
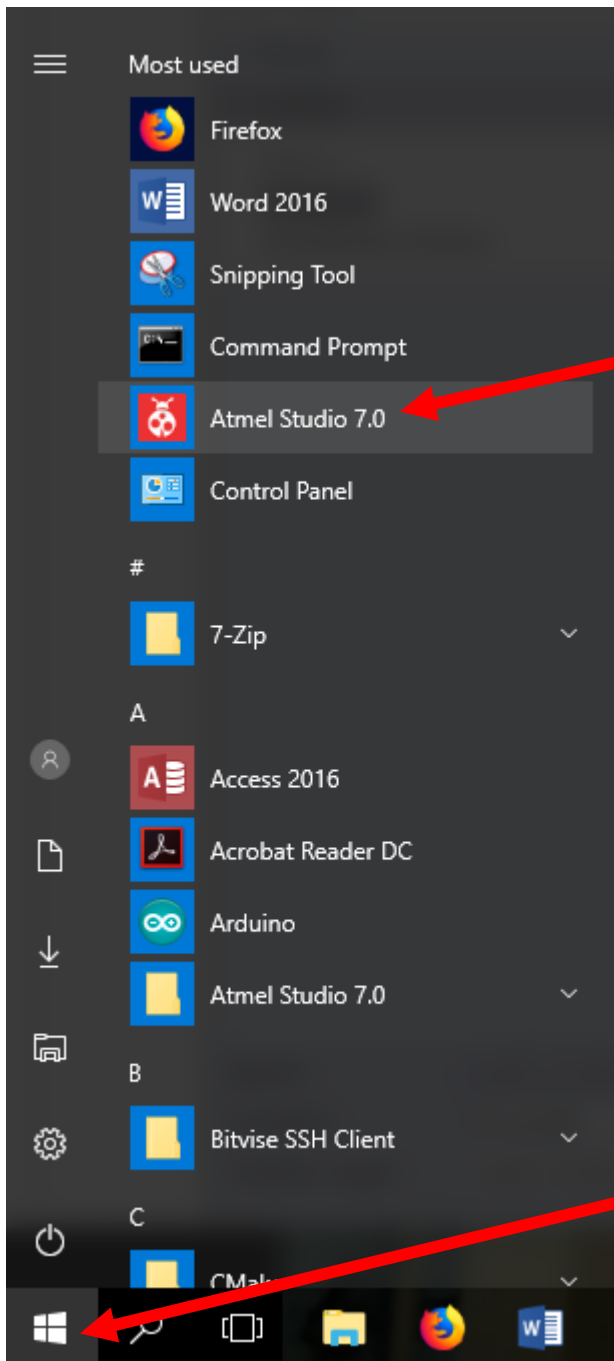


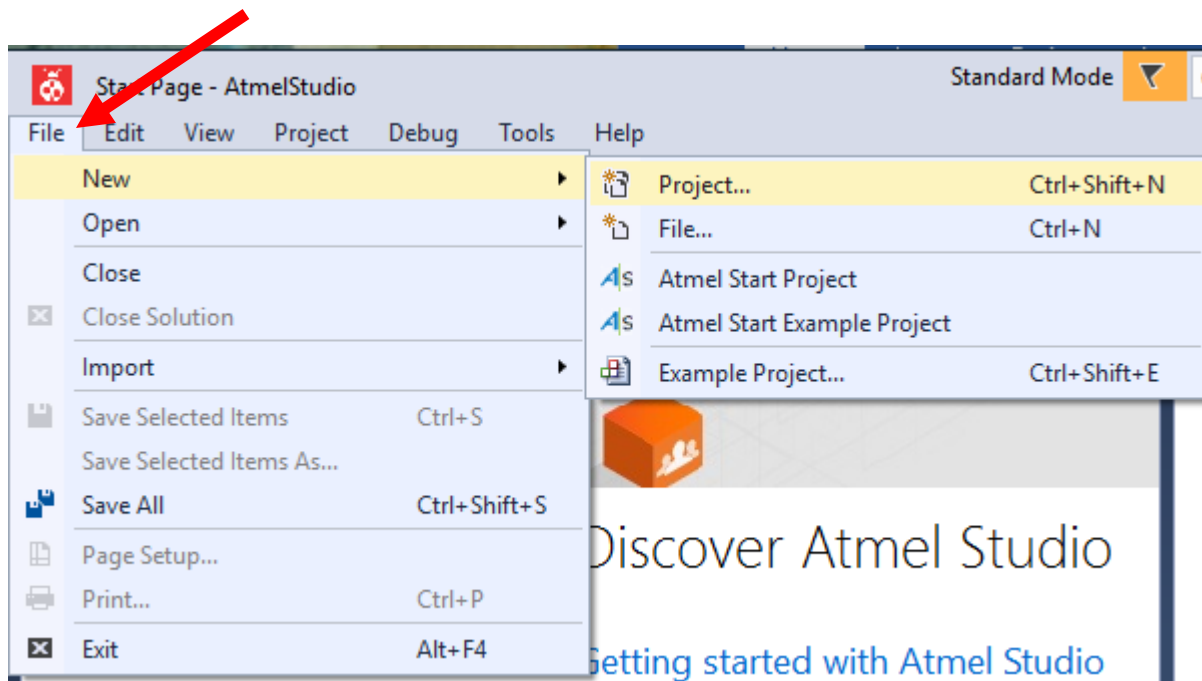
## Lab 2 Introduction to Assembly Language

