x86-64 Programming I

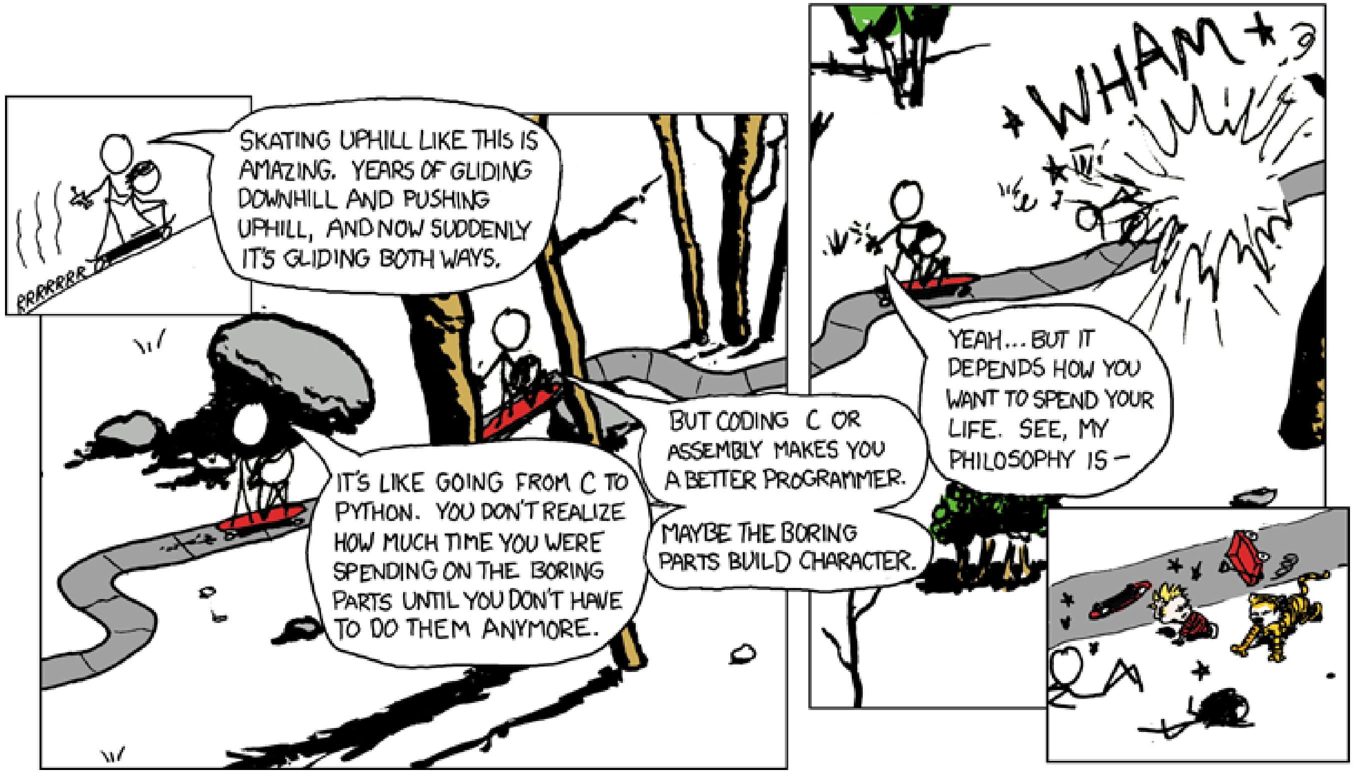
CSE 351 Autumn 2023

Instructor:

Justin Hsia

Teaching Assistants:

Afifah Kashif
Bhavik Soni
Cassandra Lam
Connie Chen
David Dai
Dawit Hailu
Ellis Haker
Eyoel Gebre
Joshua Tan
Malak Zaki
Naama Amiel
Nayha Auradkar
Nikolas McNamee
Pedro Amarante
Renee Ruan
Simran Bagaria
Will Robertson


