Introduction to Programming Languages

IN 1101 PROGRAMMING FUNDAMENTALS

Programming Language

- ❑ A programming language is a set of rules that provides a way to telling a computer what operations to perform.
- ❑ A programming language is a set of rules for communicating an algorithm.

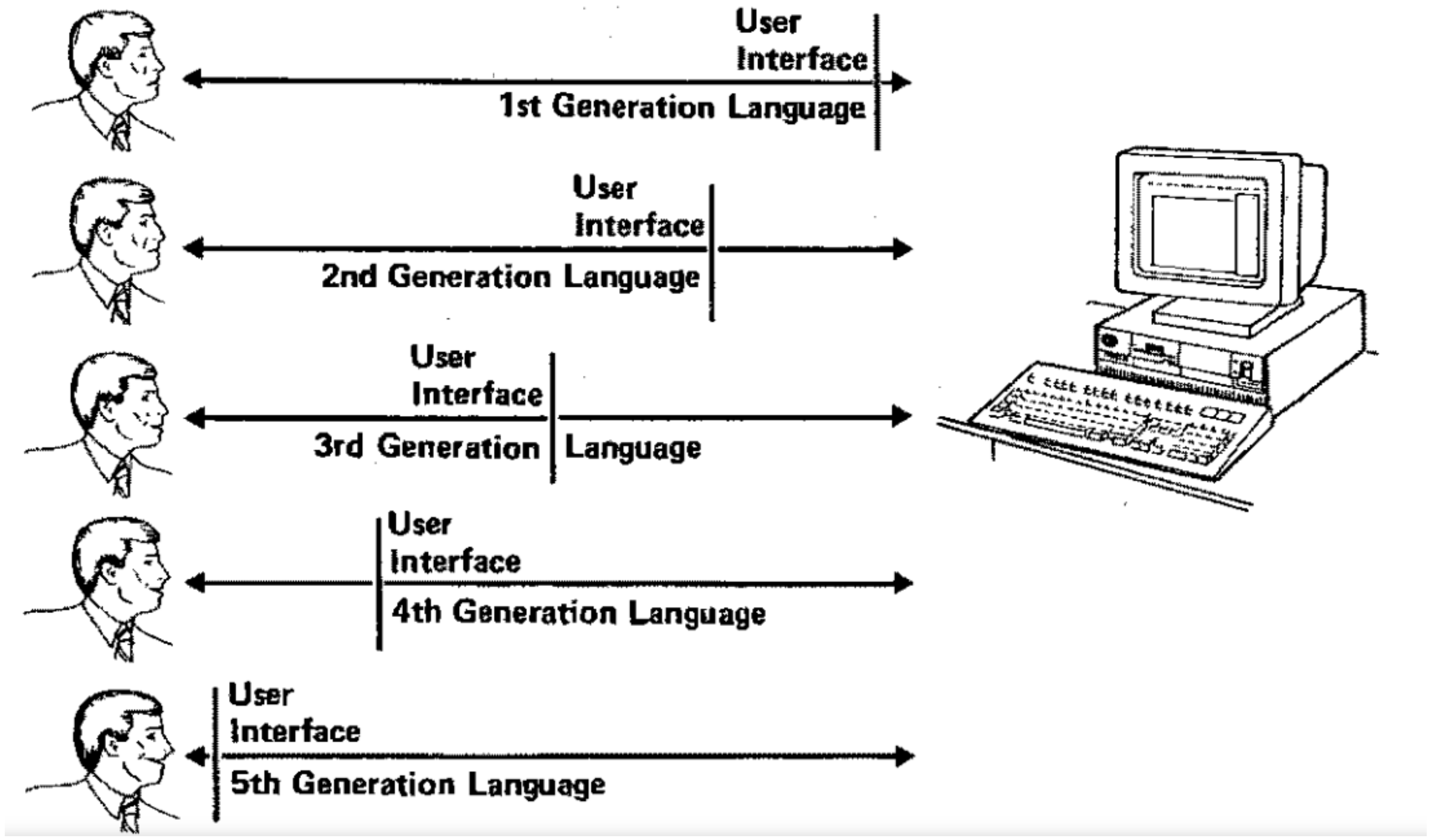
Programming Language Cont.

