EE2016 : Microprocessor Theory and Lab

Lab Report # 2

Computations using Atmel Atmega8 AVR through Assembly Program Emulation

Ajoe George, EE21B009
Mandla Siva Manoj, EE21B083
Group 3

October 3, 2022

Contents

List of Figures

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.

In this experiment we will

- Calculate the Sum of two 8-bit numbers
- Calculate the Sum of two 16-bit numbers
- Calculate the Product of two 8-bit numbers
- Identifying the largest value from a given set of numbers

2 8-Bit Addition

2.1 Flowchart

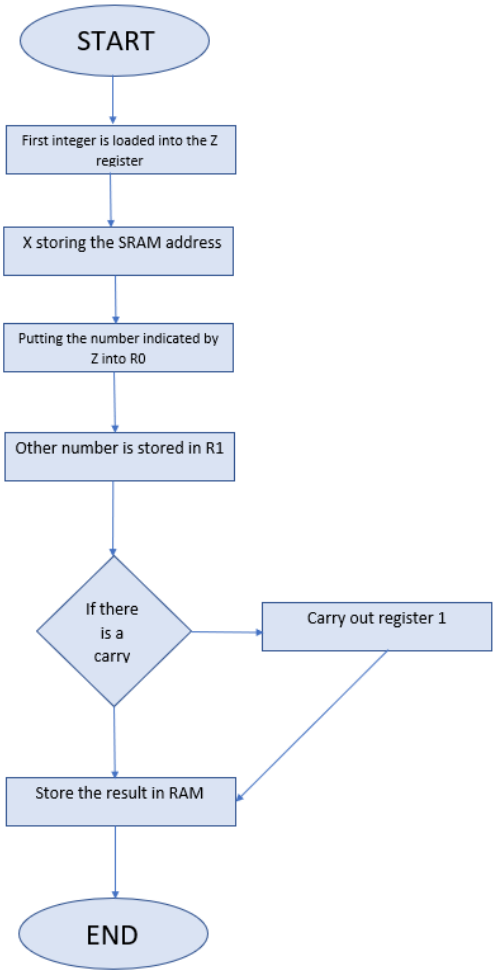

Figure 1: Flowchart for 8-bit Addition

2.2 Code

```
.CSEG ; define memory space to hold program - code segment
 LDI ZL,LOW(NUM<<1);load byte address (not word address)of
 LDI ZH,HIGH(NUM<<1); first data byte (first number)
 LDI XL,0x60;load SRAM address in X-register
 LDI XH,0x00; MSB byte
 LDI R16,00; clear R16, used to hold carry
 LPM R0,Z+; Z now points to a single byte - first number
 LPM R1,Z; Get second number into R1
 ADD R0,R1; Add R0 and R1,result in R0,carry flag affected
 BRCC abc; jump if no carry,
 LDI R16,0x01 ; else make carry 1
 abc: ST X+,R0 ; store result in RAM
 ST X,R16 ; store carry in next location
 NOP ; End of program, No operation

 NUM: .db 0xD3,0x5F; bytes to be added
; .db define data byte directive, inserts one or more
; constant bytes in the code segment (The number of
; inserted bytes must be even, otherwise the
; assembler will insert an extra zero byte)
```

