COMP9032 Lab 0

Getting Started

July, 2018

Before labs start from Week 2, it is highly recommended that you install AVR Studio 7 on your computer at home. We will use the software to develop and simulate AVR programs. The software is available on the course website (follow the link References → AVR Studio).

1. Objective

- Learn how to use AVR studio to debug and run an AVR assembly program.

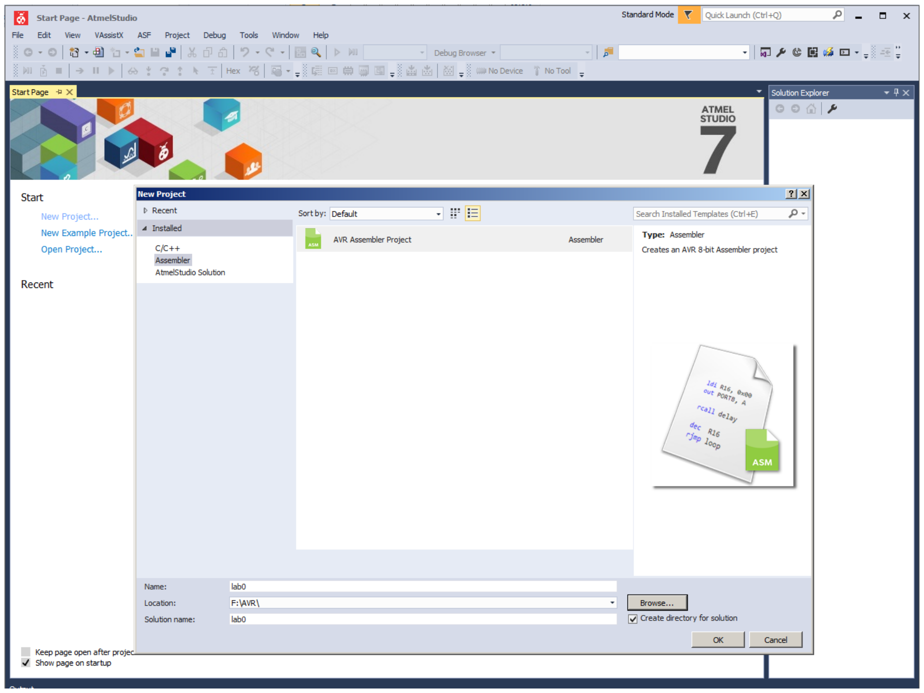
2. Introduction to AVR Studio

2.1 Start AVR Studio

To start AVR Studio, double click the AVR Studio icon, or click on Start → Programs → Atmel Studio 7.0 → Atmel Studio 7.0.

2.2 Create a New Project

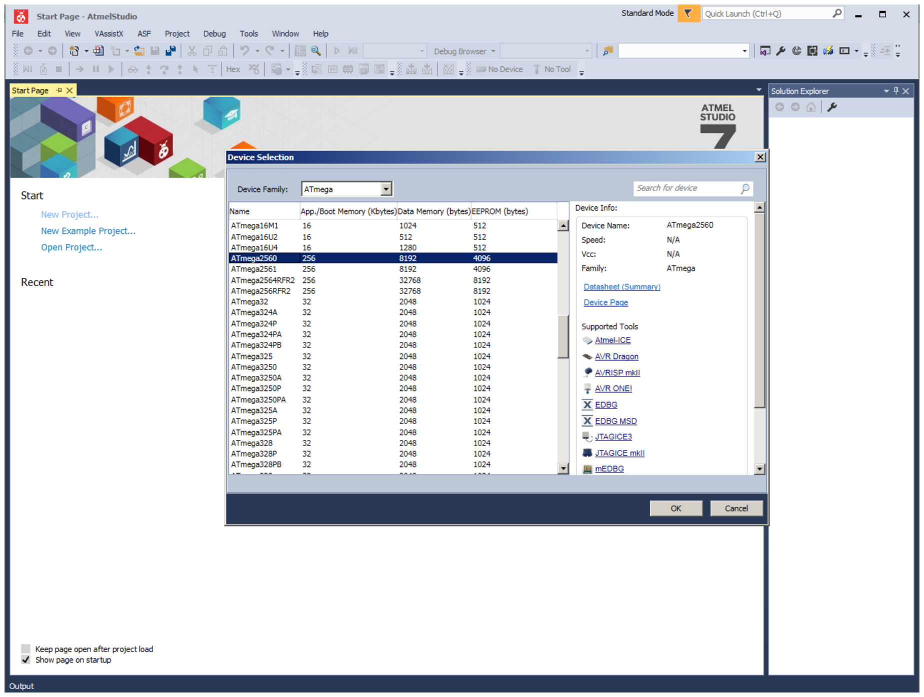
To create a new project, on the Start Page screen, click the "New Project …" link. The dialog box will appear.



