Computer Architecture Introduction

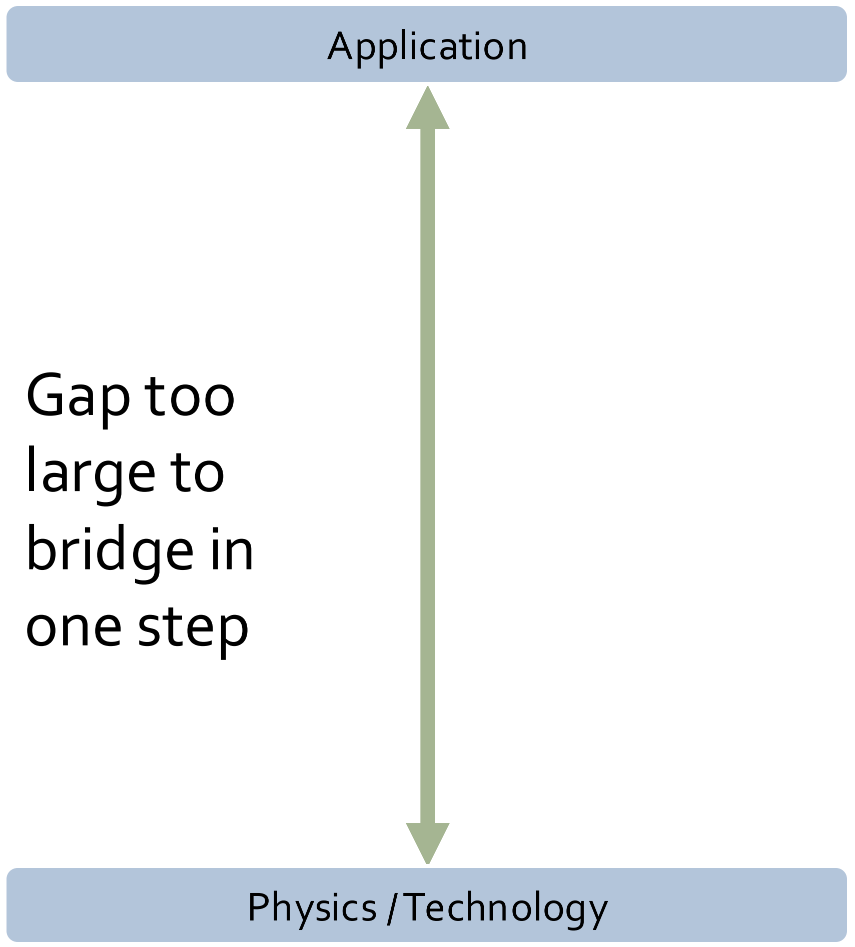
Ting-Jung Chang

NYCU CS

Agenda

- What is Computer Architecture?
- Course Admin
- ISA Review

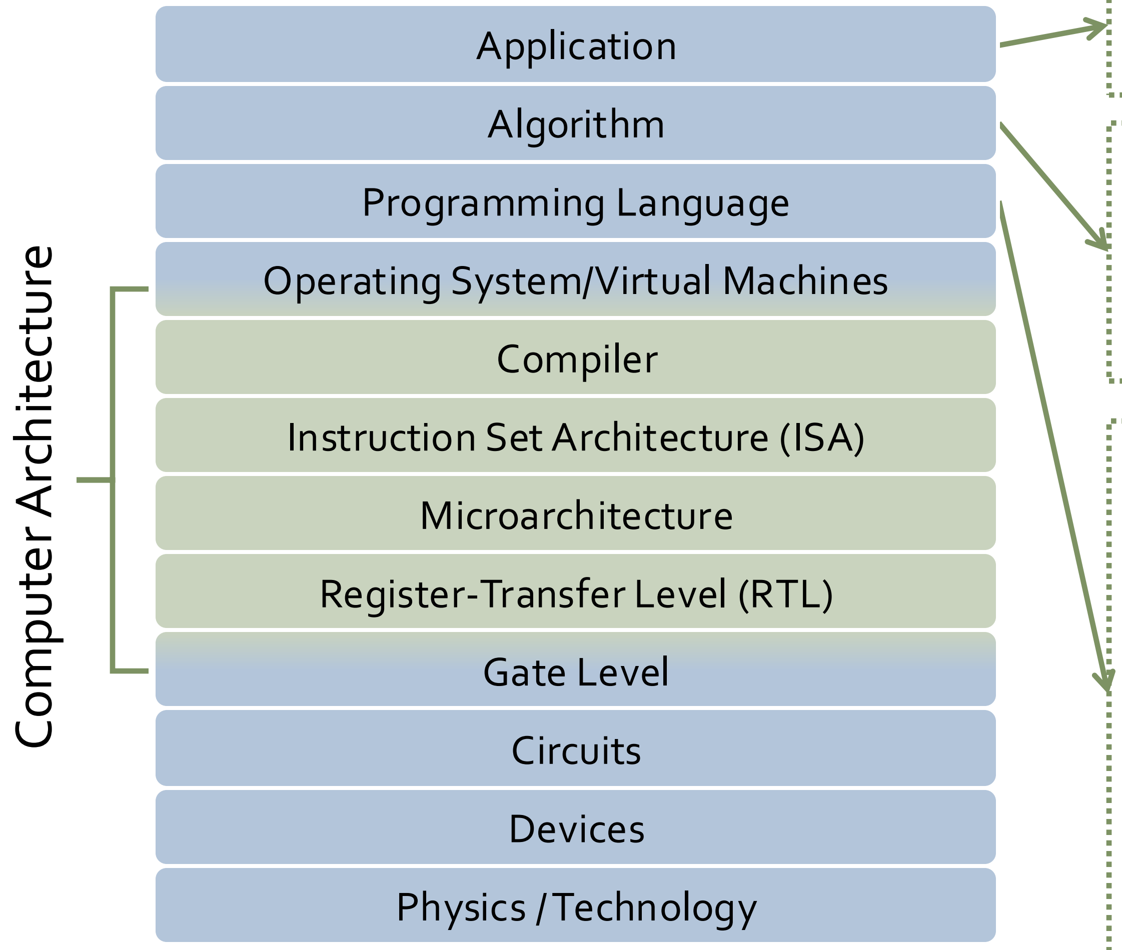
What is Computer Architecture?



Gap too large to bridge in one step

In its broadest definition, computer architecture is the design of the abstraction/ implementation layers that allow us to execute information processing applications efficiently using manufacturing technologies.

The Computer Systems Stack



Sort an array of numbers

2,6,3,8,4,5 → 2,3,4,5,6,8

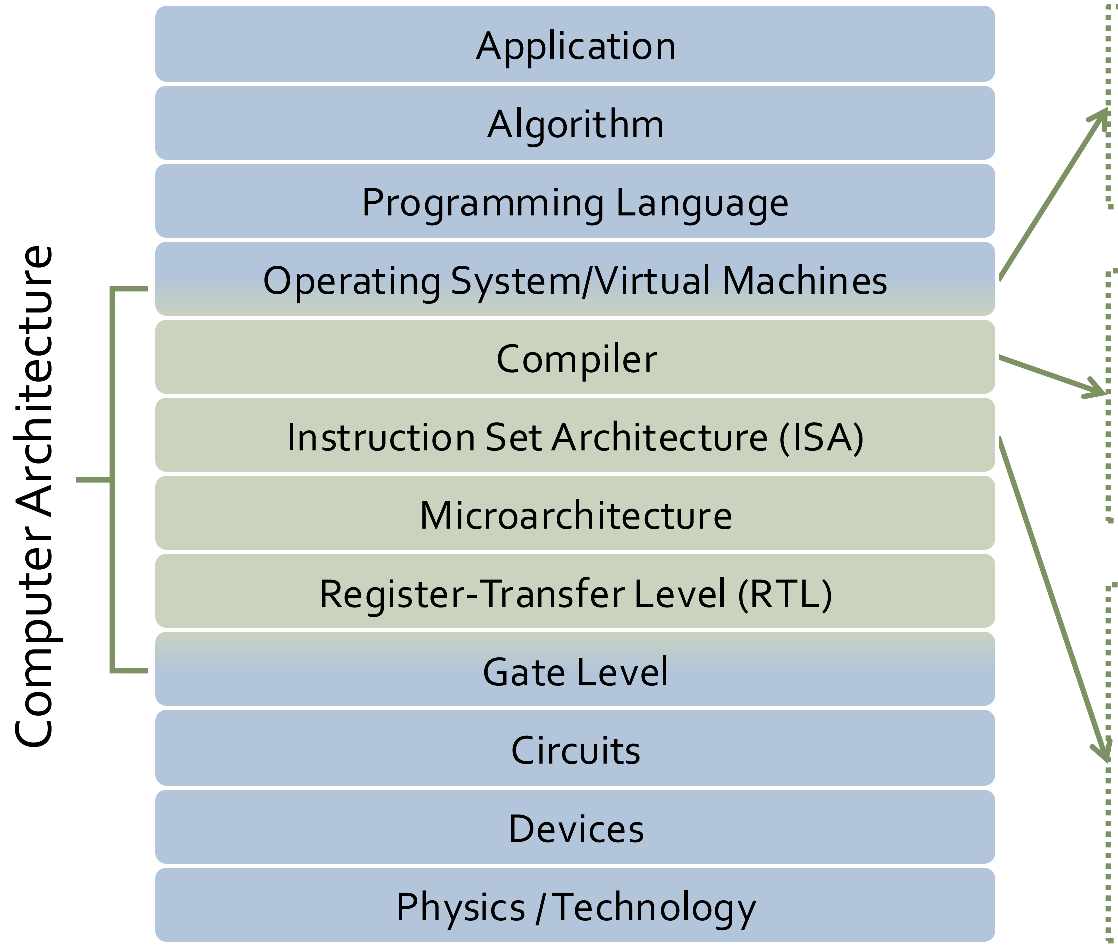
Insertion sort algorithm

1. Find minimum number in input array
2. Move minimum number into output array
3. Repeat steps 1 and 2 until finished

C implementation of insertion sort

```
void isort( int b[], int a[], int n ) {
 for ( int idx, k = 0; k < n; k++ ) {
  int min = 100;
  for ( int i = 0; i < n; i++ ) {
   if ( a[i] < min ) {
    min = a[i];
    idx = i;
   }
  }
  b[k] = min;
  a[idx] = 100;
 }
}
```

