COSC 511: Computer Architecture
Instruction Set Architecture (ISA)

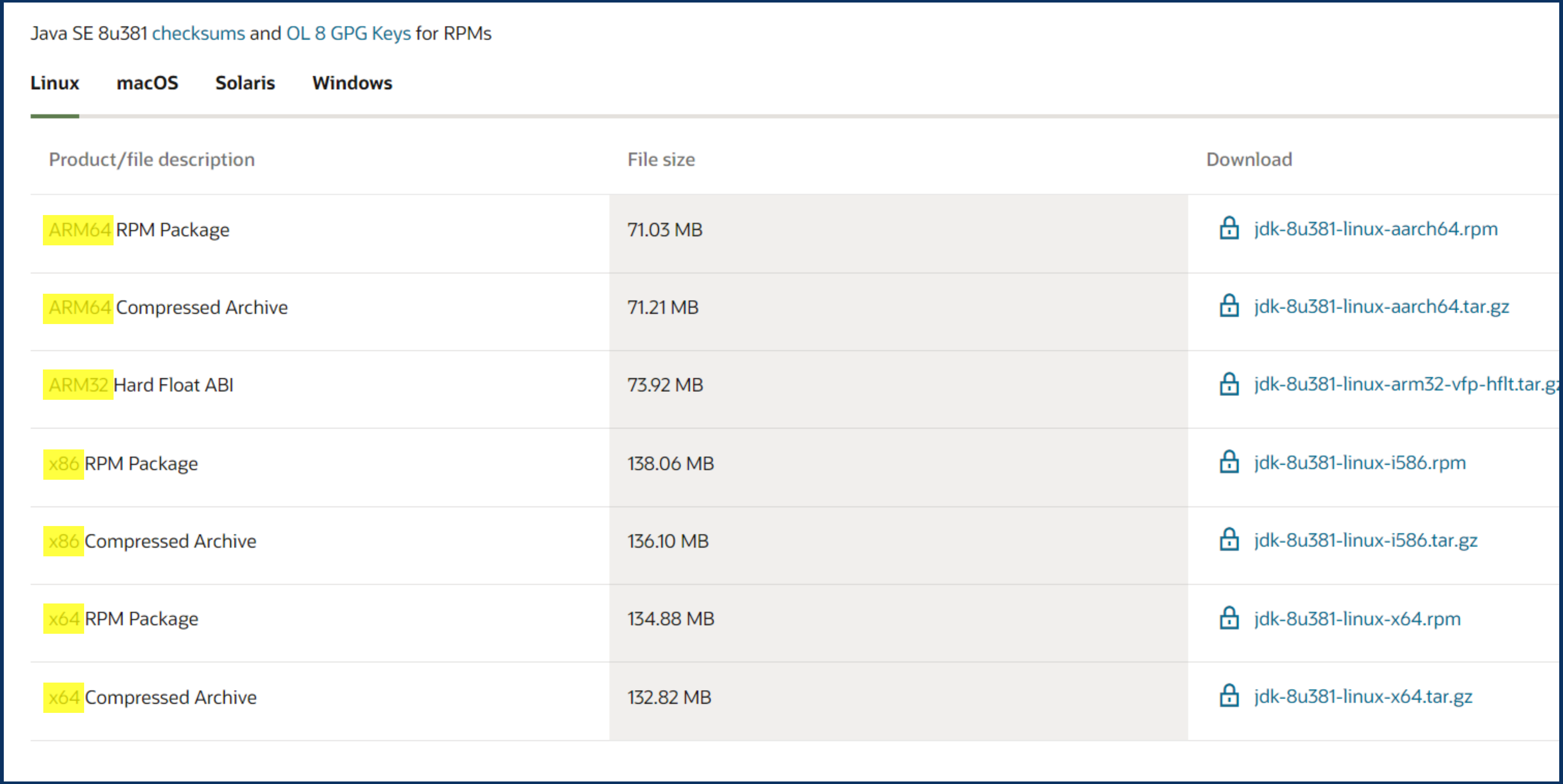
Week 2

Last Week

- Syllabus Review 🥱 😴
- Defining Performance
  - Performance means different things in different contexts.
- Layers from application software down to hardware
- High vs Low-Level Languages
- Use of abstraction to make computers easier to work with
- Response time vs Throughput
- Measuring Elapsed Time vs CPU Time
- A bunch of formulas that you should be familiar with but don't need to memorize

