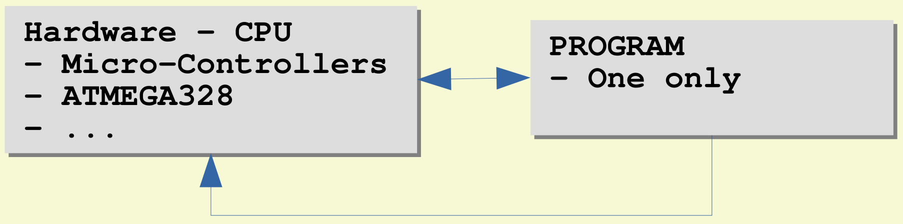
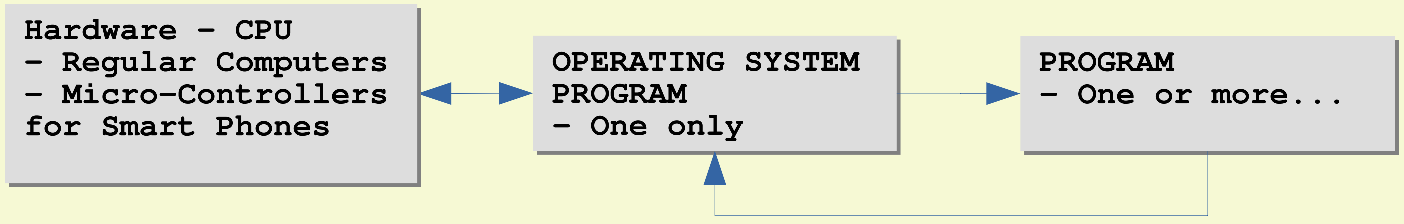
ATMEGA328/Arduino Uno - C-Language_01_Bare_Minimum
https://github.com/teaksoon/lmaewapm

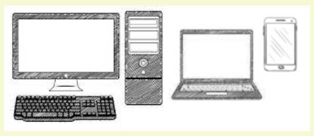
SETUP 1: PROGRAM and CPU



This is what we are going to do. The C-Language PROGRAM will interact directly with the CPU (via INSTRUCTION SET and CPU MEMORY)

SETUP 2: PROGRAM and CPU (Requires Operating System)



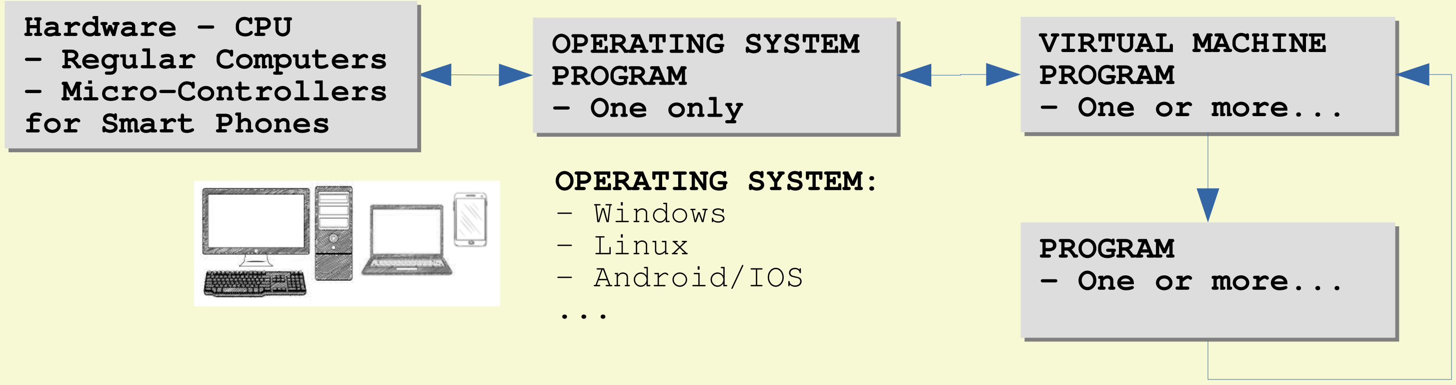


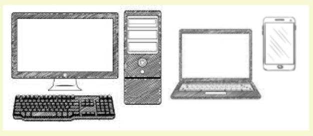
OPERATING SYSTEM:
- Windows
- Linux
- Android/IOS
...

In this type of Setup, the PROGRAM normally interact with the Operating System, the Operating System will deal with the CPU. However, if C-Language is used, the PROGRAM can interact with both OPERATING SYSTEM and the Hardware CPU. This is useful when speed and flexibility is required, direct interaction with the CPU gives the optimum result

SETUP 3: PROGRAM and CPU (Requires Operating System and Virtual Machine)

VIRTUAL MACHINE:
- Operating System
- Java
- Python
...