The Computer Systems Stack



Mac OS X, Windows, Linux

Handles low-level hardware management

C Compiler

Transform programs into assembly

```
int a = b + c;
A [i] = a;
```

$\Rightarrow$

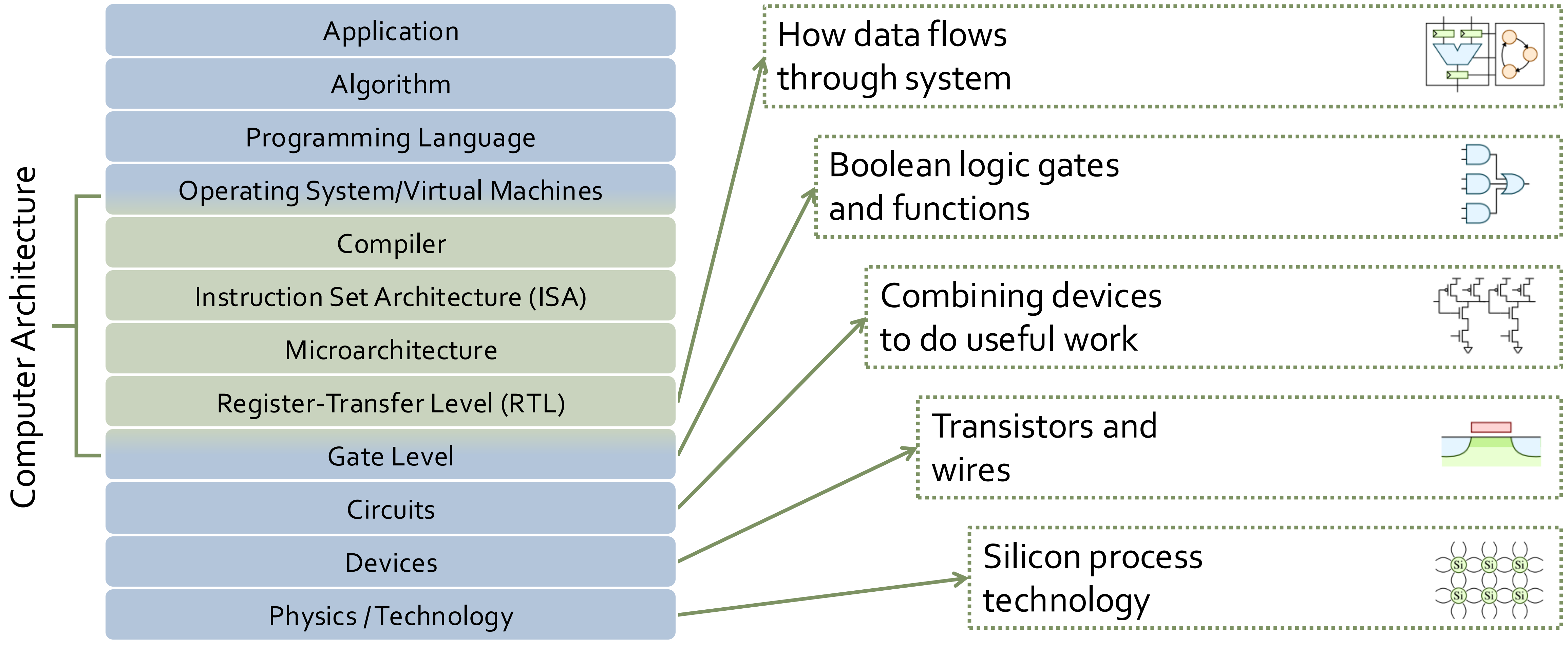
```
add x1, x2, x3
sw  x1, 0(x4)
```

RISC-V Instruction Set

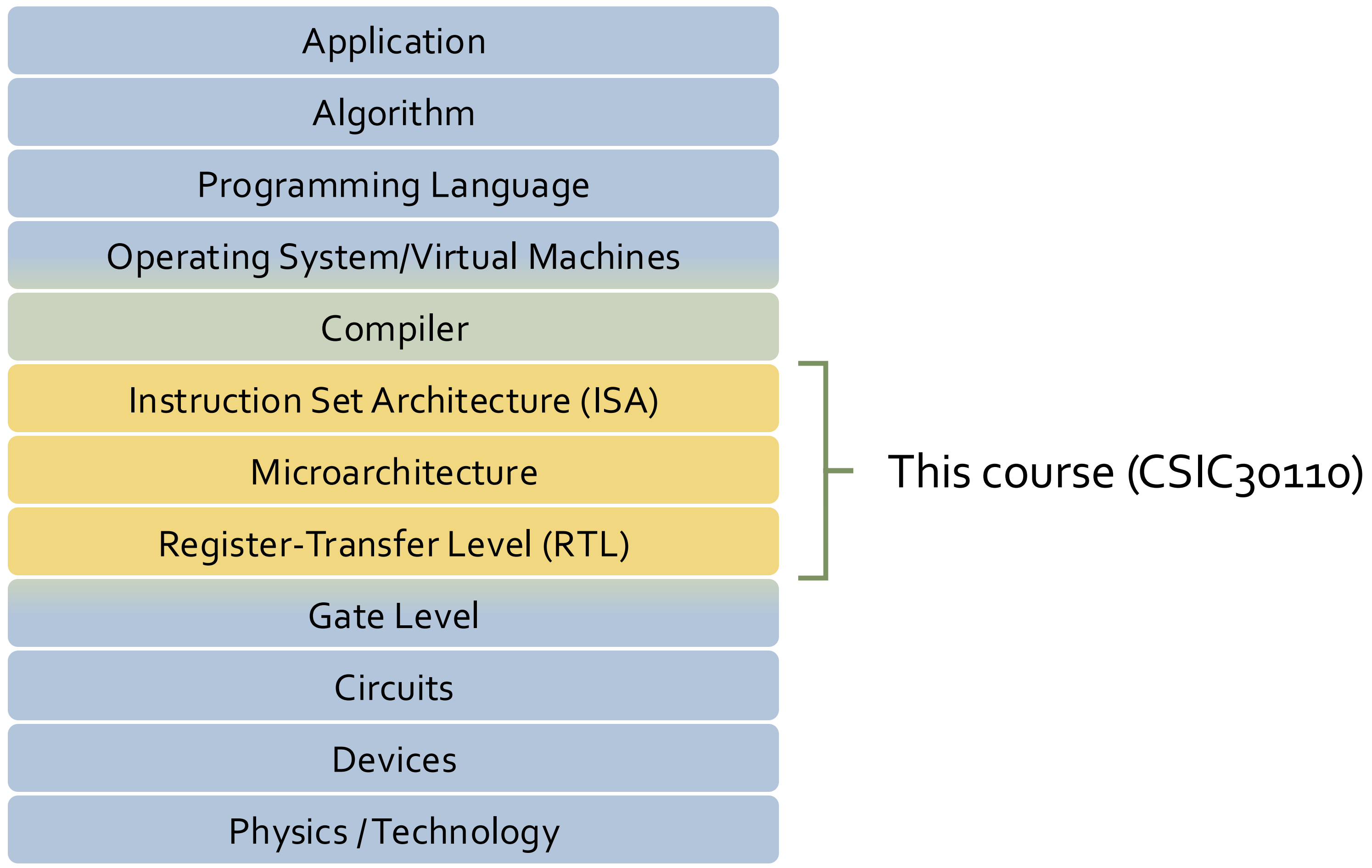
Instructions that machine executes

```
li x12, 1024
lw x2, 0(x12)
addi x13, x12,4
lw x3, 0(x13)
add x4, x2, x3
sw x4, 4(x13)
```

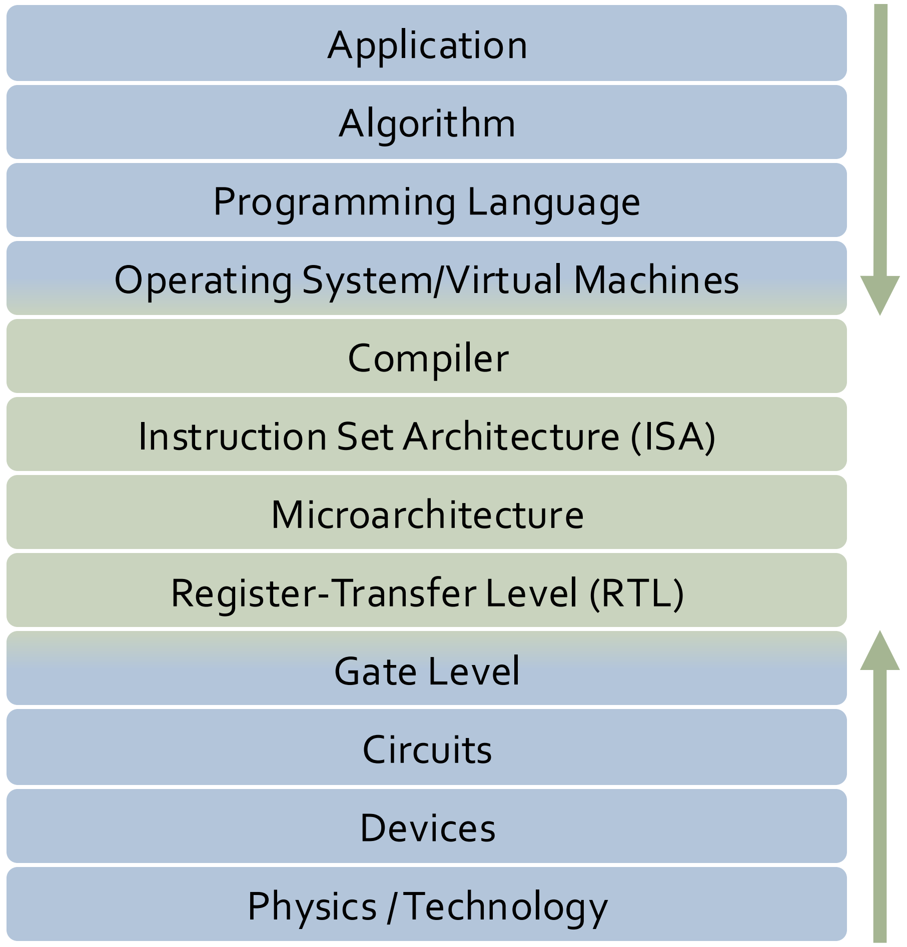
The Computer Systems Stack


Application
Algorithm
Programming Language
Operating System/Virtual Machines
Compiler
Instruction Set Architecture (ISA)
Microarchitecture
Register-Transfer Level (RTL)
Gate Level
Circuits
Devices
Physics / Technology
Computer Architecture
How data flows through system
Boolean logic gates and functions
Combining devices to do useful work
Transistors and wires
Silicon process technology
Si
Si
Si
Si
Si
Si

The Computer Systems Stack



Architecture is Constantly Changing



Application Requirements:

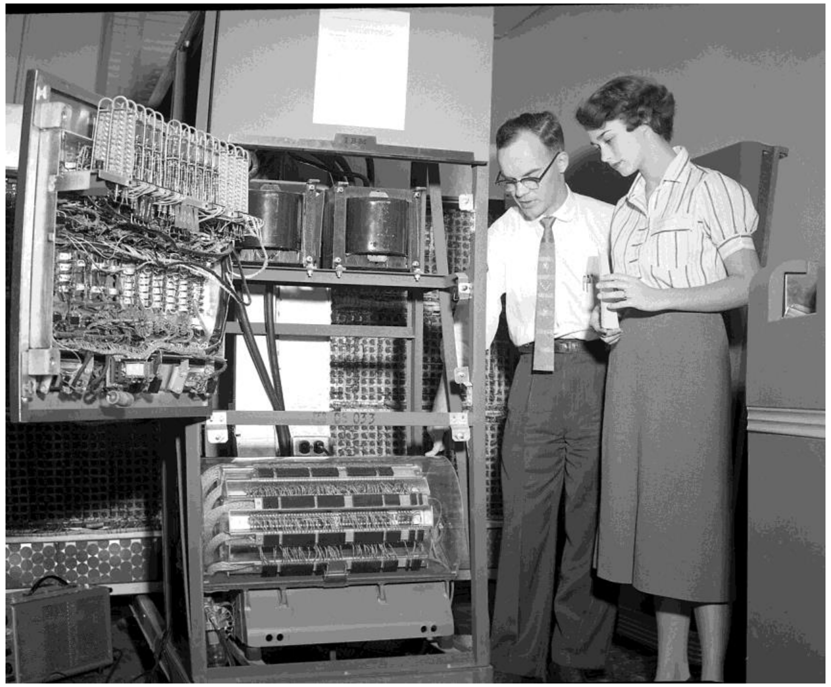
- Suggest how to improve architecture
- Provide revenue to fund development

Architecture provides feedback to guide application and technology research directions

Technology Constraints:

- Restrict what can be done efficiently
- New technologies make new arch possible

Computers Then…



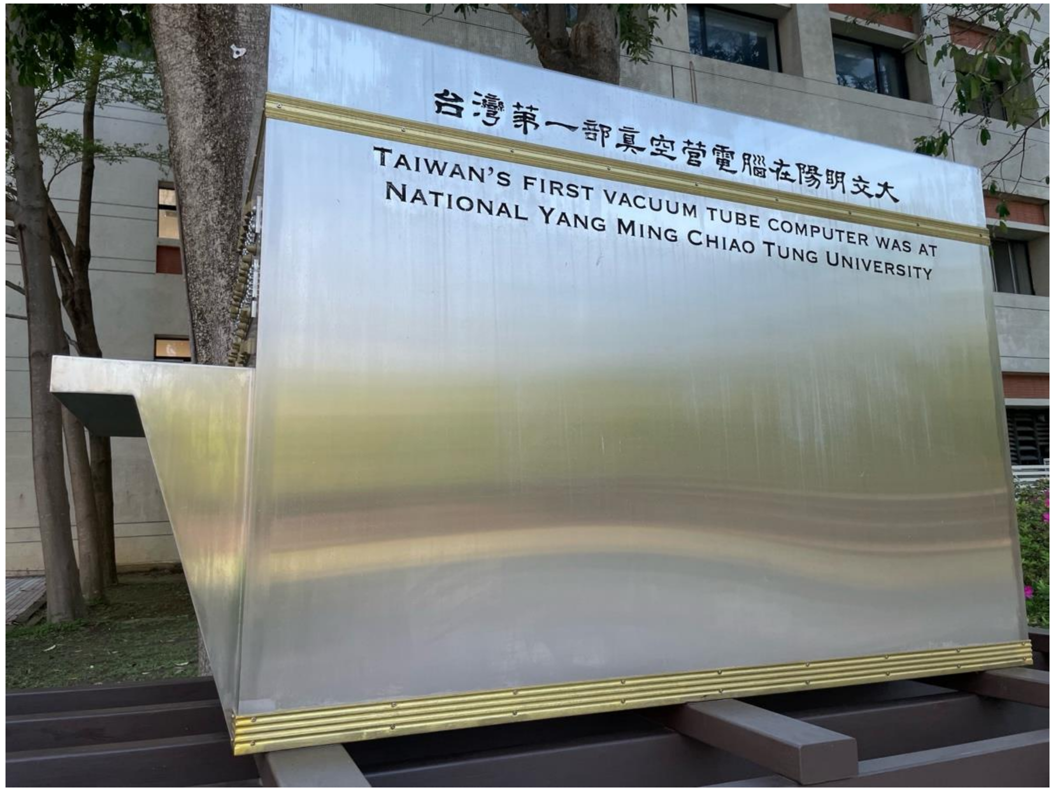
IBM 650, 1962, NCTU

- The first mass-produced computer
- Almost 2,000 produced

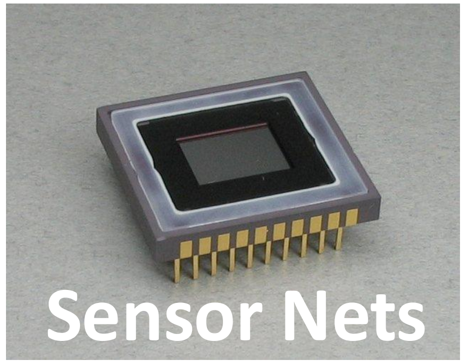
[Cushing Memorial Library and Archives, Texas A&M, Creative Commons Attribution 2.0 Generic]