Instruction Set Architecture (ISA)

- ISA – A specification that defines the behavior of machine code



Java SE 8u381 checksums and OL 8 GPG Keys for RPMs

Linux **macOS** **Solaris** **Windows**

Product/file description	File size	Download
ARM64 RPM Package	71.03 MB	jdk-8u381-linux-aarch64.rpm
ARM64 Compressed Archive	71.21 MB	jdk-8u381-linux-aarch64.tar.gz
ARM32 Hard Float ABI	73.92 MB	jdk-8u381-linux-arm32-vfp-hflt.tar.g
x86 RPM Package	138.06 MB	jdk-8u381-linux-i586.rpm
x86 Compressed Archive	136.10 MB	jdk-8u381-linux-i586.tar.gz
x64 RPM Package	134.88 MB	jdk-8u381-linux-x64.rpm
x64 Compressed Archive	132.82 MB	jdk-8u381-linux-x64.tar.gz

Instruction Set Architecture (ISA)

- ISA – A specification that defines the behavior of machine code
- Examples of ISAs
  - x86
    - Initially developed by Intel for the Intel 8086 CPU
    - This ISA has gone through several iterations
      - 16-bit variant released in 1978
      - 32-bit variant released in 1985
      - 64-bit variant released in 2003
    - Technically, the 32-bit version of this ISA is called IA-32 (Intel Architecture, 32-bit) or i386
      - In the "real world", it is usually just referred to as x86
    - The 64-bit version of the standard is usually called x64 or amd64
      - The "amd" part of the name is a reference to the fact that even though Intel created the original architecture, the work to support 64-bit was done by AMD.
    - This architecture is most common in modern computers—especially amd64.

Instruction Set Architecture (ISA)

- ISA – A specification that defines the behavior of machine code
- Examples of ISAs
  - ARM
    - First devices to use ARM launched in 1985
      - The original version of ARM used a 32-bit internal structure with a 26-bit address space
        - » 26-bit address space limited it to 64 MB of memory
        - » This limitation was removed in later iterations of ARM
    - The first version of ARM with 64-bit support launched in 2011
      - Typically, 64-bit ARM variants are referred to as arm64
    - This architecture is common for small electronics or electronics with battery limitations
    - Raspberry Pi
      - Every Raspberry Pi uses ARM
      - The Raspberry Pi 3 was the first model to support arm64
      - Most modern smartphones use arm64

Instruction Set Architecture (ISA)

- This architecture is common for small electronics or electronics with battery limitations
- Raspberry Pi
  - Every Raspberry Pi uses ARM
  - The Raspberry Pi 3 was the first model to support arm64
  - Most modern smartphones use arm64
- Android
  - First Android phone: HTC Dream (October 22, 2008) used ARM
  - First arm64-based Android: HTC Desire 510 (September 2014)
- iPhone
  - The first iPhone (released June 29, 2007) used ARM
  - First arm64-based iPhone: iPhone 5S (released September 20, 2013)
- ARM is more energy-efficient than x86

Instruction Set Architecture (ISA)

- If ARM is more energy-efficient than x86, then why hasn't it been more widely-adopted as the standard ISA for desktops and laptops?
  - Answer: Application compatibility!
- Applications are compiled to work on certain ISAs
  - Can we run x64 application on x86?
  - Can we run x86 applications on x64?
  - Can we run arm applications on arm64?
  - Can we run arm64 applications on arm?
  - Can we run arm64 applications on x64?
  - Can we run x64 applications on arm?

