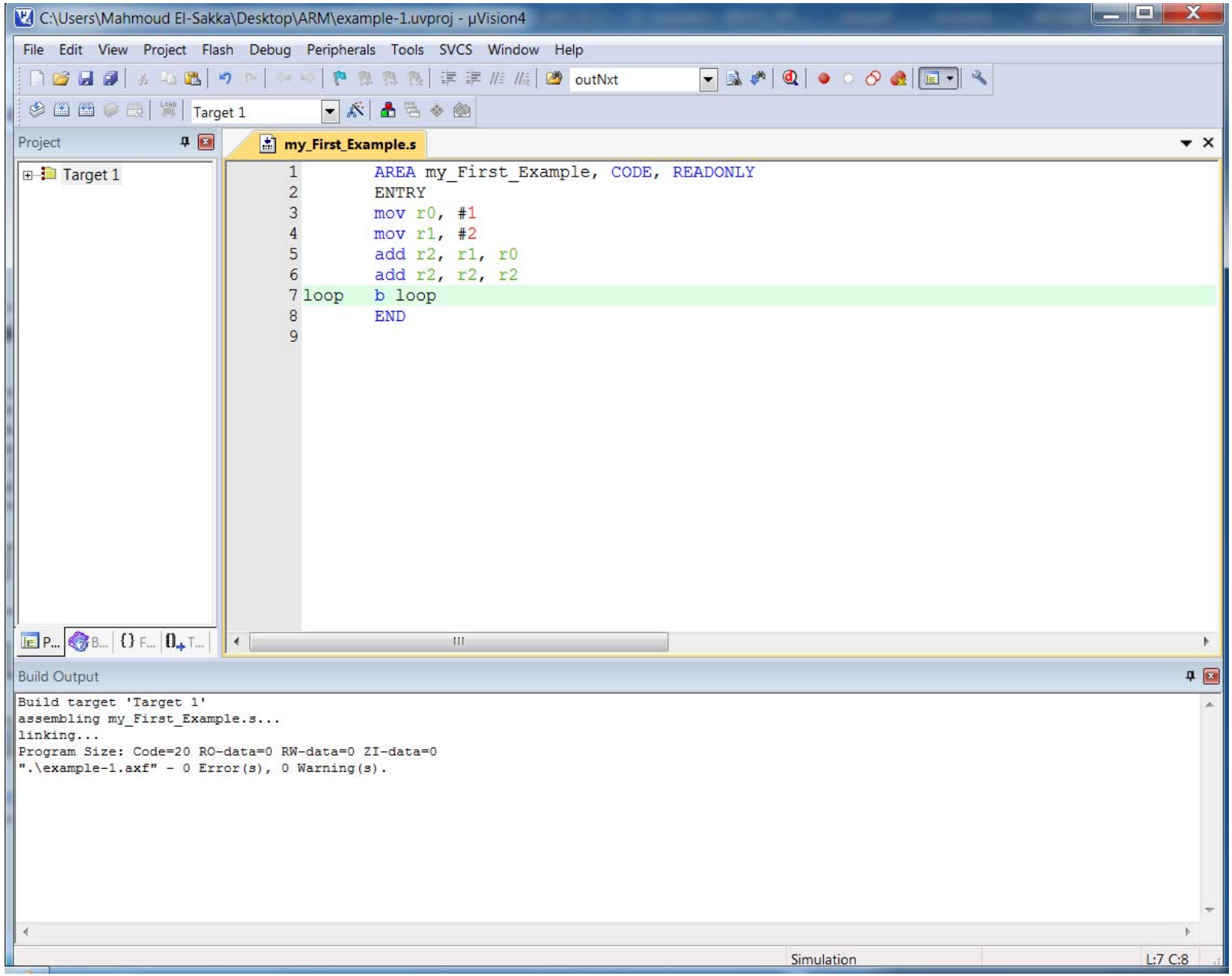
Using the Simulator—*step-by-step*



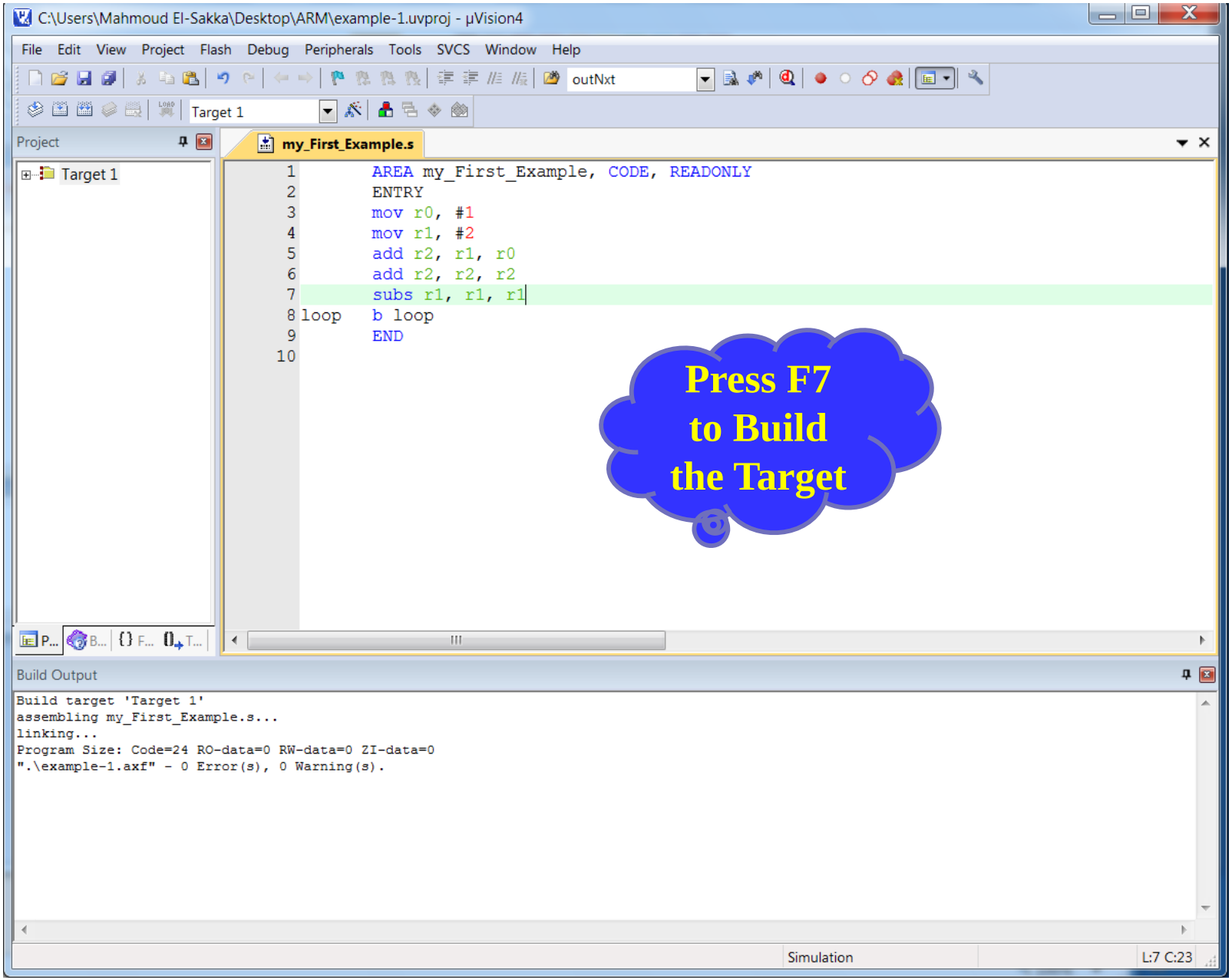
Using the Simulator—*step-by-step*



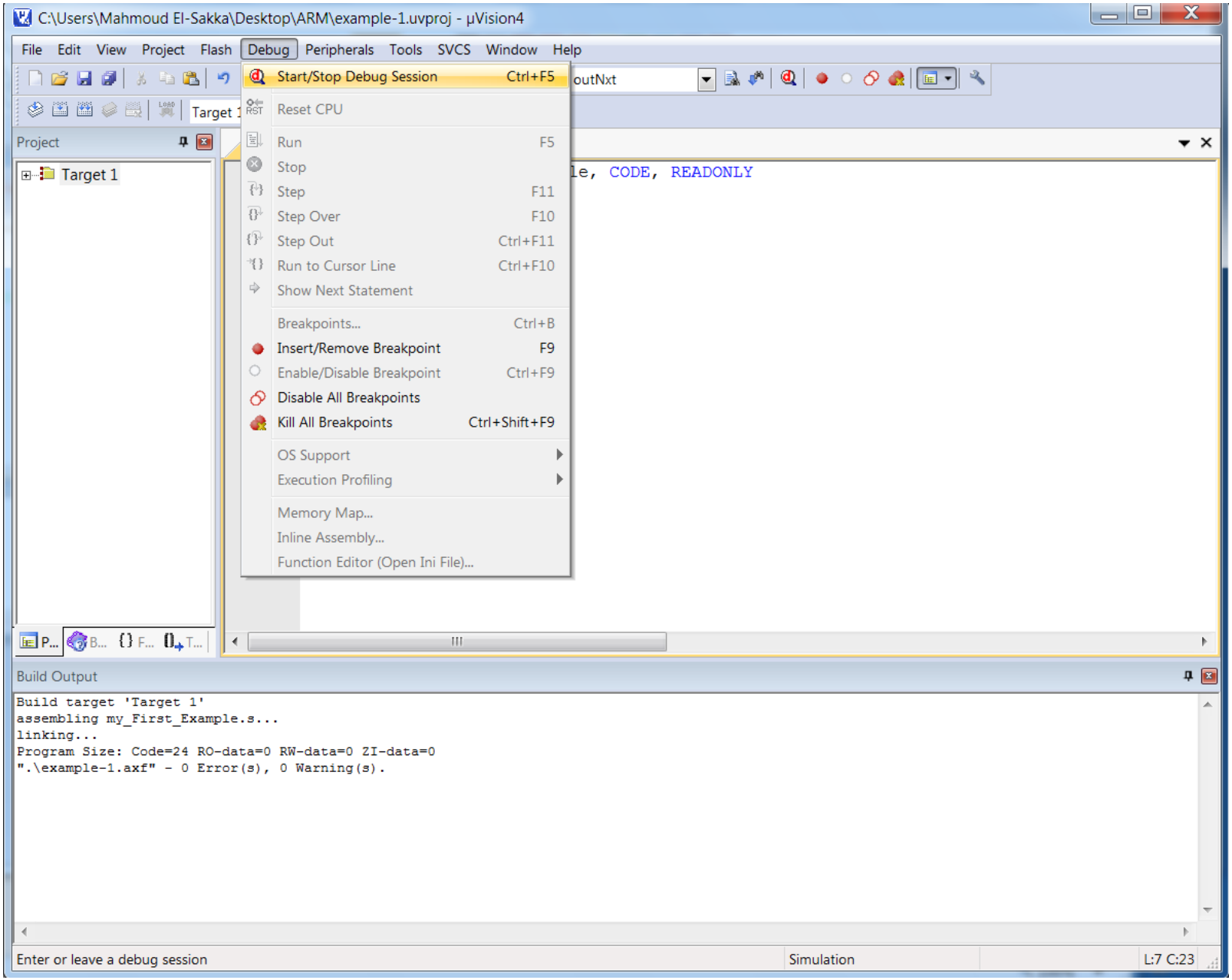
Using the Simulator—*step-by-step*



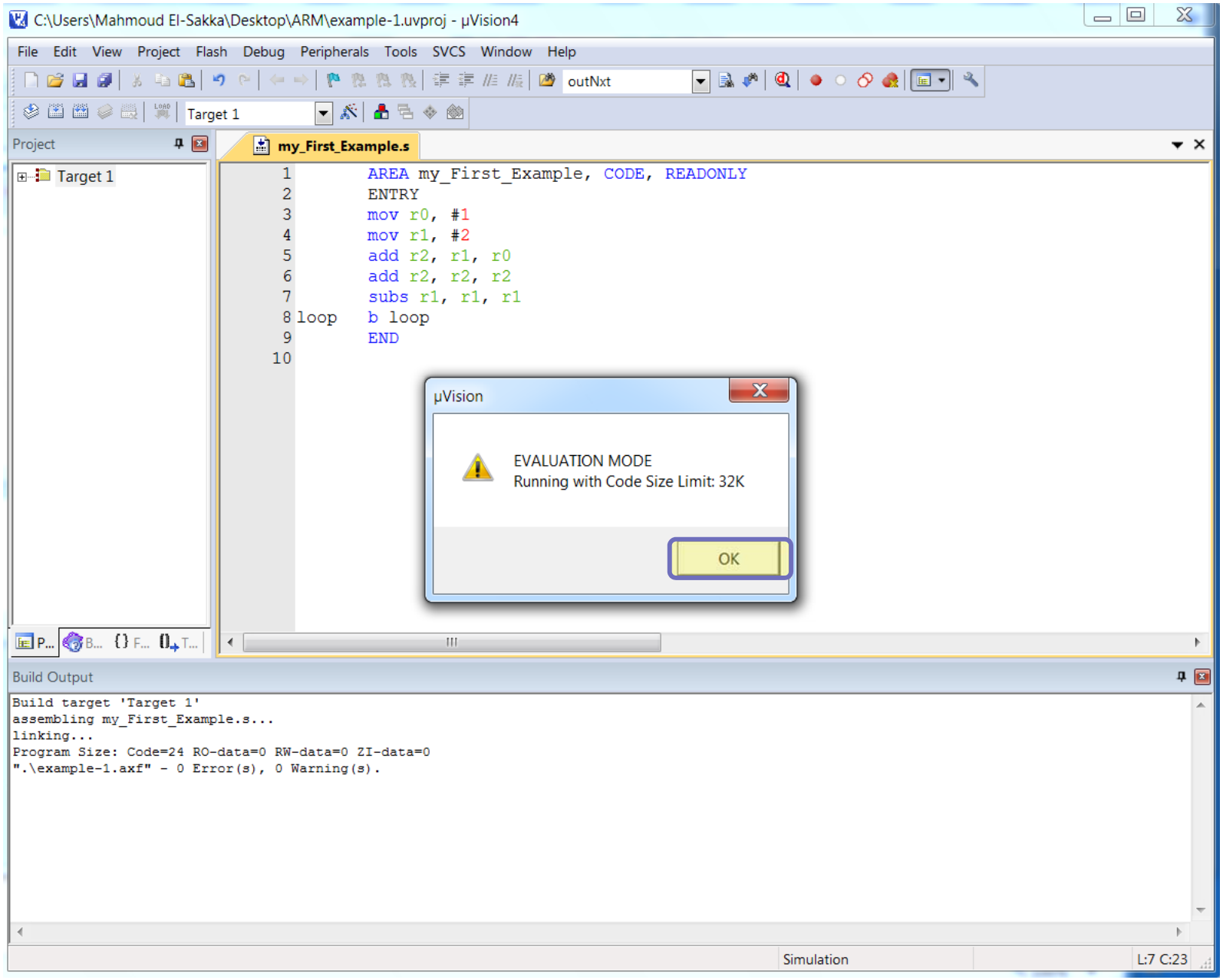
Using the Simulator—*step-by-step*



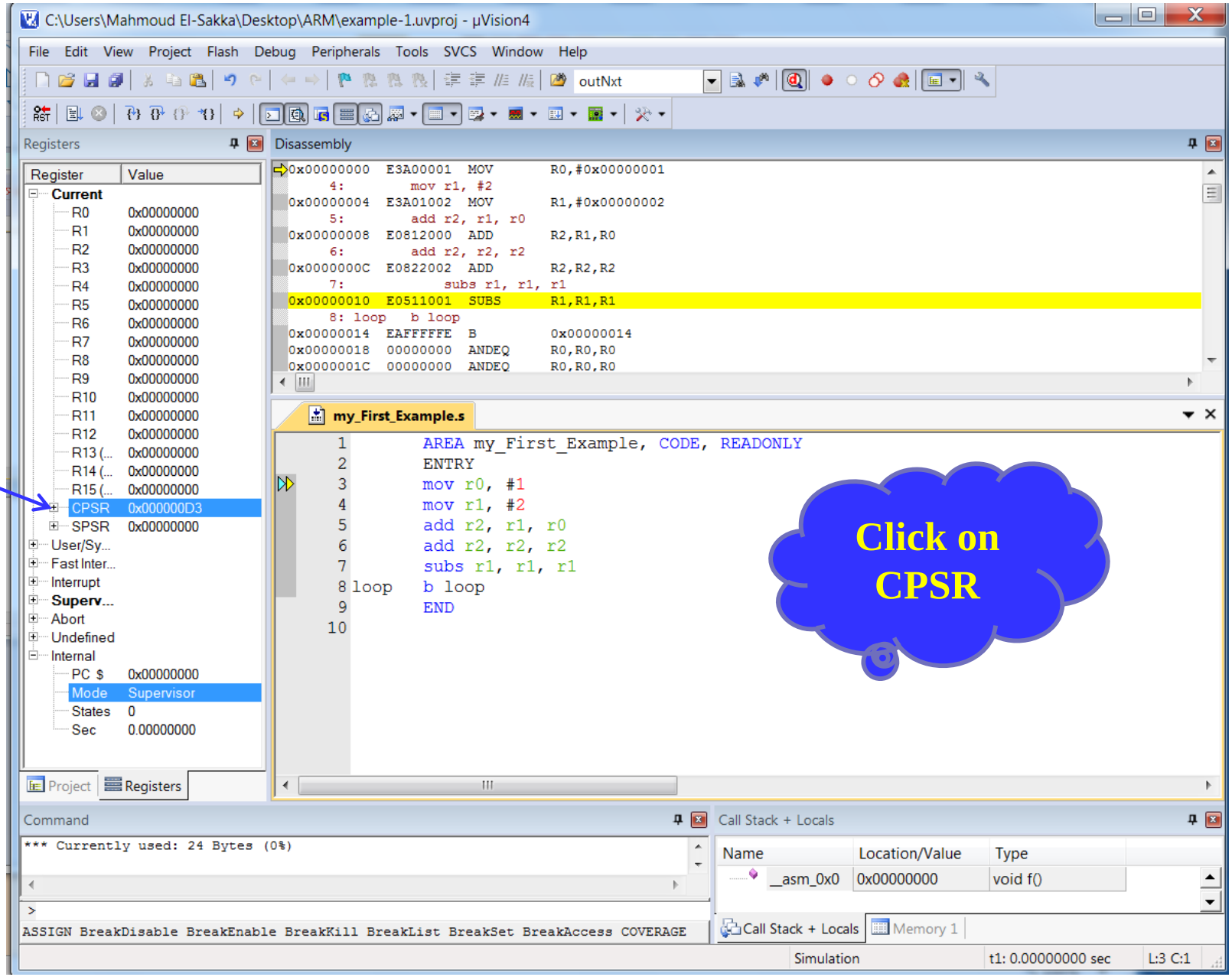
Using the Simulator—*step-by-step*



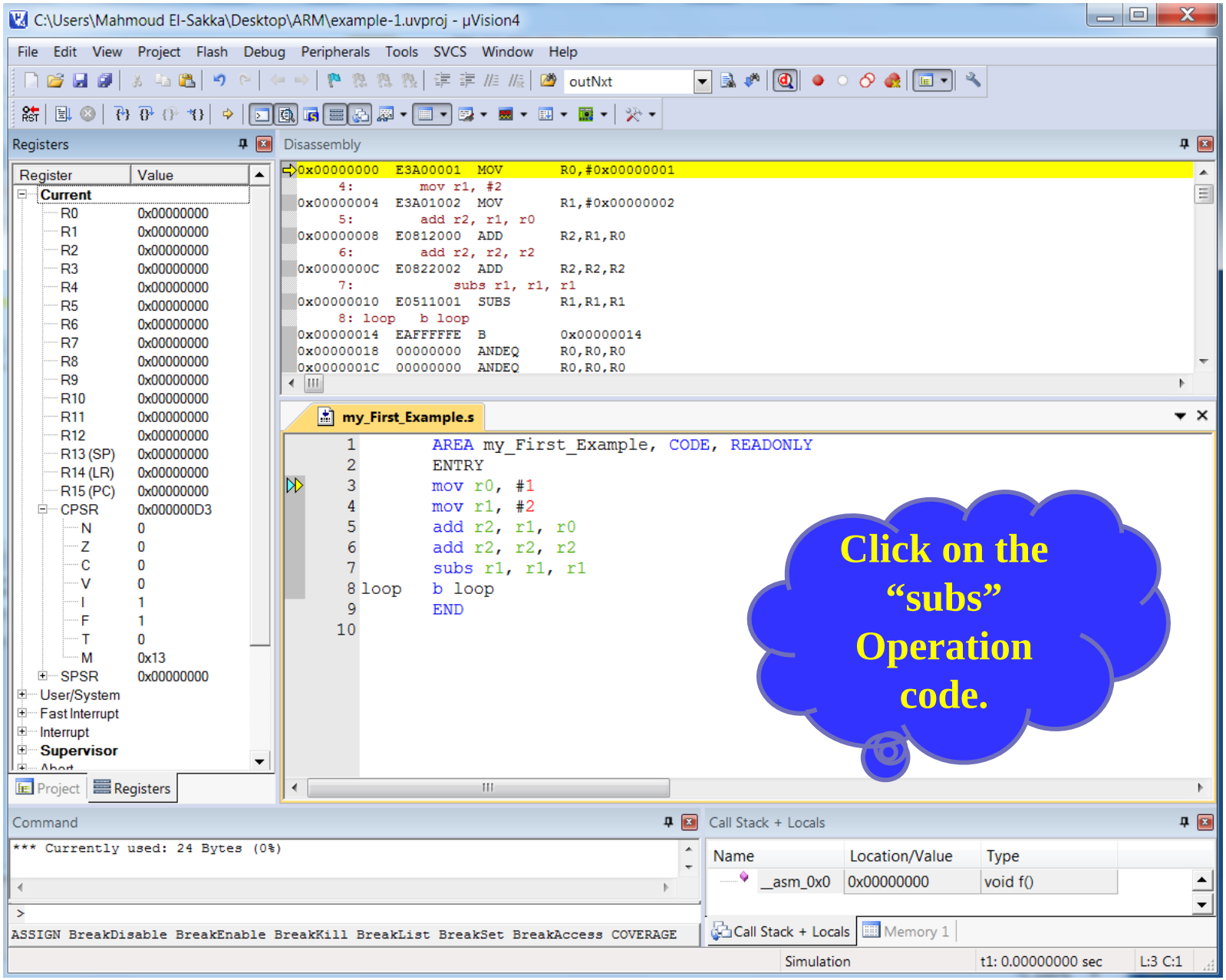
Using the Simulator—*step-by-step*



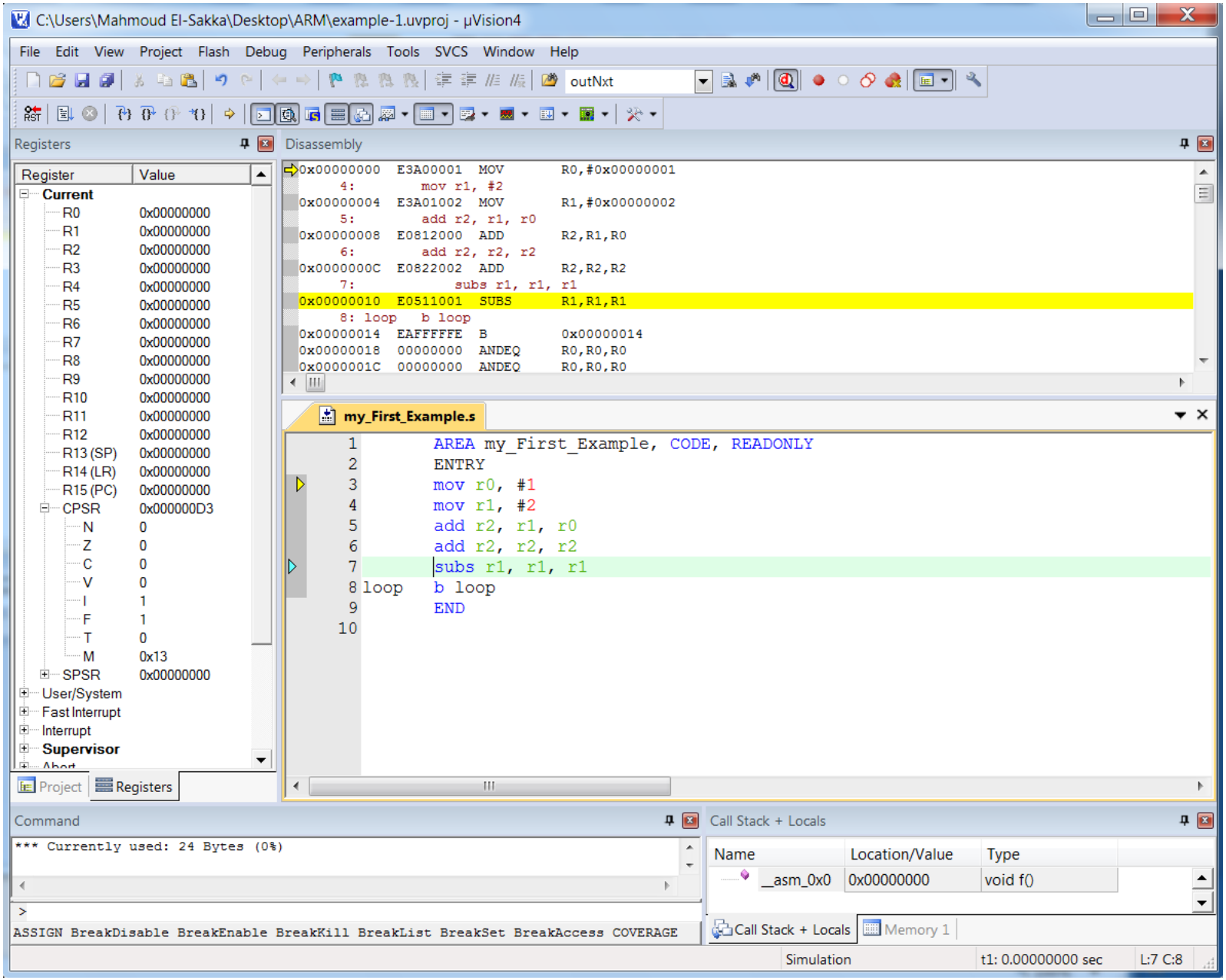