- A programming language has words, symbols, and rules of grammar.
- Grammatical rules are called ‘Syntax’.
- Each programming language has a different set of syntax rules.

Why So Many Programming Languages?

Programming Language Generations

- 1st Generation –
  - Machine Languages
- 2nd Generation –
  - Assembly Languages
- 3rd Generation –
  - High-Level Languages
- 4th Generation –
  - Very High-Level Languages
- 5th Generation –
  - Logic/AI Languages



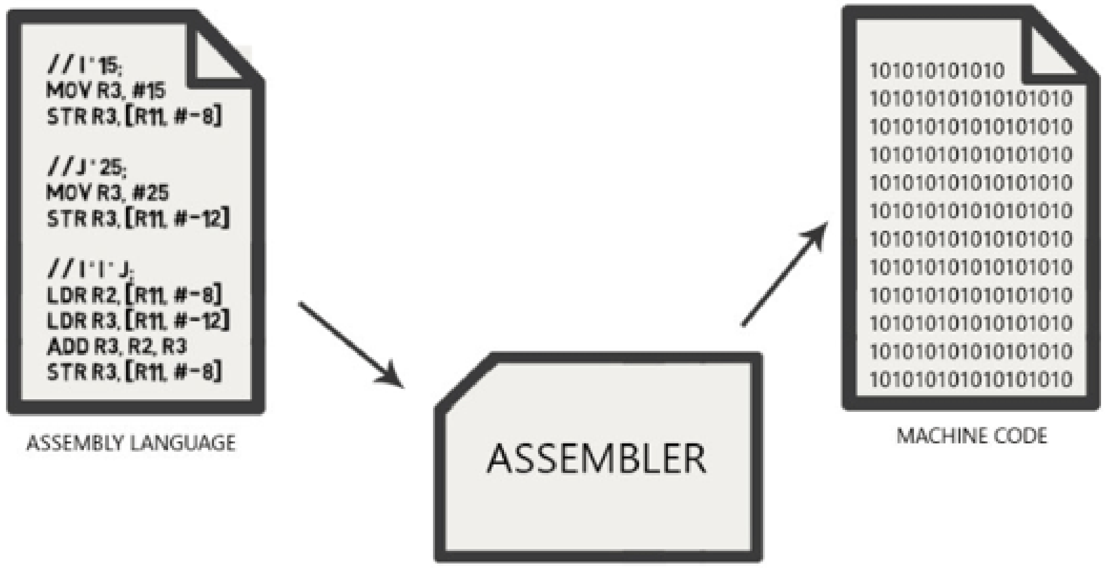
Machine Language(1GL)

- **Machine Language** :
  - The set of instruction codes, in binary, which can be directly understood by the CPU without translating the program.
  - The lowest-level, programming language.
  - Machine Dependent.
  - Difficult to program.
  - Error Prone.
  - Difficult to modify.

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111

                 . . .
```

Assembly Language(2GL)

- ❑ Assembly Language :
  - Allows the programmer to use abbreviations or words instead of binary numbers, known as mnemonics.
  - A low-level language.
  - Limitations:
    - Machine dependent.
    - Knowledge of hardware.
    - Machine level coding.



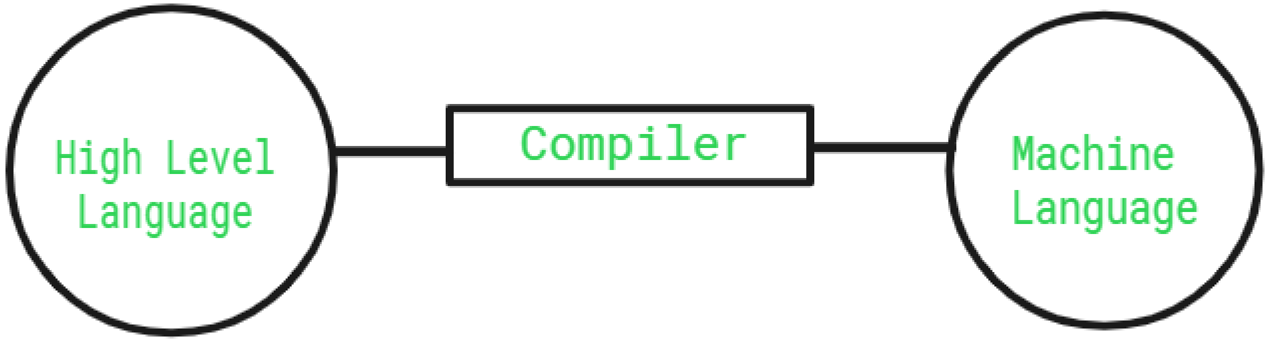
```
//I°15;
MOV R3, #15
STR R3, [R11, #-8]

