CS61C Discussion 4 – RISC-V Addressing, Inst. Formats, and CALL

1 RISC-V Instruction Formats

1.1 Overview

Instructions in RISC-V can be turned into binary numbers that the machine actually reads. There are different formats to the instructions, based on what information is need. Each of the feilds above is filled in with binary

CORE INSTRUCTION FORMATS

	31 27	26 25	24 20	19 15	14 12	11 7	6 0
R	funct7		rs2	rs1	funct3	rd	Opcode
I	imm[11:0]			rs1	funct3	rd	Opcode
S	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
SB	imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]					rd	opcode
UJ	imm[20|10:1|11|19:12]					rd	opcode

that represents the information. Each of the registers takes a 5 bit number that is the numeric name of the register (i.e. zero = 0, ra = 1, s1 = 9). See your reference card to know which register corresponds to which number.
I type instructions fill the immediate into the code. These numbers are signed 12 bit numbers.

1.2 Exercises

1. Expand `addi s0 t0 -1`

2. Expand `lw s4 5(sp)`

3. Write the format name of the following instructions:
   (a) `jal`
   (b) `lw`
   (c) `beq`
   (d) `add`
   (e) `jalr`
   (f) `sb`
   (g) `lui`

2 RISC-V Addressing

2.1 Overview

- We have several **addressing modes** to access memory (immediate not listed):
  (a) **Base displacement addressing:** Adds an immediate to a register value to create a memory address (used for lw, lb, sw, sb)
  (b) **PC-relative addressing:** Uses the PC and adds the immediate value of the instruction (multiplied by 2) to create an address (used by branch and jump instructions)
  (c) **Register Addressing:** Uses the value in a register as a memory address (jr)

2.2 Exercises

1. What is range of 32-bit instructions that can be reached from the current PC using a branch instruction?

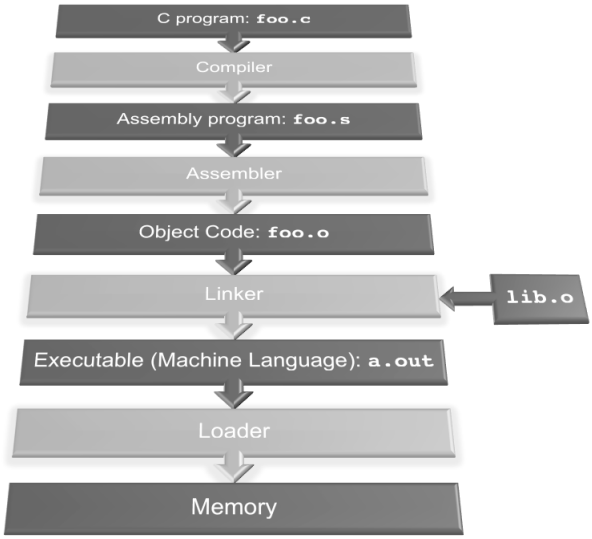
2. What is the range of 32-bit instructions that can be reached from the current PC using a jump instruction?

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 green card!).

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

3 Compile, Assemble, Link, Load, and Go!

3.1 Overview



3.2 Exercises

a. What is the Stored Program concept and what does it enable us to do?

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

c. What are the different parts of the object files output by the Assembler?

d. Which step in CALL resolves relative addressing? Absolute addressing?

e. What does RISC stand for? How is this related to pseudoinstructions?