Keil uVision 4

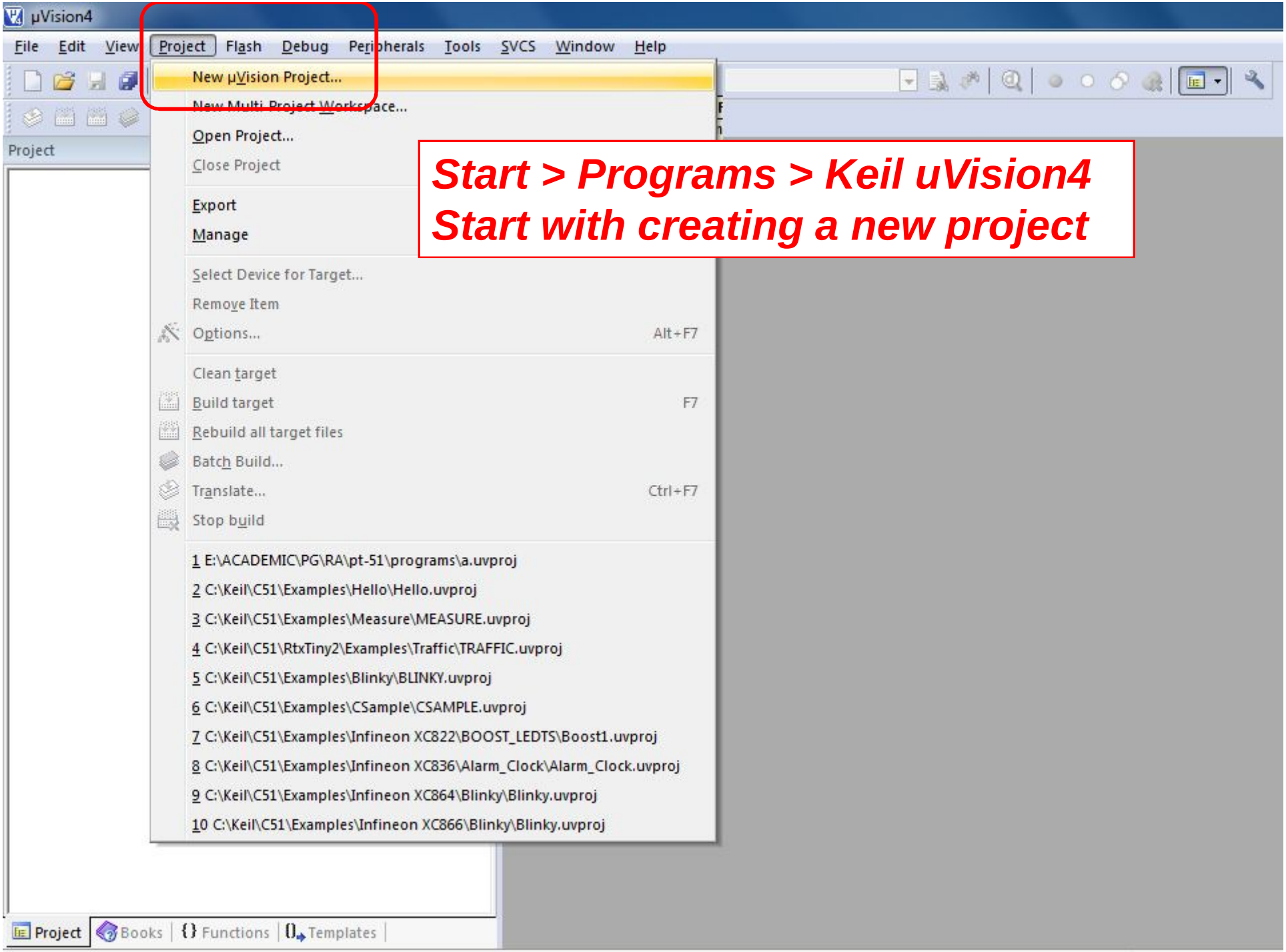
Software Development Kit (SDK) for 8051 based microcontrollers

Compiled by: Vinayak G P, Deepak Malani

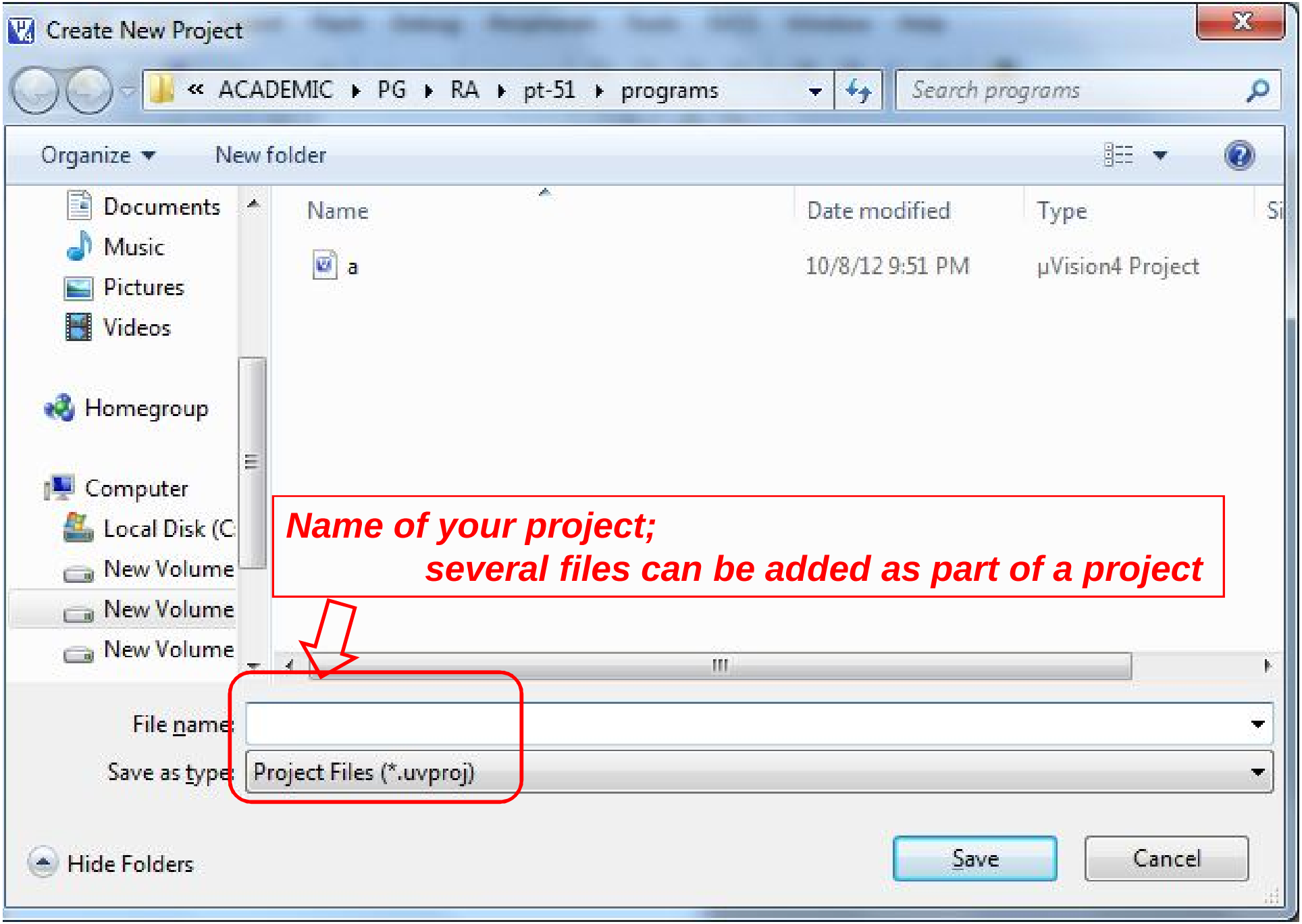
- To download Keil uvision4, goto www.keil.com/download/product/ and click on C51(development tools for all 8051 devices.
- You will be asked fill in a form, after which you can download the software.

