EE2016 Micropprocessor Lab & Theory

EE Department, IIT, Madras.

Experiment 2: Arithmetic Computations using Atmel Atmega8 AVR through Assembly Language Emulation

1 Aim

To implement basic arithmetic and logical manipulatin programs using Atmel Atmega8 microcontroller in assemly language simulation, including addition (subtraction), multiplication (division) of 8-bit data.

2 Equipments, Hardware Required

A Windows PC with Atmega AVR Simulator V.6.2

3 Basics

The basic concepts of microcontroller architecture, general purpose programming with the corresponding instruction sets should be very clear. Since the real-life/ hardware assembly language programming of Atmega microproccessor would be carried out in the next week's lab (Experiment 2), the hardware description is relegated to later classes. Here, in Experiment 1, our focus is only on the *emulation* of assembly programming of Atmega8 microcontroller.

3.1 Approaches Available

In this lab session, you are asked to add given two 8 bit numbers and save the result in a register. There are two ways of doing this: (1) real-life implimentation on the atmega processor using machine language or assembly language and (2) emulate the execution of assembly language program on a PC loaded with AVR simulator.

The first choice (1) would require (A) Atmega8 microcontroller along with other accessaries and (B) (Integrated Development Environment) IDE. This IDE primarily allows one to develop a project involving Atmega microcontroller and provides a user friendly environment of edit - run - debug - edit cycles till acceptable performance is achieved.

In practice, the IDE is used only at the development and testing phase. Once the system is tested to satisfaction, the microcontroller loaded with the

developed code (in its memory - SRAM/ SDRAM) is ready for deployment. In the case of Atmel's Atmega8 microcontroller, the IDE is the Atmel AVR Studio software. By and large the IDE is used for developmental purpose in engineering practice while it is a natural choice for educational purposes.

Thus the second choice (2) of emulation requires only a Windows PC with the Atmel Studio (for AVR class of processors) simulator software loaded.

3.2 Emulation of Atmel AVR Assembly Programming

In this experiment, we adopt the 2nd approach, i.e., emulation of Atmega8 microcontroller (leaving the real-life atmega8 microcontroller programming to the next week).

3.3 Basic Concepts and Information Required

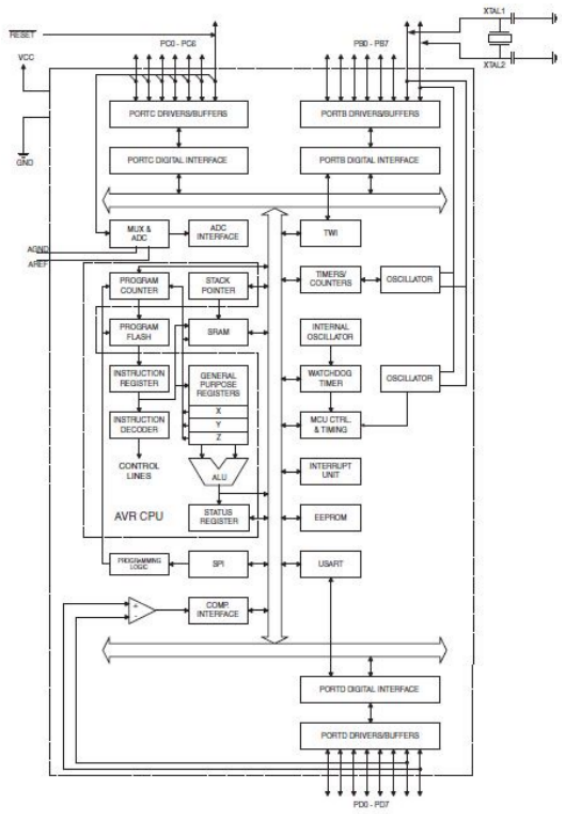
Following information is required (Refer the corresponding documents indicated in paranthesis)

1. Atmel AVR instruction set (Refer Atmel 8-bit AVR Instruction Set uploaded in moodle)
2. Atmel AVR assembly programming limited to
   (a) addition
   (b) subtraction
   (c) multiplication
   (d) division.
3. Atmega8 microcontroller architecture (of ALU, registers and instruction cycles)
4. Atmel AVR Studio 7 simulator (Refer Atmel studio user guide uploaded in moodle)