Listing 1: Code for 8-bit addition

2.3 Results

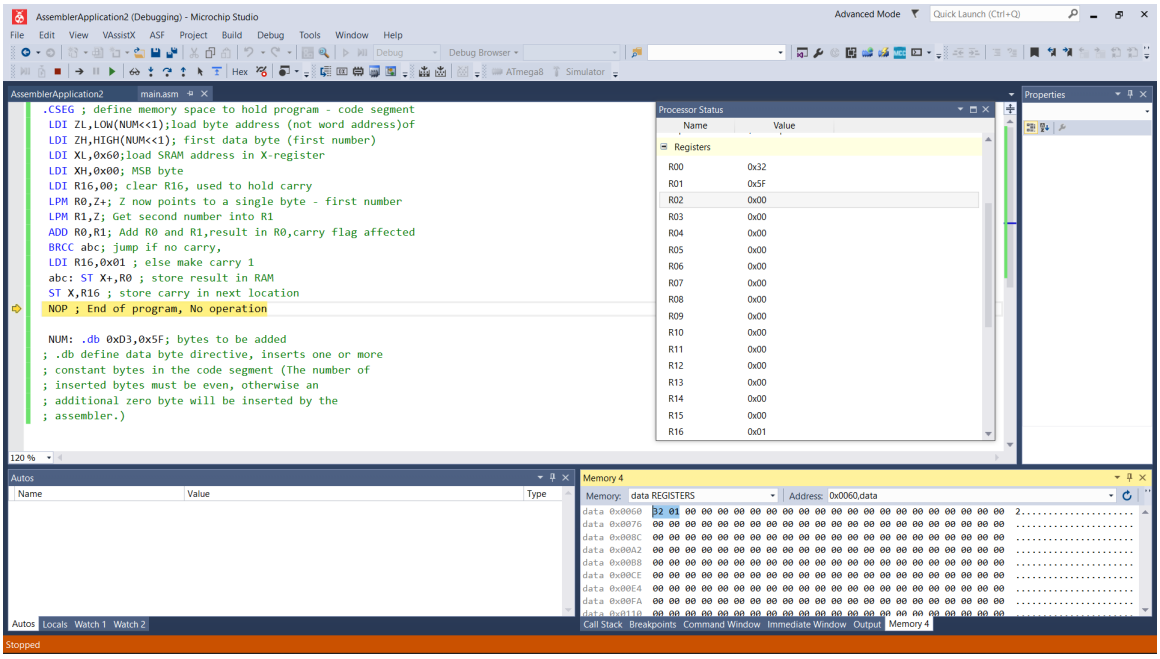

Figure 2: Output of 8-bit Addition

3 16-Bit Addition

3.1 Flowchart

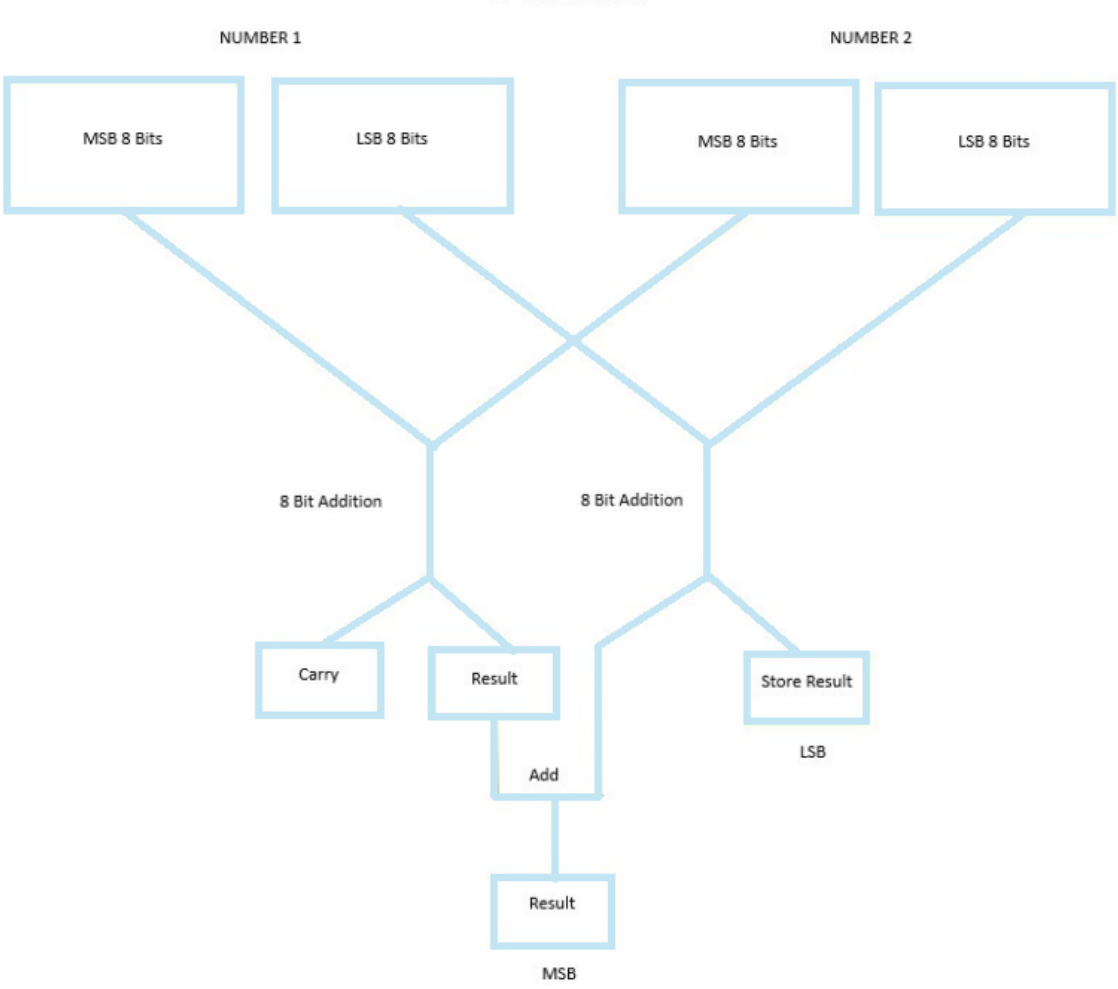

Figure 3: Flowchart for 16-bit Addition

3.2 Code

```
.CSEG; Start program

START:
//Getting NUM1 from FLASH
LDI ZL, LOW(NUM<<1);
LDI ZH, HIGH(NUM<<1); Z holds the FLASH address where the data is stored.
LPM R0, Z+;
LPM R1, Z+;

//Getting NUM2 from FLASH
LPM R20, Z+;
LPM R21, Z;