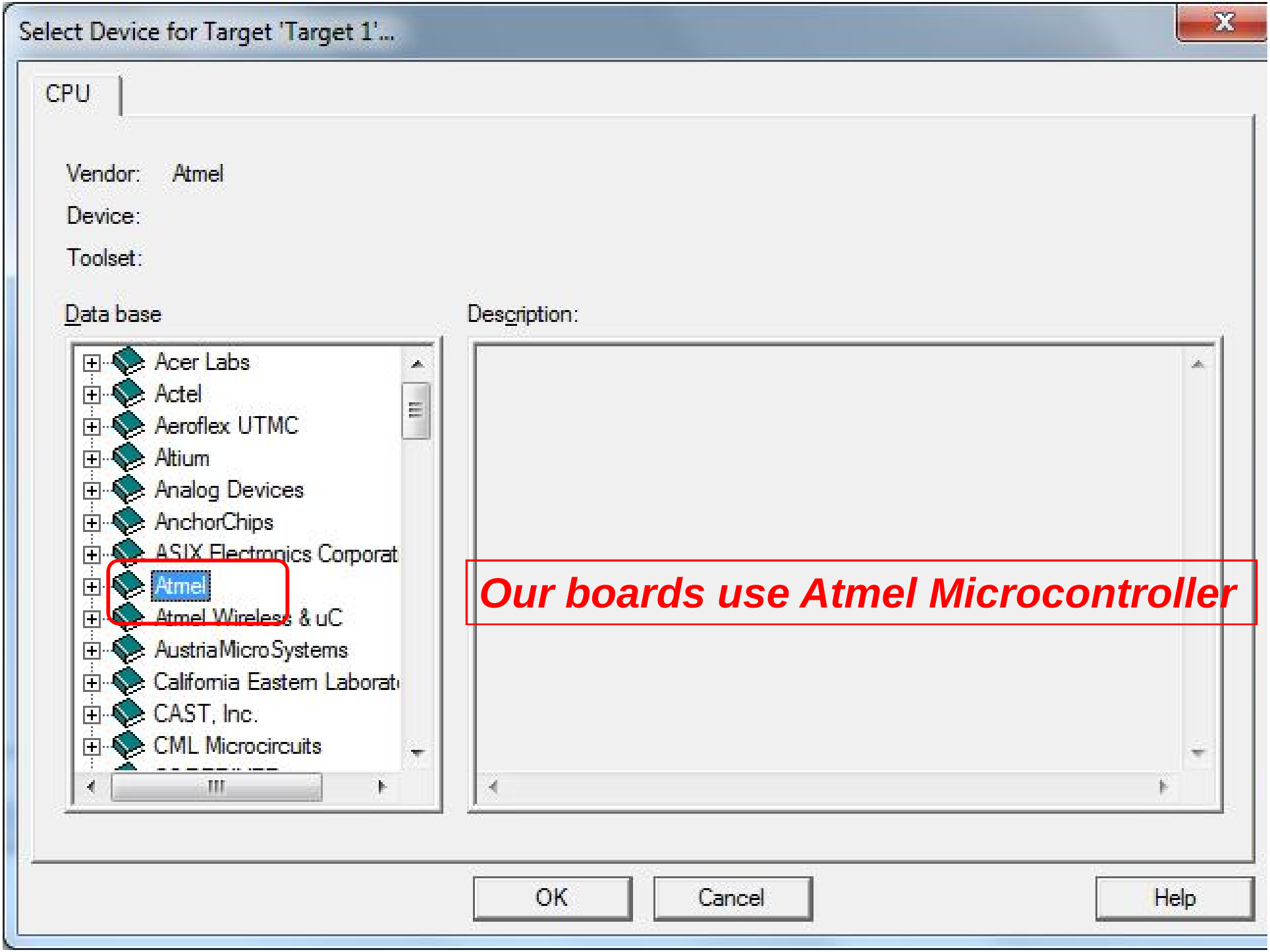
Installation

- The installation procedure is straight forward. Just follow on-screen instructions.
- Once installation is complete, open Keil uvision4.



Start > Programs > Keil uVision4
Start with creating a new project


Create New Project
« ACADEMIC ▸ PG ▸ RA ▸ pt-51 ▸ programs
Search programs
Organize
New folder
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:
New Volume
New Volume
New Volume
Name
Date modified
Type
a
10/8/12 9:51 PM
µVision4 Project
Name of your project;
several files can be added as part of a project
File name:
Save as type:
Project Files (*.uvproj)
Hide Folders
Save
Cancel


Select Device for Target 'Target 1'...
CPU
Vendor: Atmel
Device:
Toolset:
Data base
Description:
Acer Labs
Actel
Aeroflex UTMC
Altium
Analog Devices
AnchorChips
ASIX Electronics Corporat
Atmel
Atmel Wireless & uC
AustriaMicroSystems
California Eastern Laborat
CAST, Inc.
CML Microcircuits
Our boards use Atmel Microcontroller
OK
Cancel
Help

Select Device for Target 'Target 1'...

CPU

Vendor: Atmel

Device: AT89C5131A

Toolset: C51

☐ Use Extended Linker (LX51) instead of BL51

☐ Use Extended Assembler (AX51) instead of A51

Data base

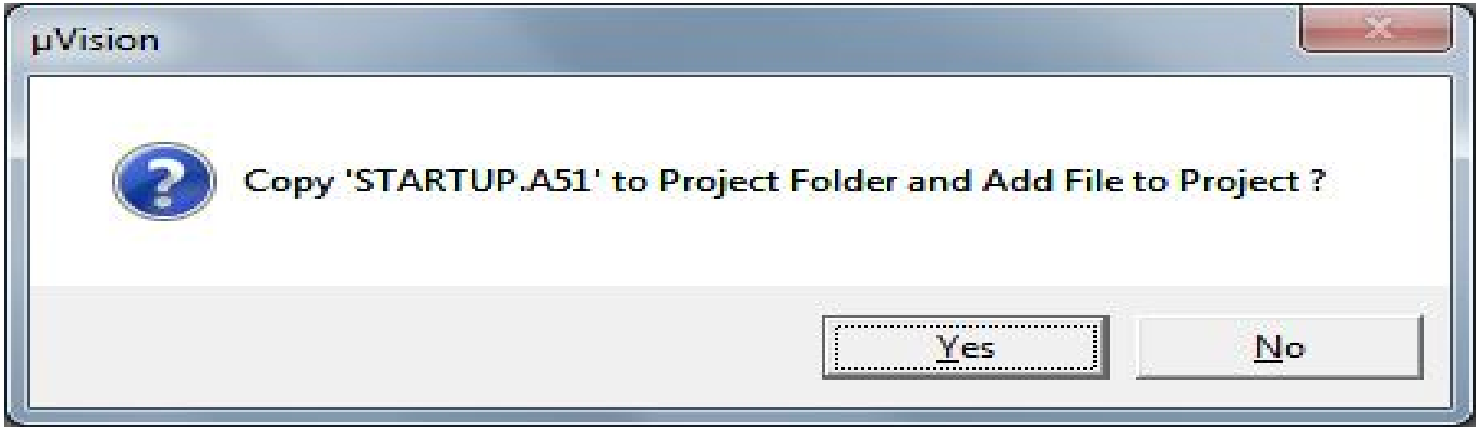
- AT89C4051
- AT89C51
- AT89C5115
- AT89C5130
- AT89C5130A
- AT89C5131
- AT89C5131A
- AT89C5132
- AT89C51AC3
- AT89C51CC03
- AT89C51ED2
- AT89C51IC2
- AT89C51ID2

Description:

8051-based microcontroller with Full Speed USB Device,
Dual Data Pointers, Enhanced UART, 3 16-bit Timers,
5 Channels PCA, WDT, 34 I/O lines, SPI, USB Module,
32 kBytes ISP Flash ROM, 1280 Bytes RAM