### I. Introduction to Atmel Studio 7.0<sup>1</sup>

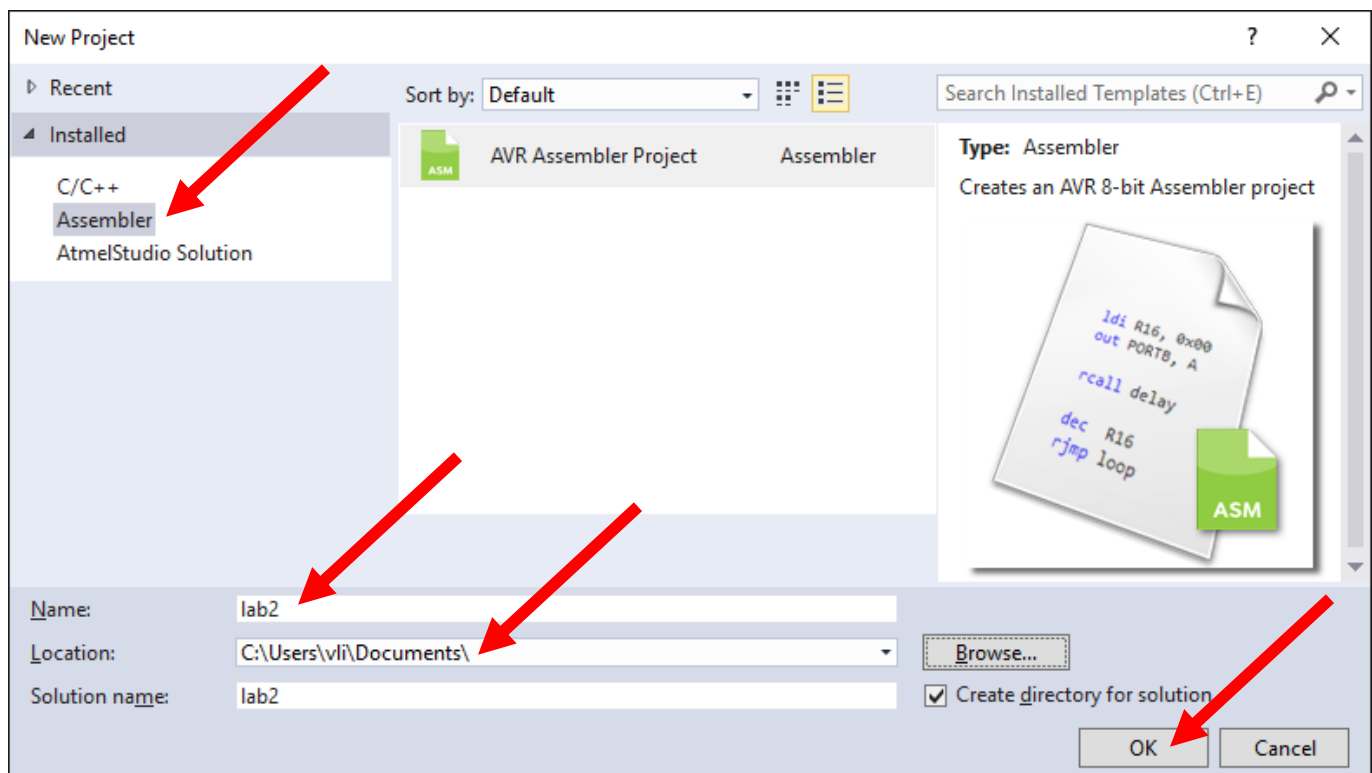
Launch *Atmel Studio 7.0* and create a new project named “lab2”: click on the *Start*  button, then click on *Atmel Studio 7.0*:



