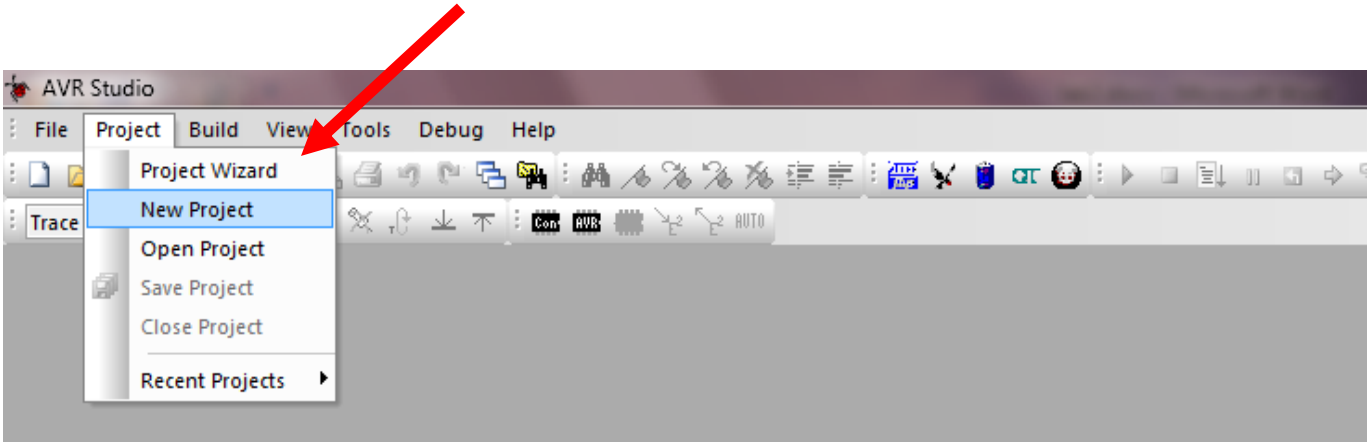
Lab 2 Introduction to Assembly Language

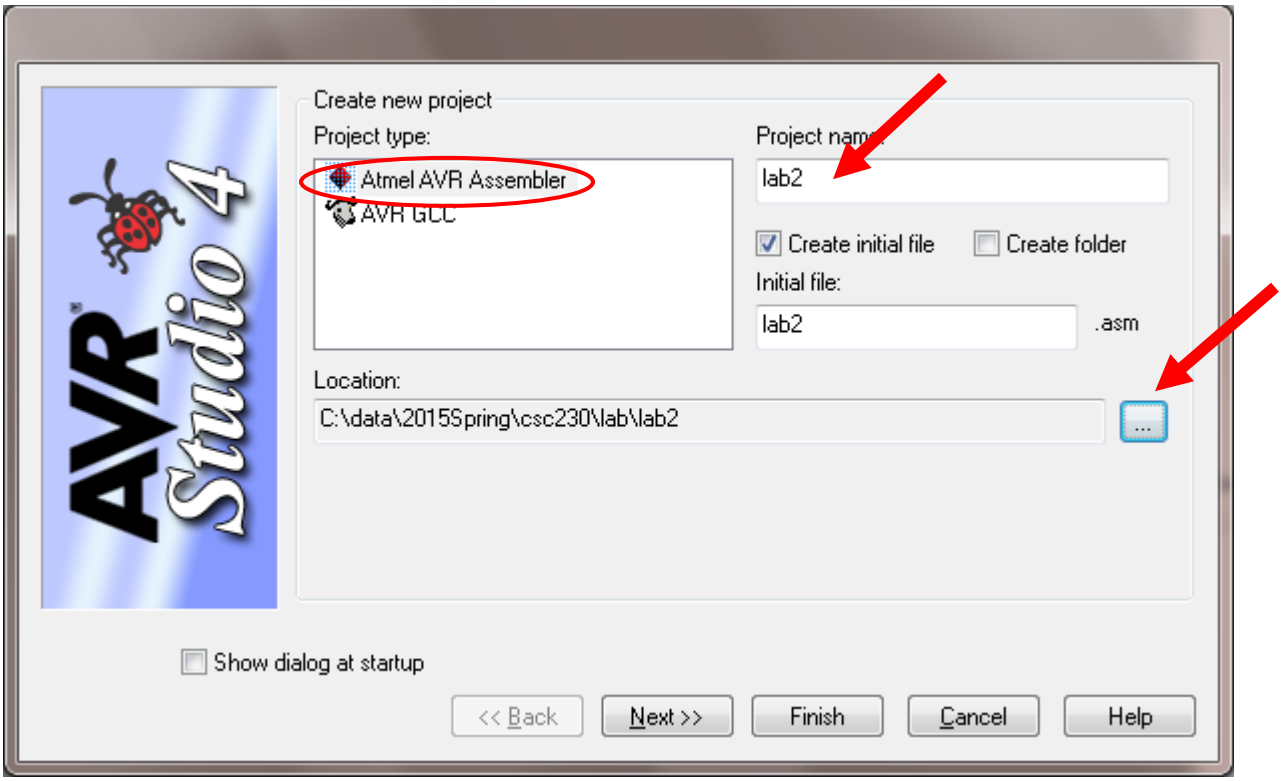
Submit checkEven.asm at the end of your lab, not your project file. This lab introduces you to the programming environment that we use in CSC 230 lab.

