EE2016 Experiment 2 Report

Student Name: Vaddiraju Anshul, Keshaw Choudhary
Roll Number: EE21B151, EE21B069
Batch Number: B33

1 Aim of the Experiment

To implement basic arithmetic and logical manipulation programs using Atmel Atmega8 microcontroller in assembly program emulation, including addition, multiplication and comparison.

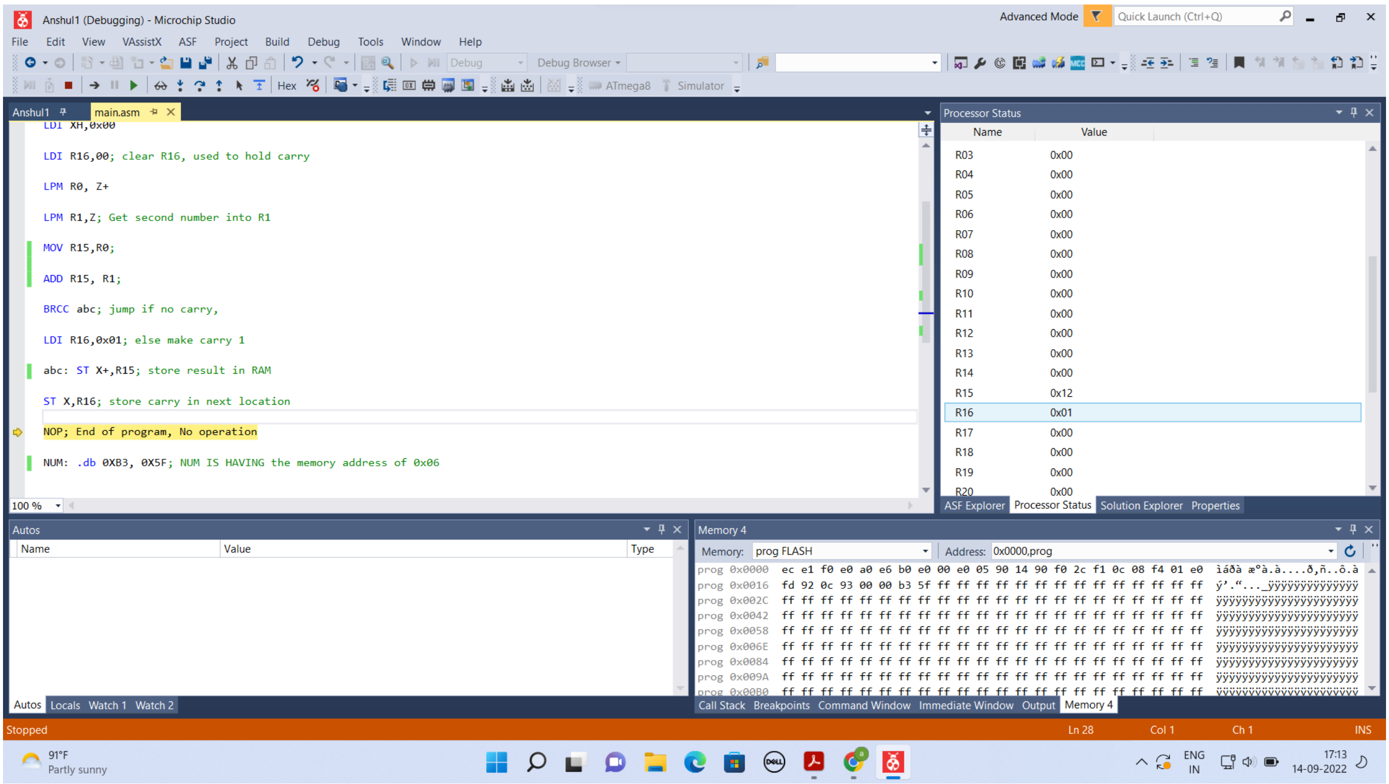
2 8-Bit Addition

We can add two 8-bit numbers in AVR by using the arithmetic instruction ADD and put the carry in with the branch instruction BRCC.