//J°25;
MOV R3, #25
STR R3, [R11, #-12]

//I°I°J;
LDR R2, [R11, #-8]
LDR R3, [R11, #-12]
ADD R3, R2, R3
STR R3, [R11, #-8]
```

```
101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
```

High-Level Language(3GL)

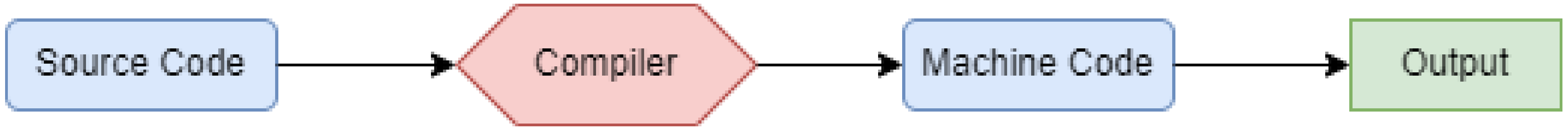
❑ High-Level Language :

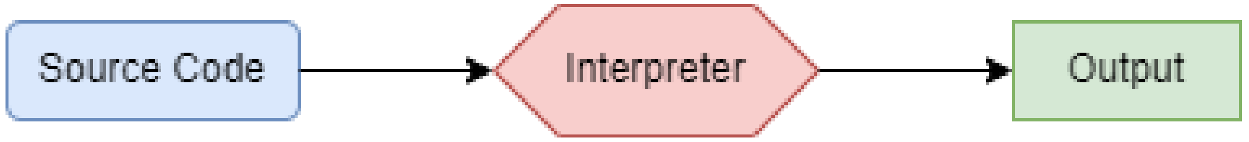
- Closer to English but included simple mathematical notation.
- Use instructions, “Statements”.
- Individual high-level language statements are macroinstructions which generates several machine instructions.
- Compiler vs. Interpreters.
- Advantages :
  - Easier to learn and understand.
  - Potential for errors is reduced.
  - Machine independent.
- Disadvantages:
  - Require a greater amount of computer time



Compilation vs. Interpretation

- Both compiler and interpreter transforms code written in high-level programming language into the machine code.
- Difference:
  - Compiler will convert the code into machine code before program run.
  - Interpreters convert code into machine code when the program is run.





Activity

“Many compilers do not directly generate machine code”

