x86_64, Debugger
Patents and Copyright
Compilation Pipeline

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

UNIVERSITY of VIRGINIA | ENGINEERING

Announcements

- Homework 6 due Monday at 11:59pm

```
        .globl  main
main:
        pushq   %rbp
    movq    $0x42, %rax
    movq    $0x15, %rbx
    movq    %rbx, %rsi
    negq    %rsi
    addq    %rax, %rsi
        leaq    fmtstring(%rip), %rdi
        callq   printf
    xorq    %rax, %rax
        popq    %rbp
        retq
fmtstring:
        .asciz  "%X\n"
```

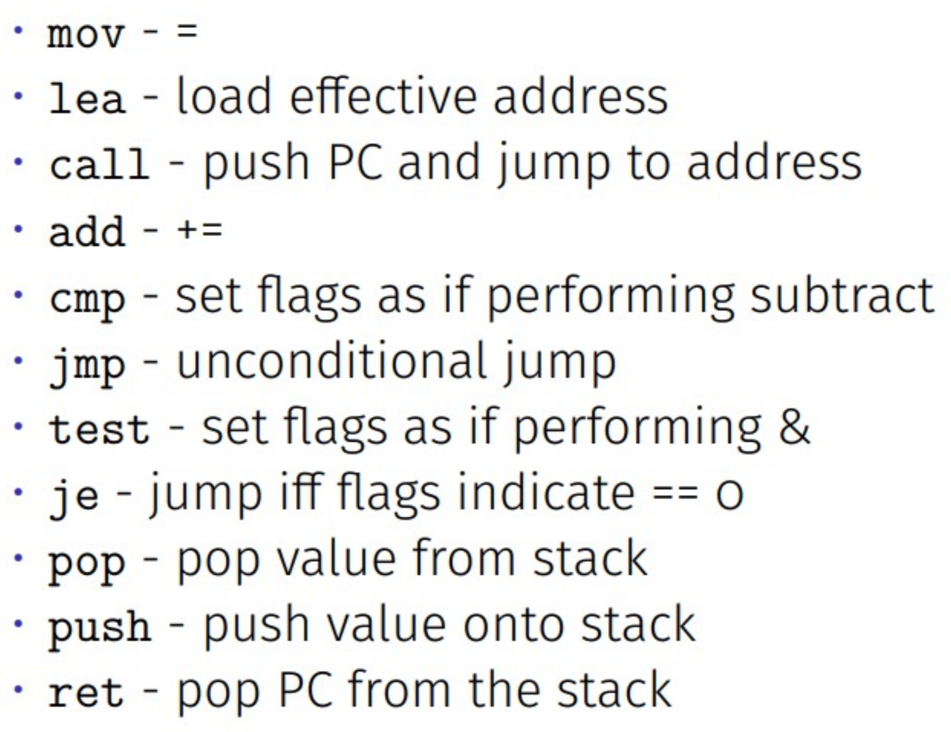
The Stack

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

```
 .globl  main
main:
   pushq   %rbp

   # Set some example values in registers
   movq    $0x42, %rax
   movq    $0x15, %rbx
   movl    $4, %esi

   # Push 64-bit rax, then 16-bit si
   pushq   %rax
   pushw   %si

   # Pop -- oops!
   popq    %rdi
   popw    %si

   # Return 0 (all is well)
   xorq    %rax, %rax
   popq    %rbp
   retq
```