Computers Then…

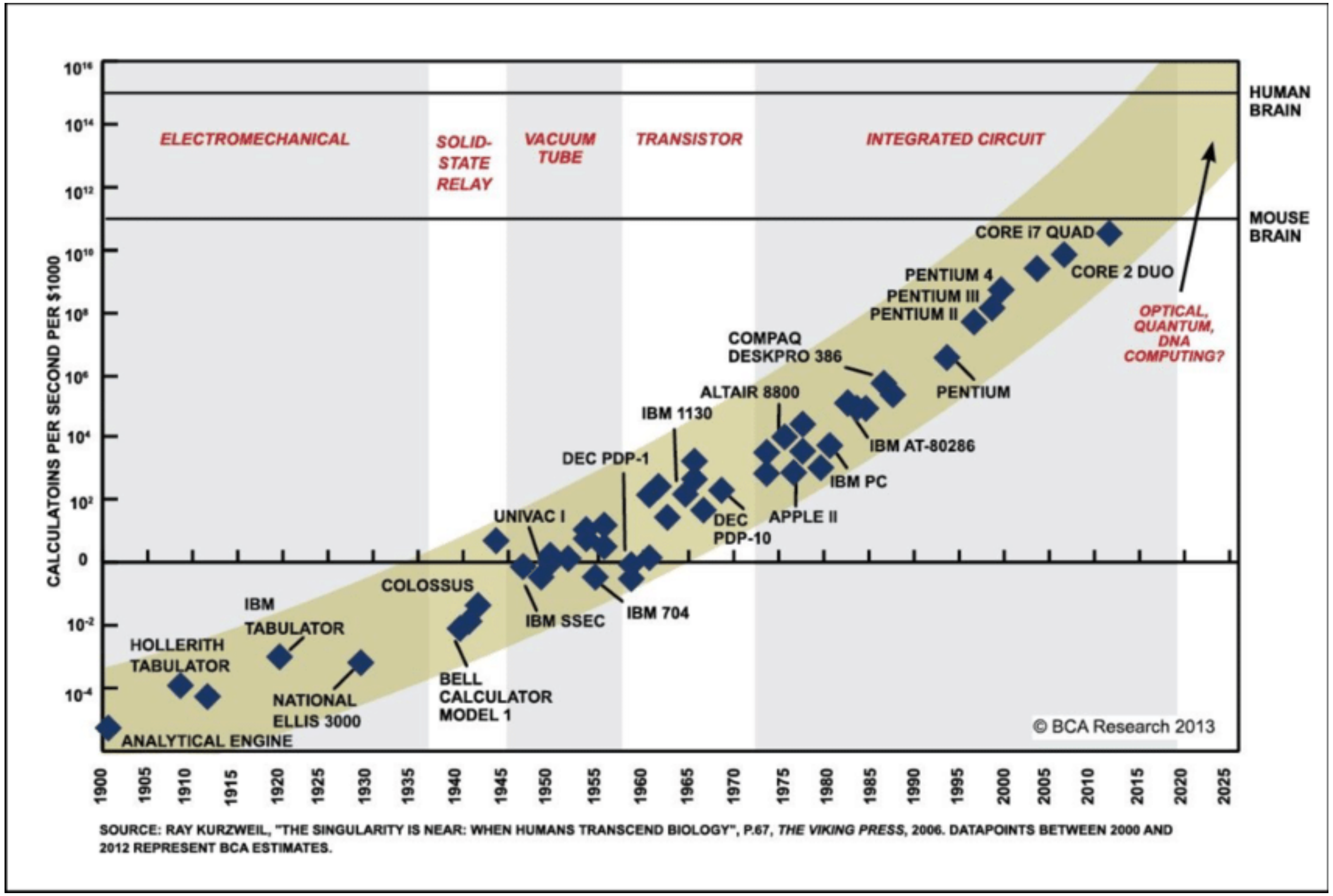
台灣第一部真空管電腦在陽明交大
TAIWAN'S FIRST VACUUM TUBE COMPUTER WAS AT NATIONAL YANG MING CHIAO TUNG UNIVERSITY


台灣第一部真空管電腦在陽明交大
TAIWAN'S FIRST VACUUM TUBE COMPUTER WAS AT
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Computers Now…

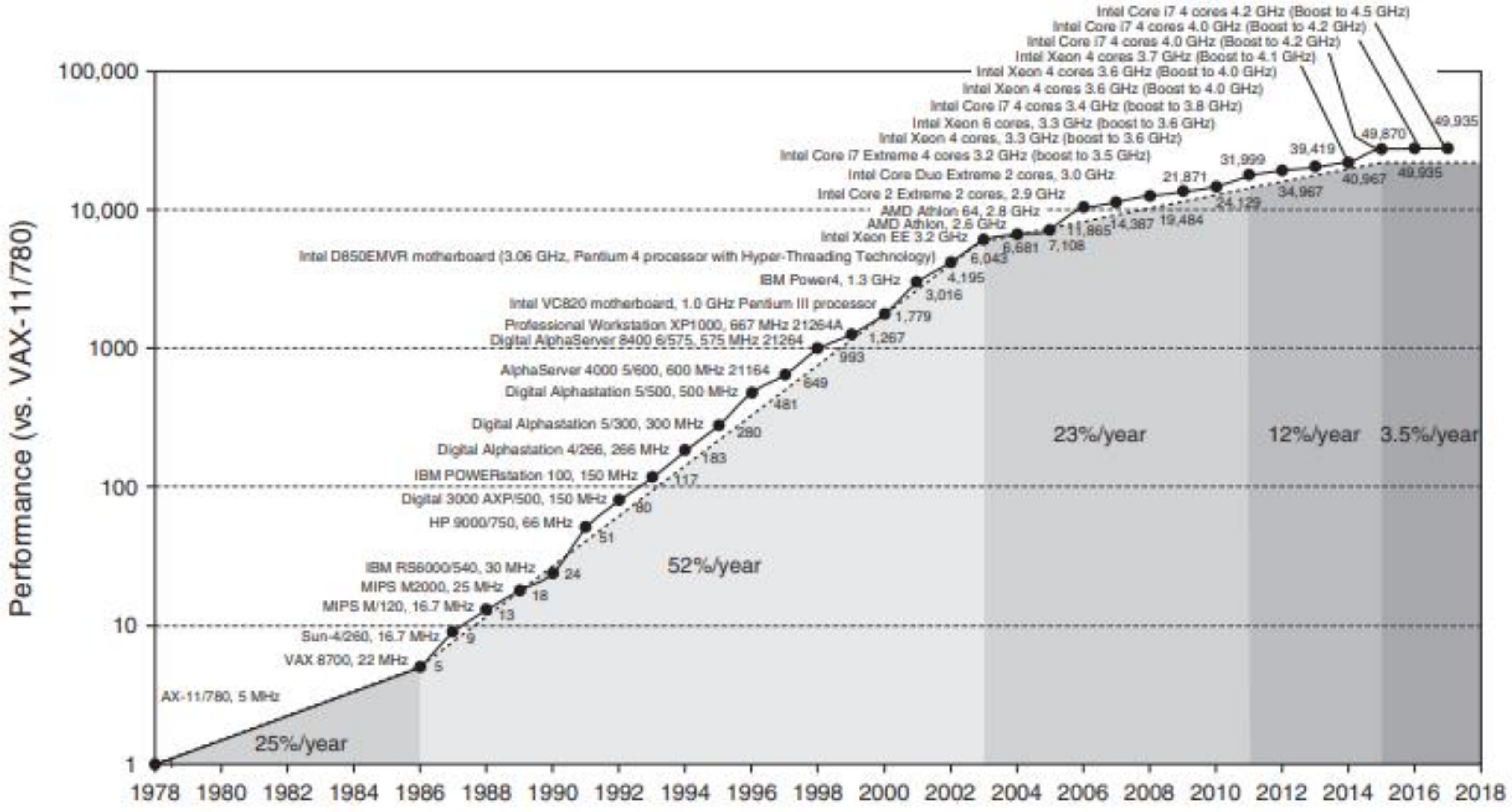


Moore’s Law

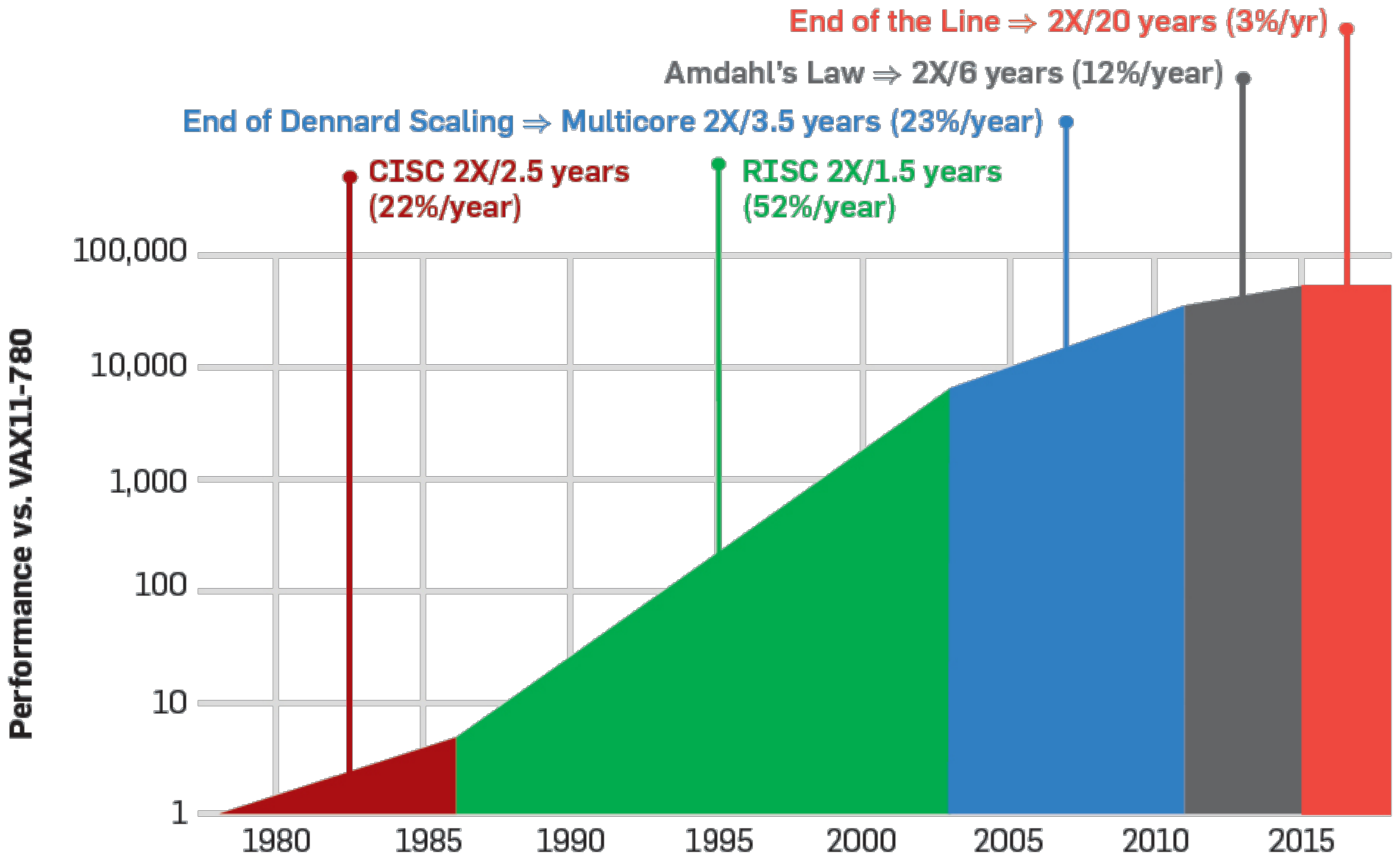


SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Sequential Processor Performance

