CS1002 – Programming Fundamentals

Lecture # 08
Tuesday, September 13, 2022
FALL 2022
FAST – NUCES, Faisalabad Campus

Muhammad Yousaf

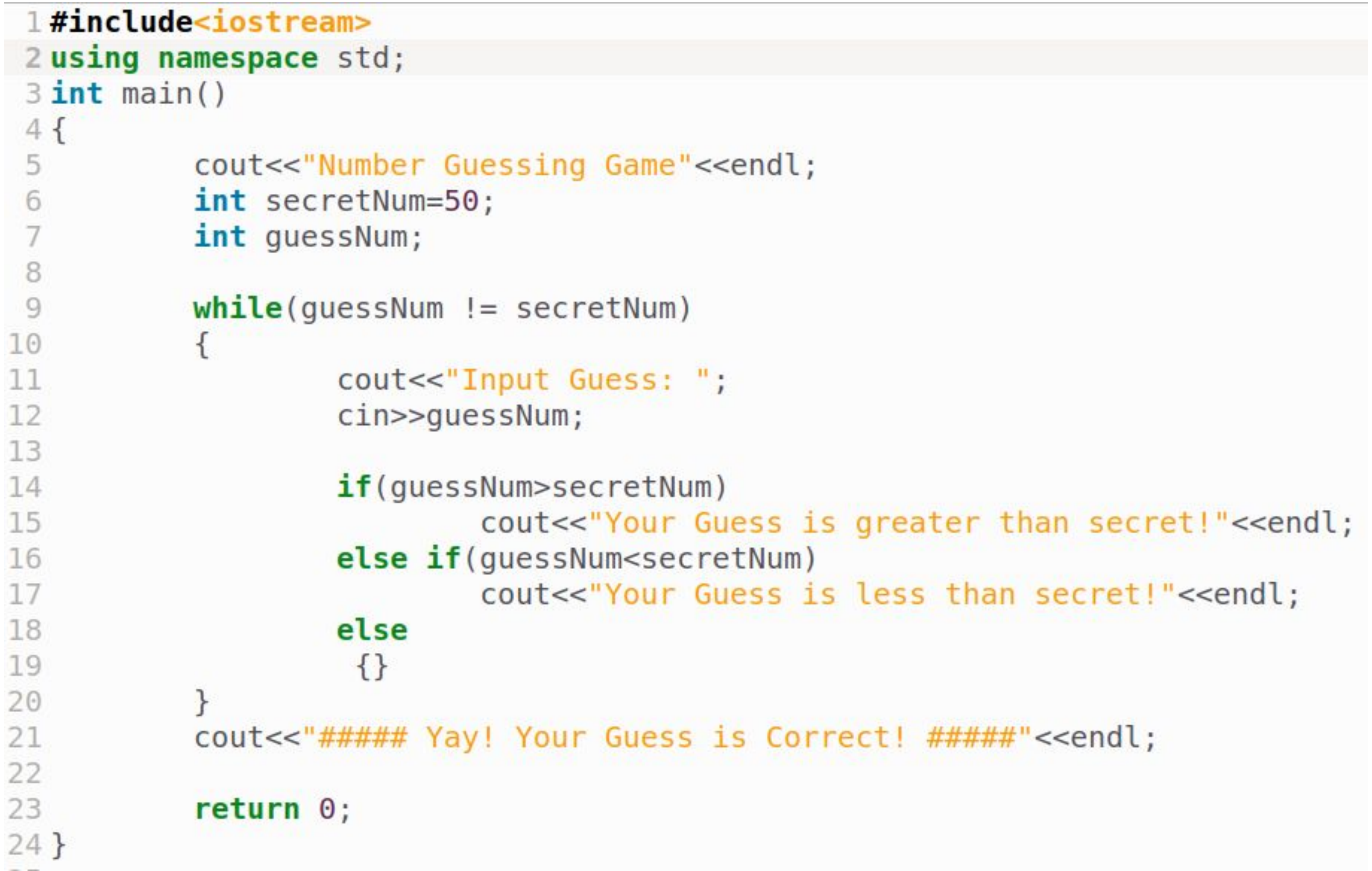
Basic Components of C++ Program

World of Opportunities…

- Computers are Universal …
- Billions of general purpose computers
- Billions more cell phones, smartphones & handheld devices
- Number of mobile internet users reached more than 4.32 billion in 2021…
- Sale of smartphones surpassed PC sales in 2015…
- In 2022, sale of smartphone users reached 1433.86 million – creating abundance of business and other professional opportunities…

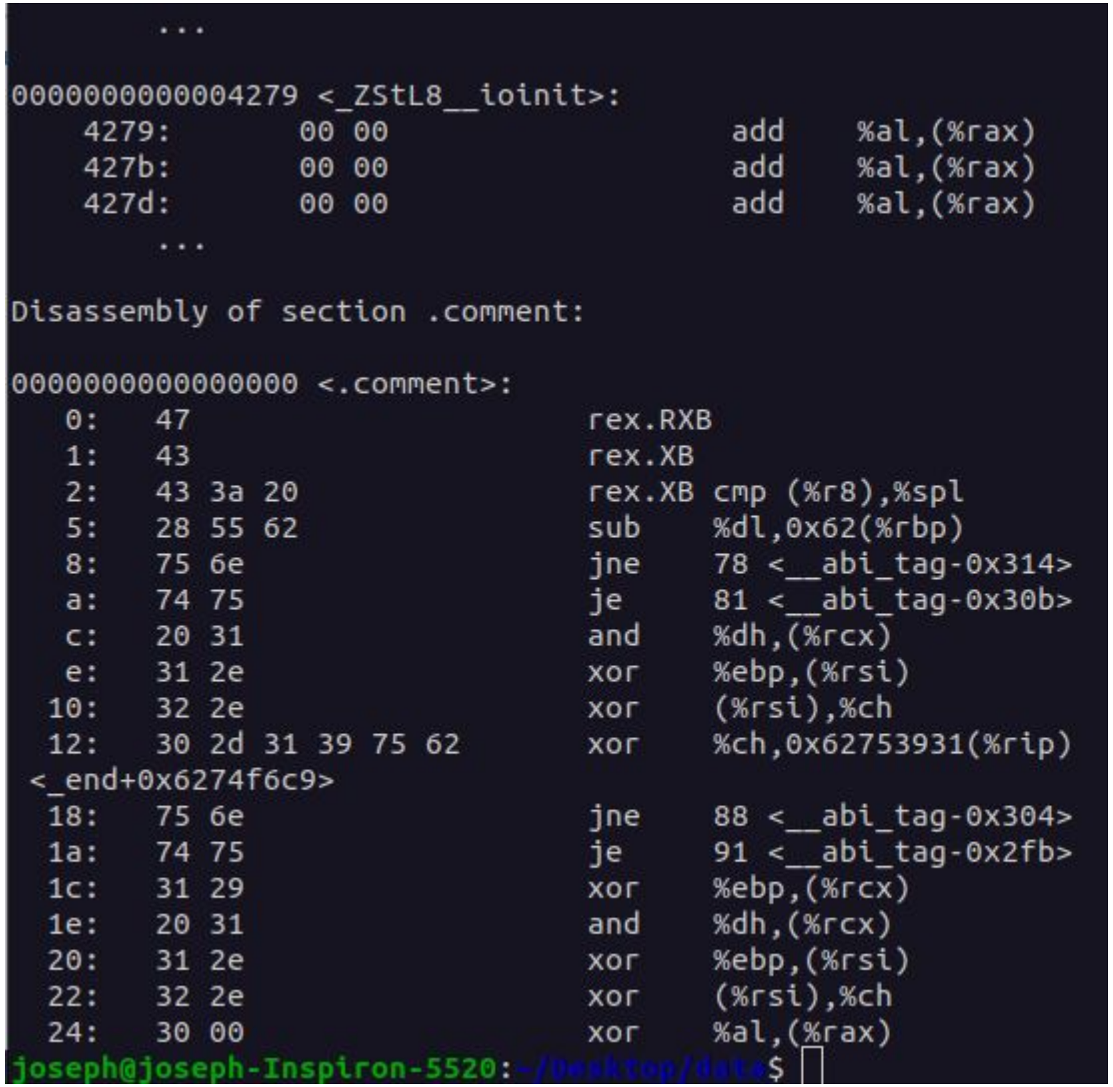
- https://www.statista.com/topics/779/mobile-internet/
- https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/

