EE2016 : Microprocessor Theory and Lab

Lab Experiment 2

Computations using Atmel Atmega8 AVR through Assembly Program Emulation

SarathChandra koppera,EE21B073
Vikas Gollapalli,EE21B051

Batch 22

October 3, 2022

Contents

1 Introduction

Aim of the Experiment

To implement basic arithmetic and logical manipulation programs using Atmel Atmega8 microcontroller in assembly program emulation. Implement for 8bit and 16 bit binary numbers:

1. Given two 8-bit binary words (byte), compute the sum and store it in a register.
2. Given two 16-bit binary words (byte), compute the sum and store it in a register.
3. Given two 8-bit binary words (byte), compute the product and store it in aregister.
4. Given a finite set of binary words, identify the largest in the given set.

Equipments

- Windows PC with Microchip Studio IDE (Integrated Development Environment) for developing and debugging AVR
- SAM microcontroller applications.

Familiarity Required

The basic concepts of microcontroller architecture, general purpose programming with the corresponding instruction sets should be very clear. In Experiment 2, our focus is only on the emulation of assembly programming of Atmega8 microcontroller.

Emulation of Atmel AVR Assembly Programming

In this experiment, we adopt the second approach, i.e., emulation of Atmega8 microcontroller (leaving the real-life atmega8 microcontroller programming to the next week) in IDE.

Basic Concepts Required

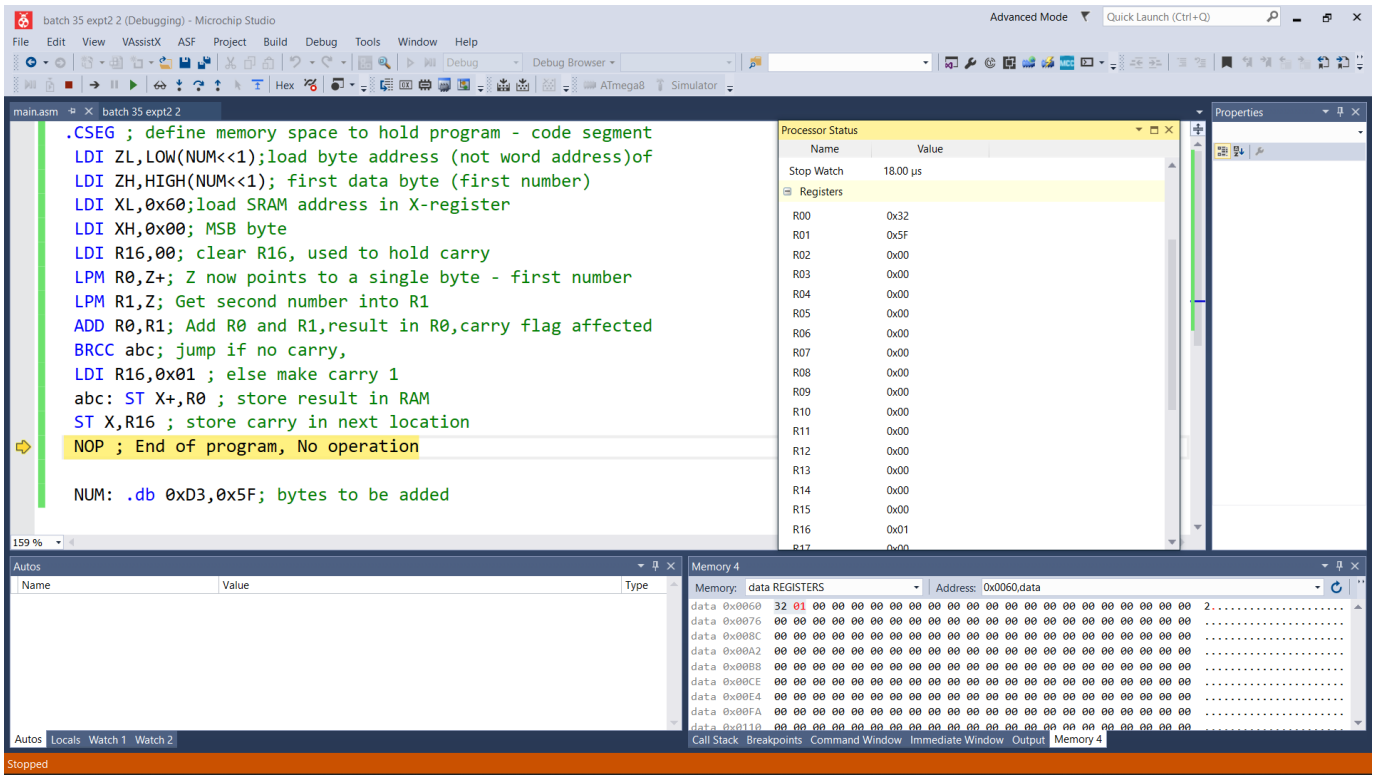
- Atmel AVR Instruction Set
- Atmel Atmel AVR Assembly Programming limited to
    5. Addition]
    6. Multiplication
    7. Comparison
- Atmega8 Microcontroller Architecture
- Atmel AVR Studio 7 simulator