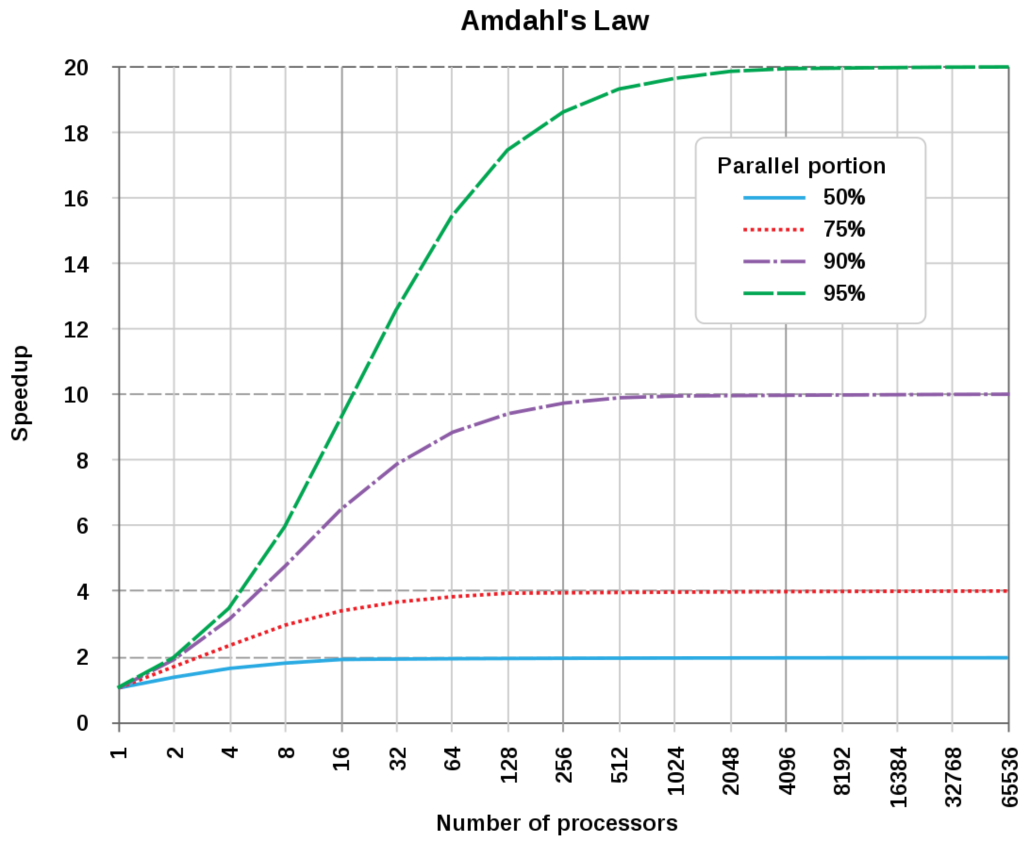


From Hennessy and Patterson Ed. 6 Image Copyright © 2019, Elsevier Inc. All rights reserved.


End of the Line ⇒ 2X/20 years (3%/yr)
Amdahl's Law ⇒ 2X/6 years (12%/year)
End of Dennard Scaling ⇒ Multicore 2X/3.5 years (23%/year)
CISC 2X/2.5 years (22%/year)
RISC 2X/1.5 years (52%/year)
Performance vs. VAX11-780
100,000
10,000
1,000
100
10
1
1980
1985
1990
1995
2000
2005
2010
2015

Amdahl's Law

- Speedup= $\dfrac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$



Upheaval in Computer Design

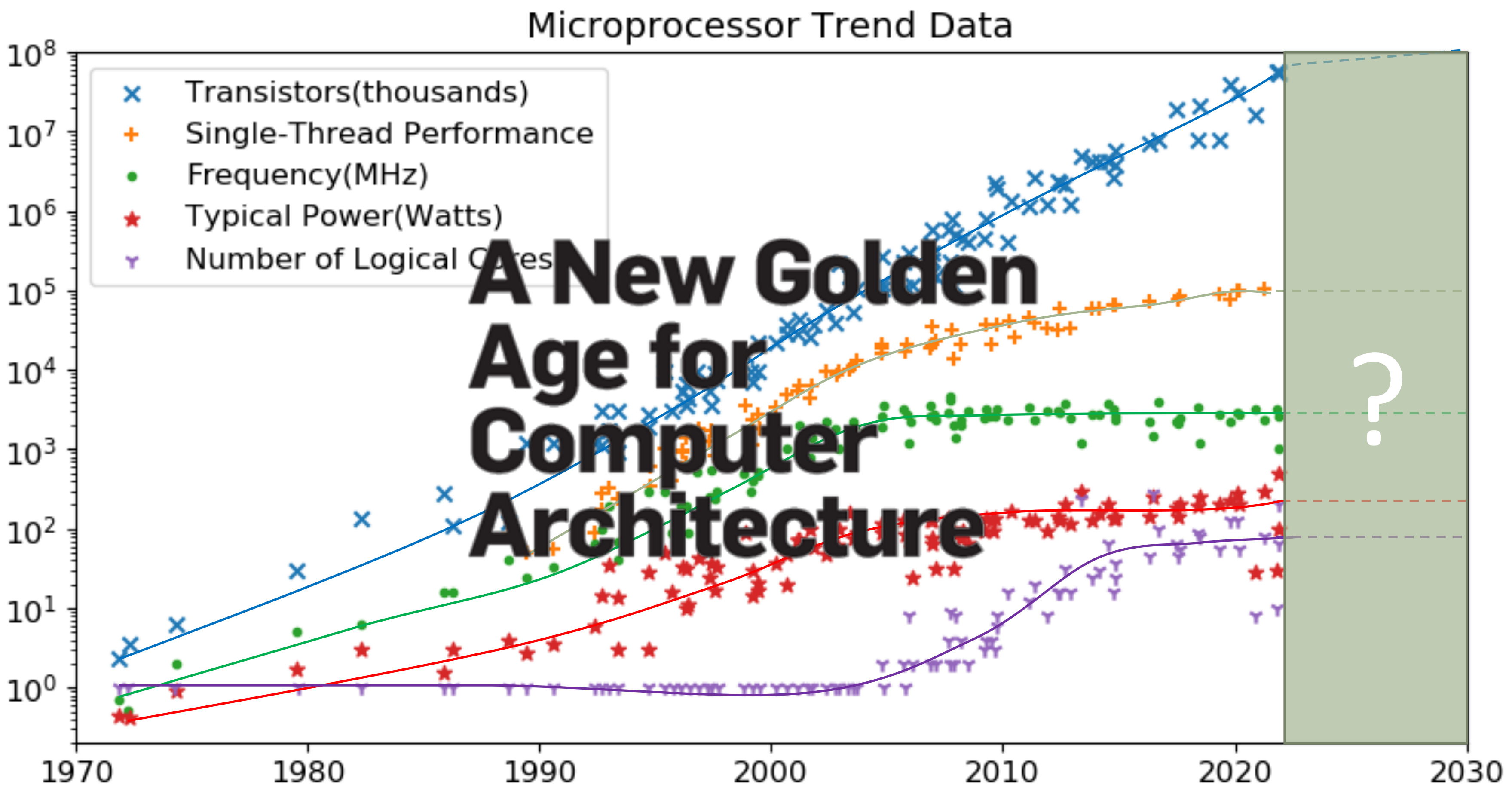
- Most of last 50 years, Moore’s Law ruled
    - Scaling improved performance/energy without changing software model
- Last decade, technology scaling slowed/stopped
    - Dennard scaling is over (supply voltage ~fixed)
    - Moore’s Law (cost/transistor) over?
    - No competitive replacement for CMOS in anytime soon
    - Energy efficiency is the main limiter
- 2020s shift
    - AI/ML drives compute demand
- No “free lunch” for software developers, must consider
    - Parallelism and heterogeneity are mandatory

Today’s Dominant Target Systems

- Mobile (smartphone/tablet)
  - >1 billion sold/year
  - Dominated by ARM ISA in SoCs
  - Ship with AI/Neural engines + accelerators (vision, audio, security, sensors)
- Warehouse-Scale Computers (WSCs)
  - 100k+ cores per warehouse, cloud datacenters
  - Dominated by x86 ISA (server CPUs) + custom accelerators
  - Energy & carbon footprint now a key bottleneck
- Embedded and Edge
  - Consumer electronics, automotive, IoT
  - Strong RISC-V growth in microcontrollers & AI edge chips
  - Edge AI (TinyML, on-device generative AI) expanding rapidly

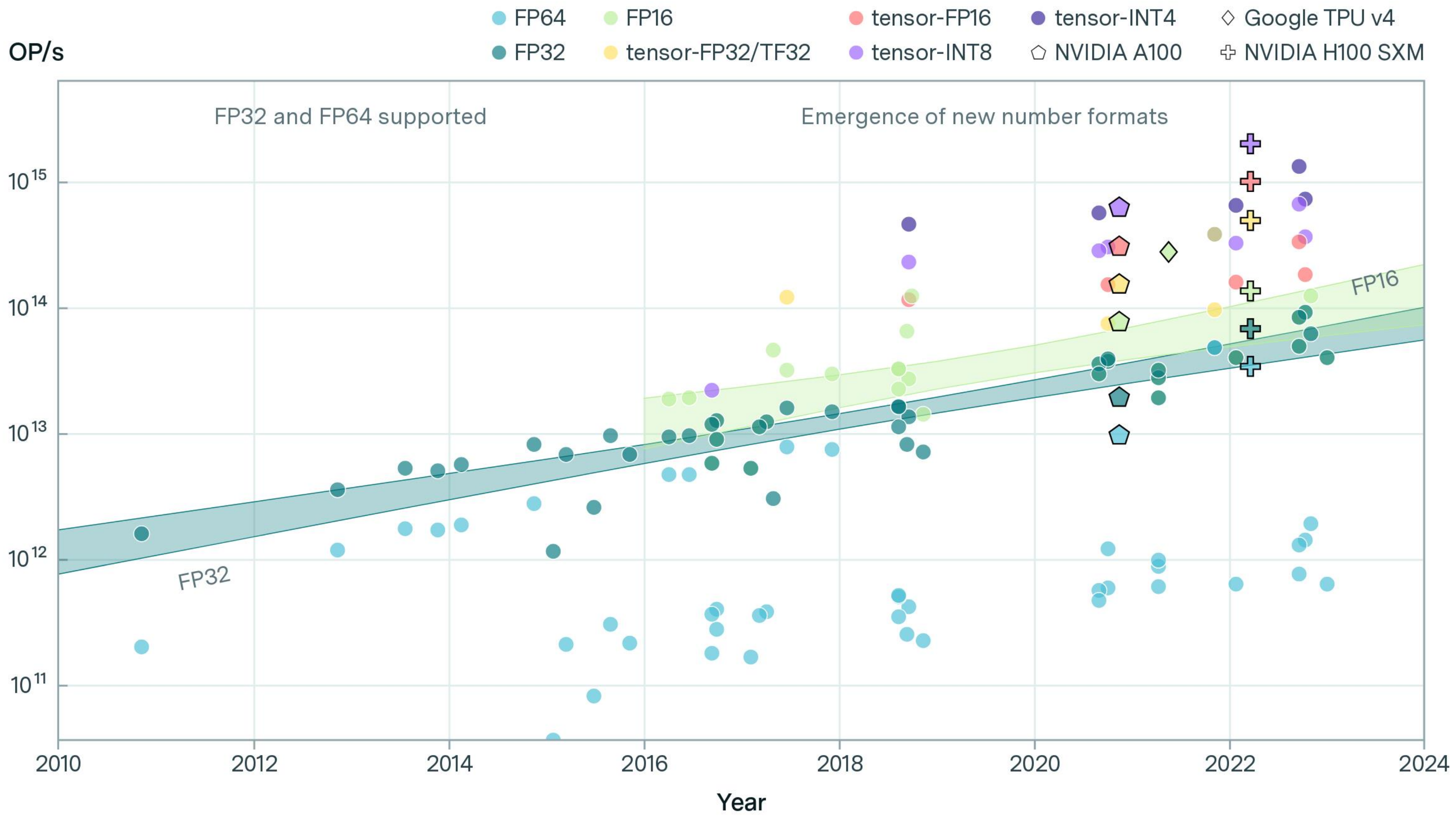
Beyond Moore's Law?

Number of accelerators?
Parallelization and specialization?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

Trends in Machine Learning Hardware?



[Image Credit: Epoch AI]

The Verticalization of Silicon

- Shift: from buying chips → to building differentiated chips
  - Apple: ditched x86, built M-series SoCs for Mac/iPad
  - Google, Amazon, Microsoft, Alibaba: datacenter CPUs & AI accelerators
  - Tesla: custom chips for autonomous driving & EVs
  - OpenAI: in-house AI accelerator to reduce Nvidia dependence?

- End-system value/profit justifies cost of chip design
  - can be >>$100M engineering cost to develop a new advanced chip!

