Experiment 2 : Computations using Atmel Atmega 8 AVR through Assembly Program Emulation

Abhinav I S EE23B002

August 26, 2024

1 Objective

To implement basic arithmetic and logical manipulation programs using Atmel Atmega8 microcontroller in asseembly program Emulation The experiment consists of four problems:

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

2 Problem 1 : Adding two binary numbers

2.1 Procedure

2.1.1 Flowchart:

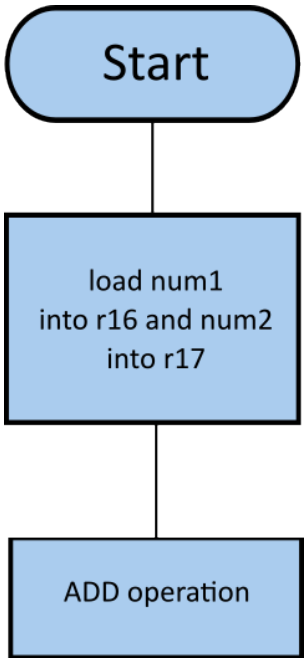


Figure 1: Flowchart for Problem1

2.1.2 Code

```
start:
  .def NUM1 = r16
  .def NUM2 = r17

  .cseg
  .org        0x00
  ldi                NUM1, 0x4
  ldi                NUM2, 0x4

  add                NUM1, NUM2
```

We can run this code in simulation in Microchip studio

1. Build the code in Microchip studio
2. Run the code in simulation in debug mode

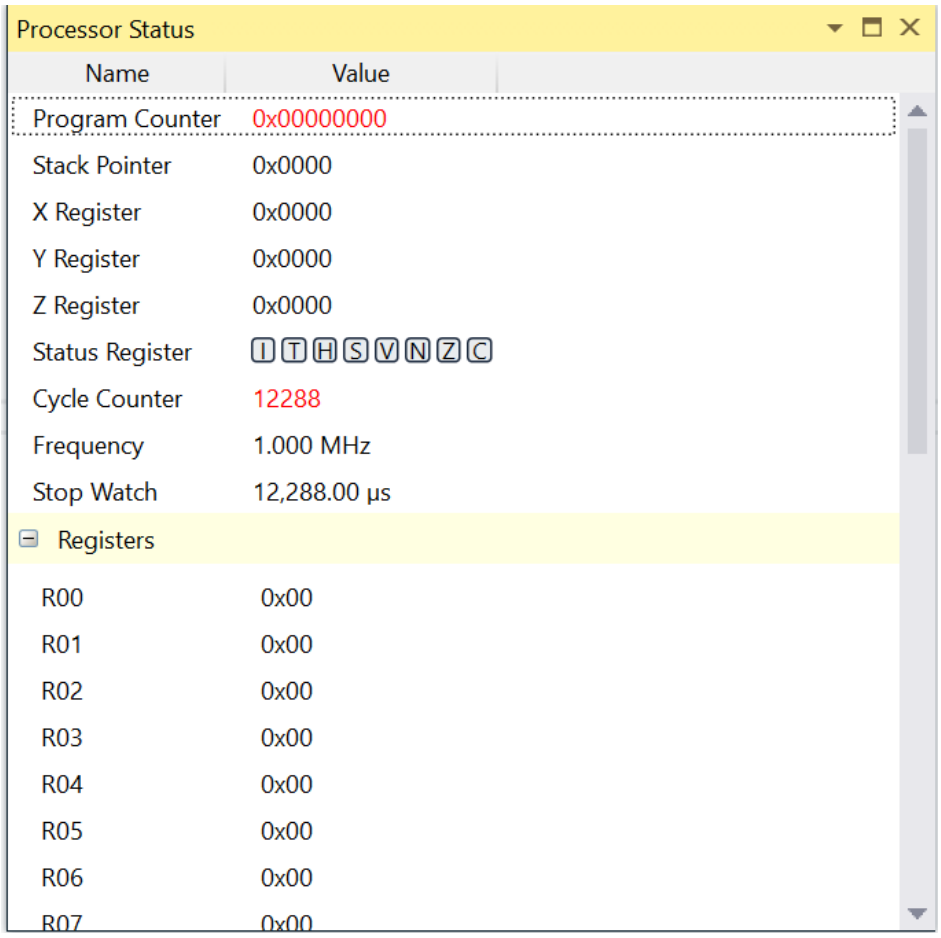


Figure 2: Registers after running the program

- The DEF directive allows the registers to be referred to through symbols. A defined symbol can be used in the rest of the program to refer to the register it is assigned to
- The CSEG directive defines the start of a Code Segment
- The ORG directive sets the location counter to an absolute value.
- LDI (Load Immediate) instruction is used to load an 8-bit constant (immediate value) directly into a register within the upper half of the register file (i.e., registers R16 to R31).
- The ADD instruction in AVR assembly is used to perform an 8-bit addition of two registers, storing the result in one of those registers.