- Do you agree with the above statement? Give reasons and explain your answer.
- Upload your answer to the provided link.

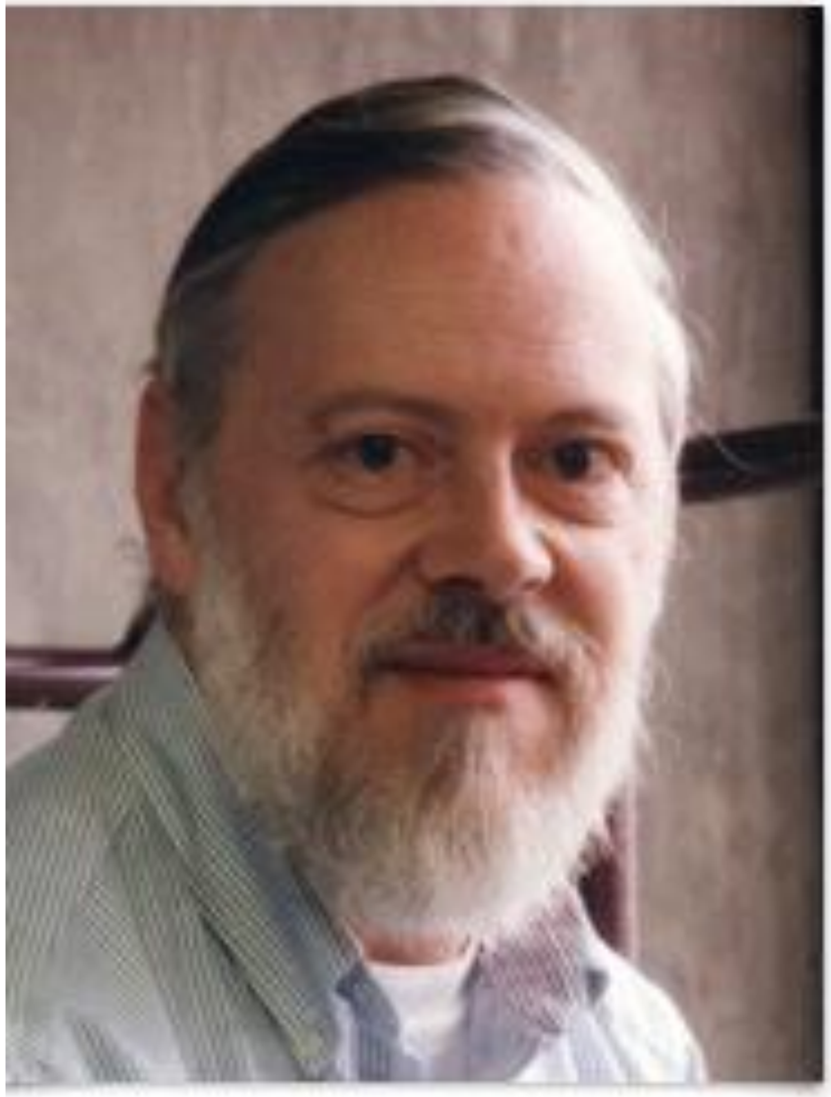
Programming Language and Human Language

- ❑ In both kind of languages, you have to learn new vocabulary, syntax and semantics.
- ❑ Ambiguity - “I saw the man with a telescope”
- ❑ Computer languages lack ambiguity and vagueness.
- ❑ In a programming language a sentence either means one thing or it means nothing.

“It’s Different”

What Determines Good Language?

- Writability
  - Quality of expressivity in a language.
  - Writability should be clear, concise, quick and correct.
- Readability
  - The quality of a language that enables the reader (even non-programmers) to understand the nature of the computation or algorithm.
- Orthogonality
  - Every combination of features is meaningful.
- Reliability
  - Assurance that a program does not behave unexpectedly.
- Maintainability
  - The ease of which errors can be found and corrected and new features added



Programming With C

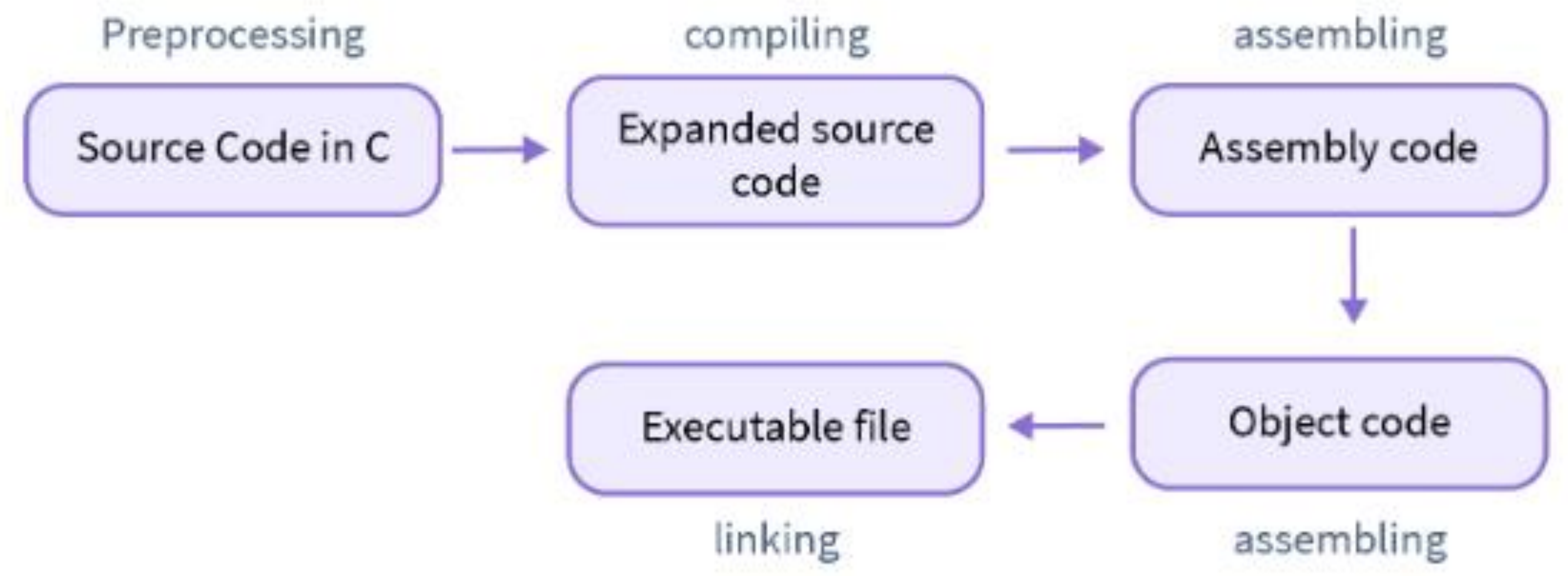