Big Tech’s Homegrown Chips

Company	Chip	Launched
Amazon	Graviton	2018
Google	Axion	2024
Microsoft	Cobalt	2023
Amazon	Trainium	2022
Amazon	Inferentia	2019
Google	TPU	2015/2017
Microsoft	MAIA	2023
Meta	MTIA	2023

Agenda

- What is Computer Architecture?
- Course Admin
- ISA Review

Course Administration

- Instructor: Prof. Ting-Jung Chang (tingchang@cs.nycu.edu.tw)
    - Office Hours: T 15:30-16:30@EC707 by appointment
- Lectures: Tuesday 13:20 – 15:10@EDB27
- Text: Computer Architecture: A Quantitative Approach **6th Edition**
- Prerequisite: Computer Organization, or equivalent
- Course Webpage: e3

TAs

- 趙家逸 (chaiyi.cs14@nycu.edu.tw)
- 謝宥逸 (yuyi92025.cs14@nycu.edu.tw)
- 吳柏頡 (pchw.cs14@nycu.edu.tw)
- 邵品翔 (amking0916.cs14@nycu.edu.tw)

- Location: EC118
- Please email in advance
- Check e3 for final updates (office hours and other info)

Course Structure

- Midterm (20%) (~10/21)
- Final (30%) (~12/16)
- Labs (50%)
  - 5 design labs (Verilog)
- Ungraded Problem Sets (0%)
  - Intended to help you learn the material
  - Feel free to discuss with other students and instructors
  - Useful for exam preparation

Lab Academic Integrity

- Do your own work
- Be careful on AI usage
  - Explore ideas or debug with AI
  - But you must check, adapt, and make it your own
  - Work that lacks originality — AI or otherwise — will be penalized
- No public sharing of labs (GitHub, Google Drive, etc.)
- Integrity matters: violations can result in penalties, including failing the course

Course Administration

- Your goal doday: decide if you're coming back
- Notices (all materials available on e3)
    - Labo released
    - Verilog tips as supplementary material

Computer Organization

- RISC-I (1982) fabbed in 5 µm NMOS, with a die area of 77 mm^2, ran at 1 MHz. This chip is probably the first VLSI RISC
- Basic pipelined processor
- ~50,000 transistors

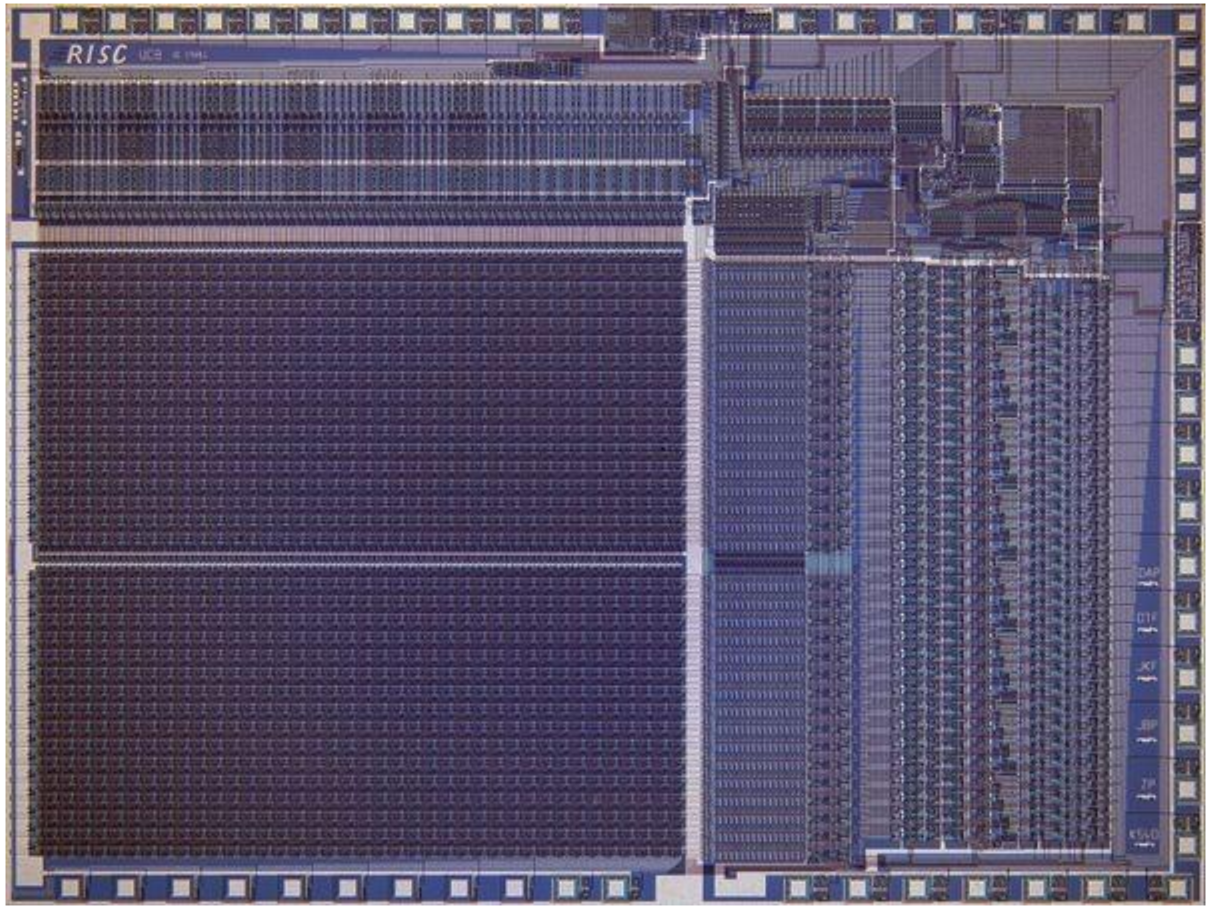


Photo of Berkeley RISC I, © University of California (Berkeley)

Computer Architecture

- Instruction Level Parallelism
  - Superscalar
  - Very Long Instruction Word (VLIW)
- Long Pipelines (Pipeline Parallelism)
- Advanced Memory and Caches
- Data Level Parallelism
  - Vector
  - GPU
- Thread Level Parallelism
  - Multithreading
  - Multiprocessor
  - Multicore
  - Manycore
- More…

~731,000,000 transistors

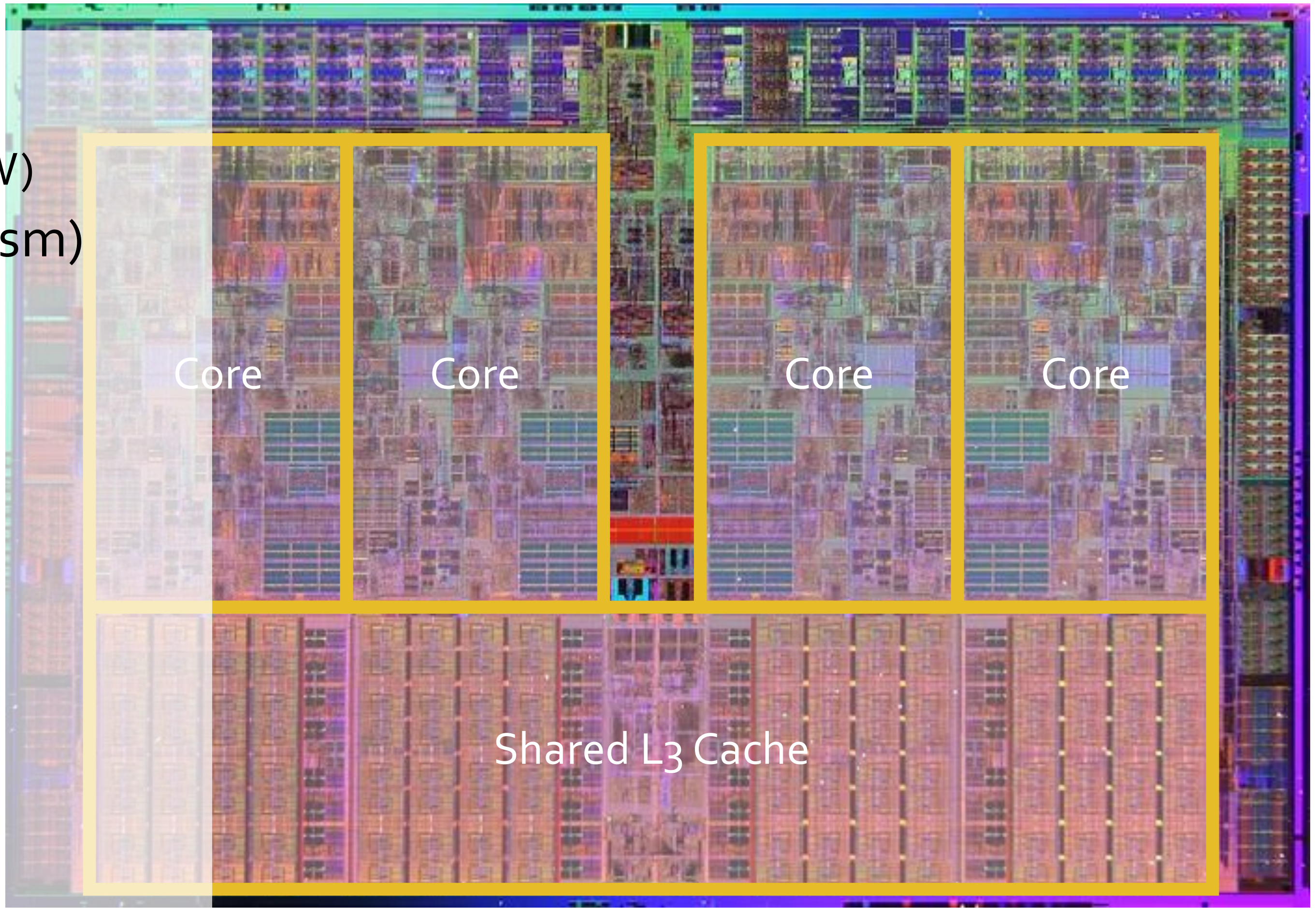


Photo of Intel Nehalem Processor, Original Core i7, © Intel

Agenda

- What is Computer Architecture?
- Course Admin
- ISA Review

Architecture vs. Microarchitecture

- **A**rchitecture/Instruction Set Architecture:
    - Programmer visible state (memory and register)
    - Operations (Instructions and how they work)
    - Execution Semantics (Interrupts)
    - Input/Output
    - Data Types/Sizes
- Microarchitecture/Organization:
    - Tradeoffs on how to implement ISA for some metric (speed, energy, cost)
    - Examples: pipeline depth, number of pipelines, cache size, silicon area, peak power, execution ordering, bus widths, ALU widths

Software Developments

up to 1955	Libraries of numerical routines • Floating point operations • Transcendental functions • Matrix manipulation, equation solvers
1955-60	High level languages – Fortran 1956 Operating Systems • Assemblers, loaders, linkers, compilers • Accounting programs to keep track of usage and charges

Machines required **experienced** operators

- Most users could not be expected to understand these programs, much less write them
- Machines had to be sold with a lot of resident software

Compatibility Problem at IBM

- By early 1960's, IBM had 4 incompatible lines of computers!
  - 701 $\rightarrow$ 7094
  - 650 $\rightarrow$ 7074
  - 702 $\rightarrow$ 7080
  - 1401 $\rightarrow$ 7010
- Each system had its own
  - Instruction set
  - I/O system and secondary storage (magnetic tapes, drums and disks)
  - Assemblers, compilers, libraries
  - Market niche business, scientific, real time, ...

IBM 360!

IBM 360: A General-Purpose Register (GPR) Machine

- Processor State
  - 16 general-purpose 32-bit registers
    - May be used as index and base register
    - Register 0 has some special properties
  - 4 floating point 64-bit registers
  - A program status word (PSW)
    - PC, condition codes, control flags
- A 32-bit machine with 24-bit addresses
  - But no instruction contains a 24-bit address!
- Data formats
  - 8-bit bytes, 16-bit half words, 32-bit words, 64-bit double words

The IBM 360 is why bytes are 8-bits long today!

IBM 360: Initial Implementations

	Model 30	Model 70
Storage	8K – 64 KB	256K – 512 KB
Datapath	8-bit	64-bit
Circuit Delay	30 nsec/level	5 nsec/level
Local Store	Main Store	Transistor Registers
Control Store	Read only 1µsec	Conventional circuits

- IBM 360 instruction set architecture (ISA) completely hid the underlying technological differences between various models.
- Milestone: The first true ISA designed as portable hardware-software interface!