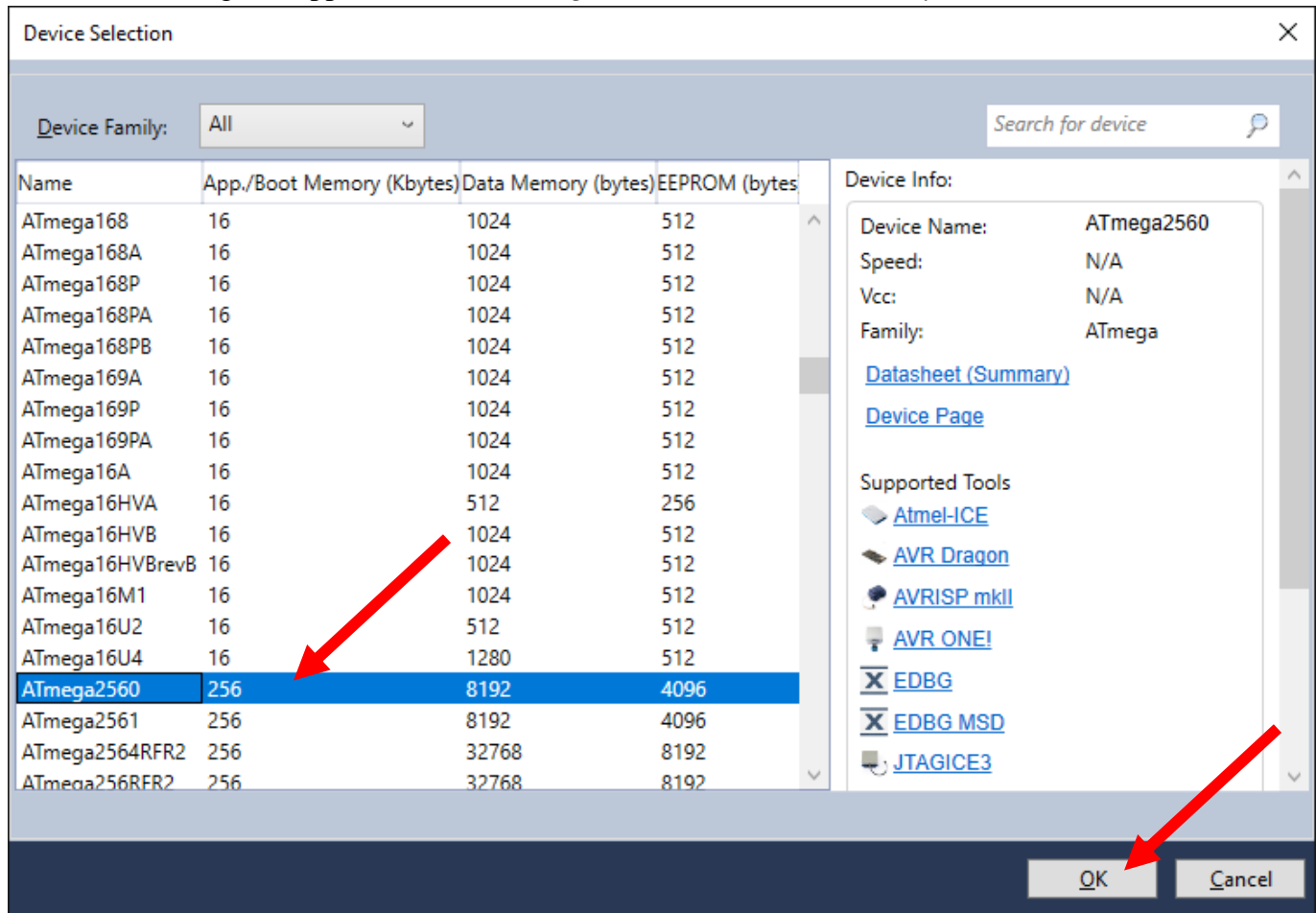
Create a new project named *lab2*: on the menu, click on File -> New -> Project:



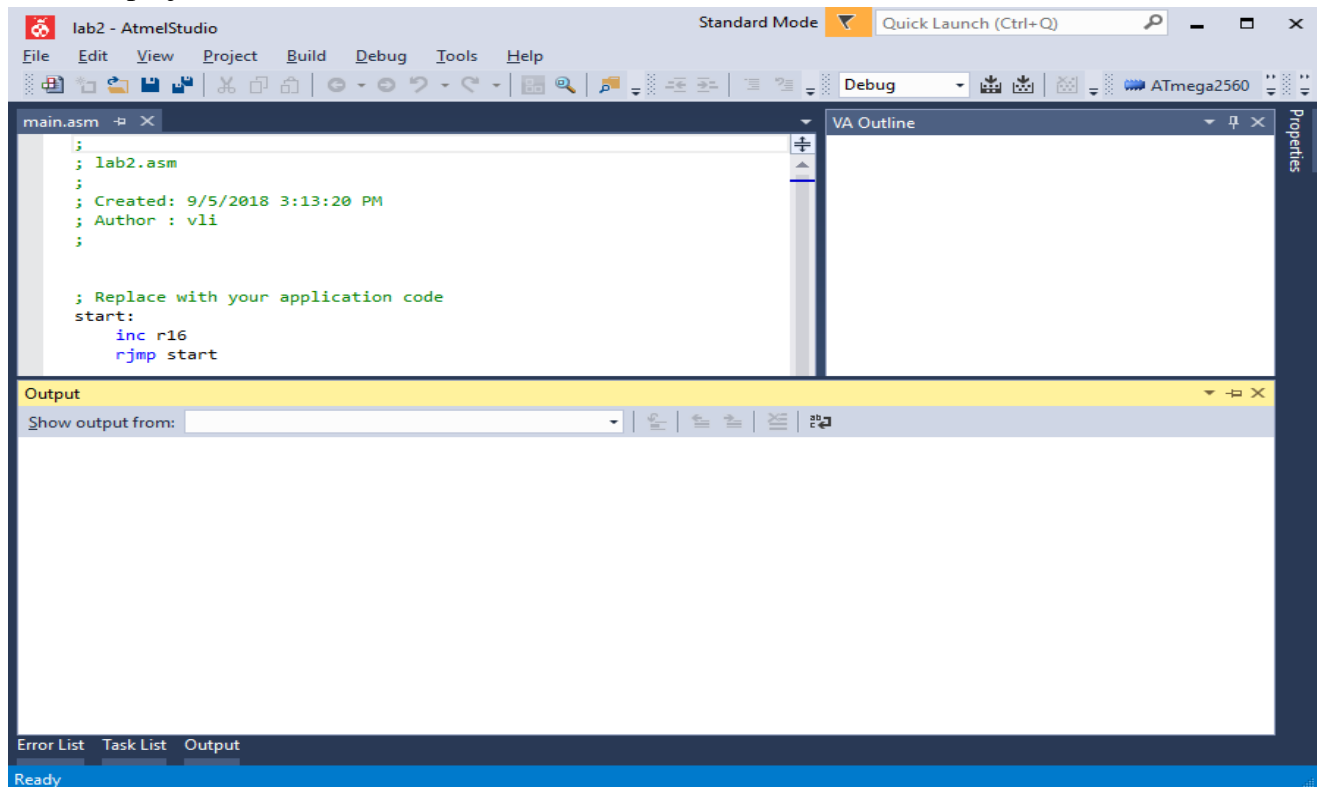
In the new dialog box, on the left pane, under *Installed*, select *Assembler*. Type the project Name *lab2* and select the Location. Click on the *OK* button.