Instruction Set Architecture (ISA)

- Unless an ISA is built to have compatibility with another ISA, we cannot run an application built for one ISA on top of a different ISA.
  - Or can we?
- Story Time
  - March 1994: Apple launches the Power Macintosh
    - The Power Macintosh used the Intel PowerPC ISA
  - June 2006: Apple ends their use of PowerPC
    - The first generation of Macs are launched using x86 and x64
    - Problem: But PowerPC applications are not compatible with x86 and x64!
    - Solution: Rosetta – A real-time instruction translator to convert PowerPC instructions

Instruction Set Architecture (ISA)

- Story Time 2: Electric Boogaloo
  - November 2020: Apple ends their use of Intel x86 and x64
    - The first generation of Macs launch with Apple Silicon
    - Apple Silicon is Apple's implementation of arm64
      - Is Apple Silicon Really ARM?
    - Problem: All modern macOS application are built on top of x86 and x64!
    - Solution: You guessed it: Rosetta 2!
      - A real-time translator from x86 and x64 to arm64.

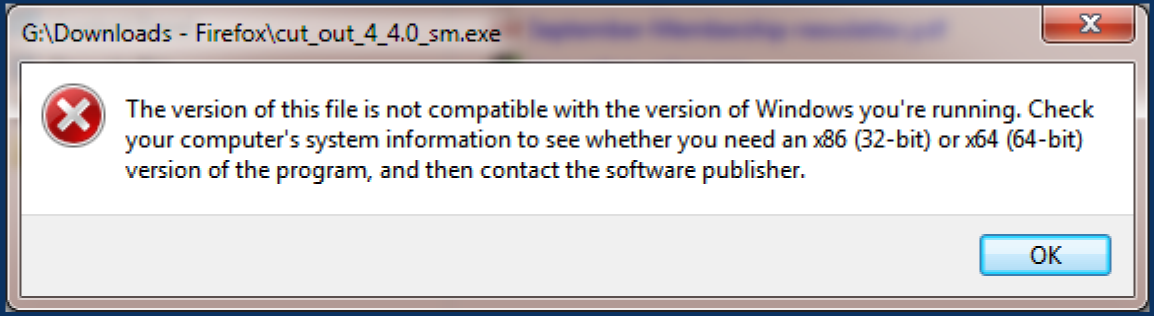
- What about Windows?
  - Windows has traditionally only been available for x86 and x64, but ARM support is in early stages.
  - Just like macOS, ARM-based devices use real-time translation to convert x86 and x64 instruction sets to ARM.

Instruction Set Architecture (ISA)

- Real-Time ISA translation is cool, but…
  - …it's very sloooooooooooow.
- Natively-compiled applications will always outperform real-time translation.
  - Think of Python, Java, and C++

Instruction Set Architecture (ISA)

- There are use cases for doing real-time ISA translation outside of major architecture migrations.



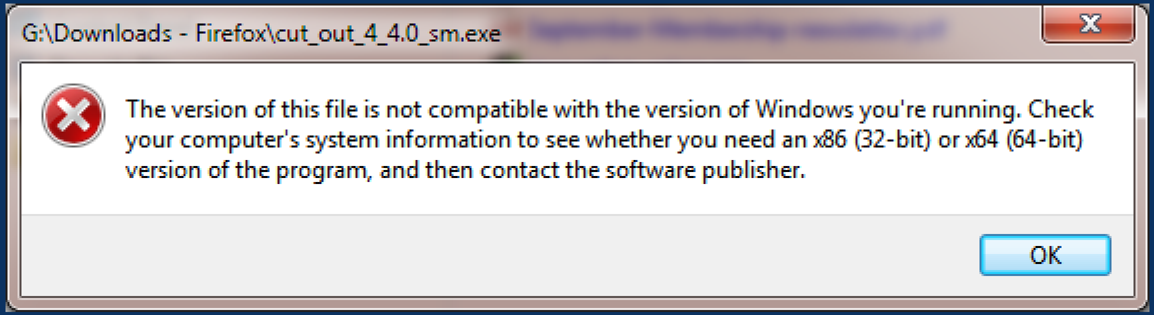
G:\Downloads - Firefox\cut_out_4_4.0_sm.exe

The version of this file is not compatible with the version of Windows you're running. Check your computer's system information to see whether you need an x86 (32-bit) or x64 (64-bit) version of the program, and then contact the software publisher.