Name: lab0
Location: F:\AVR\
Solution name: lab0
☑ Create directory for solution

Figure 1: New Project

Among the installed programs, select Assembler and enter a project name (e.g. "*lab0*"), as shown in Figure 1. Next, select the project location. This is the location where AVR Studio will store all files associated with the project. It is a good practice to create a separate directory for each lab. In the laboratory, you may need to create your working directory on the Desktop as you don't have write permission anywhere else. After choosing the project software type, name and location, press the “OK” button to continue. Then you will be asked to choose the device for your project. Choose “ATmega2560” from the Device list, as shown in Figure 2. Then press “OK” to continue.



Device Family: ATmega

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATmega16M1	16	1024	512
ATmega16U2	16	512	512
ATmega16U4	16	1280	512
ATmega2560	256	8192	4096
ATmega2561	256	8192	4096
ATmega2564RFR2	256	32768	8192
ATmega256RFR2	256	32768	8192
ATmega32	32	2048	1024
ATmega324A	32	2048	1024
ATmega324P	32	2048	1024
ATmega324PA	32	2048	1024
ATmega324PB	32	2048	1024
ATmega325	32	2048	1024
ATmega3250	32	2048	1024
ATmega3250A	32	2048	1024
ATmega3250P	32	2048	1024
ATmega3250PA	32	2048	1024
ATmega325A	32	2048	1024
ATmega325P	32	2048	1024
ATmega325PA	32	2048	1024
ATmega328	32	2048	1024
ATmega328P	32	2048	1024
ATmega328PB	32	2048	1024

Device Info:

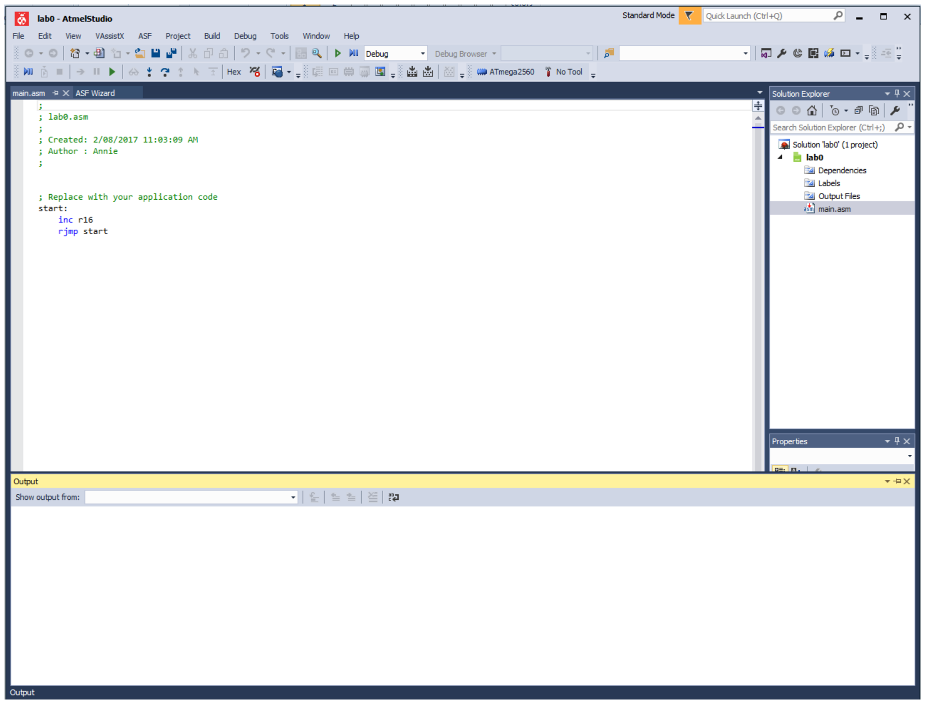
Device Name:	ATmega2560
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Datasheet (Summary)
Device Page

Supported Tools
- Atmel-ICE
- AVR Dragon
- AVRISP mkII
- AVR ONE!
- EDBG
- EDBG MSD
- JTAGICE3
- JTAGICE mkII
- mEDBG

Figure 2: Device Selection

Now the project window with a file called main.asm appears, as shown in Figure 3.



```
;
; lab0.asm
;
; Created: 2/08/2017 11:03:09 AM
; Author : Annie
;


; Replace with your application code
start:
    inc r16
    rjmp start
```