2.1 Code

```
    .CSEG

LDI ZL, LOW(NUM<<1)
LDI ZH, HIGH(NUM<<1)

LDI XL,0x60; load SRAM address in X-register
LDI XH,0x00

LDI R16,00; clear R16, used to hold carry
LPM R0, Z+
LPM R1,Z; Get second number into R1

MOV R15,R0;
ADD R15, R1;

BRCC abc; jump if no carry,
LDI R16,0x01; else make carry 1

abc: ST X+,R15; store result in RAM
ST X,R16; store carry in next location
NOP; End of program, No operation

NUM: .db 0XB3, 0X5F; NUM IS HAVING the memory address of 0x06
```

2.2 Output



Anshul1 (Debugging) - Microchip Studio

main.asm

```
LDI XH,0x00

LDI R16,00; clear R16, used to hold carry

LPM R0, Z+

LPM R1,Z; Get second number into R1

MOV R15,R0;

ADD R15, R1;

BRCC abc; jump if no carry,

LDI R16,0x01; else make carry 1

abc: ST X+,R15; store result in RAM

ST X,R16; store carry in next location

NOP; End of program, No operation

NUM: .db 0XB3, 0X5F; NUM IS HAVING the memory address of 0x06
```

Processor Status

Name	Value
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	0x12
R16	0x01
R17	0x00
R18	0x00
R19	0x00
R20	0x00

Memory 4

Memory: prog FLASH Address: 0x0000,prog

```
prog 0x0000  ec e1 f0 e0 a0 e6 b0 e0 00 e0 05 90 14 90 f0 2c f1 0c 08 f4 01 e0  ìáðà æ°à.à....ð,ñ..ô.à
prog 0x0016  fd 92 0c 93 00 00 b3 5f ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ý’.“..._ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x002C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0042  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0058  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x006E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0084  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x009A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00B0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Stopped Ln 28 Col 1 Ch 1 INS

2.3 Registers

We have used R16 to store the carry of addition and R15 to store the remaining eight bits. So,

$$\begin{gathered} \text{B3} + \text{5F} = 112 \\ \text{R16} \leftarrow 01 \\ \text{R15} \leftarrow 12 \end{gathered}$$

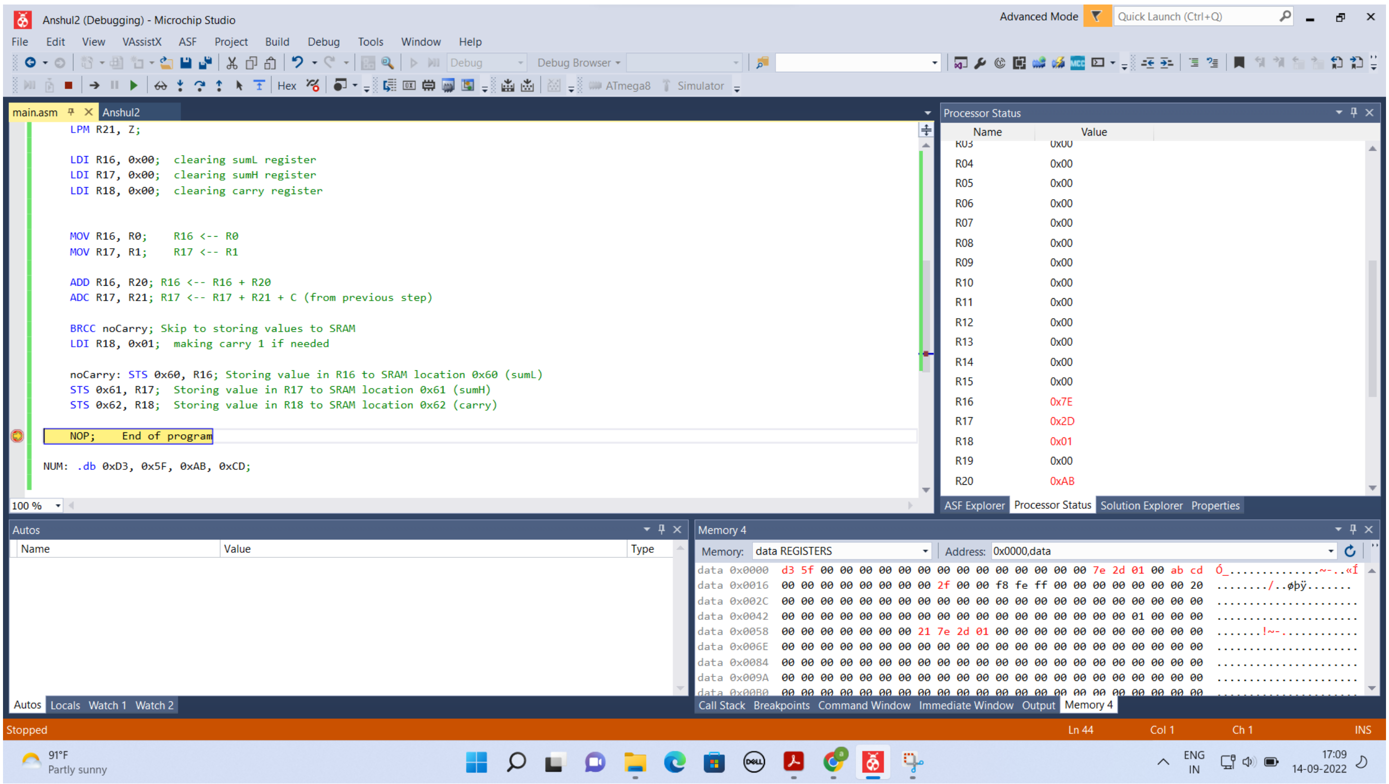
3 16-Bit Addition using an 8-Bit Processor

The AVR Architecture cannot directly add two 16-bit numbers stored in registers with a single instruction, but it can do it in two. We start by adding the lower bytes of our 16-bit numbers using the ADD instruction.Following this, we use the instruction ADC - add with carry - to add the upper bytes of our 16-bit numbers. ADC knows whether the previous add instruction resulted in an overflow. If it did, ADC will carry an extra bit into the sum to account for this.

3.1 Code

