EE23B022RprtLab2

Deepak Charan S ee23b022

August 2024

1 Introduction:

I am Deepak Charan S (Roll No: EE23B022) and this is my report for the second assignment of Microprocessor Lab, which I had done on 20/8/24.

2 Objective:

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

3 Equipments/Software Required:

A Windows PC with Microchip Studio IDE

4 Codes Used:

4.1 8 bit addition:

```
;
; 8bit_adder.asm
;
; Created: 20-08-2024 08:41:12
; Author : ielab1
;
; Program to add two 8 but numbers in memory and store
   the result in a given memory location say with
   address1
```

```
.CSEG; define memory space to hold program - code segment
LDI ZL,LOW(NUM<<1); load byte addrss of LSB of word addrss
LDI ZH,HIGH(NUM<<1);load byte addrss of MSB of word addrss
LDI XL,0x60; load SRAM LSB of 16-bit address in X-register
LDI XH,0x00; above's MSB|word address can be line number here
LDI R16,00; clear R16, used to hold carry
LPM R0,Z+;Z now follows byte addrssng. points MSB of NUM addr
LPM R1,Z; Get second number (LSB of NUM addr) 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 given address in SRAM ie 0x60
        ST X,R16 ; store carry in next location 0x61
        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 an
; additional zero byte will be inserted by the
; assembler.)
```

4.2 16 bit addition using 8 bit registers:

```
;
; 16bit_adder.asm
;
; Created: 20-08-2024 08:41:12
; Author : ielab1
;
; Program to add two 16 bit numbers in memory and store the result in a given memory location say with address1
.CSEG; define memory space to hold program - code segment
```

```
LDI ZL,LOW(NUM<<1); load byte addrss of LSB of word addrss
LDI ZH,HIGH(NUM<<1);load byte addrss of MSB of word addrss
LDI XL,0x60; load SRAM LSB of 16-bit address in X-register
LDI XH,0x00; above's MSB|word address can be line number here
LDI R16,00; clear R16, used to hold carry
LPM R0,Z+; loads first byte of first number to R0
LPM R1,Z+; loads second byte of first number to R1
LPM R2,Z+; loads first byte of second number to R2
LPM R3, Z+; loads second byte of second number to R3
ADD R0,R2; Add R0 and R2,result in R0,carry flag affected
ADC R1,R3; Adding the second bytes of the numbers
        BRCC abc; jump if no carry,
        LDI R16,0x01 ; else make carry 1
abc: ST X+,R0 ; store first byte of answer in R0
        ST X+,R1 ; store second byte of answer in R1
        ST X,R16; store carry in R16
        NOP ; End of program, No operation
NUM: .db 0x11,0x11,0xFF,0xFF; 16 bit numbers 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 an
; additional zero byte will be inserted by the
; assembler.)
```

4.3 Multiplication of two 8 bit numbers:

```
; mul.asm
;
; Created: 20-08-2024 08:41:12
; Author : ielab1
;
; Program to multiply two numbers in memory and store the result in a given memory location say with address1
.CSEG; define memory space to hold program - code segment
LDI ZL,LOW(NUM<<1); load byte addrss of LSB of word addrss
```

```
LDI ZH,HIGH(NUM<<1);load byte addrss of MSB of word addrss
LDI XL,0x60; load SRAM LSB of 16-bit address in X-register
LDI XH,0x00; above's MSB|word address can be line number here
LPM R0,Z+;Z now follows byte addrssng. points MSB of NUM addr
LPM R1,Z; Get second number (LSB of NUM addr) into R1,
MUL R0,R1; Multiply R0 and R1,result in R0,carry flag affected
ST X+,R0 ; store 8 LSBs of result in given address in SRAM ie 0x60
ST X,R1 ; store 8 MSBs in next location 0x61
NOP ; End of program, No operation
NUM: .db 0x33,0x66; bytes to be multiplied
; .db define data byte directive, inserts one or more
; constant bytes in the code segment (The number of
; inserted bytes must be even, otherwise an
; additional zero byte will be inserted by the
; assembler.)
```