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

ADDITION:
//Initialising Low and High Bytes of result with NUM1
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)

END:
NOP; End of program

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

Listing 2: Code for 16-bit addition

3.3 Results

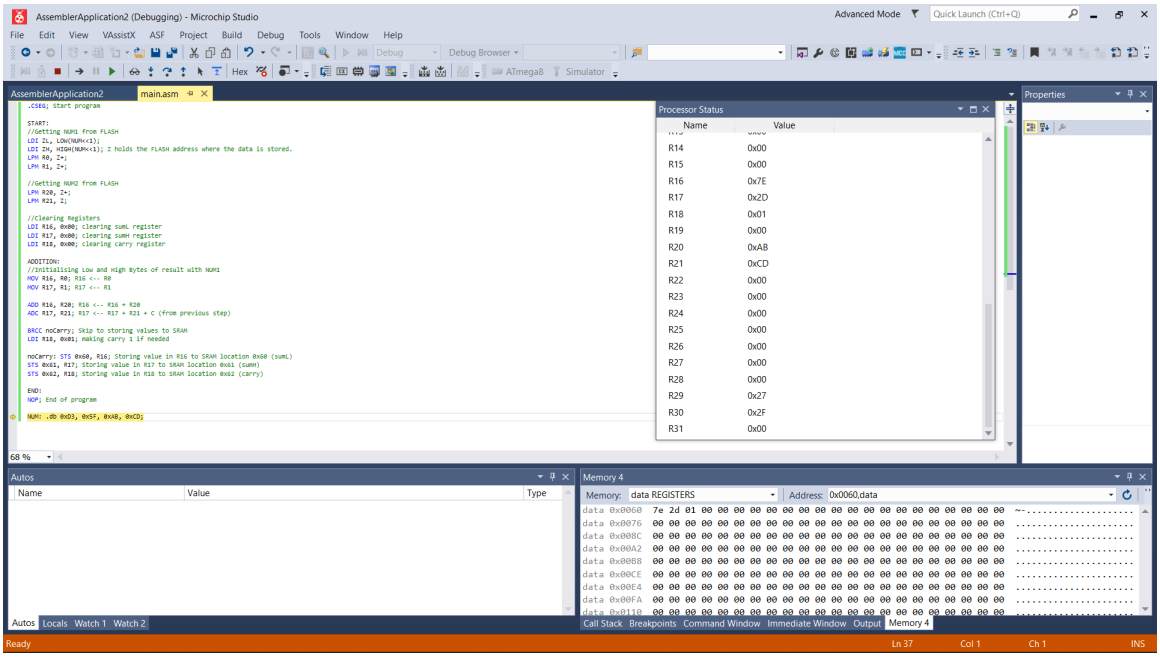

Figure 4: Output of 16-bit addition

4 8-Bit Multiplication

4.1 Flowchart

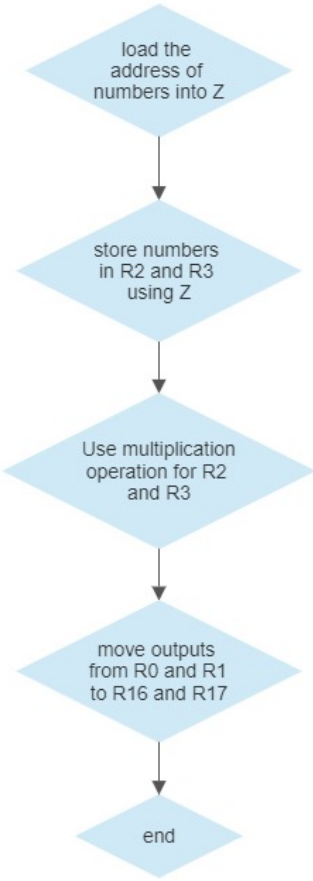

Figure 5: Flowchart of 8-bit Multiplication

4.2 Code