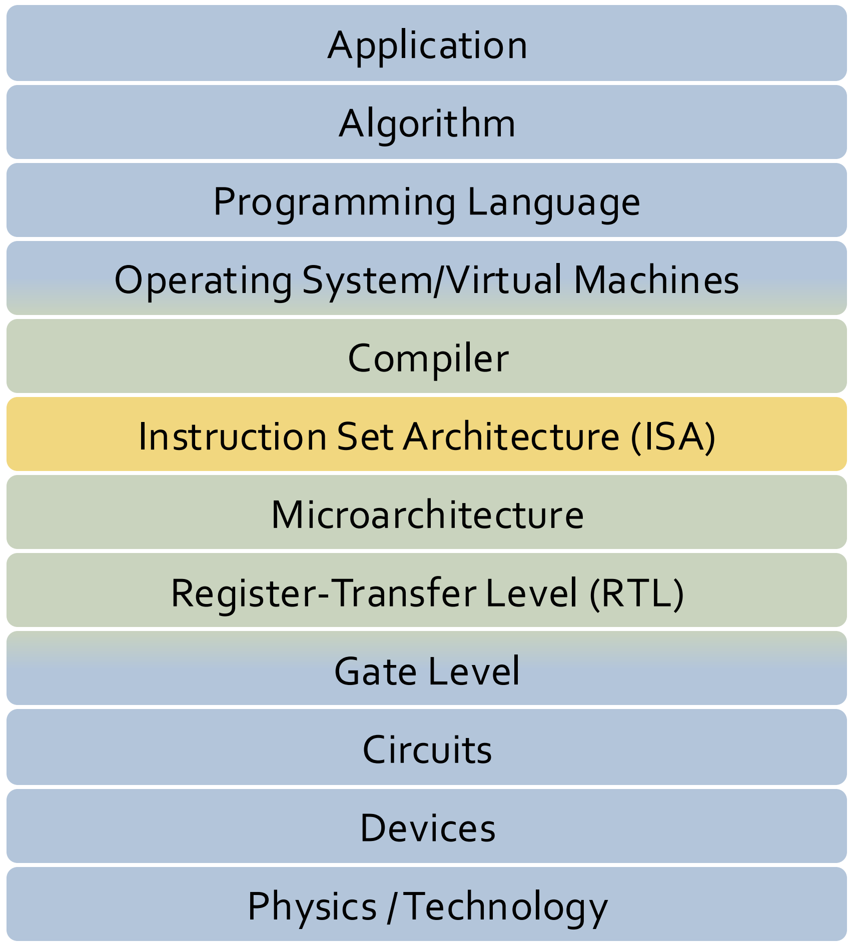
With minor modifications it still survives today!

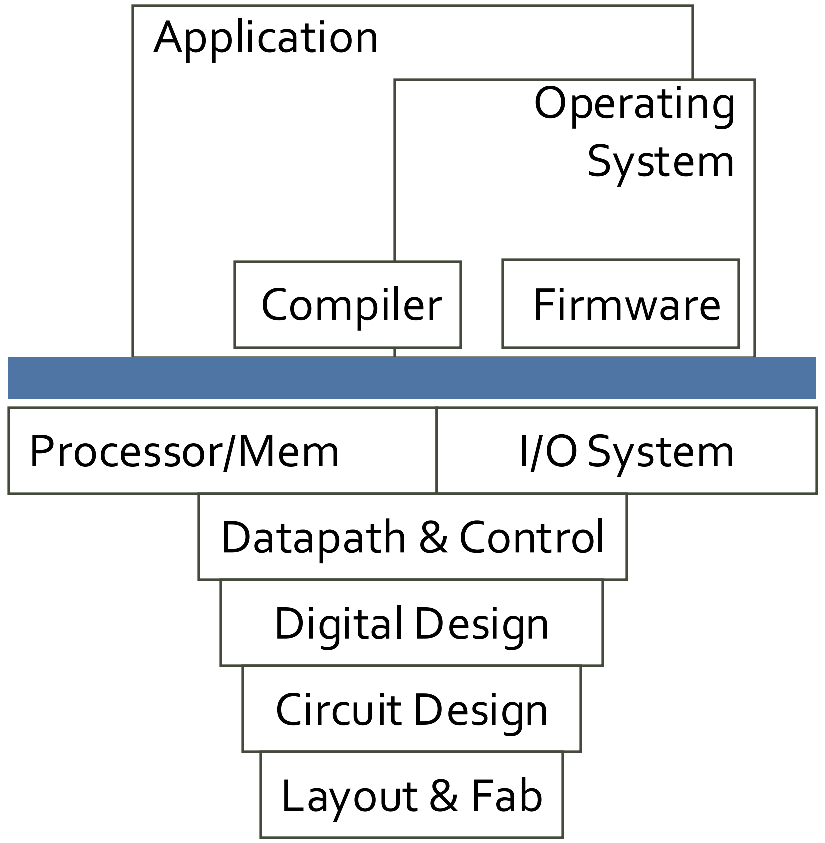
IBM 360: Over 50 years later… The zSeries z16 Microprocessor

Image Credit: IBM
© International Business Machines Corporation.

- 5.2GHz in 7nm Samsung technology
- 22.5 Billion transistors in 530 mm^2
- 64-bit virtual addressing – original S/360 was 24-bit, and S/370 was 31-bit extension
- 8 cores + L2s per chip
- 128 KB L1, 32 MB L2, 256MB L3, 2048 MB L4
- Integrated on-chip AI accelerator

Instruction Set Architecture (ISA)





Instruction Set Architecture (ISA)

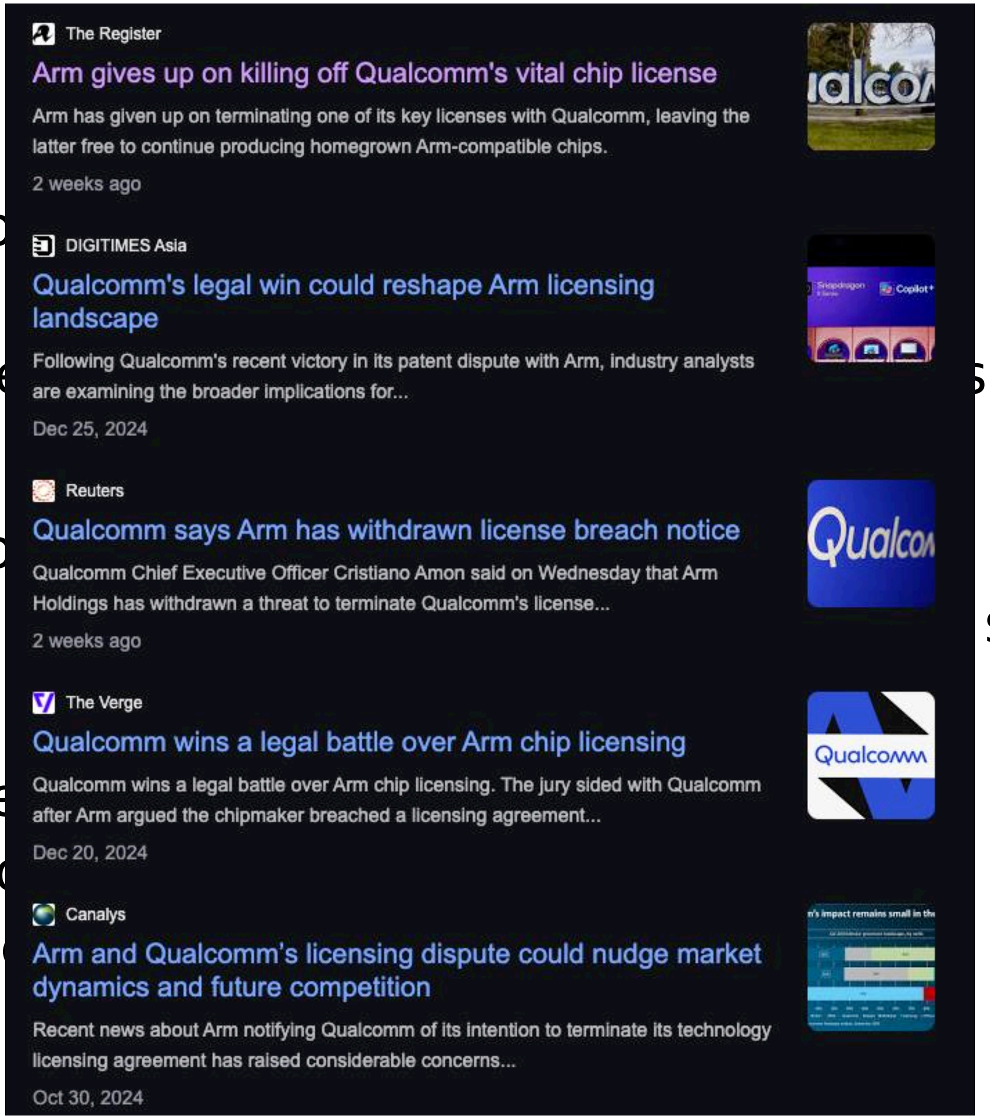
- The ISA is functional abstraction of the processor (a “mental model”)
  - What operations can be performed
  - How to name storage locations
  - The format (bit pattern) of the instructions
- ISA typically does NOT define
  - Timing of the operations
  - Power used by operations
  - How operations/storage are implemented
- Many implementations possible for a given ISA

Different Instruction Set Architecture

- ARM
  - Family of ISAs developed by ARM
  - Widely used in mobile and low-power devices
  - Now expanding into desktop, datacenters and cloud servers
- x86
  - Family of ISAs developed by Intel (and AMD)
  - Used in general-purpose computing systems (desktops and servers
- RISC-V
  - Open standard ISAs developed at UC-Berkeley
  - Mostly used in embedded systems
  - Early adoption in AI accelerators and research chips

Different Instruction Set Architecture

- ARM
  - Family of ISAs develop
  - Widely used in mobile
  - Now expanding into de s
- x86
  - Family of ISAs develop
  - Used in general-purpo servers
- RISC-V
  - Open standard ISAs de
  - Mostly used in embedd
  - Early adoption in AI ac



The Register
Arm gives up on killing off Qualcomm's vital chip license
Arm has given up on terminating one of its key licenses with Qualcomm, leaving the latter free to continue producing homegrown Arm-compatible chips.
2 weeks ago

DIGITIMES Asia
Qualcomm's legal win could reshape Arm licensing landscape
Following Qualcomm's recent victory in its patent dispute with Arm, industry analysts are examining the broader implications for...
Dec 25, 2024

Reuters
Qualcomm says Arm has withdrawn license breach notice
Qualcomm Chief Executive Officer Cristiano Amon said on Wednesday that Arm Holdings has withdrawn a threat to terminate Qualcomm's license...
2 weeks ago

The Verge
Qualcomm wins a legal battle over Arm chip licensing
Qualcomm wins a legal battle over Arm chip licensing. The jury sided with Qualcomm after Arm argued the chipmaker breached a licensing agreement...
Dec 20, 2024

Canalys
Arm and Qualcomm's licensing dispute could nudge market dynamics and future competition
Recent news about Arm notifying Qualcomm of its intention to terminate its technology licensing agreement has raised considerable concerns...
Oct 30, 2024

Same Architecture/Different Microarchitecture

Intel Xeon

- x86 instruction set
- 32+ cores
- 200W+
- Decode 6 instructions/cycle/core
- Large shared caches (tens of MB L3)
- Out-of-order
- ~ 2.0-3.5 GHz

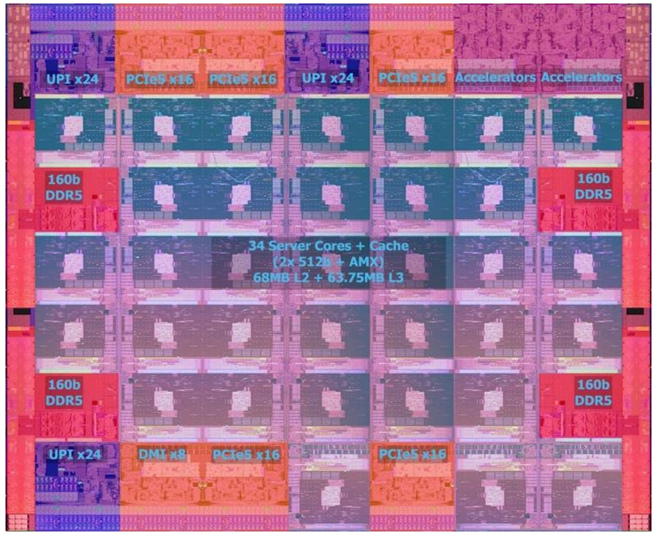


Image Credit: Intel

Intel Atom

- x86 instruction set
- Few cores (1–8)
- 2W
- Decode 2 instructions/cycle/core
- tens of KB L1, <2 MB L2
- In-order
- 1.6GHz

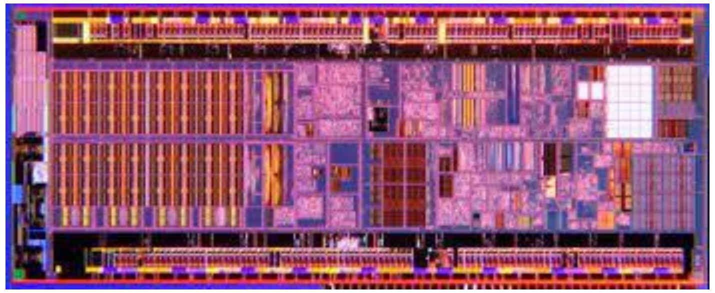


Image Credit: Intel

Same Architecture/Different Microarchitecture

Intel Xeon

- x86 instruction set
- 32+ cores
- 200W+
- Decode 6 instructions/cycle/core
- Large shared caches (tens of MB L3)
- Out-of-order
- ~ 2.0-3.5 GHz