Number Guessing Game (C++)

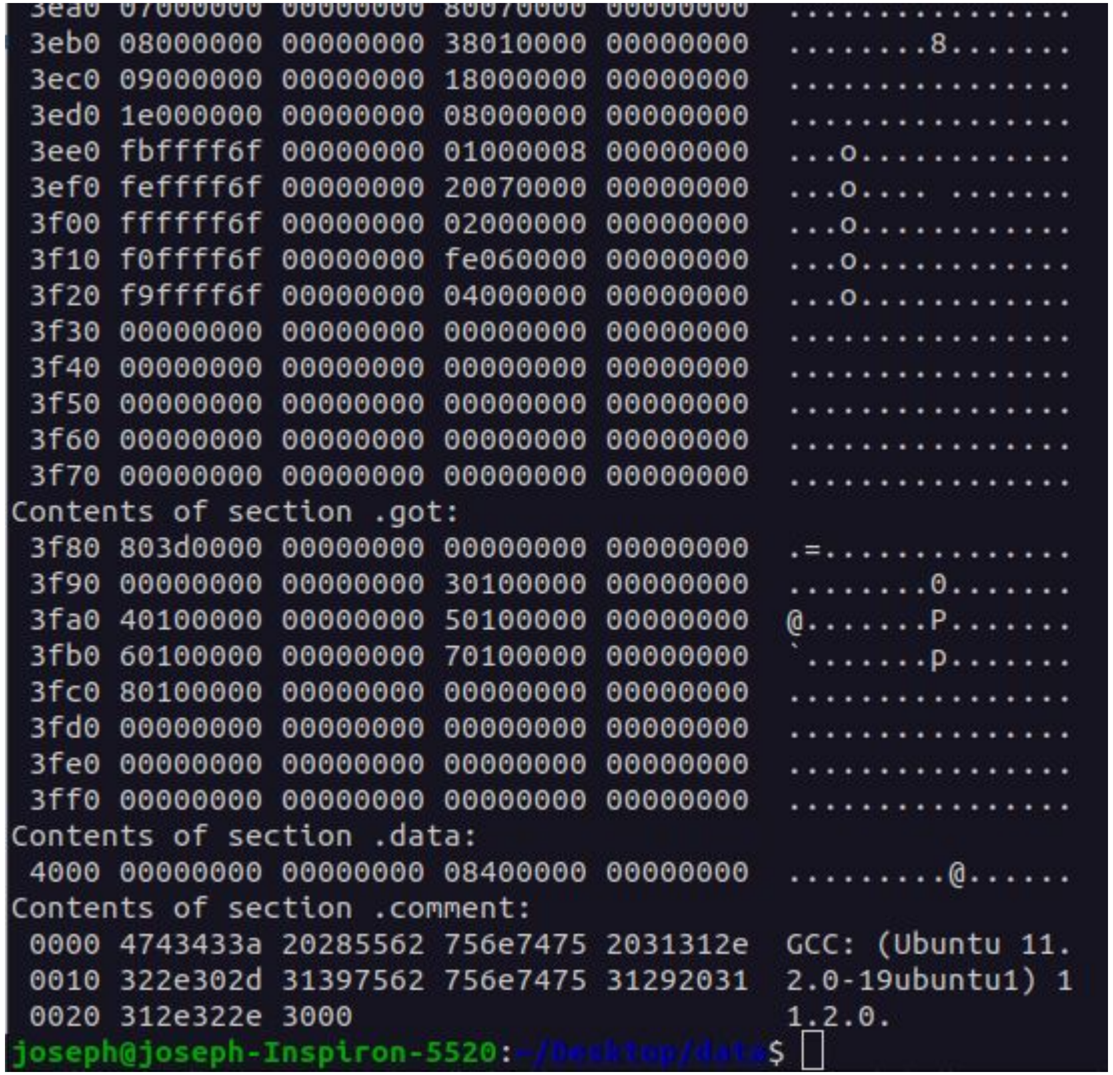


```
#include<iostream>
using namespace std;
int main()
{
        cout<<"Number Guessing Game"<<endl;
        int secretNum=50;
        int guessNum;

        while(guessNum != secretNum)
        {
                cout<<"Input Guess: ";
                cin>>guessNum;

                if(guessNum>secretNum)
                        cout<<"Your Guess is greater than secret!"<<endl;
                else if(guessNum<secretNum)
                        cout<<"Your Guess is less than secret!"<<endl;
                else
                 {}
        }
        cout<<"##### Yay! Your Guess is Correct! #####"<<endl;

        return 0;
}
```