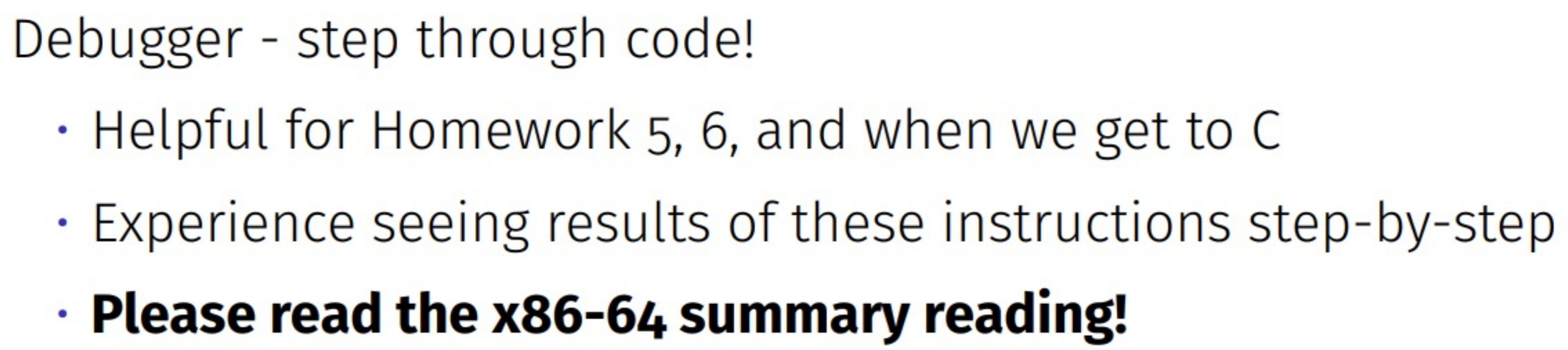
UNIVERSITY of VIRGINIA

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



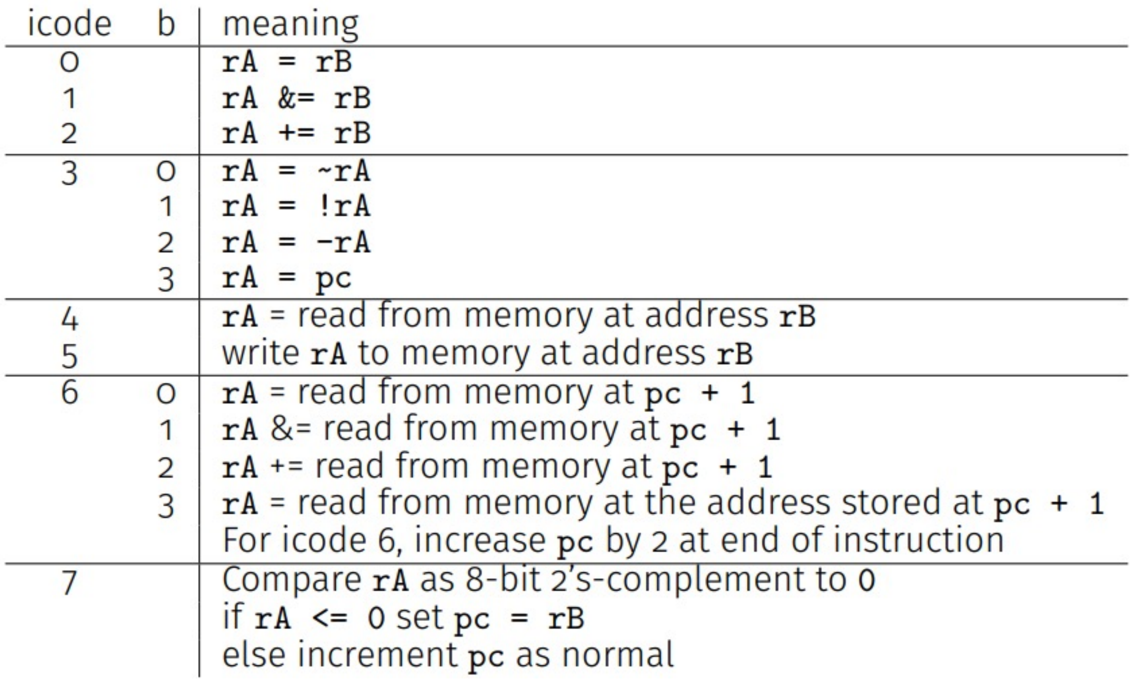
Debugger - step through code!

- Helpful for Homework 5, 6, and when we get to C
- Experience seeing results of these instructions step-by-step
- **Please read the x86-64 summary reading!**

Patents and Copyright

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 = ~rA
	1	rA = !rA
	2	rA = -rA
	3	rA = pc
4		rA = read from memory at address rB
5		write rA to memory at address rB
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 0 if rA <= 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

Category	**Patent**	**Copyright**	**License Agreement**
Protected Object	Technological inventions and innovations	Creative expressions and forms (e.g., code, text, images)	Usage rights of protected content
Source of Right	Granted by government agencies after examination	Automatically arises upon creation	Established through contractual agreement
Automatic Acquisition	❌ No	✅ Yes	❌ No (requires signing)
Duration	Typically 20 years	Author's lifetime + a number of years (e.g., 70 years in the U.S.)	Determined by contract terms
Nature of Right	Exclusive right to make, use, or sell an invention	Prevents copying or unauthorized reproduction	Defines how others can use or distribute the work
Examples	A new GPU scheduling algorithm, hardware design	Source code, research paper, textbook	MIT License, GPL, Apache 2.0, commercial EULA
Key Focus	Innovation and technical exclusivity	Expression and originality	Permission and terms of use

Patents Cold War

- The "Patent Cold War" describes a modern corporate phenomenon where major technology companies (Apple, Google, IBM, Microsoft, Qualcomm, etc.) stockpile patents—not primarily to innovate, but to defend against lawsuits or block competitors.
- It's called a "cold war" because companies deter each other through mutual threats of litigation, much like nuclear deterrence.

