

Emulating the Atmel AVR Microcontroller using AVR Assembly Language Programming on Microchip Studio

EE2016 Microprocessors

Sanjeev Subrahmaniyan S B

EE23B102

Abstract

The Atmel AVR microcontroller series are a RISC based 8-bit processor suited for beginners and professionals to implement small scale computational and embedded applications. In this experiment, the Atmel ATmega8 microcontroller is emulated on the Microchip studio software. Assembly language programming is used to implement simple programs. The simulators interfaces are used to monitor the progress of the programs and gain experience of assembly programming and memory handling. The implemented programs include addition and multiplication of 8-bit numbers, addition of 16-bit numbers and iterative maximum finding on the 8-bit microcontroller.

1 Introduction

Microcontrollers are indispensable in the electronic industry, being the base for all prototyping and testing. In this experiment, the Atmel ATmega8 microcontroller is emulated to gain experience of assembly language programming. This also helps one familiarise with the instruction set of the microcontroller and gain an under the hood understanding of the internal working of the processor and memory elements.

2 Objectives

The tasks emulated are:

1. Addition of two 8-bit numbers

2. Addition of two 16-bit numbers
3. Multiplication of two 8-bit numbers
4. Determining the maximum element from a given finite set of 8-bit numbers

3 Assembly language programs and interpretation

3.1 Addition of two 8-bit numbers

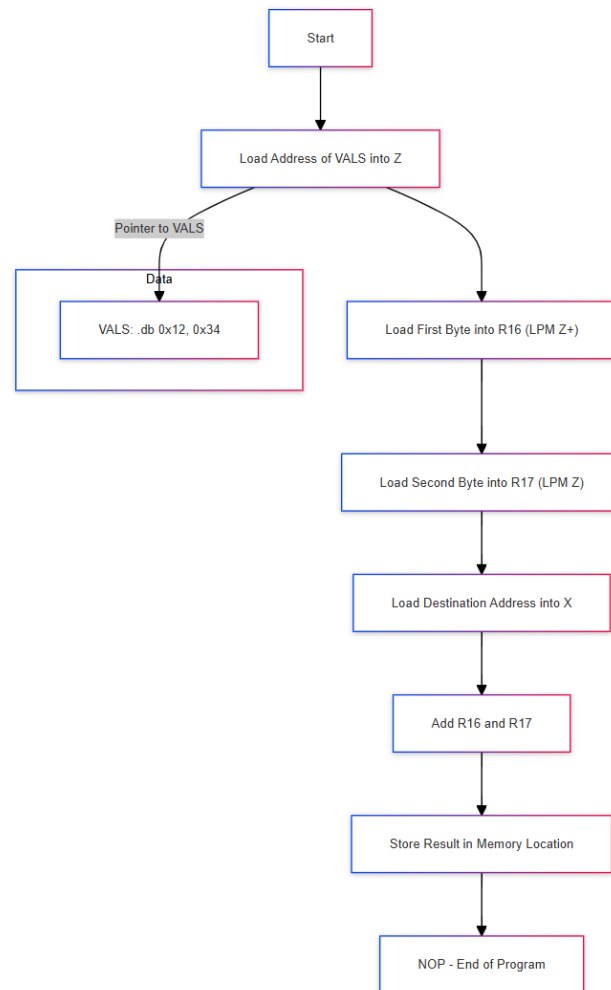


Figure 1: Pseudocode flowchart for addition

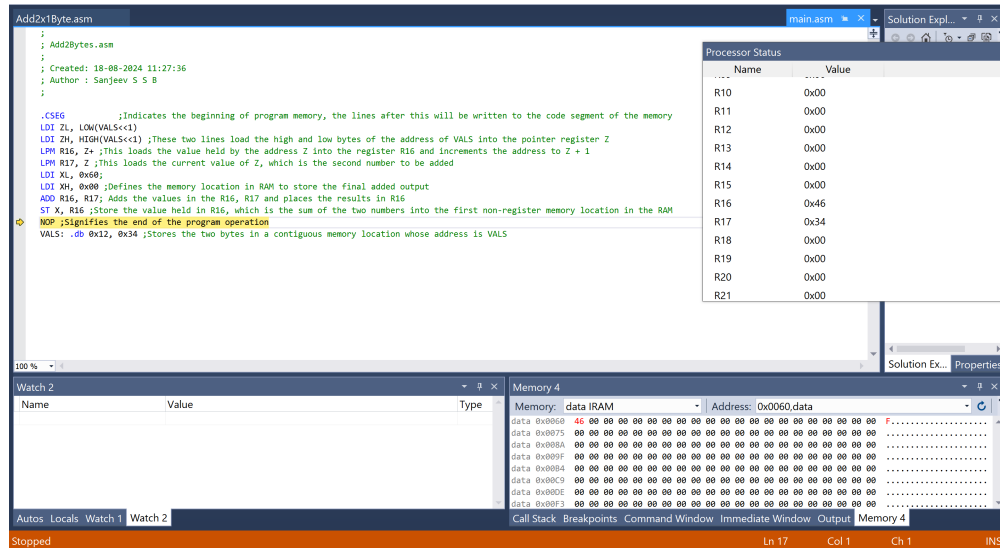


Figure 2: Addition of two 8-bit numbers

The addition of two 8-bit numbers is done using the ADD opcode. This is used because there is no carry initially. The program is explained through the comments written alongside the it. In the end, the ST operation is used to store the final answer into RAM memory, which is highlighted in red in the bottom-right corner of the image. The register values at the end of the program are also shown in the processor status dialog.

