CS 61C — Instruction Translation, CALL

Fall 2024 — Discussion 5

1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

1.1 The compiler may output pseudoinstructions.

1.2 The main job of the assembler is to perform optimizations on the assembly code.

1.3 The object files produced by the assembler are only moved, not edited, by the linker.

1.4 The destination of all jump instructions is completely determined after linking.

2 Translation

2.1 In this question, we will be translating between RISC-V code and binary/hexadecimal values.

Translate the following RISC-V instructions into binary and hexadecimal notations.

```
addi  s1  x0  -24  =      0b______________________________    =  0x____________
sh  s1  4(t1)      =      0b______________________________    =  0x____________
```

2.2 In this question, we will be translating between RISC-V code and binary/hexadecimal values.

Translate the following hexadecimal values into the relevant RISC-V instruction.

```
0x234554B7 =  _________________________________
0xFE050CE3 = _________________________________
```

3 RISC-V Addressing

We have several *addressing modes* to access memory (immediate not listed):

1. Base displacement addressing adds an immediate to a register value to create a data memory address (used for lw, lb, sw, sb).
2. PC-relative addressing uses the PC and adds the immediate value of the instruction (multiplied by 2) to create an instruction address (used by branch and jump instructions).
3. Register Addressing uses the value in a register as an instruction address. For instance, `jalr`, `jr`, and `ret`, where `jr` and `ret` are just pseudoinstructions that get converted to `jalr`.

3.1 What is the range of 32-bit instructions that can be reached from the current PC using a branch instruction? Recall that RISC-V supports 16b instructions via an extension.

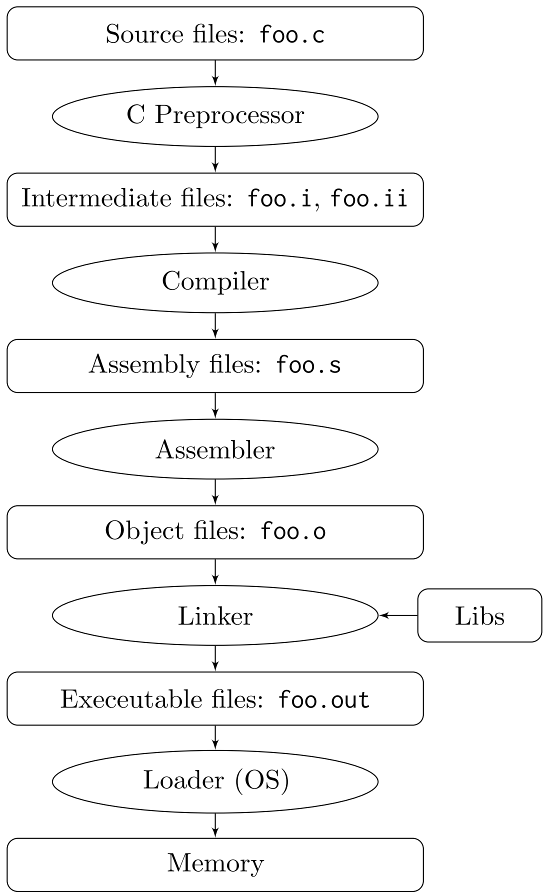
3.2 What is the maximum range of 32-bit instructions that can be reached from the current PC using a jump instruction?

3.3 Given the following RISC-V code (and instruction addresses), fill in the blank fields for the following instructions (you'll need your RISC-V reference sheet!). Each field refers to a different block of the instruction encoding.

```
0x002cff00: loop: add t1, t2, t0        |________|________|________|________|________|__0x33__|
0x002cff04:       jal ra, foo           |__________________________|_________________|__0x6F__|
0x002cff08:       bne t1, zero, loop    |________|________|________|________|________|__0x63__|
...
0x002cff2c: foo:  jr ra                ra = ____________________________
```

4 CALL

The following is a diagram of the CALL stack detailing how C programs are built and executed by machines.



4.1 How many passes through the code does the Assembler have to make? Why?

4.2 Which step in CALL resolves relative addressing? Absolute addressing?

4.3 Describe the six main parts of the object files outputted by the Assembler (Header, Text, Data, Relocation Table, Symbol Table, Debugging Information).