From Atmel family,
we use 89C5131A microcontroller

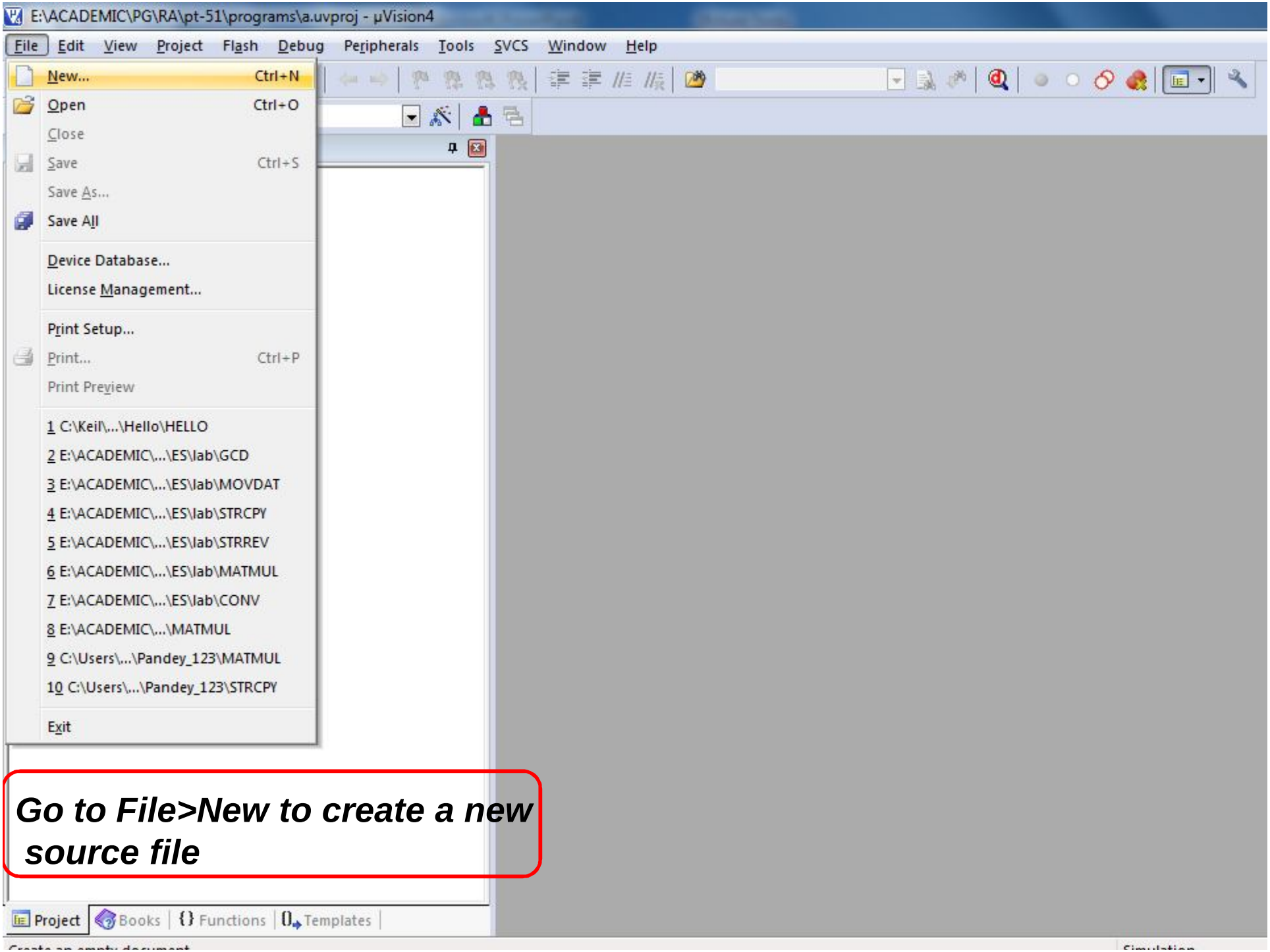
OK | Cancel | Help



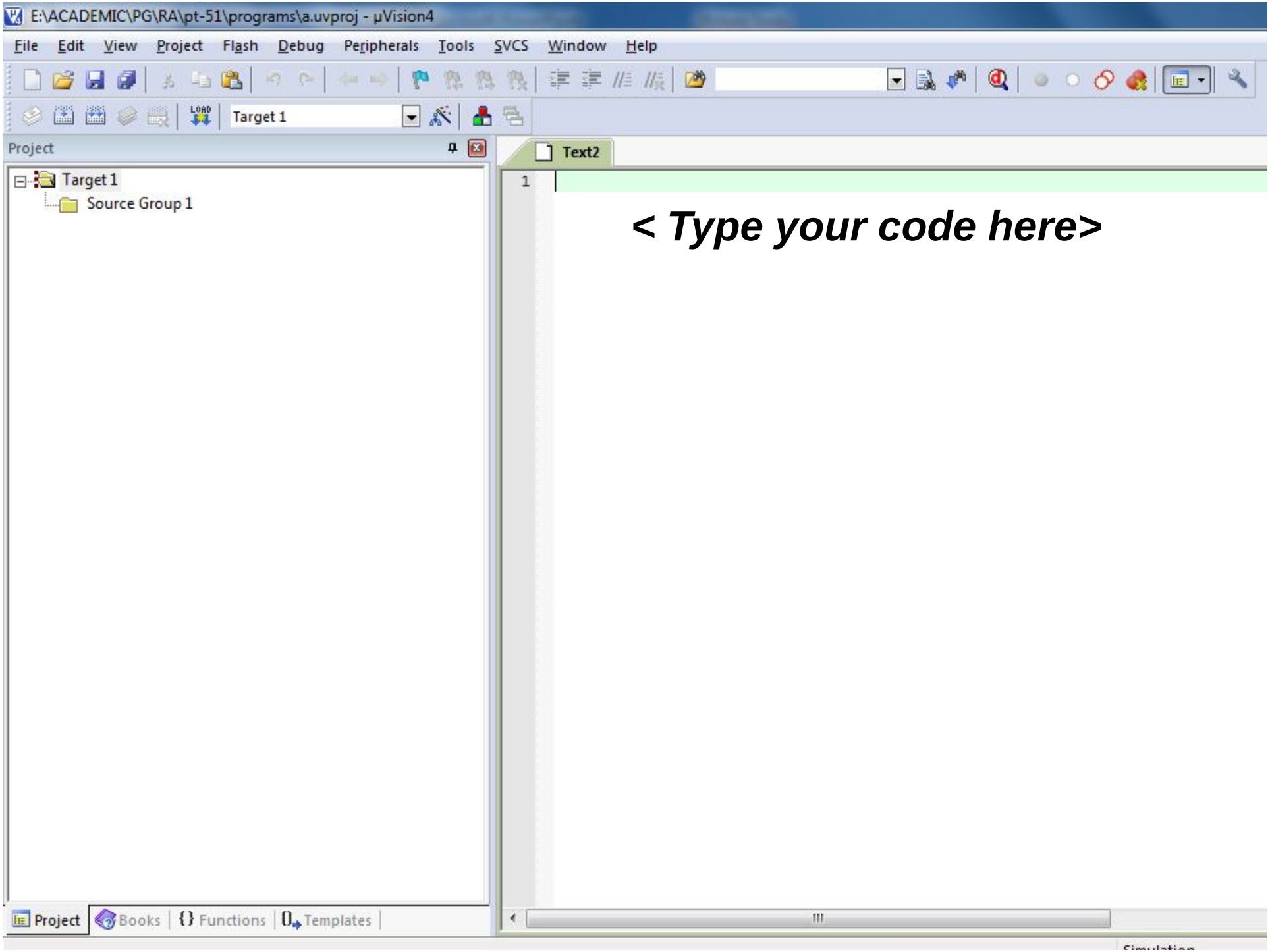
Selecting “Yes” includes an automatically generated startup code to the project.

Generally C source code requires startup file
to initialize global variables, Memory allocations, stack initialization etc.

Assembly code does not require startup file, So select no, if writing assembly code.



Go to File>New to create a new source file


E:\ACADEMIC\PG\RA\pt-51\programs\a.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
Text2
1
< Type your code here>
Project Books Functions Templates

E:\ACADEMIC\PG\RA\pt-51\programs\a.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Target 1

Project

Target 1
Source Group 1

simple_move.asm

```
1   src equ 30h
2   dst equ 40h
3   cnt equ 5
4
5   mov r0,#src
6   mov r1,#dst
7   mov r2,#cnt
8
9   back:   mov a,@r0
10          mov @r1,a
11          inc r0
12          inc r1
13
14  djnz r2,back
15
16  loop:   sjmp loop
17
18  end
```

Project | Books | Functions | Templates

Write the C or assembly code and save it as .c or .asm extension respectively.