2.2 Inputs

the Inputs are : 0x4 (4) and 0x4 (4), in registers r16 and r17 respectievely

2.3 Outputs

the output is 0x8 in r16 as shown

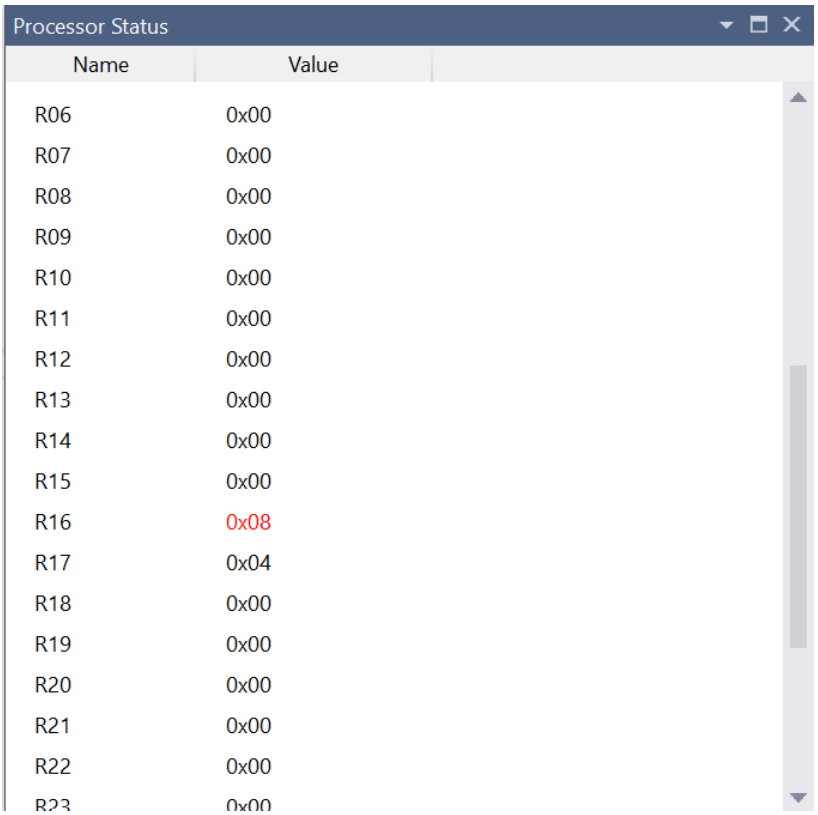


Figure 3: Output in registers

3 Problem 2 :16-bit addition using a 8-bit processor

3.1 Procedure

3.1.1 Flowchart

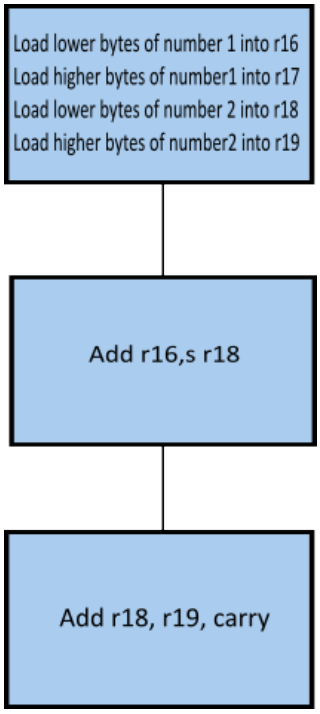


Figure 4: Flowchart

3.2 Code

```
start:
        .def        num1L = r16; define lower byte of number 1 as r16
        .def        num1H = r17        ; define upper byte of number 1 as r17
        .def        num2L = r18        ; define lower byte of number 2 as r18
        .def        num2H = r19        ; define upper byte of number 2 as r19

        .cseg
        .org        0x00
        ldi        num1L,0x16        ; load 0x56 into r16
        ldi        num1H,0x14        ; load 0x34 into r17
        ldi        num2L,0x18        ; load 0x78 into r18
        ldi        num2H,0x12        ; load 0x12 into r19

        add        num1L,num2L        ; add lower bytes of number
        adc        num1H,num2H        ; add upper bytes of number with carry
```

- The new instruction being used here is the ADC instruction
  ADC Rd, Rr: Adds the contents of the register Rr, the contents of the register Rd, and the carry flag to Rd

