Compiler and Toolchain

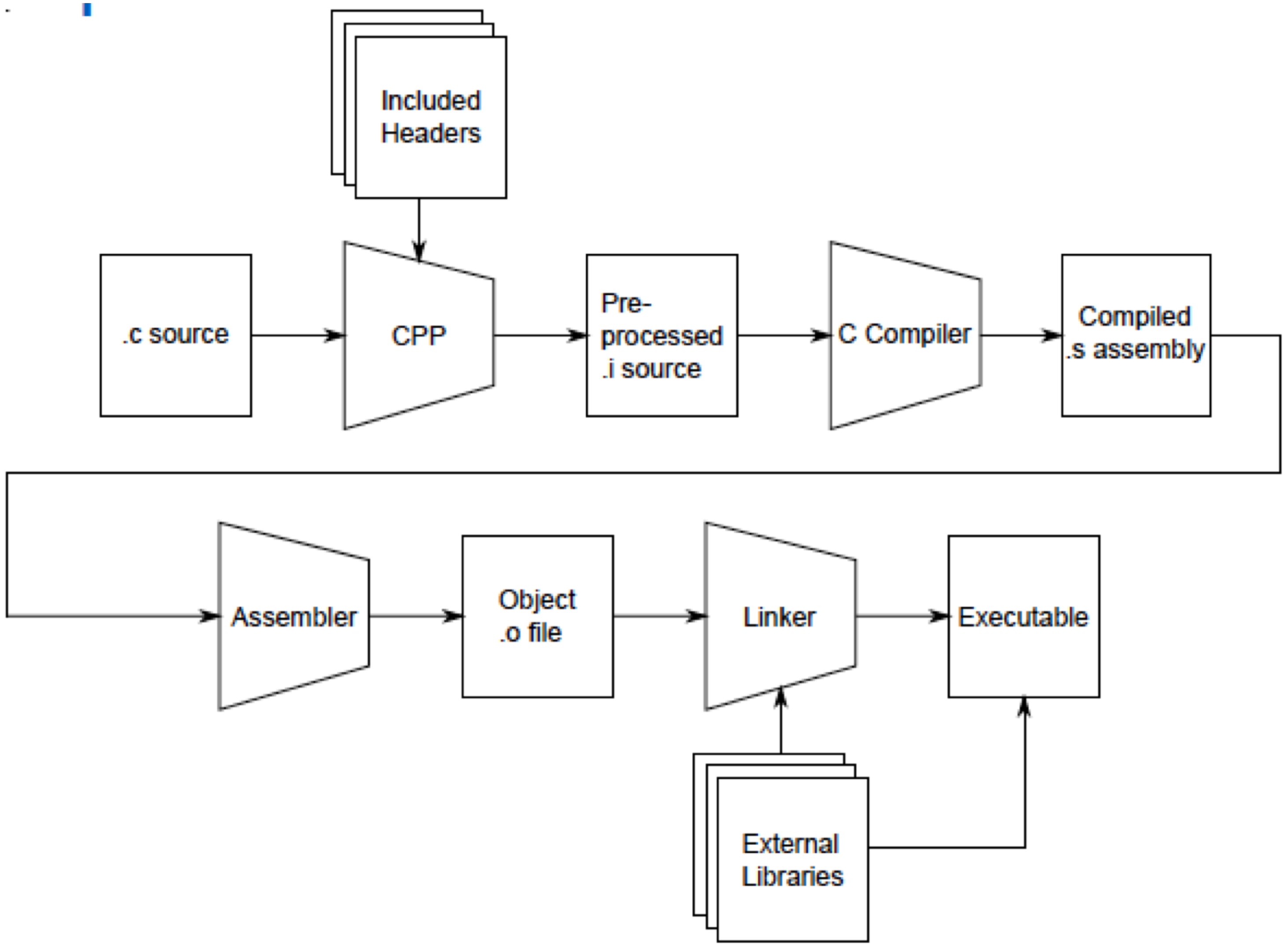
Karthik Dantu
Ethan Blanton
Computer Science and Engineering
University at Buffalo
kdantu@buffalo.edu

Portions of this lecture are from the Princeton COS 217 course slides

`gcc` – GNU Compiler Collection

- C compiler as we know it is actually many tools
- This is because
  - `gcc` history
  - Common compiler design
  - Specific design goal of compilation in parts
- We actually invoke the compiler driver
- Compiler is only a single step of a multi-step process

Complete Toolchain



Compiler Toolchain - Example

- C source
- Pre-processor
  Expanded C source
- C Compiler
  Assembly source
- Assembler
  Object code
- Linker
  Executable binary

```
#include <stdio.h>
#define NUM 42

Int main() {
  int a=NUM;
  printf("Hello,
world\n");
  return 0;
}
```