```
.CSEG; Start Program

START:
LDI ZL, LOW(NUM<<1);
LDI ZH, HIGH(NUM<<1); Z holds the FLASH address where the data is stored.
LPM R0, Z+; Multiplicand is loaded from FLASH, then Z = Z + 1
LPM R1, Z; Multiplier is loaded from the next location pointed by Z
LDI R16, 0x00; clearing productL register
LDI R17, 0x00; clearing productH register
LDI R18, 0x00; clearing temporary register

MULTIPLY1:
CLC; clear Carry Bit
ROR R1; right rotation of R0
BRCC MULTIPLY2; go to next step when last bit (carry now) is cleared.
ADD R16, R0;
```

```
ADC R17, R18;

MULTIPLY2:
CLC;
ROL R0;
ROL R18;
TST R1;
BRNE MULTIPLY1;
STS 0x60, R16; Storing value of R16 to SRAM location 0x60
STS 0x61, R17; Storing value of R17 to SRAM location 0x61

END:
NOP; End of Program

NUM: .db 0xD3, 0x5F; Inputs are defined in FLASH locations pointed by label
↪  NUM
```

Listing 3: Code for 8-bit multiplication

4.3 Results

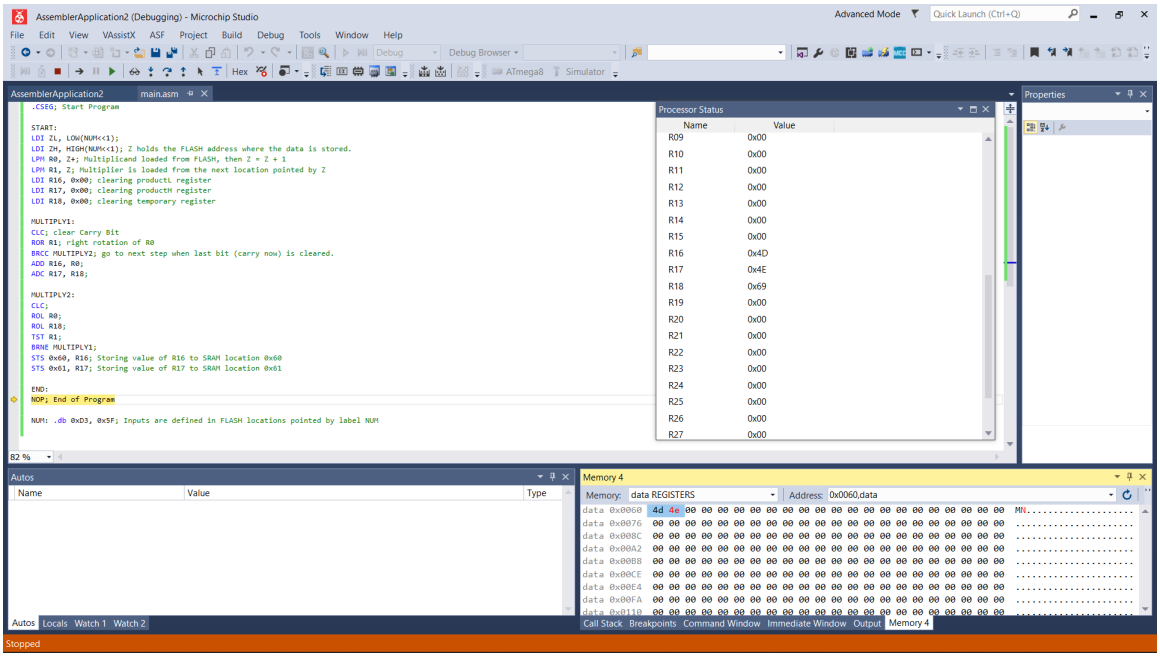

Figure 6: Output of 8-bit Multiplication

5 Comparison

5.1 Flowchart

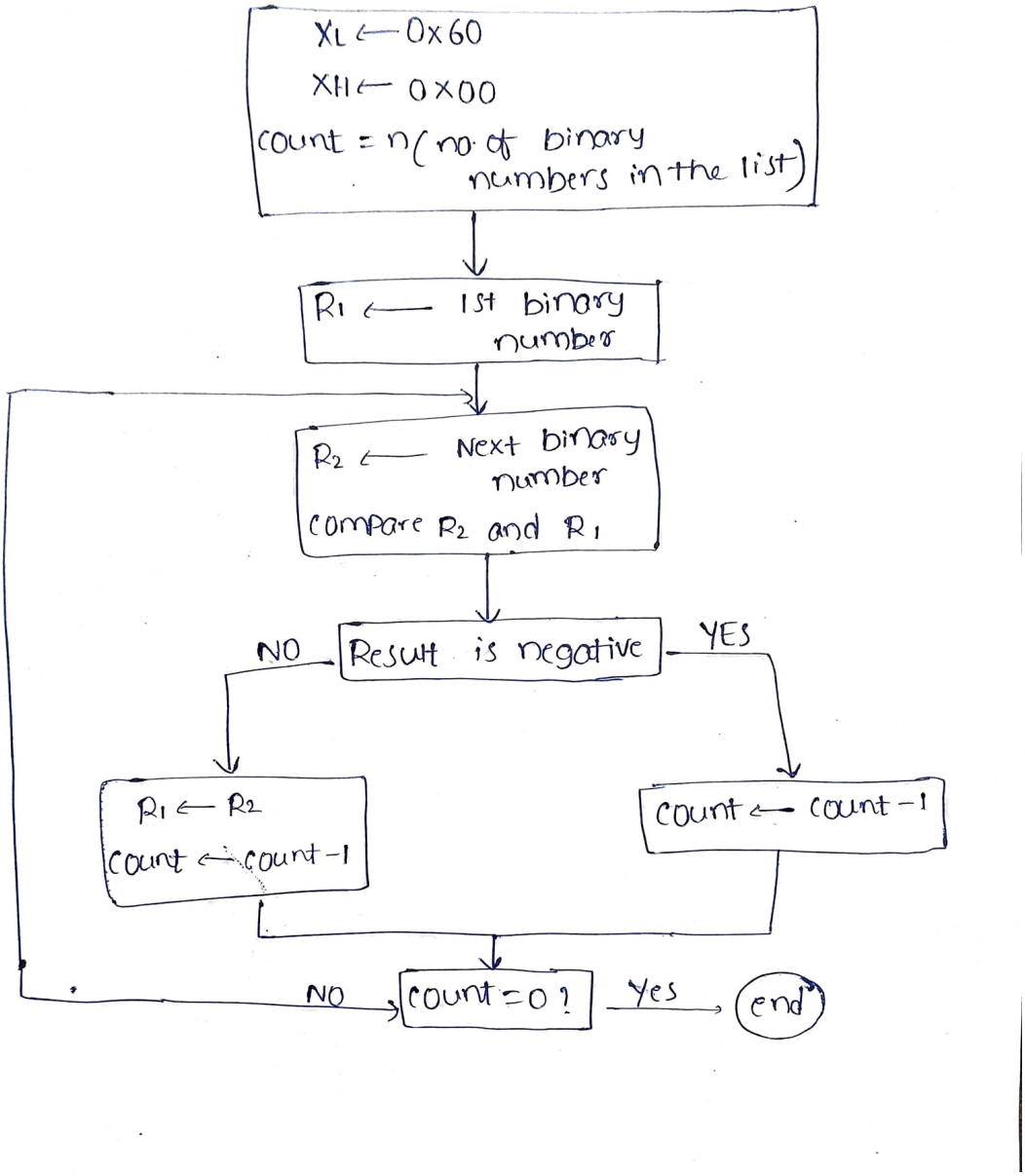

Figure 7: Flowchart for Comparison

5.2 Code