2 8 Bit Addition

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
```

Results

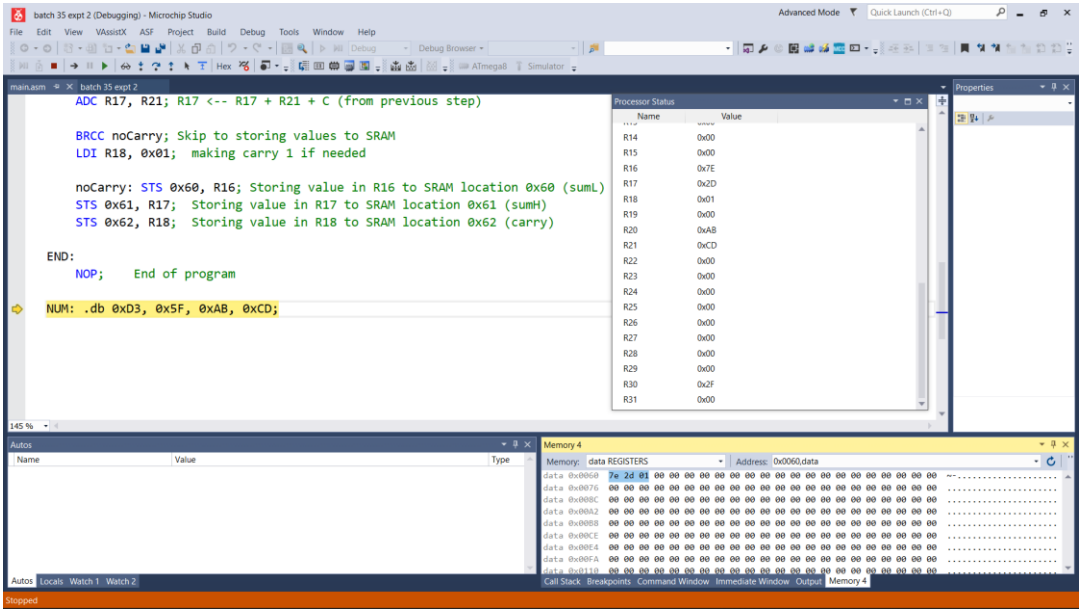


Result of 8 bit addition

3 16 Bit Addition 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;
```