- DOSBox – Allows old 16-bit applications to be run on x64-based Windows.
- QMEU – Open-source machine emulator and virtualizer
  - Useful for ISA emulation, especially ARM emulation on x86
  - QEMU used to be used for ARM emulation for Android emulators

Instruction Set Architecture (ISA)

- There are use cases for doing real-time ISA translation outside of major architecture migrations.



G:\Downloads - Firefox\cut_out_4_4.0_sm.exe

The version of this file is not compatible with the version of Windows you're running. Check your computer's system information to see whether you need an x86 (32-bit) or x64 (64-bit) version of the program, and then contact the software publisher.

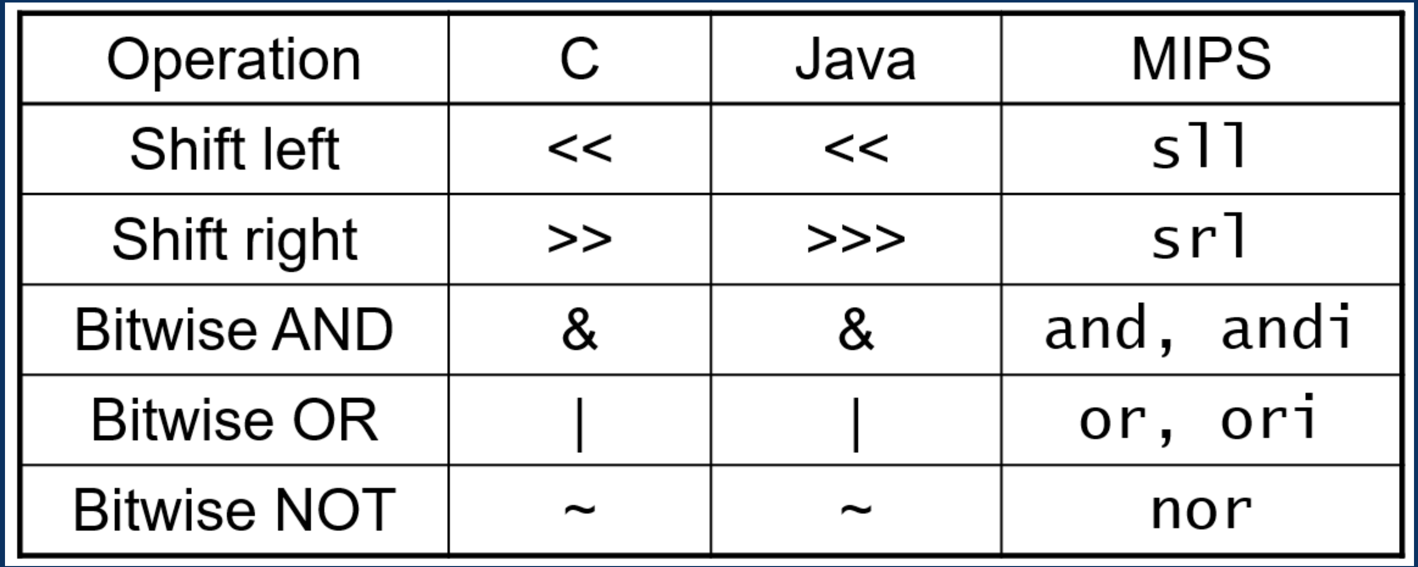
- Parallels – A user-friendly virtual machine software for running Windows on macOS.
  – Parallels does not do ISA translation! You run ARM-based Windows on Parallels and rely on the Windows VM to do it.