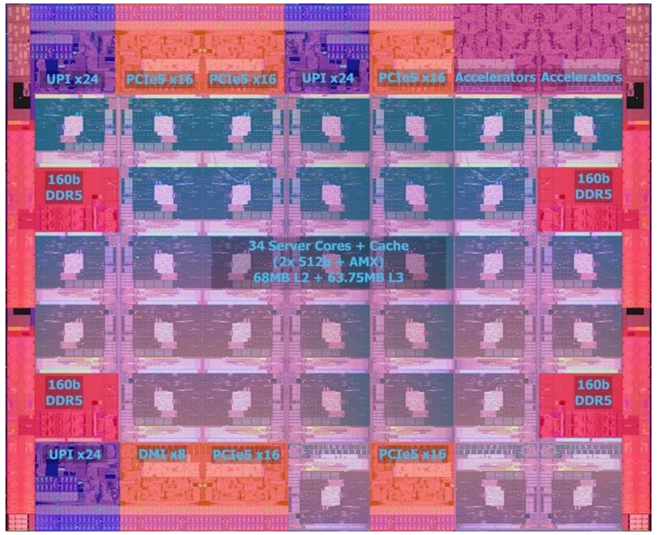


Image Credit: Intel

Apple M2 Ultra

- ARM instruction set
- Up to 24 cores
- ~90–100 W
- Decode 8+ instructions/cycle/core
- Large shared caches across clusters
- Out-of-order
- ~3.5 GHz peak frequency

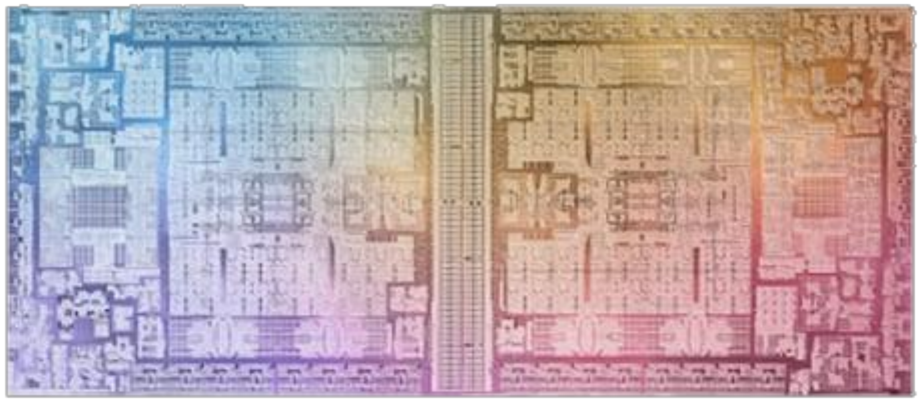


Image Credit: Apple

The Standard Structure of An Instruction

- An instruction typically has an operator (op or opcode), one or two source operands (src), and one destination operand (dest).

R1	=	R2	+	R3

goto	target

R1	=	*addr

if	(R1	>	*addr)

*addr	=	R1

if	(R1	== 0)	{goto	target}

What would these look like in RISC-V assembly?

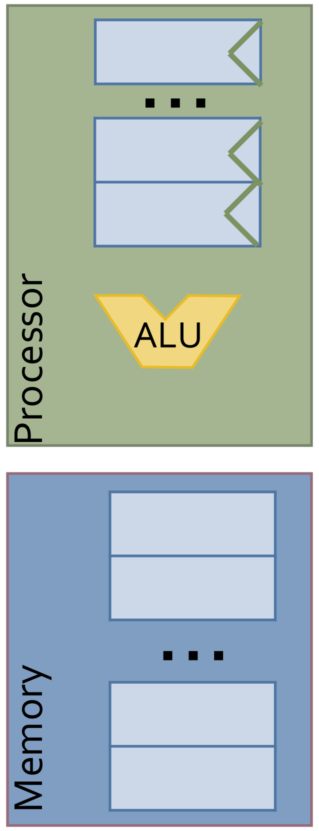
Key ISA Decisions

- Operands
  - How many?
  - Location?
  - Addressing mode?
  - Types?
- Operations
  - What kind?
  - How many?
- Instruction format
  - Length(s) of bit pattern
  - Which bits mean what

ISA Classification

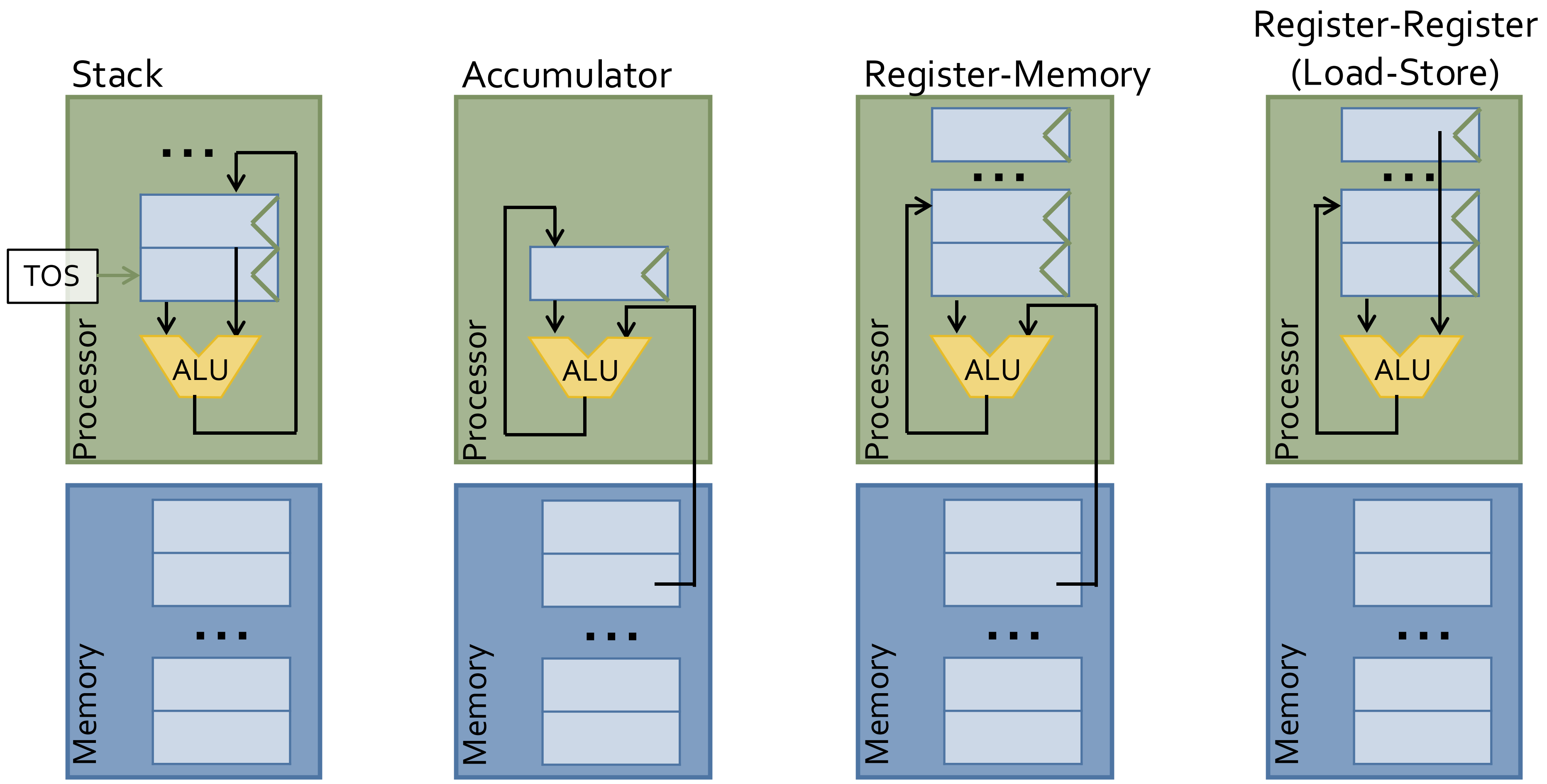
Operands

- Operands may be from
  - Stack
  - Accumulator
  - Register
  - Register and Memory



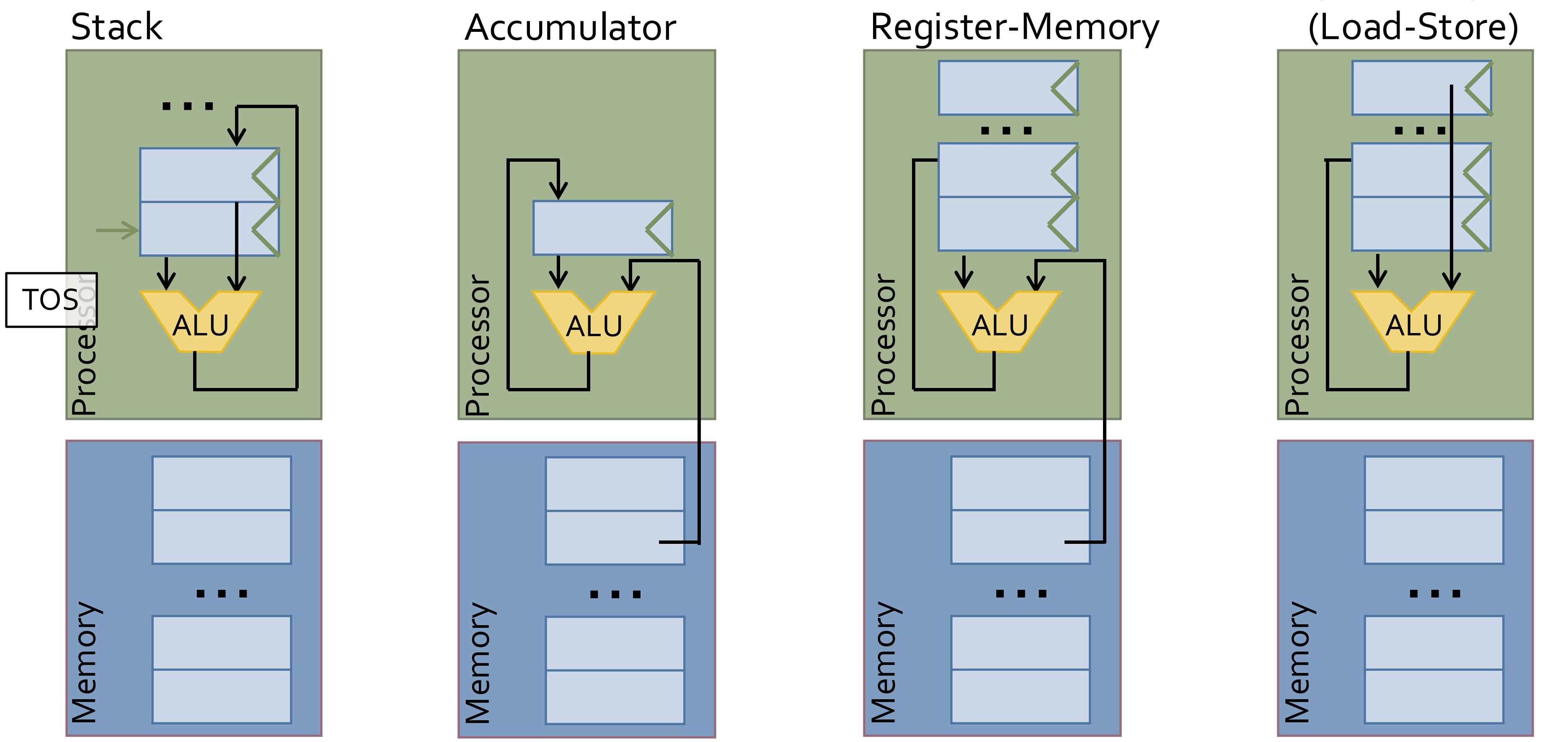
Where do operands come from and where do results go?