E:\ACADEMIC\PG\RA\pt-51\programs\a.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Target 1

Project

Target 1

Source Group 1

Options for Group 'Source Group 1'... Alt+F7

Open File

Open List File

Open Map File

Rebuild all target files

Build target F7

Translate File

Stop build

Add Group...

Add Files to Group 'Source Group 1'...

Remove Group 'Source Group 1' and its Files

Manage Components...

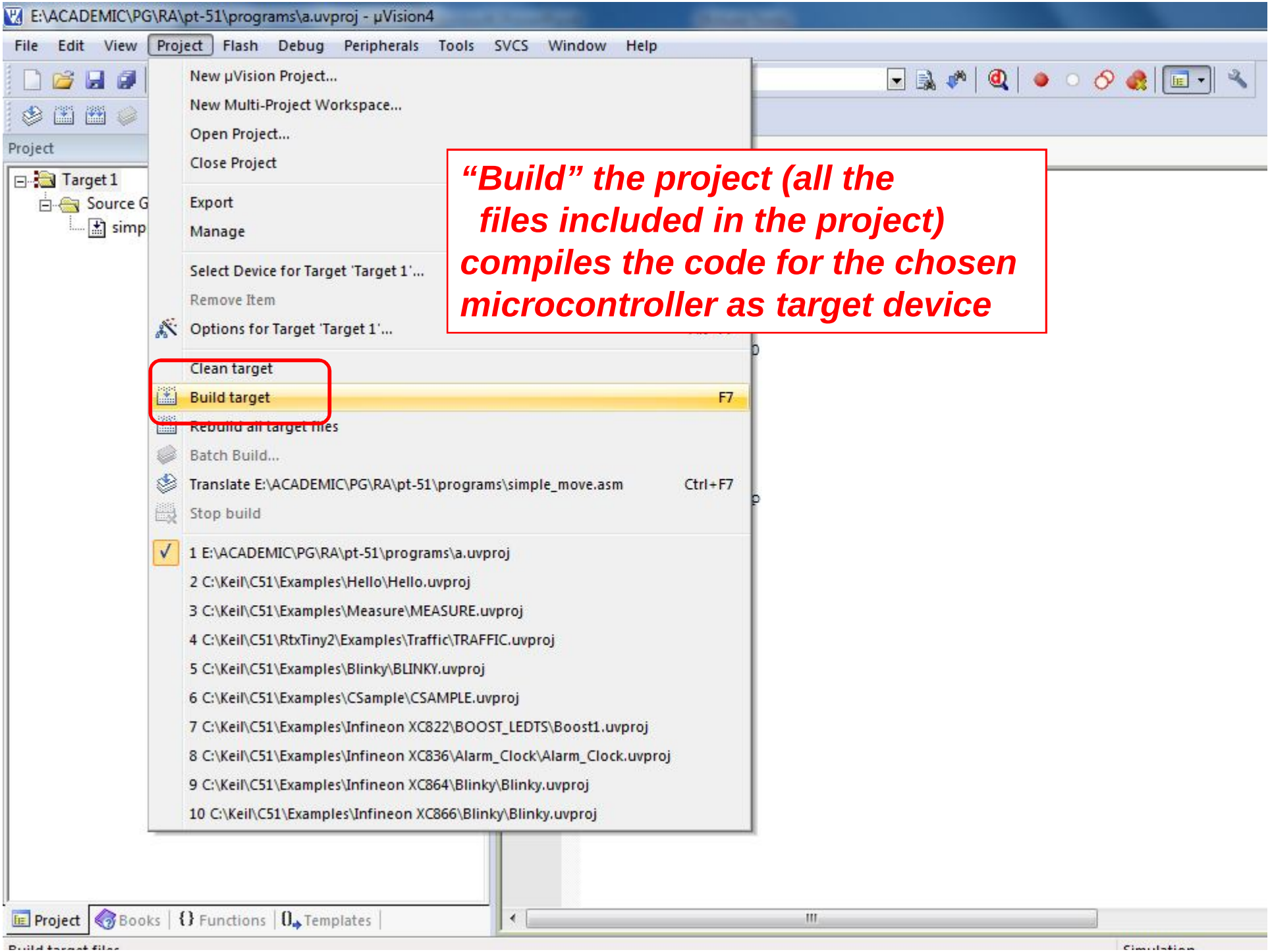
Show Include File Dependencies

simple_move.asm

```
1  src equ 30h
2  dst equ 40h
3  cnt equ 5
4
5  mov r0,#src
6  mov r1,#dst
7  mov r2,#cnt
8
9  back:   mov a,@r0
10         mov @r1,a
11         inc r0
12         inc r1
13
14 djnz r2,back
15
16 loop:   sjmp loop
17
18 end
```

Project Books Functions Templates

To compile/build the code you have to add the .c or .asm file to y project.



"Build" the project (all the files included in the project) compiles the code for the chosen microcontroller as target device

E:\ACADEMIC\PG\RA\pt-51\programs\a.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Target 1

Project

- Target 1
  - Source Group 1
    - simple_move.asm

Project | Books | Functions | Templates

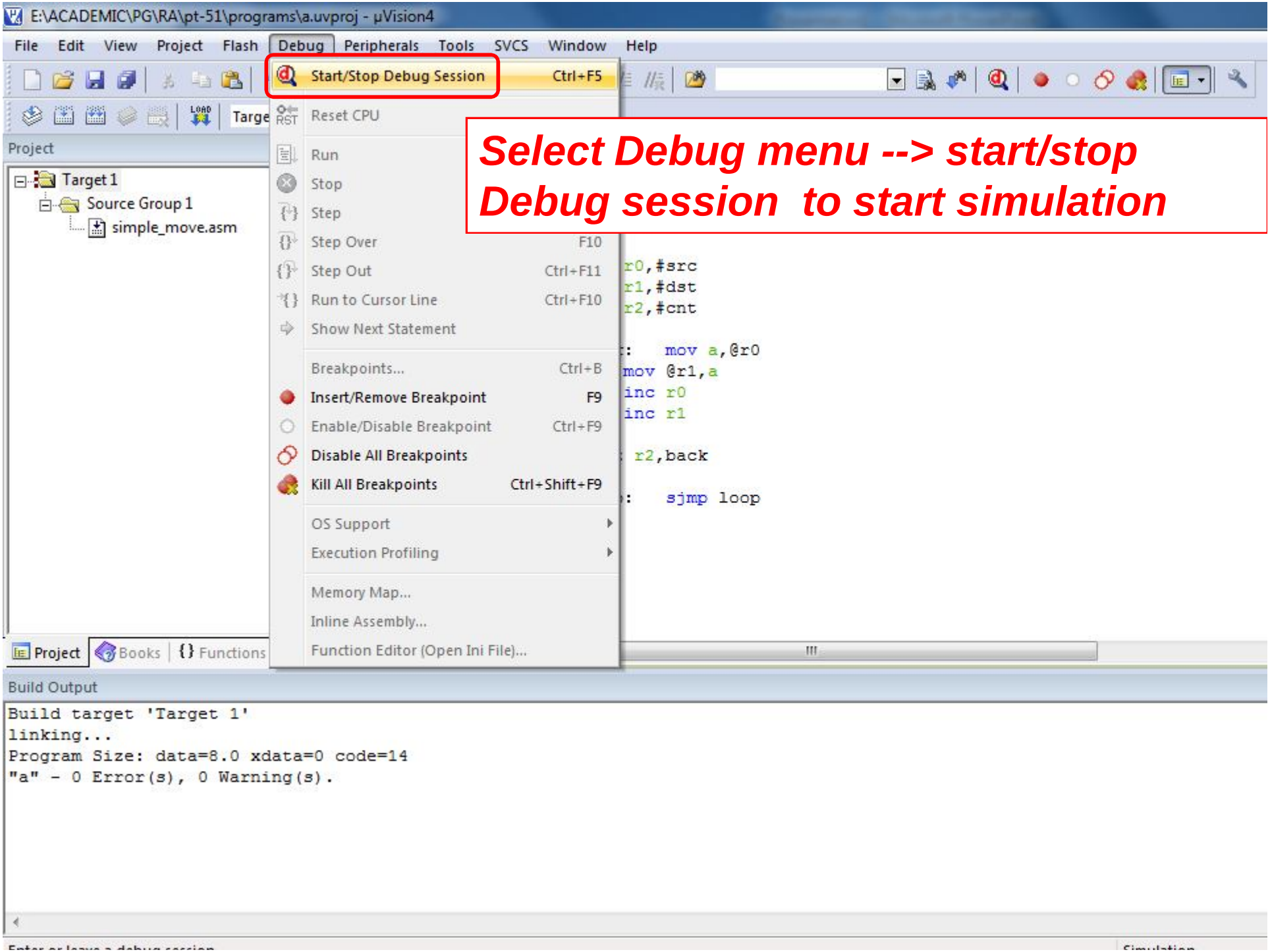
simple_move.asm

```
1   src equ 30h
2   dst equ 40h
3   cnt equ 5
4
5   mov r0,#src
6   mov r1,#dst
7   mov r2,#cnt
8
9   back:   mov a,@r0
10          mov @r1,a
11          inc r0
12          inc r1
13
14  djnz r2,back
15
16  loop:   sjmp loop
17
18  end
```