Instruction Set Architecture (ISA)

- Other ISAs:
  - MIPS (Microprocessor without Interlocked Pipeline Stages)
    - Used to be quite common for small electronics with battery limitations (sound familiar?)
    - Still used, but not as common anymore
    - This course's textbook uses MIPS for ISA-specific examples
  - Java Runtime Environment
    - Java programs compile to Java Bytecode, which defines the instructions executed by the JRE
    - You could argue that the JRE doesn't function as an ISA since its purpose is to translate Java Bytecode to the hardware's native ISA, but conceptually, it is

Instruction Set Architecture (ISA)

- What instructions do ISAs define?
  - Arithmetic Operations
  - Register Manipulation
  - Memory Manipulation
  - Logical Operations (bitwise manipulation)



Operation	C	Java	MIPS
Shift left	<<	<<	`sll`
Shift right	>>	>>>	`srl`
Bitwise AND	&	&	`and, andi`
Bitwise OR	|	|	`or, ori`
Bitwise NOT	~	~	`nor`

Instruction Set Architecture (ISA)

- What instructions do ISAs define?
  - Conditional Operations
- The specifics of how these operations are performed varies across ISAs.
  - Essentially, ISAs provide different ways of doing the same thing.

Instruction Set Architecture (ISA)

- Design Principles
  1. Simplicity over regularity
     - Regularity makes implementation easier
     - Simplicity enables higher performance at lower cost
  2. Smaller is faster
     - It is faster work with a registers for data manipulation than relying on RAM for everything
     - RAM has more storage space and is slower, registers are faster, but have little storage
  3. Design for the common use case
     - Optimizing for specific use cases usually doesn't make sense because performance is impacted for a different use case.
  4. Compromises are needed to maintain simplicity
     - Example: Use the same number of bits for all instructions even if it is technically not necessary

Instruction Set Architecture (ISA)

- The way ISAs perform certain tasks is not always intuitive but provides meaningful optimization.
  - Example: Negative numbers are stored using the two's compliment.

- For signed integers, the most significant indicates if the value is positive or negative.
  - 0 = positive
  - 1 = negative

Instruction Set Architecture (ISA)

- Intuitively, we may think:
  - Positive 66 = `01000010`
  - Negative 66 = `11000010`
- But actually:
  - Positive 66 = `01000010`
  - Negative 66 = `10111110`
- 66 is what we think, but -66 is not.

Representing a Negative Integer

- Two's compliment:
  1. Invert all bits
  2. Add one to the result of the inversion
  3. The result is the two's compliment version of the integer

- By using the two's compliment, we avoid adding extra complexity to ISAs for identifying integer signage. We just add!

Instruction Set Architecture (ISA)

- Intuitively, we may think:
  - Positive 66 = `01000010`
  - Negative 66 = `11000010`

- Remember:
  - `0 + 0 = 0`
  - `0 + 1 = 1`
  - `1 + 1 = 0, carry 1`
  - `1 + 1 + 1 = 1, carry 1`

```
 1 1    1
  01000010
+ 11000010
----------
  00000100 ==4
```

$4 \neq 0$

Instruction Set Architecture (ISA)

- But actually:
  - Positive 66 = `01000010`
  - Negative 66 = `10111110`
- Remember:
  - `0 + 0 = 0`
  - `0 + 1 = 1`
  - `1 + 1 = 0, carry 1`
  - `1 + 1 + 1 = 1, carry 1`

```
 1111111
  01000010
+ 10111110
----------
  00000000 ==0
```