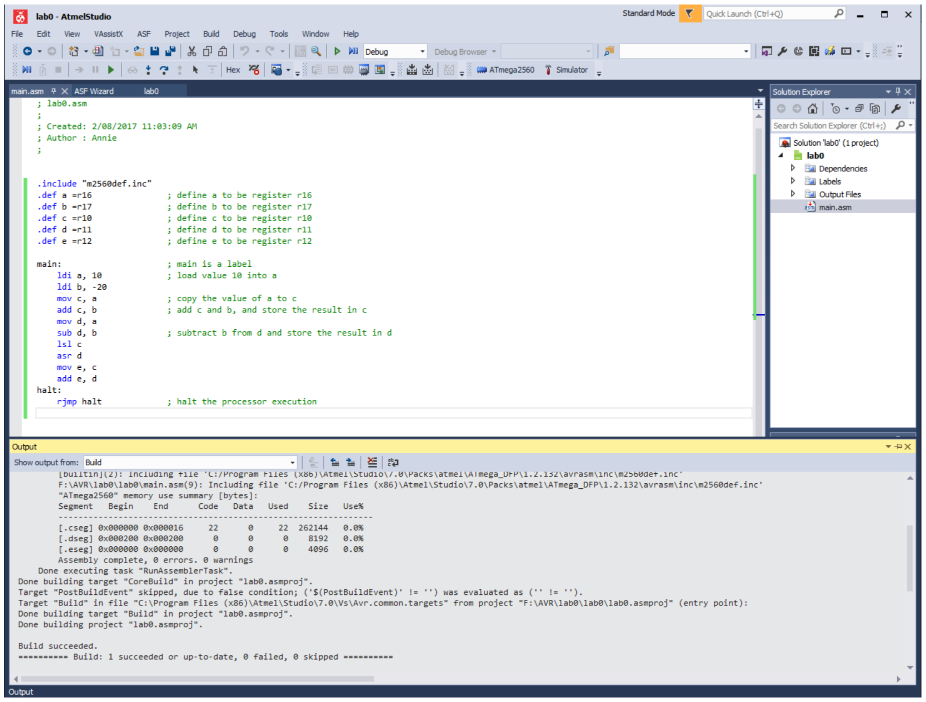
Figure 3: Project Window

Replace the sample code (that was automatically generated by the studio) with your code given in Figure 4.

```
.include "m2560def.inc"
.def a =r16                 ; define a to be register r16
.def b =r17                 ; define b to be register r17
.def c =r10                 ; define c to be register r10
.def d =r11                 ; define d to be register r11
.def e =r12                 ; define e to be register r12

main:                       ; main is a label
        ldi a, 10           ; load  value 10 into a
        ldi b, -20
        mov c, a            ; copy the value of a to c
        add c, b            ; add c and b and store the result in c
        mov d, a
        sub d, b            ; subtract b from d and store the result in d
        lsl c
        asr d
        mov e, c
        add e, d
halt:
        rjmp halt           ; halt the processor execution
```

Figure 4: Your assembly code

In the assembly code, the file m2560def.inc is included. It contains the definitions required by the assembler for the microcontroller. The next step is to build the code (i.e. assemble the assembly program into machine instructions). This is done by selecting “Build Solution” from the “Build” menu (or press F7). The Output window then displays the information from the assembler. From this window we can see that the assembly process was completed with no errors and the executable file is 22 bytes long.