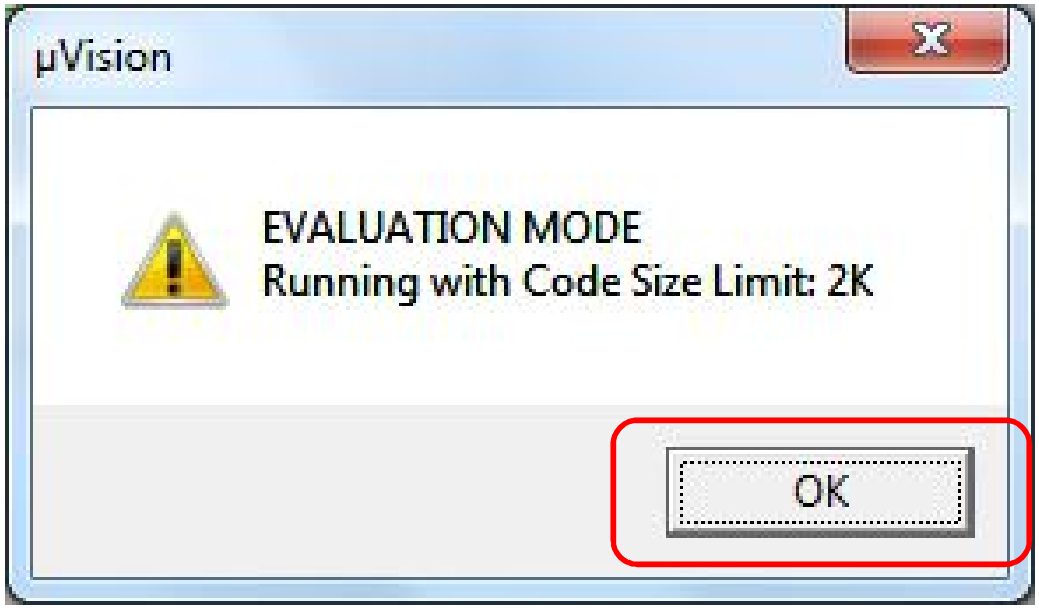
Build Output

```
Build target 'Target 1'
linking...
Program Size: data=8.0 xdata=0 code=14
"a" - 0 Error(s), 0 Warning(s).
```

Build status display, Errors & Warnings



Select Debug menu --> start/stop Debug session to start simulation



This is a free evaluation version of the development kit.

Clicking "OK" will take you into debug mode, where your code can be executed and the contents of various memory locations, ports and registers can be observed.

The execution can also be done in one-go or by single stepping.

E:\ACADEMIC\PG\RA\pt-51\programs\a.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

Project | Registers

Disassembly

```
  10:         mov @r1,a
:0x0007    F7       MOV      @R1,A
  11:         inc r0
:0x0008    08       INC      R0
  12:         inc r1
  13:
```

machine code and opcode

simple_move.asm

```
1  src equ 30h
2  dst equ 40h
3  cnt equ 5
4
5  mov r0,#src
6  mov r1,#dst
7  mov r2,#cnt
8
9  back:   mov a,@r0
10     mov @r1,a
11     inc r0
12     inc r1
13
14 djnz r2,back
```

Register Window

Command

```
Running with Code Size Limit: 2K
Load "E:\\ACADEMIC\\PG\\RA\\pt-51\\programs\\a"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Memory 1

Memory Window

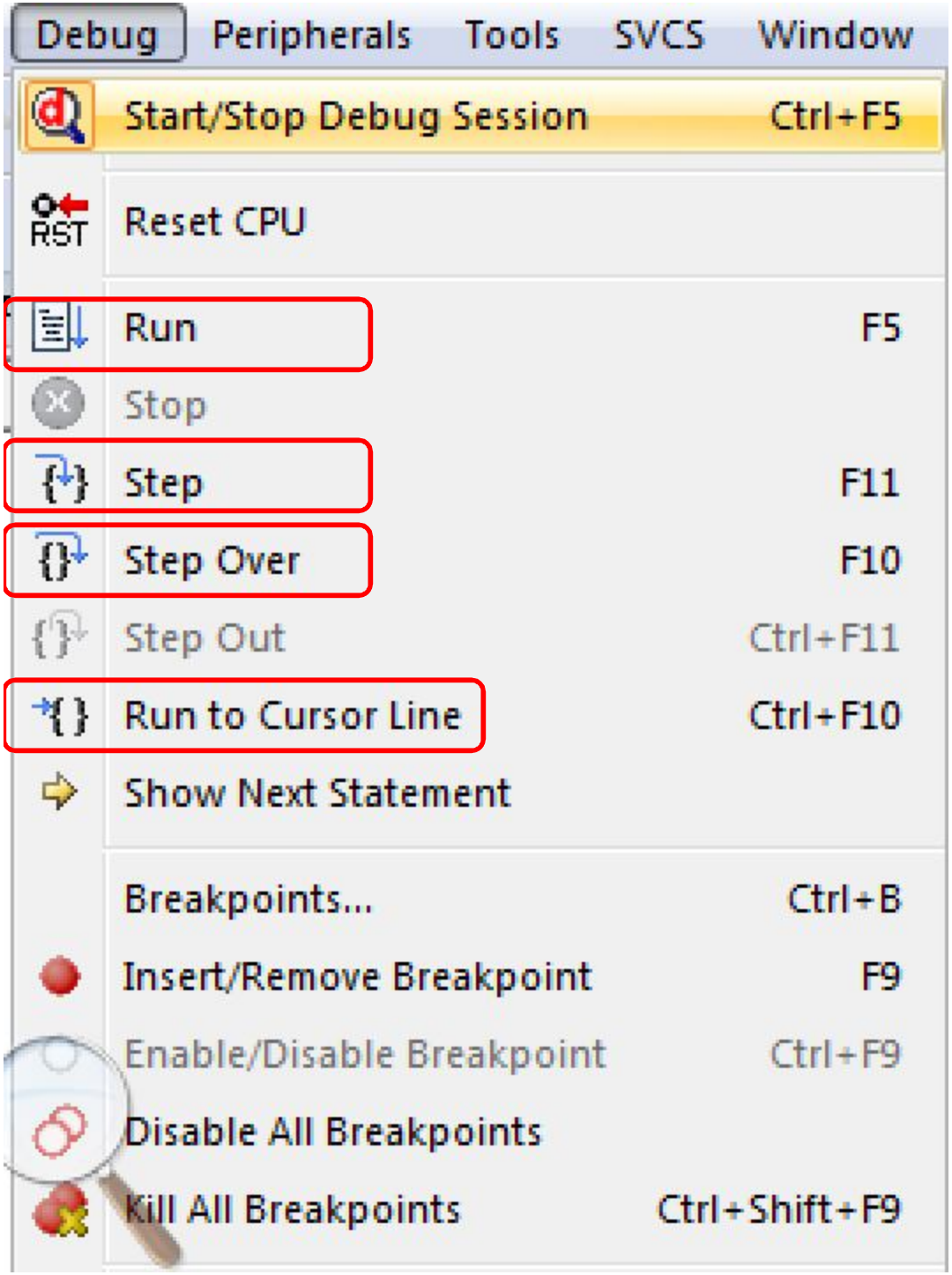
Address: d:00h