3.2 Multiplication of two 8-bit numbers

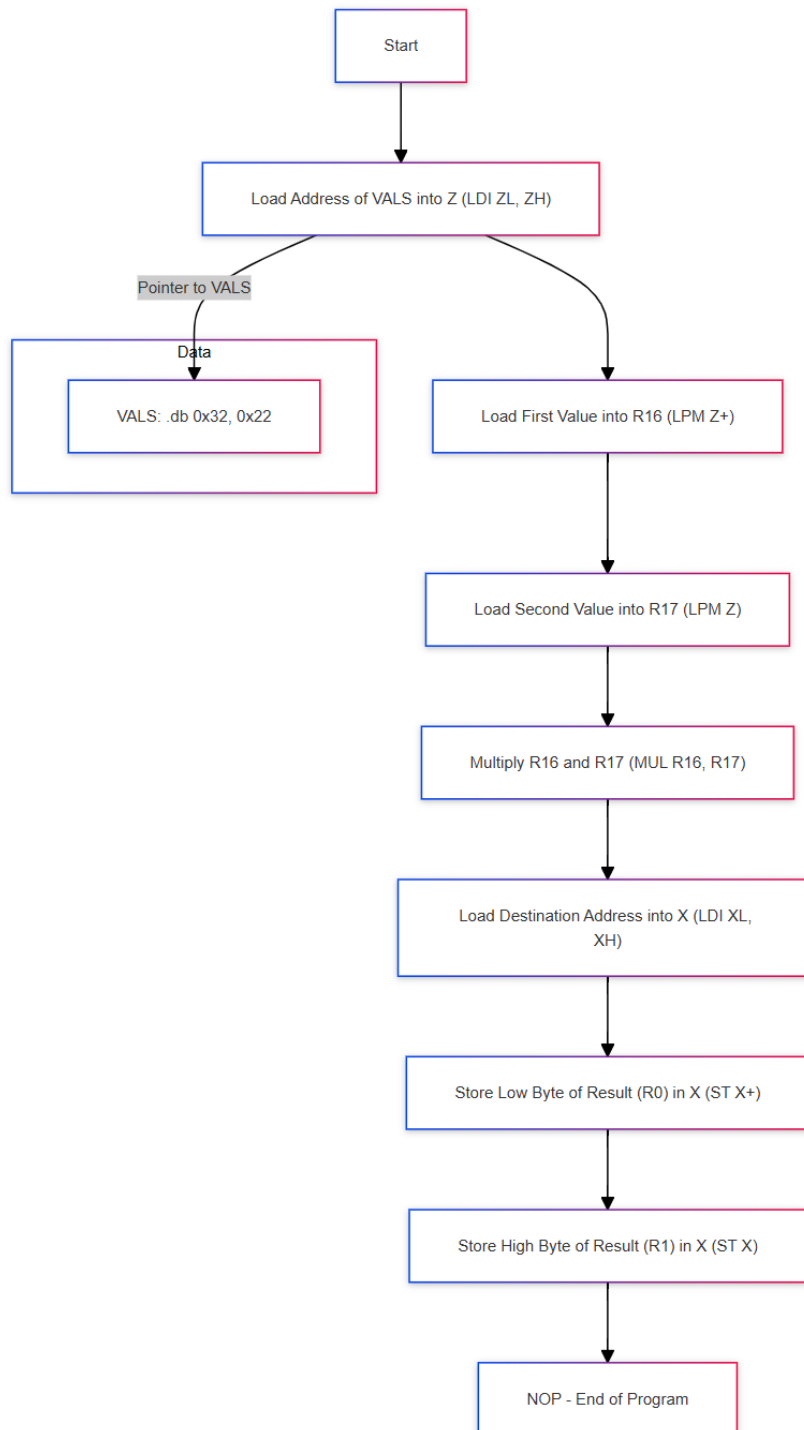


Figure 3: Flowchart for multiplication of two numbers

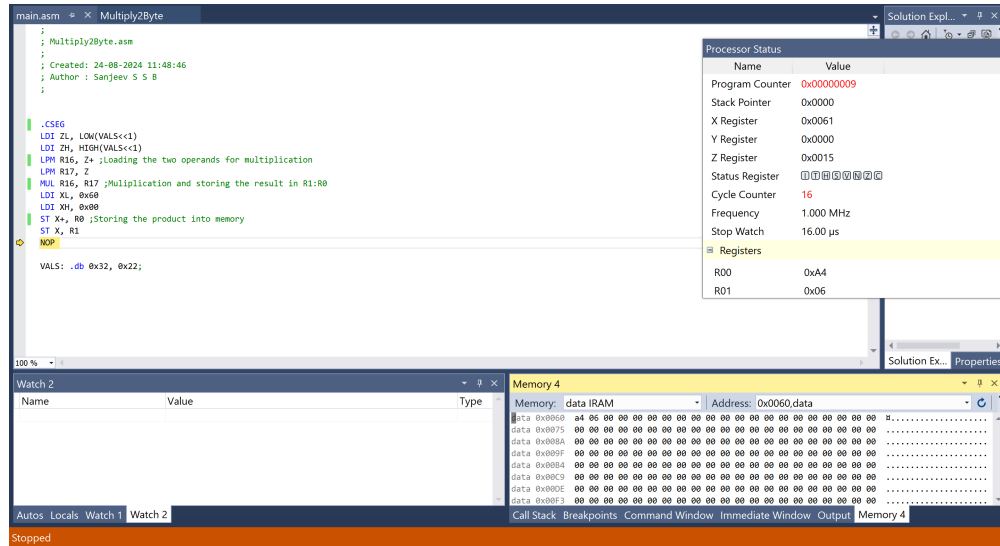


Figure 4: Multiplication of two 8-bit numbers

The multiplication of two 8-bit numbers is done similarly using the MUL operation. Similar to the previous program, the final answer, which is a 16-bit result, is stored in two contiguous locations in memory as shown in the bottom right part of the image.

