18ES611 Embedded System Programming

Sarath tv

GNU

•Collection of wholly free software(applications, libraries, developer tools, games)- giving rise to an OS.

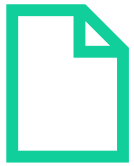
“it respects users' freedom”.

•GNU Project- *free software, mass collaboration project.*

•Freedom and control, the rights to freely run the software, copy and distribute it, study it, and modify it.

Four essential freedoms

Freedom 0



Freedom 1

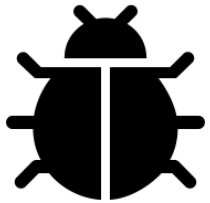
Freedom 2

Freedom 3

Development Tools

- Text editors.
- Compilers/Interpreters.
- Debug.
- Document.

- IDE



A simple C program

```
/*
* File: hello.c
* This simple C program prints out the text "Hello world!".
*/
#include<stdio.h>
void main(void)
{
printf("Hello world!\n");
}
```

How do I build and run this code ??

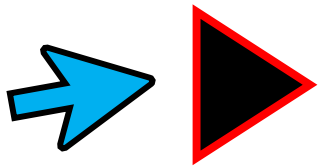
- Most common methods

- IDE
  - *On Button Click*

- Command Line Interface
  - *invoke the compiler manually*

In IDE ??

What all things happens when you click on **“Build and Run ”** option in a IDE



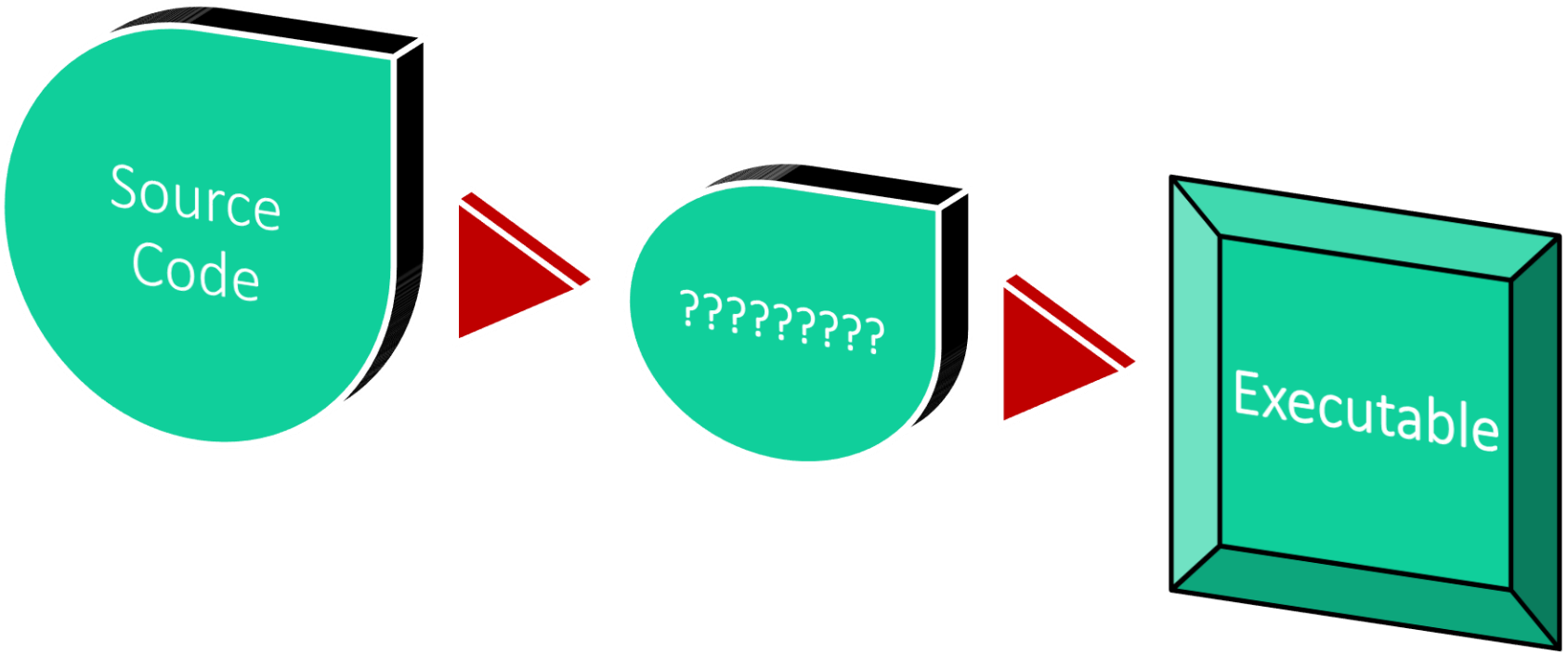
```
#include
<stdio.h>
int main()
{
printf(“Hello”)
;
return 0;
}
```

Source code

```
1000101010101010
0001001010101011
0111111001100000
0010110011010100
0101110111000110
0111110011110000
0001100111101010
0100100101010000
```