```
D:0x00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x18: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x48: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x78: 00 00 00 00 00 00 00 00 FF 07 00 00 00 00 00 10 00 00 00 00 00 00 0C 00
D:0x90: FF 00 00 00 F8 FF FF 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
```

Call Stack + Locals | Memory 1

Modify memory at cursor location | Simulation | t1: 0.00000000 sec | CAP NUM SCRL OVR R/W

10:27 PM 10/8/12



Execution can be done in different ways as can be seen in the Debug drop-down menu.

Memory 1

Address: d:00h

d: refers to data memory

```
D:0x00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x18: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x48: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x78: 00 00 00 00 00 00 00 00 FF 07 00 00 00 00 00 10 00 00 00 00 00 00 0C 00
D:0x90: FF 00 00 00 F8 FF FE 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
```

Call Stack + Locals | Memory 1

Simulation | t1: 0.00000000 sec | L:11 C:9 | CAP NUM SCRL OVR R/W

Memory 1

Address: c:00h

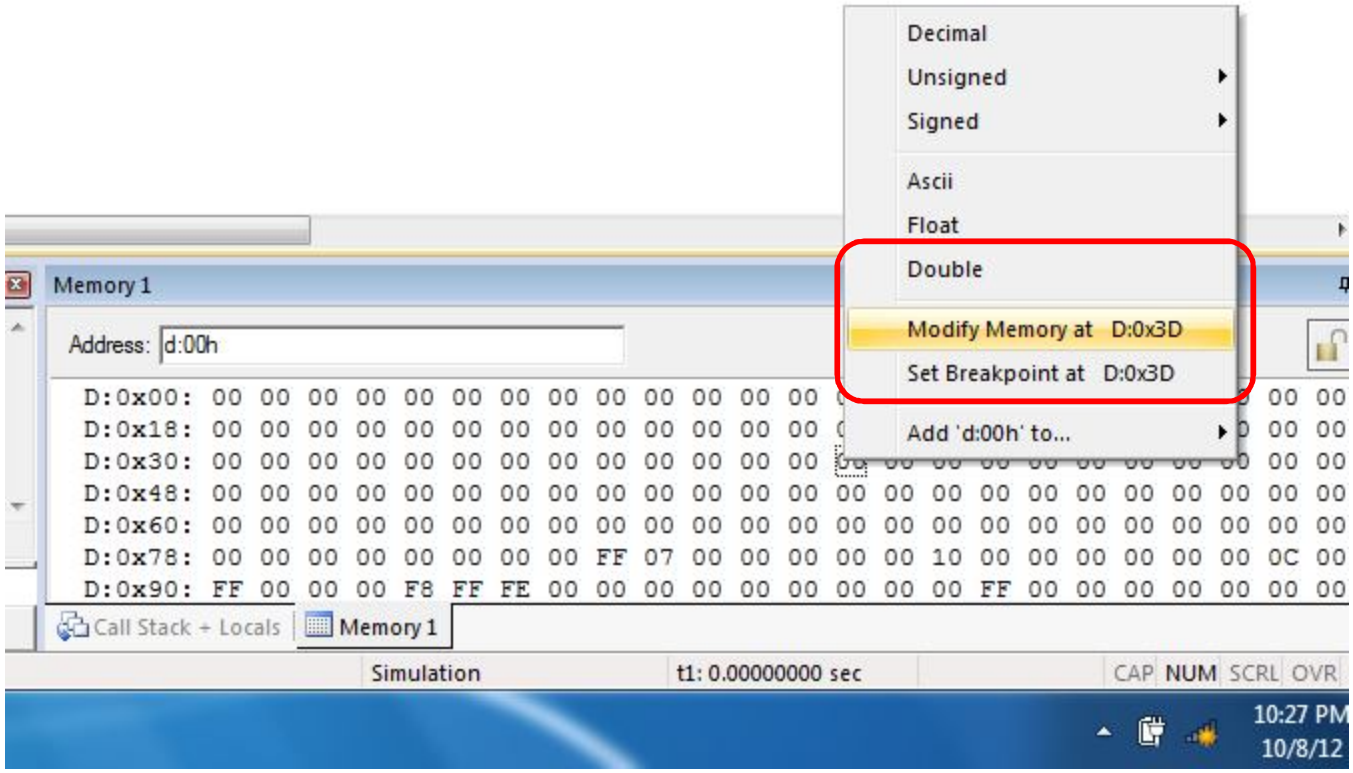
c: refers to code segment of the memory

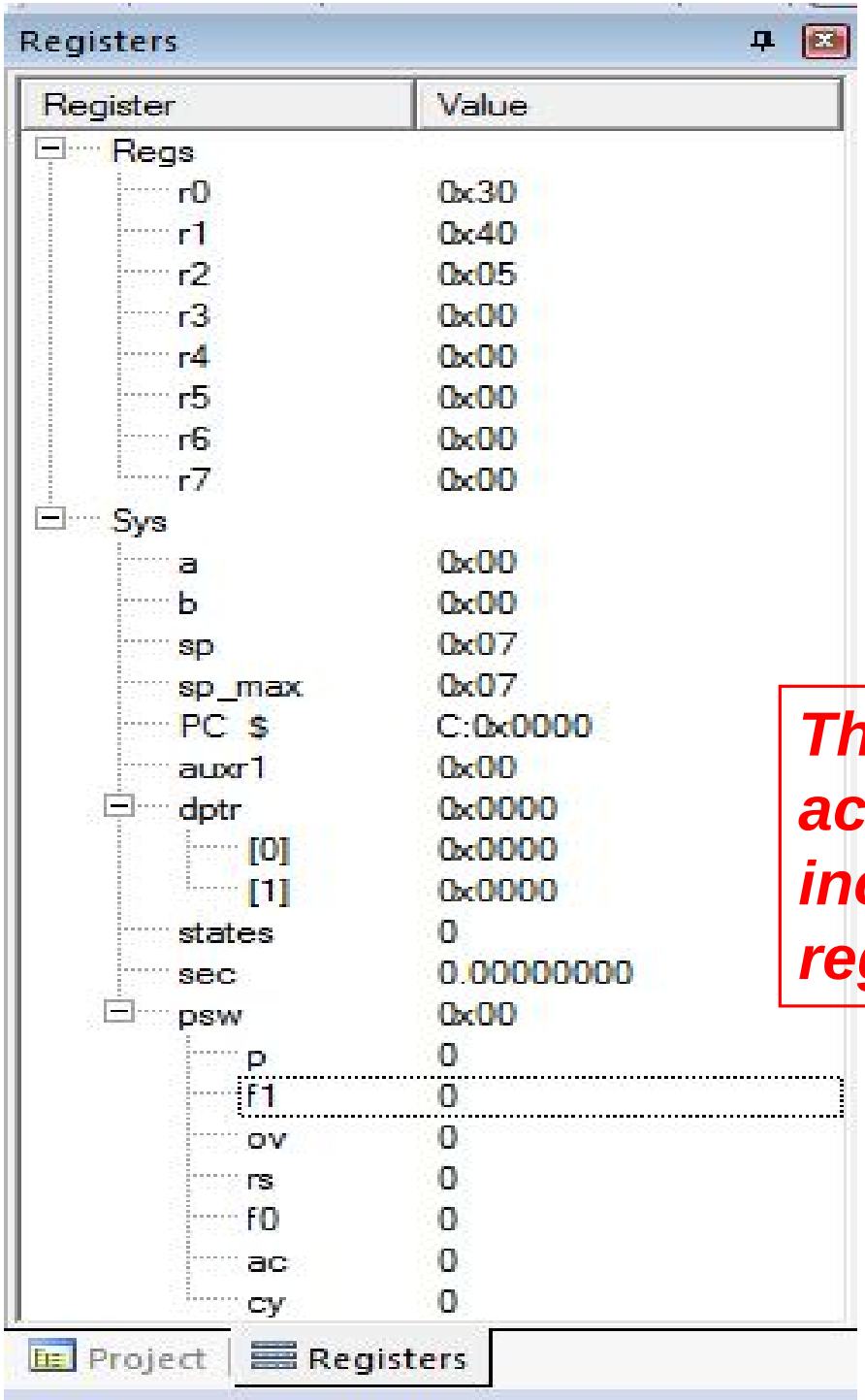
```
C:0x0000: 78 30 79 40 7A 05 E6 F7 08 09 DA FA 80 FE 00 00 00 00 00 00 00 00 00 00
C:0x0018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C:0x0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Call Stack + Locals | Memory 1