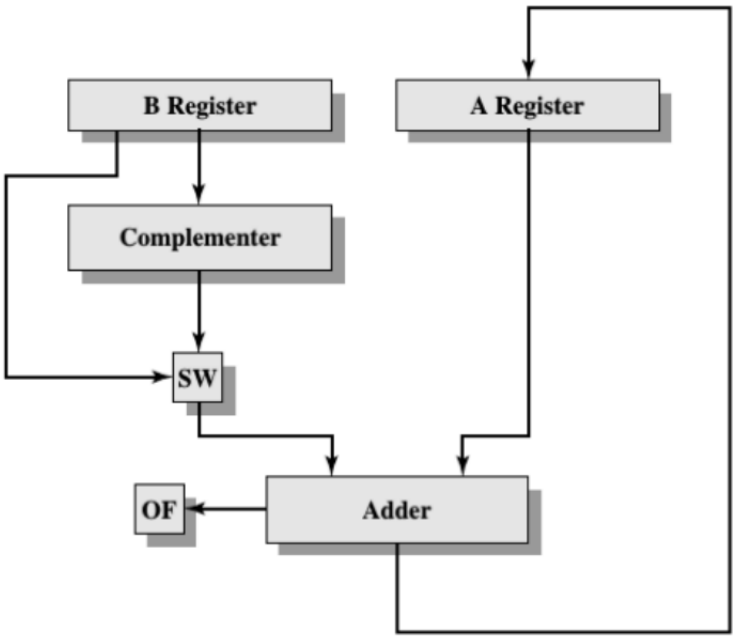
All of the above relevant documents are uploaded in moodle.

3.4 Dedicated Hardware for ALU Operations

A schematic of the hardware is given below:

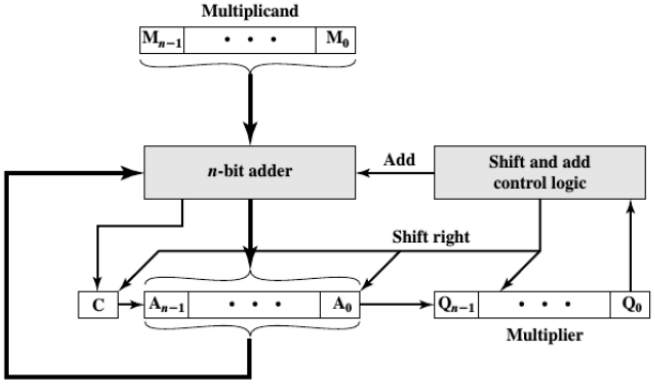


The more general ways of implementations of addition and subtraction is given below

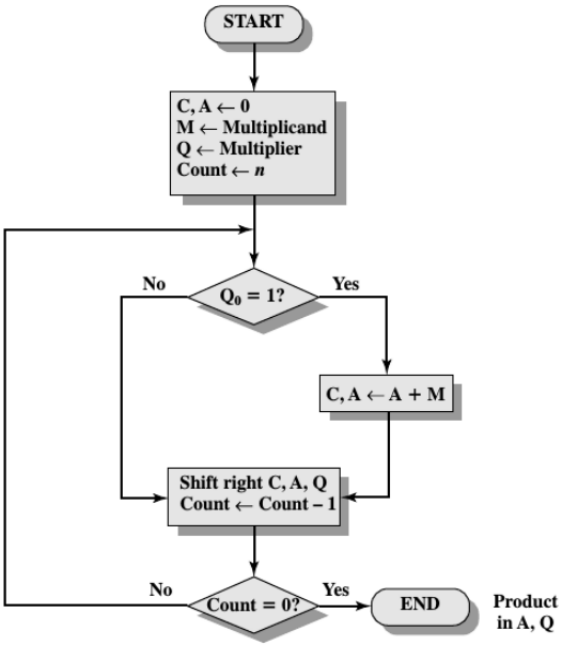


OF = Overflow bit
SW = Switch (select addition or subtraction)