RRRRRRR
SKATING UPHILL LIKE THIS IS AMAZING. YEARS OF GLIDING DOWNHILL AND PUSHING UPHILL, AND NOW SUDDENLY IT'S GLIDING BOTH WAYS.
IT'S LIKE GOING FROM C TO PYTHON. YOU DON'T REALIZE HOW MUCH TIME YOU WERE SPENDING ON THE BORING PARTS UNTIL YOU DON'T HAVE TO DO THEM ANYMORE.
BUT CODING C OR ASSEMBLY MAKES YOU A BETTER PROGRAMMER.
MAYBE THE BORING PARTS BUILD CHARACTER.
YEAH... BUT IT DEPENDS HOW YOU WANT TO SPEND YOUR LIFE. SEE, MY PHILOSOPHY IS –
WHAM

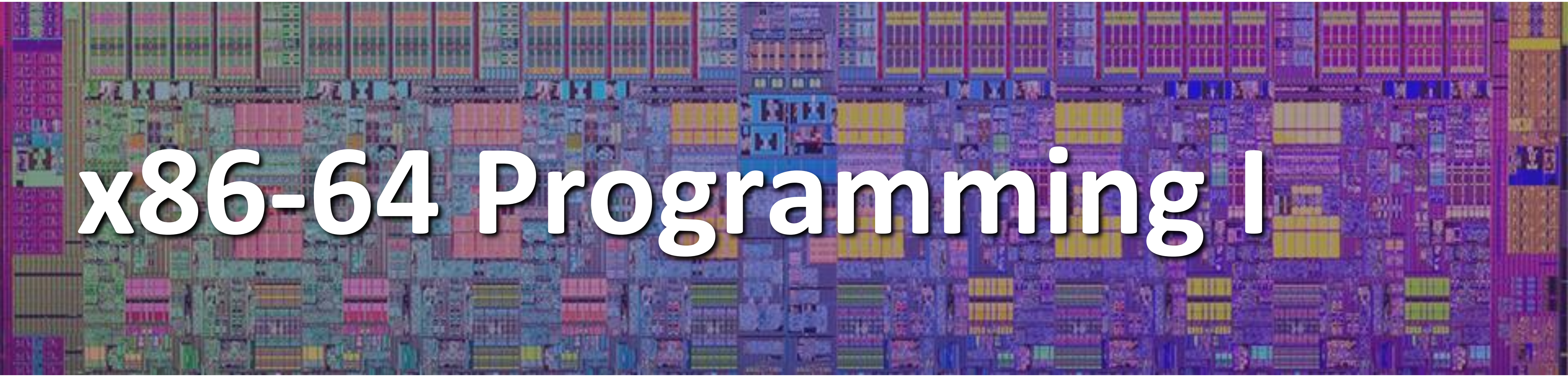
http://xkcd.com/409/

Relevant Course Information

- hw6 due Friday, hw7 due Monday
- Lab 1a: last chance to submit is tonight @ 11:59 pm
  - One submission per partnership
  - Make sure you check the Gradescope autograder output!
  - Grades hopefully released by end of Sunday (10/15)
- Lab 1b due Monday (10/16)
  - Submit `aisle_manager.c`, `store_client.c`, and `lab1Bsynthesis.txt`
  - Section tomorrow should help with Lab 1b

Getting Help with 351

- Lecture recordings, lessons, inked slides, section worksheet solutions
- Attend lectures and support hours
  - Can also chat with other students– help each other learn!
- Form a study group!
  - Good for everything but labs, which should be done in pairs
  - Communicate regularly, use the class terminology, ask and answer each others' questions, show up to SH together
- Post on Ed Discussion
- Request a 1-on-1 meeting
  - Available on a limited basis for special circumstances



x86-64 Programming I

Lesson Summary (1/2)