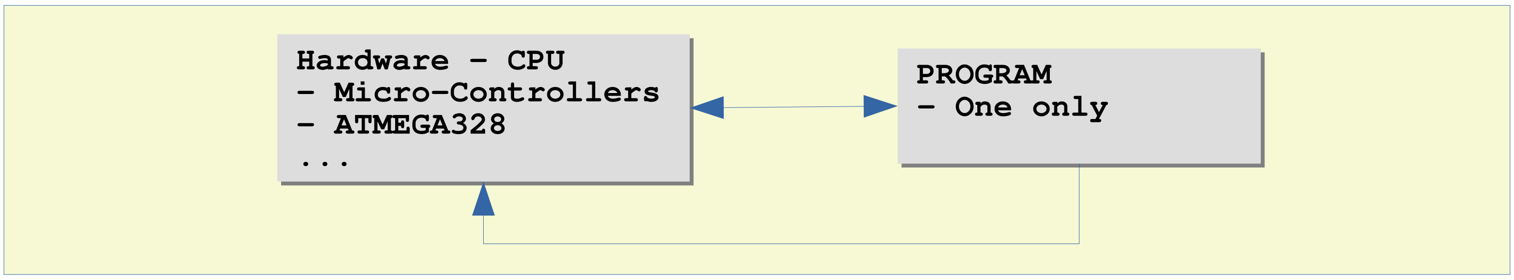




OPERATING SYSTEM:
- Windows
- Linux
- Android/IOS
...

In this type of Setup. When the PROGRAM is written in Java, Python,... they can only interact with the "VIRTUAL MACHINE" specific for each of those Language. However, if C-Language is used instead, we can bypass the "VIRTUAL MACHINE" PROGRAM and interact with the OPERATING SYSTEM or the CPU directly

SETUP 1: PROGRAM and CPU (Without Operating System)



An **ATMEGA328 micro-controller PROGRAM** is a sequence well arranged "instruction codes" from the **AVR INSTRUCTION SET** that can be recognized by the micro-controller Central Process Unit (CPU) to **manipulate the CPU Memory (REGISTERS). The ATMEGA328 micro-controller will then perform various task, mostly based on what is in the CPU Memory (REGISTERS).**

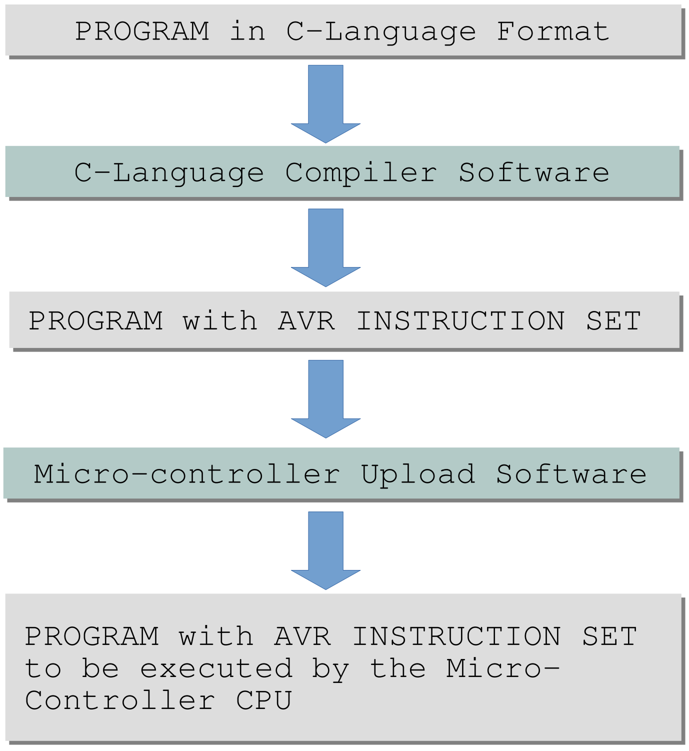
ATMEGA328 micro-controller PROGRAM is all about the following, from the ATMEGA328P Datasheet:
1. **AVR INSTRUCTION SET:** from ATMEGA328 Datasheet: Topic 31. from page 281 "31. Instruction Set Summary"
2. **ATMEGA328 CPU Memory(REGISTERS):** from ATMEGA328 Datasheet: Topic 30. from page 271 "30. Register Summary"

By just looking into the two topics above from the ATMEGA328 Datasheet, using the INSTRUCTION SET and the CPU Memory directly is not easy. Even for a simple task, we will need to do alot of work

Instead of coding the AVR INSTRUCTION SET directly, we use the C-Language

The C-Language has its own simplified "instruction set" in the form of "Keywords" and "Symbols". C-Language have some code formatting and arrangement rules that we need to comply.