I. Introduction to AVR Studio 4

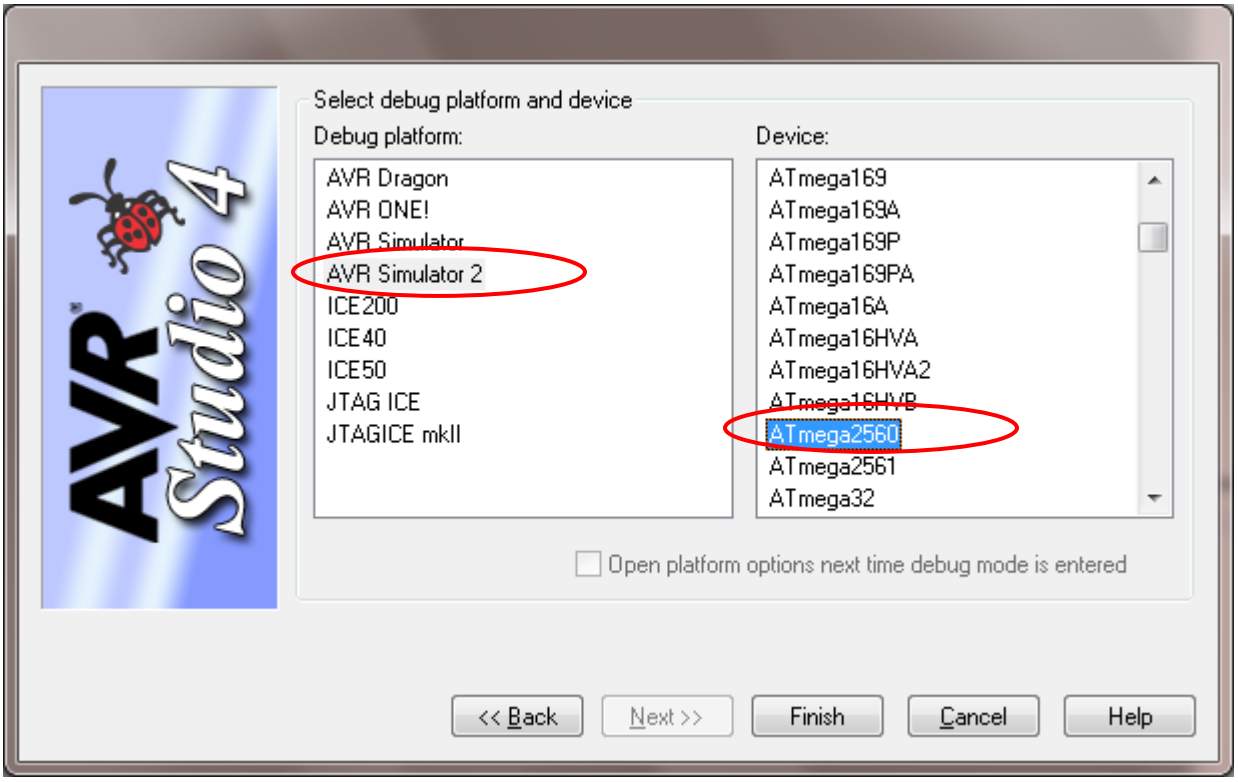
Launch AVR Studio 4 and create a new project named "lab2"



Select "Atmel AVR Assembler" for "Project Type", "lab2" for "Project name" and click on the "…" button to select a directory (suggest H drive).