C Preprocessor

- Performs source-code transformations before compiling
- Does not understand C – can be used for other things
- Three main functions
  - Apply pre-processor directives
  - Replace all macros with actual values/code
  - Remove all comments

C Preprocessor Directives

- Primary task is to apply pre-processor directives
- Directives begin with #
- `#include`: insert another file
- `#define`: Define a symbol or a macro
- `#ifdef/#endif`: Include the enclosed block only if a symbol is defined
- `#if/#endif`: Include only if a condition is true
- Preprocessor directives DO NOT end with a semicolon

Defining Symbols and Macros

- `#define` directive defines a symbol or macro

```
#define PI 3.14159
#define PLUSONE(x) (x + 1)
PLUSONE(PI) /* Becomes (3.14159 + 1) */
```

- Macros are expanded, not calculated
- Expansion given to the next stage in compilation

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Preprocessing – Conditional Compilation

- Various `#if` directives control conditional compilation

```
#ifdef ARGUMENT
/* This code will be included only if ARGUMENT is a
symbol defined by the preprocessor – regardless of its
expansion */
#endif
```

- The `#ifndef` directive requires ARGUMENT to be undefined
- The `#if` directive requires ARGUMENT to evaluate to True

The C Compiler

- Transforms C source into machine-dependent assembly code
- Produces an object file via the assembler
- Only part of the toolchain that understands C
- It understands
  - Semantics of C
  - Capabilities of the target machine
- It uses these things to transform C into assembly

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Assembly Language

- Assembly language is machine-specific, but human readable
- Assembly language contains
  - Descriptions of machine instructions
  - Descriptions of data
  - Address labels marking variables and functions (symbols)
  - Metadata about the code and compiler transformations
- All of the semantics of C are in assembly
- Structure of assembly may be very different