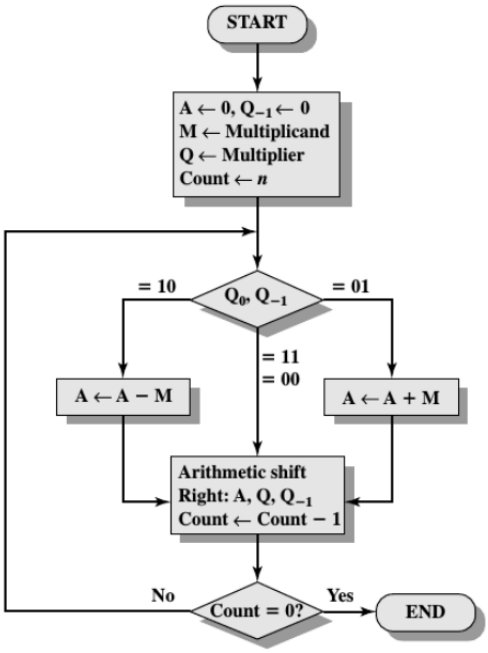
Normally 2's complement representation is used for all mathematical operations.



Flow chart for



Booth's Algorithm for Multiplication



3.5 Useful AVR Instructions for Computations

The following table provides a brief overview of some of the instructions that might be useful for this experiment.

Mnemonics	Operands	Description	Operation
ADD	Rd, Rr	Add without Carry	Rd ←Rd + Rr
ADC	Rd, Rr	Add with Carry	Rd ←Rd + Rr + C
SUB	Rd, Rr	Subtract without Carry	Rd ←Rd - Rr
SBC	Rd, Rr	Subtract with Carry	Rd ←Rd - Rr - C
AND	Rd, Rr	Logical AND	Rd ←Rd •Rr
OR	Rd, Rr	Logical OR	Rd ←Rd ∨ Rr
INC	Rd	Increment	Rd ←Rd + 1
DEC	Rd	Decrement	Rd ←Rd - 1
MUL	Rd, Rr	Multiply Unsigned	R1, R0 ←Rd * Rr
CP	Rd, Rr	Compare	Rd - Rr
BREQ	k	Branch if Equal	If (Z = 1) then PC ←PC + k + 1
MOV	Rd, Rr	Copy Register	Rd ←Rr
LDI	Rd, K	Load Immediate	Rd ←K
ST	X, Rr	Store Indirect	(X) ←Rr
LSL	Rd	Logical Shift Left	Rd(n+1) ←Rd(n), Rd(0) ←0, C ←Rd(7)

The above can be categorized as arithmetic and logic instructions, branch instructions, data transfer instructions, and bit and bit-test instructions (Identify the category under which each instruction falls). Please note that there are several variants of the above instructions, notably the "immediate","with carry","indirect", and the "if X-condition" variants for branching. To learn more about these, view the AVR Assembler User Guide on Moodle.