Click the “Next>>” button and get to this page:
Click on “AVR Simulator 2” for “Debug Platform”
Click on “ATmega2560” for “Device”
Click on “Finish”

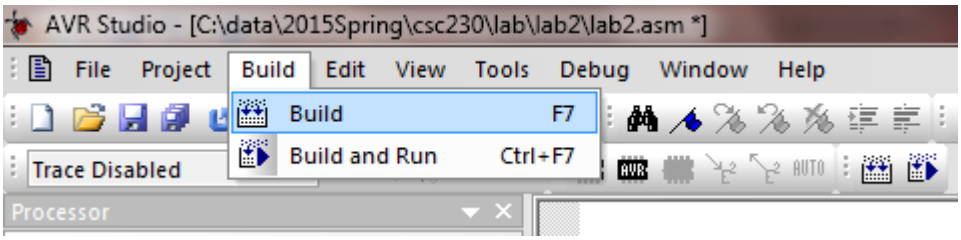


Type the following code:

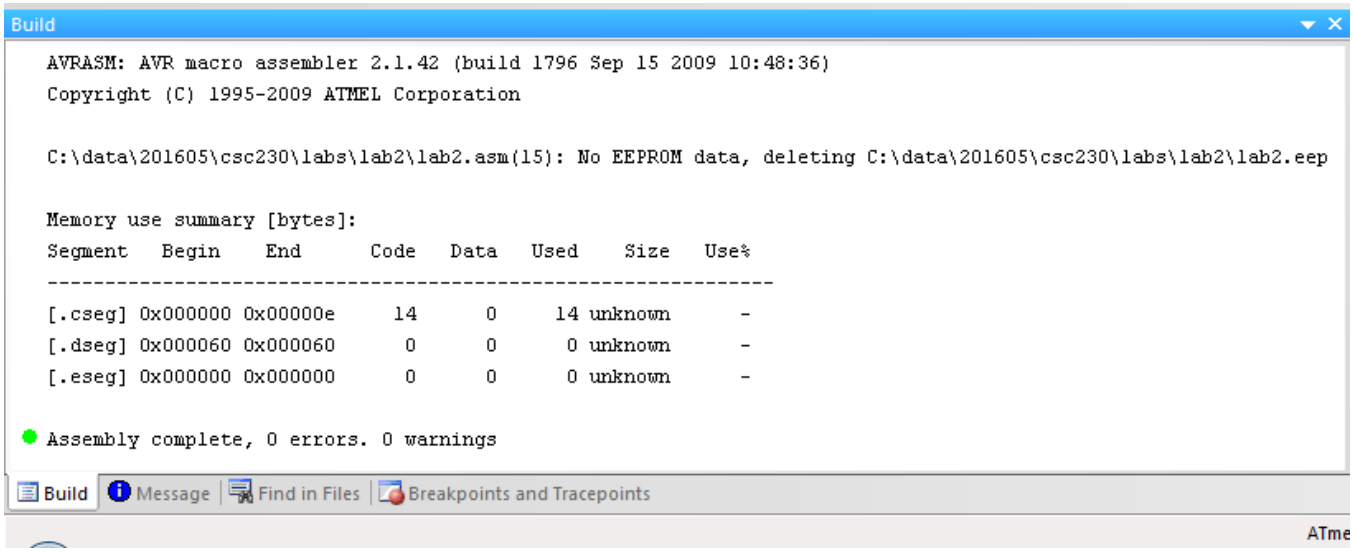
```
.cseg    ;select current segment as code
.org 0   ;begin assembling at address 0

;Define symbolic names for resources used
.def count =r16      ;Reg 16 holds counter value
    ldi count, 0x01 ;Initialize count to 1

lp:
    inc count        ;increment counter
    cpi count, 0x05
    breq done
    rjmp lp

done: jmp done
```

Save the code.

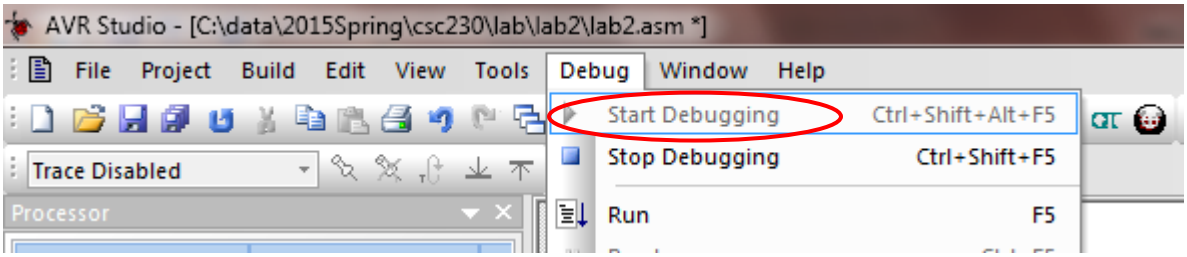
Assemble it (choose Build under the Build menu)