Using the Simulator—*step-by-step*



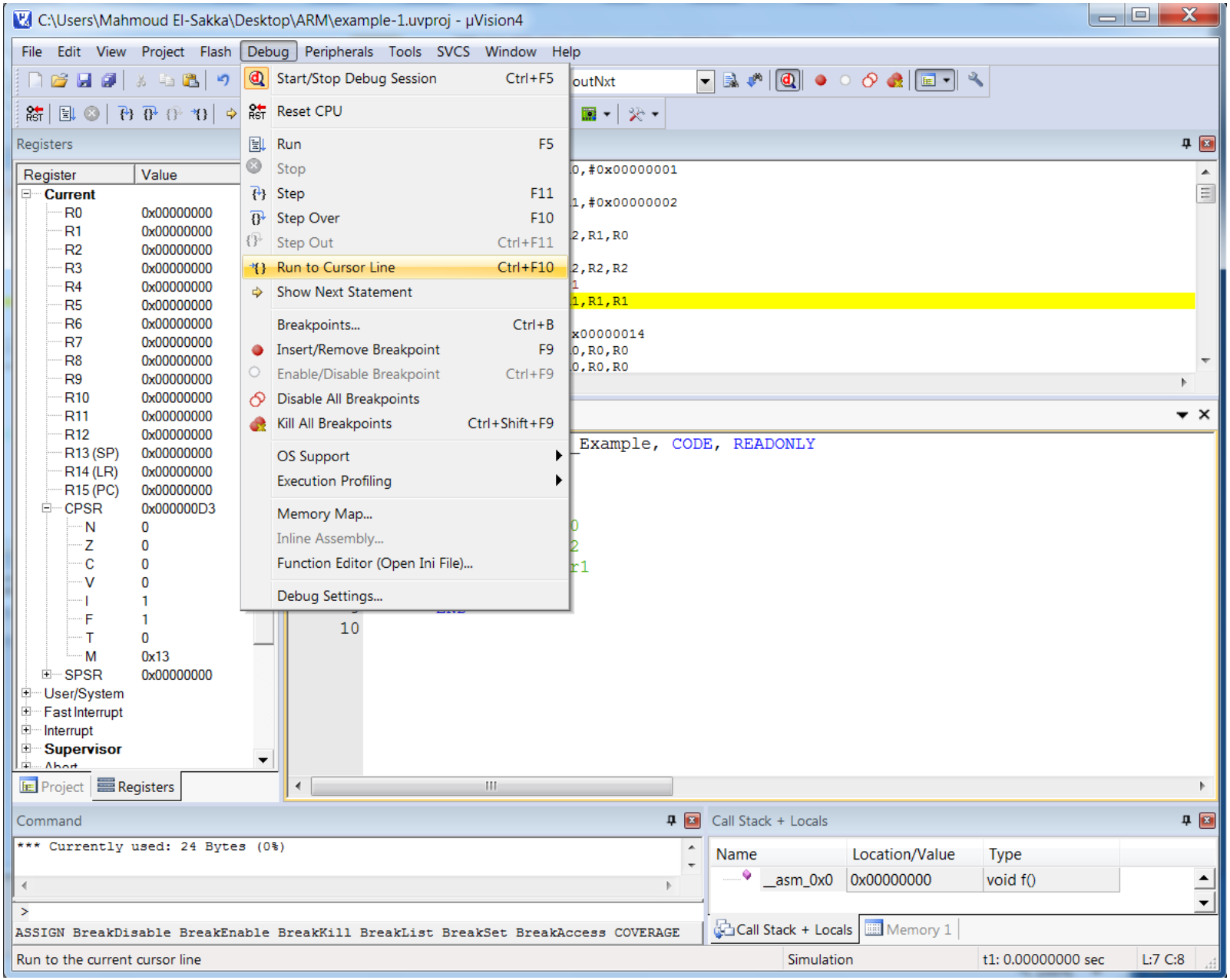
Using the Simulator—*step-by-step*



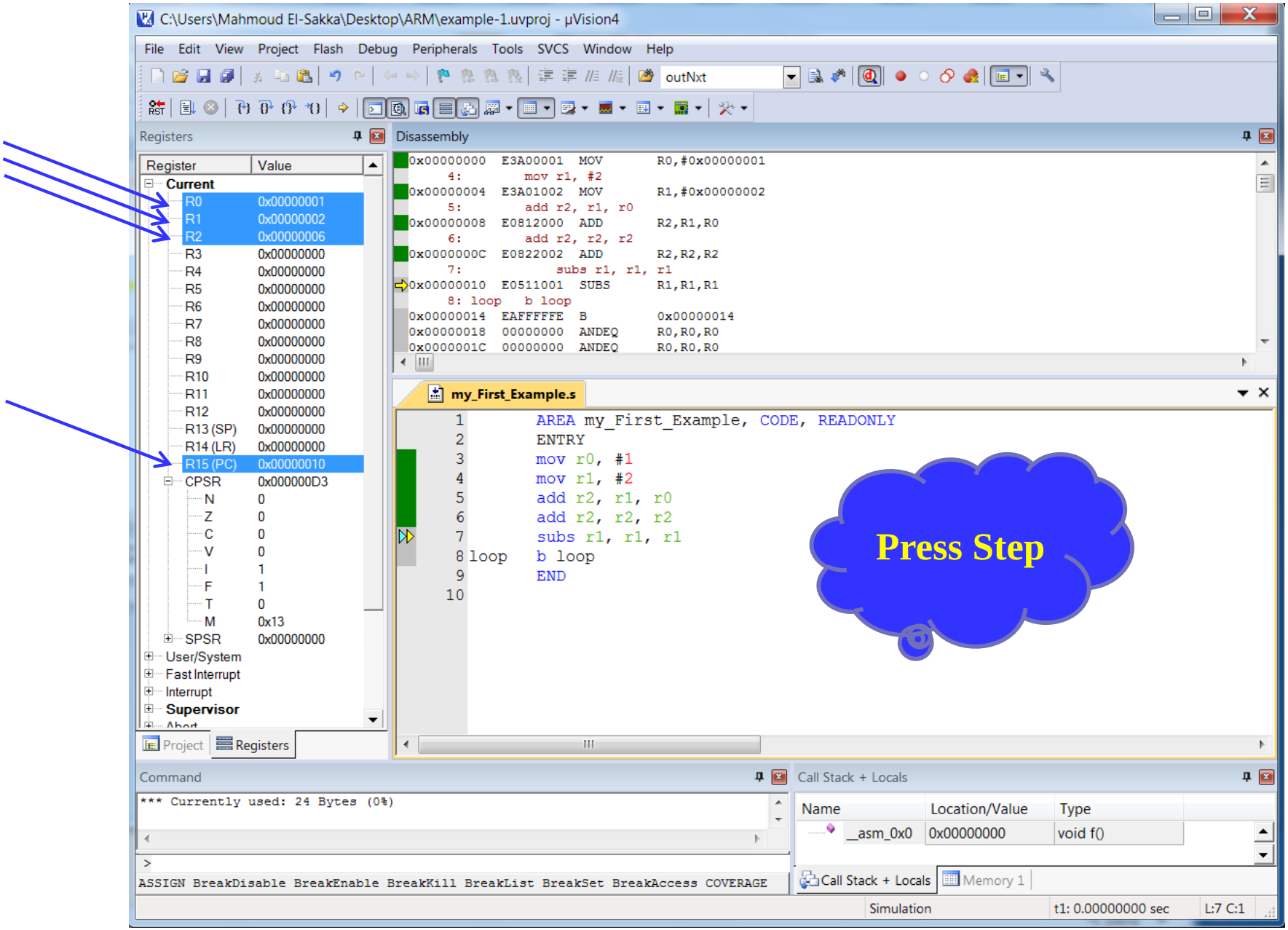
Using the Simulator—*step-by-step*



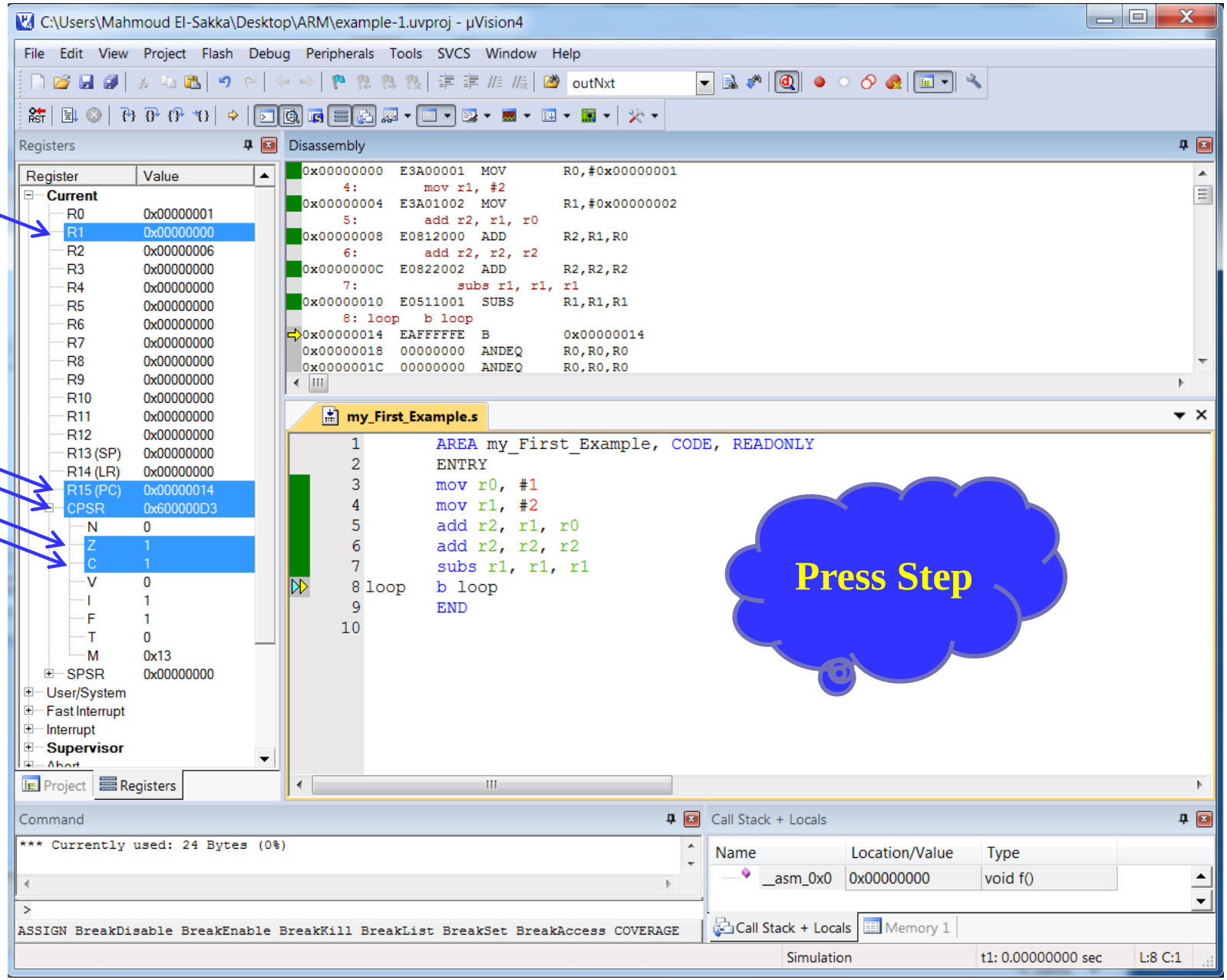
Using the Simulator—*step-by-step*



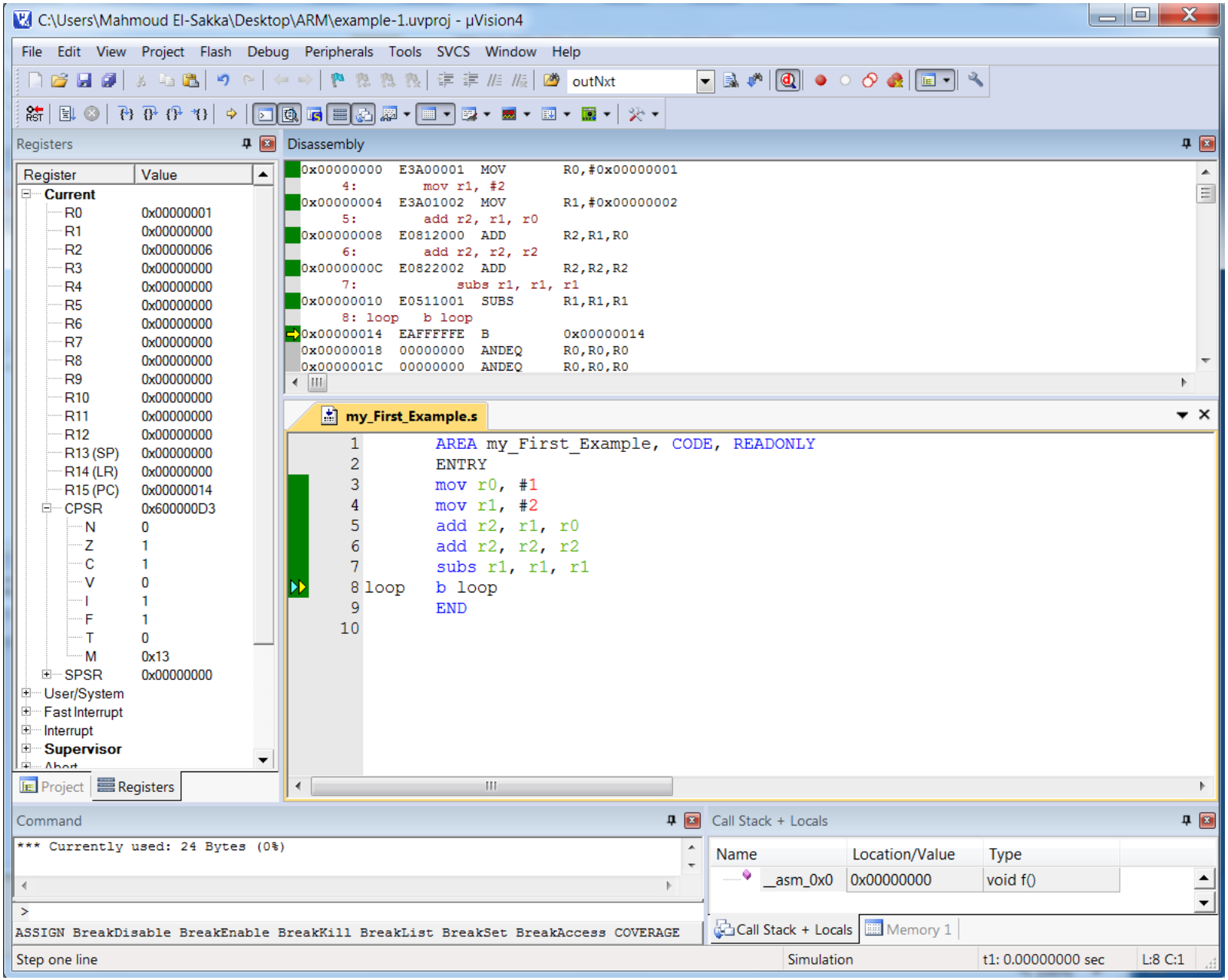
Using the Simulator—*step-by-step*



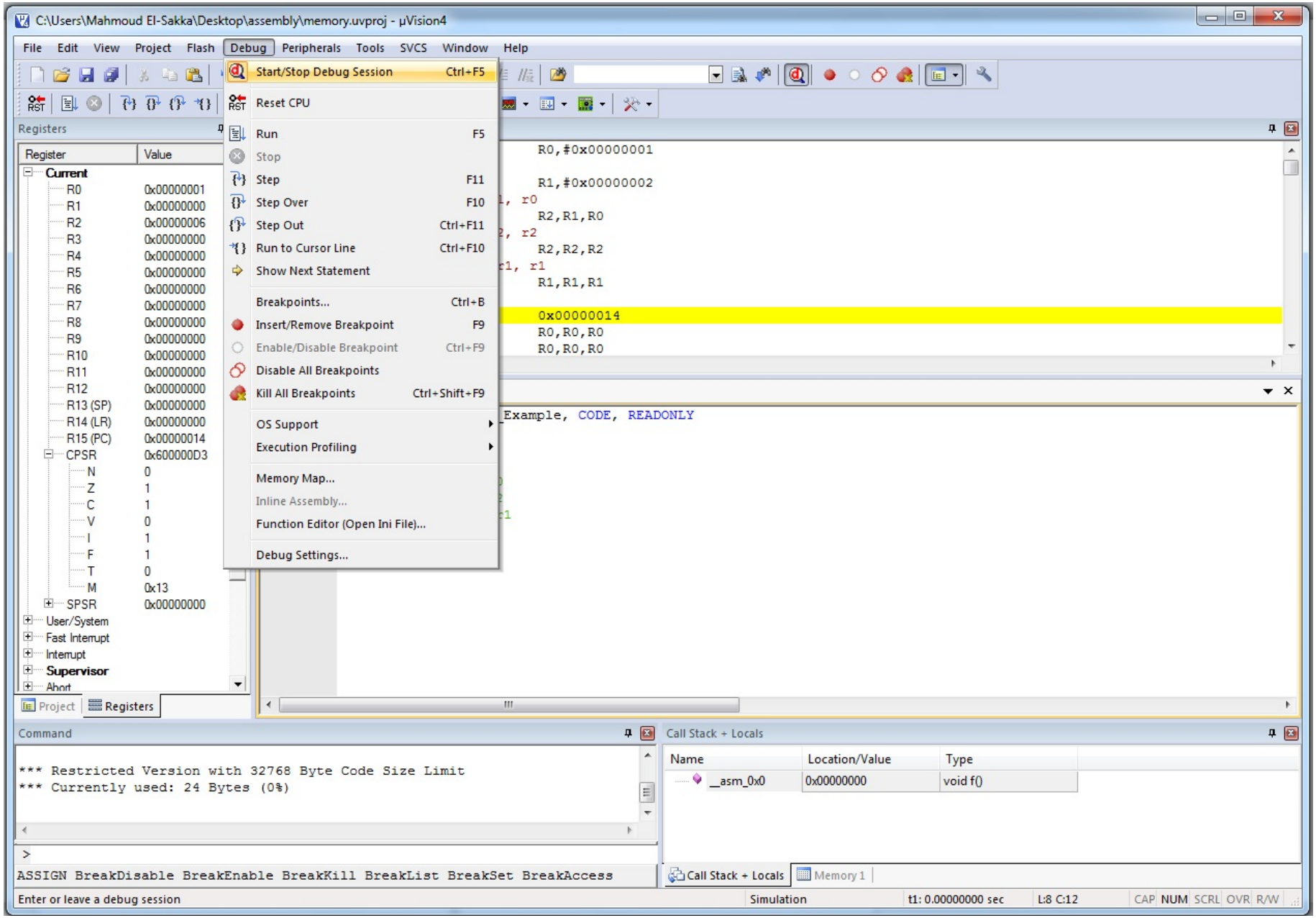
Using the Simulator—*step-by-step*



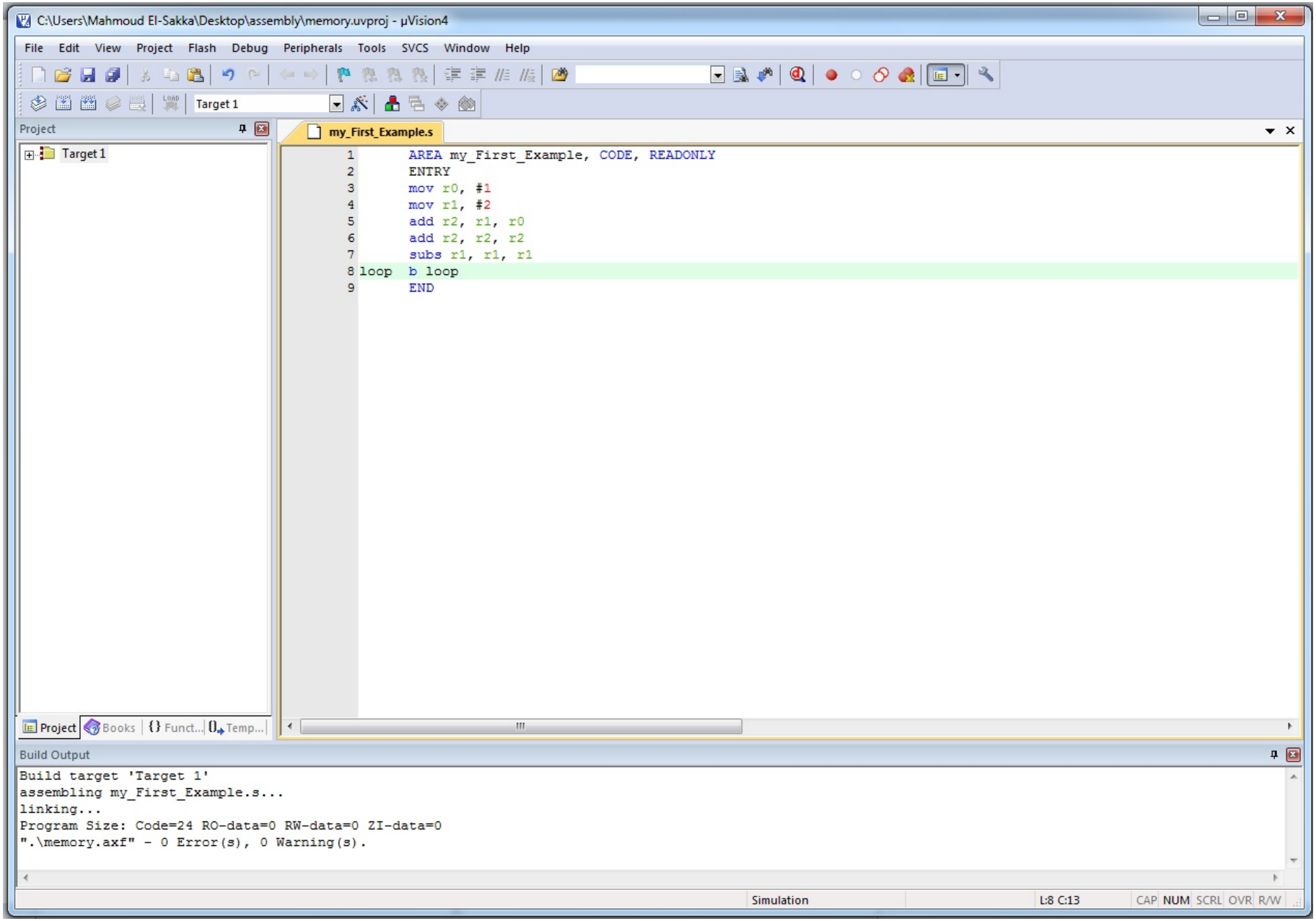
Using the Simulator—*step-by-step*