3.3 Inputs

the inputs are num1 = 0x1416, num2 = 0x1218

3.4 Outputs

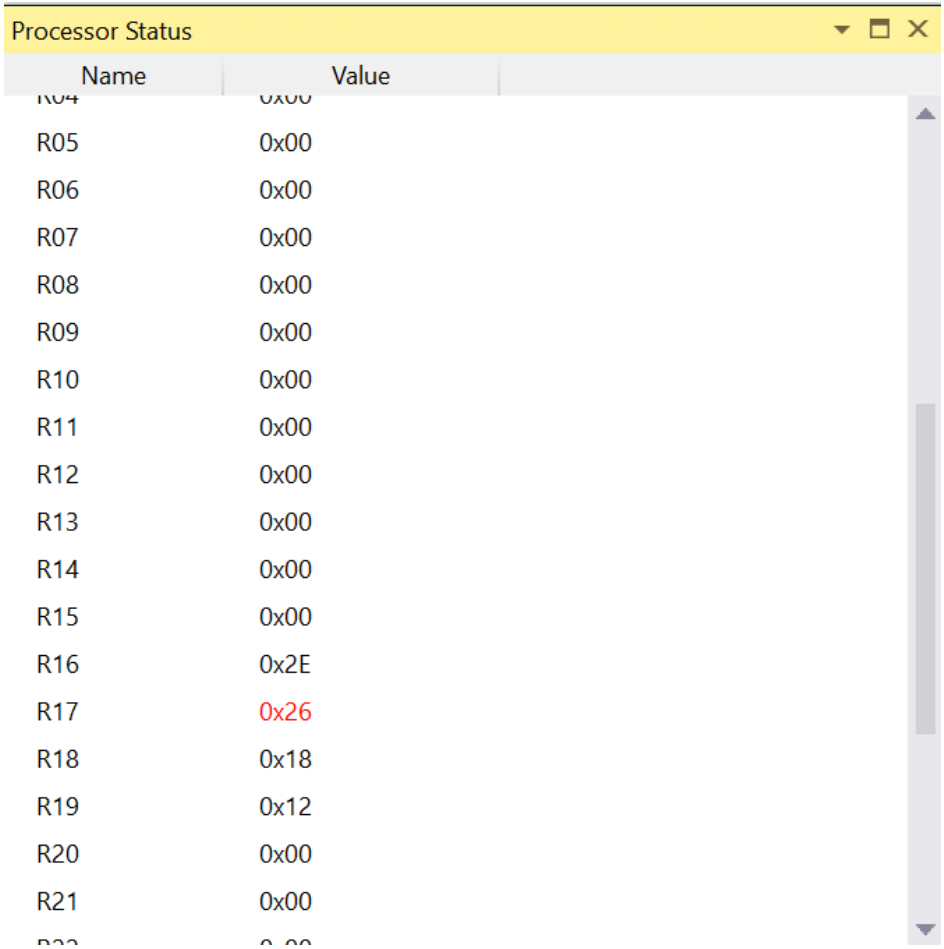


Figure 5: Outputs

The output is 0x262E

4 Problem 3 : Multiplication of 2 8 bit numbers

4.1 Procedure

4.1.1 Flowchart

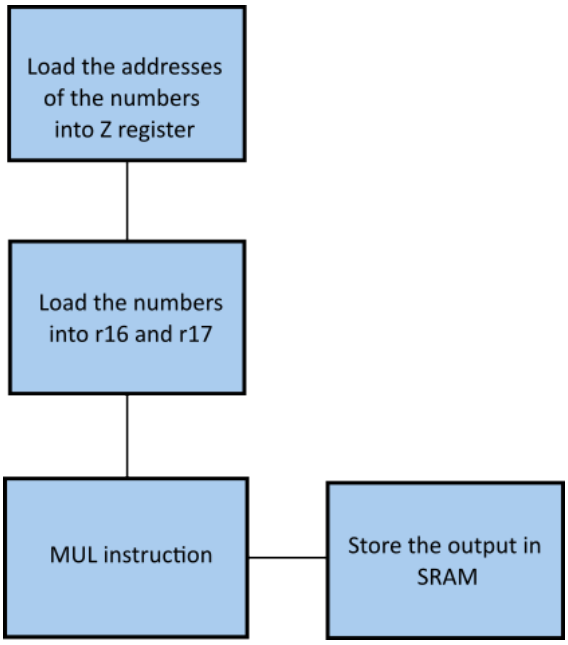


Figure 6: flowchart

4.1.2 Code

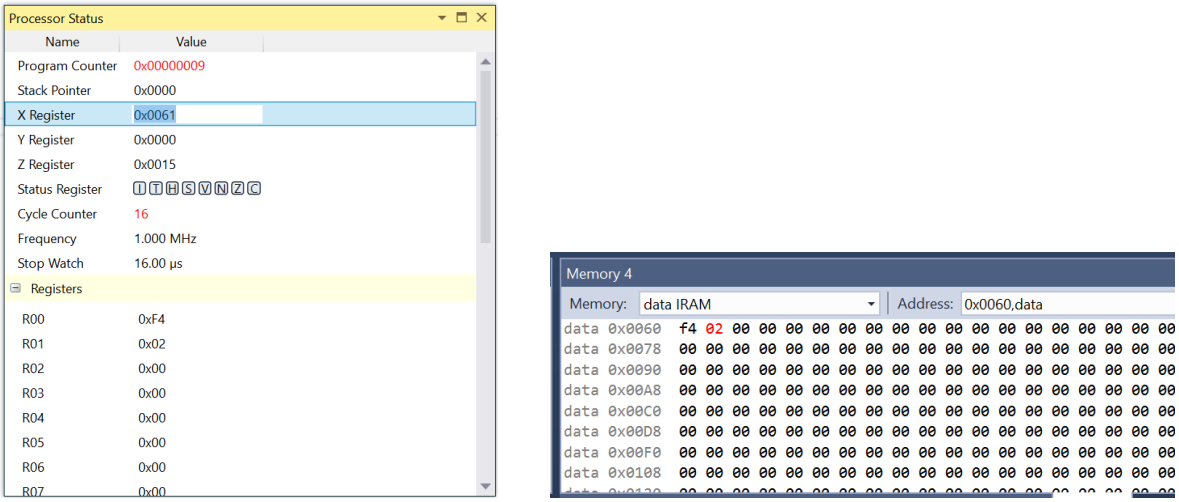
```
.CSEG
LDI ZL, LOW(VALS<<1)
LDI ZH, HIGH(VALS<<1)
LPM R16, Z+
LPM R17, Z
MUL R16, R17
LDI XL, 0x60
LDI XH, 0x00
ST X+, R0
ST X, R1
NOP

VALS: .db 0x12, 0x2A;
```

- We build and simulate the program in Microchip studio
- Since program memory is word addressed, that is each address in program memory holds 2 bytes, and Z register is byte addressed, we need to do a left shift, before loading it into ZL
- LPM instruction loads the value in the address in Z to r16, Z+ increments the Z register, so the subsequent value gets stored in r17
- Further , we store 0x60 in the X register, because it is where the RAM begins

4.2 Inputs

The inputs are 0x12 and 0x2A

4.3 Outputs



(a) The r0, r1 register holding th output

(b) The value stored in RAM (at 0x60)

Figure 7: Outputs for the program

Output is 2F4 (Answer is in little endian in memory)