Assembly Language vs Machine Code

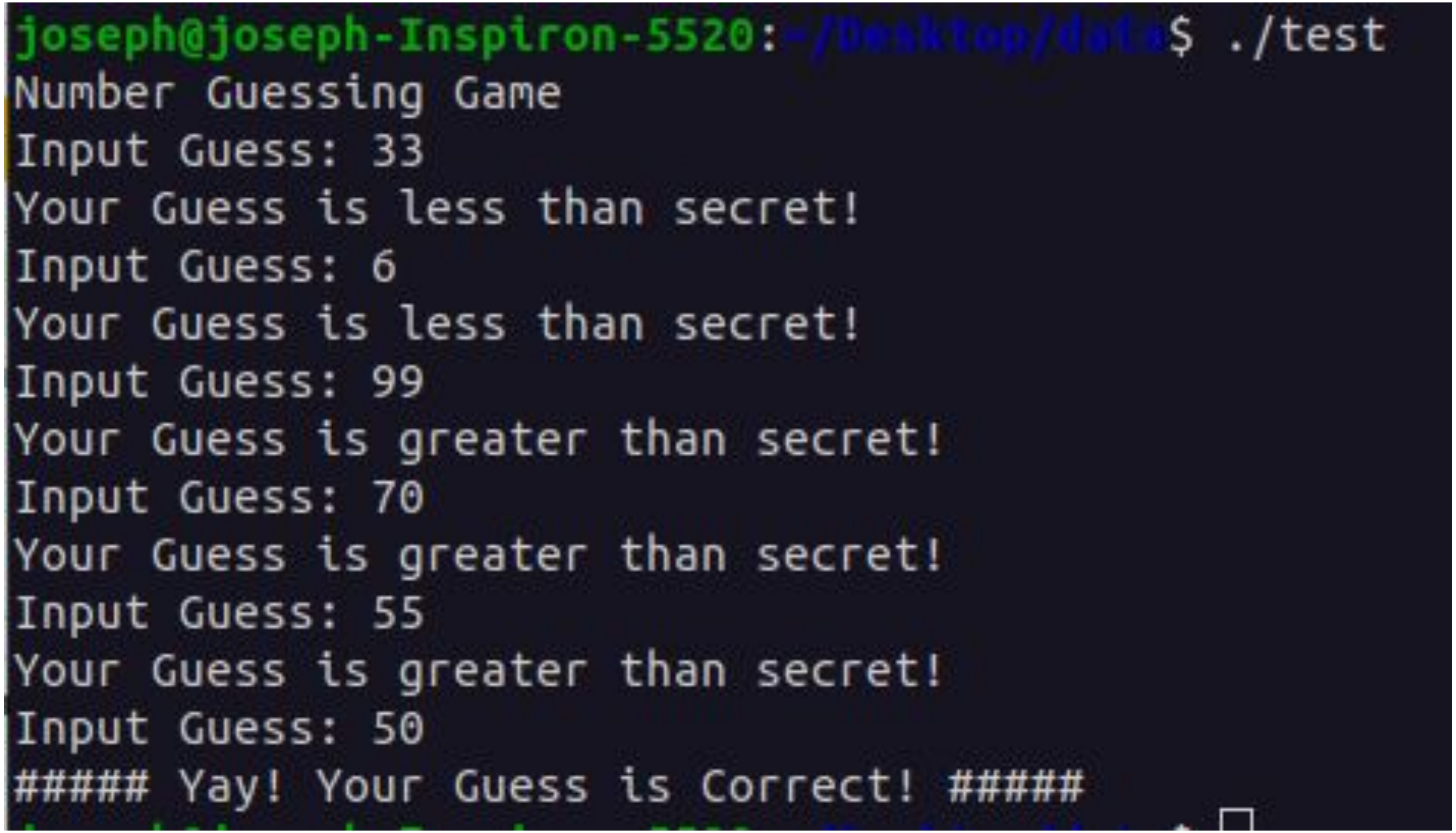


```
        ...

0000000000004279 <_ZStL8__ioinit>:
    4279:       00 00                   add    %al,(%rax)
    427b:       00 00                   add    %al,(%rax)
    427d:       00 00                   add    %al,(%rax)
        ...

Disassembly of section .comment:

0000000000000000 <.comment>:
   0:   47                      rex.RXB
   1:   43                      rex.XB
   2:   43 3a 20                rex.XB cmp (%r8),%spl
   5:   28 55 62                sub    %dl,0x62(%rbp)
   8:   75 6e                   jne    78 <__abi_tag-0x314>
   a:   74 75                   je     81 <__abi_tag-0x30b>
   c:   20 31                   and    %dh,(%rcx)
   e:   31 2e                   xor    %ebp,(%rsi)
  10:   32 2e                   xor    (%rsi),%ch
  12:   30 2d 31 39 75 62       xor    %ch,0x62753931(%rip)
 <_end+0x6274f6c9>
  18:   75 6e                   jne    88 <__abi_tag-0x304>
  1a:   74 75                   je     91 <__abi_tag-0x2fb>
  1c:   31 29                   xor    %ebp,(%rcx)
  1e:   20 31                   and    %dh,(%rcx)
  20:   31 2e                   xor    %ebp,(%rsi)
  22:   32 2e                   xor    (%rsi),%ch
  24:   30 00                   xor    %al,(%rax)
joseph@joseph-Inspiron-5520:~/Desktop/data$
```



```
 3ea0 07000000 00000000 80070000 00000000  ................
 3eb0 08000000 00000000 38010000 00000000  ........8.......
 3ec0 09000000 00000000 18000000 00000000  ................
 3ed0 1e000000 00000000 08000000 00000000  ................
 3ee0 fbffff6f 00000000 01000008 00000000  ...o............
 3ef0 feffff6f 00000000 20070000 00000000  ...o.... .......
 3f00 ffffff6f 00000000 02000000 00000000  ...o............
 3f10 f0ffff6f 00000000 fe060000 00000000  ...o............
 3f20 f9ffff6f 00000000 04000000 00000000  ...o............
 3f30 00000000 00000000 00000000 00000000  ................
 3f40 00000000 00000000 00000000 00000000  ................
 3f50 00000000 00000000 00000000 00000000  ................
 3f60 00000000 00000000 00000000 00000000  ................
 3f70 00000000 00000000 00000000 00000000  ................
Contents of section .got:
 3f80 803d0000 00000000 00000000 00000000  .=..............
 3f90 00000000 00000000 30100000 00000000  ........0.......
 3fa0 40100000 00000000 50100000 00000000  @.......P.......
 3fb0 60100000 00000000 70100000 00000000  `.......p.......
 3fc0 80100000 00000000 00000000 00000000  ................
 3fd0 00000000 00000000 00000000 00000000  ................
 3fe0 00000000 00000000 00000000 00000000  ................
 3ff0 00000000 00000000 00000000 00000000  ................
Contents of section .data:
 4000 00000000 00000000 08400000 00000000  .........@......
Contents of section .comment:
 0000 4743433a 20285562 756e7475 2031312e  GCC: (Ubuntu 11.
 0010 322e302d 31397562 756e7475 31292031  2.0-19ubuntu1) 1
 0020 312e322e 3000                        1.2.0.
joseph@joseph-Inspiron-5520:~/Desktop/data$
```

Number Guessing Game (Output)



```
joseph@joseph-Inspiron-5520:~/Desktop/data$ ./test
Number Guessing Game
Input Guess: 33
Your Guess is less than secret!
Input Guess: 6
Your Guess is less than secret!
Input Guess: 99
Your Guess is greater than secret!
Input Guess: 70
Your Guess is greater than secret!
Input Guess: 55
Your Guess is greater than secret!
Input Guess: 50
##### Yay! Your Guess is Correct! #####
```

The Evolution of Programming Languages

- Early computers were programmed in machine language
- To calculate wages = rates * hours in machine language:

```
100100 010001    //Load rates
100110 010010    //Multiply
100010 010011    //Store in wages
```

The Evolution of Programming Languages (cont'd.)

- Assembly language instructions are mnemonic
- **Assembler:** Translates a program written in assembly language into machine language

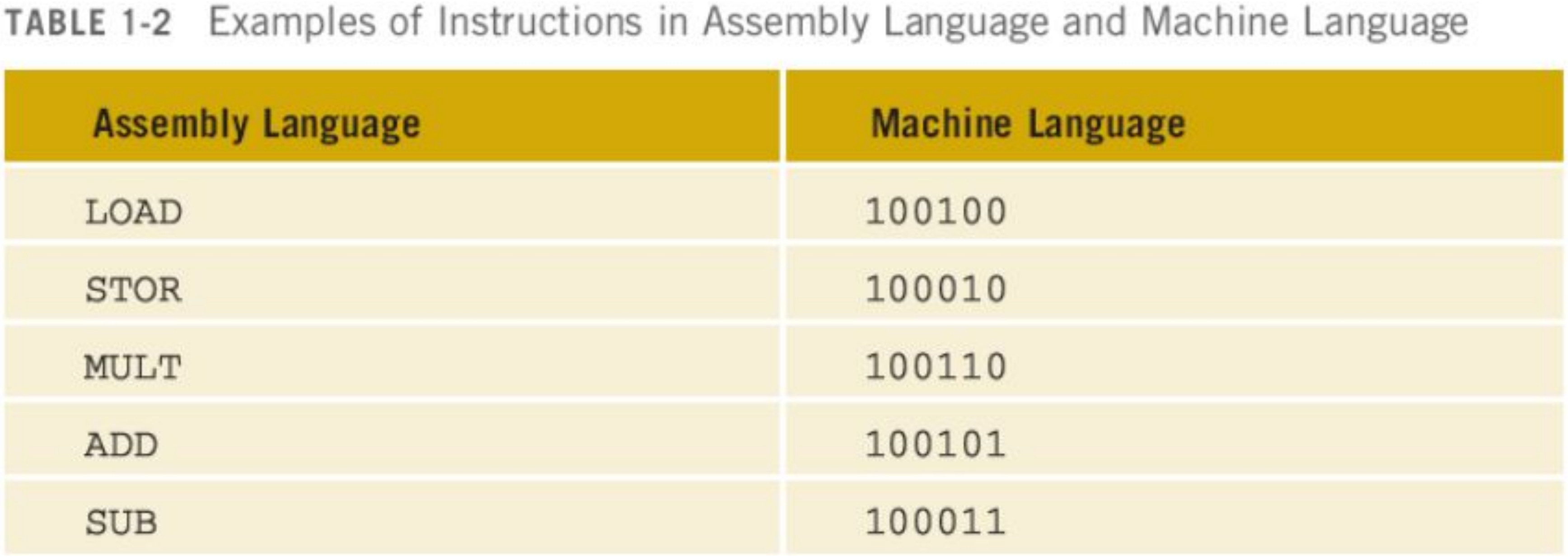


TABLE 1-2 Examples of Instructions in Assembly Language and Machine Language

Assembly Language	Machine Language
LOAD	100100
STOR	100010
MULT	100110
ADD	100101
SUB	100011

The Evolution of Programming Languages (cont'd.)

