

Introduction of Computer Organization

Kustiawanto Halim

Even 2016/2017 - Universitas Indonesia

Abstract. AVR is a family of microcontrollers developed by Atmel beginning in 1996. These are modified Harvard architecture 8-bit RISC single-chip microcontrollers. AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time. AVR microcontrollers find many applications as embedded systems; they are also used in the Arduino line of open source board designs.

Table of Contents

Abstract	1
Introduction	1
1 Getting Started	2
2 Introduction to Register	4
3 Simple Program	5
4 Exercise	7
4.1 First Problem - Discovery	7
4.2 Second Problem* - Creative	7
4.3 Third Problem* - Implementation	8

Introduction

In this tutorial, you will be introduced to AVR Studio 4 and ATmega8515. The objectives of this tutorial is to explain how to use AVR Studio 4, and introducing various features of the simulator such as register, data memory, program memory and so on. This tutorial also provide brief explanation about assembly programming language.

1 Getting Started

These are essential step needed to create a new project in AVR Studio.

1. Open AVR Studio in your computer.
2. Choose New Project.

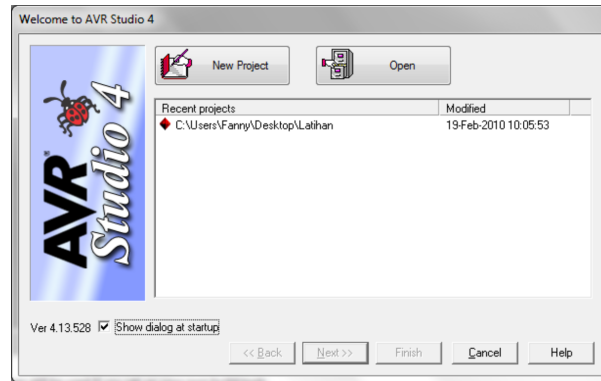


Fig. 1. Dialog window showing New Project or Open menu of AVR Studio 4

3. In Project Type window, choose Atmel AVR Assembler. Write your project name, and choose location of your project. Then click next.

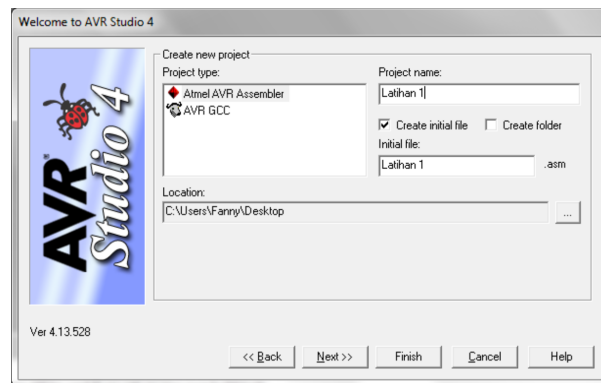


Fig. 2. Dialog window showing Project Type, make sure to choose Atmel AVR Assembler and write your Project Name

4. In Debug Platform window, choose AVR Simulator. In Device Tab, choose ATmega8515. Then click finish.

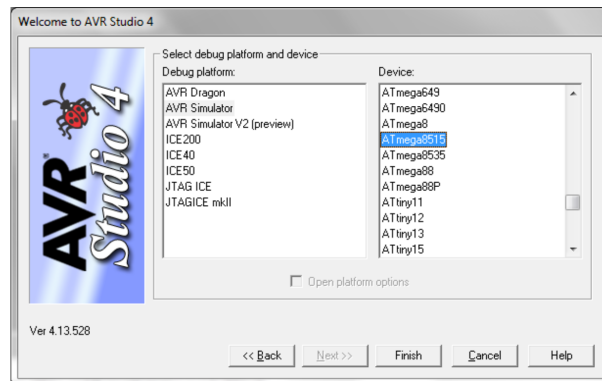


Fig. 3. Dialog window showing Project Type, make sure to choose Atmel AVR Assembler and write your Project Name

5. Make sure **Processor**, **Register**, **Program Memory** and **Data Memory** window is opened. You can open the window tab by accessing **View » Toolbars » Processor**.

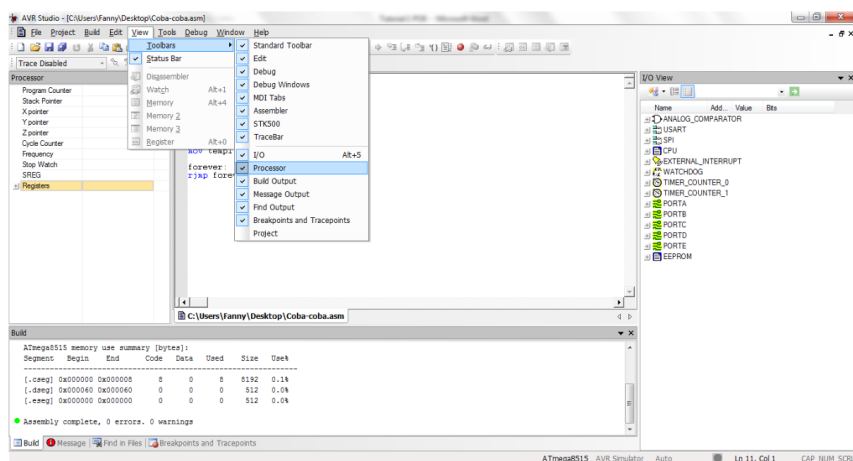


Fig. 4. Menu View in AVR Studio 4

6. For **Register**, **Program Memory**, **Data Memory** can be opened after doing Debugging process.
7. Now you have successfully created a new AVR Studio Project.