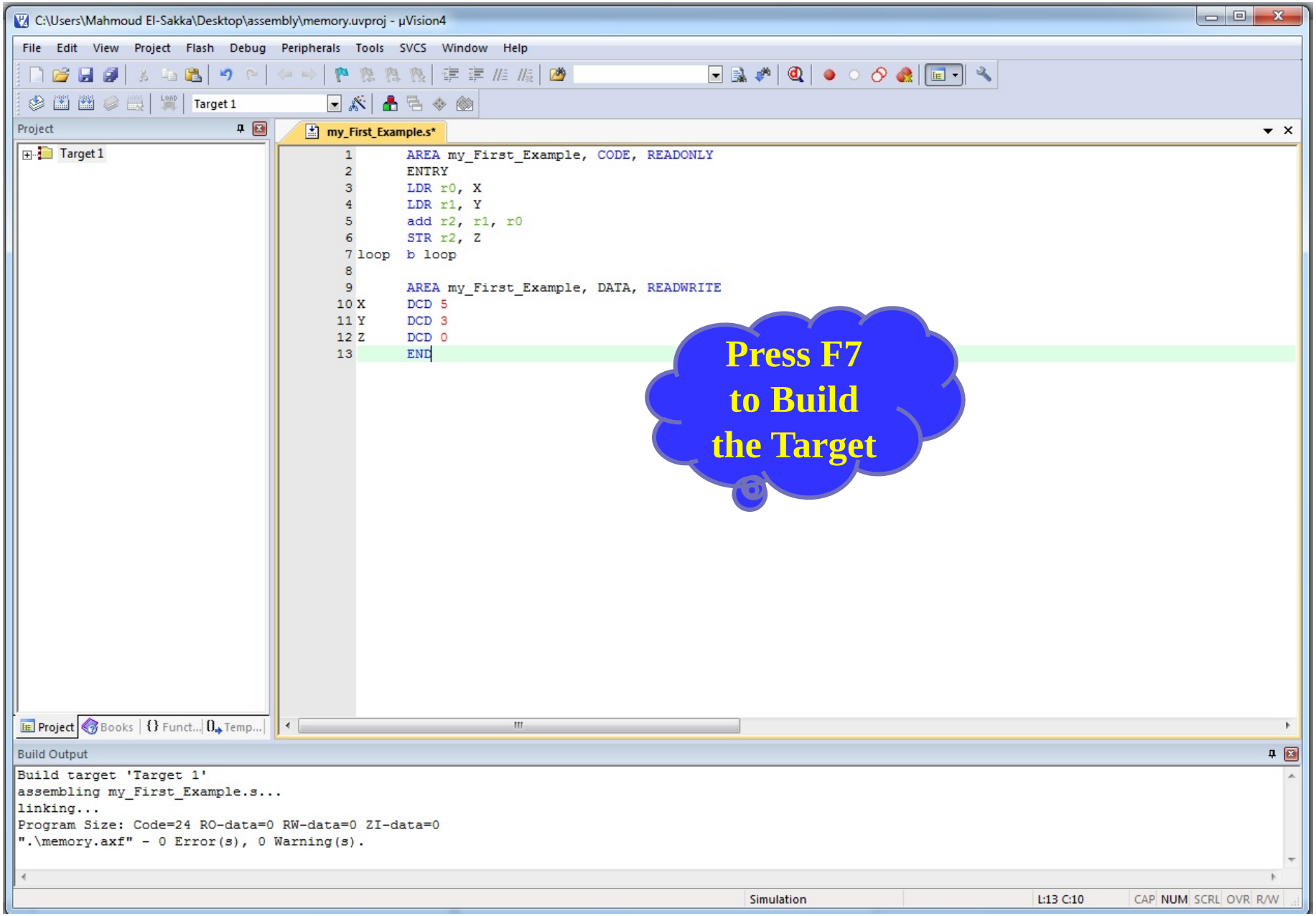
Using the Simulator—*step-by-step*



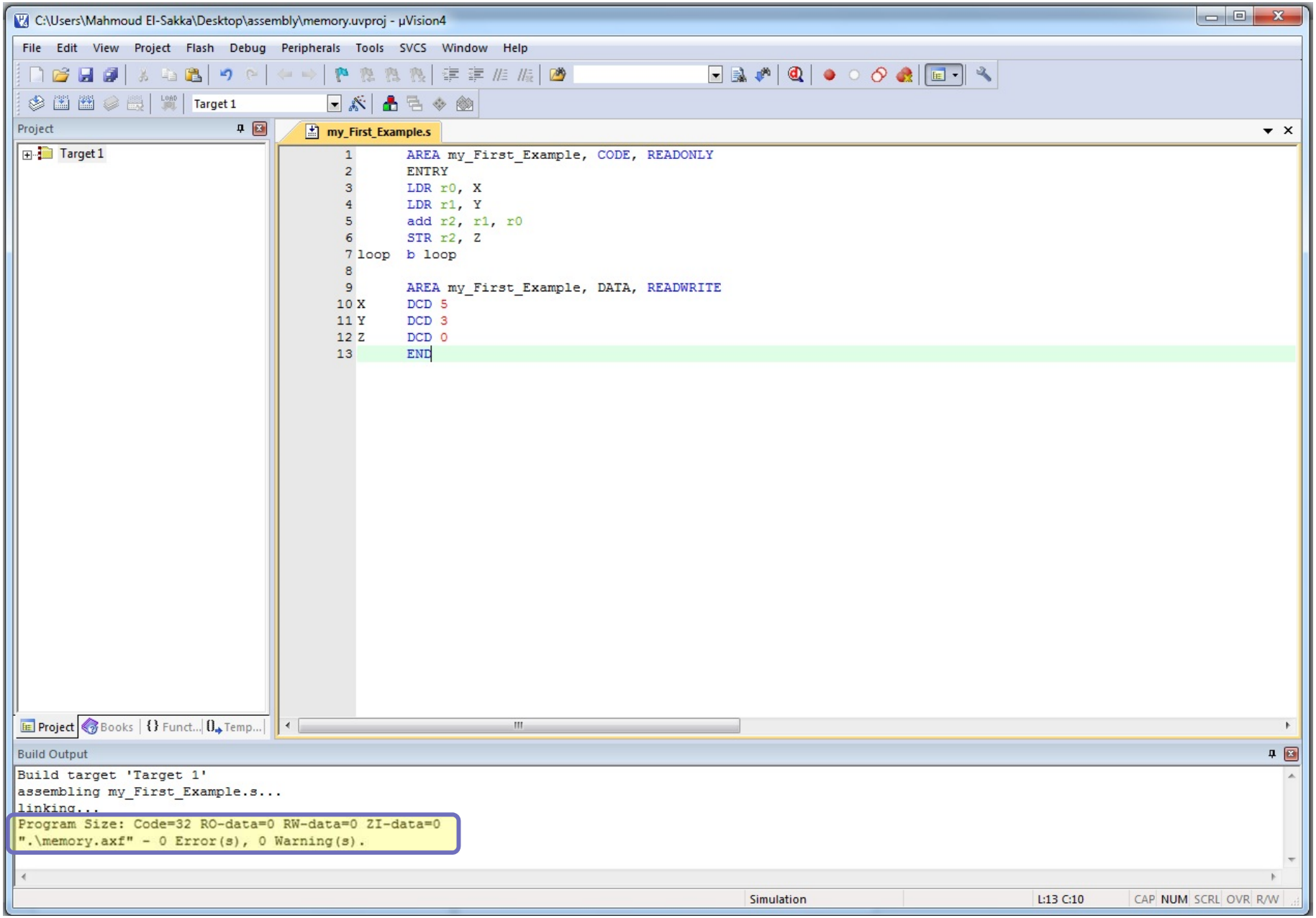
Using the Simulator—*step-by-step*



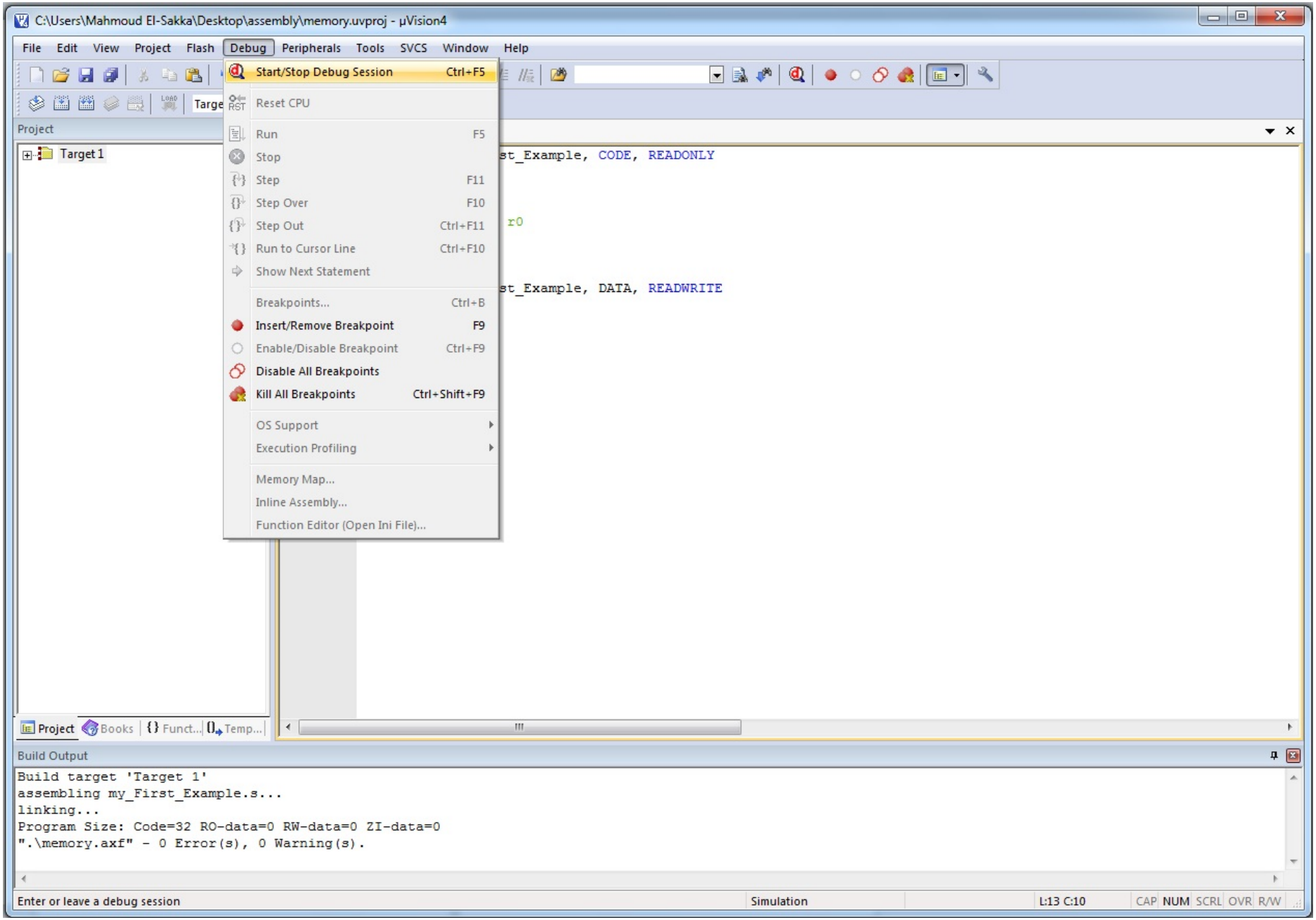
Using the Simulator—*step-by-step*



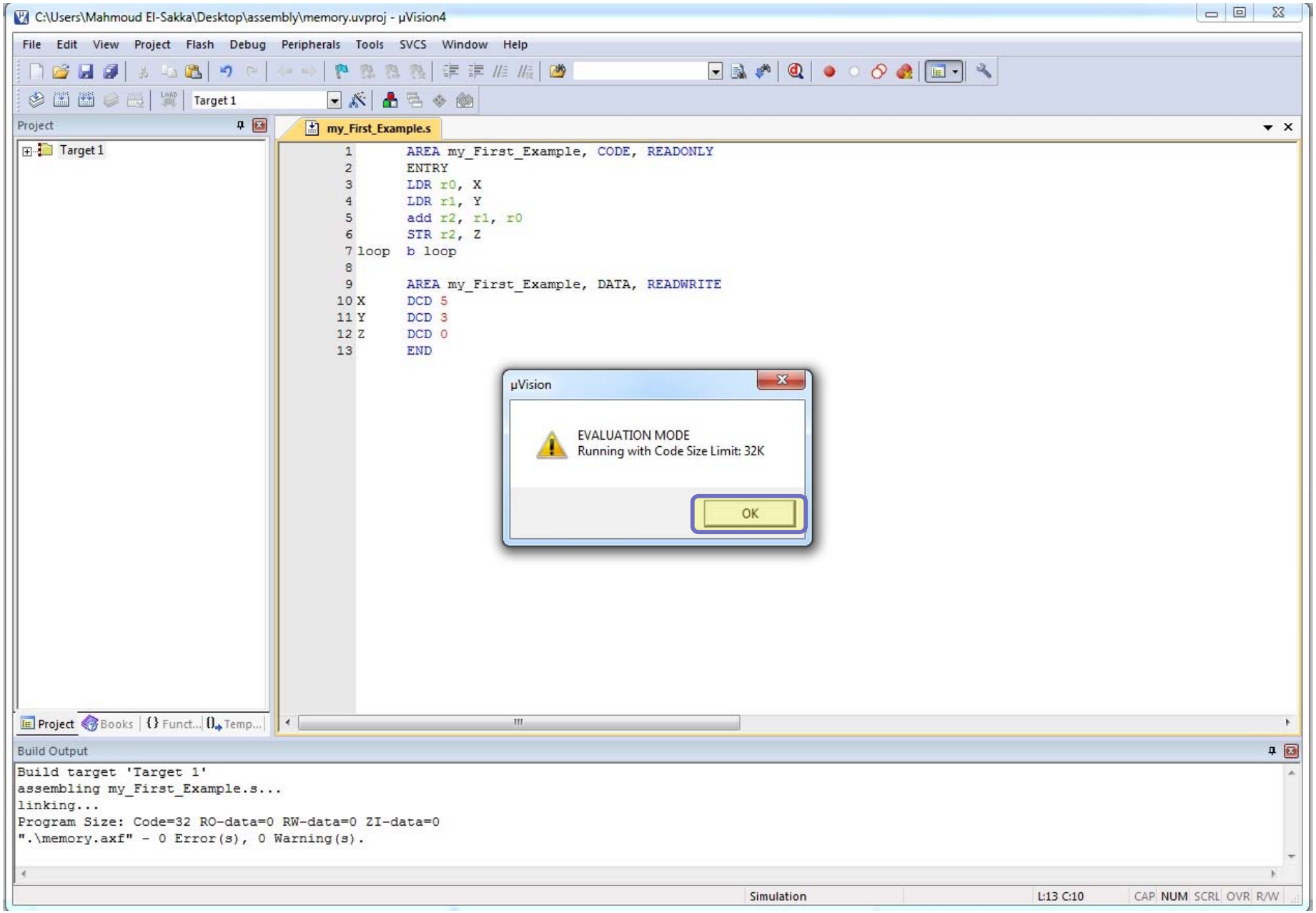
Using the Simulator—*step-by-step*



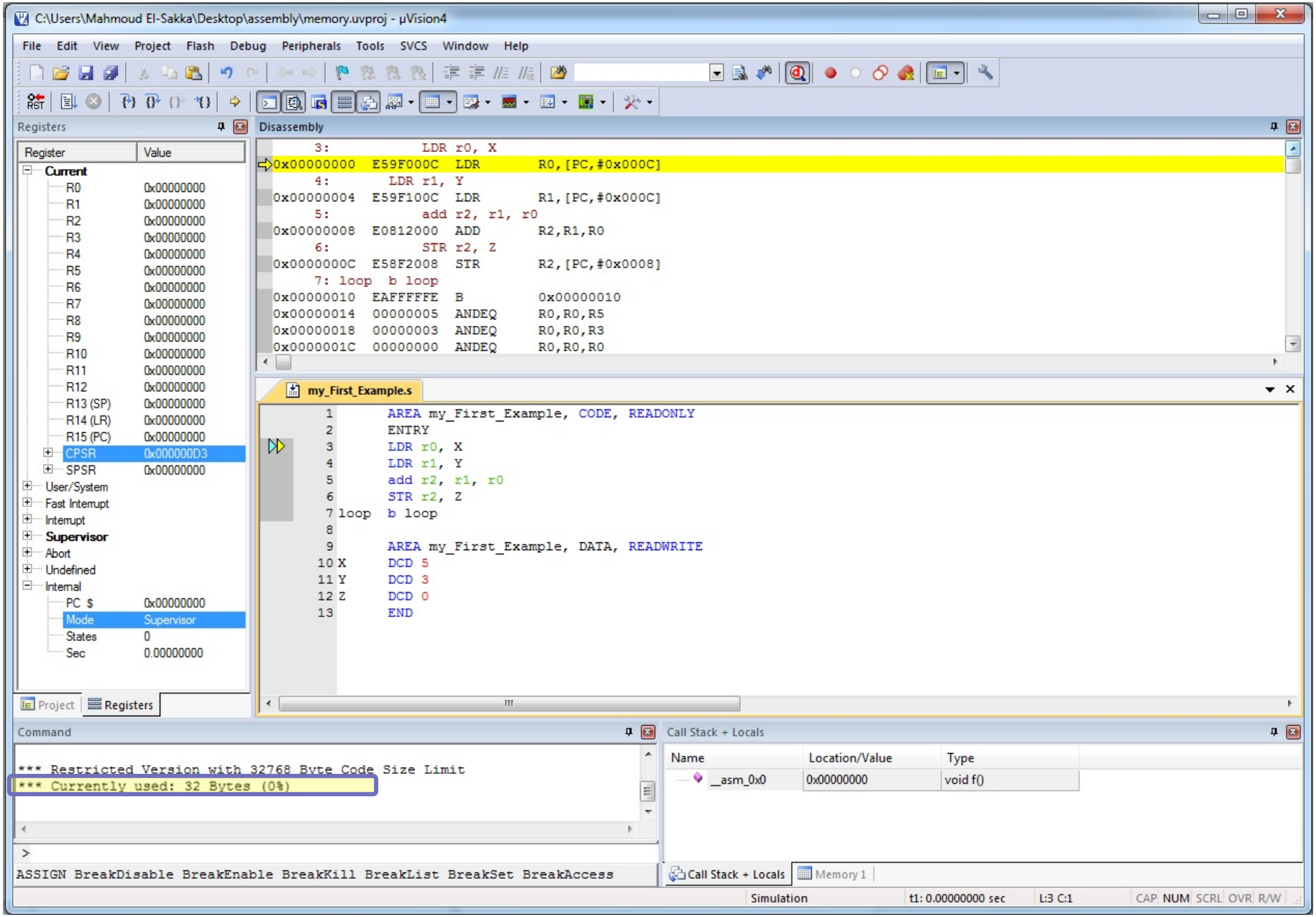
Using the Simulator—*step-by-step*



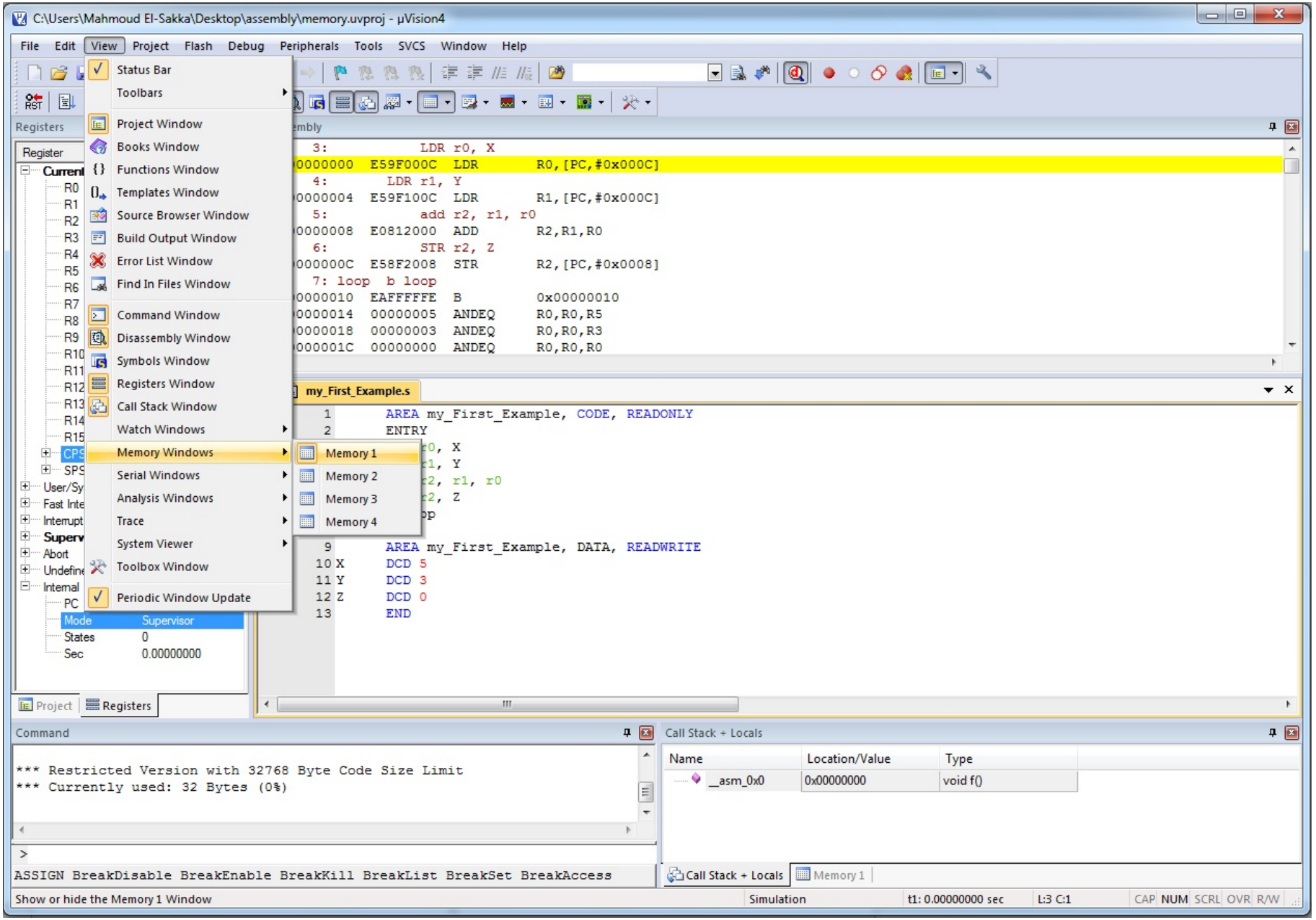
Using the Simulator—*step-by-step*



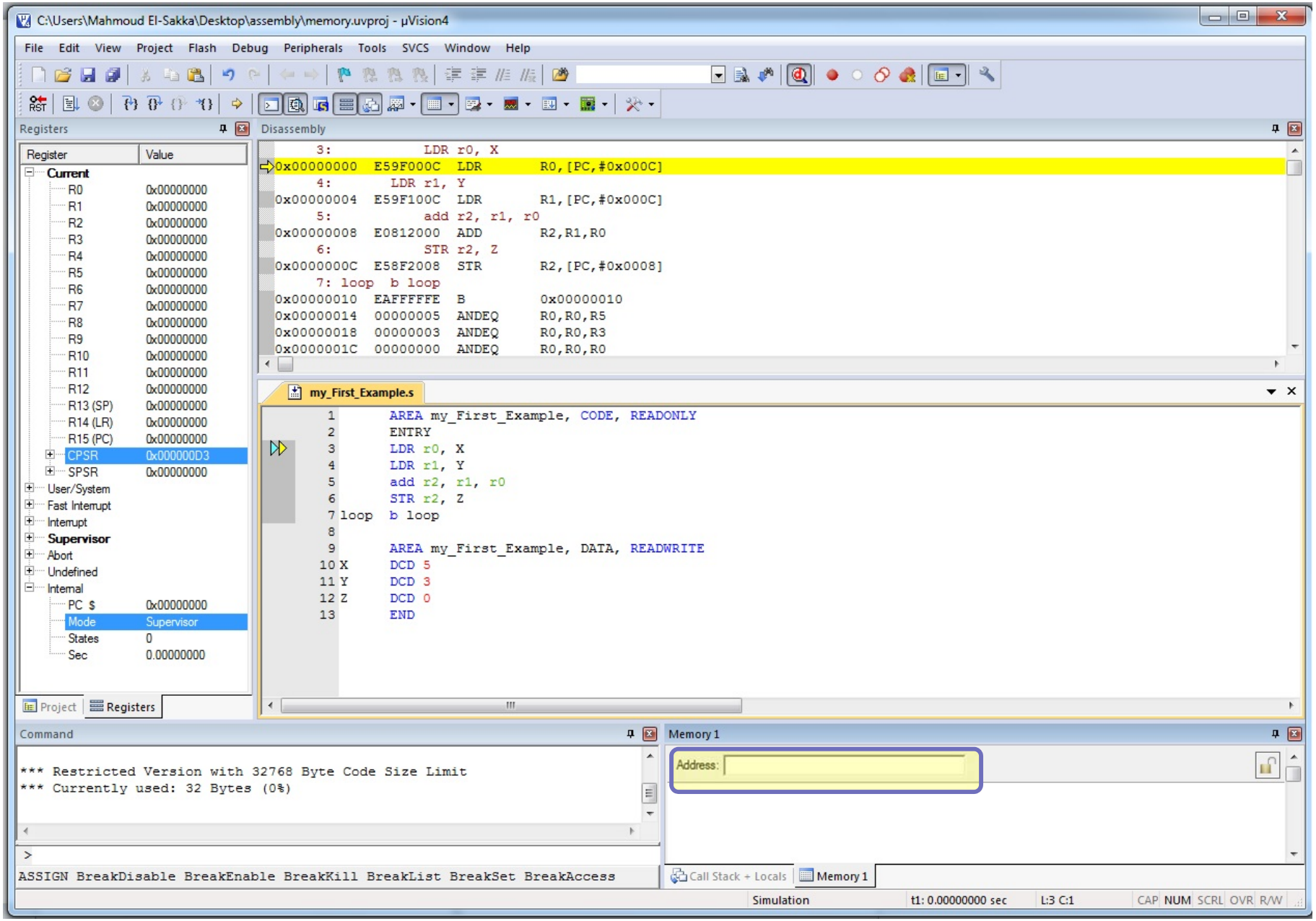