- ❑ Developed by Dennis Ritchie, Bell Labs in early 1970s.
- ❑ C is often called a ‘Middle Level’ programming language.
- ❑ Portable (Machine-Independent).

First C Program

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

Compilation Process in C

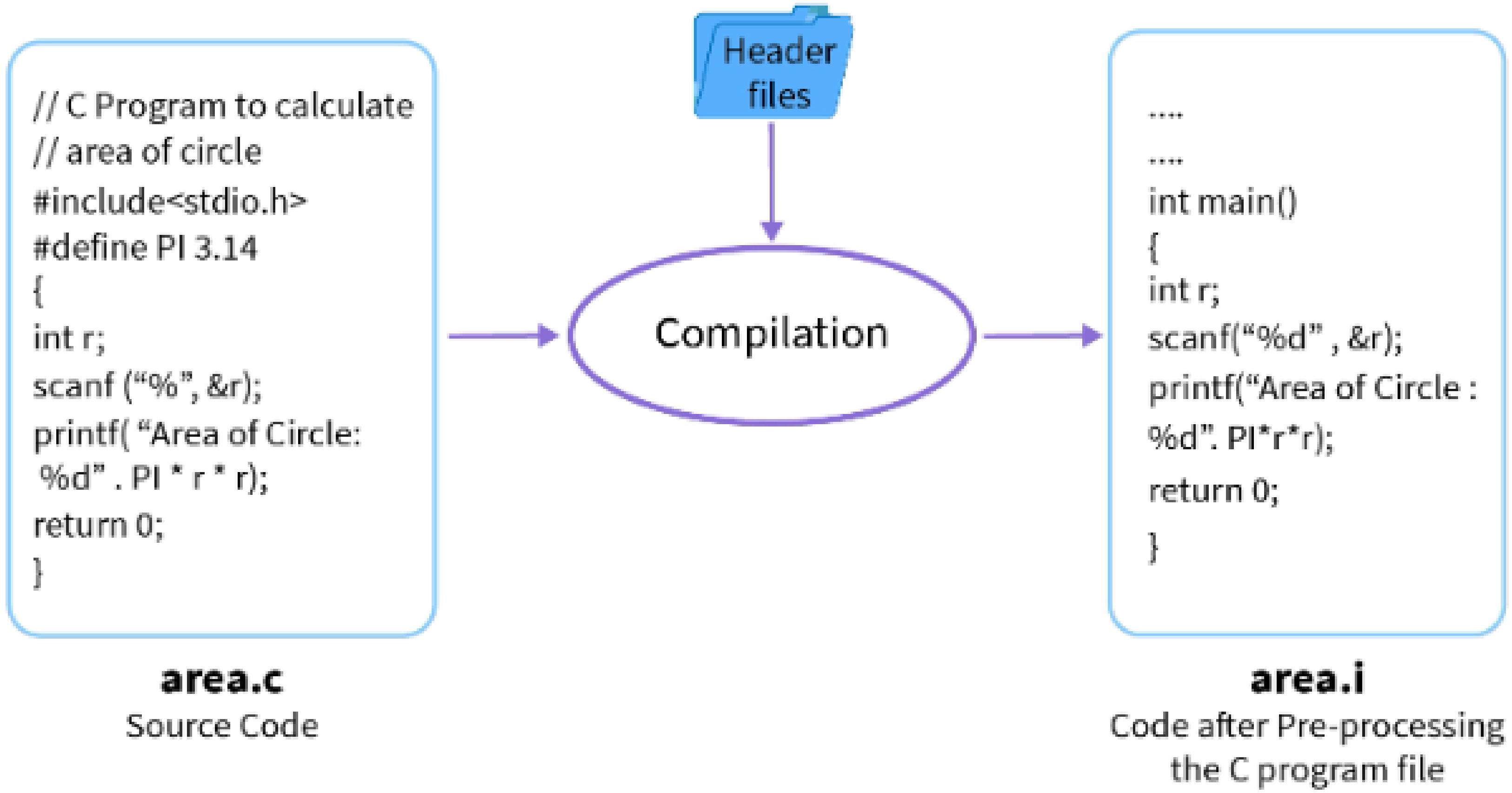
- Involves four steps:
  1. Pre-processing
  2. Compiling
  3. Assembling
  4. Linking



Compilation Process in C–Pre-processing

- Following tasks will be carried out during the pre-processing.
  - Comments removal
  - Macros expansion
    - Macros are some constant values or expressions defined using the **#define** directives.
    - Macros will be replaced by their values.
  - File inclusion
    - If the program contains #include, this line will be replaced by the original contents of the header file.
  - Conditional compilation
    - The preprocessor often operates on a conditional compilation and reduces the code by adding only those lines that fulfill the condition.
    - Some Conditional Compilation preprocessor directives are:
      - #undef , #ifdef ,#ifndef ,#if ,#else ,#elif ,#endif

Compilation Process in C–Pre-processing