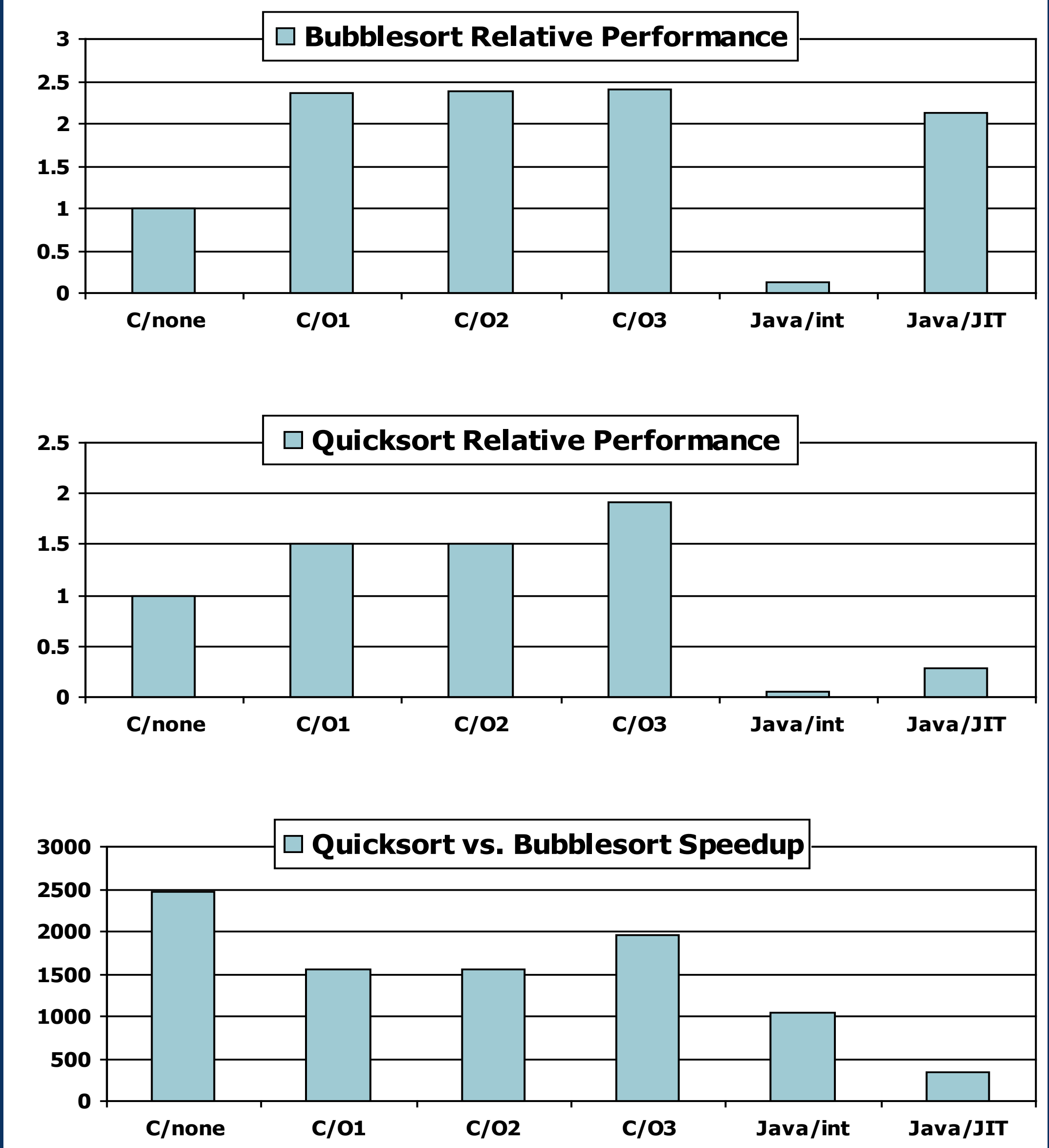
- It's harder for us, but easier for computers to do quickly.

Instruction Set Architecture (ISA)

- As programmers, we have the responsibility to ensure that our code is optimized, but performance is impacted by more than just our algorithm.

- The programming language you use matters too.