Using the Simulator—*step-by-step*



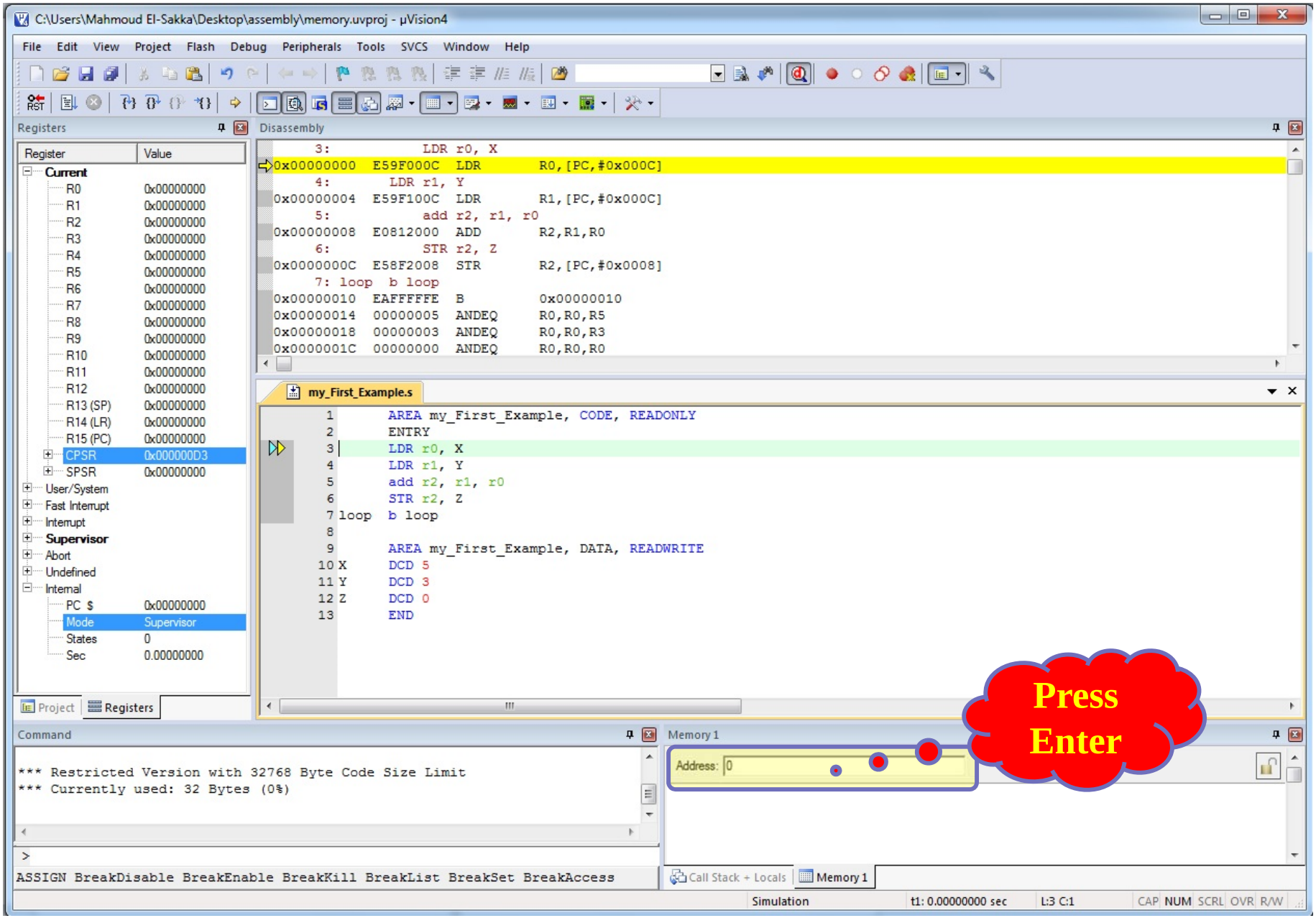
Using the Simulator—*step-by-step*



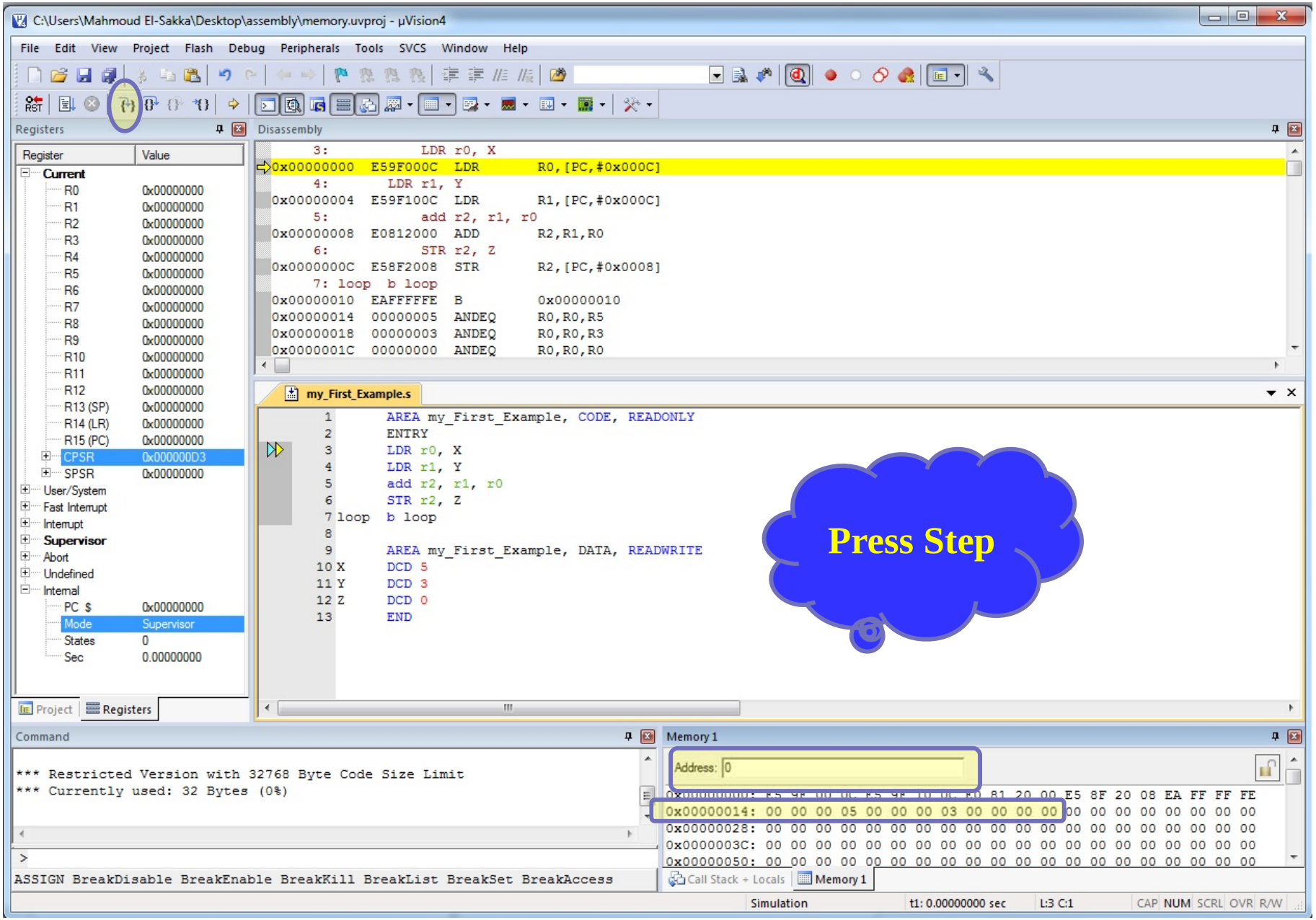
Using the Simulator—*step-by-step*



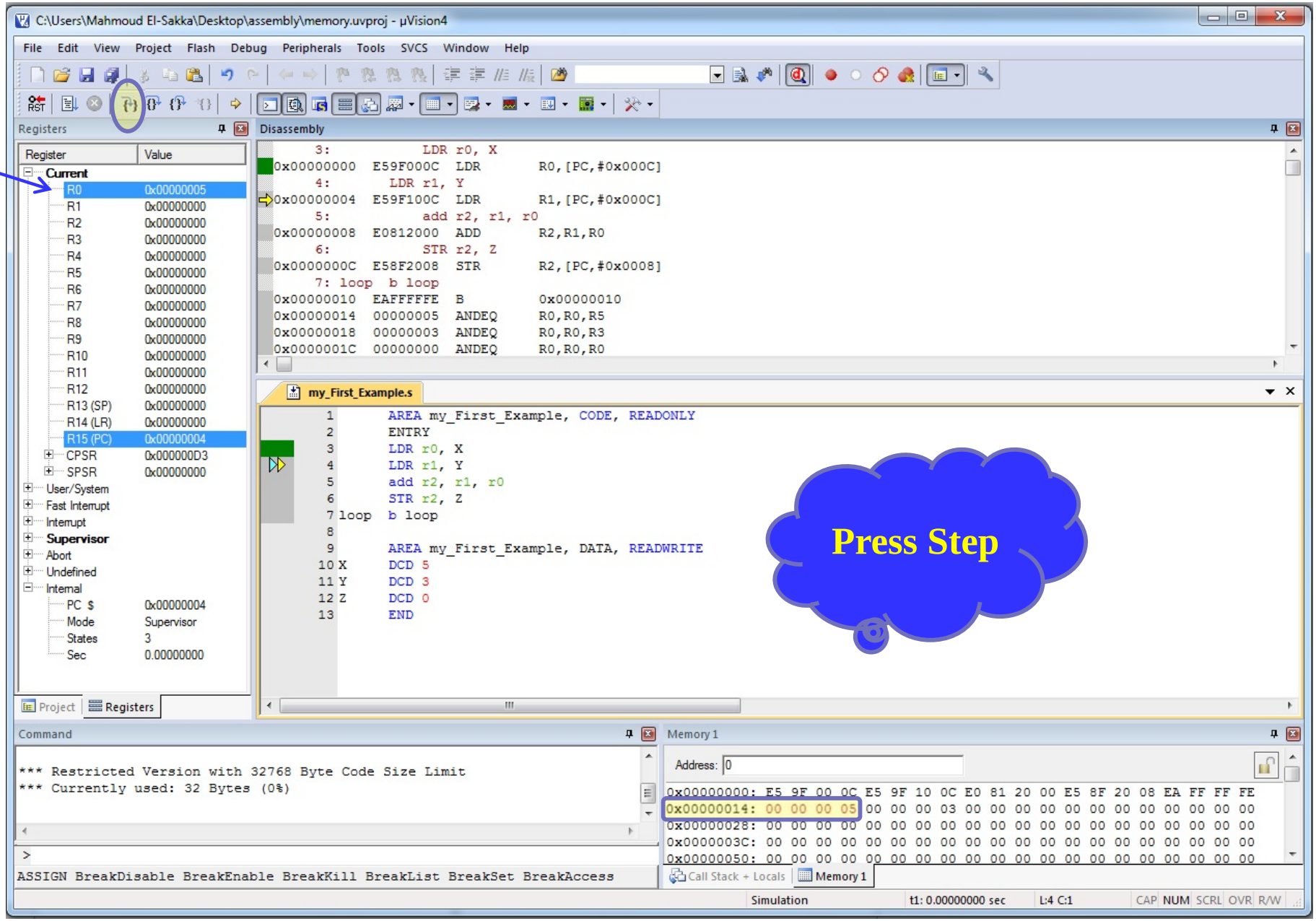
Using the Simulator—*step-by-step*



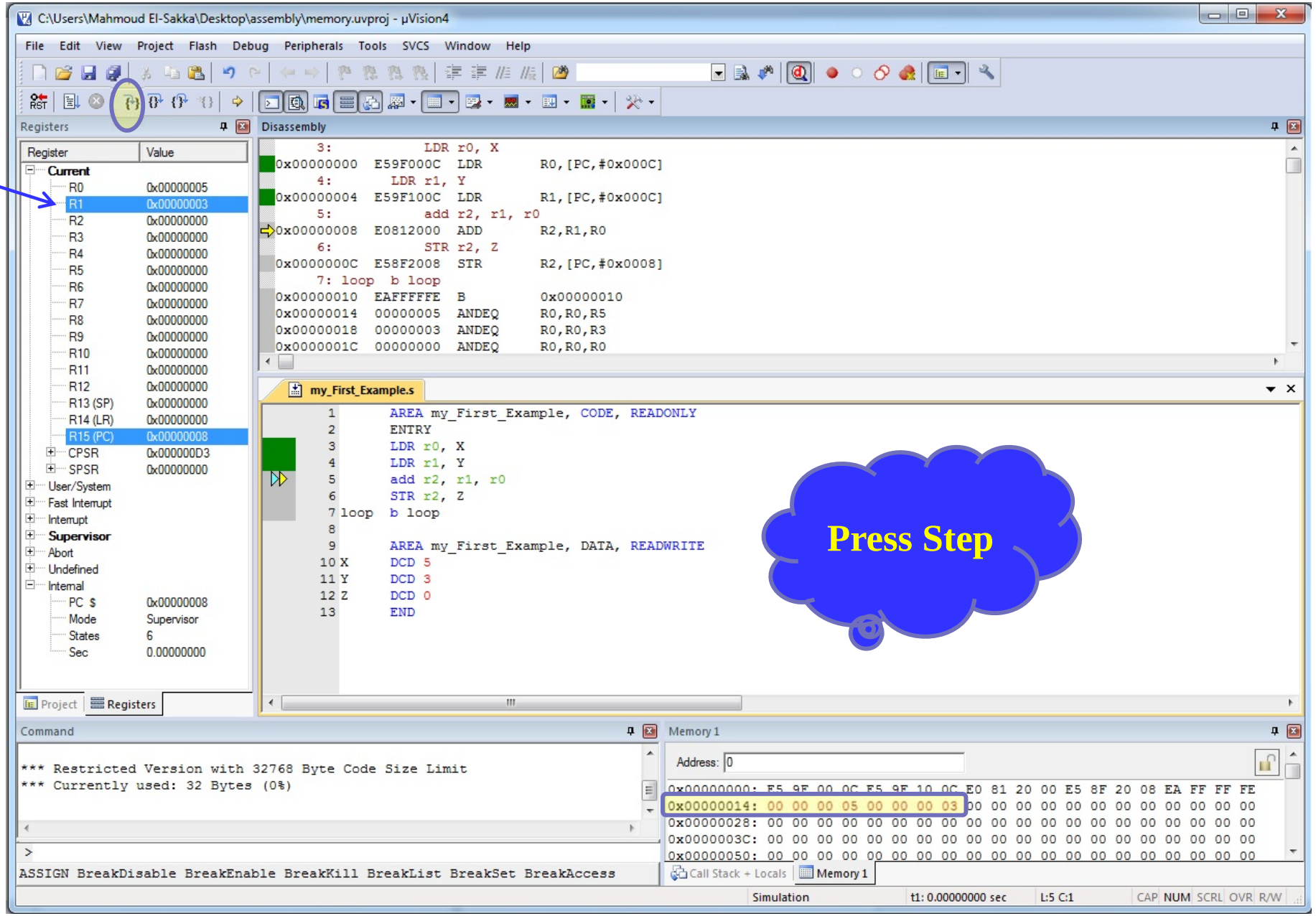
Using the Simulator—*step-by-step*



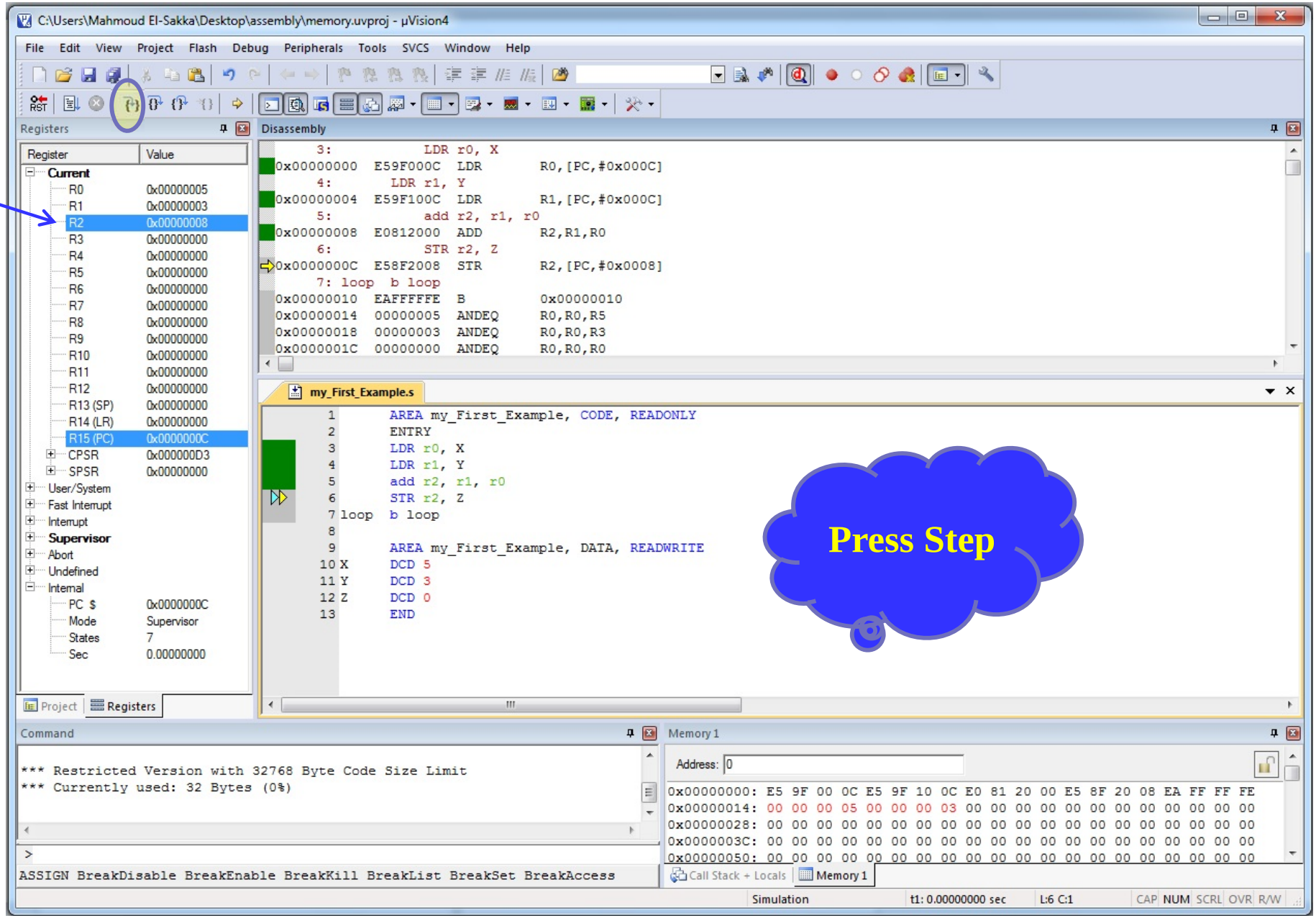
Using the Simulator—*step-by-step*



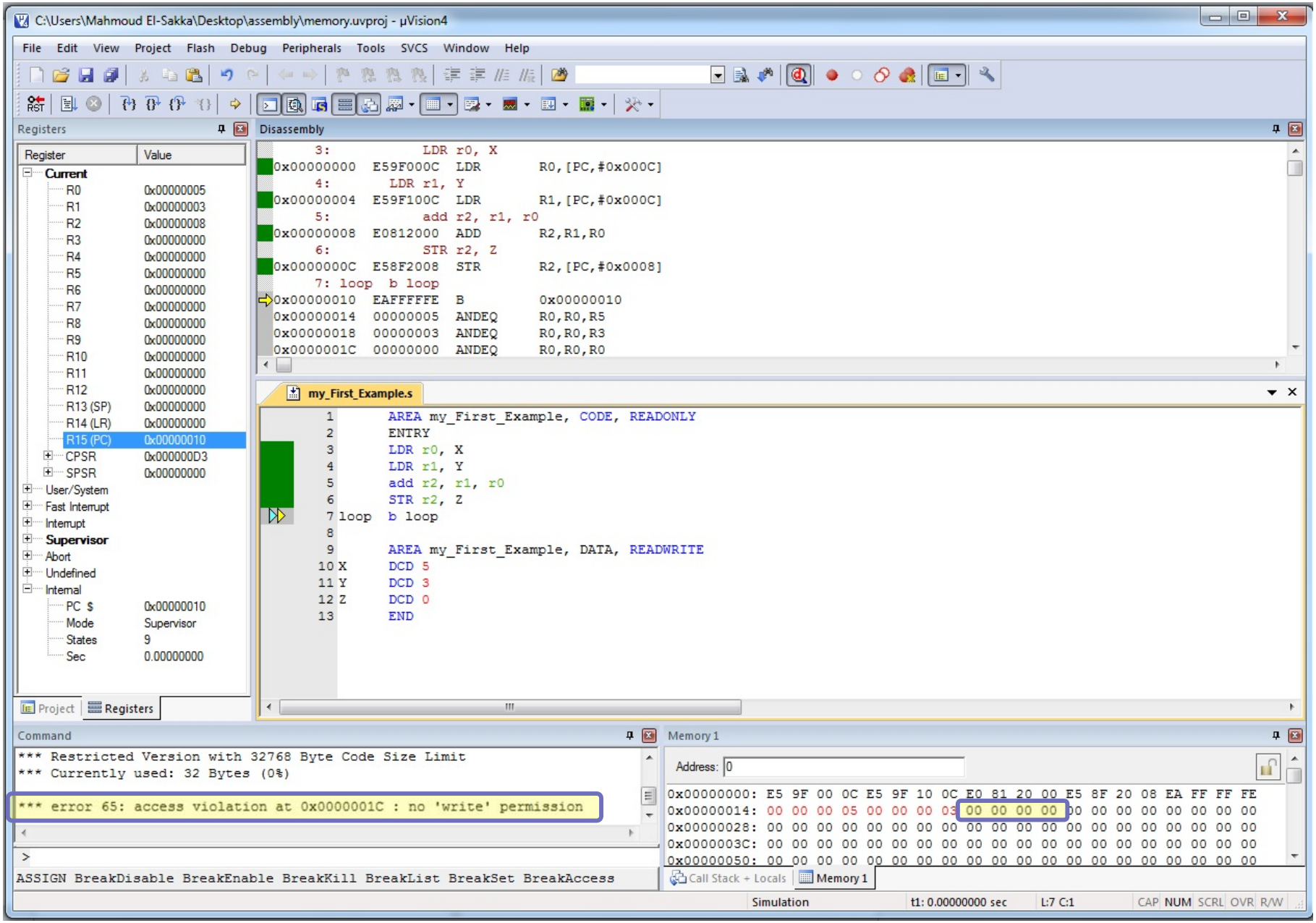
Using the Simulator—*step-by-step*



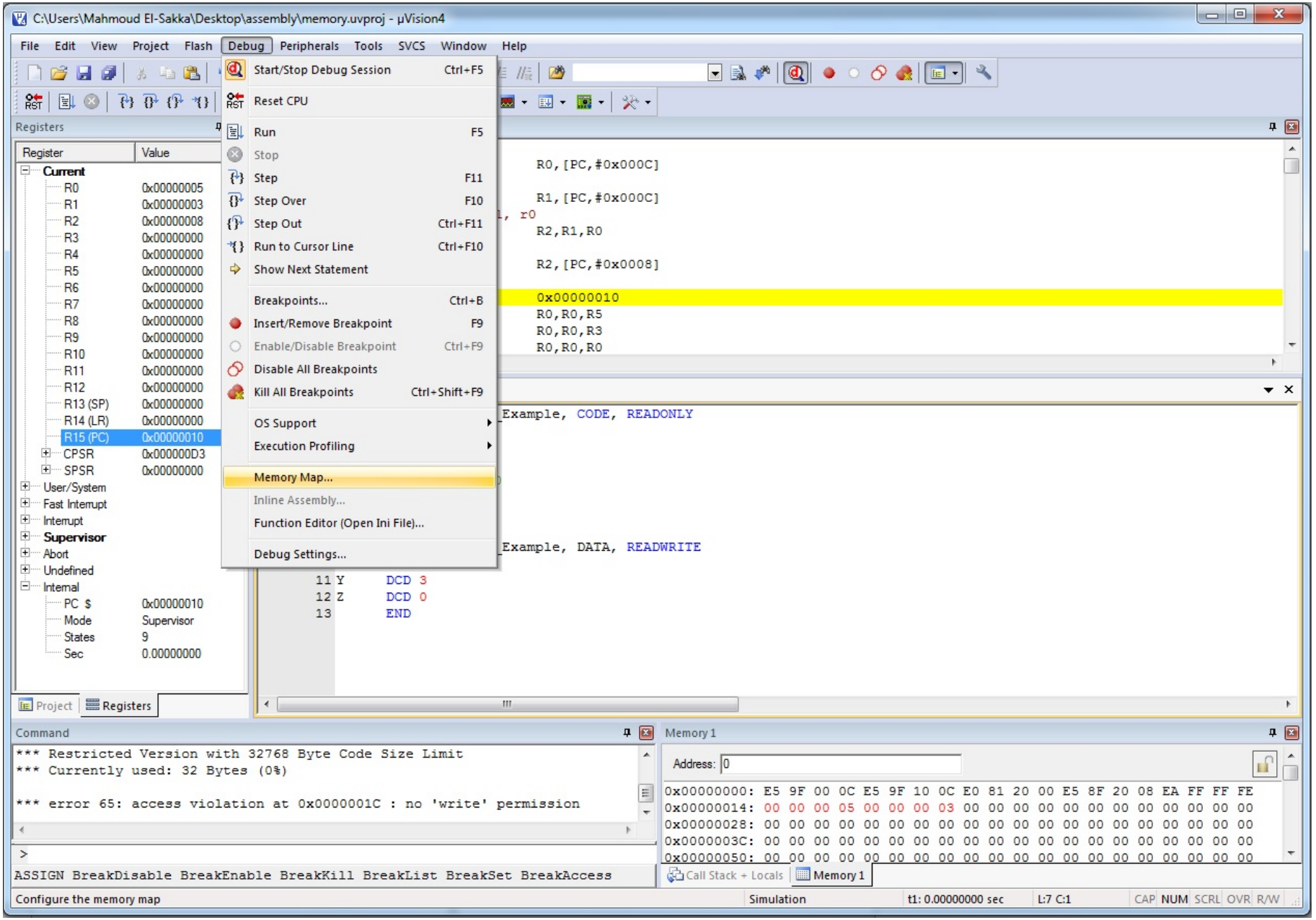
