
Compiling C Code

HW4 due Tuesday 10/29

Philipp Koehn

HW5 due Friday 11/8

~~13 April 2018~~

28 Oct 2019



- Source Code

```
#include <stdlib.h>
#include <stdio.h>

int main(void) {
    printf("Hello world!\n");
    return EXIT_SUCCESS;
}
```

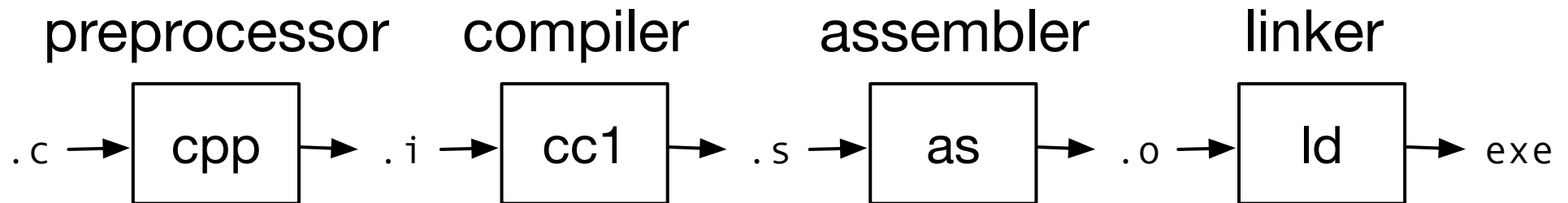
- Compile

```
linux> gcc -Og hello-world.c
```

- Execute

```
linux> ./a.out
Hello world!
```

Compilation Steps



- C code first gets compiled into assembly code
- Assembly code is then converted into machine code

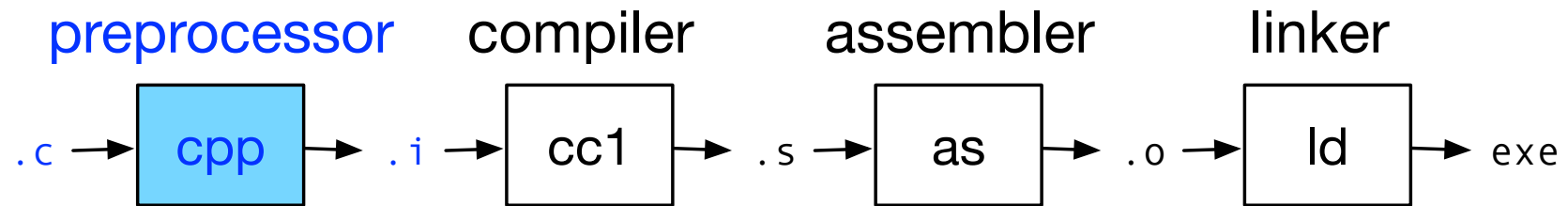
Even Simpler Program



- A simple C program: return47.c

```
#define FOURTYSEVEN 47
int main(void) {
    return FOURTYSEVEN;
}
```

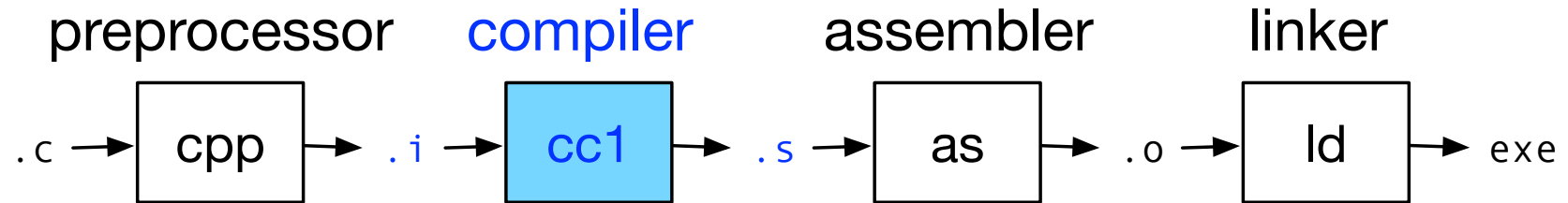
Preprocessor



- Resolves constants (#define)
- Adds additional source code (#include)
- Handles other directives like #ifdef / #endif
- Example

```
linux> gcc -Og -E return47.c  
[...]  
int main(void) {  
    return 47;  
}
```

Compiler



- Compilation into assembly code

- Example

```
linux> gcc -Og -S return47.c
```

```
linux> cat return47.s
```

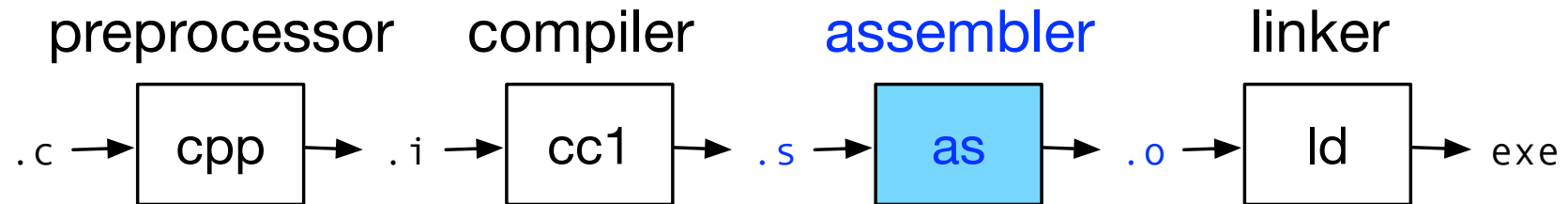
```
[...]
```

```
main:
```

```
    movl    $47, %eax  
    ret
```

return value

Assembler



2f hex = 47 dec

- Conversion into machine code

- Example

```
linux> gcc -Og -c return47.c
```

```
linux> objdump -d return47.o
```

```
[...]
```

```
00000000000000000000 <main>:
```