3.1 Results



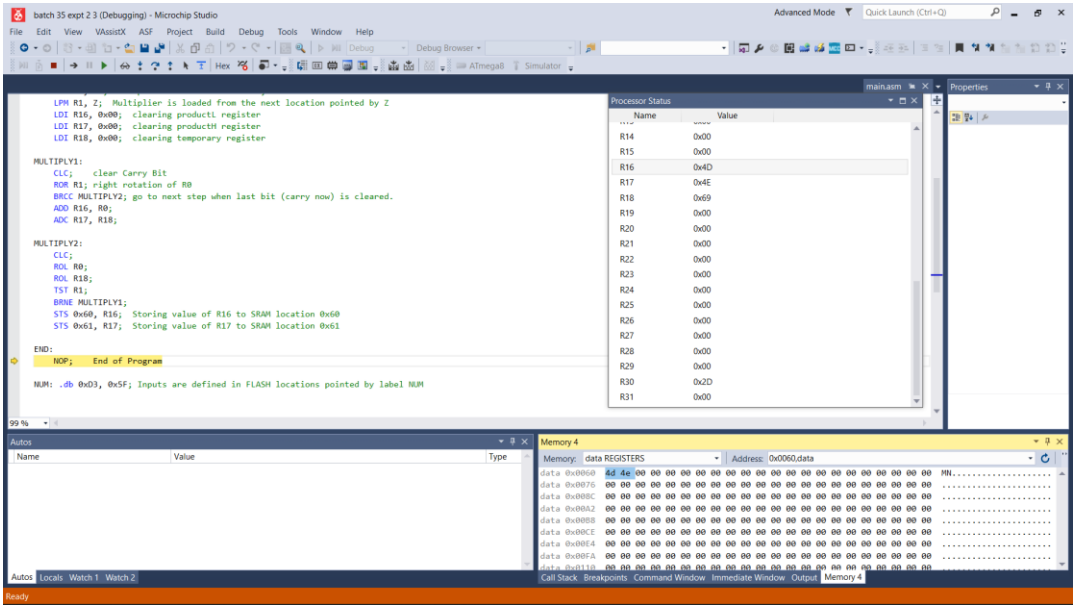
Result of 16 bit addition

4 Multiplication

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 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
```

Results

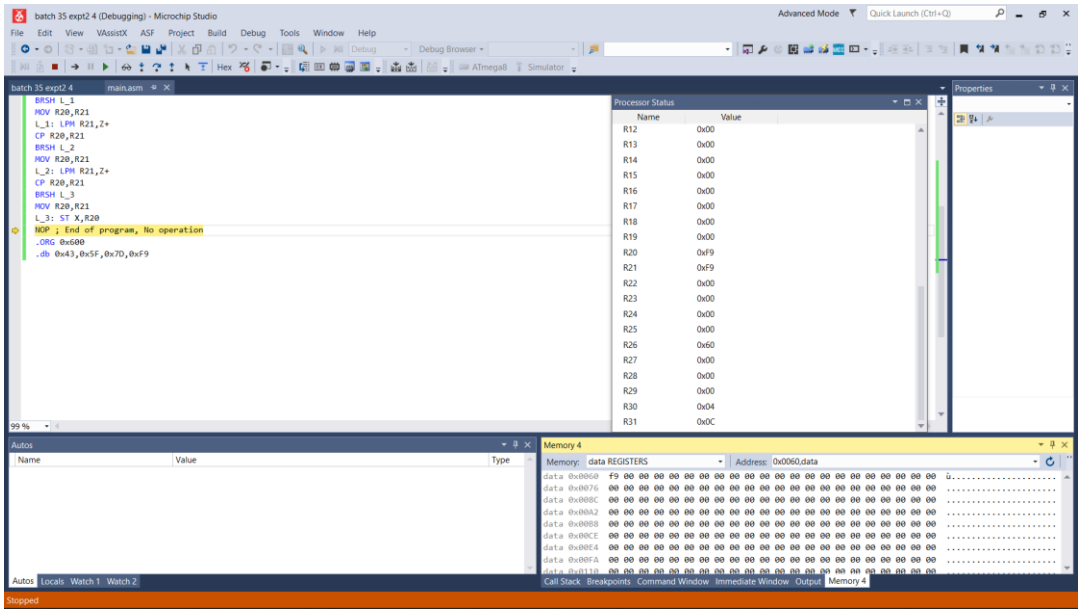


Result of 8 bit multipliication

5 Comparison 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
```

Results



Result of comparison of numbers

6 Inference

Addition, substraction, multiplication, and comparison of 8-bit / 16-bit binary numbers aredemonstrated through emulation of Atmega8 assembly programming.

7 Learning Outcomes

1. We learnt to write the code in assembly level language
2. We learnt to implement the logic of addition, comparision and substraction