A Software called "C-Language Compiler" be able to recognize the C-Language PROGRAM codes and based on its code formatting and arrangement, will convert the C-Language PROGRAM codes into AVR INSTRUCTION SET PROGRAM for us.


PROGRAM in C-Language Format
C-Language Compiler Software
PROGRAM with AVR INSTRUCTION SET
Micro-controller Upload Software
PROGRAM with AVR INSTRUCTION SET to be executed by the Micro-Controller CPU

The C-Language has its own simplified “instruction set” in the form of “Keywords” and “Symbols”. C-Language also have some code formatting and arrangement rules that we need to follow.

The C-Language Keywords and Symbols

Keywords

MEMORY	CONTROL
01.void	21.return
02.const	22.if
03.static	23.else
04.signed	24.for
05.unsigned	25.while
06.short	26.do
07.char	27.switch
08.int	28.case
09.long	29.break
10.float	30.default
11.double	31.continue
12.sizeof	32.goto
13.struct	
14.union	
15.enum	
16.typedef	
17.auto	
18.register	
19.extern	
20.volatile	

Symbols

CONTROL	LOGIC	MATH	BIT OP
#	==	*	|
< >	!=	%	&
“	<	/	^
‘	>	+	~
( )	<=	-	<<
{ }	>=		>>
[ ]			
//	&&		
/* */	||		
;	!		
,			
:			
?			
.			
\			
=			
MEMORY			
&			
*			

Below are not part of the C-Language but is often used with the C-Language.

The C-Language pre-processor

The C-Language pre-processor give special instructions to the C-Language Compiler software and also acts like a “Search and Replace” tool before the actual C-Language Compiler Software process starts.

#define	Similar to “Search and Replace”
#include	Get extra codes from “.h” Source Code file
#undef	Cancel a #define that was specifed earlier
#ifdef	If a #define already exist
#ifndef	If a #define does not exist
#if	Condition is true.
#else	Condition is false
#elif	Combination of #else and #if
#endif	Ends of the conditional #if
#error	Output error message
#pragma	Special Command to the C-Language Compiler Software

C-Language only have 32 keywords and a bunch of Symbols. Out of that, many will not even be used in most of our Programs. This may looks like very little, C-Language is a very powerful Programming Language.

Almost all modern computer Operating System is written in C-Language. Almost all Machine” Software for other Programming Language, including itself are also written in C-Language. When comes to micro-controllers, C-Language is used in almost everything

We will learn the C-Language Keywords, Symbols and coding rules by making real C-Language Program.

C-LANGUAGE CODE ARRANGEMENT and RULES:

- In C-Language PROGRAM the functional instruction codes are kept in "containers" known as "functions"

- We must have at least one "function" in a C-Language PROGRAM

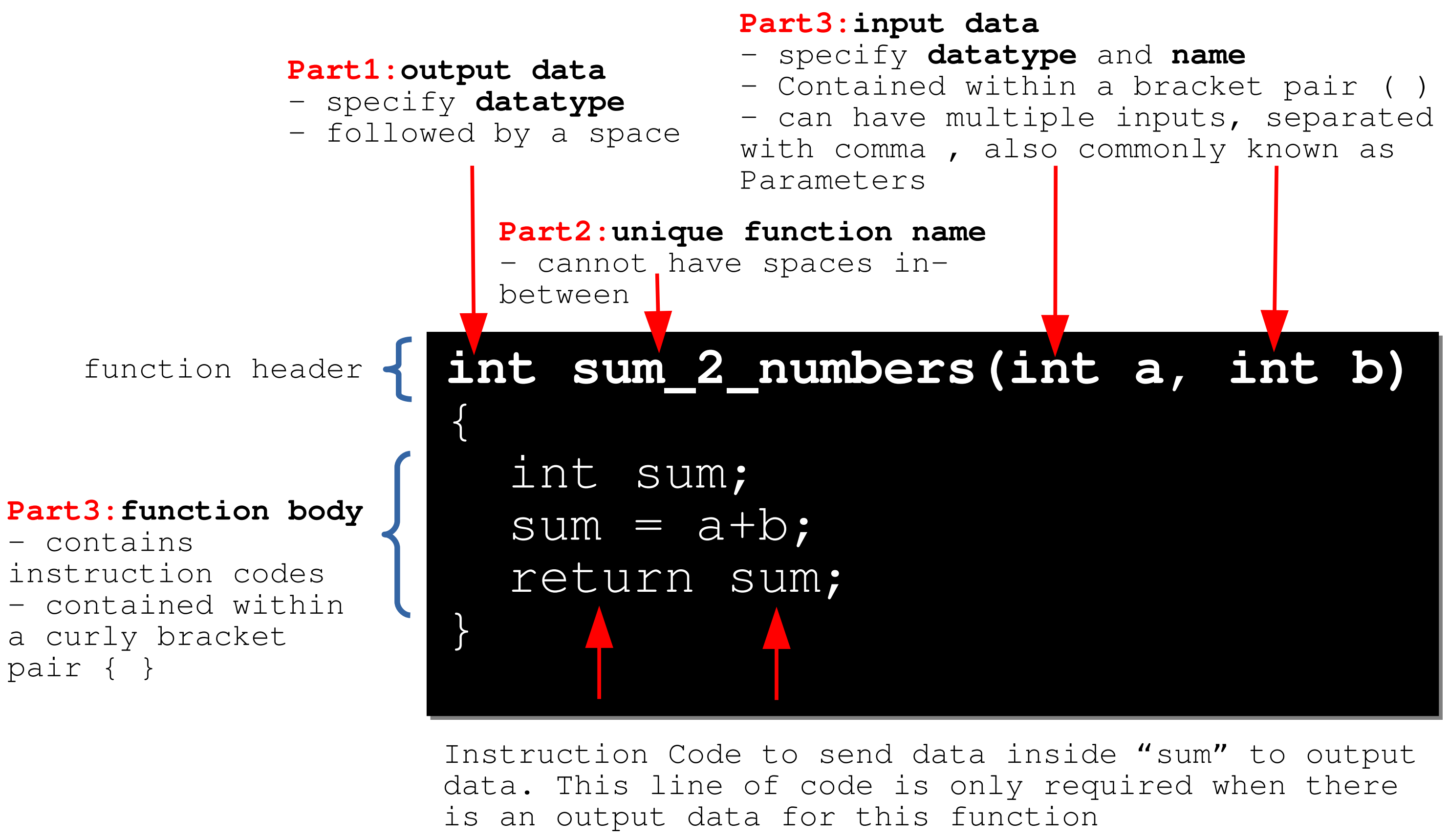
- We can have more than function in a C-Language PROGRAM, from the many functions, we must have one function with a function name "main". The C-Langauge PROGRAM will start from the "main" function