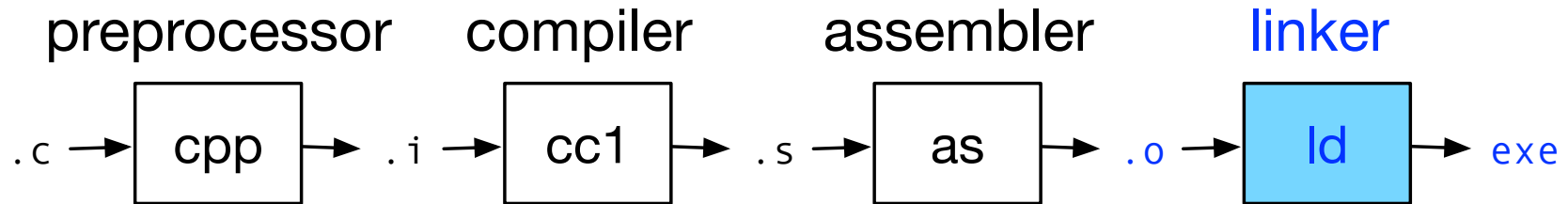
```
0:            b8 2f 00 00 00
```

```
5:            c3
```

```
mov        $0x2f,%eax
```

```
retq
```

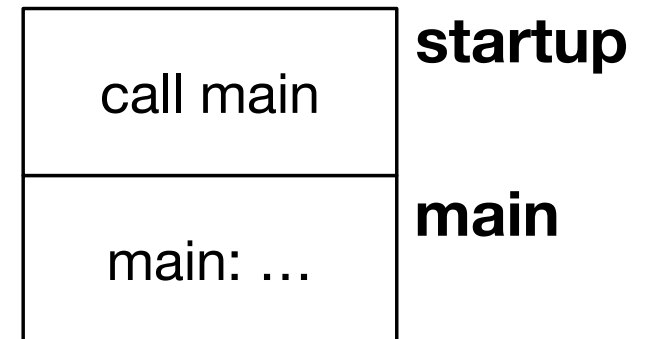
Linker



- Adds start-up code
- May combine multiple object files

- Example

```
linux> gcc -Og return47.c
linux> ./a.out
linux> echo $?
47
```



loops

Simple Program with For Loop



```
int main(void) {  
    int sum = 0;  
    for(int i=0;i<100;i++) {  
        sum += i;  
    }  
    return 0;  
}
```



Assembly Code

```
main:
    movl    $0, %eax
    jmp     .L2
.L3:
    addl   $1, %eax
.L2:
    cmpl   $99, %eax
    jle    .L3
    movl   $0, %eax
    ret
```

.L loop Start:

.L stuff

"local labels"

Assembly Code



```
main:
    movl    $0, %eax
    jmp     .L2
.L3:
    addl   $1, %eax
.L2:
    cmpl   $99, %eax
    jle    .L3
    movl   $0, %eax
    ret
```

- **Wait!** --- where is the sum computed?

Assembly Code

```
main:
    movl    $0, %eax
    jmp     .L2
.L3:
    addl   $1, %eax
.L2:
    cmpl   $99, %eax
    jle    .L3
    movl   $0, %eax
    ret
```

- **Wait!** --- where is the sum computed?
- Removed by optimizations in compiler (sum is never used)
- Compiling with `-O9` would also remove loop

Use Sum as Return Value



```
int main(void) {
    int sum = 0;
    for(int i=0;i<100;i++) {
        sum += i;
    }
    return sum;
}
```

Assembly Code

12



```
main:
.LFB0:
    movl    $0, %edx i
    movl    $0, %eax sum
    jmp     .L2
.L3:
    addl    %edx, %eax
    addl    $1, %edx
.L2:
    cmpl   $99, %edx
    jle    .L3
    rep ret
```

gcc -S foo.c
:
: -> *foo.s*

- Now sum is computed in register %eax (return value)



hello world

Source Code

```
#include <stdlib.h>
#include <stdio.h>

int main(void) {
    printf("Hello world!\n");
    return EXIT_SUCCESS;
}
```

Assembly Code

- Compiled into:

```
.section .rodata
.LC0:
    .string      "Hello world!"
    .text
    .globl      main
    .type       main, @function
main:
    subq        $8, %rsp
    movl        $.LC0, %edi
    call        puts
    movl        $0, %eax
    addq        $8, %rsp
    ret
```

make symbol table entry

code segment

stack alignment

- Calls the function "puts"

Machine Code (Disassembled)

- Object code

```
linux> objdump -t hello-world.o
```

```
[...]
```

```
0000000000000000 g      F .text          0000000000000018 main
0000000000000000          *UND*           0000000000000000 puts
```

"relocation"

- Function "puts" is labeled as undefined (*UND*)
- Linker resolves this