Using the Simulator—*step-by-step*



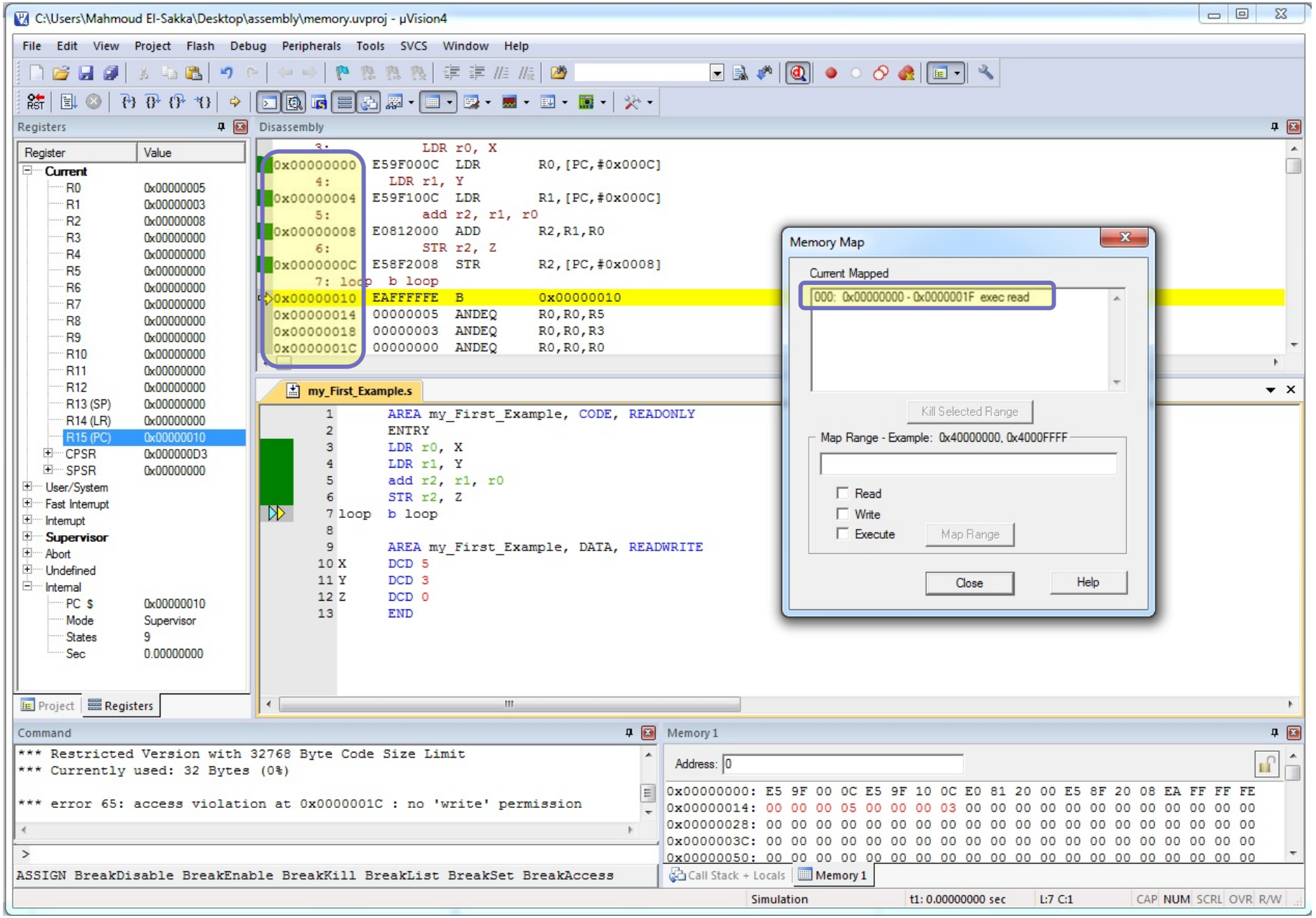
Using the Simulator—*step-by-step*



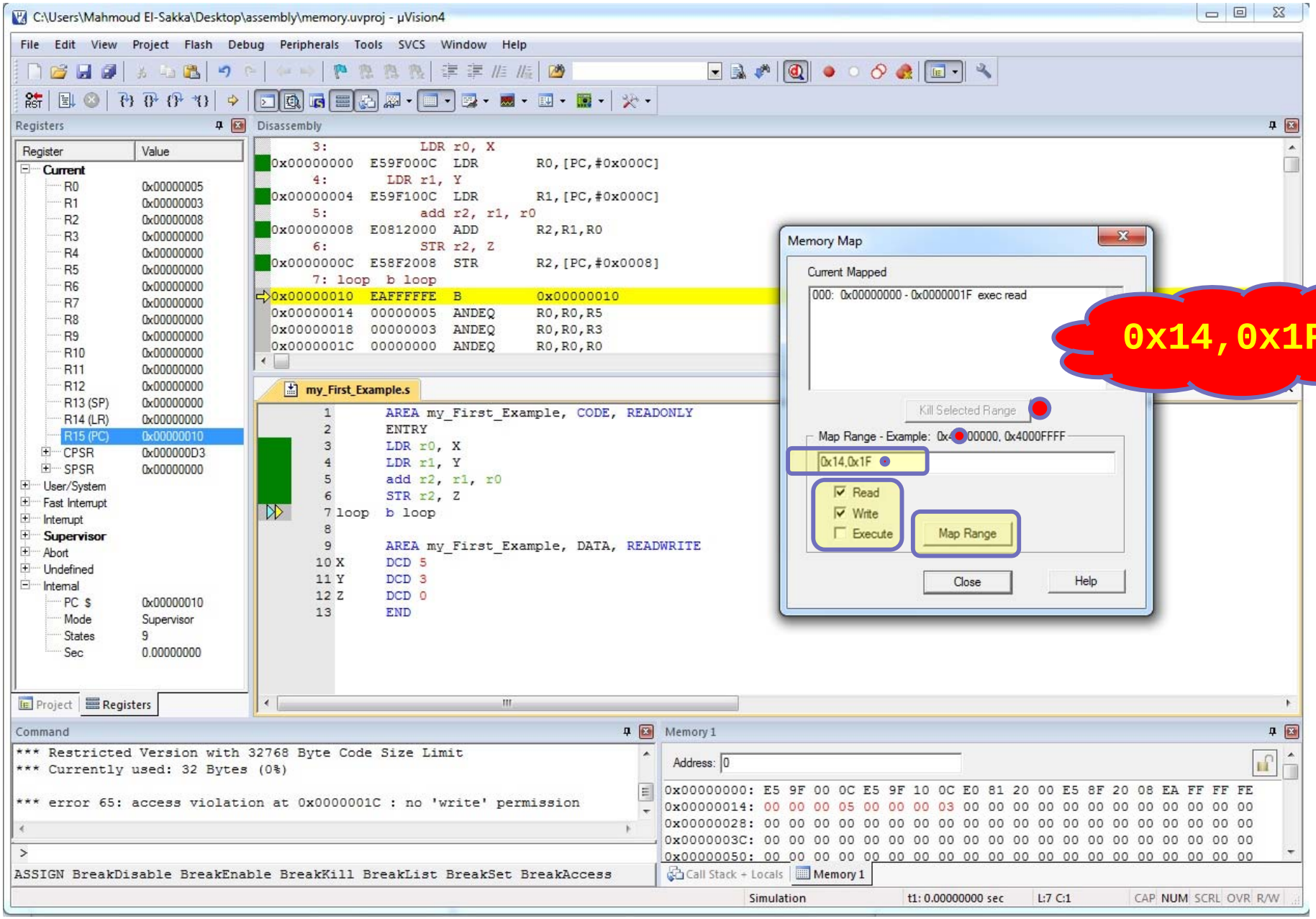
Using the Simulator—*step-by-step*



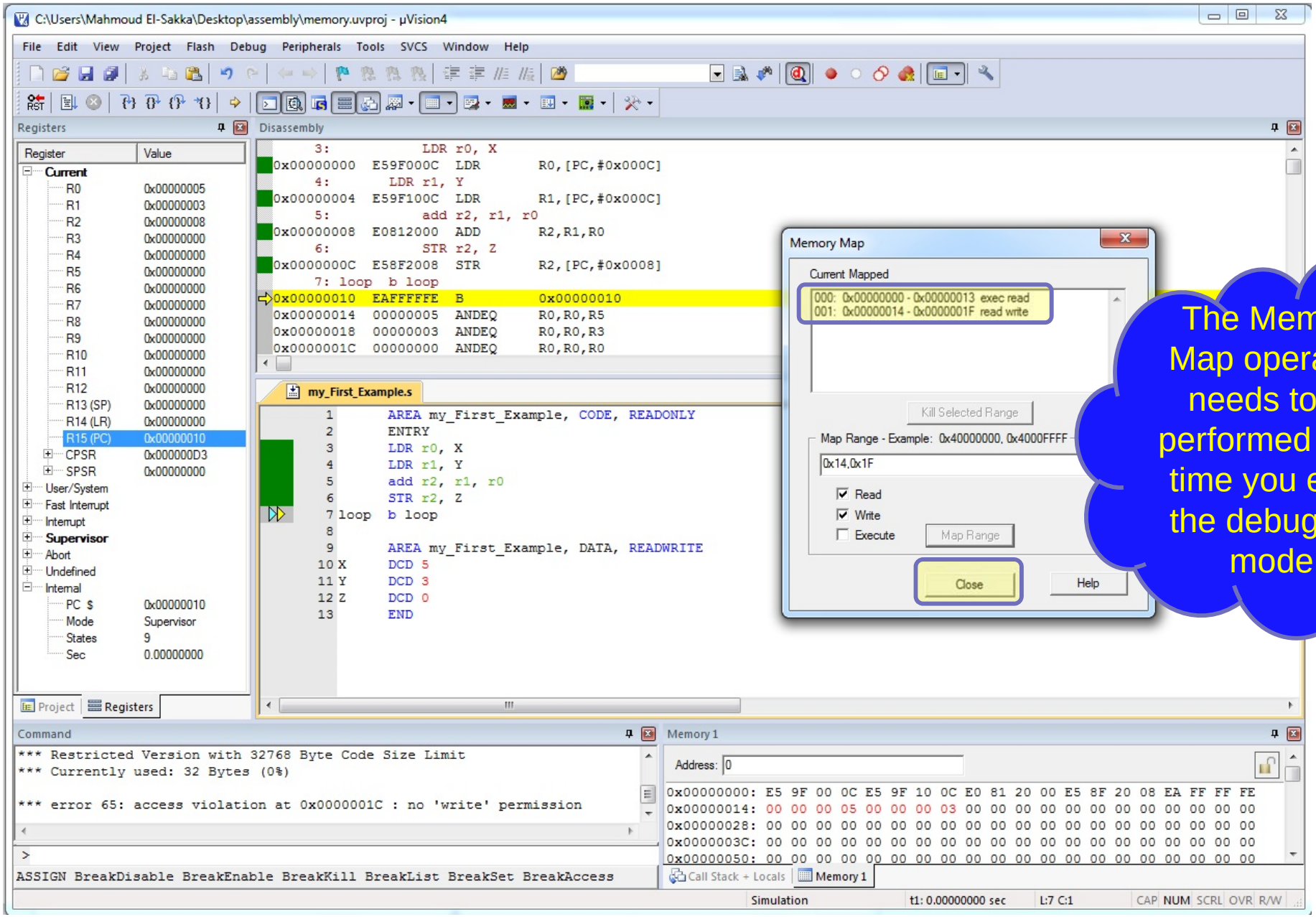
Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*

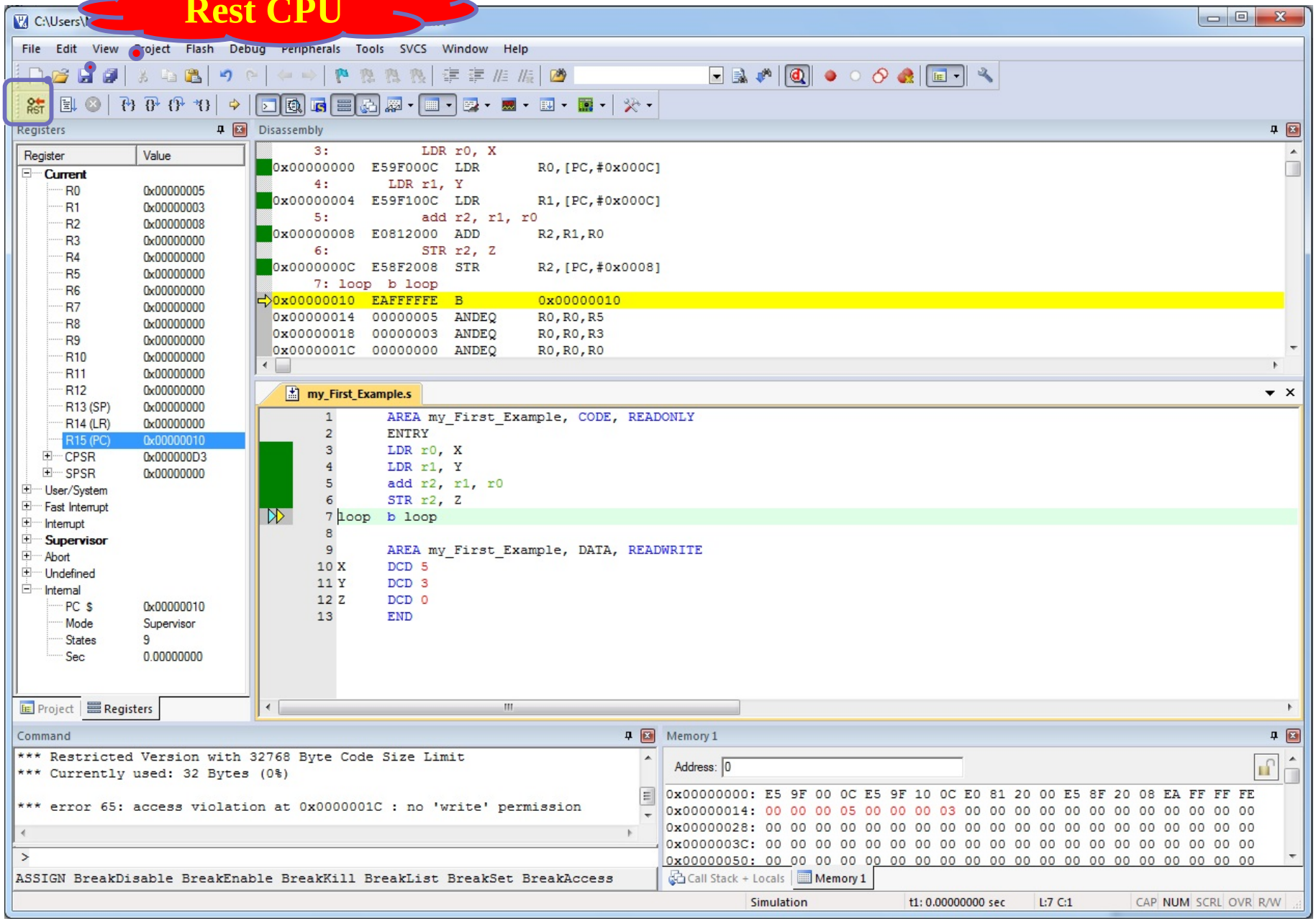


Using the Simulator—*step-by-step*

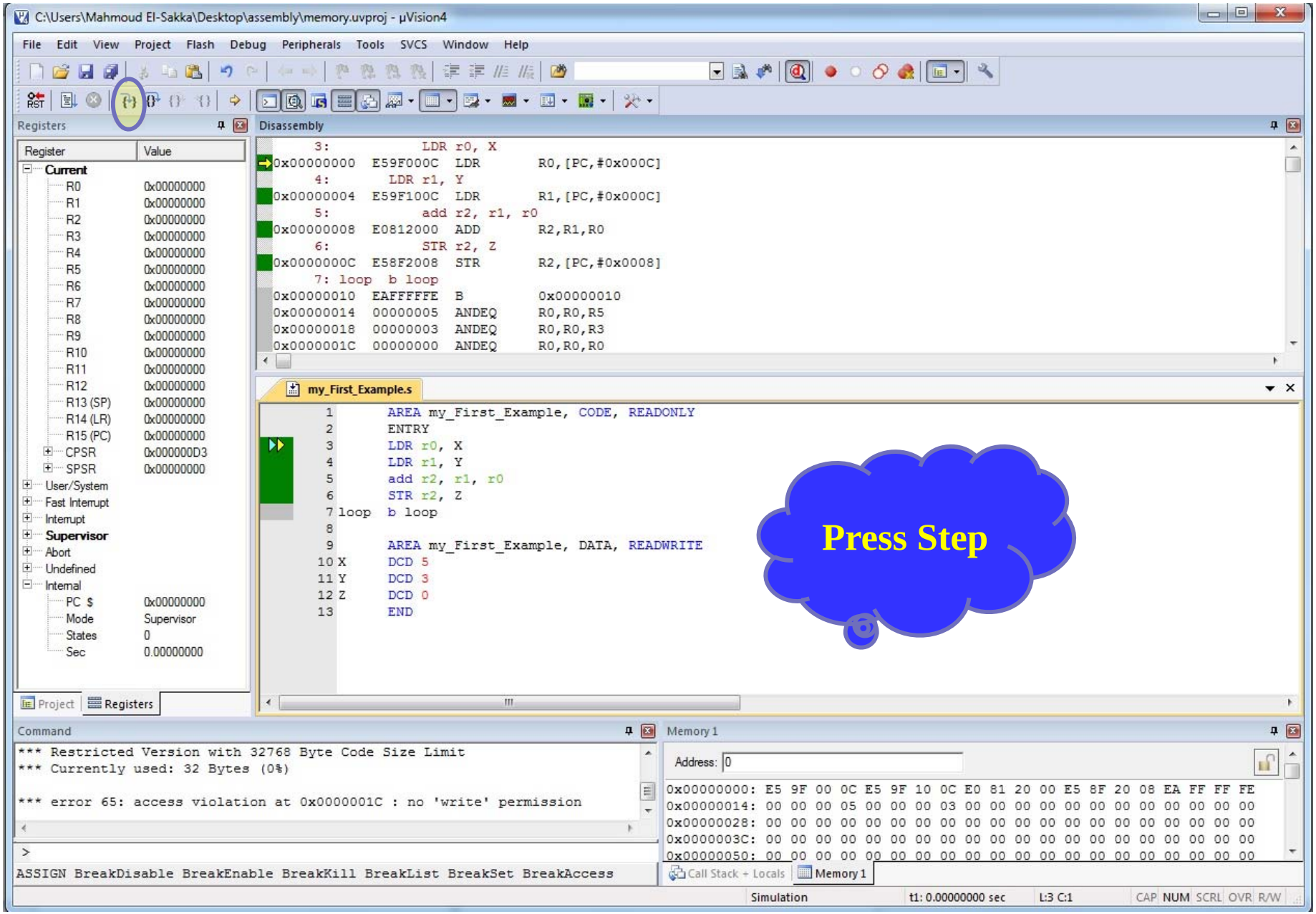


The Memory Map operation needs to be performed each time you enter the debugging mode.