- Using assembly language instructions,

  wages = rates * hours can be written as:

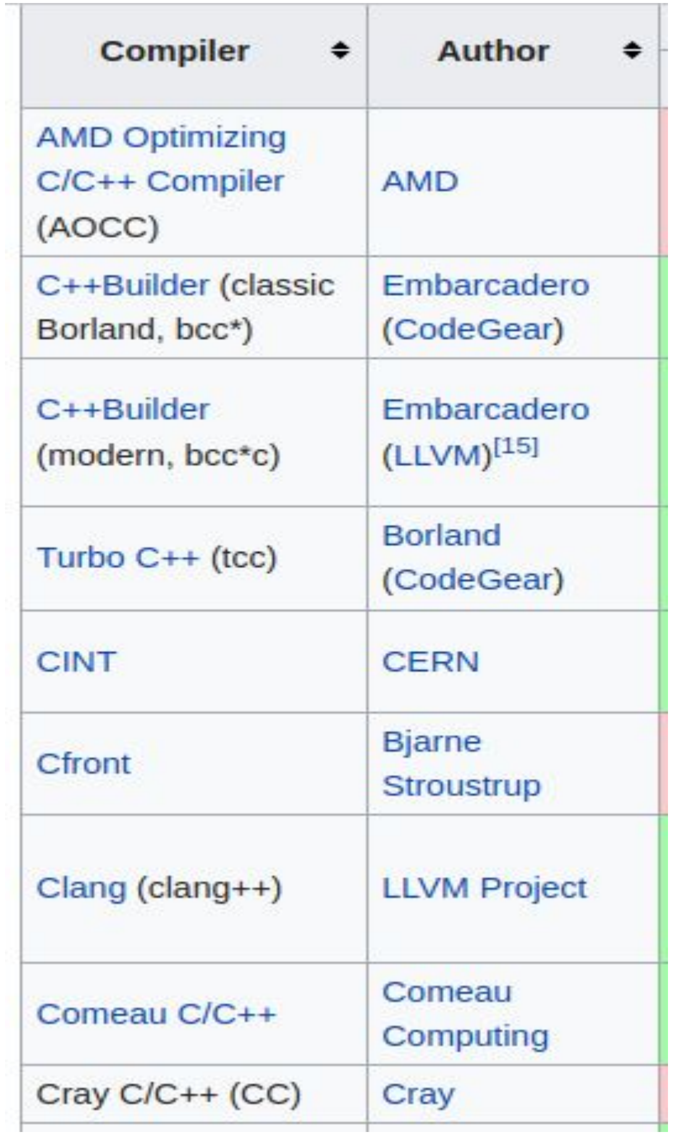
```
LOAD  rate
MULT hour
STOR  wages
```

The Evolution of Programming Languages (cont'd.)

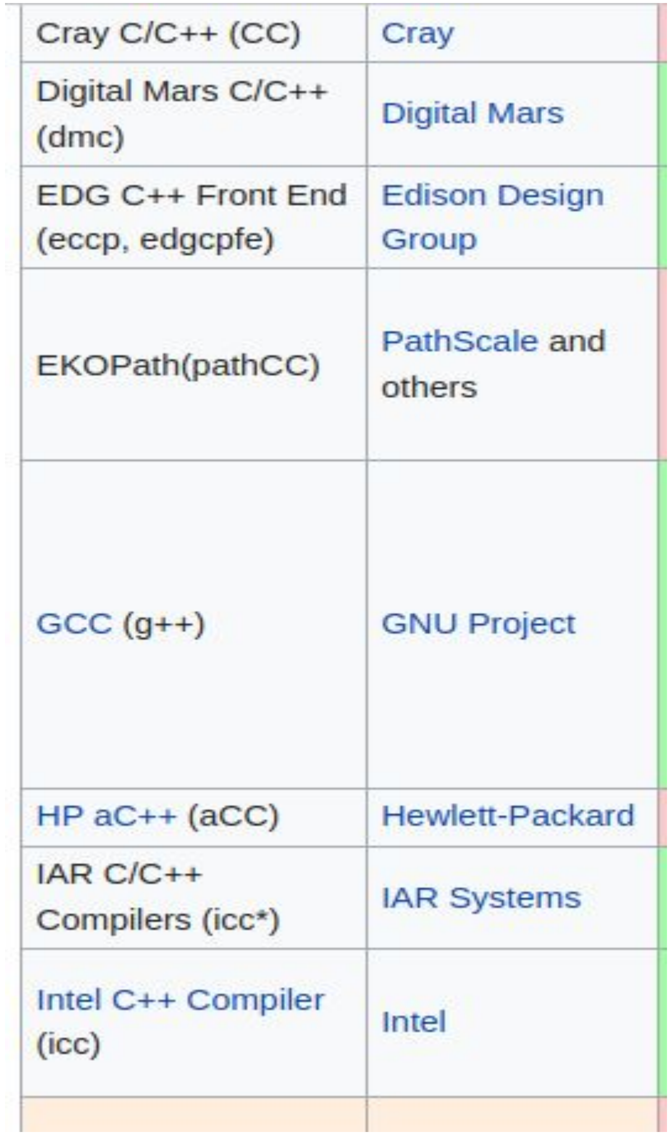
- High-level languages include Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java
- **Compiler:** Translates a program written in a high-level language to machine language
- The equation wages = rate • hours can be written in C++ as:

```
wages = rate * hours ;
```

C++ Compilers



Compiler	Author
AMD Optimizing C/C++ Compiler (AOCC)	AMD
C++Builder (classic Borland, bcc*)	Embarcadero (CodeGear)
C++Builder (modern, bcc*c)	Embarcadero (LLVM)[15]
Turbo C++ (tcc)	Borland (CodeGear)
CINT	CERN
Cfront	Bjarne Stroustrup
Clang (clang++)	LLVM Project
Comeau C/C++	Comeau Computing
Cray C/C++ (CC)	Cray



Cray C/C++ (CC)	Cray
Digital Mars C/C++ (dmc)	Digital Mars
EDG C++ Front End (eccp, edgcpfe)	Edison Design Group
EKOPath(pathCC)	PathScale and others
GCC (g++)	GNU Project
HP aC++ (aCC)	Hewlett-Packard
IAR C/C++ Compilers (icc*)	IAR Systems
Intel C++ Compiler (icc)	Intel

