Discussion 4: CALL

Tianyuan Wu

What’s CALL?

C: Compiler

A: Assembler

L: Linker

L: Loader

CALL Pipeline

$$\begin{tikzcd}\text{Compiler (CS 131, Compilers)} \arrow[r] & \text{Assembler (This course)} \arrow[r] & \text{Linker (This course)} \arrow[r] & \text{Loader (CS130, OS)}\end{tikzcd}$$

CALL Pipeline

$$\begin{tikzcd}
\text{Compiler (CS 131)} \arrow[r] & \text{Assembler (This course)} \arrow[r] & \text{Linker (This course)} \arrow[r] & \text{Loader (CS130)}
\end{tikzcd}$$

What does a compiler do?

- Translates high-level language codes(C, C++, etc) to assembly codes.

- Input: *.c/*.cpp/…; output: *.s.

- Syntax checking, Types checking, Semantics, Optimization, Codegen…

What does an assembler do?

- Translates assembly codes to machine codes.

- Input: *.s; output: *.o.

- Expands pseudo-instructions into basic ones. (e.g. la to auipc and addi, mv to addi, etc.)

- Reads and uses directives.

Directives

- .symbol
- .data
- .word
- .text
- .relocate
- ......

Symbol Table

```
label1:
        li              x10, 1
        addi       x10, x10, 1
        jr              ra
main:
        jal             label1
```

Symbol Table

```
label1:
        li              x10, 1
        addi       x10, x10, 1
        jr              ra
main:
        jal             label1
```

In real RISC-V:

```
.local        label1
.global       main
```

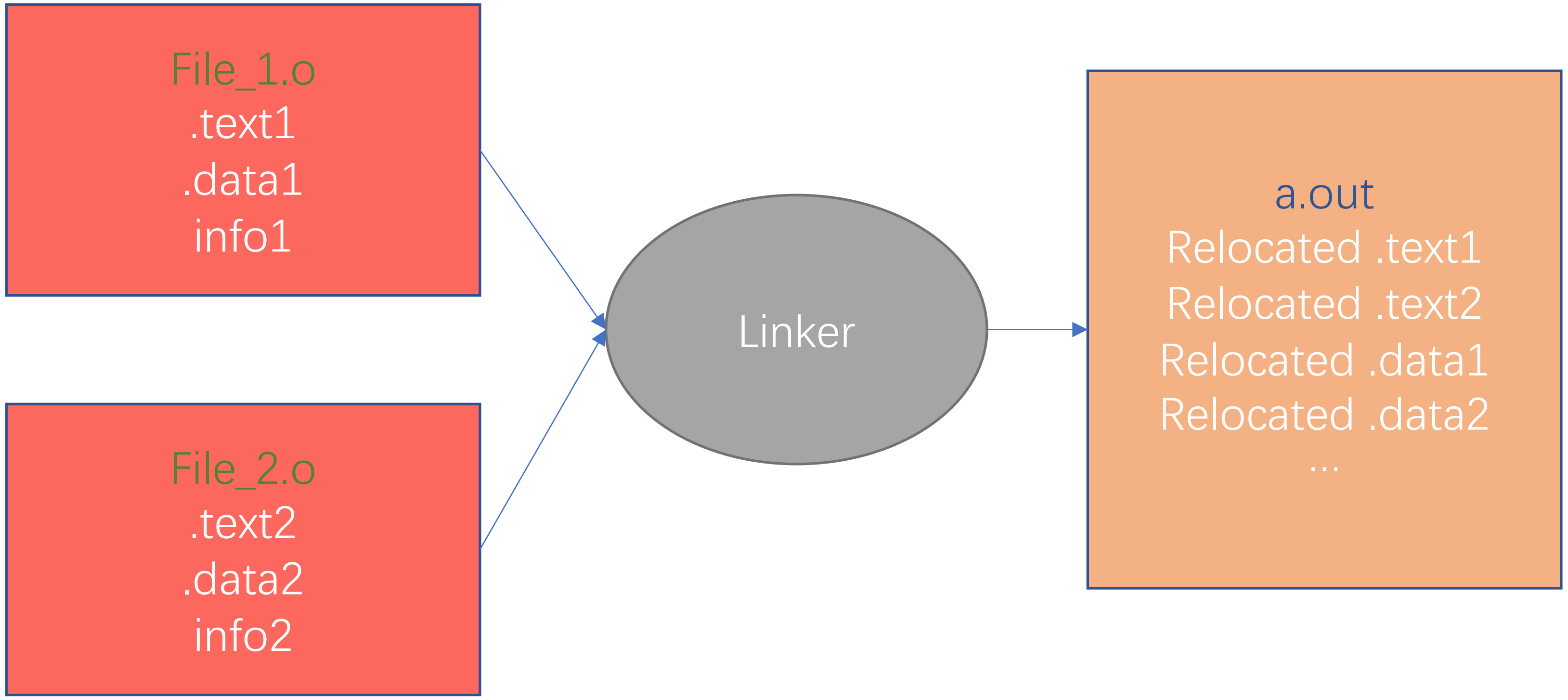
In symbol table

```
.symbol
0       label1
12      main
```

What does Linker do?

- Take text segment from each .o file and put them together
- Take data segment from each .o file, put them together, and concatenate this onto end of text segments
- Resolve references
  – Go through Relocation Table; handle each entry
  – That is, fill in all absolute addresses

What does Linker do?



Relocation Table in RISC-V

- Actually no relocation table in RISC-V

- Linking is done by linker

- Unknown absolute address is marked by:
    - %lo, %hi

Loader

- Reads executable file's header to determine size of text and data segments
- Creates new address space for program large enough to hold text and data segments, along with a stack segment
- Copies instructions and data from executable file into the new address space
- Copies arguments passed to the program onto the stack
- Initializes machine registers – Most registers cleared, but stack pointer assigned address of 1st free stack location
- Jumps to start-up routine that copies program's arguments from stack to registers & sets the PC

Loader

- Reads executable file's header to determine size of text and data segments
- Creates new address space for program large enough to hold text and data segments, along with a stack segment
- Copies instructions and data from executable file into the new address space
- Copies arguments passed to the program onto the stack
- Initializes machine registers – Most registers cleared, but stack pointer assigned address of 1st free stack location
- Jumps to start-up routine that copies program's arguments from stack to registers & sets the PC

You will learn more about loader in OS course in next semester