2 Introduction to Register

Registers are special storage capacity of 8 bits, that is located in the *datapath* in Processor. A register can store values from 0 to 255, or a value of -128 to +127 (using the 7th bit as the sign bit), or a value that represents the ASCII characters. In addition, registers are also useful as a flag, which each of eight bits in register does not represent a value, but we treat them as a single bit indicating yes/no decisions. There are 32 General Purpose Registers in AVR, they are named R0-R31. This register can be named as you wish (like a variable).

The very first step in writing your assembly language in AVR is to make sure we include the correct header of device that we use. In this case we use ATmega8515.

```
.include "m8515def.inc"
```

To rename a register, you can simply write:

```
.def registerku = r16
```

Now, look at the following instruction:

```
ldi registerku, $aa
```

The instruction above will load a value of **\$aa** to register **R16** (Load Immediate). This instruction will take a constant value and put it in desire register. An instruction can also involve two register at the same time. One of the instruction is **mov**. This instruction will copy a value of a register to another register.

```
.def registerku = r16
.def registerkamu = r15

ldi registerku, $aa
mov registerkamu, registerku
```

Take a look at the instruction above. **ldi** (load a constant value to register) operation can only use R16-R31 as destination register (You can't use R0-R15 for **ldi** operation). For further explanation about register and how to use it, you can check ICO slide in SCeLE.

3 Simple Program

This is a simple AVR program using simple operation.



```
.include "m8515def.inc"
.def temp = r16
.def temp1 = r15

.equ param = 4      ; constant initialization

ldi temp, $0A
subi temp, param    ; subtraction operation
mov temp1, temp

forever:
rjmp forever       : infinite loop
```

To run the following code of program, we need to Build/Assemble your program. Below is several step to Build/Assemble your program:

1. Choose Menu Build » Build (F7) or click on shortcut icon 
2. If the program can be build successfully (no error found), proceed by choosing Menu Build » Build and Run or using the following shortcut icon 
3. Click View » Memory View » Memory2, View » Memory3, and View » Register.
4. Now you can trace the code by running the program step by step. To run the program step by step you can hit F11.

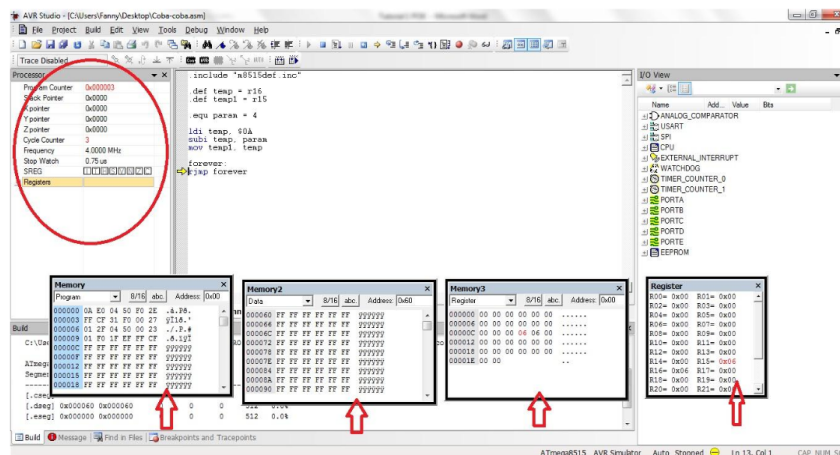



Fig. 5. Window that you need to examine

5. While the program is running, the yellow arrow on the code window will guide you to show which line of code being executed at the time. Also, the

Register, **Processor**, and **Memory** window will be changed based on the code.

6. Watch and examine the value of **Register**, **Processor**, and **Memory** window!

Several notes:

1. You can stop the process by pressing stop button 
2. Try to change the value given in the code and examine the different value in **Register**, **Processor**, and **Memory**

4 Exercise

4.1 First Problem - Discovery

Follow several steps given bellow:

1. Open `avr102.asm` and execute the program.
2. Watch and examine the value of Register, Program Memory, and Data Memory!
3. Do the same step (2) for the following code:

```
.include "m8515def.inc"

.def result = R17
.def final = R1

main :
    ldi ZL, LOW(2*SESUATU)
    ldi ZH, HIGH(2*SESUATU)

loop :
    lpm
    adiw ZL,1
    tst R0
    breq stop
    add result, R0
    rjmp loop
stop :
    mov final, result
forever :
    rjmp forever

SESUATU:
.db 1,2,3,4,5,6
.db 0
```

Attach `README.txt` and give brief explanation about the program, what happen when we execute each of every program above.

4.2 Second Problem* - Creative

Try to make a program at least using 4 of the following instruction:

LD	LDI	ST
MOV	ADD	MUL
SUB	SUBI	CLR
AND	ANDI	LSR
LSL	DEC	INC
EOR	OR	ORI

The result of the operation must be stored in R0.

You can use your creativity to create the program. Please attach `README.txt` in your submission, explain `input`, `variable` and `goal` of your program.

4.3 Third Problem* - Implementation

If we have information about diameter of a circle in R2, and `pi` is 3. Calculate:

- `Perimeter` of the circle, and stored the result in R3.
- `Area` of the circle, and stored the result in R4.

Hint: you can use `ldi`, `mov`, `mul`, `add` instruction!

* Use ATmega8515 as the device.

Submit your assignment in [Name]-[NPM]-Tutorial1.zip format :

- Folder Discovery consist of:
 - `README.txt`
- Folder Creative consist of:
 - `mycode.asm`
 - `README.txt`
- Folder Implementation consist of:
 - `circle.asm`

Good Luck!