Assembly & Machine Language

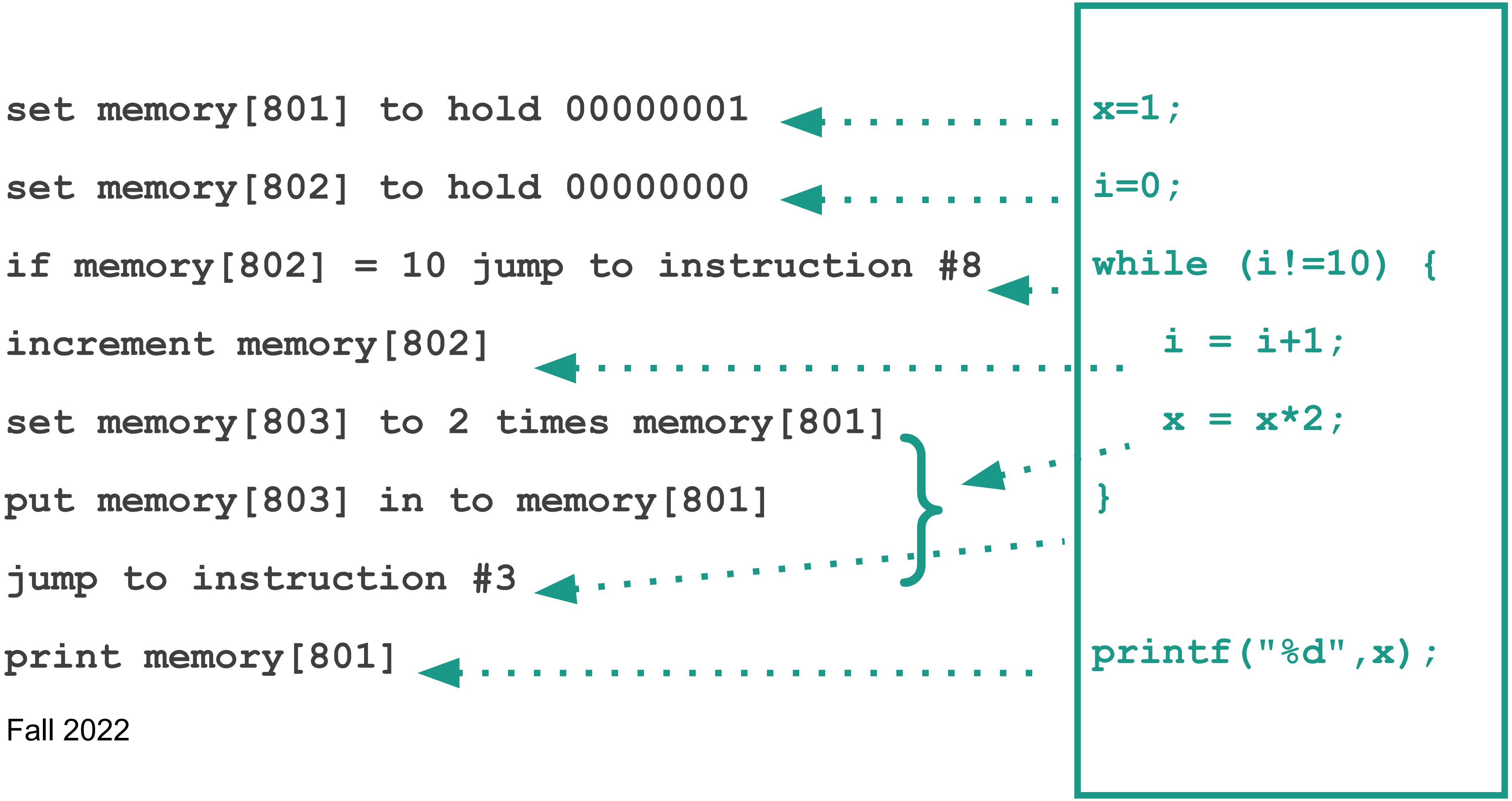
Assembly Language

```
        ST 1,[801]
        ST 0,[802]
TOP:    BEQ [802],10,BOT
        INCR [802]
        MUL [801],2,[803]
        ST [803],[801]
        JMP  TOP
BOT:    LD A,[801]
        CALL PRINT
```

Machine Language

```
00100101 11010011
00100100 11010100
10001010 01001001 11110000
01000100 01010100
01001000 10100111 10100011
11100101 10101011 00000010 00101001
11010101
11010100 10101000
10010001 01000100
```

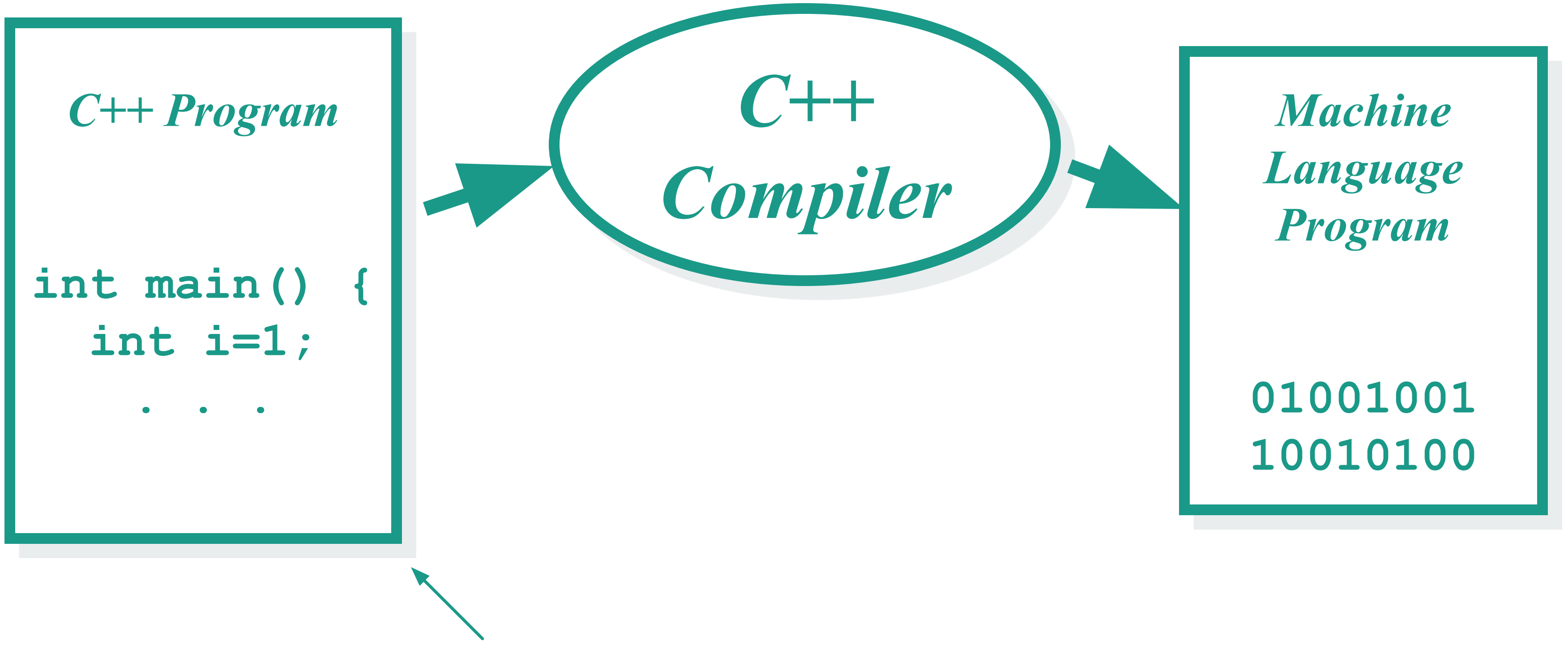
Equivalent C/C++ program



```
set memory[801] to hold 00000001
set memory[802] to hold 00000000
if memory[802] = 10 jump to instruction #8
increment memory[802]
set memory[803] to 2 times memory[801]
put memory[803] in to memory[801]
jump to instruction #3
print memory[801]
```

```
x=1;
i=0;
while (i!=10) {
    i = i+1;
    x = x*2;
}

printf("%d",x);
```