Executable code

The C compilation model

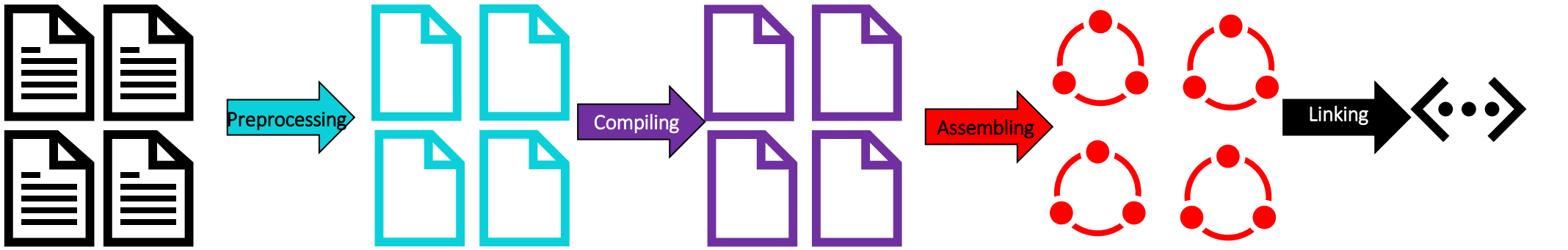


```
#include
<stdio.h>
int main()
{
printf("Hello")
;
return 0;
}
```

Source code

```
100010101010101
000100101010111
011111100110000
001011001101010
010111011100011
011111001111000
000110011110101
010010010101000
```

Executable code



SOURCE CODE

.i files

.s files

.o files

EXECUTABLE

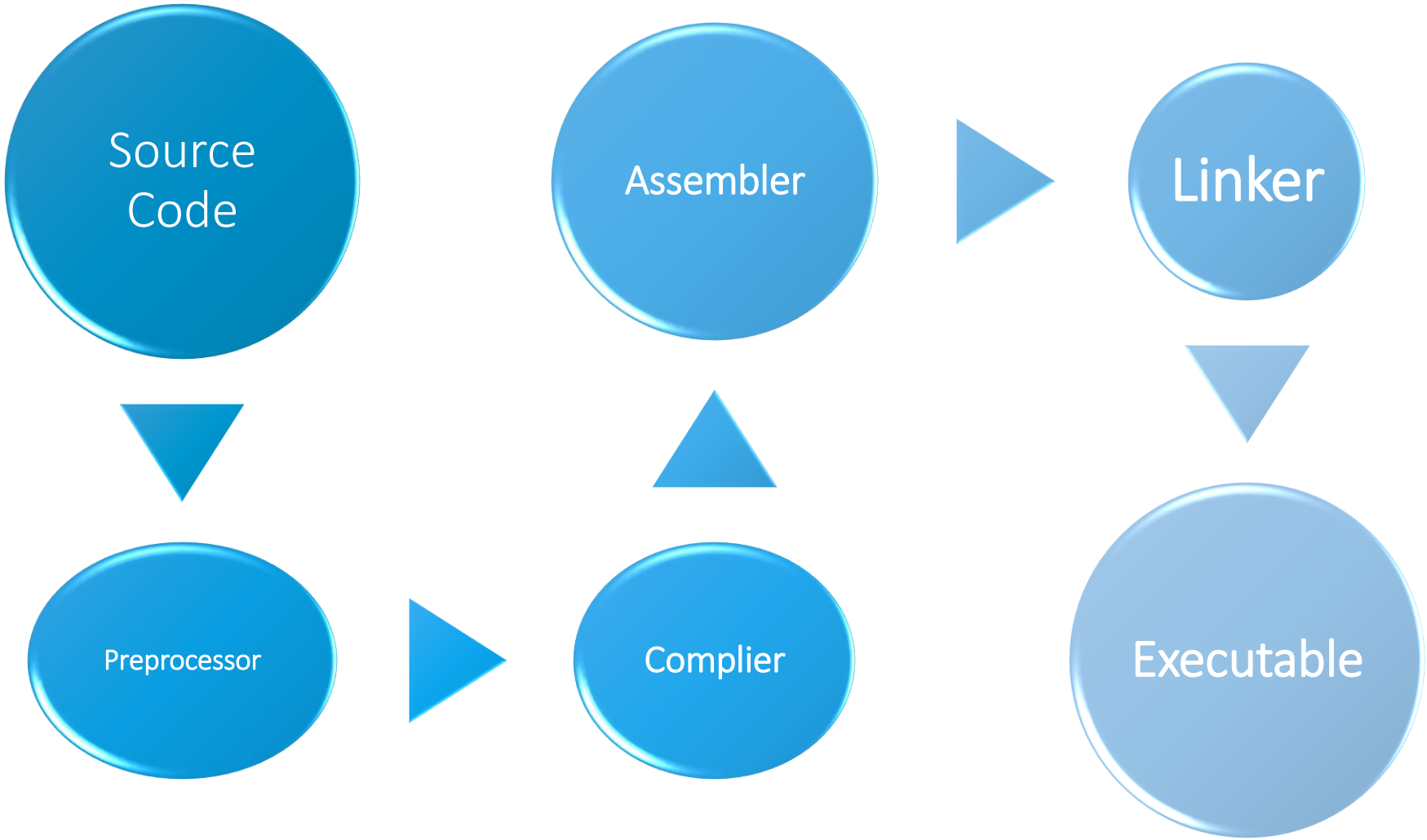
Preprocessing – the preprocessor substitutes all preprocessor directives in the original source code **.c** file with actual library code that implements those directives. For instance, library code is substituted for **#include** directives. The generated file containing the substitutions is in text format and typically has a *.i file* extension .

Compiler stage – the compiler translates the high-level instructions in the **.i** file into low-level Assembly language instructions. The generated file containing the translation is in text format and typically has a ***.s*** *file* extension.

Assembling – the assembler converts the Assembly language text instructions in the **.s** file into machine code. The generated object file containing the conversion is in binary format and typically has a **.o** file extension

Linking – the linker combines one or more binary object **.o** files into a single executable file. The generated file is in binary format and typically has a .exe or .out file extension.

The C compilation model



THANK YOU!!!!!