📱 *Apple vs. Samsung (2011–2020):* Massive smartphone patent battles over design and gestures.
🔍 *Google vs. Oracle:* Decade-long litigation over Java API copyrights, shaping software development law.
💾 *IBM and Microsoft patent pools:* Used to maintain dominance and restrict new entrants.

Some Facts

- IBM files **8,000+ patents annually**, but many are never implemented.
- Studies show that in **high-patent-density fields** (semiconductors, software), innovation rates may decline.
- The U.S. Supreme Court's *Alice Corp. v. CLS Bank (2014)* decision ruled that **abstract software algorithms are not patentable**, aiming to limit excessive claims.
- In contrast, **Tesla (2014)** opened all its electric vehicle patents to promote industry-wide innovation—a symbolic move against the "patent cold war."

Discuss

- Do modern patent systems still promote innovation—or do they mostly protect big companies?
- Should algorithms and software be patentable, given that they are often abstract ideas?
- How can startups survive in industries dominated by patent-rich corporations?
- Is it ethical for companies to file thousands of patents purely to block competition?
- Should universities and researchers embrace open-source sharing instead of patent races?

Lessons

- **The system needs rebalancing.**
  - Patent protection is still essential — without it, true innovators might lose incentives.
  - But reform is needed to ensure **patents serve the public good**, not just corporate power.
  - Solutions include improving patent quality, limiting software patentability, shortening terms for fast-moving fields, and encouraging open innovation models.
- **Emerging trend: “Open Innovation” as an antidote.**
  - Companies like Tesla, IBM (with open-source AI frameworks), and several universities are embracing openness.
  - Sharing technology publicly can **accelerate collective progress** and reduce the wasteful “arms race” of patents.

Compilation Pipeline

Turning our code into something that runs

- **Pipeline** - a sequence of steps in which each builds off the last

Why did we discuss assembly?

C

C is a thin wrapper around assembly

- This is by design!
- Invented to write an operating system
  - Can write inline assembly in C
- Many other languages decided to look like C

Simple C Example

```
int main() {
    int y = 5;
    return 0;
}
```

Compilation Pipeline

Earlier, we saw:

- C files (`.c`) compiled to assembly (`.s`)
- Assembly (`.s`) assembled into object files (`.o`)
- Object files (`.o`) linked into a program / executable

Compiling C to Assembly

Multiple stages to compile C to assembly

- Preprocess - produces C
  - C is actually implemented as 2 languages: C preprocessor language, C language
  - Removes comments, handles preprocessor directives (#)
  - `#include`, `#define`, `#if`, `#else`, ...
- Lex - breaks input into individual tokens
- Parse - assembles tokens into intended meaning (parse tree)
- Type check - ensures types match, adds casting as needed
- Code generation - creates assembly from parse tree