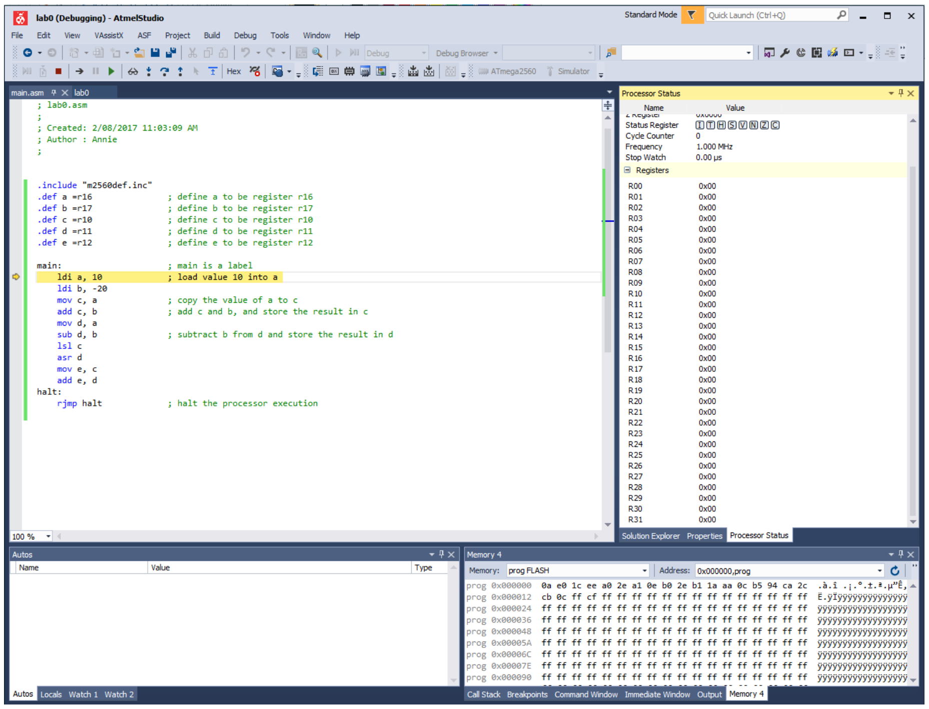


```
; lab0.asm
;
; Created: 2/08/2017 11:03:09 AM
; Author : Annie
;


.include "m2560def.inc"
.def a =r16             ; define a to be register r16
.def b =r17             ; define b to be register r17
.def c =r10             ; define c to be register r10
.def d =r11             ; define d to be register r11
.def e =r12             ; define e to be register r12

main:                   ; main is a label
    ldi a, 10           ; load value 10 into a
    ldi b, -20
    mov c, a            ; copy the value of a to c
    add c, b            ; add c and b, and store the result in c
    mov d, a
    sub d, b            ; subtract b from d and store the result in d
    lsl c
    asr d
    mov e, c
    add e, d
halt:
    rjmp halt           ; halt the processor execution
```

```
        [builtin](2): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.132\avrasm\inc\m2560def.inc'
        F:\AVR\lab0\lab0\main.asm(9): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.132\avrasm\inc\m2560def.inc'
        "ATmega2560" memory use summary [bytes]:
        Segment   Begin    End      Code   Data   Used    Size   Use%
        ---------------------------------------------------------------
        [.cseg] 0x000000 0x000016     22      0     22  262144   0.0%
        [.dseg] 0x000200 0x000200      0      0      0    8192   0.0%
        [.eseg] 0x000000 0x000000      0      0      0    4096   0.0%
        Assembly complete, 0 errors. 0 warnings
    Done executing task "RunAssemblerTask".
Done building target "CoreBuild" in project "lab0.asmproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "F:\AVR\lab0\lab0\lab0.asmproj" (entry point):
Done building target "Build" in project "lab0.asmproj".
Done building project "lab0.asmproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Figure 5: Build Your Program

Now we can investigate the execution of the code in the debug mode by selecting “Start Debugging and Break” under the Debug manu. Notice that a yellow arrow is pointing to the first instruction “ldi a, 10” in the assembly code window, as shown in Figure 6. This arrow indicates the position of the next instruction to be executed.   Before simulation, we may want to set up the Register View so that we can see the value of each register during the program execution. Select Debug→Windows→Processor Status and drag the created window to the side (see Figure 6).



```
; lab0.asm
;
; Created: 2/08/2017 11:03:09 AM
; Author : Annie
;


.include "m2560def.inc"
.def a =r16               ; define a to be register r16
.def b =r17               ; define b to be register r17
.def c =r10               ; define c to be register r10
.def d =r11               ; define d to be register r11
.def e =r12               ; define e to be register r12

main:                     ; main is a label
    ldi a, 10             ; load value 10 into a
    ldi b, -20
    mov c, a              ; copy the value of a to c
    add c, b              ; add c and b, and store the result in c
    mov d, a
    sub d, b              ; subtract b from d and store the result in d
    lsl c
    asr d
    mov e, c
    add e, d
halt:
    rjmp halt             ; halt the processor execution
```

Processor Status

Name	Value
Status Register	I T H S V N Z C
Cycle Counter	0
Frequency	1.000 MHz
Stop Watch	0.00 µs
Registers	
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	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x00
R31	0x00

Memory 4 — Memory: prog FLASH Address: 0x000000,prog

```
prog 0x000000  0a e0 1c ee a0 2e a1 0e b0 2e b1 1a aa 0c b5 94 ca 2c  .à.î .¡.°.±.ª.µ”Ê,
prog 0x000012  cb 0c ff cf ff ff ff ff ff ff ff ff ff ff ff ff ff ff  Ë.ÿÏÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000024  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00005A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00007E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000090  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