Compiler



Created with text editor or development environment

Processing a C++ Program

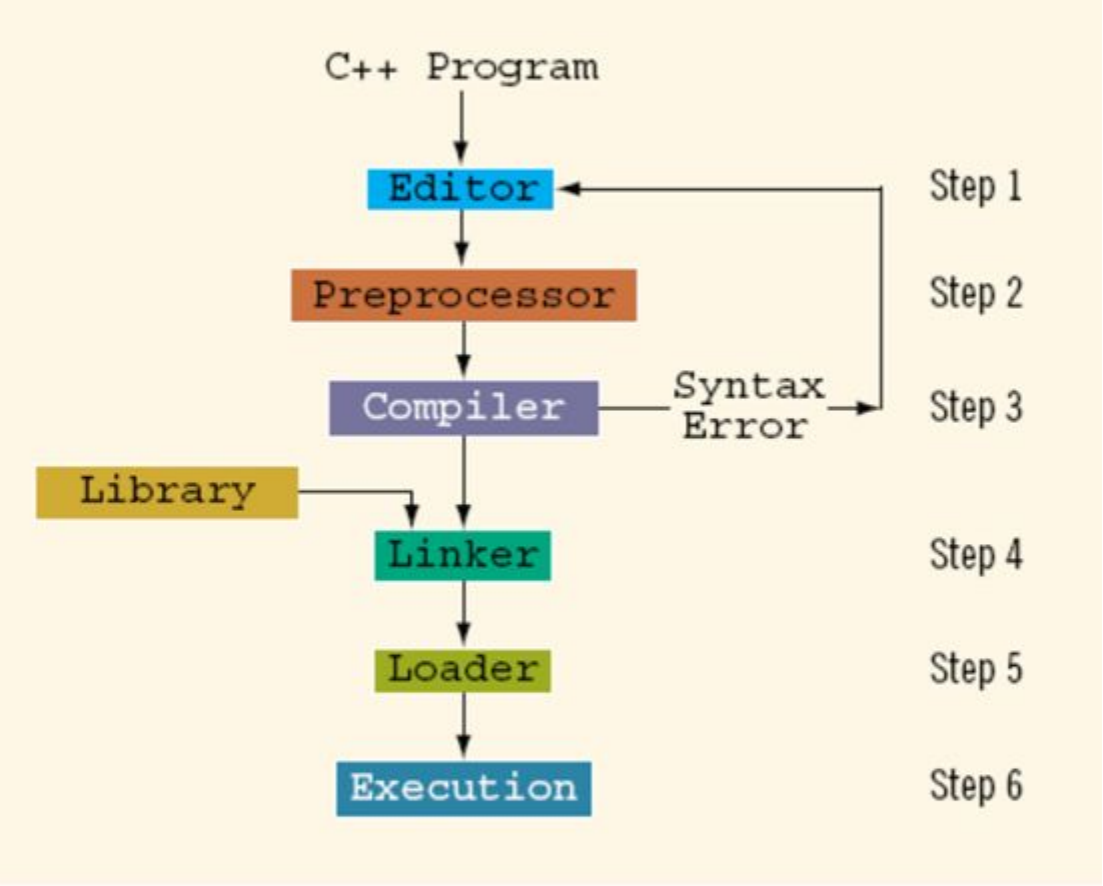
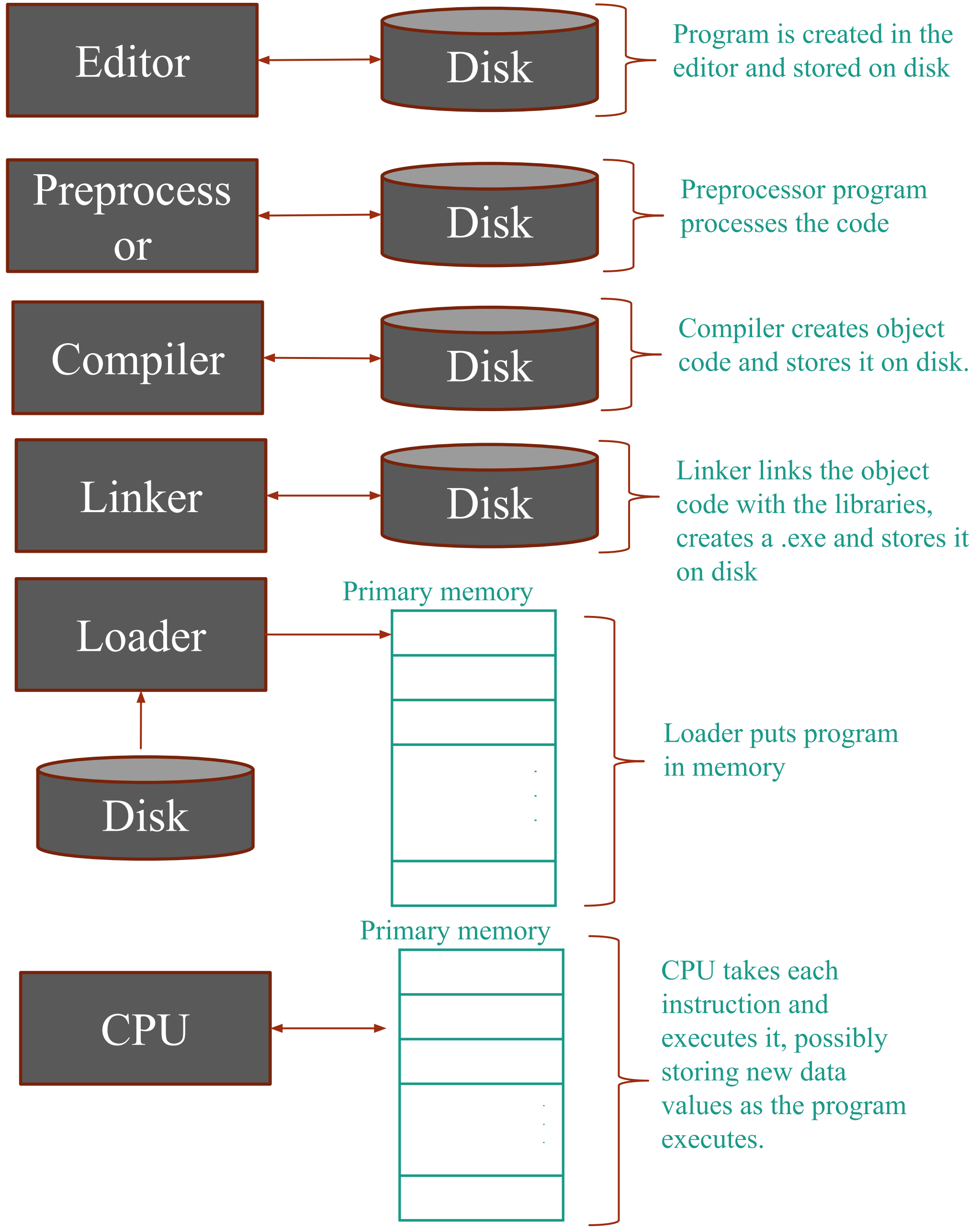


FIGURE 1-3 Processing a C++ program

Basics of a Typical C++ Environment

Phases of C++ Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



Program is created in the editor and stored on disk

Preprocessor program processes the code

Compiler creates object code and stores it on disk.

