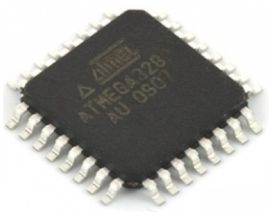
**Program are instructions from Programmer(Human) to the micro-controller CPU, asking the CPU to do task.**



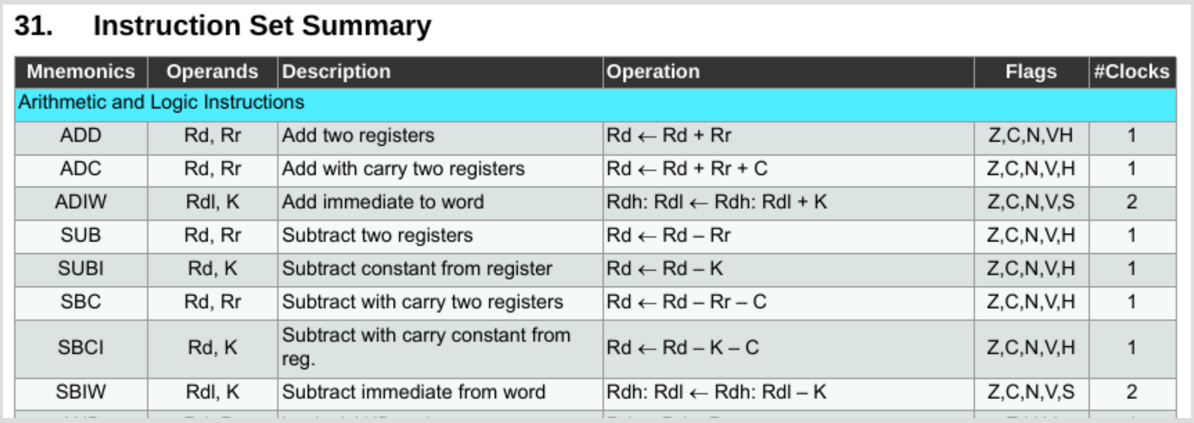
**The CPU can only do task according to what it is capable of doing.**

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” will perform a simple task

The ATMEGA328 micro-controller is using the AVR INSTRUCTION SET, The picture below shows some of the “INSTRUCTION” from the “AVR INSTRUCTION SET”



**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 |

Other micro-controller may use other type of “INSTRUCTION SET”, some of the popular ones are, ARM, x86, RISC-V,...

Sending a single “INSTRUCTION” from the “INSTRUCTION SET” to the 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 “INSTRUCTION” is known as a **“PROGRAM”**

**When we are using the Arduino Platform to write our Program for the ATMEGA328 micro-controller. We will create our PROGRAM in 3 different styles, using the Arduino IDE Software**

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

**Lets write 2 small PROGRAM using all the 3 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, you dont need to key in the codes manually. These codes are just to show you the different programming style.

1. A simple program to “Turn ON” the Arduino Uno Board test LED
intro_asm.zip
intro_avr.zip
intro_arduino.zip

2. A simple program to “Blink” the Arduino Uno Board test LED
intro_blink_asm.zip
intro_blink_avr.zip
intro_blink_arduino.zip

Each zip file opens up into a folder each, use your Arduino IDE software to open the folder to access the the source code.

If you have Arduino Uno, simply connect it to your computer and upload the Programs to try out.

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

### 1. AVR Assembly Language ( 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. The CPU Memory DDRB, PORTB are also being manipulated here.

If you wish the turn off the LED, you can code CBI PORTB,5

### 2. C-Language with AVR Library ( intro_avr )

```
#include "avr/io.h"
int main() {
  DDRB  |= (1<<PB5); // 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. This is because they are hidden from us by the C-Language. However, in this program we are still manipulating the CPU Memory via the DDRB and PORTB

If you wish the turn off the LED, you can code PORTB &= ~(1<<PB5);

### 3. C-Language with Arduino Library ( intro_arduino )

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

void loop() { }
```

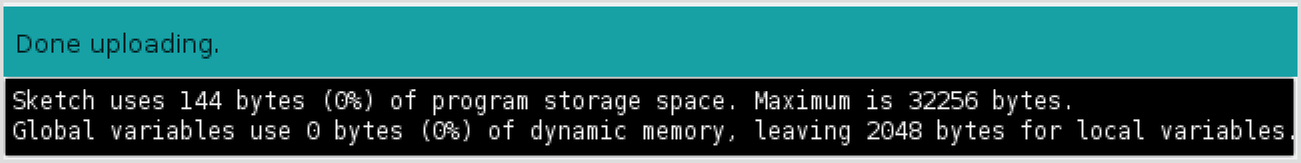
When we use C-Language with Arduino Library, both the AVR INSTRUCTION SET and the CPU Memory DDRB, PORTB are hidden from us, replaced by codes from the Arduino Library. The codes has now become so much easier.

