Discussion 04

RISC-V

Aditya Balasubramanian

`aditbala [at] berkeley [dot] edu`

Slides available at `teaching.aditbala.com`

Announcements 📣

Agenda

- RISC-V Calling Convention
- Instruction Encoding

Credits to Rosalie Fang

RISC-V Calling Convention

Credits to Rosalie Fang

Calling Convention (CC)

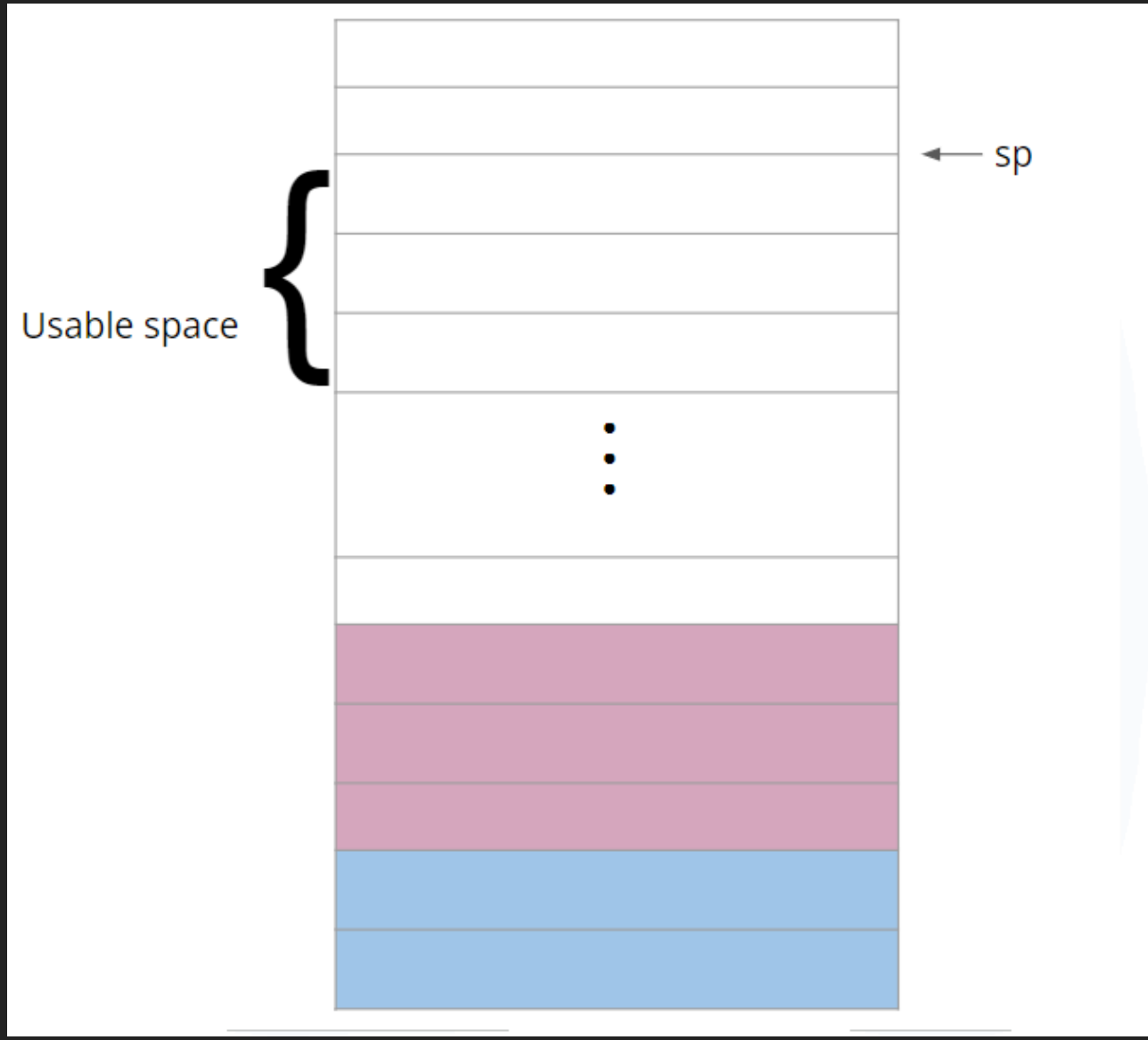
- What is it?
  - CC is the set of universal rules that all assembly of a single family is understood to follow, regardless of where the code comes from (excluding malicious usage)
- So why can't I use whatever registers I want?
  - In short: well... technically you can and no one is stopping you...
  - In reality: well... unexpected things might happen...

CC Terms

- A set of rules on registers of WHAT TO EXPECT across function calls
- **Callee saved registers**
  - Can assume it stays the same after function call
  - `s` registers, `sp`
- **Caller saved registers**
  - Might be edited after function call
  - a registers, `t` registers, `ra`

Using `sp` (Stack Pointer)

- Right below the stack pointer is usable stack space
- As long as stack pointer is restored correctly, we can recover values stored on the stack
- After storing values on the stack, we have to decrement the stack to “reserve space”



Prologue (Before you call a function)

1. Determine which registers whose values we want remain unchanged
2. Shift the stack pointer down by the number of bytes you need
    - Eq: `4 * # registers`
3. Save (`sw`) register values onto stack

RISC-V Translations

Credits to Rosalie Fang

Tips for Translation

1. Rewrite the instruction using register numbers: x_
2. Determine instruction type
3. Order instruction components with required/expected number of bits
4. Write opcode
5. Translate register values to binary, and 0-pad if necessary
6. Translate immediates if necessary, and 0-pad OR sign-extend if necessary
7. Encode offsets as two's complement signed representation (sign extension)
8. Concatenate required bits
9. Convert to hex if necessary in 4 bit increments

Credits to Rosalie Fang

Thank you!

Feedback

Credits to Rosalie Fang