Linker links the object code with the libraries, creates a .exe and stores it on disk

Loader puts program in memory

CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Compilers

- Translate high-level language to machine language
- Check that the program obeys the rules

- **Source code**
  - The original program in a high level language
- **Object code**
  - The translated version in machine language

Linkers

- Some programs we use are already compiled
  - Their object code is available for us to use
  - **For example:** Input and output routines
- A **Linker** combines
  - The object code for the programs we write

    and

  - The object code for the pre-compiled routines (**of SDK**)

    into

  - The machine language program the CPU can run
- **Loader:**
  - Loads executable program into main memory
- The last step is to execute the program

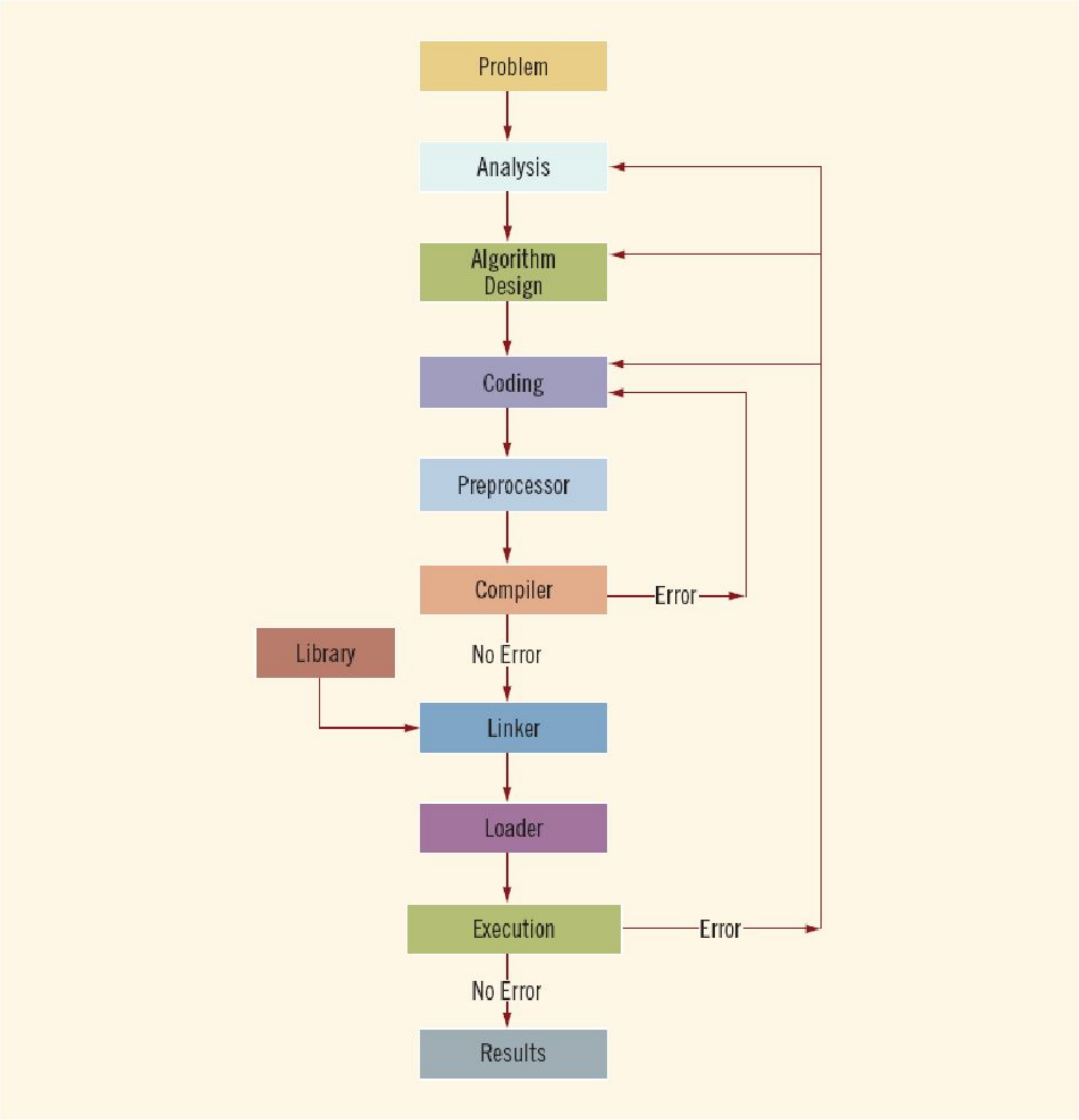


FIGURE 1-4 Problem analysis–coding–execution cycle

History of C and C++

- **History of C**
  - Evolved from two other programming languages
  - BCPL and B: “Typeless” languages
  - **Dennis Ritchie (Bell Lab):** Added typing, other features
- C is a programming language developed in the 1970's alongside the UNIX operating system
- C provides a comprehensive set of features for handling a wide variety of applications, such as systems development and scientific computation
  - 1989: ANSI standard/ ANSI/ISO 9899: 1990

History of C and C++

- **History of C++**
    - Early 1980s: **Bjarne Stroustrup** (Bell Lab)
    - Provides capabilities for object-oriented programming
        - **Objects:** reusable software components
        - Object-oriented programs
- "**Building block** approach" to creating programs
    - C++ programs are built from pieces called classes and functions
    - **C++ standard library:** Rich collections of existing classes and functions

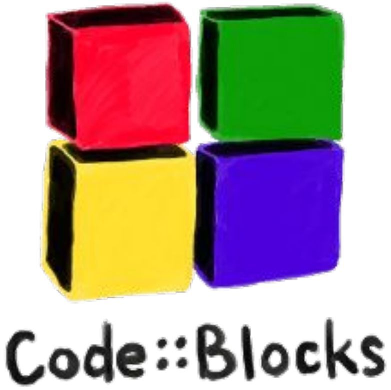
Structured/OO Programming

- **Structured programming (1960s)**
  - Disciplined approach to writing programs
  - Clear, easy to test and debug, and easy to modify
  - e.g. Pascal: 1971: Niklaus Wirth
- **OOP**
  - “Software reuse”
  - “Modularity”
  - “Extensible”
  - More understandable, better organized and easier to maintain than procedural programming

Basics of a Typical C++ Environment

- **C++ systems**
  - Program-development environment
    - Integrated Development Environment (IDE)
  - Language
  - C++ Standard Library
- **C++ program names extensions**
  - .cpp (C Plus Plus)
  - .c (C)

IDE’s



The C++ Standard Library

C/C++ programs consist of pieces/modules called functions

- A programmer can create his own functions
  - **Advantage:** the programmer knows exactly how it works
  - **Disadvantage:** time consuming
- Programmers will often use the C/C++ library functions
  - Use these as building blocks
- Avoid re-inventing the wheel
  - If a pre-made function exists, generally best to use it rather than write your own
  - Library functions carefully written, efficient, and portable

Programming Style

C++ is a free-format language, which means that:

- Extra blanks (spaces) or tabs before or after identifiers/operators are ignored
- Blank lines are ignored by the compiler just like comments
- Code can be indented in any way
- There can be more than one statement on a single line
- A single statement can continue over several lines

Programming Style (cont.)

In order to improve the readability of your program, use the following conventions:

- Start the program with a **header** that tells what the program does
- Use **meaningful variable names and Camel notation**
- **Document** each variable declaration with a comment telling what the variable is used for
- Place each **executable statement** on a **single line**
- A segment of code is a sequence of executable statements that belong together
  - Use blank lines to separate different segments of code
  - Document each segment of code with a comment telling what the segment does.

