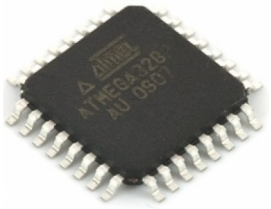
PROGRAM are instructions from Programmer(Human) to the micro-controller CPU, asking the CPU to do task.



The micro-controller CPU can only do task according to what it is capable of doing.

The manufacturer of the micro-controller will prepare a list of task that the CPU can perform, the list is known as the

"INSTRUCTION SET"

The "INSTRUCTION SET" is consist of many individual "INSTRUCTION" where each "INSTRUCTION" tell the CPU to perform a simple task

The ATMEGA328 micro-controller is using the AVR INSTRUCTION SET, The picture below (from ATMEGA328 datasheet) shows some of the "INSTRUCTION" from the "AVR INSTRUCTION SET", "ADD","ADC","ADIW",...

31. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add two registers	Rd ← Rd + Rr	Z,C,N,VH	1
ADC	Rd, Rr	Add with carry two registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl, K	Add immediate to word	Rdh: Rdl ← Rdh: Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two registers	Rd ← Rd – Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract constant from register	Rd ← Rd – K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with carry two registers	Rd ← Rd – Rr – C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with carry constant from reg.	Rd ← Rd – K – C	Z,C,N,V,H	1
SBIW	Rdl, K	Subtract immediate from word	Rdh: Rdl ← Rdh: Rdl – K	Z,C,N,V,S	2

There are other types of "INSTRUCTION SET" used by different micro-controllers, some of the popular ones are, ARM, x86, RISC-V,...

Sending a single "INSTRUCTION" from the "INSTRUCTION SET" to the micro-controller CPU will not be able to do much for us

Normally we will need to **combine many individual "INSTRUCTION" from the "INSTRUCTION SET"** before the CPU can actually perform a meaningful task for us.

This combined INSTRUCTIONS is known as a **"PROGRAM"**

When we are using the Arduino Platform to write our Program for the ATMEGA328 micro-controller. We can make our PROGRAM in three different Programming styles by using the Arduino IDE Software

1. AVR Assembly Language
2. C-Language with AVR Library
3. C-Language with Arduino Library

We will make two small PROGRAM to demonstrate all the three different Programming styles

1. A simple program to “Turn ON” the Arduino Uno Board test LED
2. A simple program to “Blink” the Arduino Uno Board test LED

Source Codes are available for download.

https://github.com/teaksoon/lmaewapm/blob/main/2021_11_20_intro_program_source.zip

Download and unzip the source .zip file and use the Arduino IDE Software to open the Program in the following folders.

1. A simple program to “Turn ON” the Arduino Uno Board test LED
intro_asm – Coding in Assembly Language
intro_avr – Coding in “C-Language with AVR Library”
intro_arduino – Coding in “C-Language with Arduino Library”

2. A simple program to “Blink” the Arduino Uno Board test LED
intro_blink_asm – Coding in Assembly Language
intro_blink_avr – Coding in “C-Language with AVR Library”
intro_blink_arduino – Coding in “C-Language with Arduino Library”

These codes are just to demonstrate the different programming styles that we can do with the Arduino IDE Software for our ATMEGA328 micro-controller

Example 1:
A simple program to “Turn ON” the Arduino Uno Board test LED

AVR Assembly Language (program name: intro_asm)

```
#define __SFR_OFFSET 0
#include "avr/io.h"
.global main
main:
  SBI  DDRB,5   ; set Arduino Pin 13 Output Pin
  SBI  PORTB,5  ; set High to Arduino Pin 13
  CALL forever
  RET
forever:
  RJMP forever
```

The most obvious thing we see in this program is the INSTRUCTIONS from the AVR INSTRUCTION SET. “SBI”,”CALL”,”RET”,”RJMP” and CPU Memory DDRB and PORTB

Note:
to turn off LED, change “SBI PORTB,5” to “CBI PORTB,5”

C-Language with AVR Library (program name: intro_avr)

```
#include "avr/io.h"
int main() {
  DDRB  |= (1<<DDB5); // set Arduino Pin 13 Output Pin
  PORTB |= (1<<PB5);  // set High to Arduino Pin 13
  forever();
  return 0;
}

void forever() {
  while(1){};
}
```