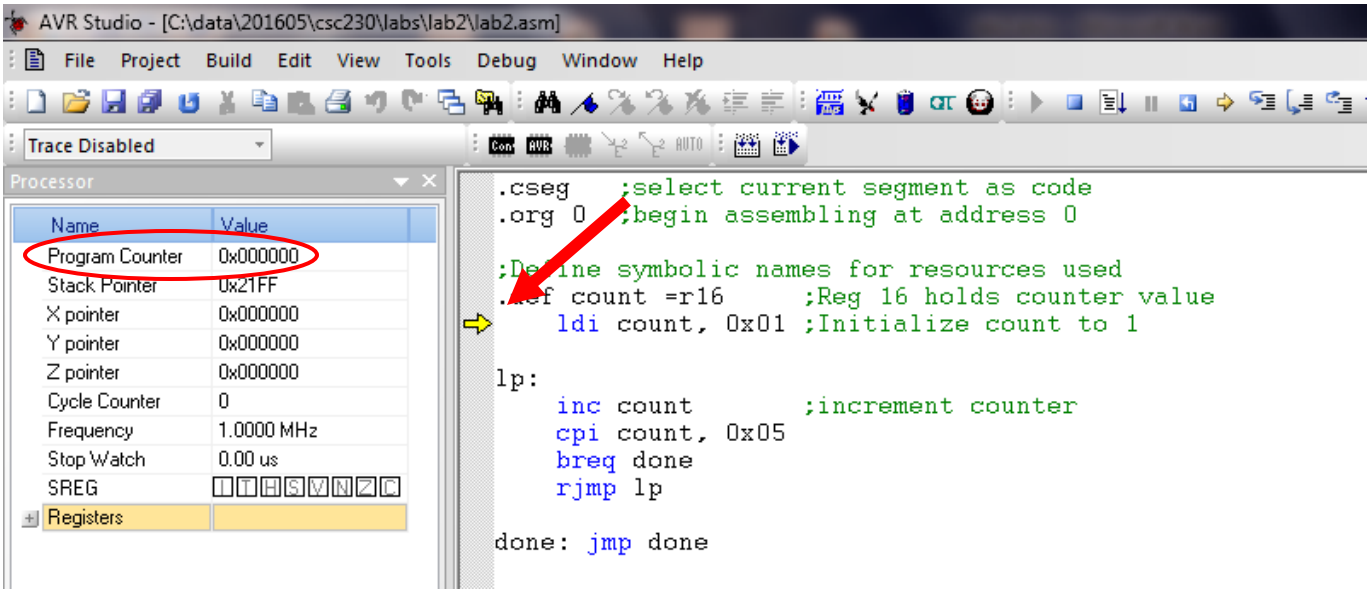
Output from the build at the bottom of the screen:



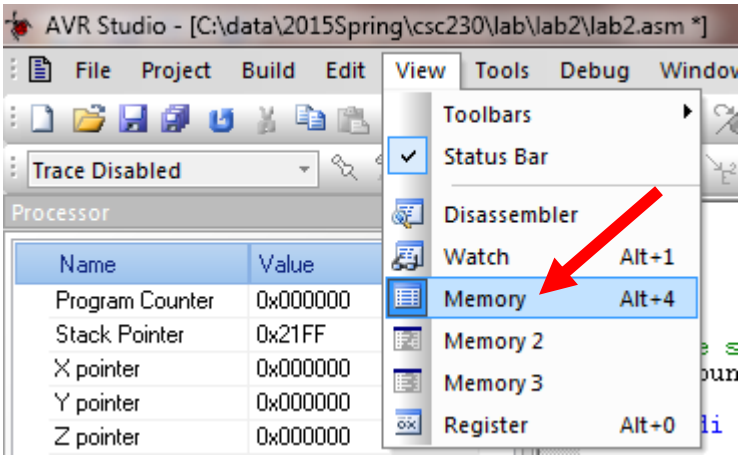
If no errors are detected, start the simulator (debugger). Select "Start Debugging command" under the "Debug" menu.