```
// C Program to calculate
// area of circle
#include<stdio.h>
#define PI 3.14
{
int r;
scanf ("%", &r);
printf( "Area of Circle:
 %d" . PI * r * r);
return 0;
}
```

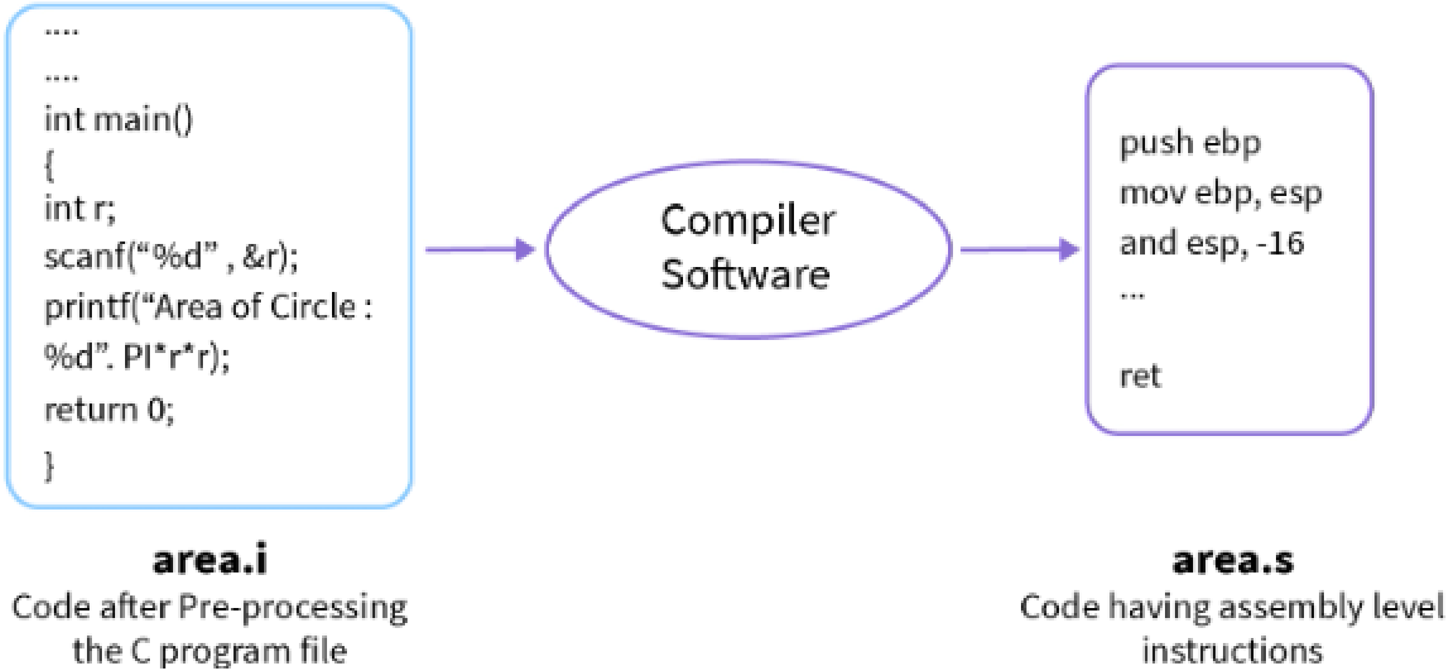
area.c
Source Code