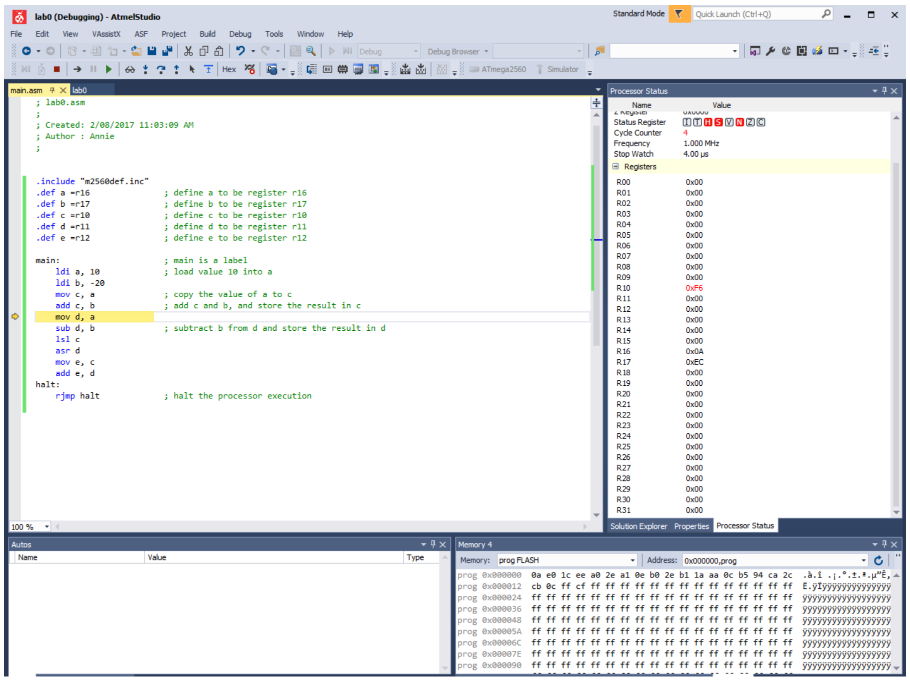
lab0 (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Standard Mode
Quick Launch (Ctrl+Q)
Debug
Debug Browser
Hex
ATmega2560
Simulator
main.asm
lab0
100 %
Autos
Name
Value
Type
Autos Locals Watch 1 Watch 2
Solution Explorer Properties Processor Status
Call Stack Breakpoints Command Window Immediate Window Output Memory 4

Figure 6: Start Debugging

The value of each register is initially 0. These registers will be dynamically updated during the program execution. You can also assign and change values of these registers by clicking on the value of a register and entering the appropriate value at any time during the simulation. You can also change the value display format by right clicking on the value and selecting the format you want to use.

2.3 Run Simulation

There are commands to single step through the code: “Step Over” F10 and “Step Into” F11. The difference between these commands is that F10 does not trace into subroutines. Since our example does not contain any subroutines, there is no difference between the two here. Now, single-step down to the fifth line of the code by repeatedly pressing the F11 key. Notice how the color changes from black to red on the registers when their values are updated. This makes it easy to identify which register changes its value when an instruction is executed.