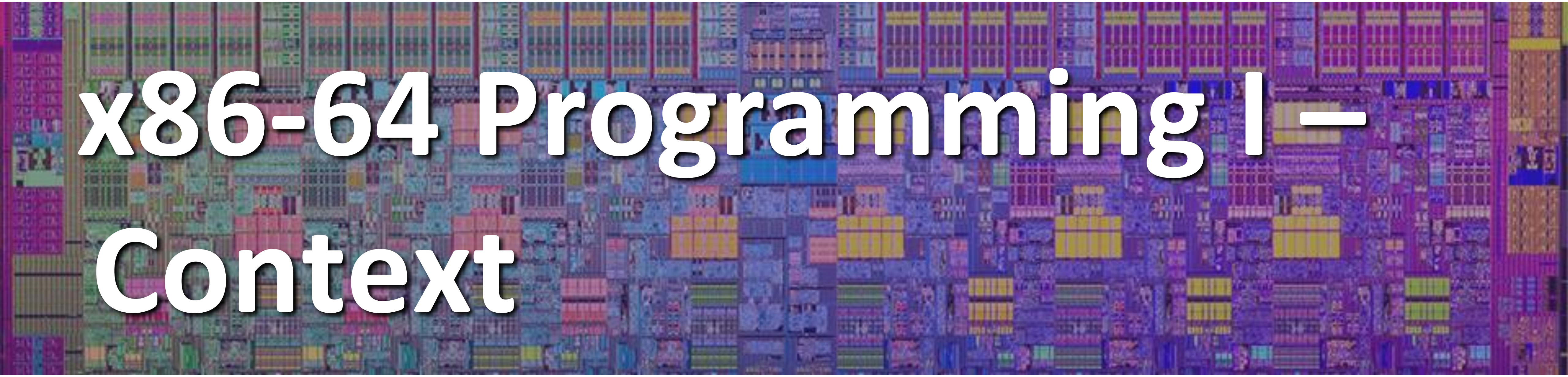
- x86-64 is a complex instruction set computing (CISC) architecture
  - There are 3 types of instructions in x86-64
    - Data transfer (`mov`), Arithmetic, Control Flow
    - Fixed width specified by size suffix: `b` (1 byte), `w` (2 bytes), `l` (4 bytes), or `q` (8 bytes)
  - There are 3 types of operands in x86-64
    - Immediate (`$`) are literals
    - Register (`%`) is one of 16 general-purpose integer register names (or sub-register names)
    - Memory(`()`) is a way to express an address

Lesson Summary (2/2)

- Terminology:
  - Instruction Set Architecture (ISA): CISC vs. RISC
  - Instructions: data transfer, arithmetic/logical, control flow
    - Size specifiers: `b`, `w`, `l`, `q`
  - Operands: immediates, registers, memory

- Learning Objectives:
  - Without executing, describe the overall purpose of snippets of x86-64 assembly code containing arithmetic, [if-else statements, and/or loops].

- What lingering questions do you have from the lesson?



x86-64 Programming I – Context

Instruction Set Philosophies, Revisited

- *Complex Instruction Set Computing* (CISC):
  Add more and more elaborate and specialized instructions as needed
  - **Design goals:** complete tasks in as few instructions as possible; minimize memory accesses for instructions

- *Reduced Instruction Set Computing* (RISC):
  Keep instruction set small and regular
  - **Design goals:** build fast hardware; instructions should complete in few clock cycles (ideally 1); minimize complexity and maximize performance

- How different are these two philosophies, really?

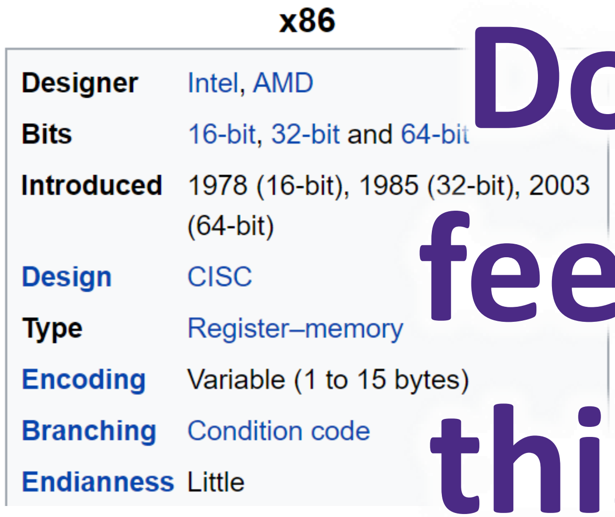
Instruction Set Philosophies, Revisited

- *Complex Instruction Set Computing* (CISC):
  Add more and more elaborate and specialized instructions as needed
  - **Design goals:** complete tasks in as few instructions as possible; minimize memory accesses for instructions

