EE2016 Microprocessor Lab & Theory. Aug - Nov, 2021

Dept. of Electrical Engineering, IIT, Madras.

Experiment 1: Computations using Atmel Atmega8 AVR through Assembly Language Emulation

1 Aim

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

2 Equipments (Hardware/ Software) Required

A Windows PC with Microchip Studio V.7.0

3 Basic Concepts / Familiarity Required

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 de- scription 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 implementation 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 an 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 Microchip 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 Microchip Studio (for AVR class of processors) simulator software loaded.

3.2 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).

3.3 Basic Concepts and Information Required

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

A) Atmel AVR instruction set (Refer Atmel 8-bit AVR Instruction Set uploaded in Moodle).

A) Atmel AVR assembly programming limited to

  a) binary addition
  a) binary multiplication
  a) compare two binary words

a) storing and loading bytes from memory

(Refer Atmel AVR Assembler User Guide 1998)

A) Atmega8 microcontroller architecture (of ALU, registers and instruction cycles)

A) Microchip AVR Studio 7 simulator (Installation demo done last week; may still refer Microchip Studio guide sent to students)

All of the above relevant documents are uploaded in Moodle.

3.4 Demo Program

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 $\bigvee$ Rr
INC	Rd	Increment	Rd ←Rd + 1
DEC	Rd	Decrement	Rd ←Rd - 1
MUL	Rd, Rr R1,	Multiply Unsigned	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.

3.4.1 .Demo Program

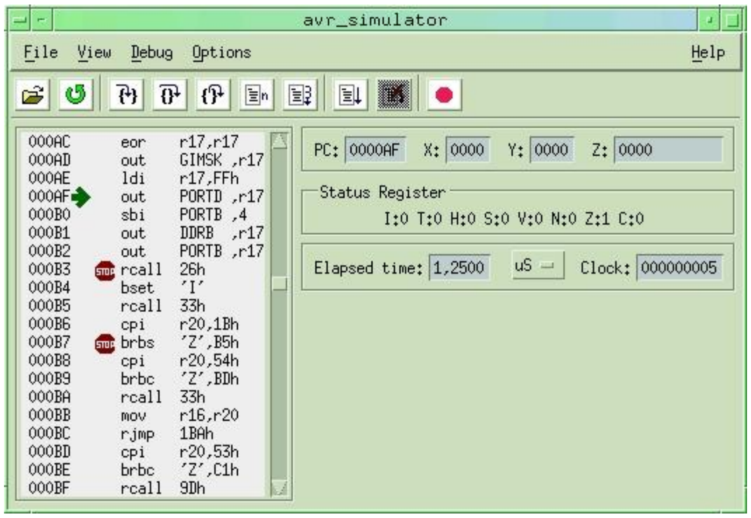
The following is the demo program which performs the addition of two numbers in Atmel Atmega 8 processor. With this the student is expected to write programs which does subtraction and multiplication.

```
/*
* 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 ; defines memory space to hold program
LDI ZL,LOW(NUM<<1);load address of first data byte
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 added, result put in SRAM from 0x60
```

4 Running Experiments in Atmel Studio 6.2

A screen shot of Atmel AVR Studio Simulator (An older Version):

(a) Main Window



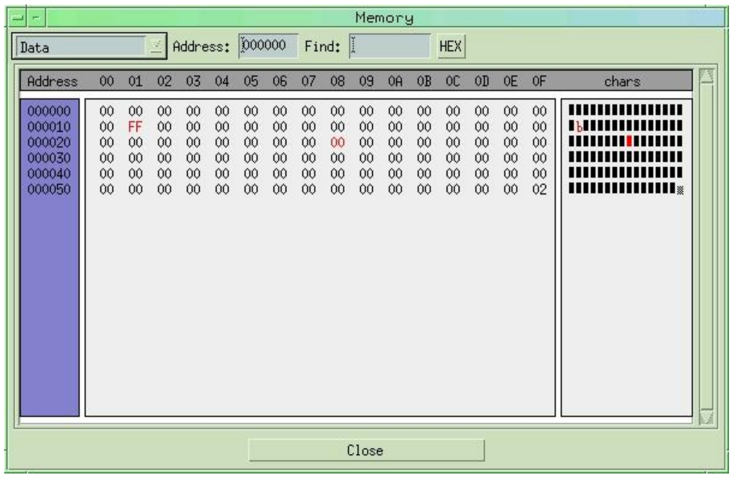
```
000AC     eor    r17,r17
000AD     out    GIMSK ,r17
000AE     ldi    r17,FFh
000AF     out    PORTD ,r17
000B0     sbi    PORTB ,4
000B1     out    DDRB  ,r17
000B2     out    PORTB ,r17
000B3     rcall  26h
000B4     bset   'I'
000B5     rcall  33h
000B6     cpi    r20,1Bh
000B7     brbs   'Z',B5h
000B8     cpi    r20,54h
000B9     brbc   'Z',BDh
000BA     rcall  33h
000BB     mov    r16,r20
000BC     rjmp   1BAh
000BD     cpi    r20,53h
000BE     brbc   'Z',C1h
000BF     rcall  9Dh
```

PC: 0000AF X: 0000 Y: 0000 Z: 0000

Status Register
I:0 T:0 H:0 S:0 V:0 N:0 Z:1 C:0

Elapsed time: 1,2500 uS Clock: 000000005

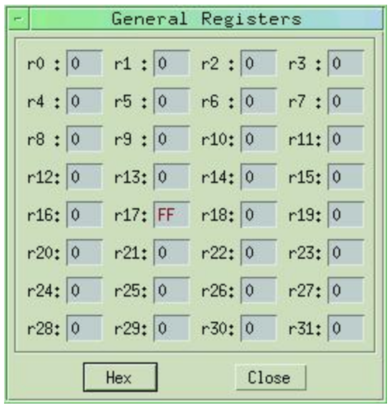
(b) Memory Window



Data Address: 000000 Find: HEX

```
Address  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   chars
000000   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000010   00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000030   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02
```

(c) Content of General Registers



r0 : 0	r1 : 0	r2 : 0	r3 : 0
r4 : 0	r5 : 0	r6 : 0	r7 : 0
r8 : 0	r9 : 0	r10: 0	r11: 0
r12: 0	r13: 0	r14: 0	r15: 0
r16: 0	r17: FF	r18: 0	r19: 0
r20: 0	r21: 0	r22: 0	r23: 0
r24: 0	r25: 0	r26: 0	r27: 0
r28: 0	r29: 0	r30: 0	r31: 0

5 Experiment

5.1 Problems to Do

Problem 1 (Common 8-bit mathematical operation): Given two 8-bit binary words (byte), compute the sum and store it in register(s).
Problem 2 (16-bit addition using a 8-bit processor): Given two 16-bit binary words (byte), compute the sum and store it in register(s).
Problem 3 (Multiplication of two 8-bit numbers): Given two 8-bit binary words (byte), compute the product and store it in register(s).
Problem 4 (Largest of number given): Given a finite set of binary words, load them-from memory, identify the largest in the given set, and store this largest number in SRAM.

5.2 General Procedure

Step 1 Draw the flowchart 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. Keep an eye on the values of the register for each cycle [Ask the TA, how to debug using AVR simulator].

Step 5 Verify the result with the manually calculated answer. If not, debug [Take help of TA, if necessary]. Demonstrate it to TA once your code does what it is supposed to do.

6 Results and Inference

Addition, subtraction, multiplication, and comparison of 8-bit / 16-bit binary numbers are demonstrated through emulation of Atmega8 assembly programming.