```
....
....
int main()
{
int r;
scanf("%d" , &r);
printf("Area of Circle :
%d". PI*r*r);
return 0;
}
```

area.i
Code after Pre-processing the C program file

Compilation Process in C–Compiling

- Convert the intermediate (.i) file into an assembly file (.s) having assembly level(low level) instructions.
- Uses and inbuilt compiler software.



```
....
....
int main()
{
int r;
scanf("%d" , &r);
printf("Area of Circle :
%d". PI*r*r);
return 0;
}
```

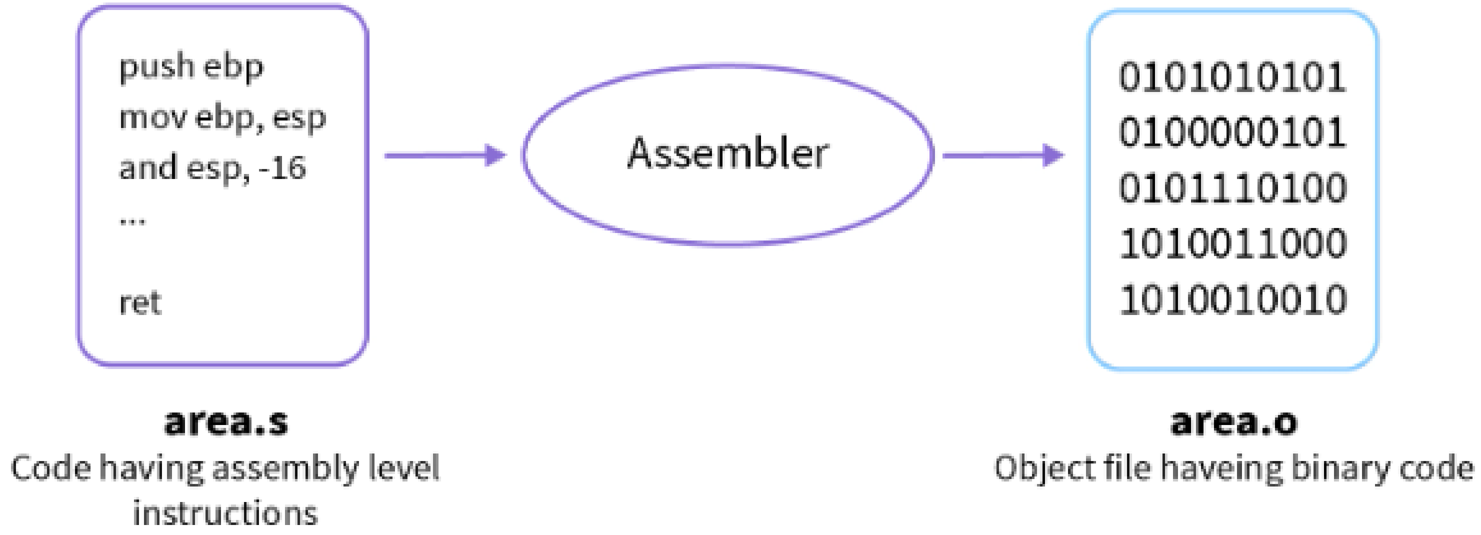
area.i
Code after Pre-processing the C program file