C++ keywords

- Keywords appear in **blue** in Visual C++
- Each keyword has a predefined purpose in the language
- Do not use keywords as variable and constant names!!

- We shall cover most of the following keywords in this class:

bool, break, case, char, const, continue, do, default, double, else, extern, false, float, for, if, int, long, namespace, return, short, static, struct, switch, typedef, true, unsigned, void, while

Structure of a C++ Program

A C++ program is a collection of definitions and declarations:

- data type definitions
- global data declarations
- function definitions (subroutines)
- class definitions
- a special function called
  - **main()** (where the action starts)

General form of a C++ program

```
// Program description
#include directives
global declarations
int main()
{
     constant declarations
     variable declarations
     executable statements
     return 0;
}
```

Why int main () …?

- **void** means that the function has no return-value, so you don't need return
- **int** -- which is also implied for functions with no stated return-type -- means that the function should return an integer, so you must use return
- The return-value of **main()** is also the return-value for the program, and it's value is often used to tell the operating-system if the program ran successfully or not. Typically a return-value of **0 (zero)** tells that the program ran **without a problem**, while various **non-zero values** indicates **different problems** (**e.g. 1=couldn't read a file, 2=bad arguments, 3=overflow**). Some OSs define which values should be used with what type of errors, others let's the programmer choose what he'd like for his program
- You should use a **main()** of type **int**, and return a **value (zero)** at the **end of your program**. The revised language-definition for C++ makes **int** the only legal type for **main()** (many compilers will at least warn you otherwise)... For C, void as return-type for main() has never really been in use... So use **"int main()"** and **"return 0"**.

- When you use void main(), you're making a mistake. Please don't do it
- C programming language: ISO 9899 paragraph 5.1.2.2.1.1 Program startup
- The function called at program startup is named main. It shall be defined with a return type of int and with no parameters: int main(void) { /* ... */ } or with two parameters int main(int argc, char *argv[]) { /* ... */ } or equivalent.
- C++ programming language : ISO 14882 paragraph 3.6.1 Main function
- A program shall contain a global function called main, which is the designated start of the program [...] This function shall not be overloaded. It shall have a return type of type int, but otherwise its type is implementation-defined. All implementations shall allow both of the following definitions of main: int main() { /* ... */ } and int main(int argc, char* argv[]) { /* ... */ }

Includes

- The statement: **#include <iostream>** inserts the contents of the file **iostream** inside your file before the compiler starts
- Definitions that allow your program to use the functions and classes that make up the standard C++ library are in these files.
- You can include your own file(s):

```
#include "myfile.h"
```

C++ compiler directives

- Compiler directives appear in **blue** in Visual C++
- The **#include** directive tells the compiler to include some already existing C++ code in your program
- The included file is then linked with the program
- There are two forms of #include statements:

```
#include <iostream> //for pre-defined files
#include "my_lib.h"   //for user-defined files
```

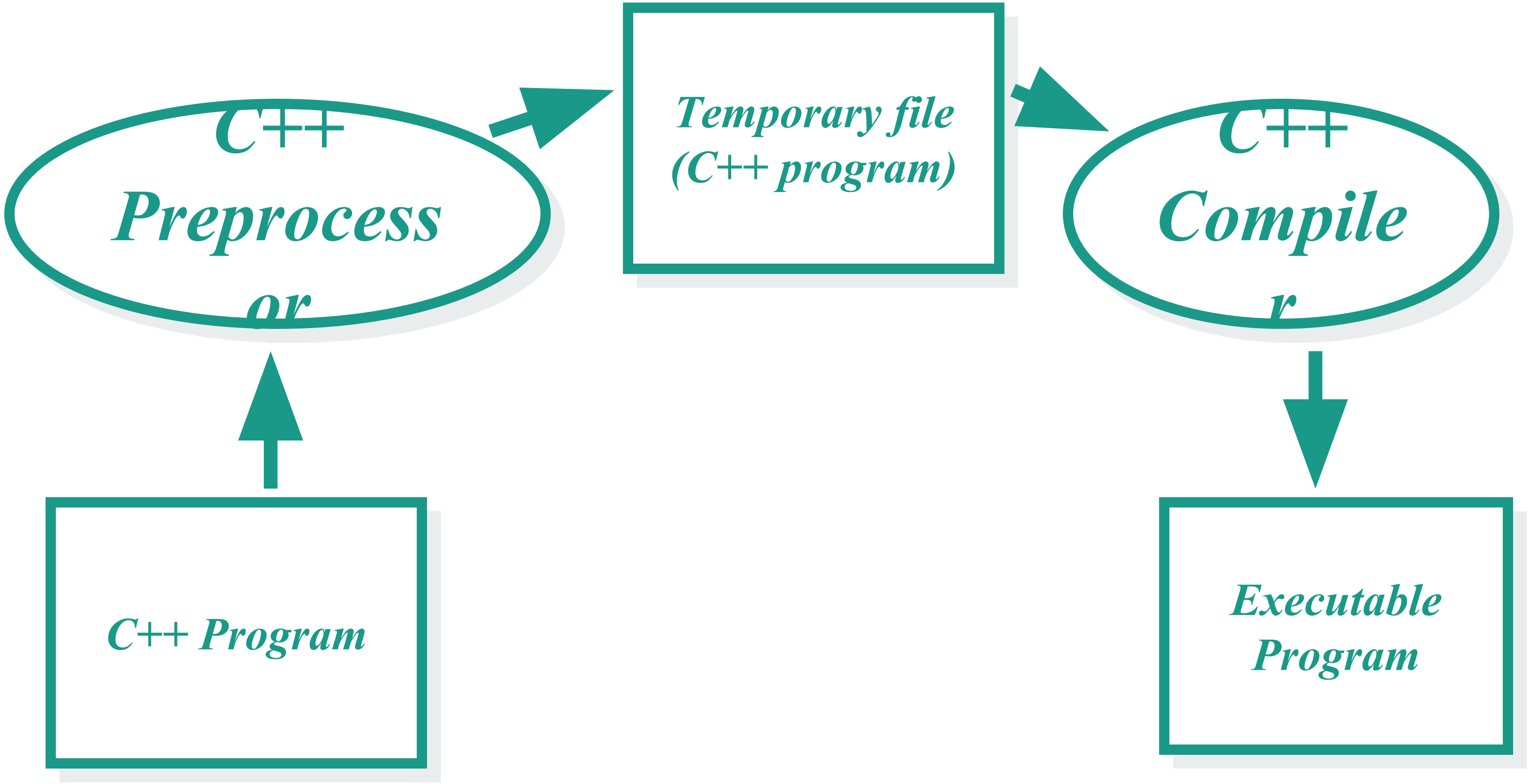
The Preprocessor

- Lines that start with the character **'#'** are special instructions to a preprocessor
- The preprocessor can replace the line with something else:
  - **include:** replaced with contents of a file
- Other *directives* tell the preprocessor to look for patterns in the program and do some fancy processing

C++ Preprocessor

- C++ Compilers **automatically invoke a preprocessor** that takes care of #include statements and some other special directives
- Definitions that **allow your program** to use the functions and classes that make up the standard C++ library are in these files
- You don't need to do anything special to run the preprocessor - it happens automatically

Preprocessing



Preprocessor Directives

- Preprocessor directives: Begin with #

- Processed before compiling

- #include
- #define

Some common include statements

- Basic I/O: iostream.h
  - Provides functionality of input and output
- I/O manipulation: iomanip.h
  - Format’s the input and output
- Standard Library: stdlib.h
  - Functions for memory allocation, process control, conversion etc.
- Time and Date support: time.h
  - Functionality of time manipulation
- Mathematics support: math.h
  - Functionality of basic mathematical functions

Questions