# COMPILATION PROCESS IN C PROGRAM

Faculty of Information Technology
University of Moratuwa

# What is Compilation?

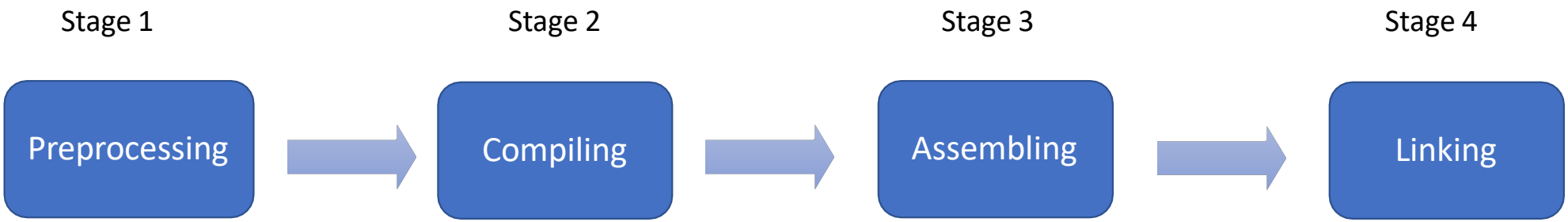
```c
#include <stdio.h>

int main() {
    printf("Hello world");
    return 0;
}
```

```
11010101010101001010001
10010111100001000100101
01010101010100101110011
01000101001010111000100
01010101000011111010101
01010101010101001000011
```

Source code

Machine code

# 4 stages of compilation process



# Stage 1 - Preprocessing

- Preprocessor take care of this stage.
- Preprocessor goes through the entire source code and :
  - ✓ Remove comments
  - ✓ Expand macros
  - ✓ Expand included files
- Source code get expanded.
- Command : gcc hello.c -E -o hello.i
- Output file : hello.i

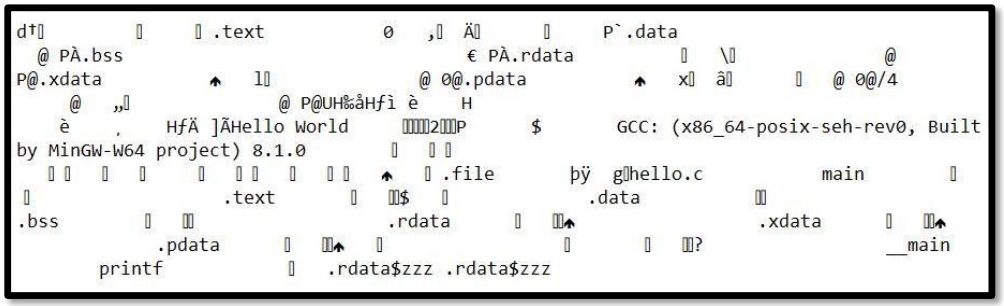
# Stage 2 - Compiling

- C Compiler is the tool used in here.
- Checks for valid syntax of C language.
- If any wrong syntax is found throws an error.
- Converts the pre-processed code into assembly code.
- Command : gcc hello.i -S -o hello.s
- Output file : hello.s

```
        .file   "hello.c"
        .text
        .def    __main; .scl    2;      .type   32;     .endef
        .section .rdata,"dr"
.LC0:
        .ascii "Hello World\0"
        .text
        .globl  main
        .def    main;   .scl    2;      .type   32;     .endef
        .seh_proc       main
main:
        pushq   %rbp
        .seh_pushreg    %rbp
        movq    %rsp, %rbp
        .seh_setframe   %rbp, 0
        subq    $32, %rsp
        .seh_stackalloc 32
        .seh_endprologue
        call    __main
        leaq    .LC0(%rip), %rcx
        call    printf
        movl    $0, %eax
        addq    $32, %rsp
        popq    %rbp
        ret
        .seh_endproc
        .ident  "GCC: (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0"
        .def    printf; .scl    2;      .type   32;     .endef
```

