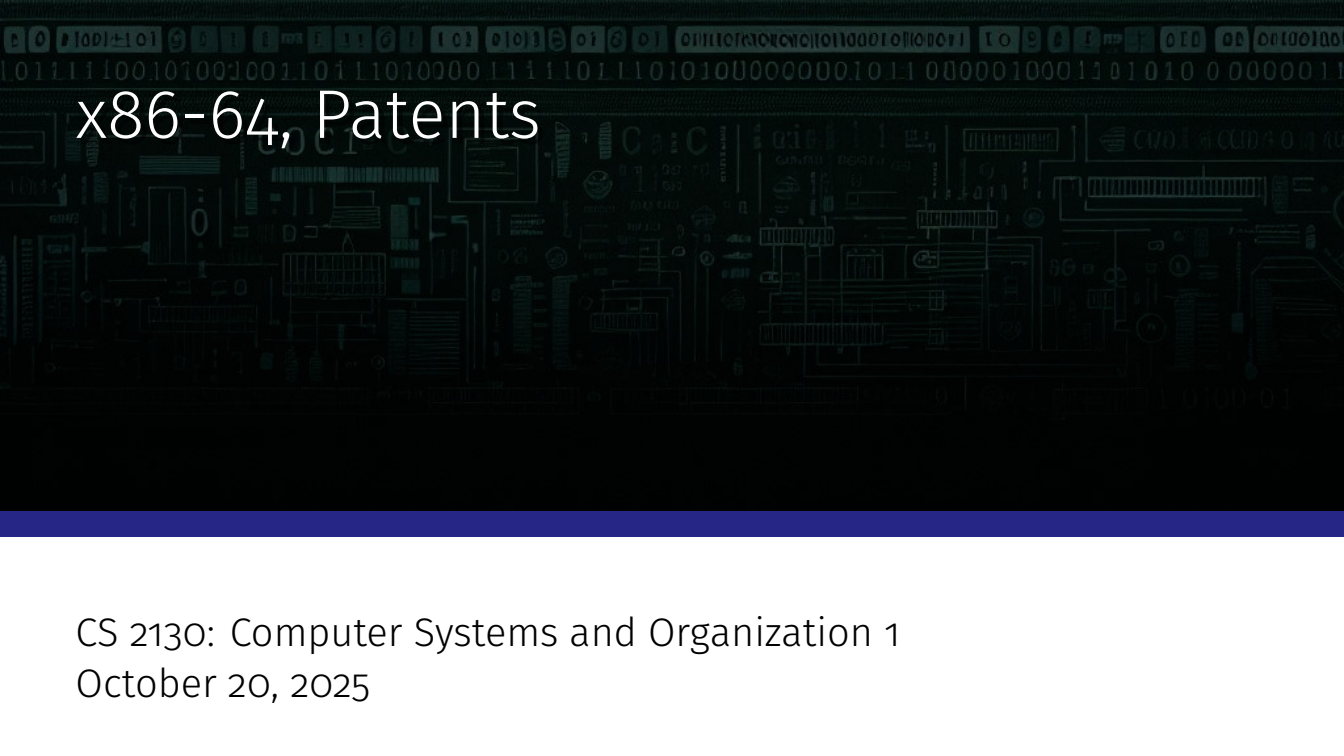




x86-64, Patents

CS 2130: Computer Systems and Organization 1
October 20, 2025

Announcements

- Homework 5 **due tonight at 11:59pm** on Gradescope
- Homework 6 available tomorrow with lab 7

Most Common Instructions

- `mov` - `=`
- `lea` - load effective address
- `call` - push PC and jump to address
- `add` - `+=`
- `cmp` - set flags as if performing subtract `-=`
- `jmp` - unconditional jump
- `test` - set flags as if performing `&`
- `je` - jump iff flags indicate `== 0`
- `pop` - pop value from stack
- `push` - push value onto stack
- `ret` - pop PC from the stack

Debugger

Debugger - step through code!

- Helpful for Homework 5
- You will be using this for lab 7
- Experience seeing results of these instructions step-by-step
- **Please read the x86-64 summary reading!**

Function Calls: Calling Conventions

`callq myfun`

- Push return address, then jump to myfun
- Convention: Store arguments in registers and stack before call
 - First 6 arguments (in order): rdi, rsi, rdx, rcx, r8, r9
 - If more arguments, pushed onto stack (last to first)

`retq`

- Pop return address from stack and jump back
- Convention: store return value in `rax` before calling `retq`

This is similar to our Toy ISA's function calls in homework 4

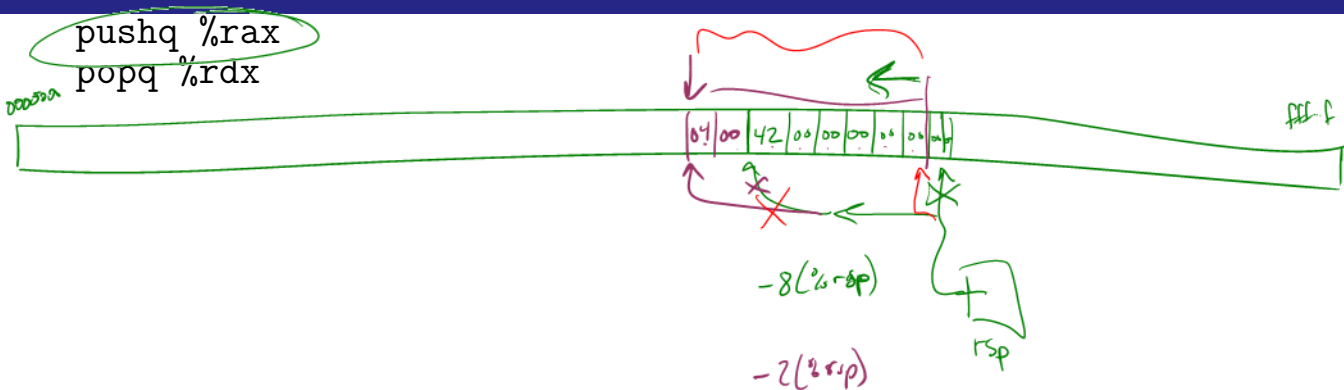
Calling Conventions: Registers

Calling conventions - recommendations for making function calls

- Where to put arguments/parameters for the function call?
- Where to put return value? in `rax` before calling `retq`
- What happens to values in the registers?
 - **Callee-save** - The function should ensure the values in these registers are unchanged when the function returns
 - * `rbx, rsp, rbp, r12, r13, r14, r15`
 - **Caller-save** - Before making a function call, save the value, since the function may change it

The Stack

```
pushq %rax
popq %rdx
```



Patents and Copyright

Remember our Toy ISA. Can we patent our ISA? Should we?

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

Patents and 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

Patents

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?

Patents

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