3.3 Addition of two 16-bit numbers

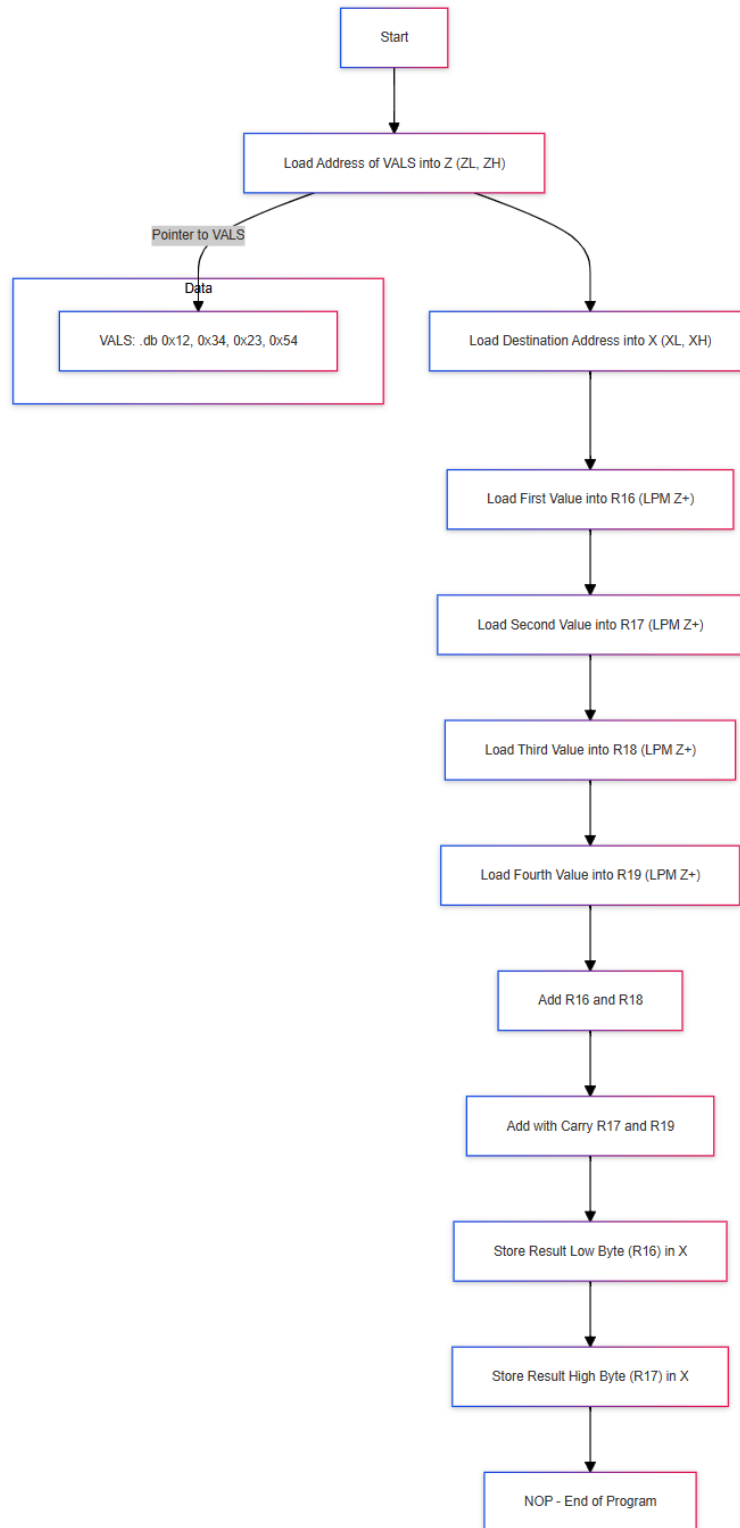


Figure 5: Pseudocode flowchart for addition of words(16-bit)