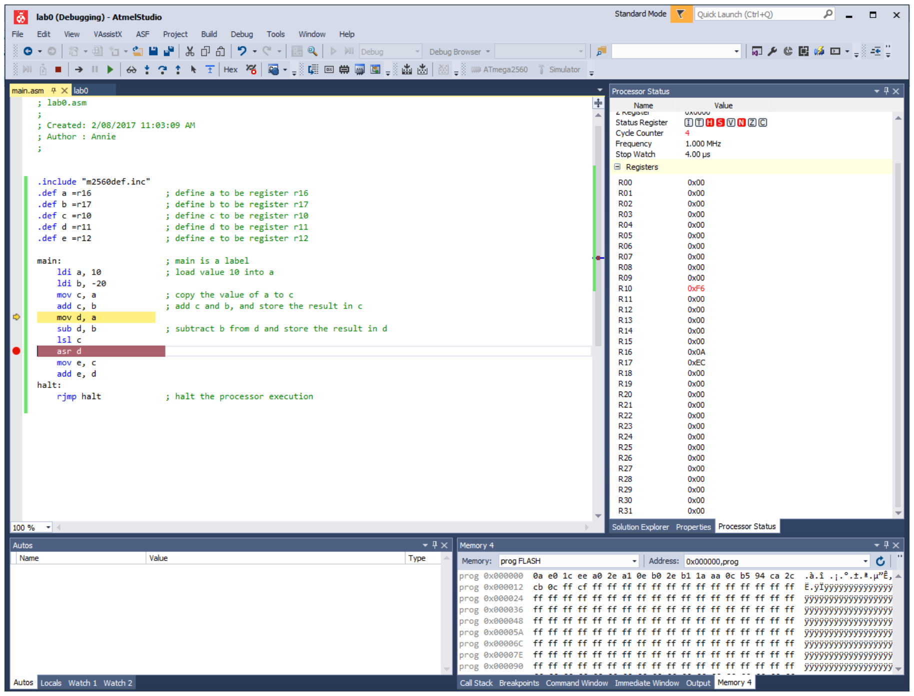
```
; lab0.asm
;
; Created: 2/08/2017 11:03:09 AM
; Author : Annie
;


.include "m2560def.inc"
.def a =r16                 ; define a to be register r16
.def b =r17                 ; define b to be register r17
.def c =r10                 ; define c to be register r10
.def d =r11                 ; define d to be register r11
.def e =r12                 ; define e to be register r12

main:                       ; main is a label
    ldi a, 10               ; load value 10 into a
    ldi b, -20
    mov c, a                ; copy the value of a to c
    add c, b                ; add c and b, and store the result in c
    mov d, a
    sub d, b                ; subtract b from d and store the result in d
    lsl c
    asr d
    mov e, c
    add e, d
halt:
    rjmp halt               ; halt the processor execution
```

```
Processor Status
Name              Value
Status Register   I T H S V N Z C
Cycle Counter     4
Frequency         1.000 MHz
Stop Watch        4.00 µs
Registers
R00   0x00
R01   0x00
R02   0x00
R03   0x00
R04   0x00
R05   0x00
R06   0x00
R07   0x00
R08   0x00
R09   0x00
R10   0xF6
R11   0x00
R12   0x00
R13   0x00
R14   0x00
R15   0x00
R16   0x0A
R17   0xEC
R18   0x00
R19   0x00
R20   0x00
R21   0x00
R22   0x00
R23   0x00
R24   0x00
R25   0x00
R26   0x00
R27   0x00
R28   0x00
R29   0x00
R30   0x00
R31   0x00
```

```
Memory 4
Memory: prog FLASH      Address: 0x000000,prog
prog 0x000000  0a e0 1c ee a0 2e a1 0e b0 2e b1 1a aa 0c b5 94 ca 2c  .à.î .¡.°.±.ª.µ”Ê,
prog 0x000012  cb 0c ff cf ff ff ff ff ff ff ff ff ff ff ff ff ff ff  Ë.ÿÏÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000024  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00005A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00007E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000090  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Figure 7: Single-Step Execution

You can also run the code with breakpoints. To add a breakpoint, move the cursor to the instruction you want to stop on, and right click mouse and select Breakpoint→Insert Breakpoint. Figure 8 shows a breakpoint (indicated by the red dot) is added on the *asr* instruction.



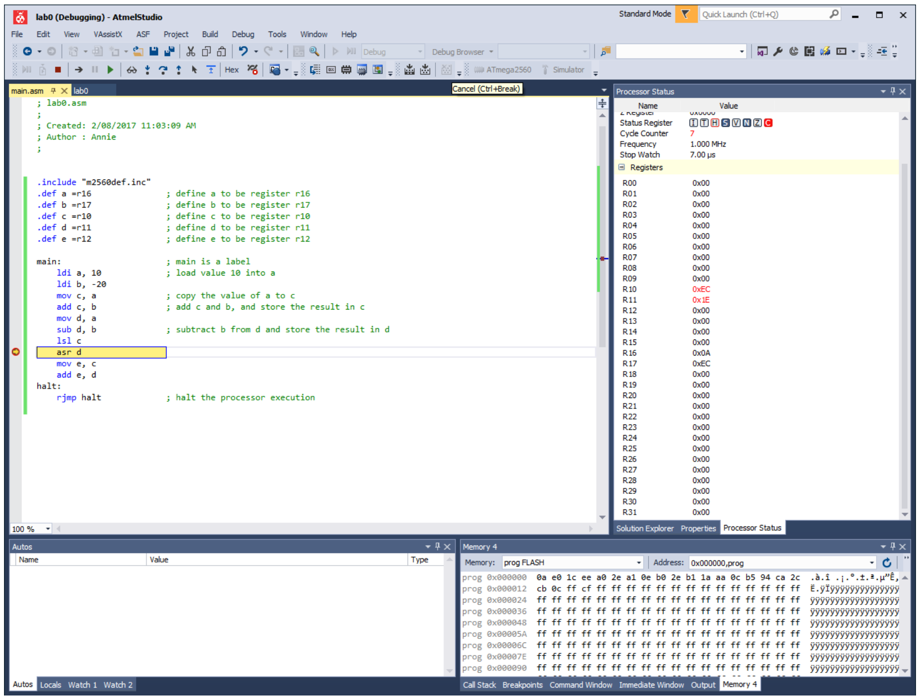
```
; lab0.asm
;
; Created: 2/08/2017 11:03:09 AM
; Author : Annie
;


.include "m2560def.inc"
.def a =r16                 ; define a to be register r16
.def b =r17                 ; define b to be register r17
.def c =r10                 ; define c to be register r10
.def d =r11                 ; define d to be register r11
.def e =r12                 ; define e to be register r12

main:                       ; main is a label
    ldi a, 10               ; load value 10 into a
    ldi b, -20
    mov c, a                ; copy the value of a to c
    add c, b                ; add c and b, and store the result in c
    mov d, a
    sub d, b                ; subtract b from d and store the result in d
    lsl c
    asr d
    mov e, c
    add e, d
halt:
    rjmp halt               ; halt the processor execution
```