Then a new dialog box appears, choose *ATmega2560* for the *Device Family*. Click on the *OK* button.



The new project looks like this:



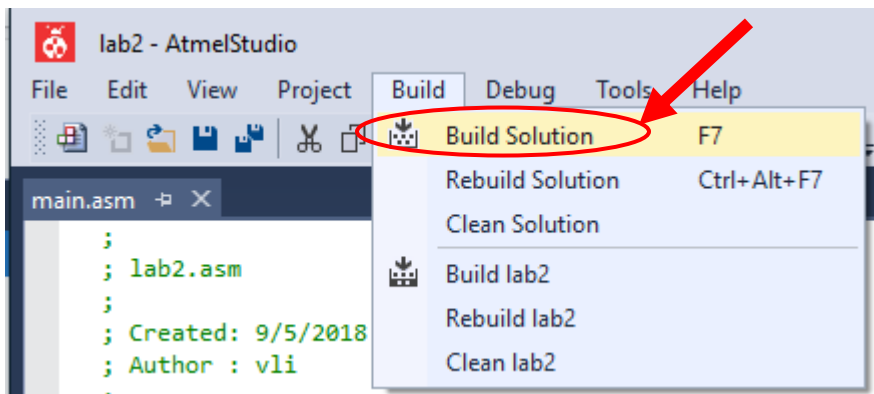
Type the following code:

```
.cseg ;select current segment as code
.org 0 ;begin assembling at address 0 of the flash memory

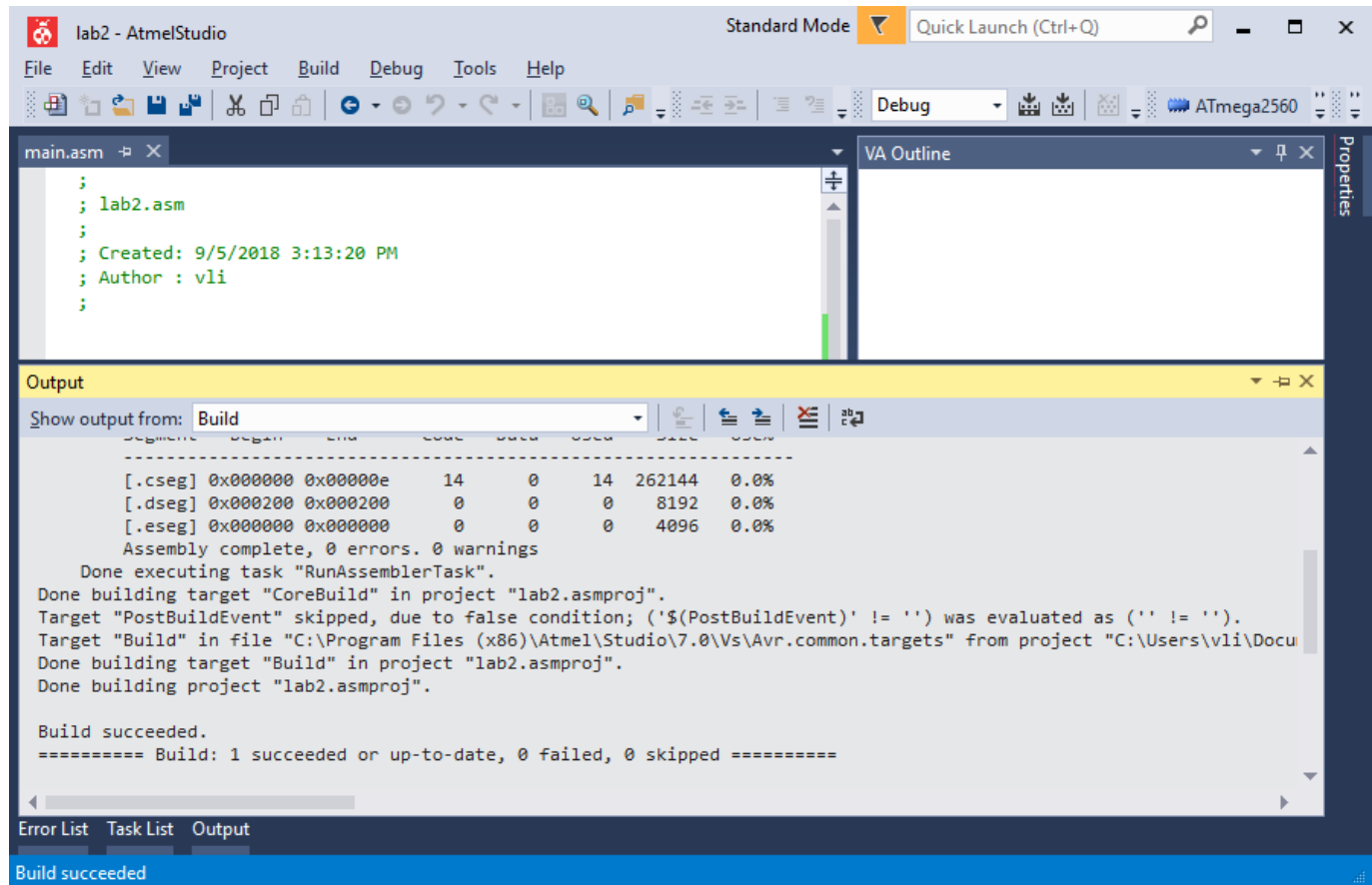
;define symbolic names for resources (e.g. registers) used
.def count = r16 ;register 16 holds a number

ldi count, 1 ;initialize count to 1
lp:
    inc count ;increment the counter
    cpi count, 0x05
    breq done
    rjmp lp
done: jmp done
```