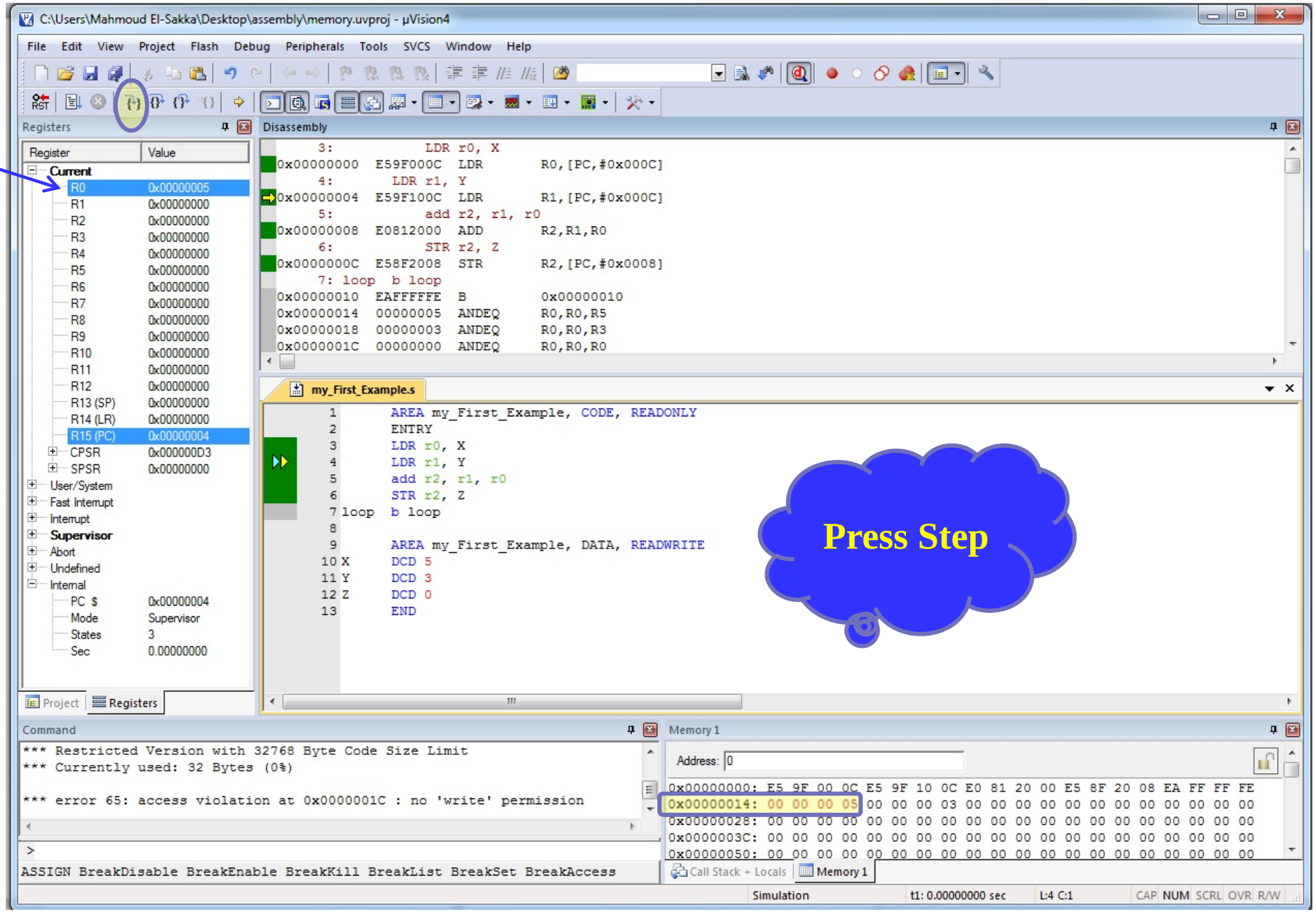
Using the Simulator—*step-by-step*



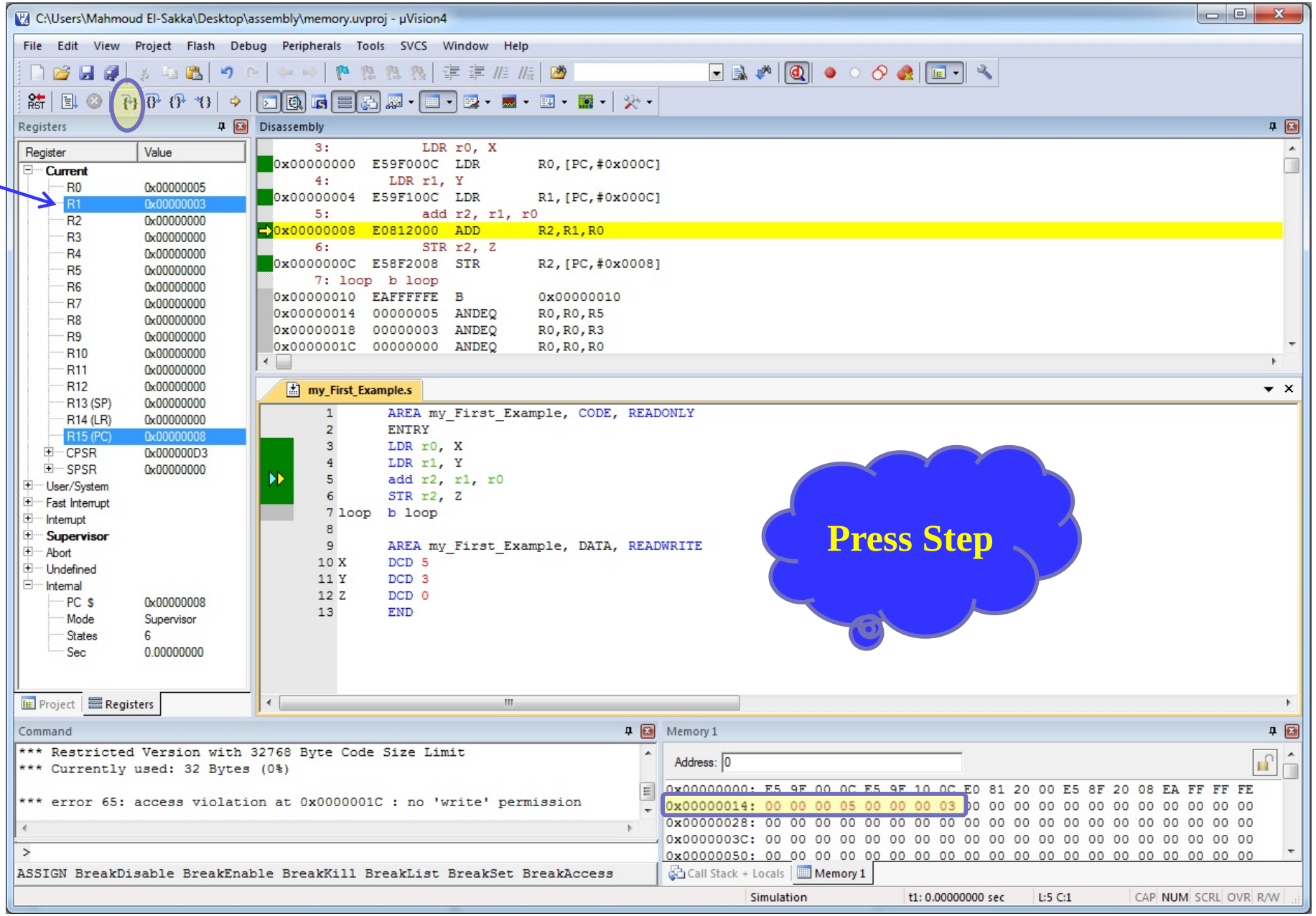
Using the Simulator—*step-by-step*



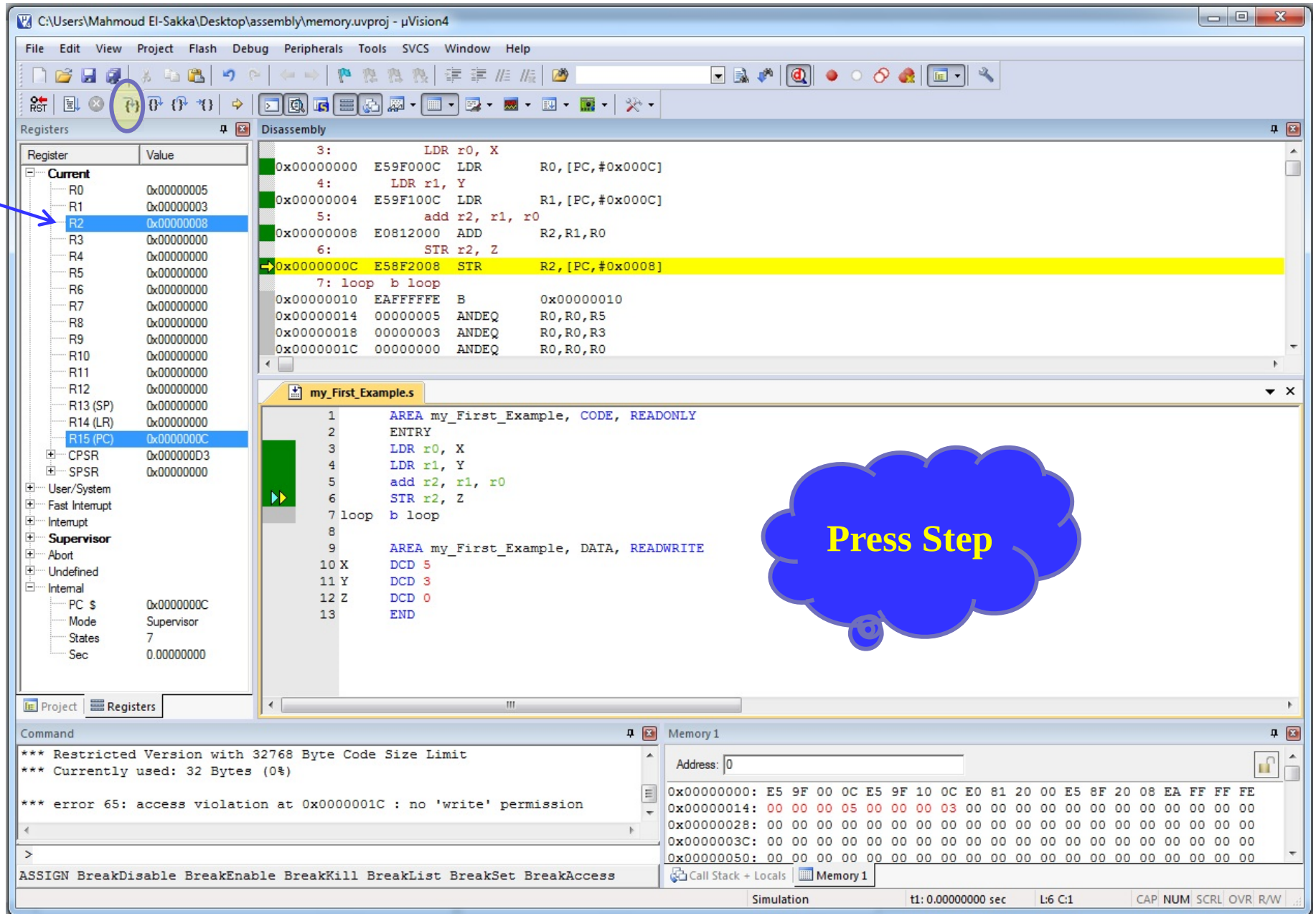
Using the Simulator—*step-by-step*



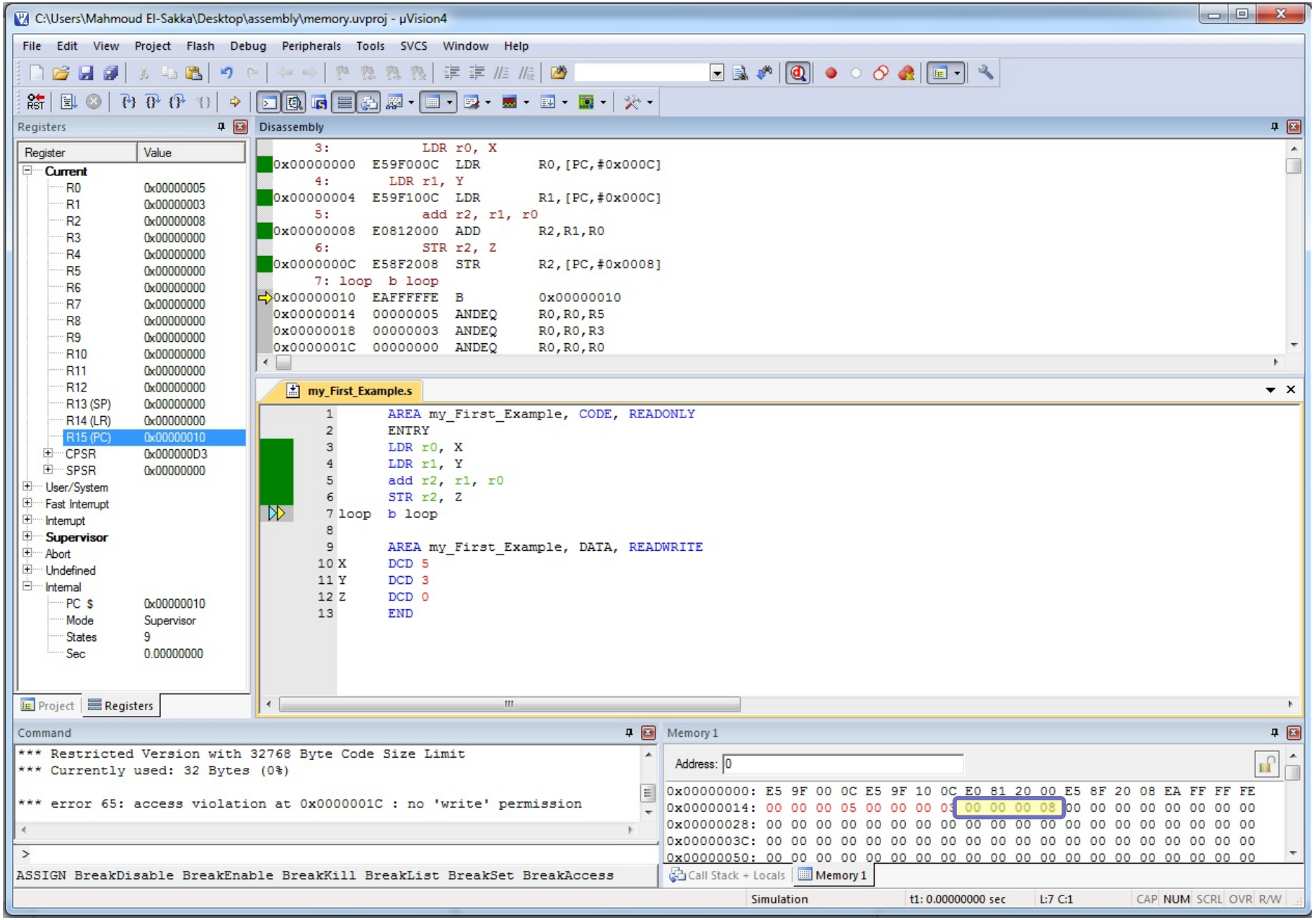
Using the Simulator—*step-by-step*



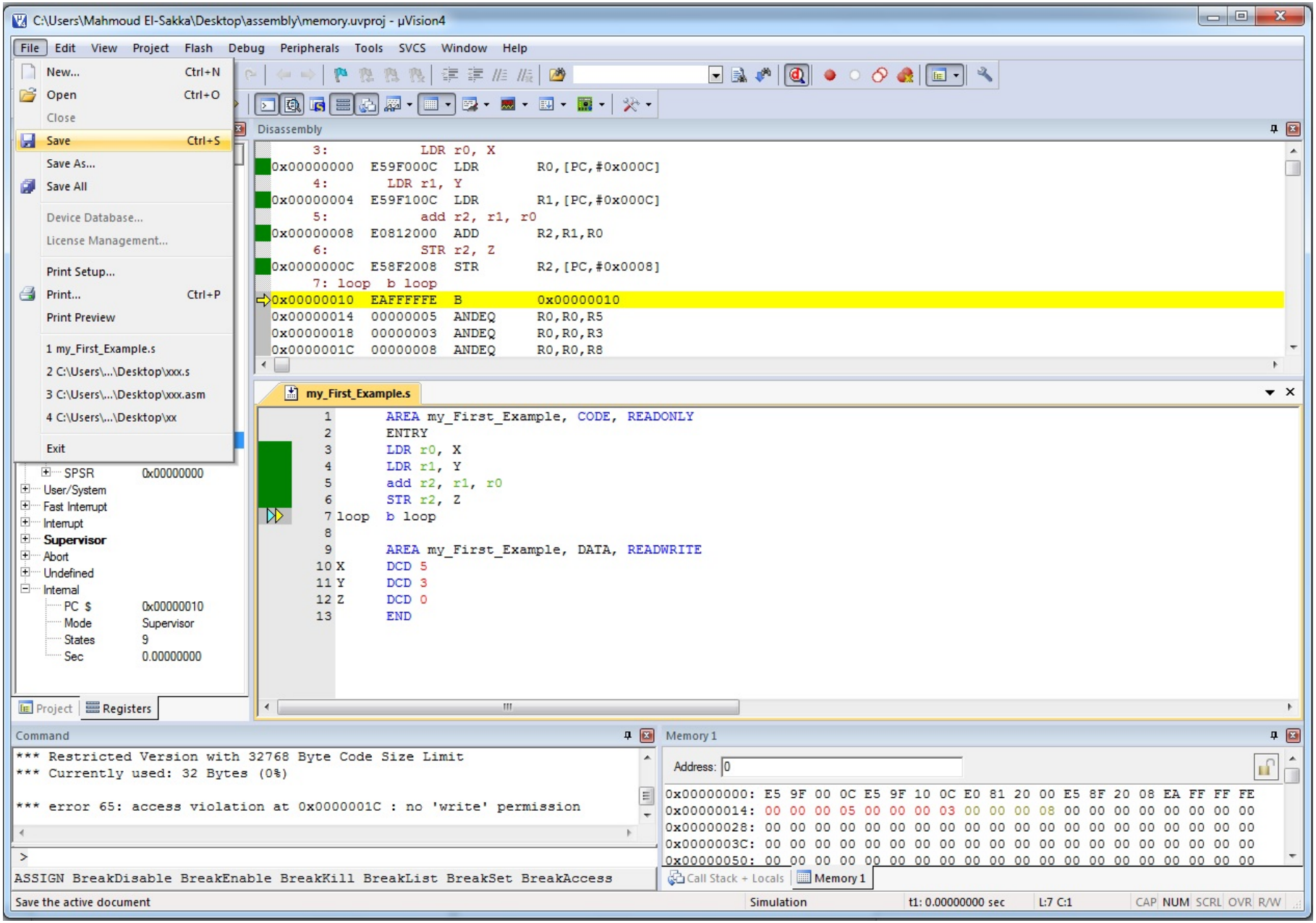
Using the Simulator—*step-by-step*



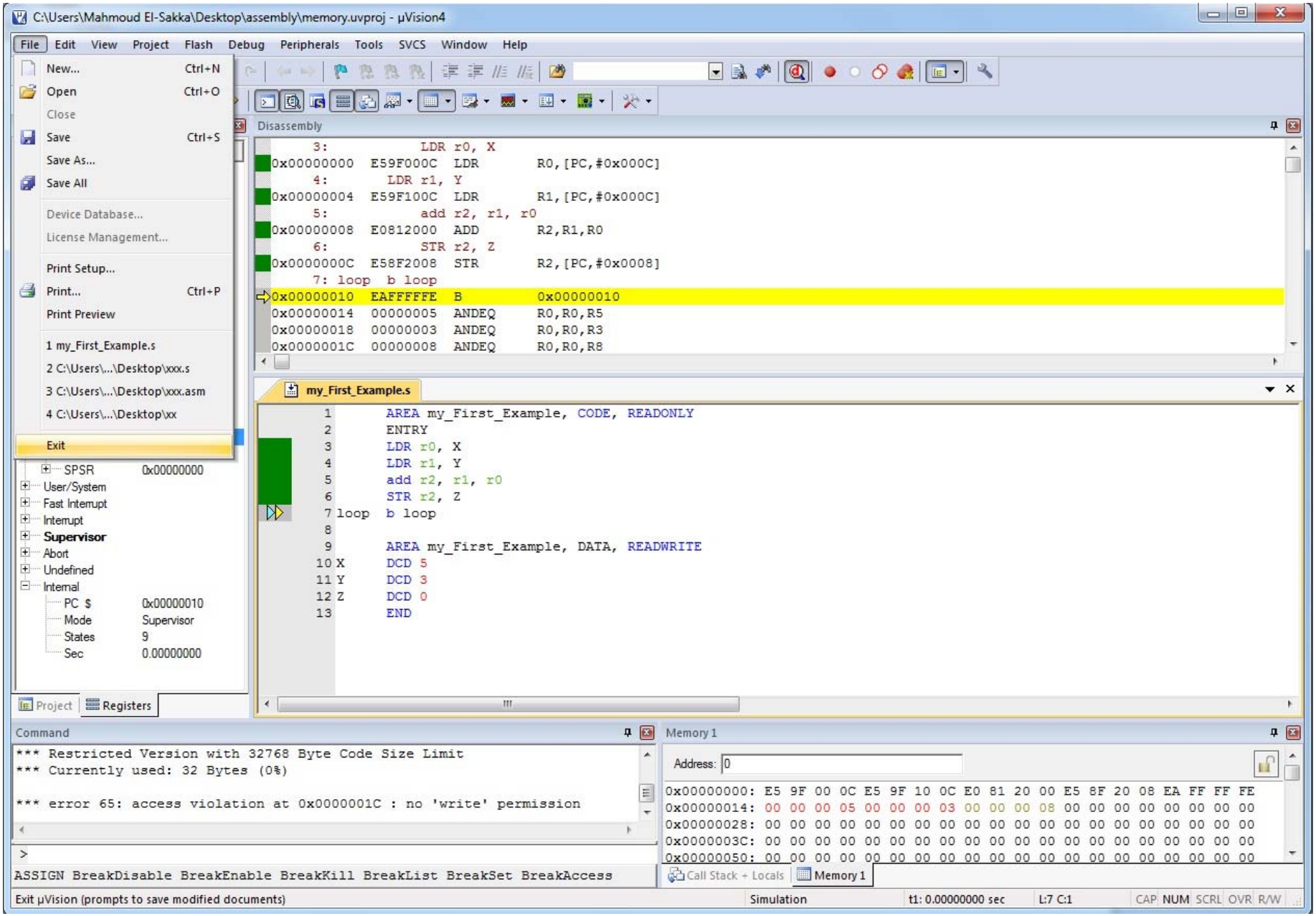
Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*

- In this last example, we demonstrated a way to change a range of the current memory map from execute/read to read/write using the Debug/Memory Map menu option.
- However, this memory map operation ***needs to be performed each time you enter the debugging mode***
- The other way to perform this memory map operation is to include it with the project.
  This way, the memory map operation ***will stay active whenever you enter the debugging mode***.

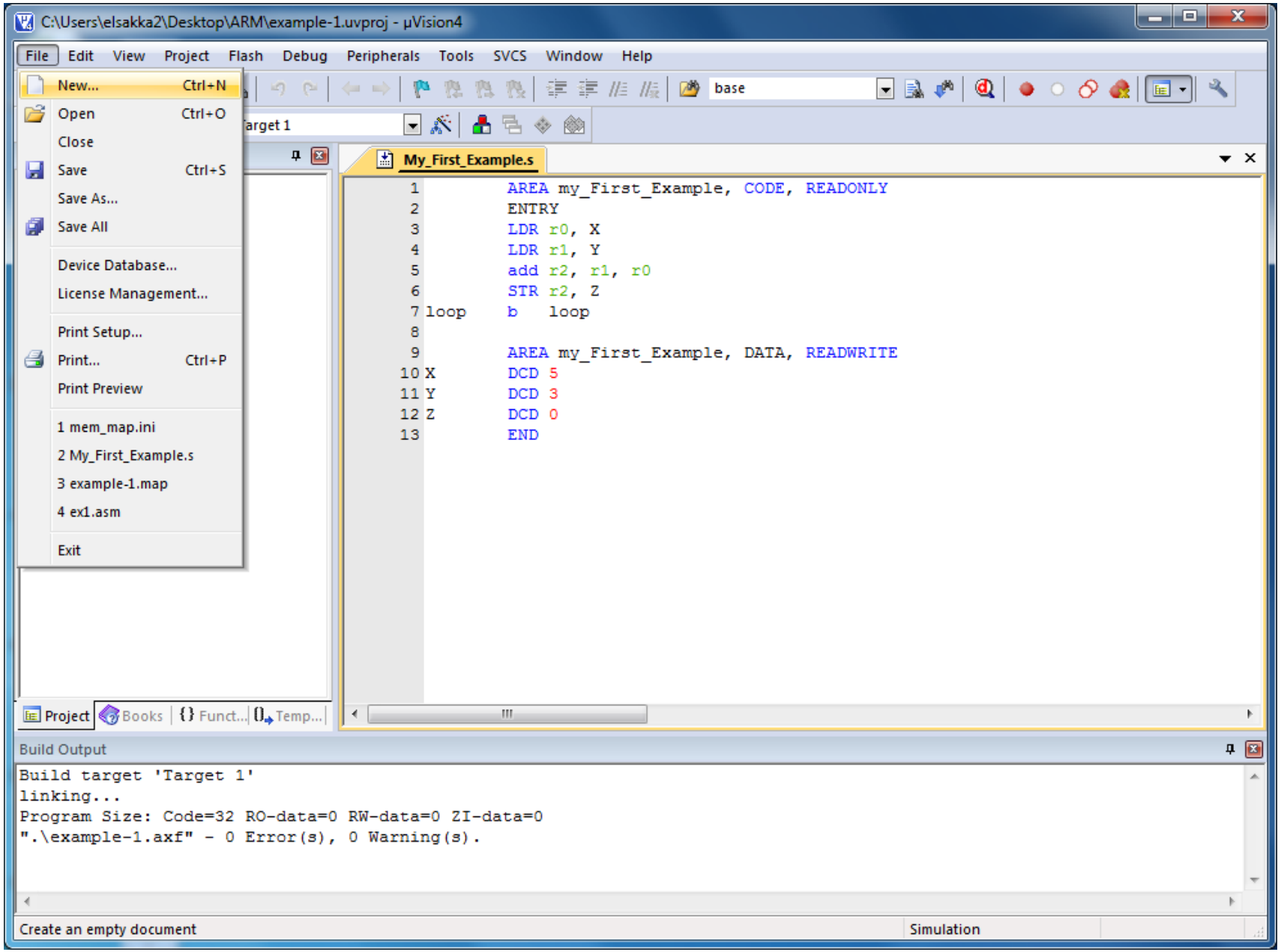
Using the Simulator—*step-by-step*

- First, you need to create a text file with the following content:

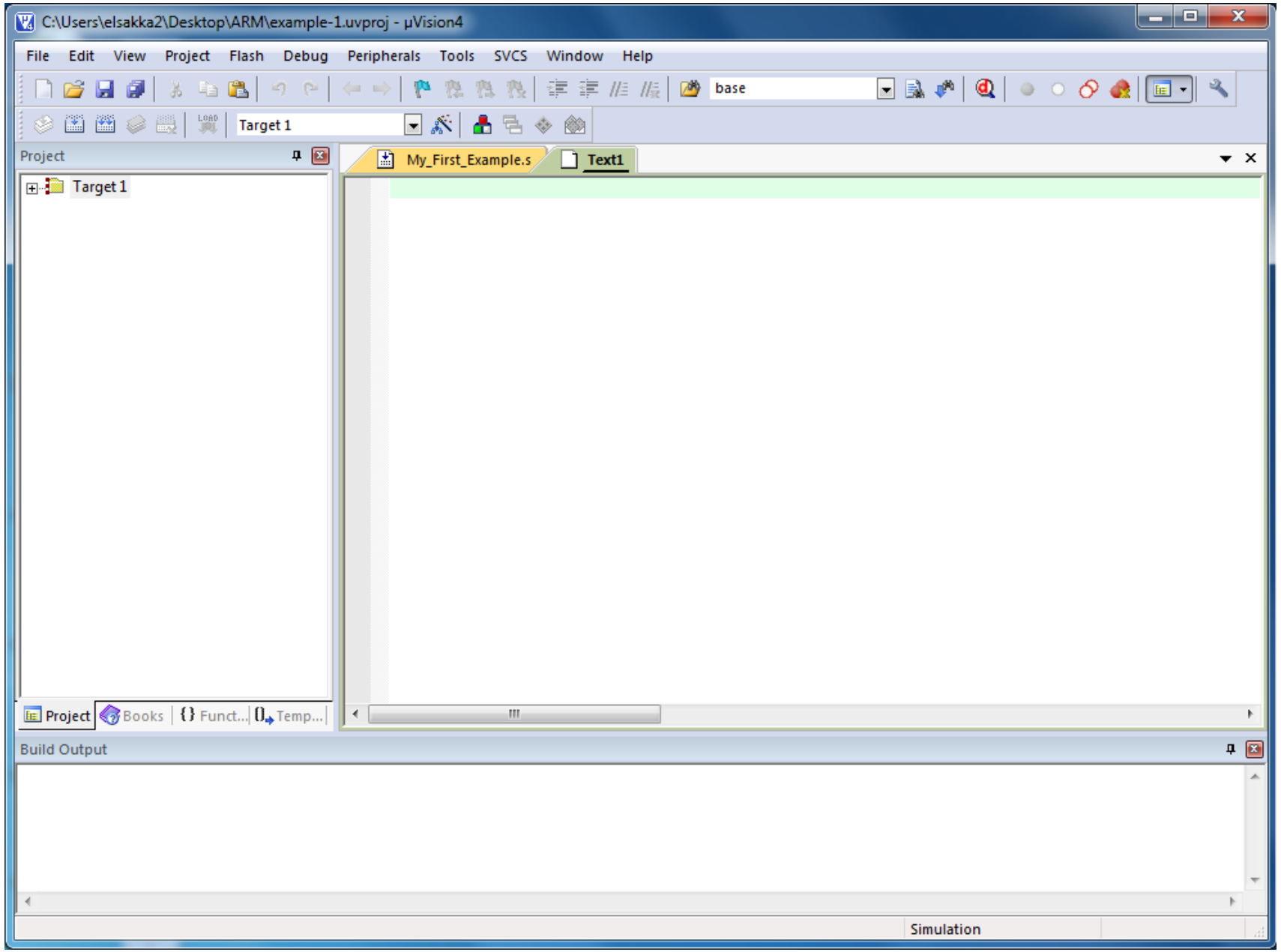
```
/* Map Permission*/
MAP 0x14, 0x1F READ WRITE // allow R/W access to the DATA space
```

- Note that whatever written between `/*...*/` is a comment, and whatever comes after `//` is a comment as well
- The main part of this file is : `MAP 0x14, 0x1F READ WRITE`

