Lab 0

Familiarization with The Lab

EEL 4746C / EEL 5756C: Microcomputers

Fall 2018

Student Name:	
Student ID:	
Section:	

Contents

1 Objective

The objective of this experiment is to familiarize students with the software tools that they will use during this semester.

2 Introduction

2.1 Definition

- **Assembly language:** A human readable form of the machine language. Every instruction in Assembly is a one-to-one map to the machine instructions.
- **Machine language:** It is a set of 0s and 1s vectors that tells the machine what exactly it needs to perform.
- **Assembler:** A program that translates assembly language into machine language.

2.2 AVR *µ*C

The AVR is a microcontroller based on a modified 8-bit RISC Harvard architecture processor. The µC comes with on chip integrated Flash, EEPROM, and SRAM. The Flash is used to store the programs. The EEPROM is used to store date in a non-volatile memory. And the SRAM is the running memory for the microcontroller. Most of the AVR instructions take either one or two clock cycles.

2.3 GNU toolchain for AVR

GNU toolchain for AVR contains many tools. Here are the most popular set of tools:

- **`avr-as`:** Assembler
- **`avr-gcc`:** C Compiler
- **`avr-gdb`:** debugger
- **`avr-ld`:** Linker
- **`avr-nm`:** display symbol list
- **`avr-objcopy`:** translate object files into different formats
- **`avr-objdump`:** display object information
- **`avr-size`:** display section sizes and total size.

2.3.1 avr-as

Usage:

```
avr-as -mmcu=atmega328p –o [object_file.o] [source_file.asm]
```

Options:

mmcu: Microcontroller unit selection
o: name of the object file

2.3.2 avr-ld

Usage:

```
avr-ld -o [executable_file.elf] [object_file.o]
```

Options:

o: name of the executable file

2.3.3 avr-gdb

2.3.3.1 avr-gdb | target remote

Usage:

```
(gdb) target remote [machine_ip/name]:[port]
```

Connect to remote GDB server using ip and port.

2.3.3.2 avr-gdb | target sim

Usage:

```
(gdb) target sim
```

Use GDB internal simulator for AVR.

2.3.3.3 avr-gdb | file

Usage:

```
(gdb) file [filename]
```

Use filename as the file to debug.

2.3.3.4 avr-gdb | load

Usage:

```
(gdb) file [filename]
```

Use filename as the file to debug.

2.3.3.5 avr-gdb | disassemble

Usage:

```
(gdb) disassemble [starting_address], [ending_address]
```

Display the assembly instructions for the machine code within the memory

2.3.3.6 avr-gdb | info registers

Usage:

```
(gdb) info registers
```

Display the content of the registers

2.3.3.7 avr-gdb | continue

Usage:
```
(gdb) continue
```

Continue running

2.3.3.8 avr-gdb | continue

Usage:
```
(gdb) continue
```

Continue running

2.4 simulavr

Usage:
```
simulavr -d atmega328 -g -F 20000000
```

Options:

d: device name
g: listen to gdb connection
F: Set the CPU frequency in Hz

3 Procedure

1) Write the following code in an editor and save it to as lab1.asm

```
.global start
.set number1, 0x01AA02FE
.text
start:
                        ldi r20, 0
                        ldi r16, lo8(number1)
                        ldi r17, hi8(number1)
                        ldi r18, hh8(number1)
                        ldi r19, hhi8(number1)
                        add r16, r17
                        brcc  j01
                        breq  j02
                        brpl  j03
                        brge  j04
j01:
                        ldi r20, 1
                        rjmp  infiniteloop
j02:
                        ori r20, 2
                        rjmp  infiniteloop
j03:
                        ori r20, 4
                        rjmp  infiniteloop
```

```
j04:
                        ori r20, 8
infiniteloop:
                        sts result1, r20
                        rjmp  infiniteloop
.data
.org  0x00A0
result1:
                        .skip 4, 0xaa
result2:
                        .skip 4, 0x55
```

2) Assemble the code above.
3) Link and generate the executable.
4) Run `avr-gdb` and load the code to the emulator.
5) Disassemble the code.
6) Run the code.