```
.CSEG;
LDI ZL, LOW(NUM<<);
LDI ZH, HIGH(NUM<<1);
LDI XL, 0x60;
LDI XH, 0x00;
```

```
6  LPM R0, Z+;
7  DEC R0;
8  LPM R1, Z+;
9  LOOP: LPM R2, Z+;
10 CP R2, R1;
11 BRMI shift;
12 MOV R1, R2;
13 shift: DEC R0;
14 BRNE LOOP;
15 ST X, R1;
16 NOP;
17 NUM: .db 0x05, 0x04, 0x02, 0x01, 0x11;
```

Listing 4: Code for comparison

5.3 Results

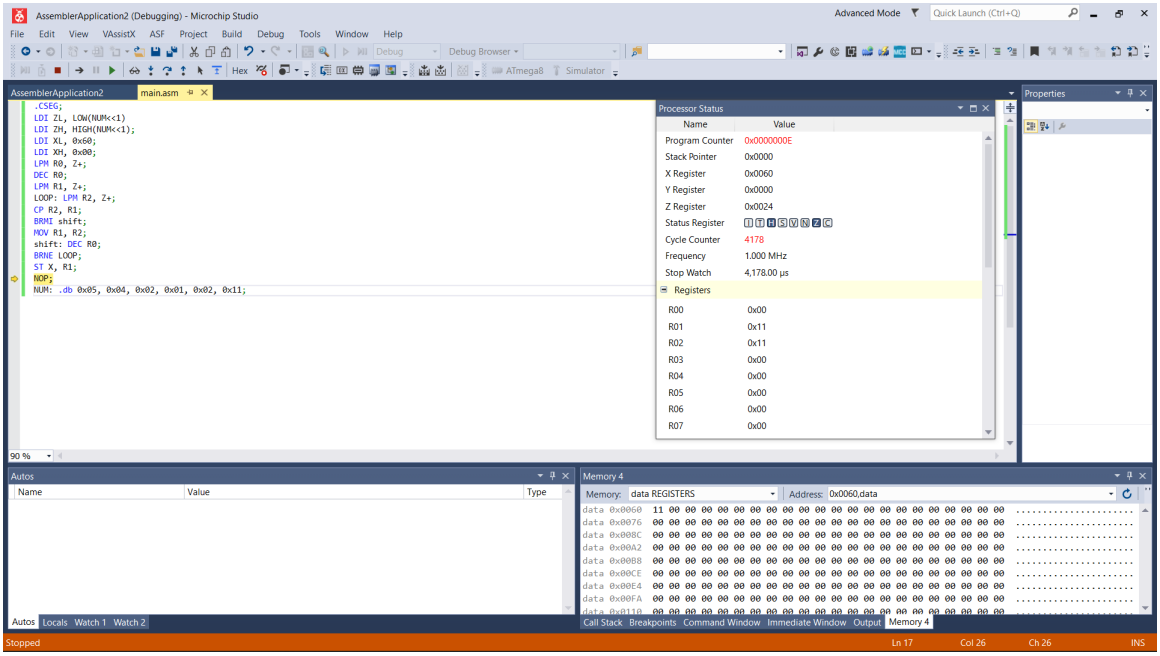

Figure 8: Output of Comparison

6 Learning Outcomes

By doing this experiment we were able to:

- Learn the logic of various operations like 8-bit addition, 16-bit addition, 8-bit multiplication and comparison.
- Learn how to program in assembly language.
- Get familiar with the software Atmel Studio