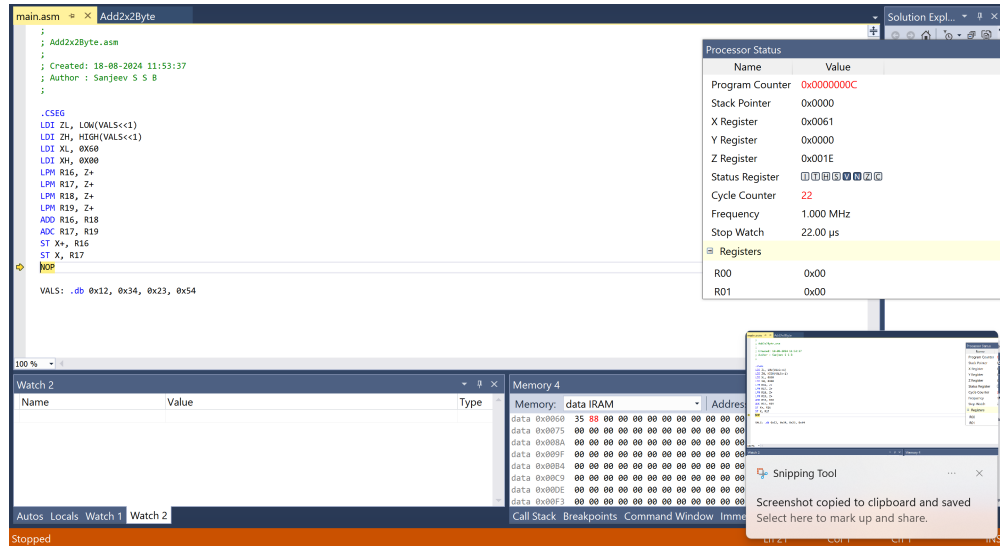


Figure 6: Addition of two 16-bit numbers

The 16-bit addition is handled by first adding the last bytes of both the numbers using the ADD operation and then using the ADC operation to add the first bytes along with the carry from the first addition. Similar to the previous programs, the final answer is stored in a memory location as seen in the bottom-right of the image.