- *Reduced Instruction Set Computing* (RISC):
  Keep instruction set small and regular
  - **Design goals:** build fast hardware; instructions should complete in few clock cycles (ideally 1); minimize complexity and maximize performance

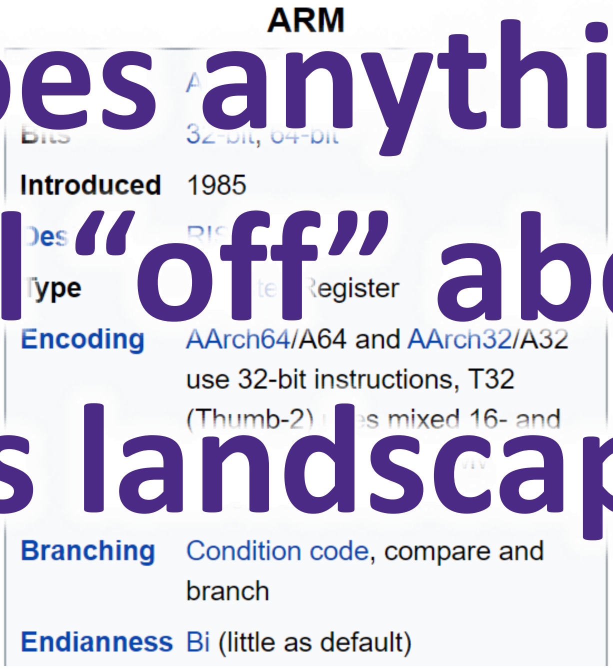
- How different are these two philosophies, really?
  - Both pursue **efficiency** (**minimalism** is a means to an end)

Mainstream ISAs, Revisited



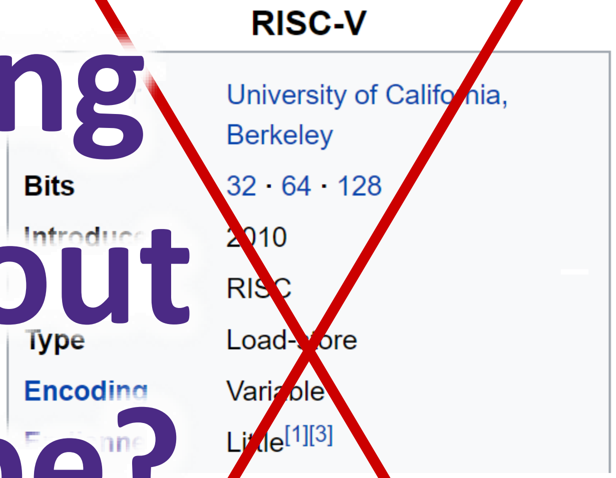
x86

Designer	Intel, AMD
Bits	16-bit, 32-bit and 64-bit
Introduced	1978 (16-bit), 1985 (32-bit), 2003 (64-bit)
Design	CISC
Type	Register–memory
Encoding	Variable (1 to 15 bytes)
Branching	Condition code
Endianness	Little



ARM

[illegible]	A[illegible]
Bits	32-bit, 64-bit
Introduced	1985
Des[illegible]	RI[illegible]
Type	[illegible] Register
Encoding	AArch64/A64 and AArch32/A32 use 32-bit instructions, T32 (Thumb-2) [illegible]s mixed 16- and [illegible]
Branching	Condition code, compare and branch
Endianness	Bi (little as default)



RISC-V

[illegible]	University of California, Berkeley
Bits	32 · 64 · 128
Introduc[illegible]	2010
[illegible]	RISC
Type	Load–store
Encoding	Variable
E[illegible]	Little[1][3]

Does anything feel "off" about this landscape?