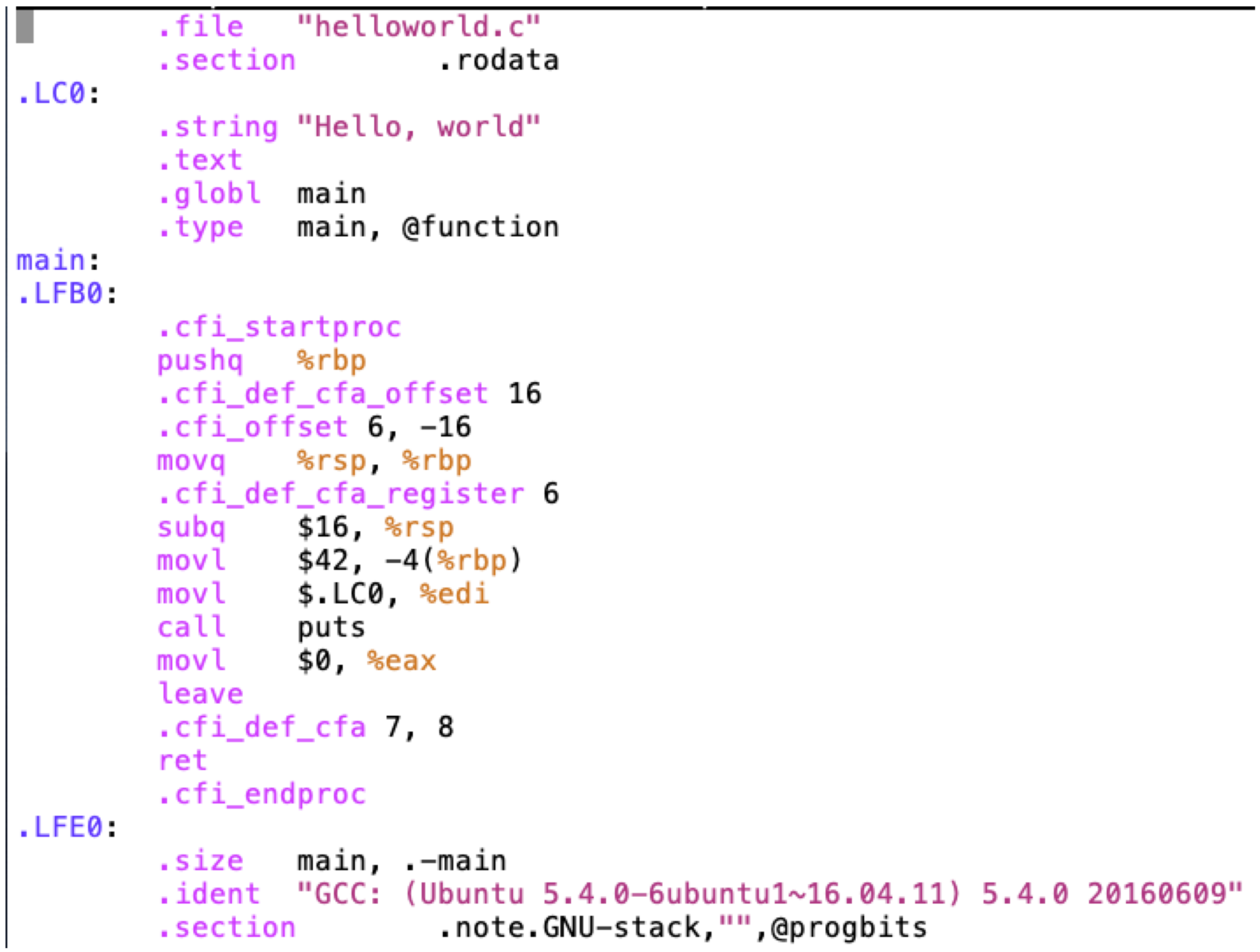
Compiling to Assembly

- We can compile to assembly using –S option in `gcc`

```
$ gcc –S helloworld.c
```

- This produces a file called `helloworld.s`

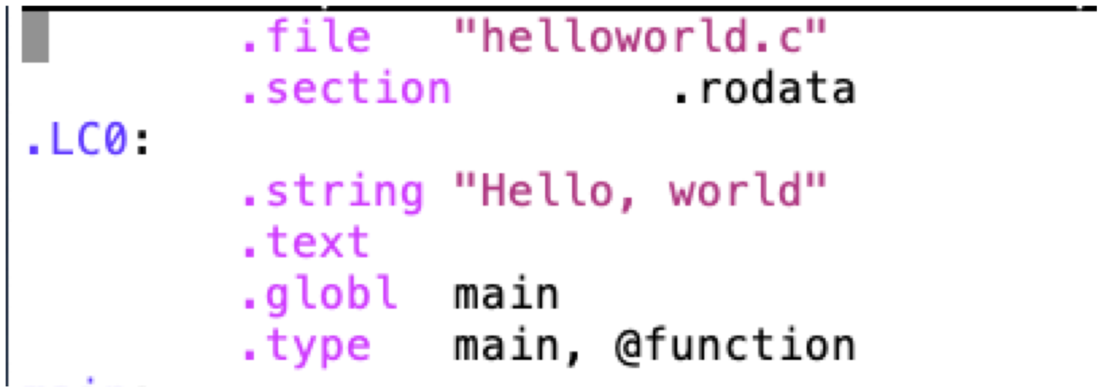
Helloworld.s



```
        .file   "helloworld.c"
        .section        .rodata
.LC0:
        .string "Hello, world"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $42, -4(%rbp)
        movl    $.LC0, %edi
        call    puts
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609"
        .section        .note.GNU-stack,"",@progbits
```

Helloworld.s - I

- `.LC0`: local label
- `.string` declares string constant
- `.globl` and `.type` directives declare that we're defining a global function named `main`



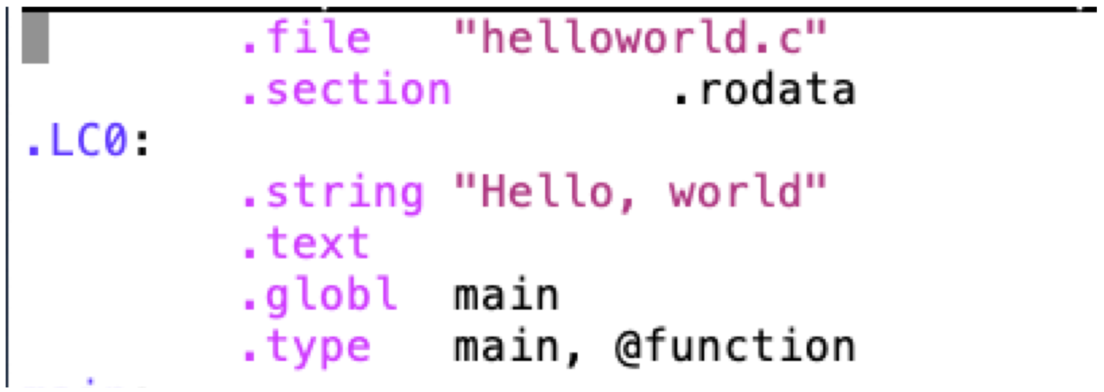
```
        .file   "helloworld.c"
        .section        .rodata
.LC0:
        .string "Hello, world"
        .text
        .globl  main
        .type   main, @function
```

Karthik Dantu

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Helloworld.s - I

- `.LC0`: local label
- `.string` declares string constant
- `.globl` and `.type` directives declare that we're defining a global function named `main`



```
        .file   "helloworld.c"
        .section        .rodata
.LC0:
        .string "Hello, world"
        .text
        .globl  main
        .type   main, @function
```

Karthik Dantu