3.5.1 Demo Program

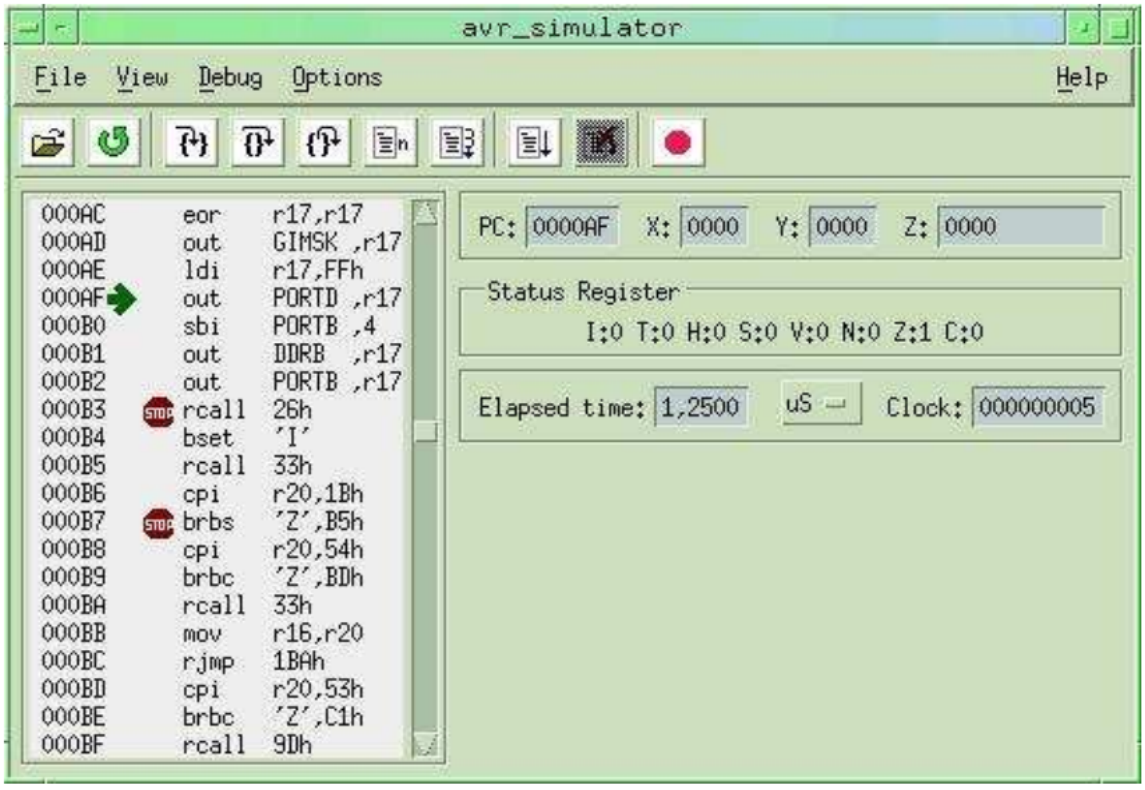
The following is the demo program which does the addition of two nos in Atmel Atmega8 processor. With this the student is expected to write programs which does subtraction, multiplication, finding

```
/*
* add.asm
*
* Created: 8/3/2018 11:43:15 AM
* Author: Students
* Program to add two numbers in memory and store the result in memory
*/
.CSEG ; code segment - define memory space to hold program
LDI ZL,LOW(NUM<<1) ; load address of first number
LDI ZH,HIGH(NUM<<1) ;
LDI XL,0x60 ; load SRAM address in X register
LDI XH,0x00
LDI R16,00 ; clear R16,used to hold carry
LPM R0,Z+
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,
```

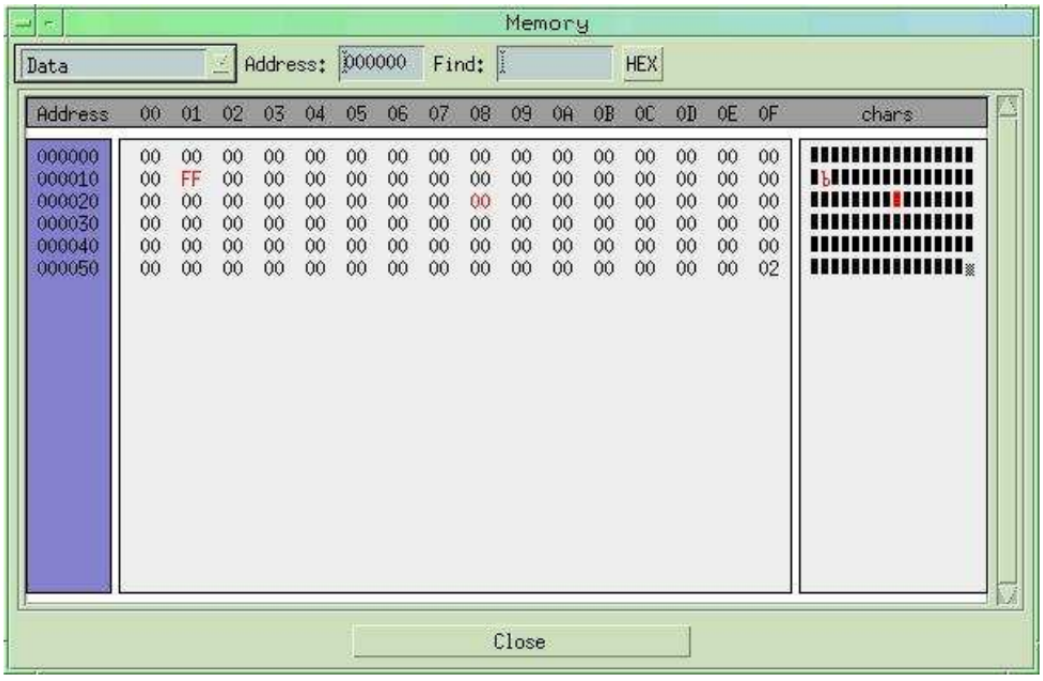
LDI R16,0x01 ; else make carry 1
abc: ST X+,R0 ; store result in RAM
ST X,R16 ; store carry in next location
NOP ; End of program , No operation
NUM: .db 0xD3,0x5F; The two numbers to be added , result to be put in SRAM starting at 0x60