Macbooks & PCs
(Core i3, i5, i7, M)
x86-64 Instruction Set

Smartphone-like devices
(iPhone, iPad, Raspberry Pi)
ARM Instruction Set

~~Mostly research~~
~~(some traction in embedded)~~
~~RISC-V Instruction Set~~

Tech Monopolization

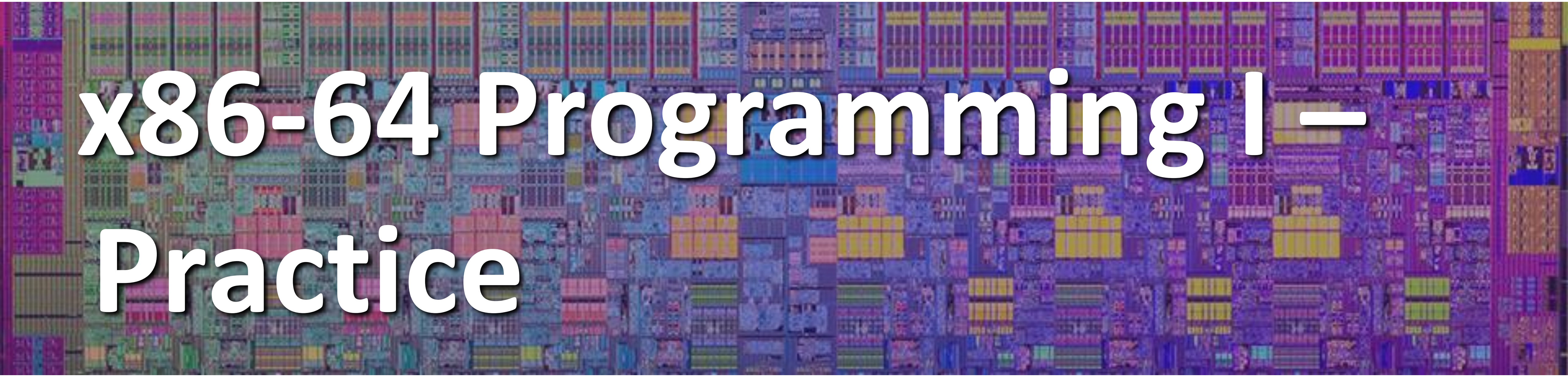
- How many “dominant” ISAs are there?
- How many “dominant” phone brands are there?
- How many “dominant” operating systems are there?
- How many “dominant” chip manufacturers are there?
- It wasn’t always this way!
  - Combination of antitrust policies and (lack of) enforcement

Discussion Question

- Discuss the following question(s) in groups of 3-4 students
  - I will call on a few groups afterwards so please be prepared to share out
  - Be respectful of others' opinions and experiences

- How do you feel about tech monopolization?
  - What are the benefits and disadvantages of this landscape for (1) the monopolizing companies and (2) the consumers?

  - These big tech companies are now worth billions of dollars. What might we try if we wanted to break up the monopolization?



x86-64 Programming I – Practice

Group Work Time

- During this time, you are encouraged to work on the following:
  1) If desired, continue your discussion
  2) Work on the lesson problems (solutions at the end of class)
  3) Work on the homework problems

- Resources:
  - You can revisit the lesson material
  - Work together in groups and help each other out
  - Course staff will circle around to provide support

Practice Questions (1/2)

- Assume that the register `%rax` currently holds the value `0x 01 02 03 04 05 06 07 08`

- Answer the questions on Ed Lessons about the following instruction (<instr> <src> <dst>):

```
xorw $-1, %ax
```

  - Operation type:
  - Operand types:
  - Operation width:
  - (extra) Result in `%rax`:

Practice Questions (2/2)

- Which of the following are valid implementations of `rcx = rax + rbx`?

  -
    ```
    addq %rax, %rcx
    addq %rbx, %rcx
    ```

  -
    ```
    movq %rax, %rcx
    addq %rbx, %rcx
    ```

  -
    ```
    movq $0, %rcx
    addq %rbx, %rcx
    addq %rax, %rcx
    ```

  -
    ```
    xorq %rax, %rax
    addq %rax, %rcx
    addq %rbx, %rcx
    ```