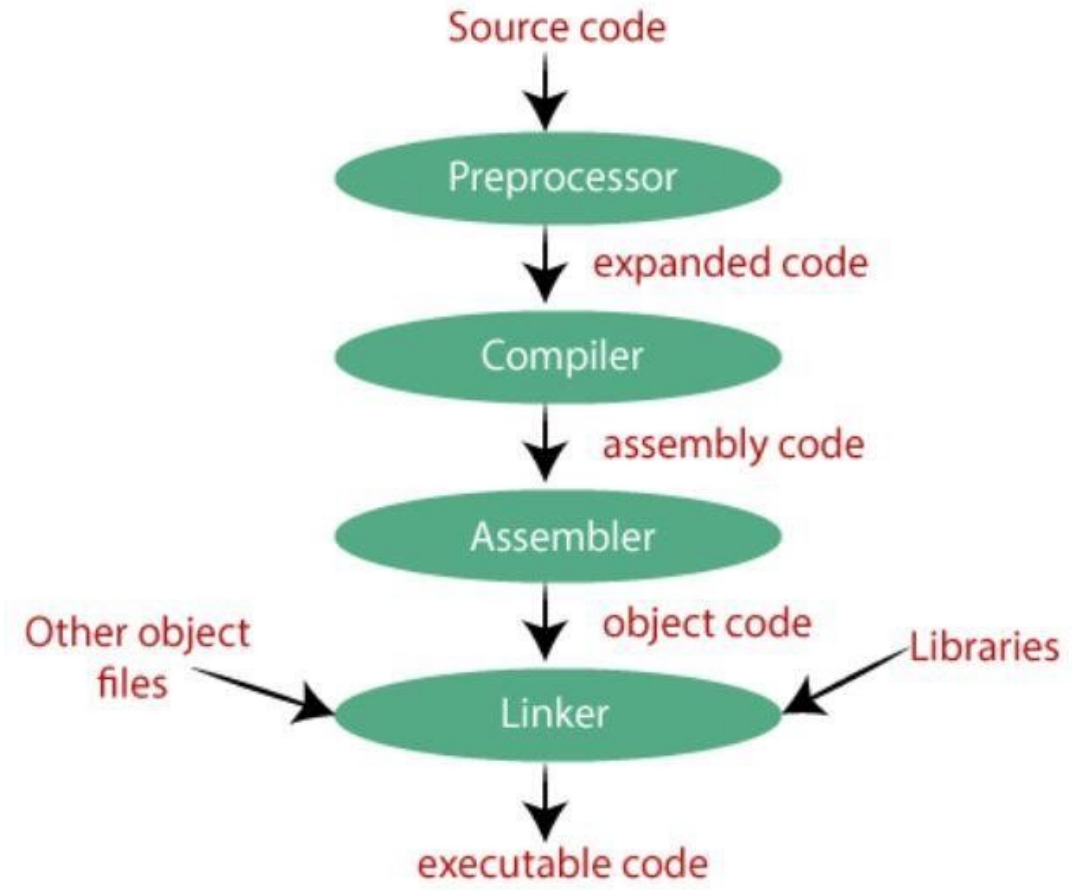
# Stage 3 - Assembling

- Assembler take care of this stage.
- Convert the assembly code into pure binary code or machine code.
- Also, the output code is known as the object code.
- Command : gcc hello.s -c -o hello.o
- Output file : hello.o



# Stage 4 - Linking

- Linking is the final step of compilation.
- The linker merges all the object code from multiple modules into a single one.
- Produces final executable file.
- Command : gcc  hello.c -o  hello.exe
- Output file : hello.exe


Source code
Preprocessor
expanded code
Compiler
assembly code
Assembler
object code
Other object files
Libraries
Linker
executable code

## Does the executable file depend on OS?

Refer the link : https://www.quora.com/Are-executable-files-exe-generated-by-compilers-machine-specific-and-or-operating-system-specific-In-other-words-can-exe-files-be-ported-from-one-machine-to-another-having-the-same-or-different-OS

## Does assembly languages depend on OS?

Refer the link : https://stackoverflow.com/questions/6859348/how-do-assembly-languages-depend-on-operating-systems

## Are compilers only operating system dependent and not hardware dependent?

Refer the link : https://www.quora.com/Why-are-compilers-only-operating-system-dependent-and-not-hardware-dependent

## Do all programming languages compile to the same machine code?

Refer the link : https://www.quora.com/Do-all-programming-languages-compile-to-the-same-machine-code