Save the code and build the program: on the menu, click on *Build -> Build Solution*:

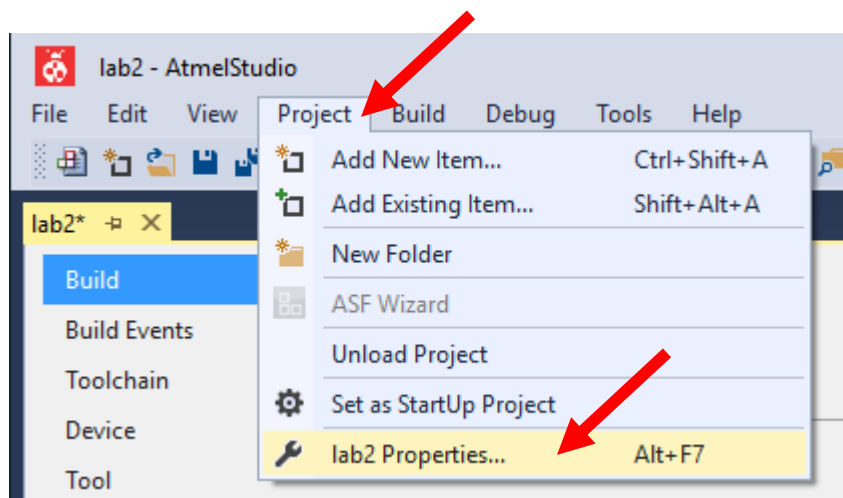


The screen looks like this:

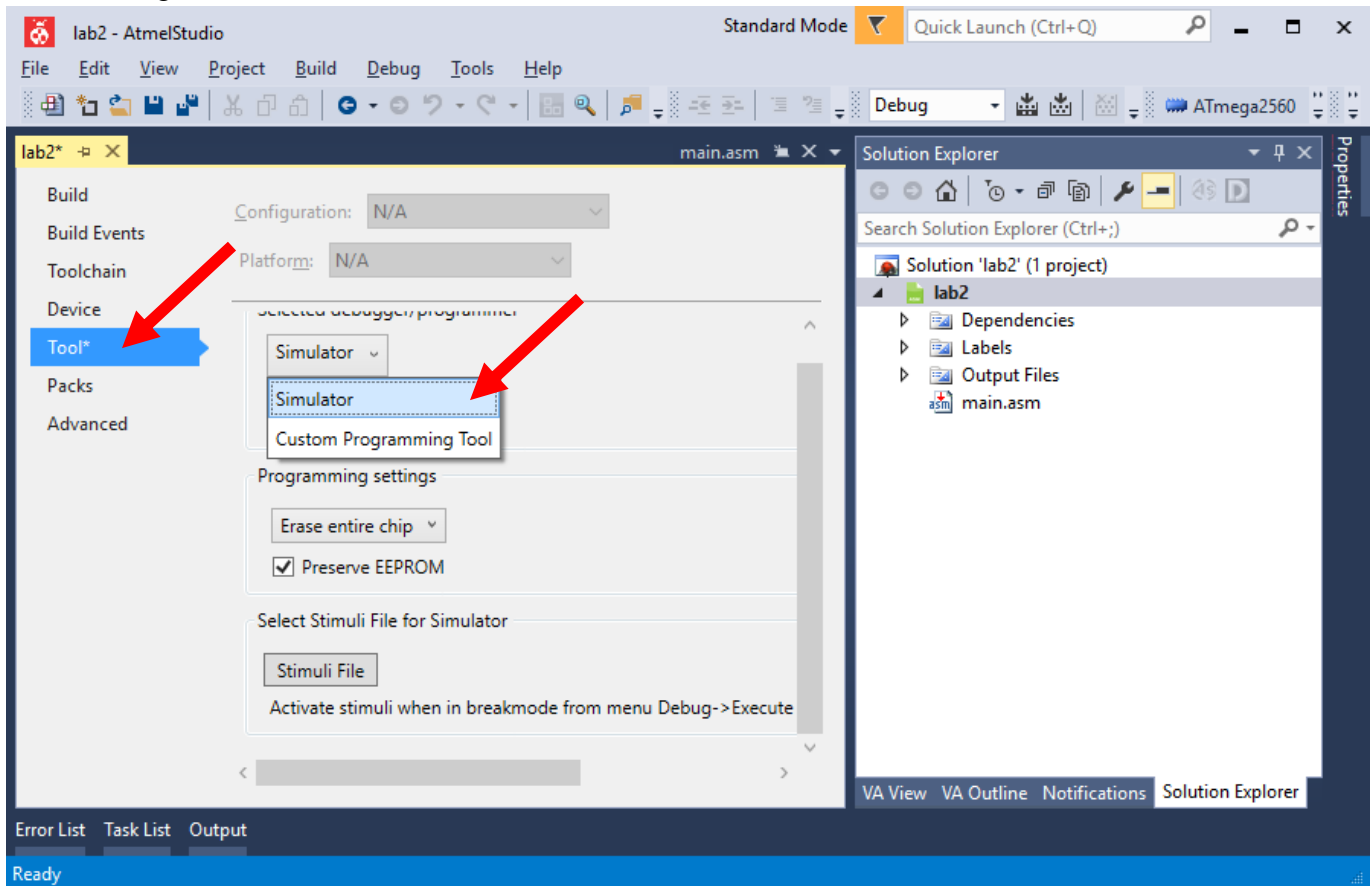


If there are any errors, you must fix them and rebuild your program until no build errors.

Now, you are ready to run your program using the simulator. Set up the configuration of the simulator: from the *Project* menu -> *lab2 Properties...*:

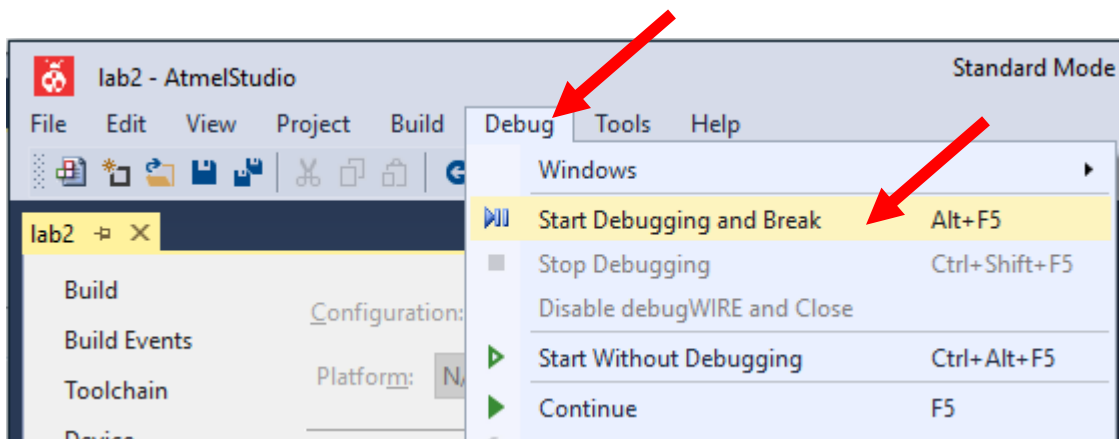


In the settings window, click on Tool and select Simulator:

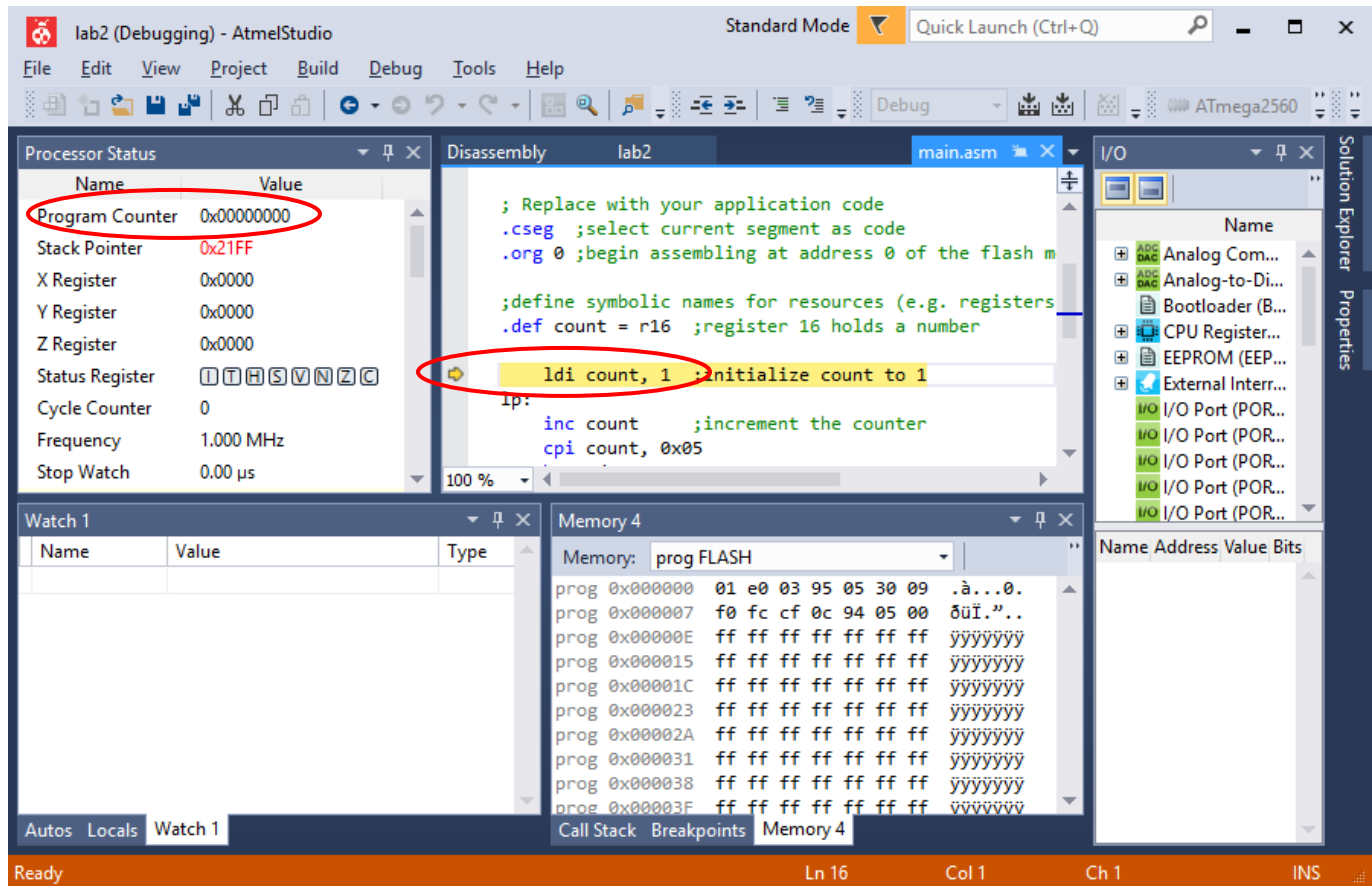


Save the project.

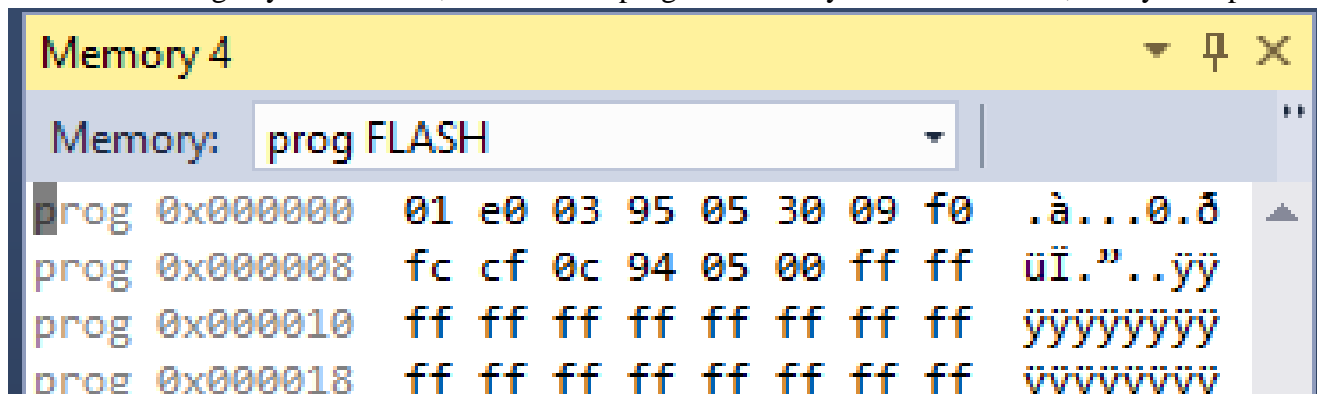
Start the simulator: from the menu, click on *Debug -> Start Debugging and Break*



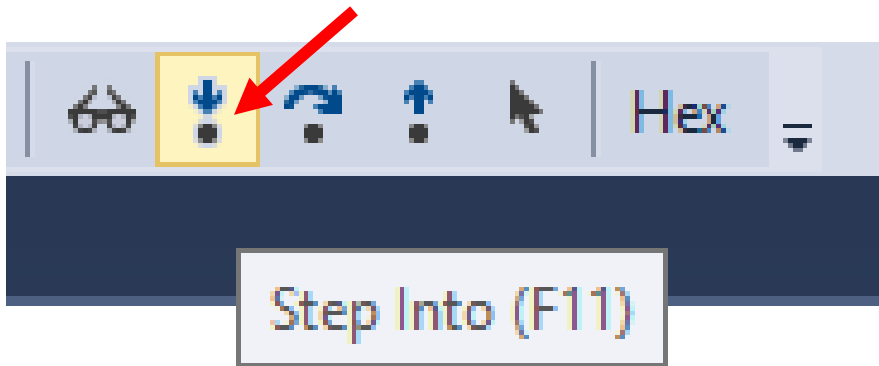
The editor should show a yellow arrow indicating the instruction about to be fetched. The left panel shows the “*Processor Status*”, where you can examine the register and processor values. The “*Program Counter*” shows the memory address of the next instruction to be fetched.