```
push ebp
mov ebp, esp
and esp, -16
...

ret
```

area.s
Code having assembly level instructions

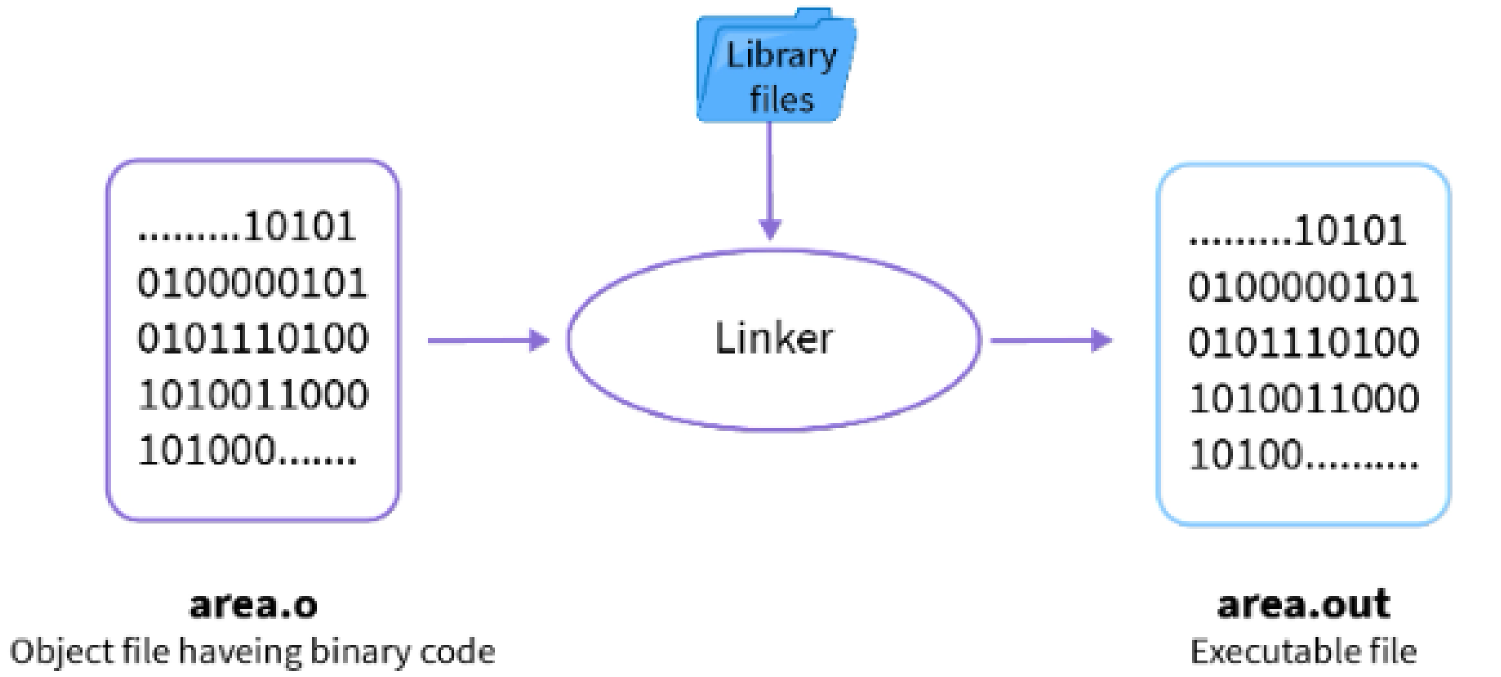
Compilation Process in C–Assembling

- Assembly level code (.s file) is converted into a machine-understandable code.
- Done by a pre-written program known as 'Assembler'.
- Generated file is known as 'Object file' with an extension of .obj in DOS and .o in UNIX.



Compilation Process in C–Linking

- Process of including the library files into the program.
- Linking process generates an executable file.



Questions?