If you wish the turn off the LED, you can code digitalWrite(13,LOW);

## 1. AVR Assembly Language ( intro_avr )

Pros:
1. We use the INSTRUCTION SET and all the intenrnals directly, means we have the maximum flexbility here

2. Our program will be very small, it was just 144 bytes


```
Done uploading.
Sketch uses 144 bytes (0%) of program storage space. Maximum is 32256 bytes.
Global variables use 0 bytes (0%) of dynamic memory, leaving 2048 bytes for local variables.
```

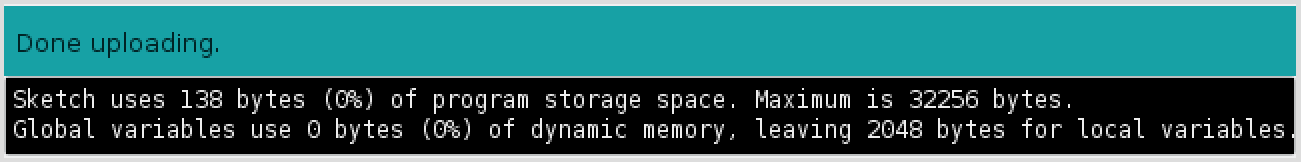
Cons:
1. Very difficult to code, this program is small so it does not look difficult, when program grows bigger... it will become very difficult

## 2. C-Language with AVR Library ( intro_avr )

Pros:
1. The INSTRUCTION SET is hidden from us, coding has become easier

2. While the INSTRUCTION SET is hidden from us, we can still access and manipulate the CPU Memory directly, means we still have a very high degree of flexbility

3. Our program will be as small as the Assembly Language, in this example it is even smaller than Assembly at 138 bytes only


```
Done uploading.
Sketch uses 138 bytes (0%) of program storage space. Maximum is 32256 bytes.
Global variables use 0 bytes (0%) of dynamic memory, leaving 2048 bytes for local variables.
```

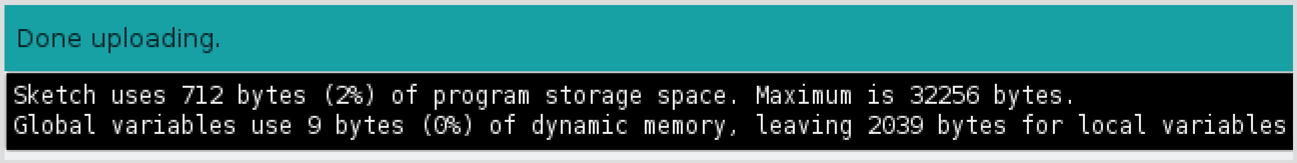
Cons:
1. We still have to deal with CPU Memory, can still be difficult because manipulating CPU Memory is complicated and involves alot of work

## 3. C-Language with Arduino Library ( intro_arduino )

Pros:
1. This is the easiest to program, we do not need to deal with the INSTRUCTION SET, we also do not need to deal with the CPU Memory ( we still can access to them if we wish to ). It is easy and yet very flexible. This is what makes the Arduino Platform so popular

2. This same code can even be used on some non AVR based micro-controllers

Cons:
1. Program size are bloated, it uses 712 bytes, more than 5 times larger than the code written in the other 2 styles


```
Done uploading.
Sketch uses 712 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables
```

## 2. A simple program to “Blink” the Arduino Uno Board test LED

These program are now much bigger, we wont be looking at the codes in this page. Source Codes is available for download,

blink_asm.zip
blink_c.zip
blink_arduino.zip

just open them with the Arduino IDE Software.

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 code, just to make a time delay of 250ms will take quite an effort. From that code alone, you will probably understand why normally avoid Assembly Language and use the C-Language instead.

DO NOT WORRY IF YOU ARE LOST IN THIS TUTORIAL

The purpose of this tutorial is to demonstrate to you the different styles of Programming that we can do with the Arduino IDE Software.

We will still code from the basics, we have not not even started on how to deal with the I/O Pins yet

We will be learning Programming by using the 3rd Option

**3. C-Language with Arduino Library**