Microprocessors Lab

Dr Asimakis Leros

Lab exercise 1

Objective

- Familiarizarion with the Atmel Studio development environment.
- Introduction to the AVR assembly language
- Data transfer instructions (load, store, move)
- Arithmetic operations (ADD, SUB)
- Logical operations (AND, OR, EOR)

Please note:

- You may install the Atmel Studio or Microchip Studio in your PC, if you wish to practise, from `https://www.microchip.com/en-us/tools-resources/develop/microchip-studio`.
- The Atmel Studio uses a structure of projects and solutions for storing and maitaining your code. Each project consists of source files, libraries etc related to the same software development project. A solution consists of different projects which use common tools and regard the same microcontroller. My advice is to have a project per lab exercise (even if it contains a single source file) and a solution folder to include everything.
- We will not be using the Arduino board in this exercise.
- In your lab report you must provide answers to all of the questions.

1 Addition of two variables

We start with a program that does nothing:

```
start:
rjmp start
```

The following code adds variables `var1` and `var2` and stores the result in `result`.

```
.dseg
.ORG 0x0100

var1: .byte 1
var2: .byte 1
result: .byte 1

.cseg
.ORG 0x0000

start:

lds r16, var1
lds r17, var2
add r17, r16
sts result, r17

rjmp start
```

(1) Use the memory window of Atmel Studio to place in memory the values `0x48` and `0x07`. Run your program step-by-step and note (at each step) the values of the two registers, the `result` variable in memory, and the SREG `(flags)`.

(2) Repeat with the following pairs of values (all values in hex). In each case write down the result in hex, signed and unsigned decimal. Also write down the values of the S, V, N, Z, and C flags. Explain how these values are obtained and what the various flags signify (1 or 2 lines per case).

var1	var2	hex	signed	unsigned	S V N Z C
48	07				
48	48				
72	81				
72	60				
72	FF				
01	FF				

(3) What is the total data size in bytes? In which memory address is each variable located?

(4) What is the code size in bytes and in words? What are the memory addresses of the first and the last instructions?

(5) Write down the machine language equivalent of the assembly instruction `add r17, r16` (use the memory window of the Atmel Studio) in hex and binary. Using the assembly language manual, find out what each individual bit signifies.

(6) Do the same for the instruction `sts result, r17` (two words).

2 Subtraction of two variables

Change the addition instruction of the previous example into subtraction and run your code for the following pairs of values. Again, explain what the result and flags signify.

var1	var2	hex	signed	unsigned	S V N Z C
48	29				
48	48				
48	58				
00	01				
72	FF				

3 Increment and decrement of a counter

(1) What happens when a counter overflows, i.e. it counts past its range? Write down the values of the flags and the `r16` register after the execution of each `INC` instruction in the following code.

```
ldi r16, 254  ; hex 0xFE
inc r16
inc r16
```

(2) Write down the values of each flag and the `r16` register after the execution of each `DEC` instruction in the following code.

```
ldi r16, 1
dec r16
dec r16
```

4 Logical operations

(1) The `AND` instruction can be used, among other things, to clear (set to zero) some bits of a variable. The following code segment clears the four most significant bits of `r16`, while keeping the four least significant bits unchanged. Run the code for `r16 = 34h` and `r16 = 30h`. Note down and explain the result (`r16` and individual flags).

```
ldi r16, 0x34
ldi r17, 0x0F
and r16, r17
```

(2) The `OR` instruction can set to 1 selected bits of a register variable. The following code segment sets to 1 the four least significant bits of `r16` and leaves the rest bits unchanged. Run the code and explain the values of the result and flags.

```
ldi r16, 0x34
ldi r17, 0x0F
or r16, r17
```

(3) The EOR (exclusive OR) instruction can be used to invert selected bits. Run the following code segment and explain the values of the result and flags after each EOR.

```
ldi r16, 0x55
ldi r17, 0xFF
eor r16, r17
eor r16, r16
```

5 Program flow instructions

The main program flow instructions in the AVR microprocessors are:

- Jump, or unconditional branch, instructions: JMP and RJMP. They correspond to a goto.
- Conditional branch instructions. They have the form BRcc, where the last two characters correspond to a condition, e.g. BREQ (branch if equal). These instructions are equivalent to an if then goto.

The following code segment checks whether a variable var1 is positive or zero; if not, the variable is replaced by its opposite.

```
lds r16, var1
ldi r17, 0x00
cp r16, r17
brpl positive
sub r17, r16
rjmp finish
positive:
mov r17, r16
finish:
```

(1) Explain briefly what each line of code does.

(2) Run the code step-by-step for a positive and negative value of the variable and note the results.