Compiling C to Assembly

Compiling C to Assembly

Errors

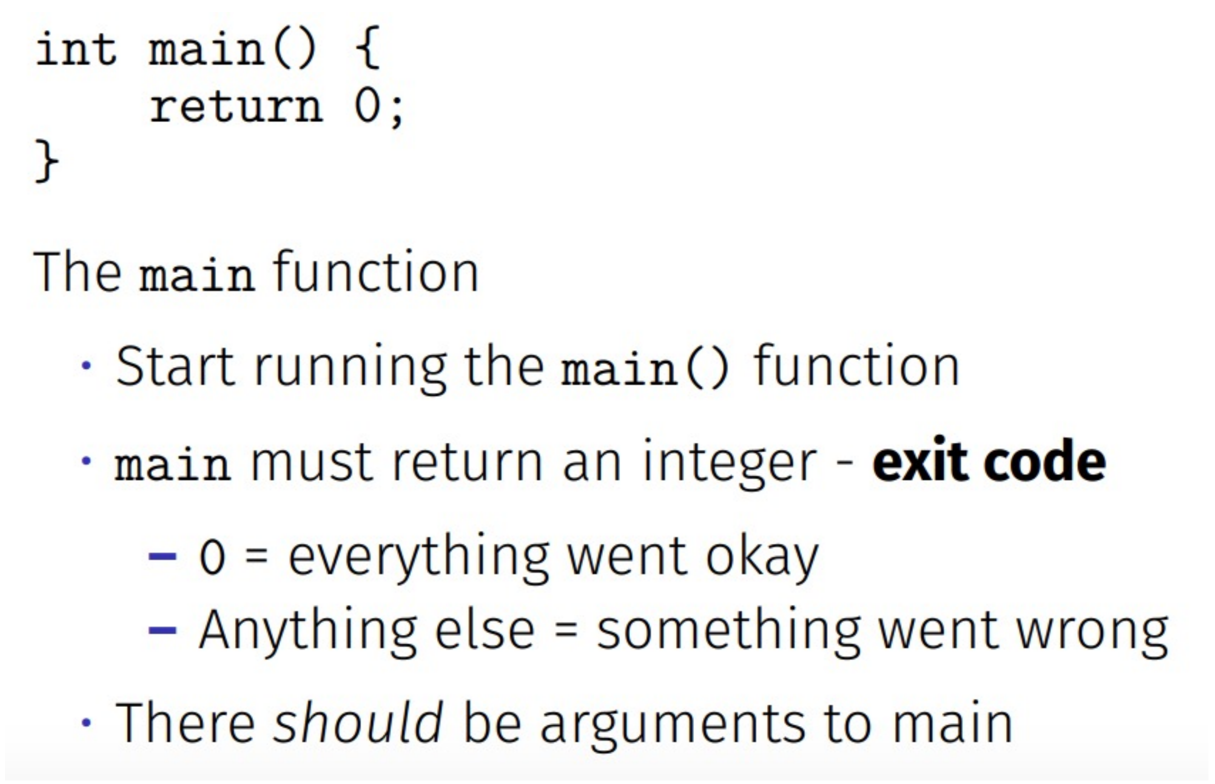
Compile-time errors

- Errors we can catch during compilation (this process)
- **Before** running our program

Runtime errors

- Errors that occur when running our programs

Simple C Example



```c
int main() {
    return 0;
}
```

The `main` function

- Start running the `main()` function
- `main` must return an integer - **exit code**
  - 0 = everything went okay
  - Anything else = something went wrong
- There *should* be arguments to main

Data Types in C

Integer data types

Data type	Size
char	
short	
int	
long	
long long	

Each has 2 versions: *signed* and *unsigned*

Data Types in C

Floating point
- float
- double

Data Types in C

Data Types in C

Pointers - how C uses addresses!

Data Types in C

Pointers - how C uses addresses!

- Hold the address of a position in memory
- Need to know the kind of information stored at that location

Example

```
int main() {
    int x = 3;
    long y = 4;
    int *a = &x;
    long *b = &y;
    long z = *a;
    int w = *b;
    return 0;
}
```

Example

```c
int main() {
    int x = 3;
    long y = 4;
    int *a = &x;
    long *b = &y;
    long z = *a;
    int w = *b;
    return 0;
}
```

```
0000000000000000 <main>:
   0:	55                   	push   %rbp
   1:	48 89 e5             	mov    %rsp,%rbp
   4:	31 c0                	xor    %eax,%eax
   6:	c7 45 fc 00 00 00 00 	movl   $0x0,-0x4(%rbp)
   d:	c7 45 f8 03 00 00 00 	movl   $0x3,-0x8(%rbp)
  14:	48 c7 45 f0 04 00 00 	movq   $0x4,-0x10(%rbp)
  1b:	00
  1c:	48 8d 4d f8          	lea    -0x8(%rbp),%rcx
  20:	48 89 4d e8          	mov    %rcx,-0x18(%rbp)
  24:	48 8d 4d f0          	lea    -0x10(%rbp),%rcx
  28:	48 89 4d e0          	mov    %rcx,-0x20(%rbp)
  2c:	48 8b 4d e8          	mov    -0x18(%rbp),%rcx
  30:	48 63 09             	movslq (%rcx),%rcx
  33:	48 89 4d d8          	mov    %rcx,-0x28(%rbp)
  37:	48 8b 4d e0          	mov    -0x20(%rbp),%rcx
  3b:	48 8b 09             	mov    (%rcx),%rcx
  3e:	89 4d d4             	mov    %ecx,-0x2c(%rbp)
  41:	5d                   	pop    %rbp
  42:	c3                   	retq
```