Simulation | t1: 0.00000000 sec | L:10 C:14 | CAP NUM SCRL OVR R/W

After starting the execution, user can right click on the required memory location in the memory window to modify RAM data.
You also have other functionalities like selecting the number system in which the memory contents are to be displayed.
Note: to initialize memory values on hardware, user has to add necessary instructions in the program code.





The Registers window provides access to all the registers including the flag register , DPTRs etc.

E:\ACADEMIC\PG\RA\pt-51\programs\a.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

- Status Bar
- Toolbars
- Project Window
- Books Window
- Functions Window
- Templates Window
- Source Browser Window
- Build Output Window
- Find In Files Window
- Command Window
- Disassembly Window
- Symbol Window
- Registers Window
- Call Stack Window
- Watch Windows
- Memory Windows
  - Memory 1
  - Memory 2
  - Memory 3
  - Memory 4
- Serial Windows
- Analysis Windows
- Trace
- System Viewer
- Toolbox Window
- Periodic Window Update

```
 5: mov r0,#src
0000    7830    MOV    R0,#0x30
 6: mov r1,#dst
0002    7940    MOV    R1,#0x40
 7: mov r2,#cnt
 8:
```

simple_move.asm

```
src equ 30h
dst equ 40h
cnt equ 5

mov r0,#src
mov r1,#dst
mov r2,#cnt
```

If some windows are not being displayed then use the "View" menu to get them on the window.

Command

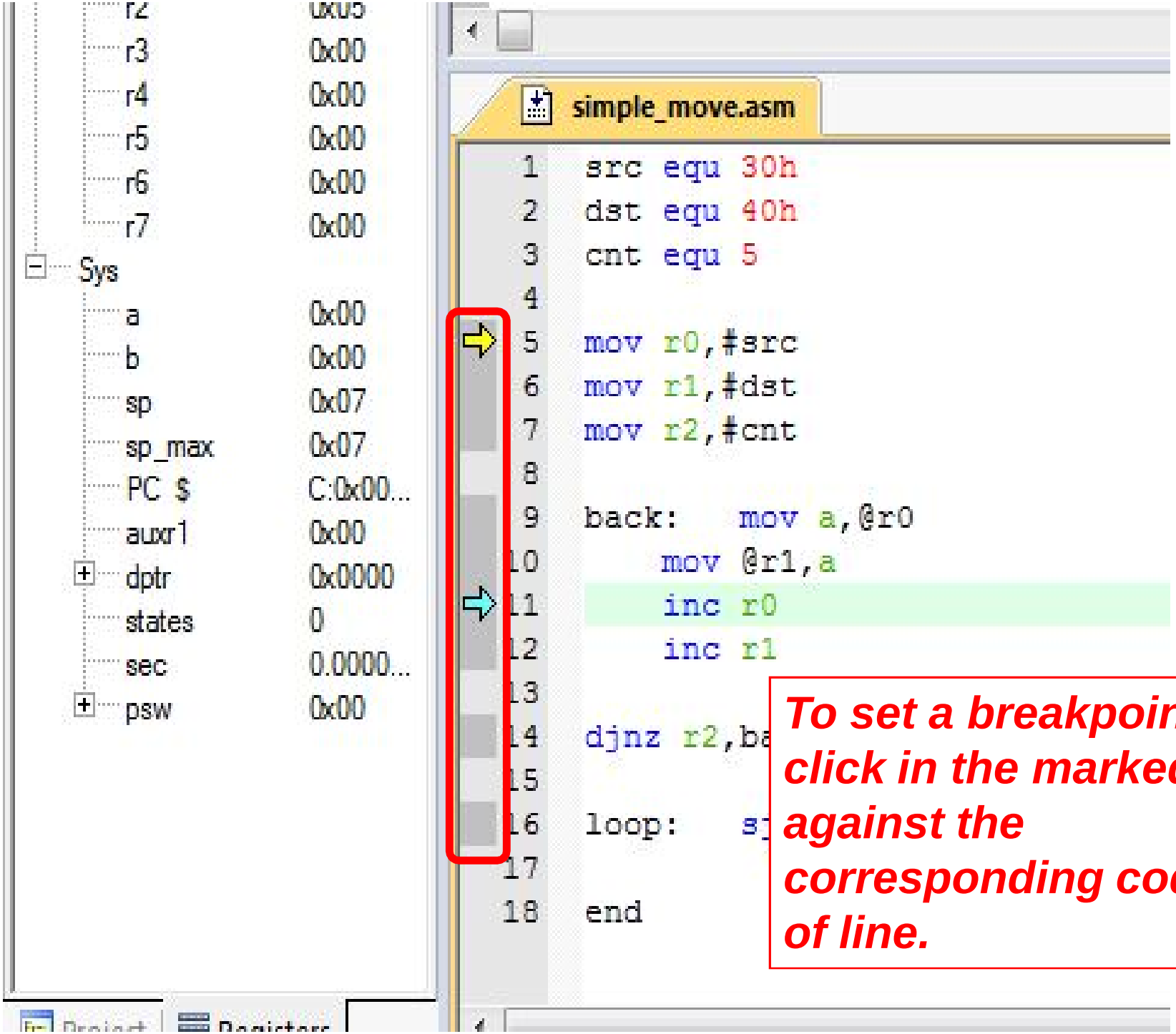
```
Load "E:\\ACADEMIC\\PG\\RA\\pt-51\\programs\\a"
```

Memory 1

Address: d:00h

```
D:0x00: 35 45 00 00 00 00 00 00
D:0x18: 00 00 00 00 00 00 00 00
D:0x30: 00 00 00 00 00 00 00 00
D:0x48: 00 00 00 00 00 00 00 00
D:0x60: 00 00 00 00 00 00 00 00
D:0x78: 00 00 00 00 00 00 00 00
D:0x90: FF 00 00 00 F8 FF FE 00
```

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess



To set a breakpoint, click in the marked area against the corresponding code of line.