5 Problem4 : Given a finite set of binary words, iden tify the largest in the given set

5.1 Procedure

5.1.1 Flowchart

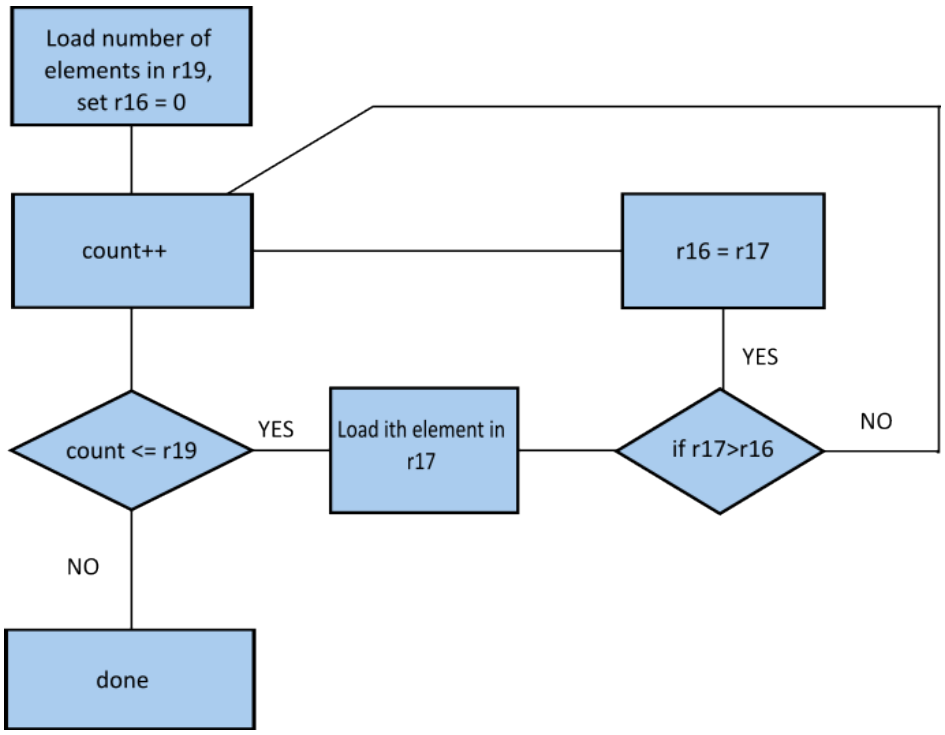


Figure 8: Flowchart

5.1.2 Code

```
;
; Lab1.asm
;
; Created: 20-08-2024 13:28:14
; Author : abhin
;


; Replace with your application code
start:
        ldi r16, 0
        ldi r18, 1
        ldi r19, 5

        ldi ZL, LOW(2*mynum)
loop_start:
        lpm r17, Z+
```

```
        ldi r20, 1
        add r18, r20
        cp r18, r19
        brge done

        cp r17, r16
        brge greater_label
        rjmp loop_start




greater_label:
        mov r16, r17
        rjmp loop_start

done:
        rjmp done

mynum: .db        0x05,0x09,0x08,0x07
```

- my num contains 4 numbers, 0x5, 0x9, 0x8, 0x7

5.2 Inputs

The input set of numberrs are 5, 9, 8, 7

5.3 Outputs

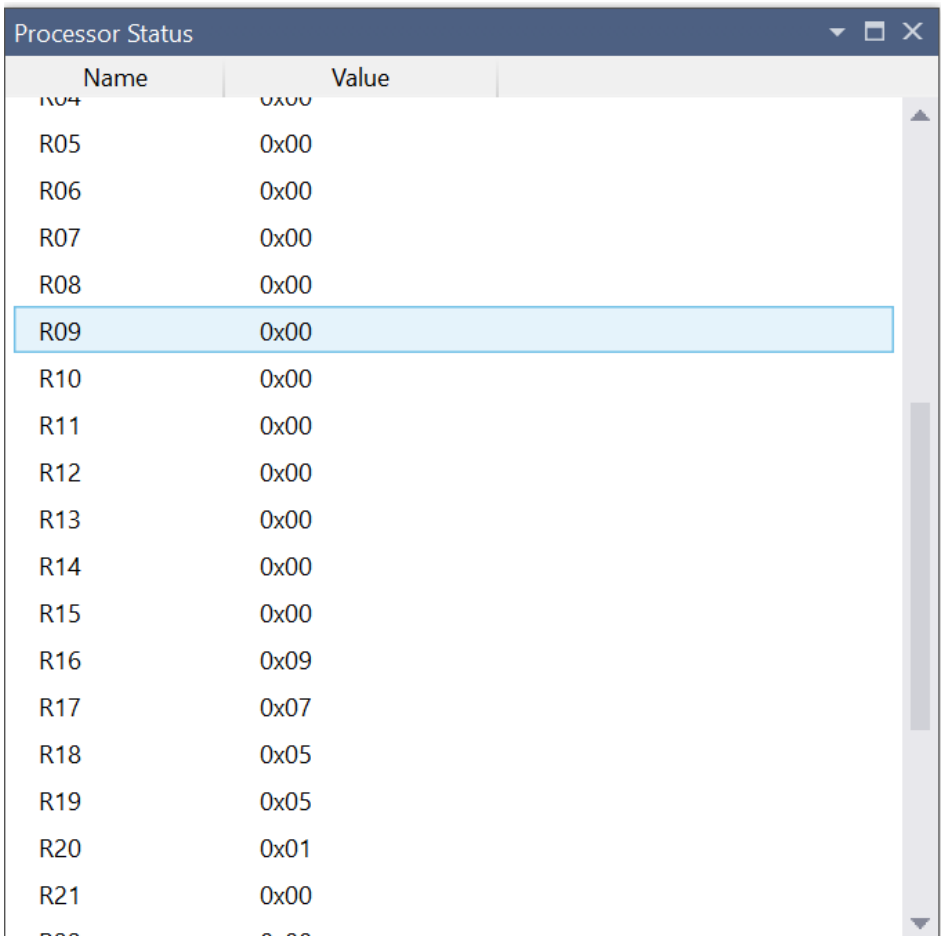


Figure 9: Output in r16

Output is 9, given in r16