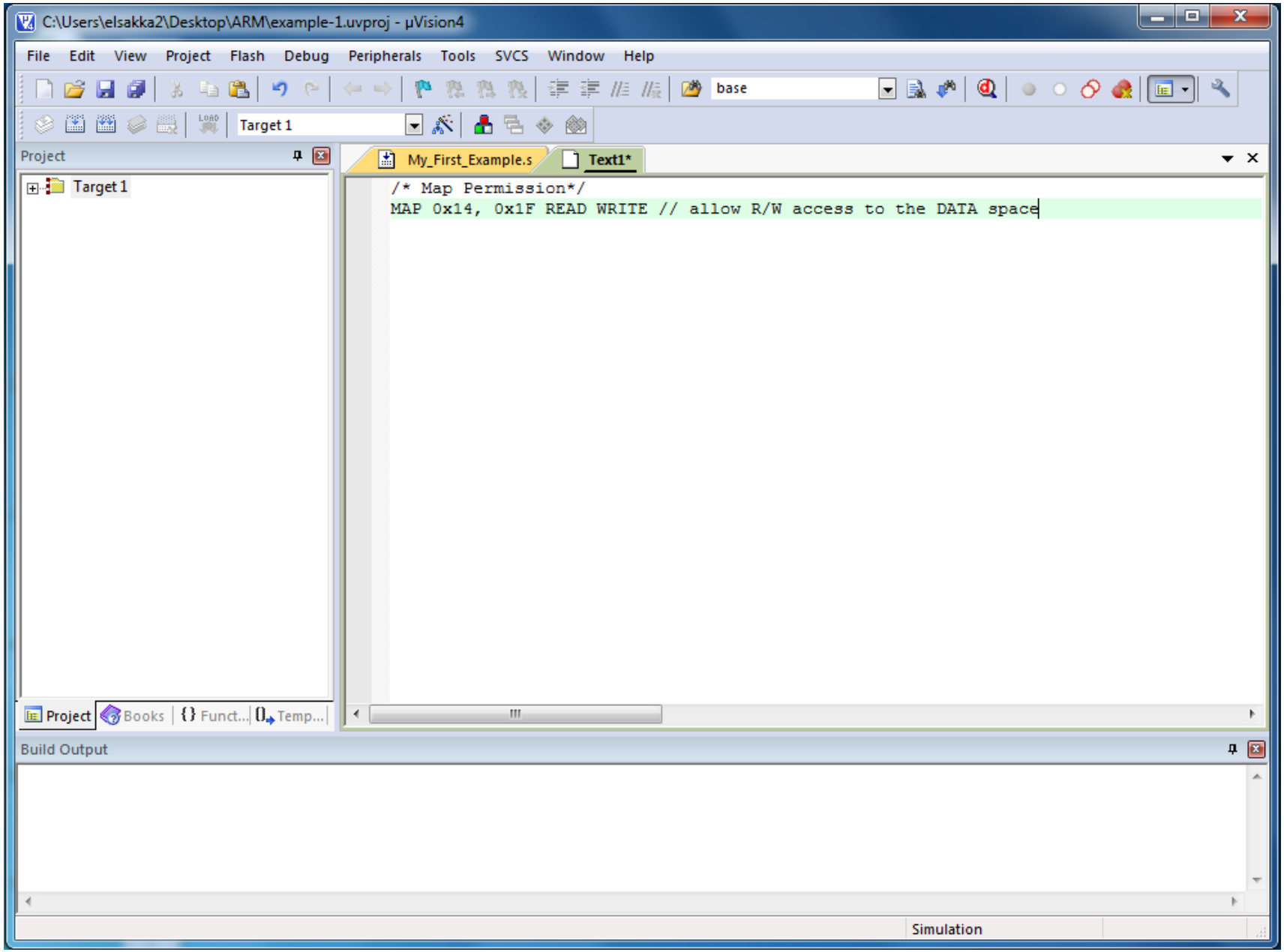
Using the Simulator—*step-by-step*



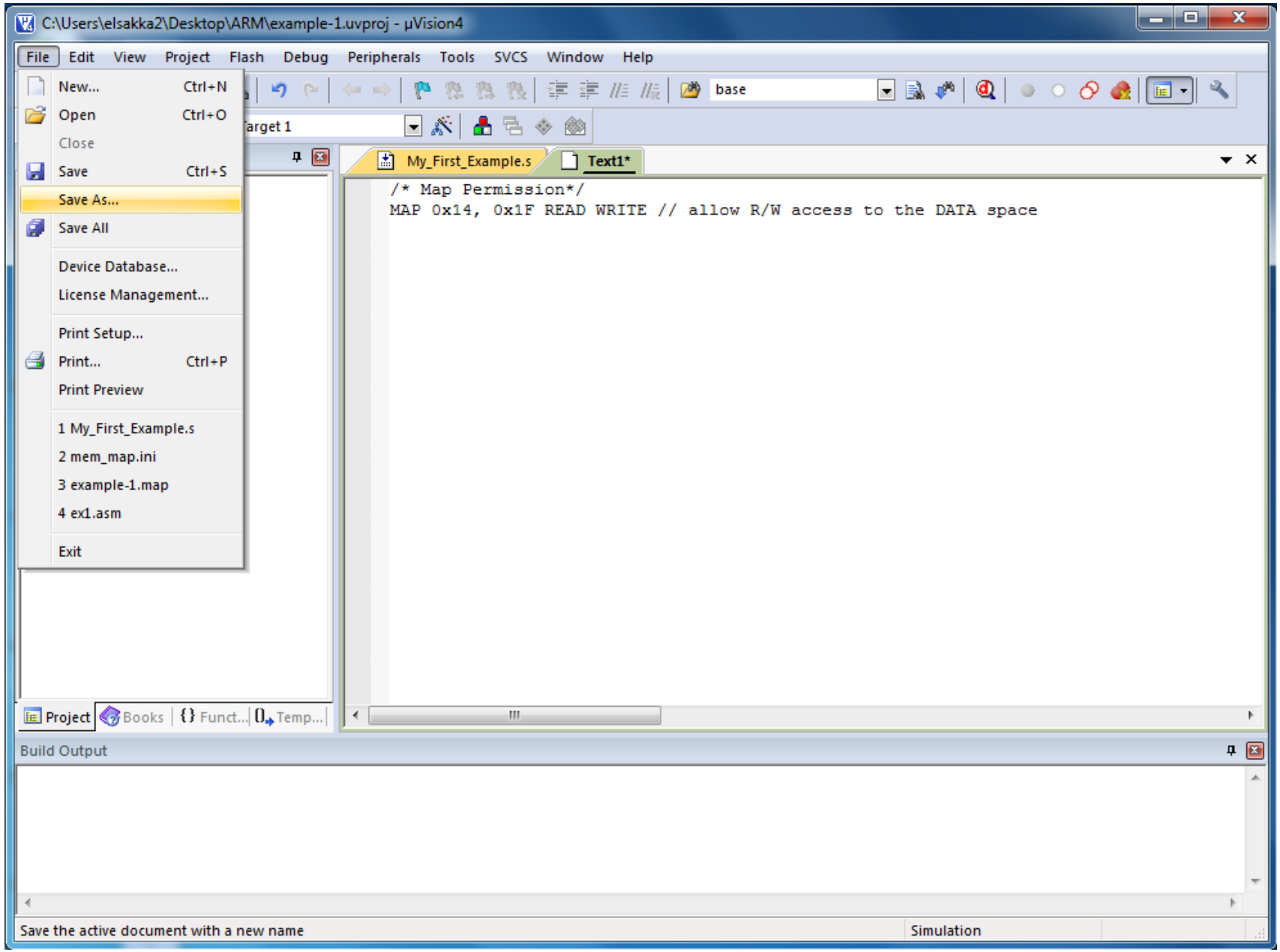
Using the Simulator—*step-by-step*



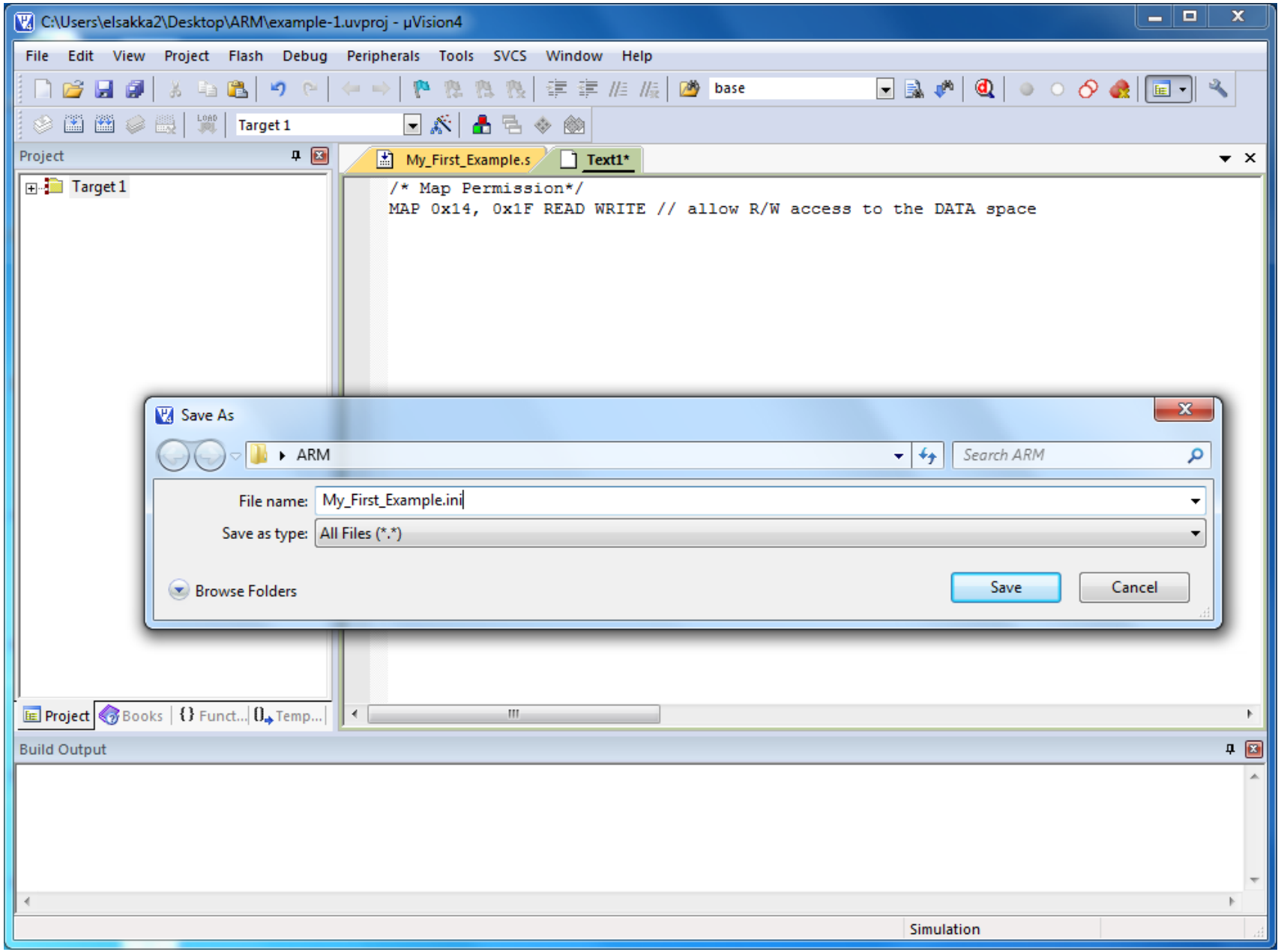
Using the Simulator—*step-by-step*



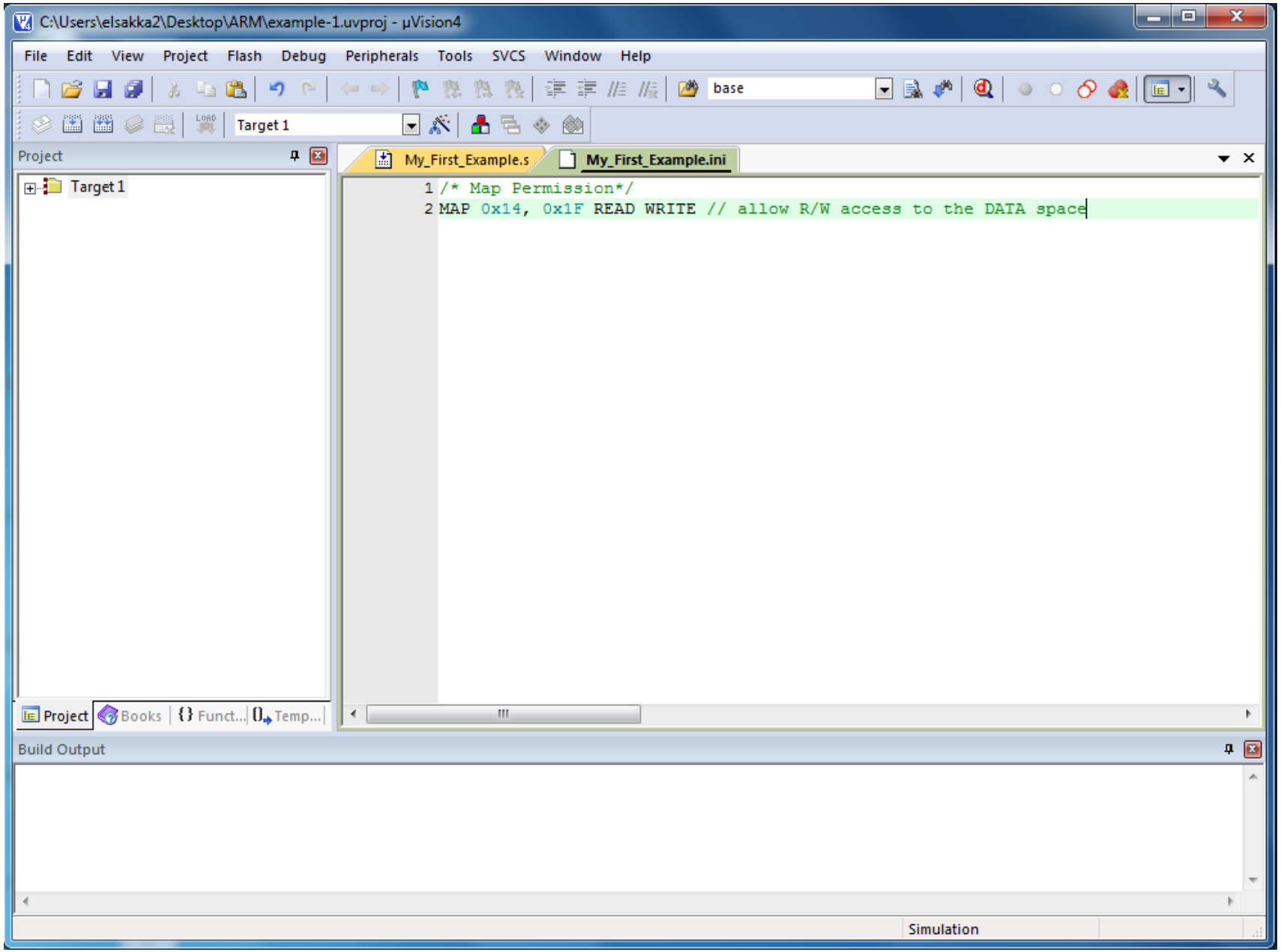
Using the Simulator—*step-by-step*



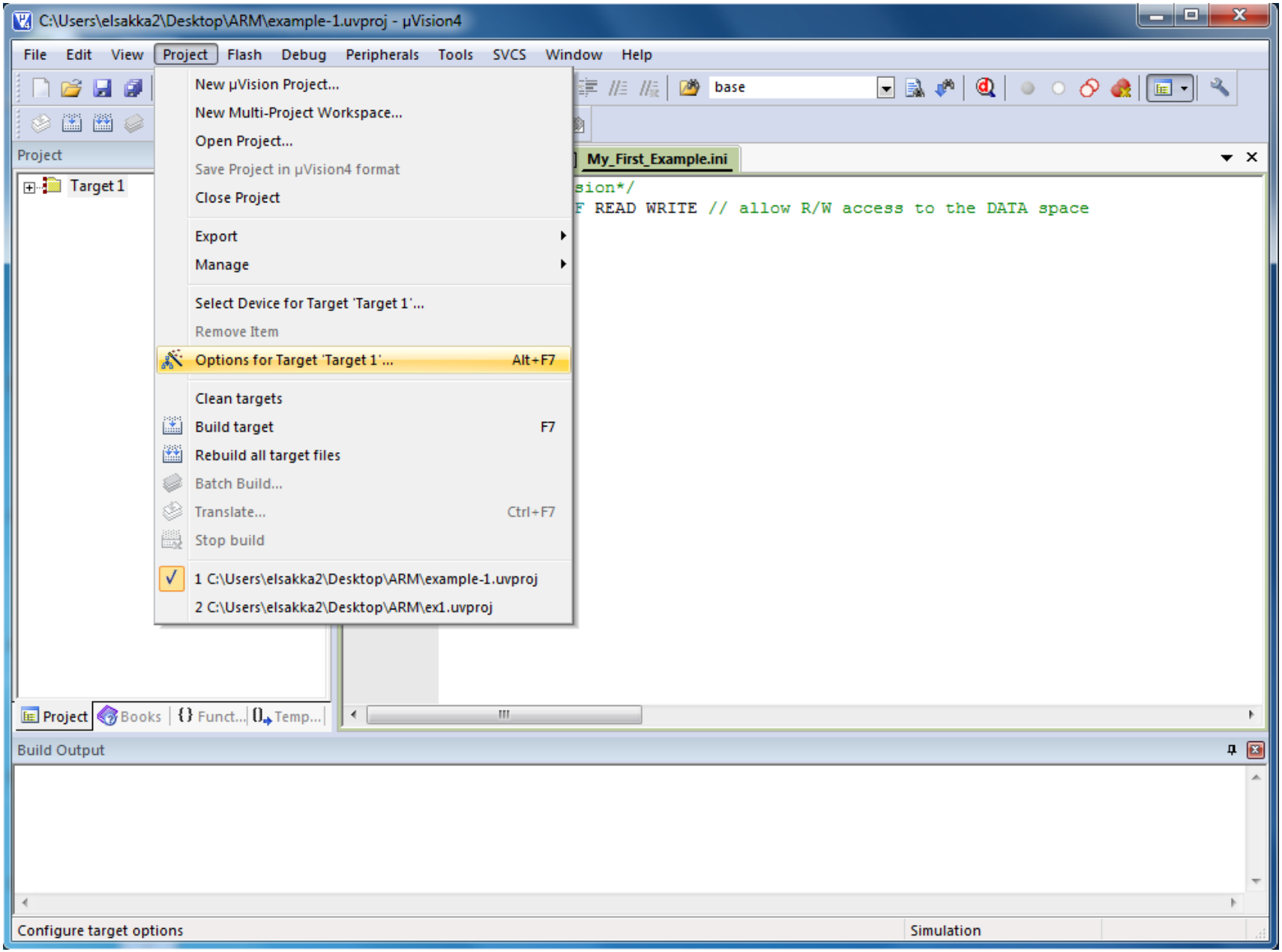
Using the Simulator—*step-by-step*



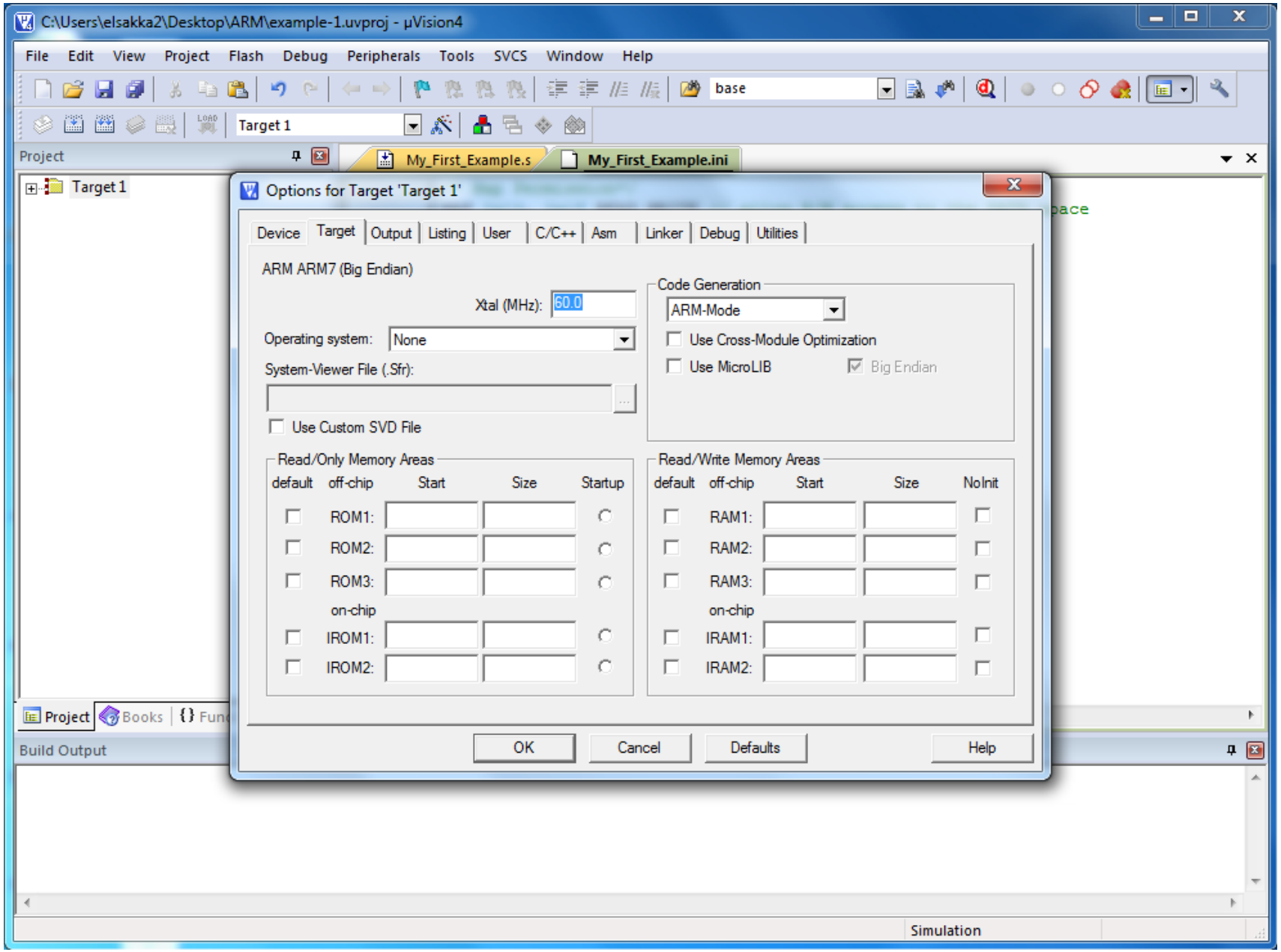
Using the Simulator—*step-by-step*



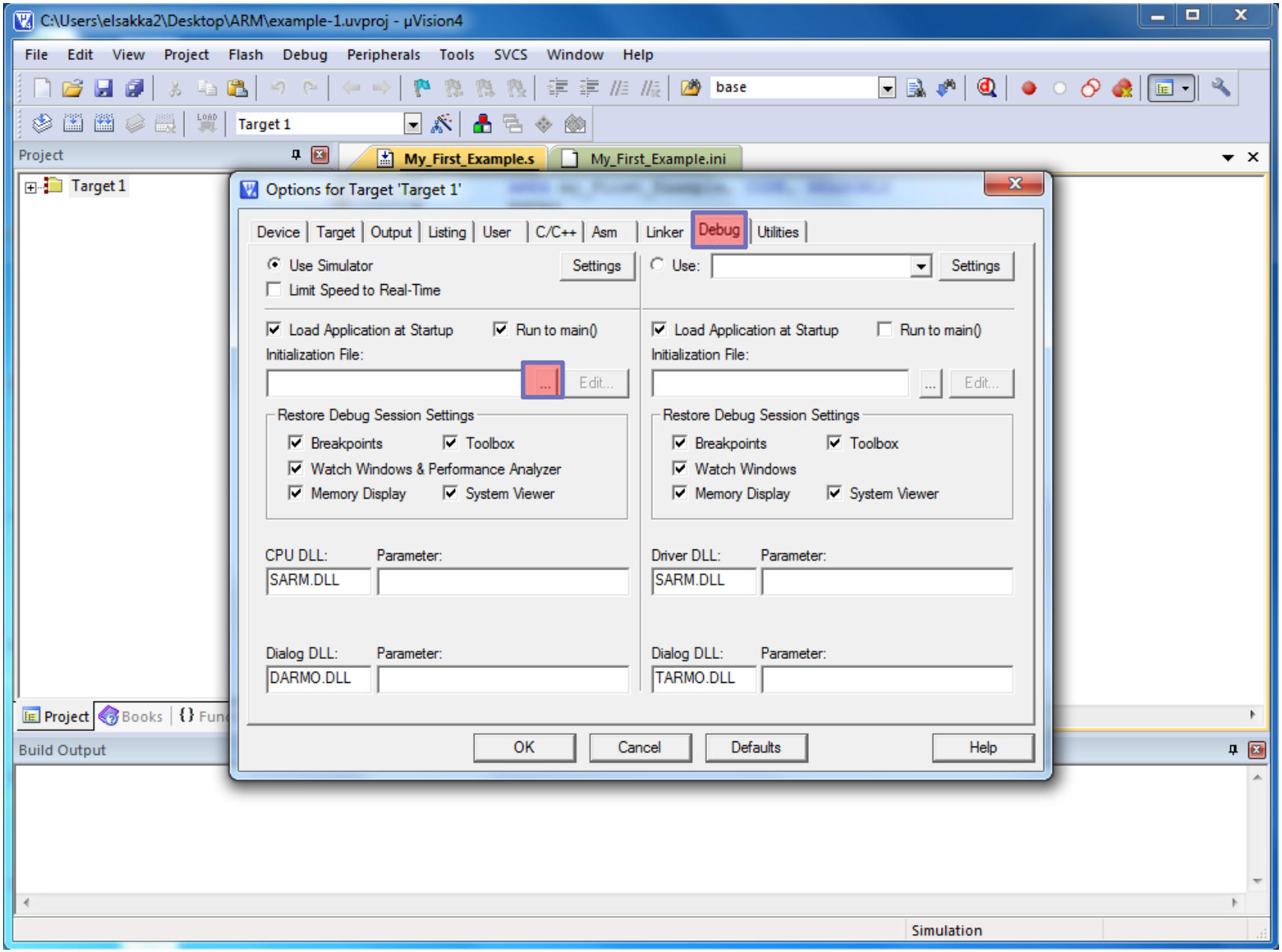
Using the Simulator—*step-by-step*



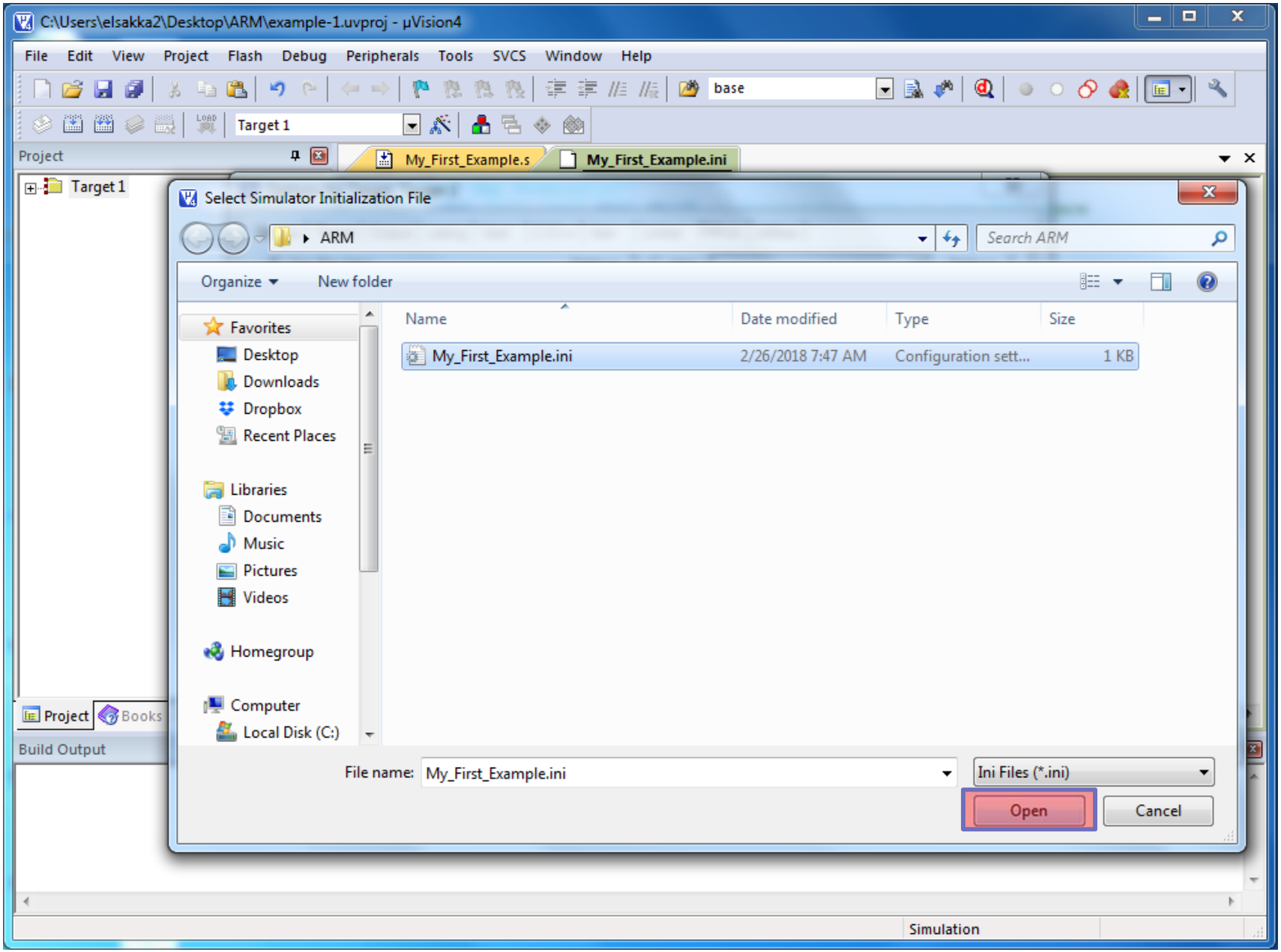
Using the Simulator—*step-by-step*



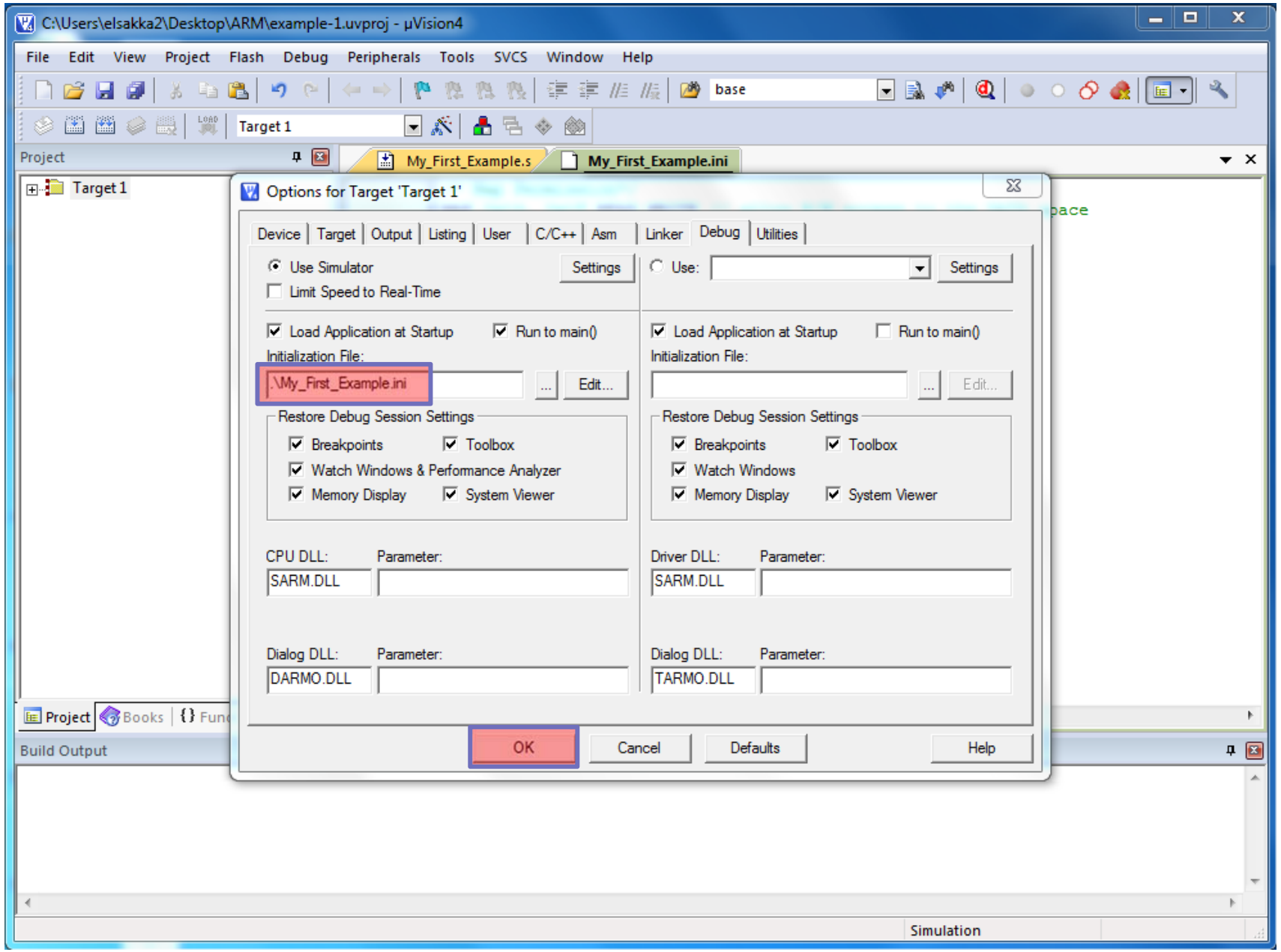
Using the Simulator—*step-by-step*



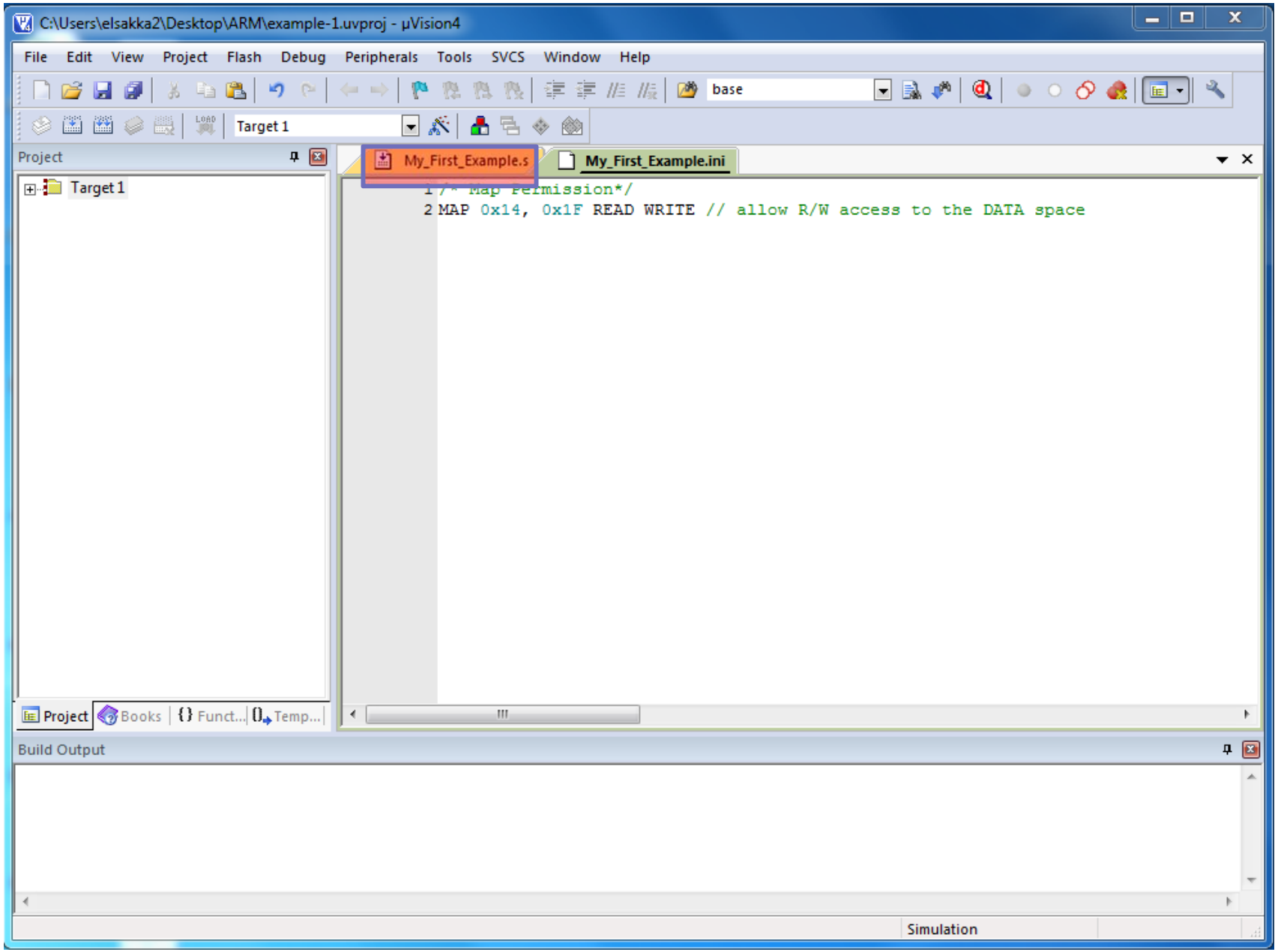
Using the Simulator—*step-by-step*



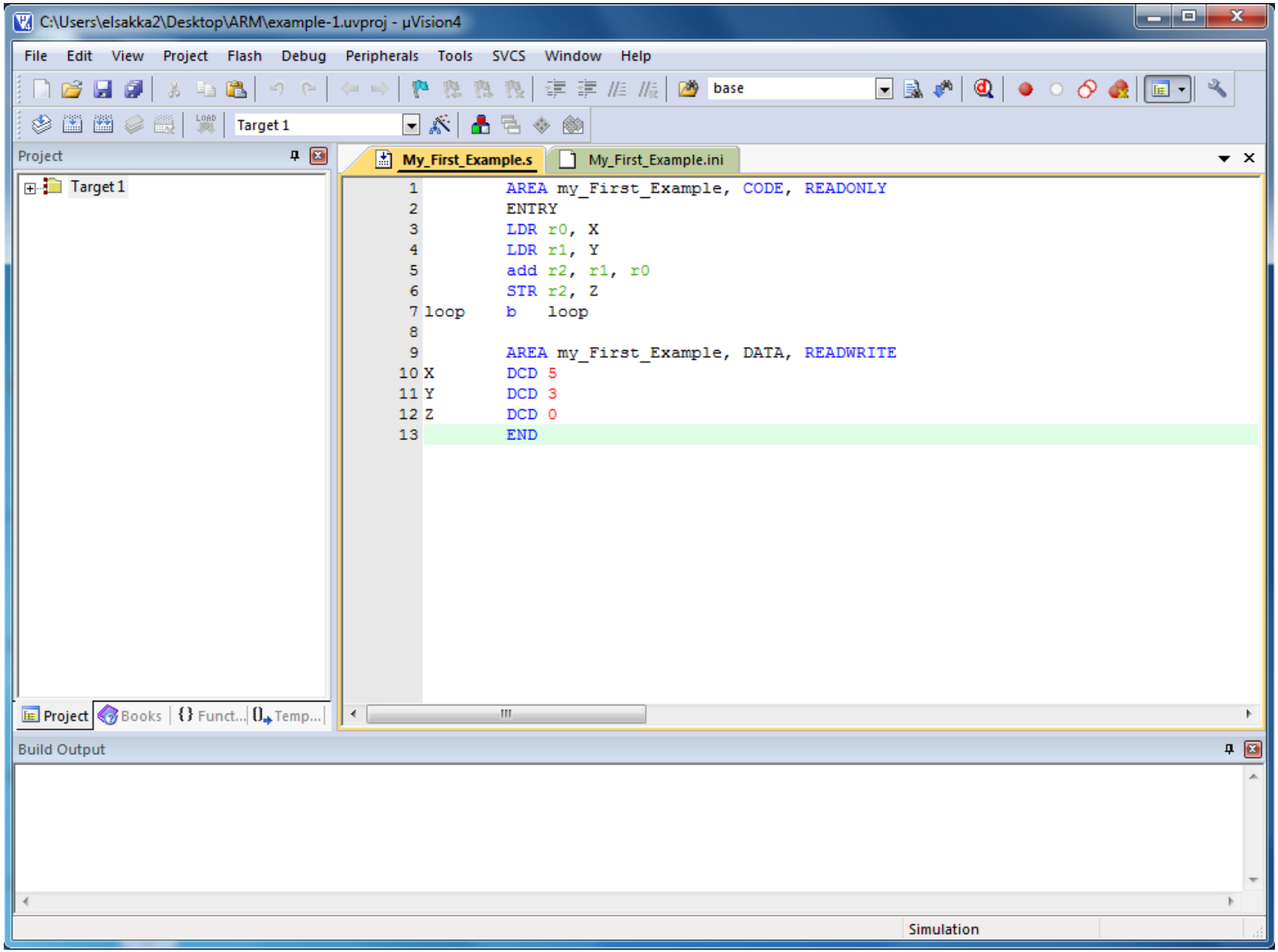
Using the Simulator—*step-by-step*



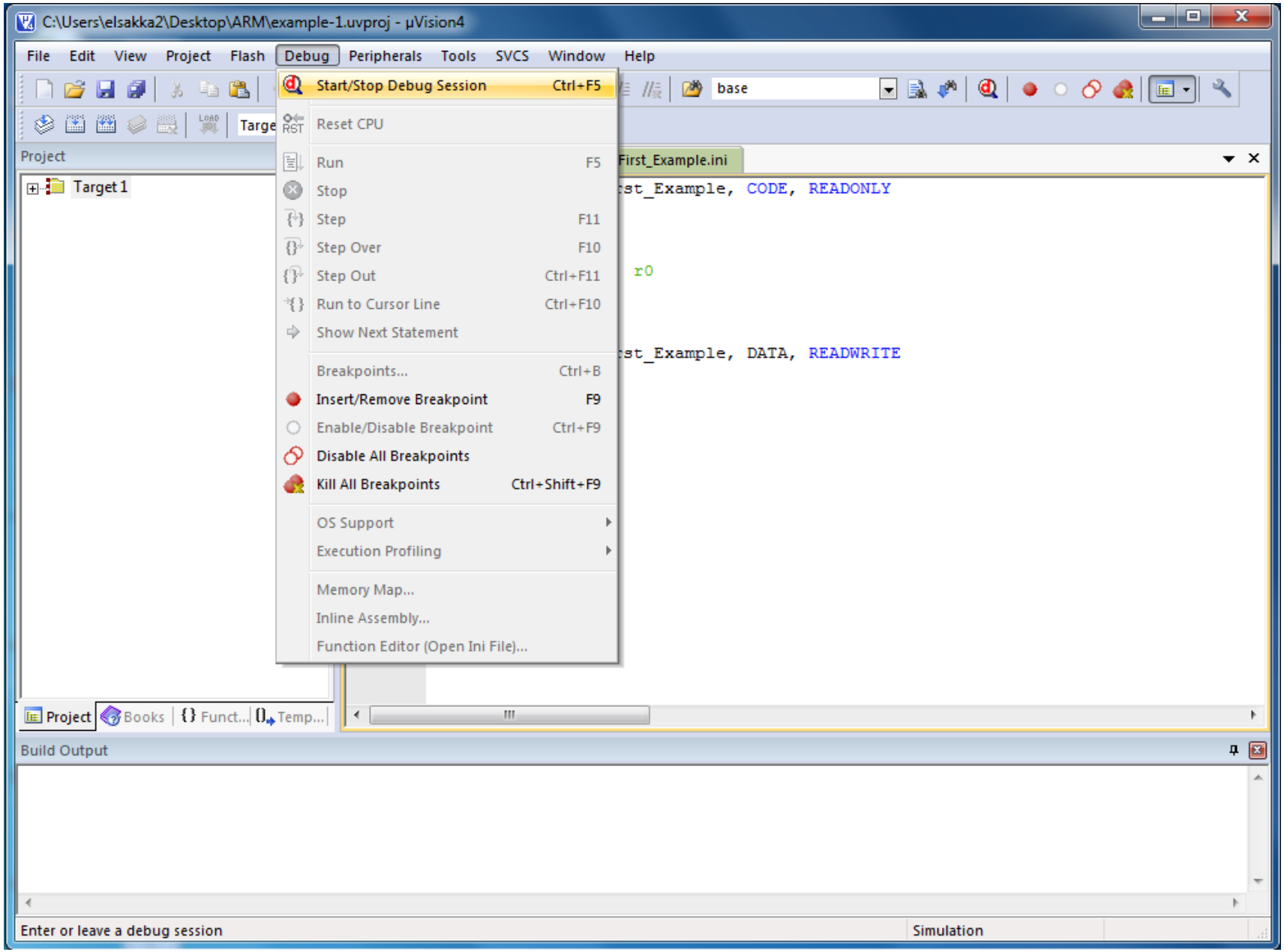
Using the Simulator—*step-by-step*



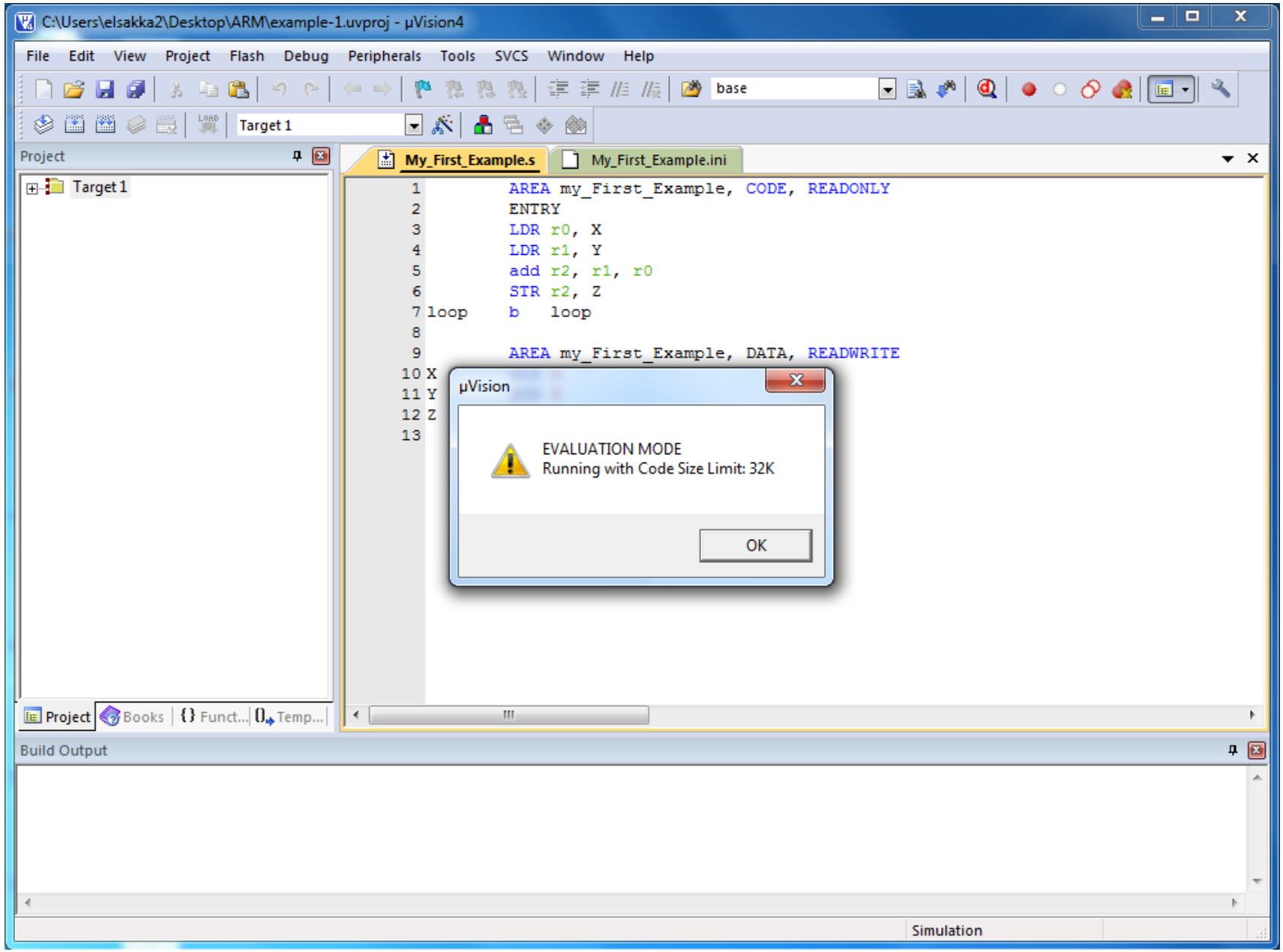
Using the Simulator—*step-by-step*

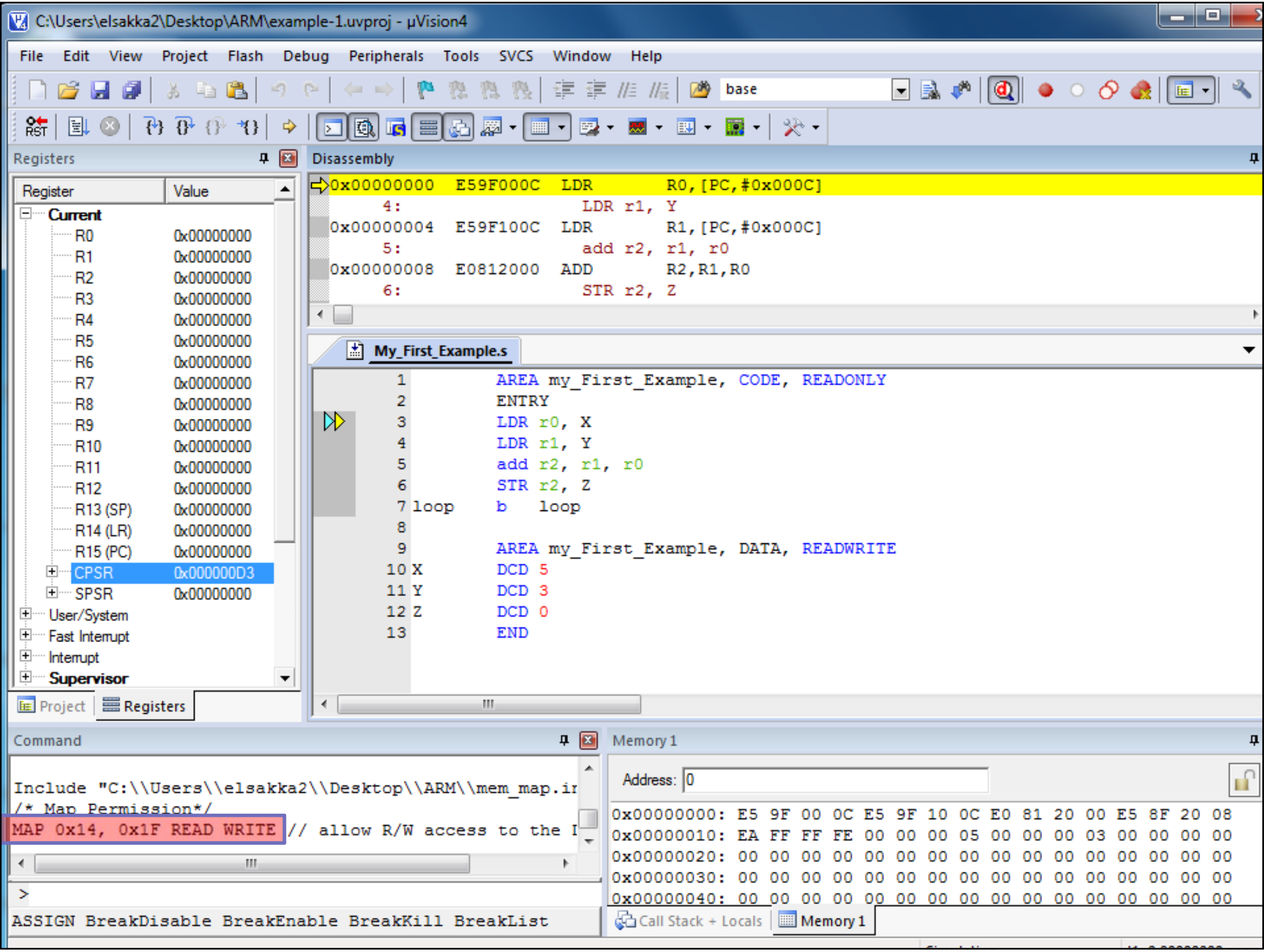


Using the Simulator—*step-by-step*

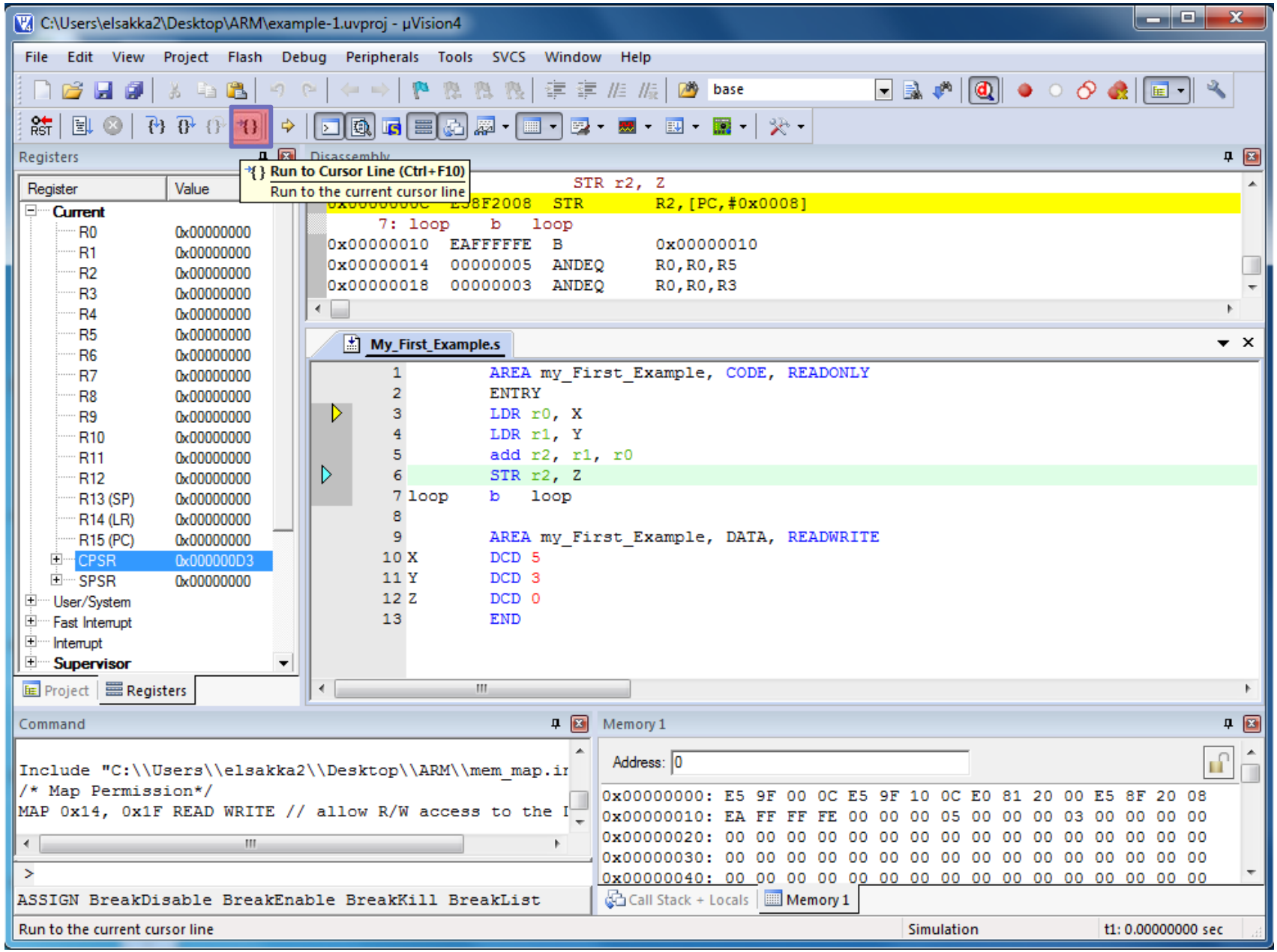


Using the Simulator—*step-by-step*

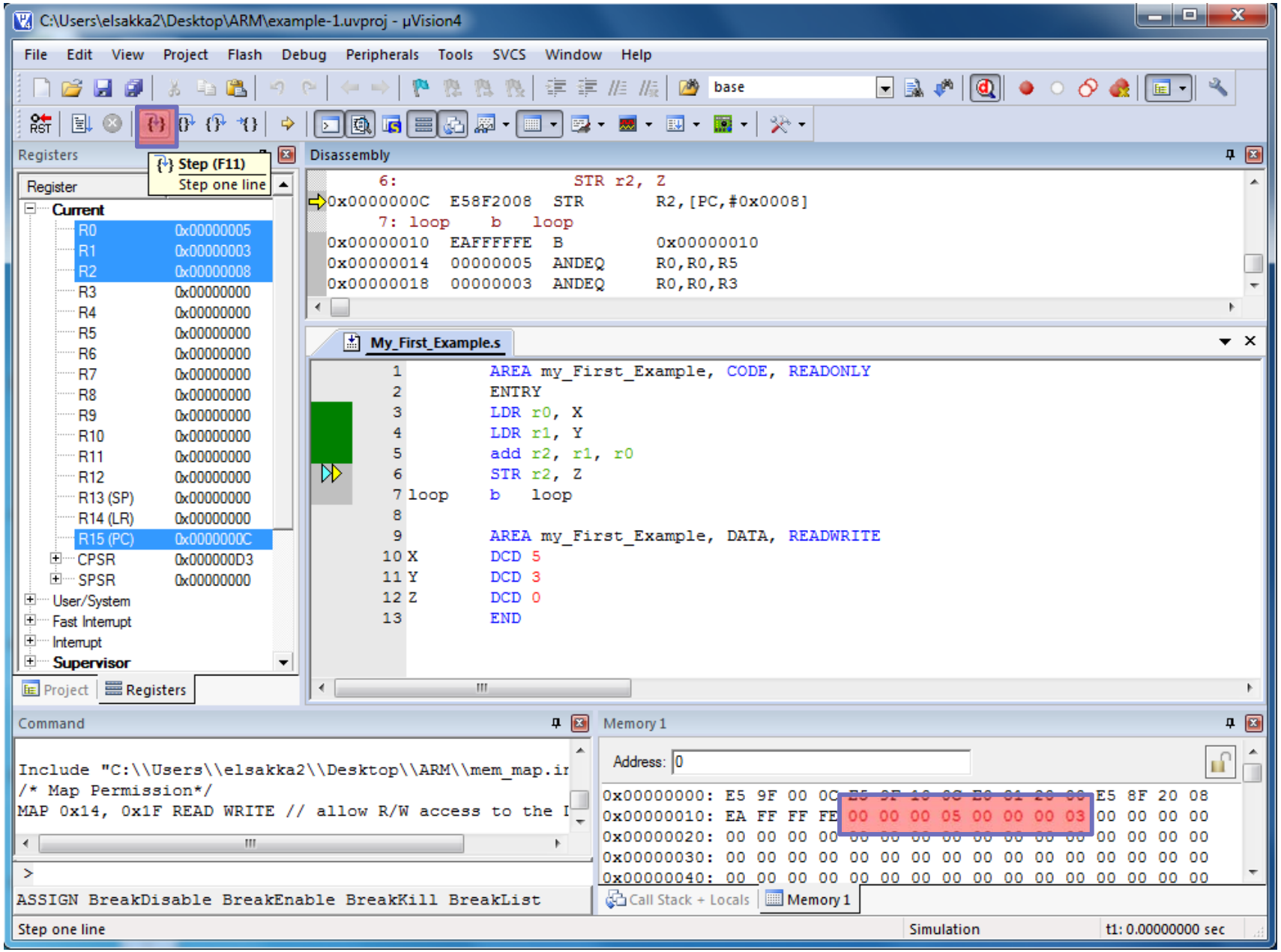