```
Processor Status
Name              Value
Status Register   I T H S V N Z C
Cycle Counter     4
Frequency         1.000 MHz
Stop Watch        4.00 µs
Registers
R00   0x00
R01   0x00
R02   0x00
R03   0x00
R04   0x00
R05   0x00
R06   0x00
R07   0x00
R08   0x00
R09   0x00
R10   0xF6
R11   0x00
R12   0x00
R13   0x00
R14   0x00
R15   0x00
R16   0x0A
R17   0xEC
R18   0x00
R19   0x00
R20   0x00
R21   0x00
R22   0x00
R23   0x00
R24   0x00
R25   0x00
R26   0x00
R27   0x00
R28   0x00
R29   0x00
R30   0x00
R31   0x00
```

```
Memory 4
Memory: prog FLASH      Address: 0x000000,prog
prog 0x000000  0a e0 1c ee a0 2e a1 0e b0 2e b1 1a aa 0c b5 94 ca 2c  .à.î .¡.°.±.ª.µ”Ê,
prog 0x000012  cb 0c ff cf ff ff ff ff ff ff ff ff ff ff ff ff ff ff  Ë.ÿÏÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000024  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00005A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00007E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000090  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Figure 8: Setting Breakpoint

Now press the Continue (F5) button. The simulation will continue and then stop at the breakpoint, as shown in Figure 9. The breakpoint can be removed/disabled by selecting Breakpoint → Deleting Breakpoint/Disable Breakpoint.



```
; lab0.asm
;
; Created: 2/08/2017 11:03:09 AM
; Author : Annie
;


.include "m2560def.inc"
.def a =r16                 ; define a to be register r16
.def b =r17                 ; define b to be register r17
.def c =r10                 ; define c to be register r10
.def d =r11                 ; define d to be register r11
.def e =r12                 ; define e to be register r12

main:                       ; main is a label
    ldi a, 10               ; load value 10 into a
    ldi b, -20
    mov c, a                ; copy the value of a to c
    add c, b                ; add c and b, and store the result in c
    mov d, a
    sub d, b                ; subtract b from d and store the result in d
    lsl c
    asr d
    mov e, c
    add e, d
halt:
    rjmp halt               ; halt the processor execution
```

Name	Value
Status Register	I T H S V N Z C
Cycle Counter	7
Frequency	1.000 MHz
Stop Watch	7.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0xEC
R11	0x1E
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x0A
R17	0xEC
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x00
R31	0x00

Memory: prog FLASH Address: 0x000000,prog

```
prog 0x000000  0a e0 1c ee a0 2e a1 0e b0 2e b1 1a aa 0c b5 94 ca 2c  .à.î .¡.°.±.ª.µ”Ê,
prog 0x000012  cb 0c ff cf ff ff ff ff ff ff ff ff ff ff ff ff ff ff  Ë.ÿÏÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000024  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00005A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00007E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000090  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

lab0 (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Standard Mode
Quick Launch (Ctrl+Q)
main.asm
lab0
Cancel (Ctrl+Break)
Processor Status
Autos
Name Value Type
Memory 4
Solution Explorer Properties Processor Status
Autos Locals Watch 1 Watch 2
Call Stack Breakpoints Command Window Immediate Window Output Memory 4

Figure 9: Execution with Breakpoint

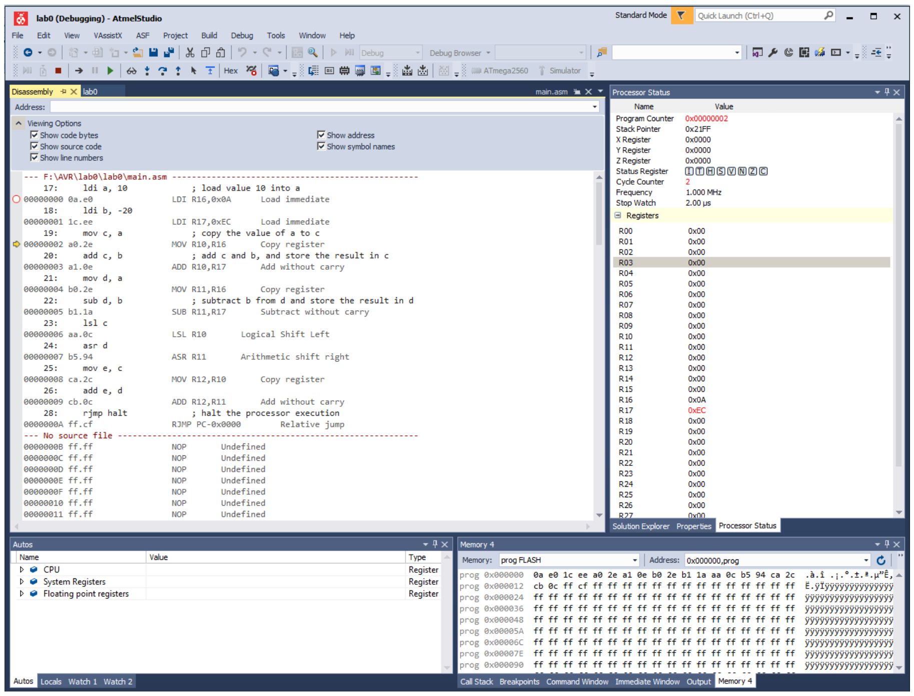
To end the debugging session, press “Ctrl+Shift+F5” or selecting Debug → Stop Debugging from the menu.

2.4 Status Register

Each AVR microcontroller has a Status REGister (named as SREG) that keeps 8 flags such as C, S and V. The definitions of all 8 flags in SREG can be found in Mega2560 Data Sheet (page 14), which is available on the course website (follow the link References → Documents → Mega2560 Data Sheet). These flags are dynamically updated during the program execution, and can also be observed in the Processor Status window. When a flag bit is set, the corresponding bit block is highlighted in red. For example, after the instruction “add c, b” is finished, bit 2 (N), bit 4 (S), and bit 5 (H) are set, as shown in Figure 8.

2.5 Disassembly

AVR Studio provides a disassembler which lists the assembly details. Using the disassembler, you can see the corresponding machine code and the memory address of each instruction. To run the disassembler, start debugging, and select Debug→Windows→Disassembly. You will see a listing of your disassembled program, as shown in Figure 10. For each instruction its memory address (in the first column), machine code (the second column, in the hex format), assembly code (the third column), and its operation (the last column) are given.


lab0 (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Standard Mode
Quick Launch (Ctrl+Q)
Hex
ATmega2560
Simulator
Disassembly
lab0
main.asm
Viewing Options
Solution Explorer Properties Processor Status
Autos Locals Watch 1 Watch 2
Call Stack Breakpoints Command Window Immediate Window Output Memory 4

Address:

Viewing Options
- [✓] Show code bytes
- [✓] Show source code
- [✓] Show line numbers
- [✓] Show address
- [✓] Show symbol names