We do not see INSTRUCTION SET codes here anymore. They are hidden from us by the C-Language. However, in this program we are still manipulating the CPU Memory directly via the DDRB and PORTB

Note:
to turn off LED, change “PORTB |= (1<<PB5);” to “PORTB &= ~(1<<PB5);”

C-Language with Arduino Library (program name: intro_arduino)

```
void setup() {
  pinMode(13, OUTPUT);    // set Arduino Pin 13 Output Pin
  digitalWrite(13, HIGH); // set High to Arduino Pin 13
}

void loop() { }
```

We do not see the INSTRUCTION SET codes here, we also do not need to deal with the CPU Memory via the DDRB and PORTB. They are hidden and replaced by the functions from the Arduino Library. The Program Codeshas now become so much easier now

Note:
to turn off LED, change “digitalWrite(13,HIGH);” to “digitalWrite(13,LOW);”

ATMEGA328/Arduino Uno - Introduction to Programming
https://github.com/teaksoon/lmaewapm

AVR Assembly Language (program name: intro_asm)

Pros:
We can use the entire INSTRUCTION SET and access the micro-controller CPU Memory directly, we have maximum flexbility and we can use the micro-controller to its maximum abilities. Our program will be very small as we only use what is required. This example program is only 144 bytes in size

Cons:
Assembly Language is difficult to code. You need to know how the INSTRUCTION SET and micro-controller CPU Memory works. Assembly Coding is one and this can be very difficult as the program grows bigger or when it becomes more complicated

C-Language with AVR Library (program name: intro_avr)

Pros:
The INSTRUCTION SET are hidden from us, a single line of C-Language Codes can sometimes perform multiple INSTRUCTIONS from the INSTRUCTION SET for us behind the scene. While the INSTRUCTION SET is hidden from us, we can still access and manipulate CPU Memory directly. We still have a very high degree of flexbility. Our program can still be very small. In this example, the Program size is even smaller than the Assembly Language, just 138 bytes in size

Cons:
We still have to deal with micro-controller CPU Memory, can still be difficult because manipulating CPU Memory requires the understanding on how the CPU Memory works

C-Language with Arduino Library (program name: intro_arduino)

Pros:
The INSTRUCTION SET and CPU Memory are totally hidden from us. We do not need to know anything about how INSTRUCTION SET and CPU Memory works. Program are way easier to code and our Program can even work on non AVR based micro-controllers as are codes are not tied to a particular INSTRUCTION SET or a particular CPU Memory.

Cons:
Program size are now bloated. This example uses 712 bytes, which is more than 5 times larger than the other 2 styles and codes will also run way slower than the other 2 styles. Coding is not flexible anymore and some features in the INSTRUCTION SET or the CPU Memory may not even be utilized

Example 2:
A simple program to “Blink” the Arduino Uno Board test LED

Source Codes are available for download.

https://github.com/teaksoon/lmaewapm/blob/main/2021_11_20_intro_program_source.zip

Download and unzip the source .zip file and use the Arduino IDE Software to open the Program in the following folders.

intro_blink_asm – Coding in Assembly Language
intro_blink_avr – Coding in “C-Language with AVR Library”
intro_blink_arduino – Coding in “C-Language with Arduino Library”

1. Open the Program files from those folder into the Arduino IDE Software

2. Click Upload and watch the test LED on the Arduino Uno Board turning ON and OFF every 250ms.

IF you look at the Assembly Language Source code (blink_asm), you will notice that, just to make a time delay of 250ms will take quite an effort. From that code alone, you will probably understand why we normally avoid Assembly Language and use the two C-Language style instead.

DO NOT WORRY IF YOU ARE LOST IN THIS TUTORIAL

The purpose of this tutorial is to demonstrate all the different styles of Programming that we can use when coding with the Arduino IDE Software. We will still learn Programing from the very basics in other lessons

Actually, we have another Programming option. That is to mix all the 3 Programming Styles together. We might just do that as we go along in our future lessons.

We will not touch that for now