```
    .CSEG; Start program

LDI ZL, LOW(NUM<<1);
LDI ZH, HIGH(NUM<<1);
LPM R0, Z+;
LPM R1, Z+;
LPM R20, Z+;
LPM R21, Z;

LDI R16, 0x00; clearing sumL register
LDI R17, 0x00; clearing sumH register
LDI R18, 0x00; clearing carry register


MOV R16, R0; R16 <-- R0
MOV R17, R1; R17 <-- R1

ADD R16, R20; R16 <-- R16 + R20
ADC R17, R21; R17 <-- R17 + R21 + C (from previous step)

BRCC noCarry; Skip to storing values to SRAM
LDI R18, 0x01; making carry 1 if needed

noCarry: STS 0x60, R16; Storing value in R16 to SRAM location 0x60 (sumL)
STS 0x61, R17; Storing value in R17 to SRAM location 0x61 (sumH)
STS 0x62, R18; Storing value in R18 to SRAM location 0x62 (carry)

NOP; End of program

NUM: .db 0xD3, 0x5F, 0xAB, 0xCD;
```

3.2 Output



Anshul2 (Debugging) - Microchip Studio

main.asm

```
    LPM R21, Z;

    LDI R16, 0x00;  clearing sumL register
    LDI R17, 0x00;  clearing sumH register
    LDI R18, 0x00;  clearing carry register


    MOV R16, R0;    R16 <-- R0
    MOV R17, R1;    R17 <-- R1

    ADD R16, R20; R16 <-- R16 + R20
    ADC R17, R21; R17 <-- R17 + R21 + C (from previous step)

    BRCC noCarry; Skip to storing values to SRAM
    LDI R18, 0x01;  making carry 1 if needed

    noCarry: STS 0x60, R16; Storing value in R16 to SRAM location 0x60 (sumL)
    STS 0x61, R17;  Storing value in R17 to SRAM location 0x61 (sumH)
    STS 0x62, R18;  Storing value in R18 to SRAM location 0x62 (carry)

    NOP;    End of program

NUM: .db 0xD3, 0x5F, 0xAB, 0xCD;
```

Processor Status

Name	Value
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	0x7E
R17	0x2D
R18	0x01
R19	0x00
R20	0xAB

Memory 4

Memory: data REGISTERS   Address: 0x0000,data

