

# ISAs, Stacks, and Endianness

---

CS 2130: Computer Systems and Organization 1

October 5, 2022

Hi!

# Announcements

- Homework 4 due Monday, 11pm on Gradescope

# Instruction Set Architecture

**Instruction Set Architecture (ISA)** is an abstract model of a computer defining how the CPU is controlled by software

- Provides an abstraction layer between:
  - Everything computer is really doing (hardware)
  - What programmer using the computer needs to know (software)
- Hardware and Software engineers have freedom of design, if conforming to ISA
- Can change the machine without breaking any programs

*CSO: covering many of the times we'll need to think across this barrier*

# Instruction Set Architecture

## Backwards compatibility

- Include flexibility to add additional instructions later
- Original instructions will still work
- Same program can be run on PC from 10+ years ago and new PC today

Most manufacturers choose an ISA and stick with it

- Notable Exception: Apple

# Our Instruction Set Architecture

What about our ISA?

- Enough instructions to compute what we need
- As is, lot of things that are painful to do
  - This was on purpose! So we can see limitations of ISAs early

# Our Instruction Set Architecture

What about our ISA?

- Enough instructions to compute what we need
- As is, lot of things that are painful to do
  - This was on purpose! So we can see limitations of ISAs early
- Add any number of new instructions using the reserved bit (7)

# Our Instruction Set Architecture

icode	b	meaning
0		$rA = rB$
1		$rA += rB$
2		$rA \&= rB$
3		$rA =$ read from memory at address $rB$
4		write $rA$ to memory at address $rB$
5	0	$rA = \sim rA$
	1	$rA = -rA$
	2	$rA = !rA$
	3	$rA = pc$
6	0	$rA =$ read from memory at $pc + 1$
	1	$rA +=$ read from memory at $pc + 1$
	2	$rA \&=$ read from memory at $pc + 1$
	3	$rA =$ read from memory at the address stored at $pc + 1$ For icode 6, increase $pc$ by 2 at end of instruction
7		Compare $rA$ as 8-bit 2's-complement to $\theta$ if $rA \leq \theta$ set $pc = rB$ else increment $pc$ as normal

What about real ISAs?



# Our Instruction Set Architecture

What about our ISA?

- Enough instructions to compute what we need
- As is, lot of things that are painful to do
  - This was on purpose! So we can see limitations of ISAs early
- Add any number of new instructions using the reserved bit (7)
- Missing something important: *Help to put variables in memory*

# Storing Variables in Memory

So far... we/compiler chose location for variable

Consider the following example:

f(x):

a = x

if (x <= 0) return 0

else return f(x-1) + a

Recursion

- The formal study of a function that calls itself

# Storing Variables in Memory

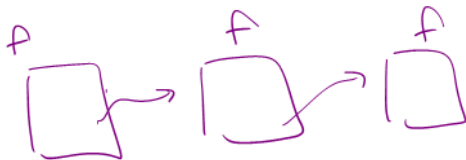
`f(x):`

`a = x`

`if (x <= 0) return 0`

`else return f(x-1) + a`

Where do we store `a`?



# The Stack

**Stack** - a last-in-first-out (LIFO) data structure

- *The* solution for solving this problem

**rsp** - Special register - the *stack pointer*

- Points to a special location in memory
- Two operations most ISAs support:
  - **push** - put a new value on the stack
  - **pop** - return the top value off the stack

# The Stack: Push and Pop

push r0

- Put a value onto the "top" of the stack

$rsp -= 1$

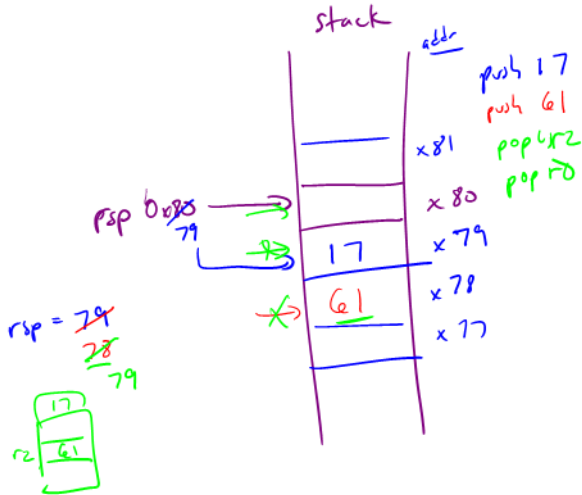
$M[rsp] = r0$

pop r2

- Read value from "top", save to register

$r2$  =  $M[rsp]$

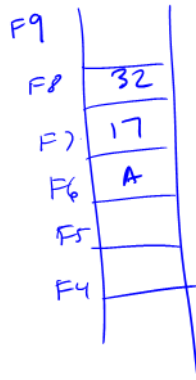
$rsp += 1$



# The Stack: Push and Pop

push 27  
r1 = pop  
push 32  
push 17  
push A  
pop r1  
pop r1

rsp  
~~F9~~  
~~F8~~  
~~F9~~  
~~E8~~  
~~E7~~  
~~E6~~  
~~E7~~  
F8



r1 = 17

# The Stack: Push and Pop

# Patents and Copyright

Can we patent our ISA? Should we?

icode	b	meaning
0		$rA = rB$
1		$rA += rB$
2		$rA \&= rB$
3		$rA =$ read from memory at address $rB$
4		write $rA$ to memory at address $rB$
5	0	$rA = \sim rA$
	1	$rA = -rA$
	2	$rA = !rA$
	3	$rA = pc$
6	0	$rA =$ read from memory at $pc + 1$
	1	$rA +=$ read from memory at $pc + 1$
	2	$rA \&=$ read from memory at $pc + 1$
	3	$rA =$ read from memory at the address stored at $pc + 1$
		For icode 6, increase $pc$ by 2 at end of instruction
7		Compare $rA$ as 8-bit 2's-complement to $\theta$ if $rA \leq \theta$ set $pc = rB$ else increment $pc$ as normal



# Patents and Copyright

## Copyright

- “Everyone is a copyright owner. Once you create an original work and fix it, like taking a photograph, writing a poem or blog, or recording a new song, you are the author and the owner.”

from <https://www.copyright.gov/what-is-copyright/>

## Patent

- “Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.”

from 35 U.S.C. 101

In software and hardware, patents become messy

- Code is a description of a process we want the computer to do
- Do not have to implement the process to patent it

Question: Should we patent something like our ISA?

In software and hardware, patents become messy

- Code is a description of a process we want the computer to do
- Do not have to implement the process to patent it

Question: Should we patent something like our ISA?

What is the current state of the art?

# Common Approaches to Software

How can we get value from what we create?

- Copyright - distribute closed source software
- License Agreements (in contract law)
- Always innovate