3.4 Finding maximum of a finite set of numbers

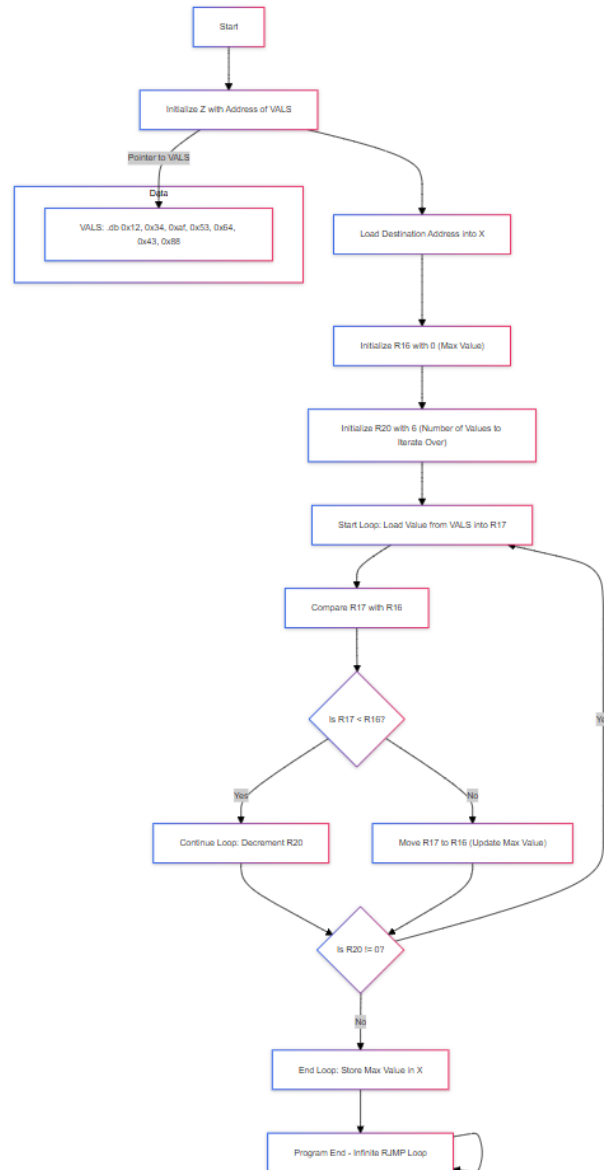


Figure 7: Pseudocode flowchart for finding the maximum of the set of numbers

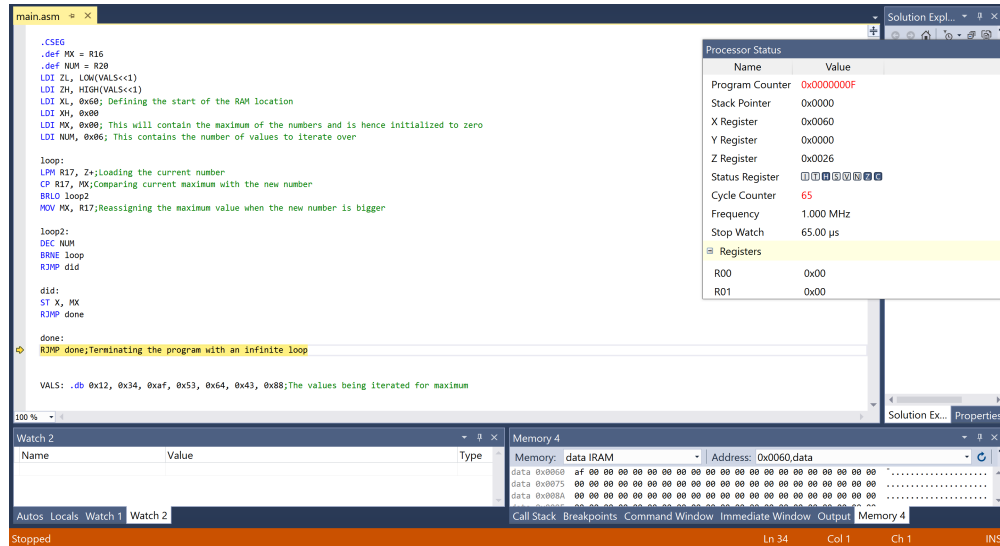


Figure 8: Finding maximum of a set of numbers

The determination of the maximum of numbers is done by iteratively comparing the values using the CP operation, which are held in a temporary register R17. A loop is implemented which checks if the new number is bigger than the current maximum and updates the maximum if that is the case. A counter variable NUM is used to keep track of the number of elements iterated over. The final answer is stored into the RAM as seen in the bottom-right of the image.

4 Procedure

The procedure undertaken for the experiment is as follows:

1. Create an assembler project on the Microchip Studio software.
2. Create flowcharts to ideate on the solution and validate its functionality.
3. Write the assembly language program to perform the given task.
4. Build the solution and debug the solution using breakpoints.
5. Monitor the processor registers and memory values to infer the completion of the program.
6. Report the observations.

5 My contribution to the experiment

1. Write the assembly programs and psedocode to perform the given tasks(because this was a simple assignment, we all wrote our own programs).
2. Debug and execute the programs and compile the results.

6 Conclusion

The experiment provided hands-on experience with assembly language programming on the Atmel ATmega8 microcontroller using Microchip Studio. By implementing fundamental operations such as addition, multiplication, and maximum finding, a deeper understanding of low-level programming concepts, processor architecture, and memory management was gained. The iterative nature of the tasks reinforced the importance of logical flow and the correct use of instructions like ADD, ADC, MUL, and CP in managing data efficiently. The use of simulation tools in Microchip Studio offered valuable insights into how microcontrollers execute instructions and handle registers and memory, contributing to a more comprehensive understanding of embedded systems development.