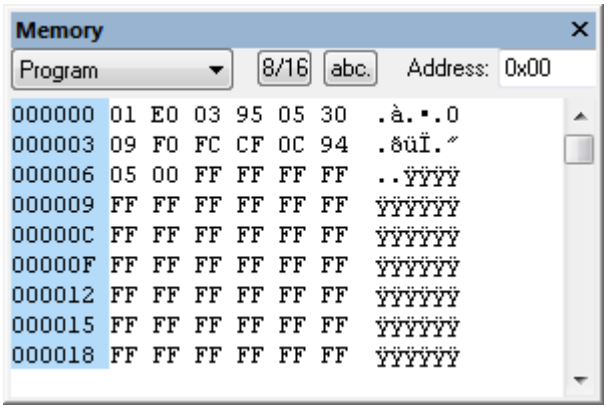
The editor should show a yellow arrow indicating the instruction about to be fetched. The left panel should show the "Processor panel", where you can examine register and processor values.



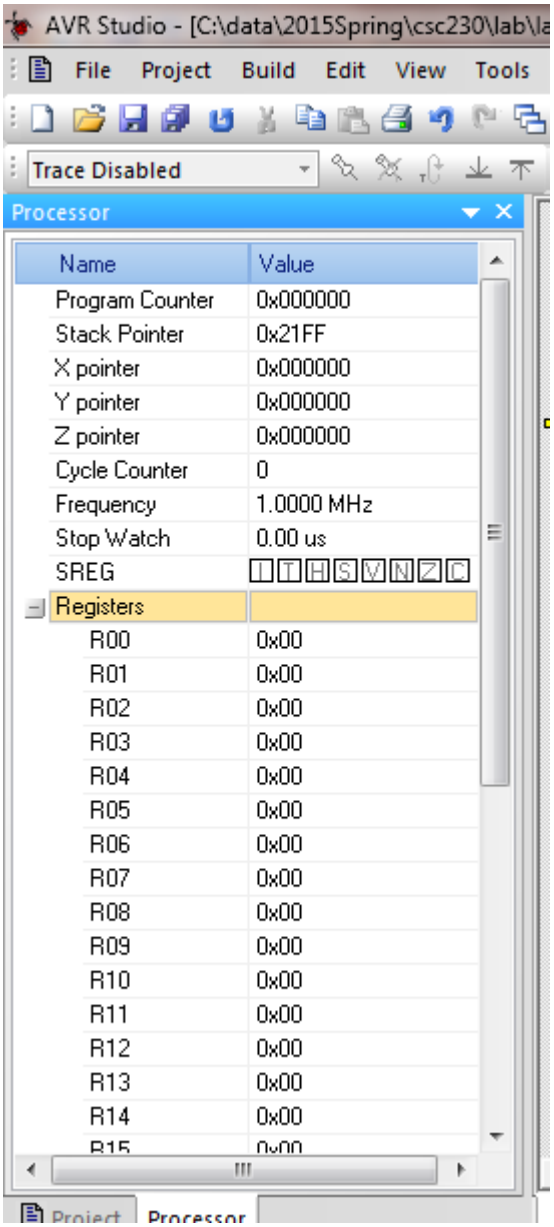
Before executing any instructions, open the "Memory" window (Under the "View" menu):



The memory looks like this: (verify the opcode)



Expand the tree node labeled “Registers”, or choose “View/Register” from the “View” menu to open a popup window.



Select the “Step Into” option from the “Debug” menu (or press F11) to allow the program to fetch and execute the first instruction. Observe the changes in PC (program counter) and register 16 (r16).

Stop the debussing session by using the 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 checkEven.asm. The program checks if an 8-bit unsigned number is an even number. If the number is even, set register R19 to 1, otherwise, set it to 0. Here is the algorithm:

R18 <= 0x09 (hint: in binary 0b 0000 1001)
Make all the other bits, except the least significant bit of R18 to 0 (hint: use ANDI and a mask. Convert the number in R18 to 0b 0000 000**?**)
Set R19 to 0x00
Compare R18 with 0x00
If (R18 == 0x00)
    R19 <= 0x01
Done

How about in higher programming language?

```
unsigned int n=9
unsigned int isEven=0
if (n%2==0)
    isEven=1
```

Submit checkEven.asm at the end of your lab using Connex.

This lab is derived from the Chapter 2 of your textbook (Some Assembly Required by Timothy S. Margush)