```
data 0x0000  d3 5f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7e 2d 01 00 ab cd  Ó_..............~-..«Í
data 0x0016  00 00 00 00 00 00 00 00 2f 00 00 f8 fe ff 00 00 00 00 00 00 00 20  ......../..øþÿ.......
data 0x002C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x0042  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00  ......................
data 0x0058  00 00 00 00 00 00 00 21 7e 2d 01 00 00 00 00 00 00 00 00 00 00 00  .......!~-............
data 0x006E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x0084  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x009A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x00B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
```

Stopped   Ln 44   Col 1   Ch 1   INS

3.3 Registers

We require 3 registers to store sum of two 16-bit numbers i.e a 17-bit number. We use R18 to store the MSB, R17 and R16 to store the other 16 bits of the number So,

$$
\begin{gathered}
5\text{FD}3 + \text{CDAB} = 12\text{D}7\text{E} \\
\text{R}18 \leftarrow 01 \\
\text{R}17 \leftarrow 2D \\
\text{R}16 \leftarrow 7E
\end{gathered}
$$

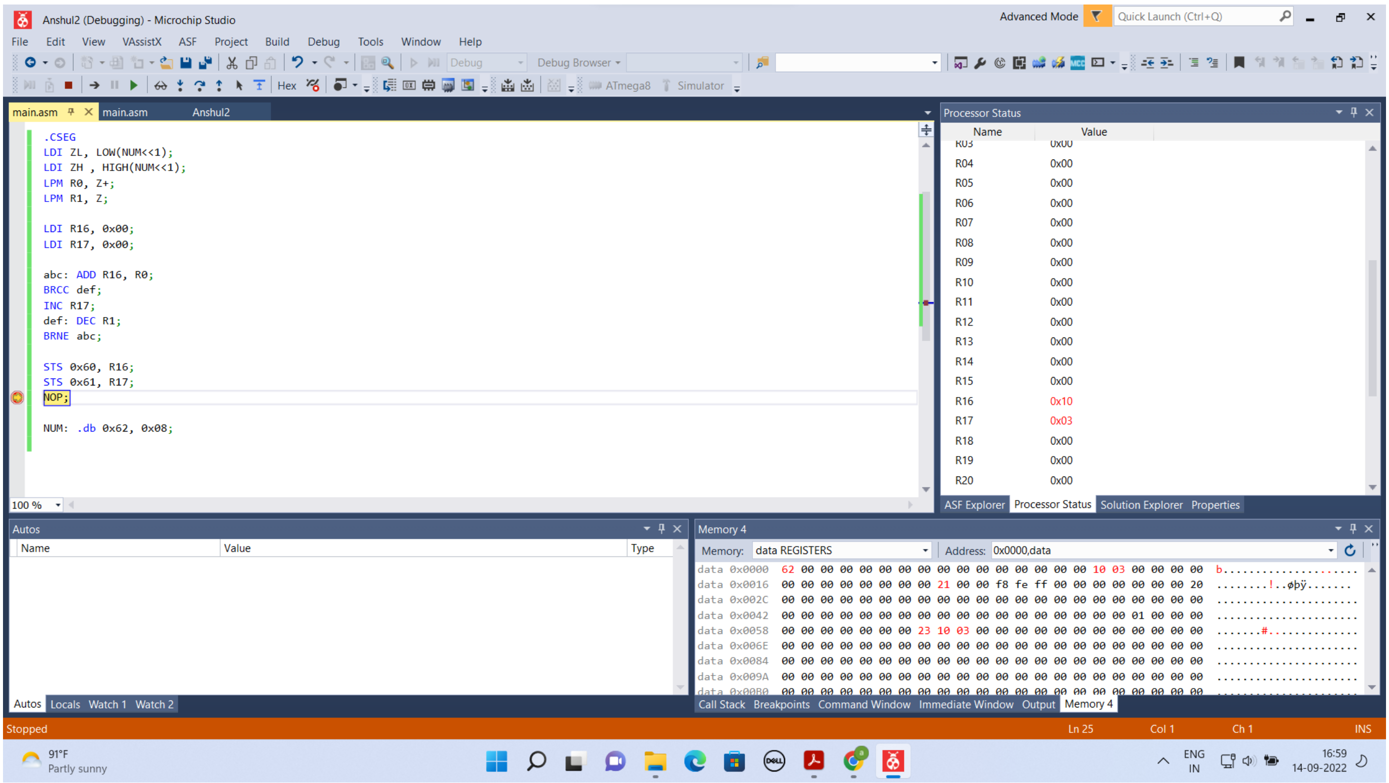
4 Multiplication of Two 8-Bit Numbers

We can think of multiplication of two numbers as the sum of the first number repeated second number times

4.1 Code

```
    LDI ZL, LOW(NUM<<1);
LDI ZH, HIGH (NUM<<1);

LPM R0, Z+;
LPM R1, Z;

LDI R16, 0x00;
LDI R17, 0x00;

abc: ADD R16, R0;
BRCC def;
INC R17;
def: DEC R1;
BRNE abc;

STS 0x60, R16;
STS 0x61, R17;
NOP;
NUM: .db 0x62, 0x08;
```

4.2 Output



Anshul2 (Debugging) - Microchip Studio

```
.CSEG
LDI ZL, LOW(NUM<<1);
LDI ZH , HIGH(NUM<<1);
LPM R0, Z+;
LPM R1, Z;