Now we know that we need at least one "function" to make a C-Language PROGRAM,

WHAT IS A FUNCTION and HOW TO MAKE ONE ?

A function have 4 Parts,
Part1:output data – Optional: if none simply code "void")
Part2:function name
Part3:input data – Optional: if none, just leave empty bracket pair ( )
Part4:function body – Optional: if none, just leave empty curly bracket pair { }

BELOW is a typical C-Language function



Part1:output data
- specify **datatype**
- followed by a space

Part3:input data
- specify **datatype** and **name**
- Contained within a bracket pair ( )
- can have multiple inputs, separated with comma , also commonly known as Parameters

Part2:unique function name
- cannot have spaces in-between

function header

```
int sum_2_numbers(int a, int b)
{
   int sum;
   sum = a+b;
   return sum;
}
```

Part3:function body
- contains instruction codes
- contained within a curly bracket pair { }

Instruction Code to send data inside "sum" to output data. This line of code is only required when there is an output data for this function

Do not worry about the codes and symbols in the example function above, we will learn about them in later lessons.

C-LANGUAGE CODE ARRANGEMENT and RULES:

– In C-Language PROGRAM the functional instruction codes are kept in "containers" known as "functions"

– We must have at least one "function" in a C-Language PROGRAM

– We can have more than function in a C-Language PROGRAM, from the many functions, **we must have one function with a function name "main".** The C-Langauge PROGRAM will start from the "main" function

Lets make a most basic legal C-Language PROGRAM.
From the Rules above we only need to make a "main" function

Lets make our "main" function step by step, (this way of coding can be used for making other functions)

Step 1: Make a skeleton function with function name "main", assuming there is no output and no input required

```
void main(){}
```

Step 2: Does the function produce output ? If yes specify the output "datatype", then followed by a space or more
C-Language Rules(for "main" function only): "main" function must have an "int" datatype output

```
int main(){}
```

Step 3: if the function produce output, we must have an "instruction code" to return an output value, we put the code in the Function Body within the curly bracket { } pair
C-Language rules(for any functions): Each line of "instruction code" inside the Function Body must end with a semi-colon ;

```
int main(){
  return 0;
}
```

Step 4: does our function requires any input values ? Our "main" function does not need an input value. So, we leave the bracket ( ) pair empty

```
int main(){
  return 0;
}
```

Thats is it. Wwe are done. We make our "main" function in C-Language following all the C-Language rules. We can put that code in our Arduino IDE Software and upload it into our Micro-controller on our Arduino Uno

In Arduino IDE Software, Save as: "c_language_bare_minimum" and upload

```
int main() {
  return 0;
}
```

This is a bare minimum C-Language PROGRAM. Although it does nothing, it is a LEGAL C-Language PROGRAM. From this small program alone, we used two C-Language Keywords "return" and "int" also a few symbols ( ) { } and ;