Discussion 4: CALL and hints for Proj 1.1

Zhijie Yang

CALL Pipeline

CALL Pipeline

What does an assembler do?

- Translates assembly codes to machine codes.

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

- Expands pseudo-instructions into basic ones.

- Reads and uses directives.

Directives

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

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 project 1.1 (for simplicity):

```
.symbol
0   label1
8   main
```

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

Example of Handling Abs. Addressing

foo.c:

```
.data
a:  .word   1, 2, 3
b:  .word   4, 5, 6
.text
main:
    la      a0, a
    la      a1, b
```

bar.c:

```
.data
c:  .word   0, 2, 4
d:  .word   1, 3, 5
.text
main:
    la      a2, c
    la      a3, d
```

Example of Handling Abs. Addressing

foo.c:

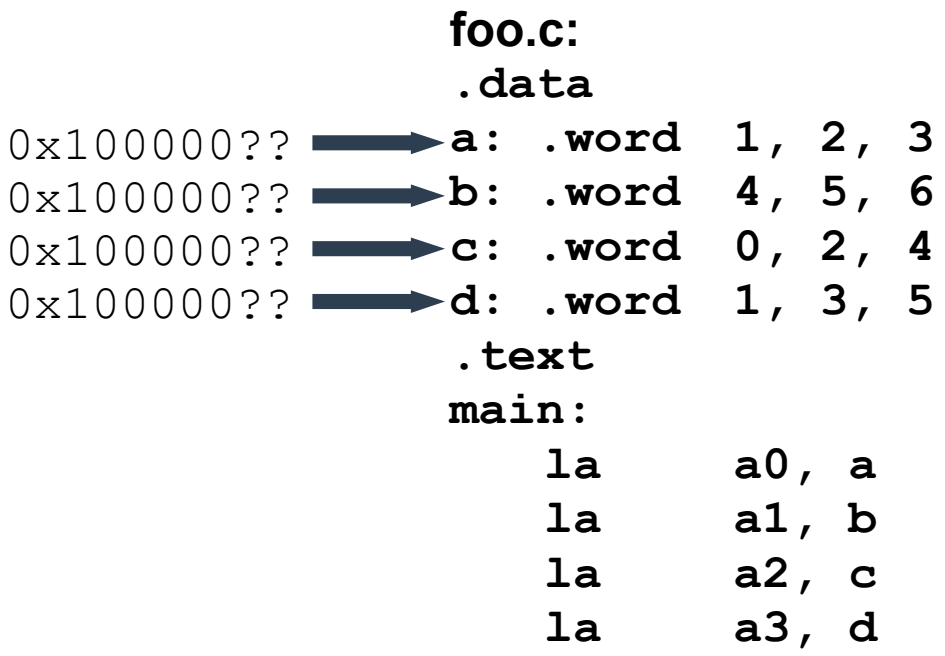
```
                .data
0x10000000 →    a:  .word   1, 2, 3
0x1000000C →    b:  .word   4, 5, 6
                .text
                main:
                    la      a0, a
                    la      a1, b
```

bar.c:

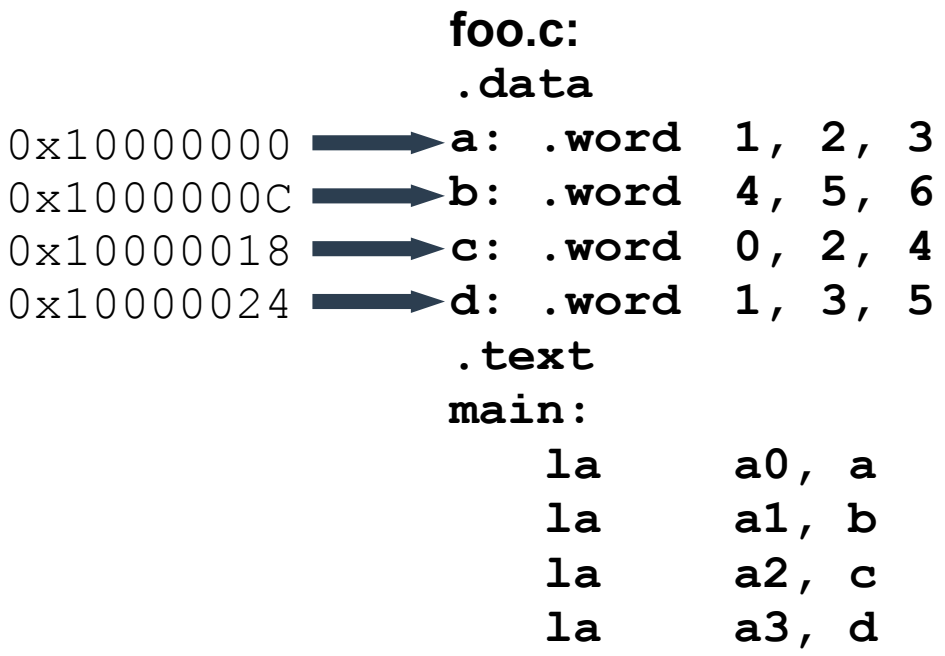
```
                .data
0x10000000 →    c:  .word   0, 2, 4
0x1000000C →    d:  .word   1, 3, 5
                .text
                main:
                    la      a2, c
                    la      a3, d
```

```
la      a0, a   →   lui     a0, %hi(a)
                    addi    a0, a0, %lo(a)
```

Example of Handling Abs. Addressing



Example of Handling Abs. Addressing



How did linker know this?

- Take a simpler example

```
static int a[5] = {1, 2, 3, 4, 5};
int main ()
{
    for (int i = 0; i < 5; i++)
        a[i] = i;
}
```

```
    …
lui     a5,%hi(a)
    …
addi    a5,a5,%lo(a)
    …
```



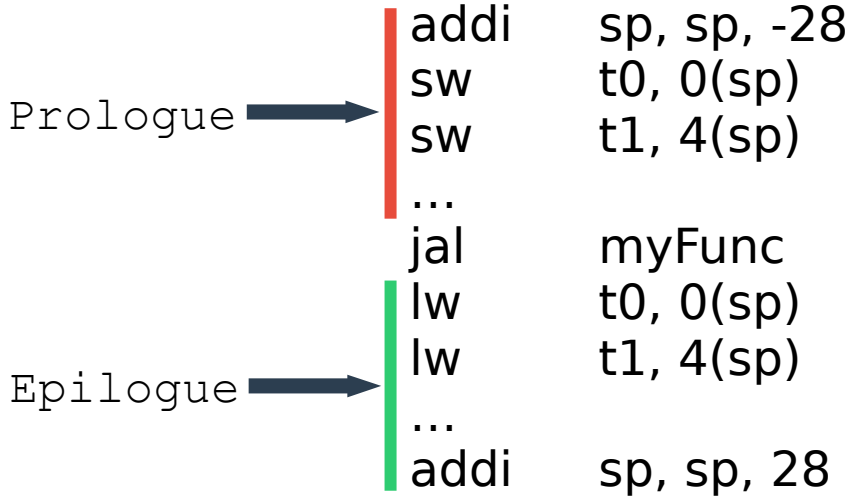
```
 c:  000007b7            lui a5,0x0
        c: R_RISCV_HI20  a
        c: R_RISCV_RELAX *ABS*
10:  …
14:  …
16:  00078793              mv  a5,a5
        16: R_RISCV_LO12_I  a
        16: R_RISCV_RELAX   *ABS*
```


More reference:
1. https://www.sifive.com/blog/all-aboard-part-2-relocations
2. https://www.sifive.com/blog/all-aboard-part-3-linker-relaxation-in-riscv-toolchain

Hints on Homework 3

- **Use of stack**
- **Idea of frame***
- **Saving registers while calling**
  - prologue
  - epilogue

Stack Maintenance




```
        addi    sp, sp, -28
        sw      t0, 0(sp)
        sw      t1, 4(sp)
        ...
        jal     myFunc
        lw      t0, 0(sp)
        lw      t1, 4(sp)
        ...
        addi    sp, sp, 28
MyFunc:
        addi    sp, sp, -16
        sw      s0, 0(sp)
        ...
        {function body}
        lw      s0, 0(sp)
        ...
        addi    sp, sp, 16
        jr      ra
```

Hint on Project 1.1

- Number translating
- Pseudo-instruction expansion
- Invalid input handling
- Memory management