Instruction Set Architecture (ISA)

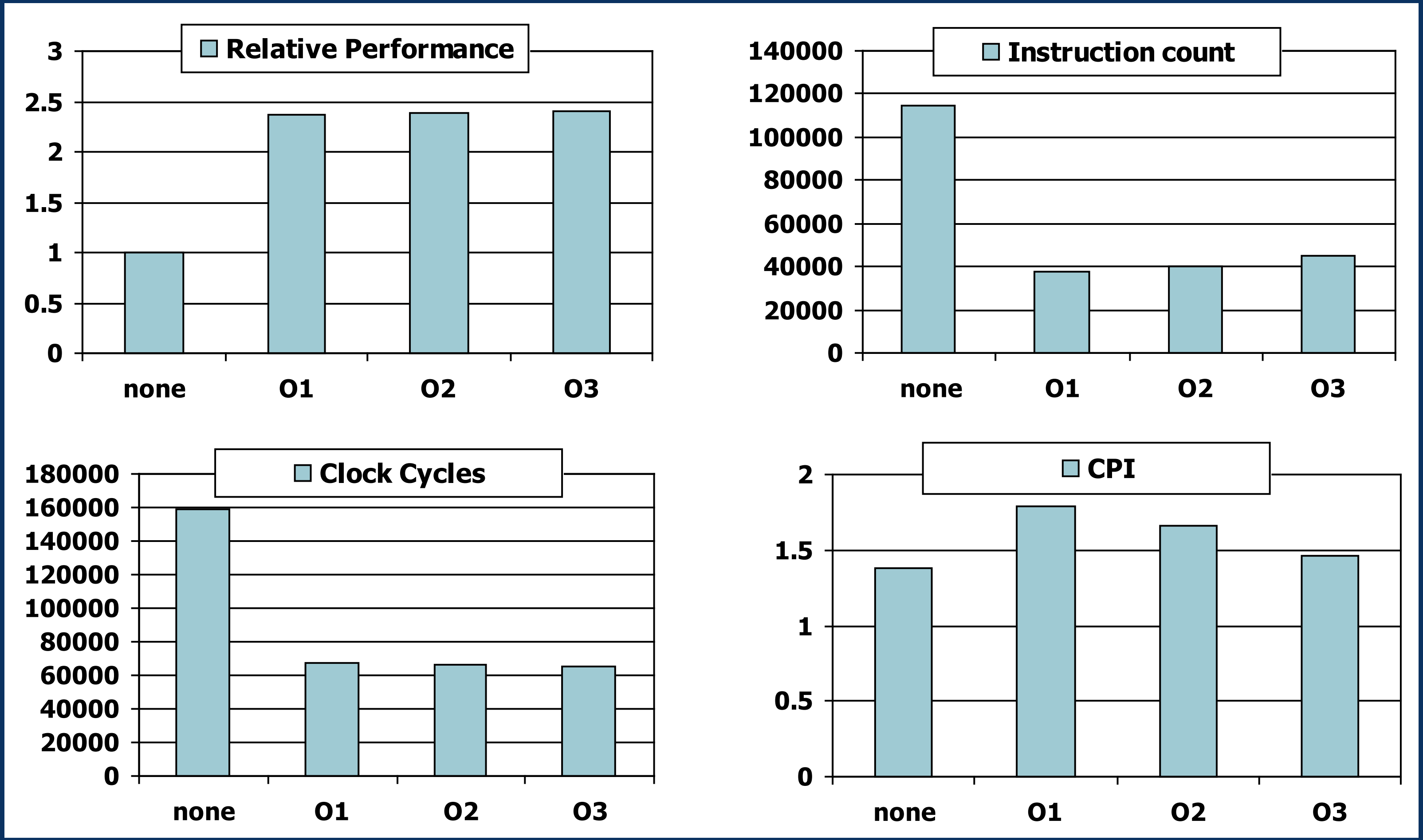


Instruction Set Architecture (ISA)

- Depending on the programming language, your compiler can help too!
  - Remember: Compilers convert high level languages to Assembly, and assemblers convert Assembly to machine code.
  - Example: **gcc**
    - **gcc** is a very popular open-source compiler for C and C++.
    - **gcc** allows users to specify how aggressive the code optimization process is.
  - Tradeoffs:
    - More optimization means longer compile times but faster execution of the compiled code.
    - In some cases, aggressive optimization can cause your program to not work as you intend.

Instruction Set Architecture (ISA)

This example shows the impact of compiling an application with gcc using different optimization levels.