Registers

Register	Value
Regs	
r0	0x30
r1	0x40
r2	0x05
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x00...
auxr1	0x00
dptr	0x0000
states	0
sec	0.0000...
psw	0x00

Disassembly

```
    11:          inc r0
C:0x0008     08        INC       R0
    12:          inc r1
    13:
C:0x0009     09        INC       R1
```

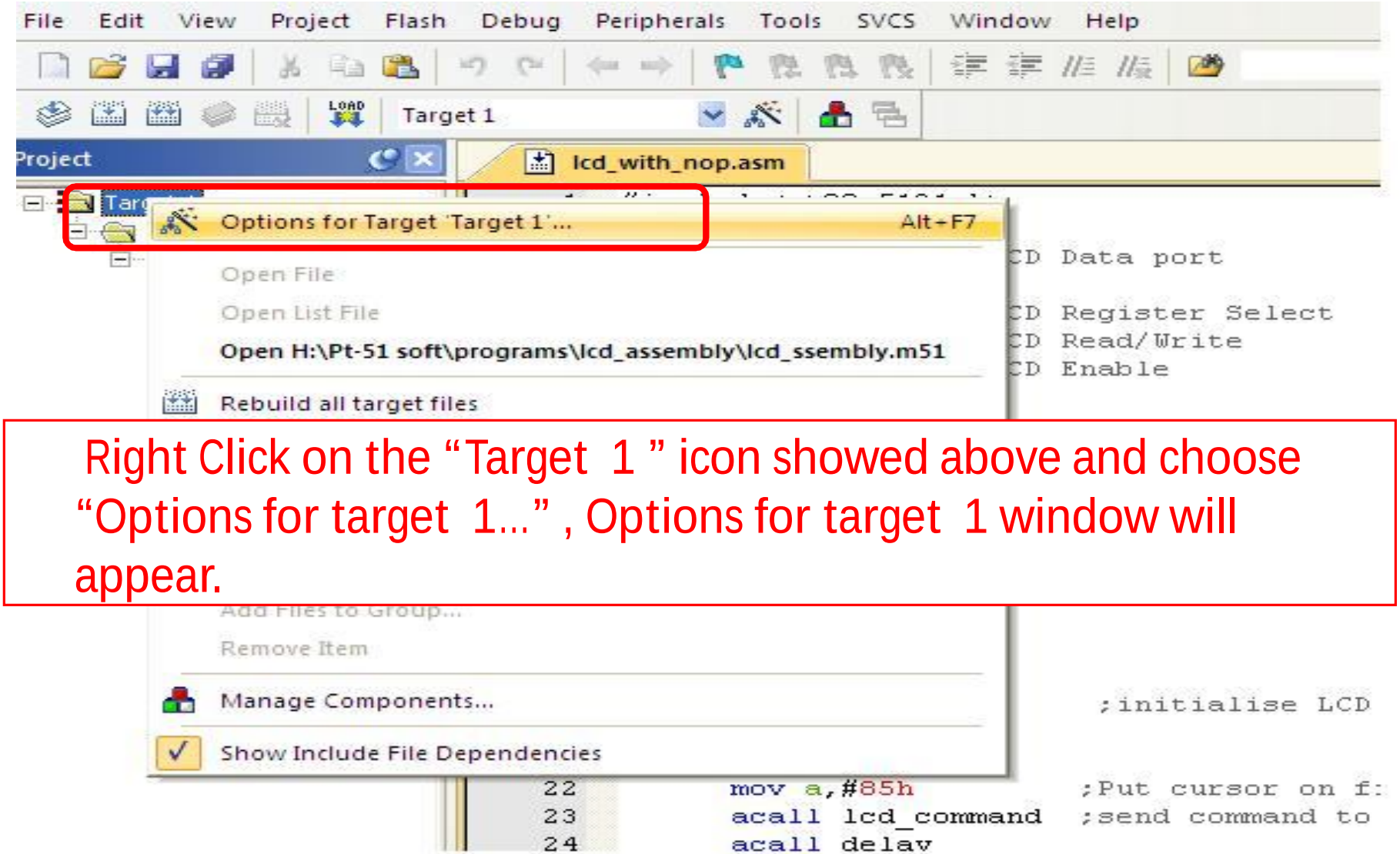
simple_move.asm

```
1   src equ 30h
2   dst equ 40h
3   cnt equ 5
4
5   mov r0,#src
6   mov r1,#dst
7   mov r2,#cnt
8
9   back:    mov a,
10       mov @r1,a
11       inc r0
12       inc r1
13
14  djnz r2,back
15
16  loop:    sjmp loop
17
18  end
```

The breakpoint set is shown as a red dot against the line.

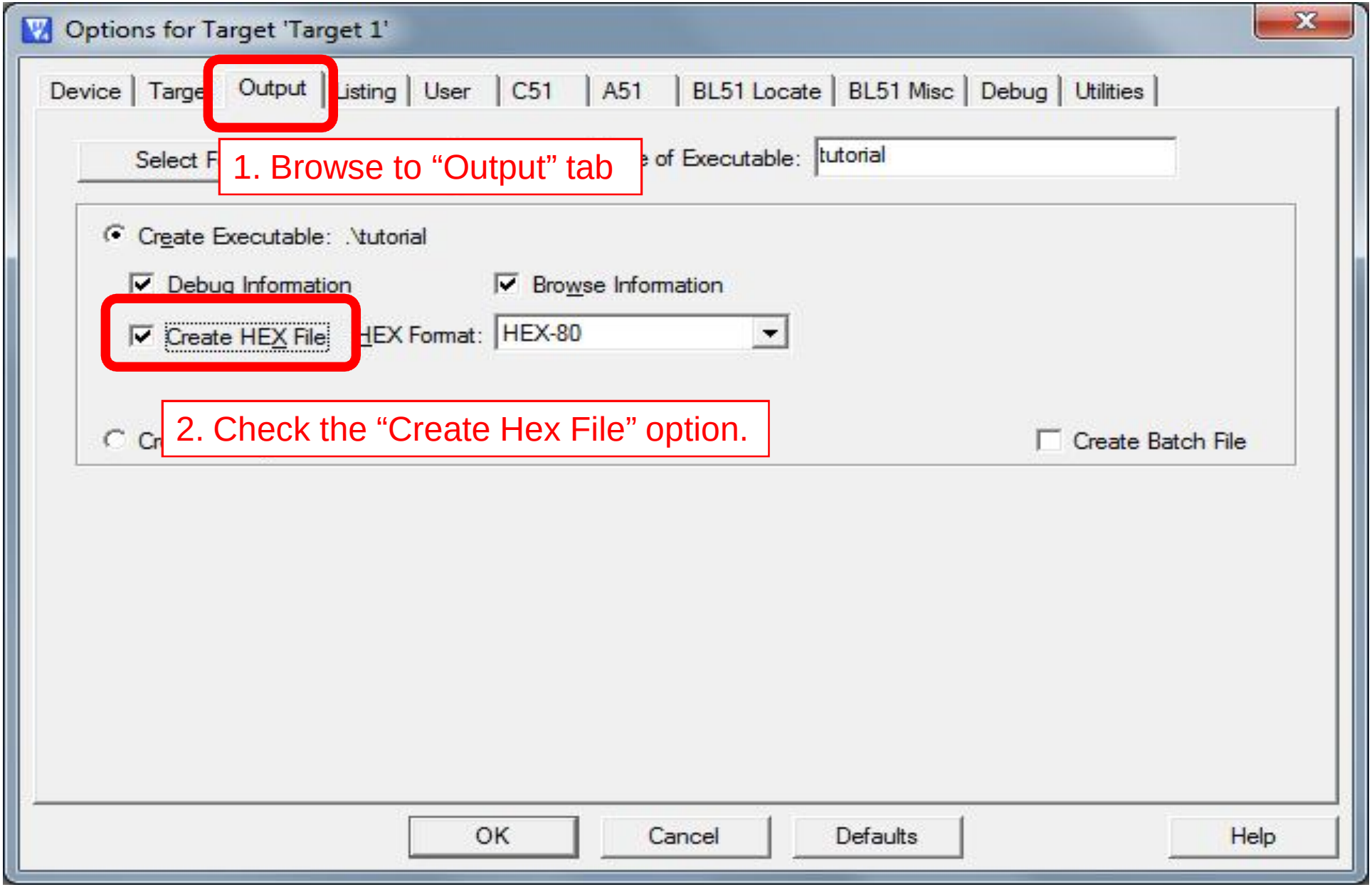
As can be seen, the breakpoint is automatically displayed at the equivalent line in the disassembly window too.

To generate HEX file



Right Click on the "Target 1" icon showed above and choose "Options for target 1…", Options for target 1 window will appear.

“Options for target 1…” window



```
54          else
55            led2=0;      //off led2
56
57          if(SW3==0)     //chech switch3
58            led3=1;      //on led3
59          else
60            led3=0;      //off led3
61
62          if(SW4==0)     //check switch4
```

Pr... B... F... Te...

Build Output

```
compiling lcd.c...
compiling delay.c...
linking...
Program Size: data=12.0 xdata=0 code=362
creating hex file from "89c5131_LCD"...
"89c5131_LCD" - 0 Error(s), 0 Warning(s).
```

The HEX file generated, is by default, given the name of the Project and stored in the project folder