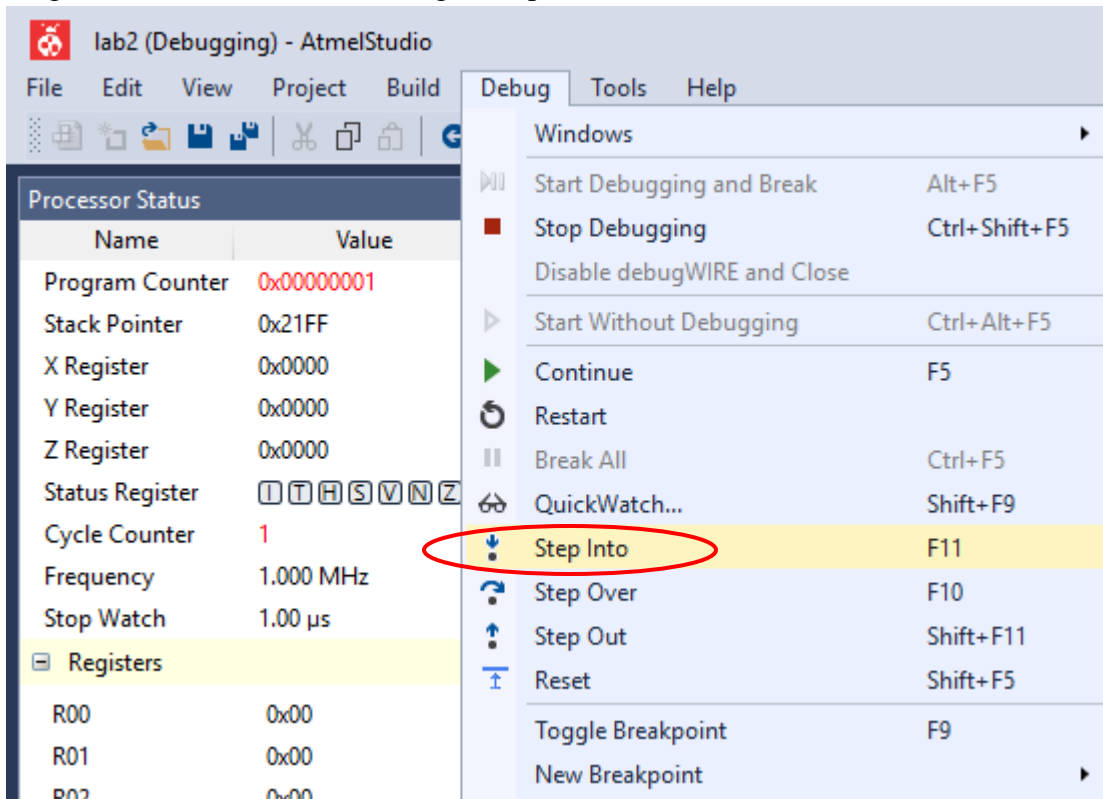
Before executing any instructions, examine the program memory. It looks like this, verify the opcode:



Fetch and execute the first instruction: click on the *Step Into* button (or press the F11 key)



Or go to the menu, click on Debug ->Step Into:





After executing the first instruction, the value in register 16 (R16) is changed from 0x00 to 0x01. The changed values are in red, such as the Program Counter, Cycle Counter, in addition to R16:

The screenshot shows the AtmelStudio interface during a debugging session. The 'Processor Status' window on the left lists various system variables and registers. The 'Disassembly' window on the right shows the assembly code being executed.

| Name             | Value           |
|------------------|-----------------|
| Program Counter  | 0x00000001      |
| Stack Pointer    | 0x21FF          |
| X Register       | 0x0000          |
| Y Register       | 0x0000          |
| Z Register       | 0x0000          |
| Status Register  | 1 1 1 1 1 1 1 1 |
| Cycle Counter    | 1               |
| Frequency        | 1.000 MHz       |
| Stop Watch       | 1.00 µs         |
| <b>Registers</b> |                 |
| R00              | 0x00            |
| R01              | 0x00            |
| R02              | 0x00            |
| R03              | 0x00            |
| R04              | 0x00            |
| R05              | 0x00            |
| R06              | 0x00            |
| R07              | 0x00            |
| R08              | 0x00            |
| R09              | 0x00            |
| R10              | 0x00            |
| R11              | 0x00            |
| R12              | 0x00            |
| R13              | 0x00            |
| R14              | 0x00            |
| R15              | 0x00            |
| R16              | 0x01            |

```

; Replace with yo
.cseg ;select cu
.org 0 ;begin ass

;define symbolic
.def count = r16

    ldi count, 1
lp:
    inc count
    cpi count, 0x
    breq done
    rjmp lp
done: jmp done
  
```

Stop the debugging session by using the “Stop Debugging” command under the “Debug” menu.

**II. Write some opcodes and verify them using the opcodes in the memory. Are they big-endian or little-endian? Choose one line of code and figure out its opcode.**

**III. Write a program called lab1.asm.** The program calculates the number of students in csc189. There are two lab sections B01 and B02. The number of students registered:

B01: 23

B02: 21

The maximum enrollment for the course is 60. If the course enrollment is bigger than 60, set register 0 to 1, 0 otherwise. Store the course enrollment in register 19.

In high level programming language, it can be done like this:

```
unsigned int x=23
unsigned int y=21
unsigned int sum=x+y
boolean over_enrollment=false
if (sum>60)
    over_enrollment=true
```

**No submissions for the lab.**

1. Adapted from the lab notes written by Dr. Bill Bird for csc 230 in the summer of 2018.