LDI R16, 0x00;
LDI R17, 0x00;

abc: ADD R16, R0;
BRCC def;
INC R17;
def: DEC R1;
BRNE abc;

STS 0x60, R16;
STS 0x61, R17;
NOP;

NUM: .db 0x62, 0x08;
```

Processor Status

Name	Value
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	0x10
R17	0x03
R18	0x00
R19	0x00
R20	0x00

Memory 4

Memory: data REGISTERS   Address: 0x0000,data

```
data 0x0000  62 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 03 00 00 00 00  b.....................
data 0x0016  00 00 00 00 00 00 00 00 21 00 00 f8 fe ff 00 00 00 00 00 00 00 20  ........!..øþÿ.......
data 0x002C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x0042  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00  ......................
data 0x0058  00 00 00 00 00 00 00 23 10 03 00 00 00 00 00 00 00 00 00 00 00 00  .......#..............
data 0x006E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x0084  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x009A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
data 0x00B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ......................
```

Stopped   Ln 25   Col 1   Ch 1   INS

File Edit View VAssistX ASF Project Build Debug Tools Window Help
Advanced Mode
Quick Launch (Ctrl+Q)
Debug
Debug Browser
Hex
ATmega8
Simulator
main.asm
Anshul2
100 %
Autos
Name
Value
Type
Autos Locals Watch 1 Watch 2
ASF Explorer Processor Status Solution Explorer Properties
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
91°F
Partly sunny
ENG
IN
16:59
14-09-2022

4.3 Registers

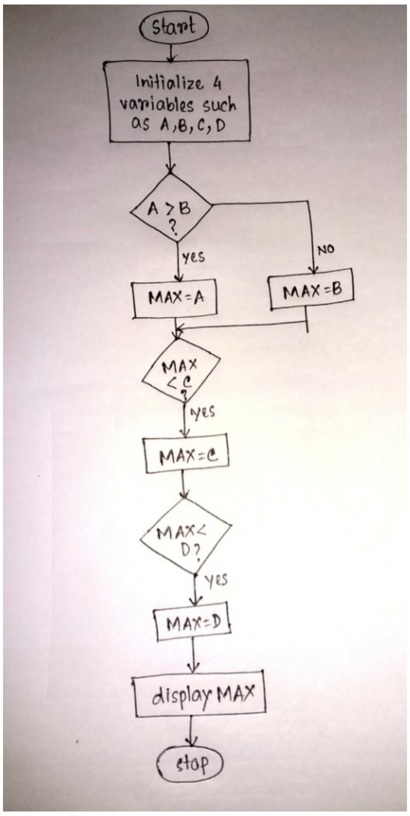
We require 2 registers to store product of two 8-bit numbers i.e a 16-bit number. We use R17 to store the first 8 bits and R16 to store the other 8 bits of the number So,

$$62 \text{ x } 08 = 0310$$
$$\text{R17} \leftarrow 03$$
$$\text{R16} \leftarrow 10$$

5 Largest of Numbers Given

We use CP instruction to compare and BRSH to branch and output the largest number in this algorithm.

5.1 Algorithm

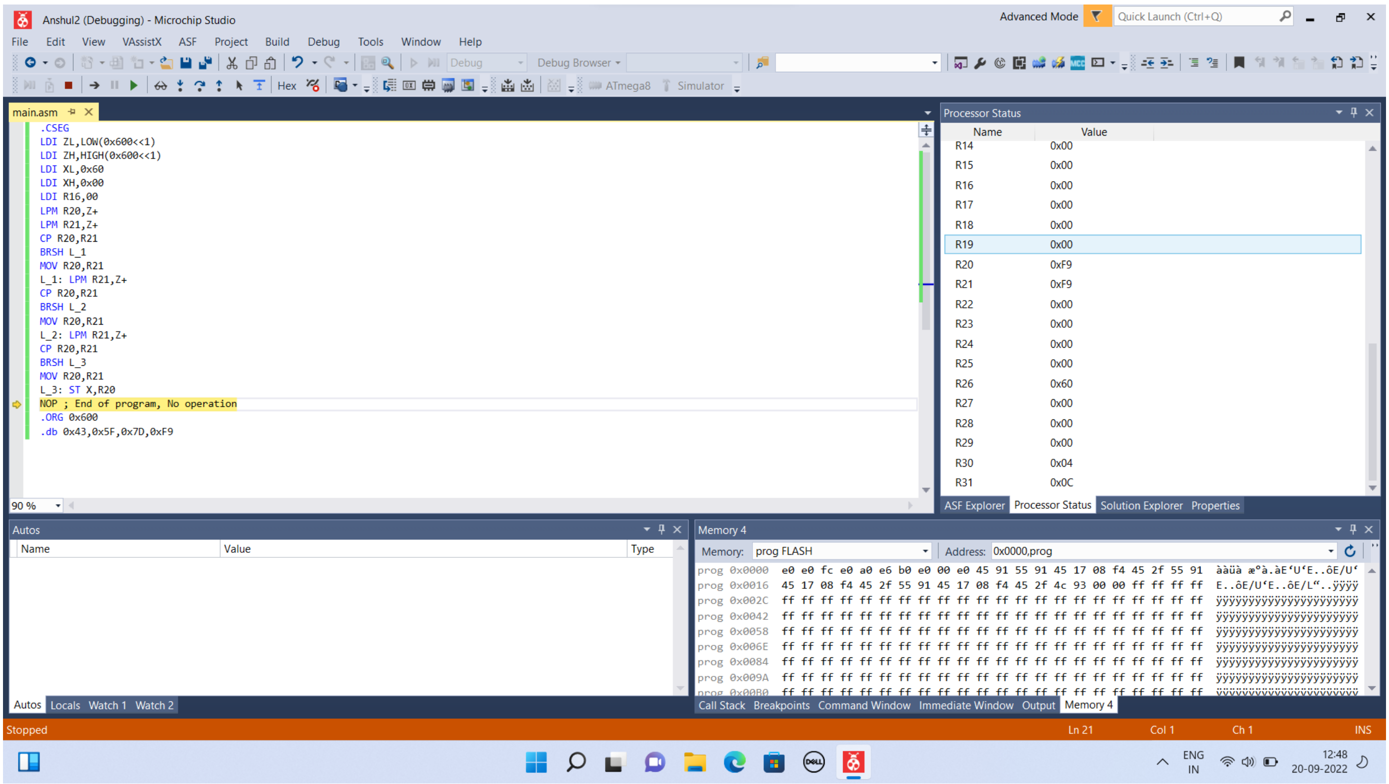


5.2 Code