4.4 Comparing two numbers:

```
;
; compare.asm
;
; Created: 20-08-2024 08:41:12
; Author : ielab1
;
; Program to compare the largest number in a finite set of numbers
.CSEG ; define memory space to hold program - code
 LDI ZL , LOW ( NUM <<1) ; load byte addrss of LSB of word
 LDI ZH , HIGH ( NUM <<1) ; load byte addrss of MSB of word
 LDI XL ,0x60 ; load SRAM LSB of 16 - bit address in X
 LDI XH ,0x00 ; above MSB word address can be line number
 LDI R16 ,00 ; clear R16 , used to hold answer
 LPM R0 , Z+ ; Z now follows byte addrssng. points MSB of NUM
```

```
LPM R1 , Z+ ; Get second number ( LSB of NUM addr ) into
    R1 ,
LPM R2, Z+;
LPM R3, Z;
 CP R0 , R1 ; Compare R0 and R1 , result in carry flag
BRCC abc ; jump if no carry ,
 MOV R0,R1 ; else make register holding answer
abc: CP R0,R2;
 BRCC bcd ; jump if no carry ,
 MOV R0,R2 ; else make register holding answer
 BRCC bcd ; jump if no carry ,
 MOV R0,R2 ; else make register holding answer
bcd: CP R0,R3
BRCC cde;
MOV R0,R3;
cde: ST X + , R0 ; store result in given address in SRAM
    ie
 ST X , R0 ; store answer in next location 0 x61
 NOP ; End of program , No operation

NUM : .db 0x03 ,0x5F, 0x6F,0xFF
; .db define data byte directive, inserts one or more
; constant bytes in the code segment (The number of
; inserted bytes must be even, otherwise an
; additional zero byte will be inserted by the
; assembler.)
```

5 Procedure:

1. Write the AVR assembly code on microchip studio IDE to accomplish all the four objectives
2. Clean existing solutions and build new solution at the start of debugging
3. Debug and test them out line by line on microchip studio and analyse the values in registers, SRAM and carry flags
4. Verify the result with manually calculated answer

Some flowcharts are provided below for better visualisation of control flow

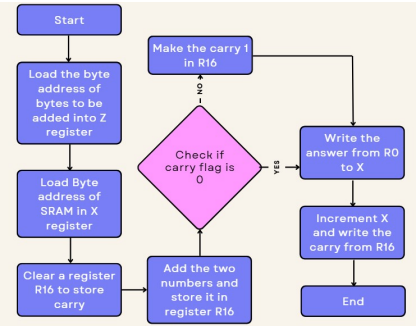

Figure 1: 8 bit addition

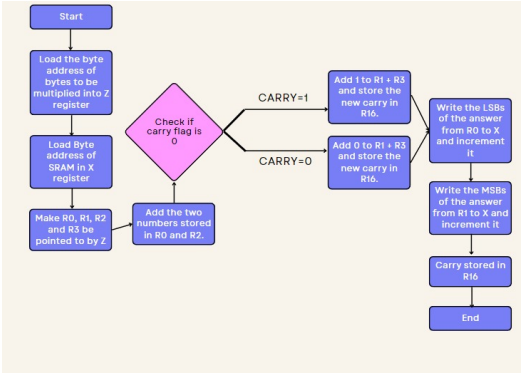

Figure 2: 16 bit addition

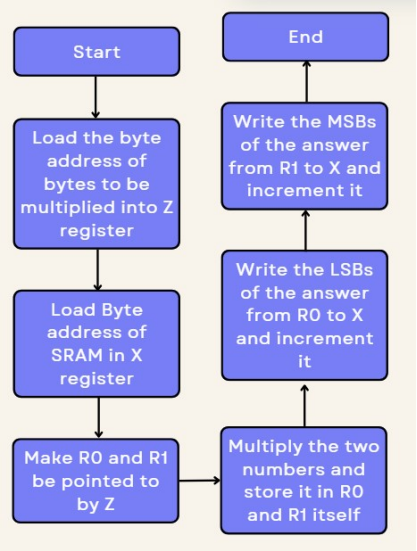

Figure 3: Multiplier

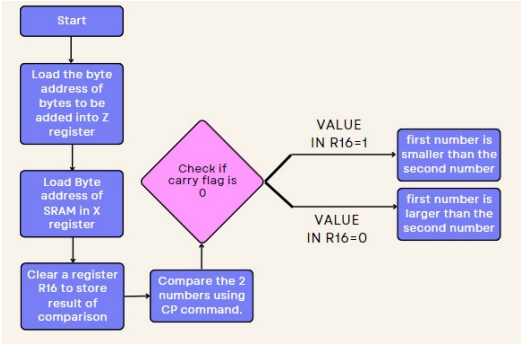

Figure 4: Comparison

6 Interpretations of result

Addition, substraction, multiplication, and comparison of 8-bit / 16-bit binary numbers are demonstrated through emulation of Atmega8 assembly programming in microchip studio IDE.

Note: The code for 8 bit adder and comparision of two 8 bit numbers was done by me.

7 References:

Video Recordings, Handouts and Instruction Manual of AVR provided in moodle. Slight reference from Mazidi's "The AVR Microcontroller and embedded systems" was also used.