C:\Users\elsakka2\Desktop\ARM\example-1.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
base
Registers
Register Value
Current
R0 0x00000000
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000000
CPSR 0x000000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Project Registers
Disassembly
0x00000000 E59F000C LDR R0,[PC,#0x000C]
4: LDR r1, Y
0x00000004 E59F100C LDR R1,[PC,#0x000C]
5: add r2, r1, r0
0x00000008 E0812000 ADD R2,R1,R0
6: STR r2, Z
My_First_Example.s
1 AREA my_First_Example, CODE, READONLY
2 ENTRY
3 LDR r0, X
4 LDR r1, Y
5 add r2, r1, r0
6 STR r2, Z
7 loop b loop
8
9 AREA my_First_Example, DATA, READWRITE
10 X DCD 5
11 Y DCD 3
12 Z DCD 0
13 END
Command
Include "C:\\Users\\elsakka2\\Desktop\\ARM\\mem_map.ir
/* Map Permission*/
MAP 0x14, 0x1F READ WRITE // allow R/W access to the I
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
Memory 1
Address: 0
0x00000000: E5 9F 00 0C E5 9F 10 0C E0 81 20 00 E5 8F 20 08
0x00000010: EA FF FF FE 00 00 00 05 00 00 00 03 00 00 00 00
0x00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack + Locals Memory 1

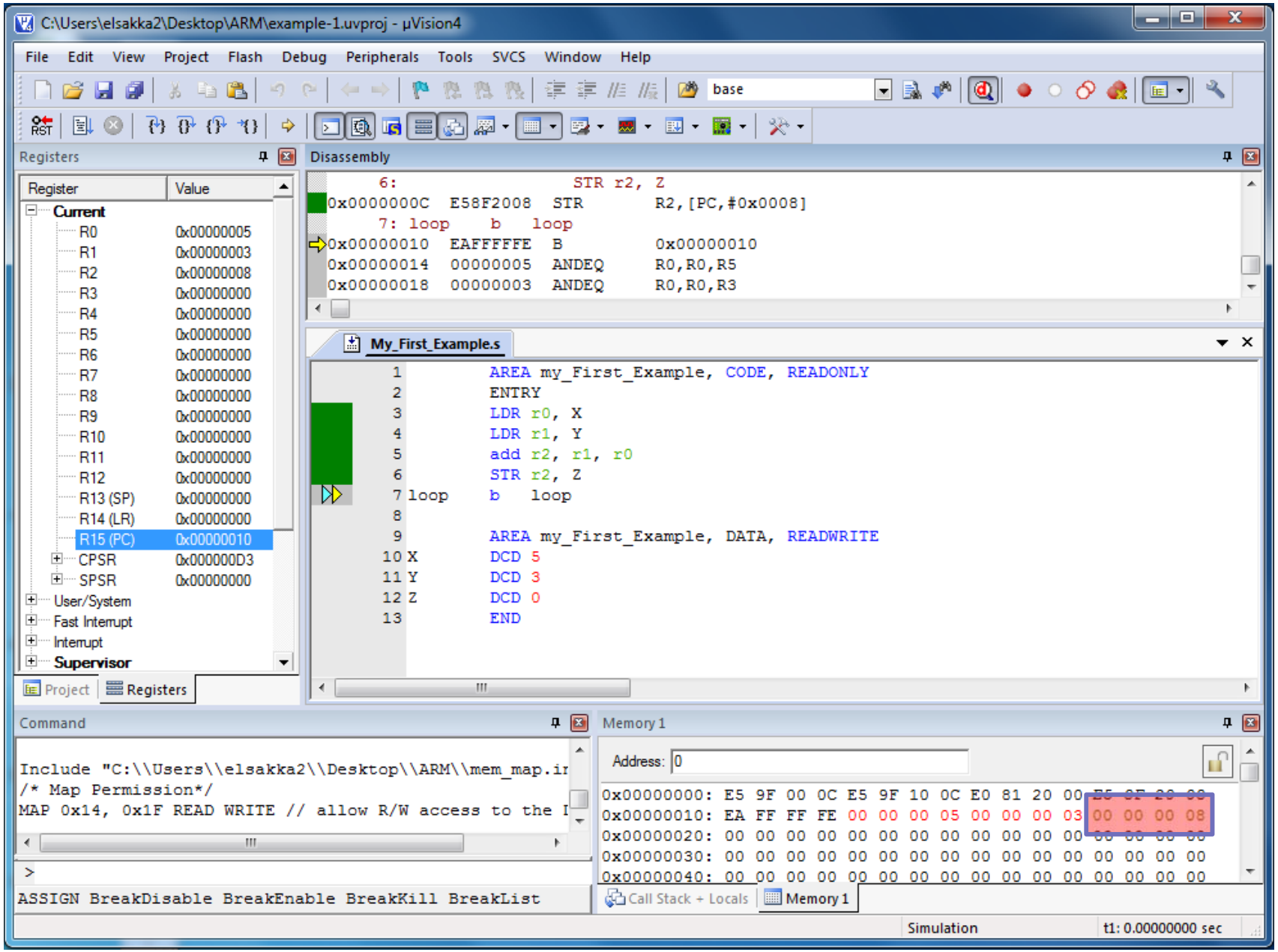
Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*



Using the Simulator—*step-by-step*



Steps Summary