```
.CSEG
LDI ZL,LOW(0x600<<1)
LDI ZH,HIGH(0x600<<1)

LDI XL,0x60
LDI XH,0x00

LDI R16,00
LPM R20,Z+
LPM R21,Z+

CP R20,R21
BRSH L_1
MOV R20,R21
L_1: LPM R21,Z+

CP R20,R21
BRSH L_2
MOV R20,R21
```

```
L_2: LPM R21,Z+

CP R20,R21
BRSH L_3
MOV R20,R21
L_3: ST X,R20
NOP ; End of program, No operation
.ORG 0x600
.db 0x43,0x5F,0x7D,0xF9
```

5.3 Output



Anshul2 (Debugging) - Microchip Studio

main.asm

```
.CSEG
LDI ZL,LOW(0x600<<1)
LDI ZH,HIGH(0x600<<1)
LDI XL,0x60
LDI XH,0x00
LDI R16,00
LPM R20,Z+
LPM R21,Z+
CP R20,R21
BRSH L_1
MOV R20,R21
L_1: LPM R21,Z+
CP R20,R21
BRSH L_2
MOV R20,R21
L_2: LPM R21,Z+
CP R20,R21
BRSH L_3
MOV R20,R21
L_3: ST X,R20
NOP ; End of program, No operation
.ORG 0x600
.db 0x43,0x5F,0x7D,0xF9
```

Processor Status

Name	Value
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0xF9
R21	0xF9
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x60
R27	0x00
R28	0x00
R29	0x00
R30	0x04
R31	0x0C

Memory 4

Memory: prog FLASH Address: 0x0000,prog

```
prog 0x0000  e0 e0 fc e0 a0 e6 b0 e0 00 e0 45 91 55 91 45 17 08 f4 45 2f 55 91  àâüà æ°à.àE‘U‘E..ôE/U‘
prog 0x0016  45 17 08 f4 45 2f 55 91 45 17 08 f4 45 2f 4c 93 00 00 ff ff ff ff  E..ôE/U‘E..ôE/L“..ÿÿÿÿ
prog 0x002C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0042  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0058  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x006E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0084  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x009A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00B0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

5.4 Registers

We require 2 registers, R21 to store each number and R20 to store the largest number and compare.
So,

$$\mathrm{R17} \leftarrow F9$$
$$\mathrm{R16} \leftarrow F9$$