```
--- F:\AVR\lab0\lab0\main.asm ---------------------------------------------------
    17:     ldi a, 10               ; load value 10 into a
00000000 0a.e0              LDI R16,0x0A      Load immediate
    18:     ldi b, -20
00000001 1c.ee              LDI R17,0xEC      Load immediate
    19:     mov c, a                ; copy the value of a to c
00000002 a0.2e              MOV R10,R16       Copy register
    20:     add c, b                ; add c and b, and store the result in c
00000003 a1.0e              ADD R10,R17       Add without carry
    21:     mov d, a
00000004 b0.2e              MOV R11,R16       Copy register
    22:     sub d, b                ; subtract b from d and store the result in d
00000005 b1.1a              SUB R11,R17       Subtract without carry
    23:     lsl c
00000006 aa.0c              LSL R10       Logical Shift Left
    24:     asr d
00000007 b5.94              ASR R11       Arithmetic shift right
    25:     mov e, c
00000008 ca.2c              MOV R12,R10       Copy register
    26:     add e, d
00000009 cb.0c              ADD R12,R11       Add without carry
    28:     rjmp halt               ; halt the processor execution
0000000A ff.cf              RJMP PC-0x0000        Relative jump
--- No source file --------------------------------------------------------------
0000000B ff.ff              NOP       Undefined
0000000C ff.ff              NOP       Undefined
0000000D ff.ff              NOP       Undefined
0000000E ff.ff              NOP       Undefined
0000000F ff.ff              NOP       Undefined
00000010 ff.ff              NOP       Undefined
00000011 ff.ff              NOP       Undefined
```

Processor Status

Name	Value
Program Counter	0x00000002
Stack Pointer	0x21FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	2
Frequency	1.000 MHz
Stop Watch	2.00 µs
Registers	
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	0x0A
R17	0xEC
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00

Autos

Name	Value	Type
CPU		Register
System Registers		Register
Floating point registers		Register

Memory 4

Memory: prog FLASH   Address: 0x000000,prog

```
prog 0x000000  0a e0 1c ee a0 2e a1 0e b0 2e b1 1a aa 0c b5 94 ca 2c  .à.î .¡.°.±.ª.µ”Ê,
prog 0x000012  cb 0c ff cf ff ff ff ff ff ff ff ff ff ff ff ff ff ff  Ë.ÿÏÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000024  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00005A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00007E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000090  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Figure 10: Disassemble Code

Note that you don’t need to attend laboratory for this lab. However, you are strongly recommended to complete it at home. If you have any questions, feel free to ask the lab tutor in week 2.