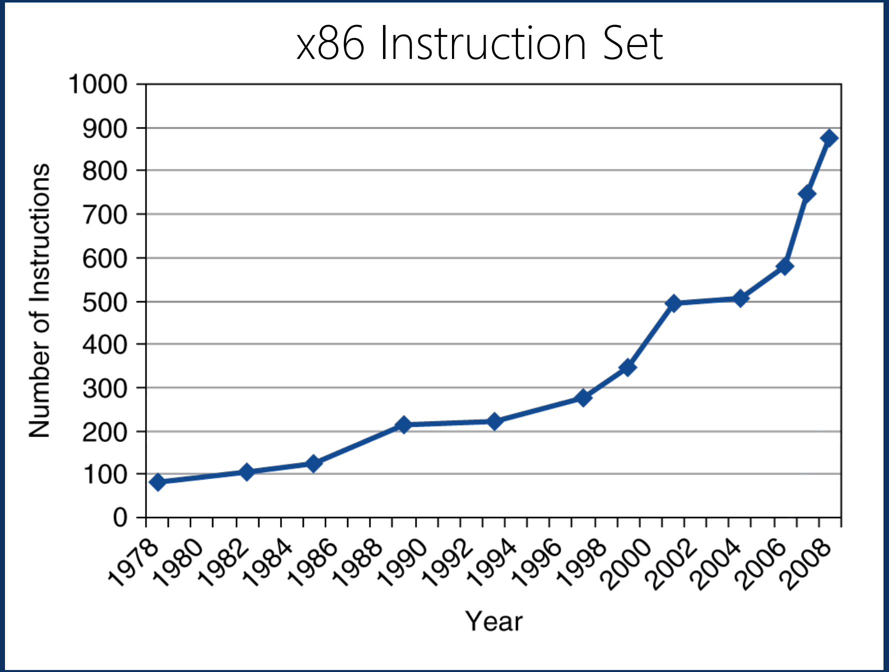


Instruction Set Architecture (ISA)

- Common Misconceptions
  - Powerful instructions lead to better performance
    - Fewer instructions are required for a single task, but it's harder to implement
    - Added complexity can lead to worse performance for all instructions, even simple ones
    - It's better to let compiles handle generation of many simple instructions
  - For best performance, write your code in Assembly
    - For most use cases, compilers are very good at generating optimized Assembly
    - Writing in Assembly is very hard and verbose, which will likely lead to more errors and less developer productivity

Instruction Set Architecture (ISA)

- Common Misconceptions
  - Maintaining backwards compatibility does not cause the instruction set to change.
    - Maintaining backwards compatibility requires new instructions to be added for new features while also retaining old instructions to ensure that old applications work properly.



Instruction Set Architecture (ISA)

- ISAs are classified into two categories
  - Reduced Instruction Set Computing (RISC)
  - Complex Instruction Set Computing (CISC)
- RISC vs CISC trades off where complexity is located.
  - RISC: Fewer instructions available, which makes compiling harder since more verbosity is required in the resulting Assembly.
  - CISC: More instructions available, which makes compiling easier since less verbosity is required.

- x86 is CISC
- ARM is RISC

Instruction Set Architecture (ISA)

- ISAs—especially complex ones—can be dangerous.
  - Two undocumented Intel x86 instructions discovered that can be used to modify microcode
  - ISA implementations may unintentionally allow for instructions to execute that are not supposed to be valid.
    - We call this an illegal opcode.
    - At worst, illegal opcodes can be security risks, at best, they can provide unintended extra functionality that turns out to be useful.

Instruction Set Architecture (ISA)

- Concluding Points
  - Fundamentally, all ISAs serve the same purpose.
  - When developing ISAs, we make tradeoffs between what is easy and what is fast.
  - Compatibility between ISAs adds complexity.
  - ISAs should be designed for general best performance.
    - Exception: When designing an ISA that is intended for a very specific use case, it makes sense to optimize for it.
    - Example: GPUs are CPUs optimized for matrix math and thus, it would make sense for GPU ISAs to be optimized to be really good at matrix math at the expense of other types of operations.