4 Running the Experiment in Atmel Studio 6.2

A screen shot of Atmel AVR Simulator: (a) Main Window



(b) Memory Window



(c) Contents of General Registers

5 Experiments

5.1 Problems to Do

Problem 1 (Addition of two 8 bit numbers): Given two 8-bit numbers, compute the sum and difference and display the results.

Hints/ Examples :

SRAM Location	Values	Remarks	Input/ Output
0060	D3	Number 1	**Inputs**
0061	5F	Number 2	
0062	32	LB of Sum	**Outputs**
0063	01	HB of Sum	
0064	74	Difference	
0065	00	Sign (00 for positive, FF for negative)	

Problem 2 (Division of two 8 bit numbers): Given two 8-bit numbers, perform division and store the quotient and reminder and display the results.

Hints/ Examples :

SRAM Location	Values	Remarks	Input/ Output
0060	D3	Numerator	**Inputs**
0061	5F	Denominator	
0062	02	Quotient	**Outputs**
0063	15	Reminder	

Problem 3 (Multiplication of two 8 bit numbers): Given two 8-bit numbers, perform multiplication (without using the instruction MUL) and store and display the results.

Hints/ Examples :

SRAM/ EEPROM/ Flash Location	Values	Remarks	Input/ Output
(EEPROM location)NUM:	D3	Multiplicand	**Inputs**
	5F	Multiplier	
(SRAM location)0060	4D	LB of Multiplication	**Outputs**
(SRAM location)0061	4E	HB of Multiplication	

To load data from EEPROM/ Flash use the instruction LPM, please see the example below:

.....

.....

LDI ZL,LOW(NUM<<1)

LDI ZH,HIGH(NUM<<1) ; Z holds the EEPROM address where the data is stored.

LPM R0,Z+ ; Multiplicand loaded from EEPROM, then Z = Z + 1

LPM R1,Z ; Multiplier is loaded from the next location pointed by Z

.....

....

NUM: .db 0XD3, 0X5F ; The inputs are defined in the EEPROM locations pointed by the label NUM:

Problem 4 (16 bit addition in 8-bit processor): Given two 16-bit binary words, compute the sum and display the results.

Hints/ Examples :

SRAM/ EEPROM/ Flash Location	Values	Remarks	Input/ Output
NUM:	D3	Number 1	**Inputs**
	5F		
	AB	Number 2	
	CD		
0060	7E	Result	**Outputs**
0061	2D		
0062	01		

Problem 5 (Compute odd or even): Given an 8 bit number, check whether it is even or odd (Even result indicate as 00 and odd result indicate as 01).

Hints/ Examples :

SRAM Location	Values	Remarks	Input/ Output
0060	D3	Number	**Input**
0061	01	Odd	**Output**

Problem 4 (Use of Pointers, indirect addressing): Given a set of 8 bit numbers, compute the entire sum and store and display the results. Hints/ Examples :

SRAM/ EEPROM/ Flash Location	Values	Remarks	Input/ Output
(EEPROM location) NUM:	03	Count of numbers	**Inputs**
	5F	Number 1	
	32	Number 2	
	01	Number 3	
(SRAM location) 0060	92	LB of result	**Outputs**
0061	00	HB of result	

5.2 General Procedure

Step 1 Draw the flow chart for the above problem

Step 2 Convert the flow chart into the assembly language program, identifying the corresponding instructions from the Atmega8 instruction set [Till now, get the procedure corrected by the TA].

Step 3 Get it compiled by the AVR compiler.

Step 4 Run the program.

Step 5 Verify the result with the manually calculated sum.

Keep an eye on the values of the register for each cycle [Ask the TA, how to debug using AVR simulator].

6 Results & Inference

Addition, substraction, multiplication, division of two 8-bit binary numbers along with 16 bit addition of two numbers, odd or even number checking and sum of a series of 8 bit numbers are demonstrated through emulation of Atmega8 assembly programming.