Machine Models of ISA



Number Explicitly Named Operands:	0	1	2 or 3	2 or 3

Summary: Machine Model



C = A + B

Stack	Accumulator	Register-Memory	Register-Register (Load-Store)
Push A	Load A	Load R1, [A]	Load R1, [A]
Push B	Add [B]	Add R3, R1, [B]	Load R2, [B]
Add	Store C	Store R3, [C]	Add R3, R1, R2
Pop C			Store R3, [C]

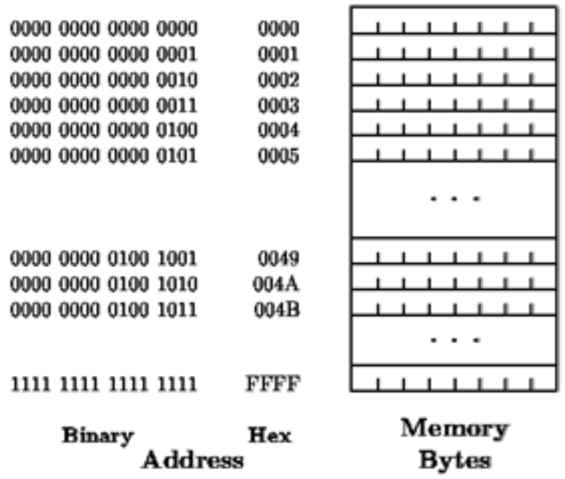
ISA Classification

Operand Addressing Mode

- Addressing mode specifies how operands are located from memory and/or registers.
- Common addressing modes
  - Register
  - Immediate
  - Register indirect
  - Displacement
  - Indexed
  - Direct
  - Memory indirect
  - Auto-increment and auto-decrement
  - Scaled

Memory Addressing

- Objects have byte addresses
  - the number of bytes counted from the beginning of memory
- Object Length:
  - bytes (8 bits), half words (16 bits)
  - words (32 bits), and double words (64 bits).
  - The type is implied in opcode, e.g.,
    - lb – load byte
    - lw – load word, etc.
- Addressing mode specifies how operands are located from memory and/or registers



Memory Addressing

- Byte Ordering
  - Little Endian: Stores the least significant byte (LSB) at the lowest memory address
    - Ex: 0x12345678

Address	0x100	0x101	0x102	0x103
	0x78	0x56	0x34	0x12

  - Big Endian: Stores the most significant byte (MSB) at the lowest memory address
    - Ex: 0x12345678

Address	0x100	0x101	0x102	0x103
	0x12	0x34	0x56	0x78

  - Problem occurs when exchanging data among machines with different orderings

Summary: Addressing Modes

Addressing Mode	Instruction	Function
Register	Add R4, R3, R2	Regs[R4] <- Regs[R3] + Regs[R2] **
Immediate	Add R4, R3, #5	Regs[R4] <- Regs[R3] + 5 **
Displacement	Add R4, R3, 100(R1)	Regs[R4] <- Regs[R3] + Mem[100 + Regs[R1]]
Register Indirect	Add R4, R3, (R1)	Regs[R4] <- Regs[R3] + Mem[Regs[R1]]
Absolute/Direct	Add R4, R3, (0x475)	Regs[R4] <- Regs[R3] + Mem[0x475]
Memory Indirect	Add R4, R3, @(R1)	Regs[R4] <- Regs[R3] + Mem[Mem[Regs[R1]]]
PC relative	Add R4, R3, 100(PC)	Regs[R4] <- Regs[R3] + Mem[100 + PC]
Scaled	Add R4, R3, 100(R1)[R5]	Regs[R4] <- Regs[R3] + Mem[100 + Regs[R1] + Regs[R5] * 4]

** May not actually access memory!

Type And Size of Operands

- Types
  - Binary Integer
  - Binary Coded Decimal (BCD)
  - Floating Point
    - IEEE 754
    - Cray Floating Point
  - Intel Extended Precision (80-bit)
  - Packed Vector Data
  - Addresses

- Width
  - Binary Integer (8-bit, 16-bit, 32-bit, 64-bit)
  - Floating Point (32-bit, 40-bit, 64-bit, 80-bit)
  - Addresses (16-bit, 24-bit, 32-bit, 48-bit, 64-bit)

Key ISA Decisions

- Operands
  - How many?
  - Location?
  - Addressing mode?
  - Types?
- Operations
  - What kind?
  - How many?
- Instruction format
  - Length(s) of bit pattern
  - Which bits mean what

Classes of Instructions

- Arithmetic & Logical
  - Integer arithmetic: add, sub, mul, div, shift
  - Logical operation: and, or, xor, not
- Data Transfer
  - copy, load, store (w/ memory addressing)
- Control Flow
  - branch, jump, call, return, trap
- Floating Point
- System: OS and memory management
- Graphics: pixel and vertex, compression/decompression operations
- String
  - move, compare, search

Key ISA Decisions

- Operands
  - How many?
  - Location?
  - Addressing mode?
  - Types?
- Operations
  - What kind?
  - How many?
- Instruction format
  - Length(s) of bit pattern
  - Which bits mean what

Encoding an Instruction Set

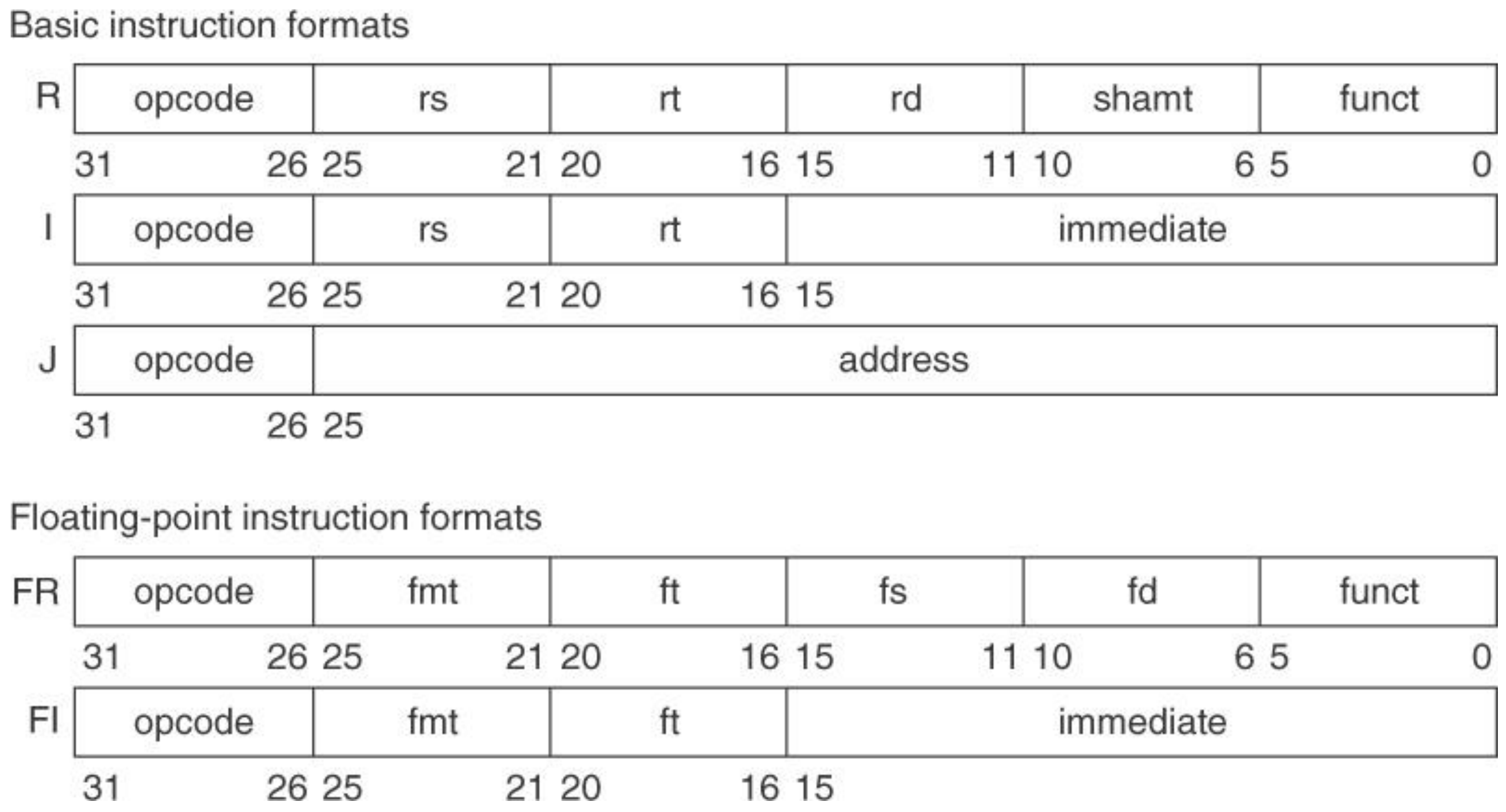
- Opcode: specifying the operation
- # of operands:
  - addressing mode
  - address specifier tells what addressing mode is used
  - Load-store computer
    - Only one memory operand
    - Only one or two addressing modes
- The architecture must balance several competing forces when encoding the instruction set:
  - # of registers && addressing modes
  - Size of registers && addressing mode fields
  - Average instruction size && average program size.
  - Easy to handle in pipeline implementation

Example: x86 Instruction Encoding

Instruction Prefixes	Opcode	ModR/M	Scale, Index, Base	Displacement	Immediate
Up to four Prefixes (1 byte each)	1,2, or 3 bytes	1 byte (if needed)	1 byte (if needed)	0,1,2, or 4 bytes	0,1,2, or 4 bytes

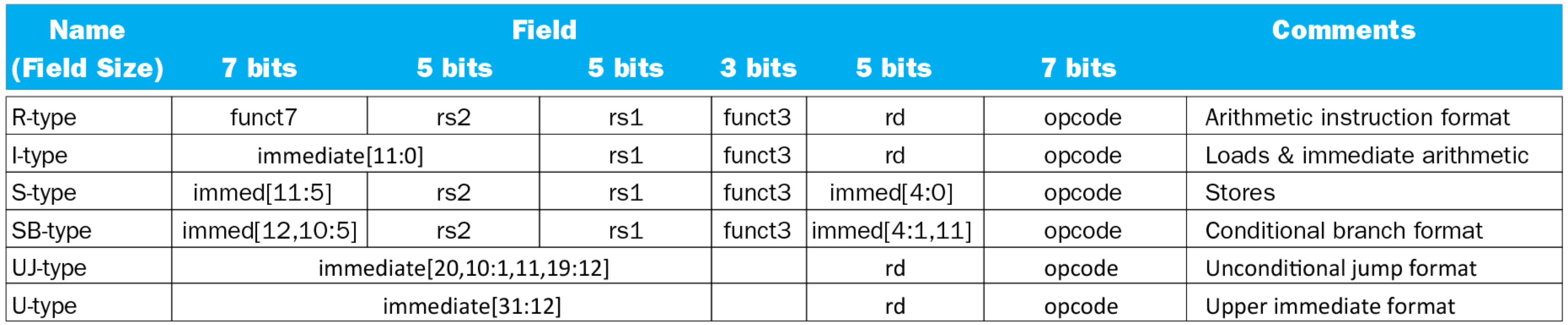
Possible instructions 1 to 15 bytes long

MIPS64 Instruction Encoding



RISC-V Instruction Encoding

- Restrictions
  - 4 bytes per instruction
  - Different instructions have different parameters (registers, immediates, …)
  - Various fields should be encoded to consistent locations
    - Simpler decoding circuitry
- RISC-V uses 6 "types" of instruction encoding



Name	Field						Comments
(Field Size)	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

ISA Classification

Instruction Length

- **Fixed Width:** Every Instruction has same width
  - + Easy to decode (**RISC** Architectures: MIPS, PowerPC, SPARC, ARM, RISC-V…)
  - − Wasted bits in instructions (Why is this bad?)
  - − Harder-to-extend ISA (how to add new instructions?)
  - Ex: MIPS, RISC-V, ARM, …
- **Variable Length:** Instructions can vary in width
  - + Takes less space in memory and caches (**CISC** Architectures: x86, VAX…)
  - − More logic to decode a single instruction
  - − Harder to decode multiple instructions concurrently
  - Ex: x86, instructions 1-byte up to 18-bytes

ISA Classification

Instruction Length

- **Hybrid**
  - Support 16-bit and 32-bit instructions in RISC. Narrow instructions support fewer operations, smaller address and immediate fields, fewer registers, and two-address format rather than the classic three-address format
  - Claim a code size reduction of up to 40%
  - Ex: ARM Thumb, MIPS16e, and RISC-V
- **Compressed:**
  - PowerPC and some VLIWs (Store instructions compressed, decompress into Instruction Cache
- **(Very) Long Instruction Word (VLIW):**
  - Multiple instructions in a fixed width bundle
  - Ex: Multiflow, HP/ST Lx, TI C6000

Instruction Length Tradeoffs

- Instructions are eventually encoded with 0s and 1s.
  - Each instruction is encoded into several bytes of binary numbers.
- Tradeoffs
  - Code size (memory space, bandwidth, latency) vs. hardware complexity
  - ISA extensibility and expressiveness
  - Performance? Smaller code vs. imperfect decode

Real World Instruction Sets

RISC? CISC?

Arch	Type	# Oper	# Mem	Data Size	# Regs	Addr Size	Use
Alpha	Reg-Reg	3	0	64-bit	32	64-bit	Workstation
ARM	Reg-Reg	3	0	32/64-bit	16	32/64-bit	Cell Phones, Embedded
MIPS	Reg-Reg	3	0	32/64-bit	32	32/64-bit	Workstation, Embedded
SPARC	Reg-Reg	3	0	32/64-bit	24-32	32/64-bit	Workstation
TI C6000	Reg-Reg	3	0	32-bit	32	32-bit	DSP
IBM 360	Reg-Mem	2	1	8/16/32/64-bit	16	24/31/64-bit	Mainframe
x86	Reg-Mem	2	1	8/16/32/64-bit	4/8/16	16/32/64-bit	Personal Computers, HPC
RISC-V	Reg-Reg	3	0	32/64/128-bit	32	32/64-bit	Embedded

Why the Diversity in ISAs?

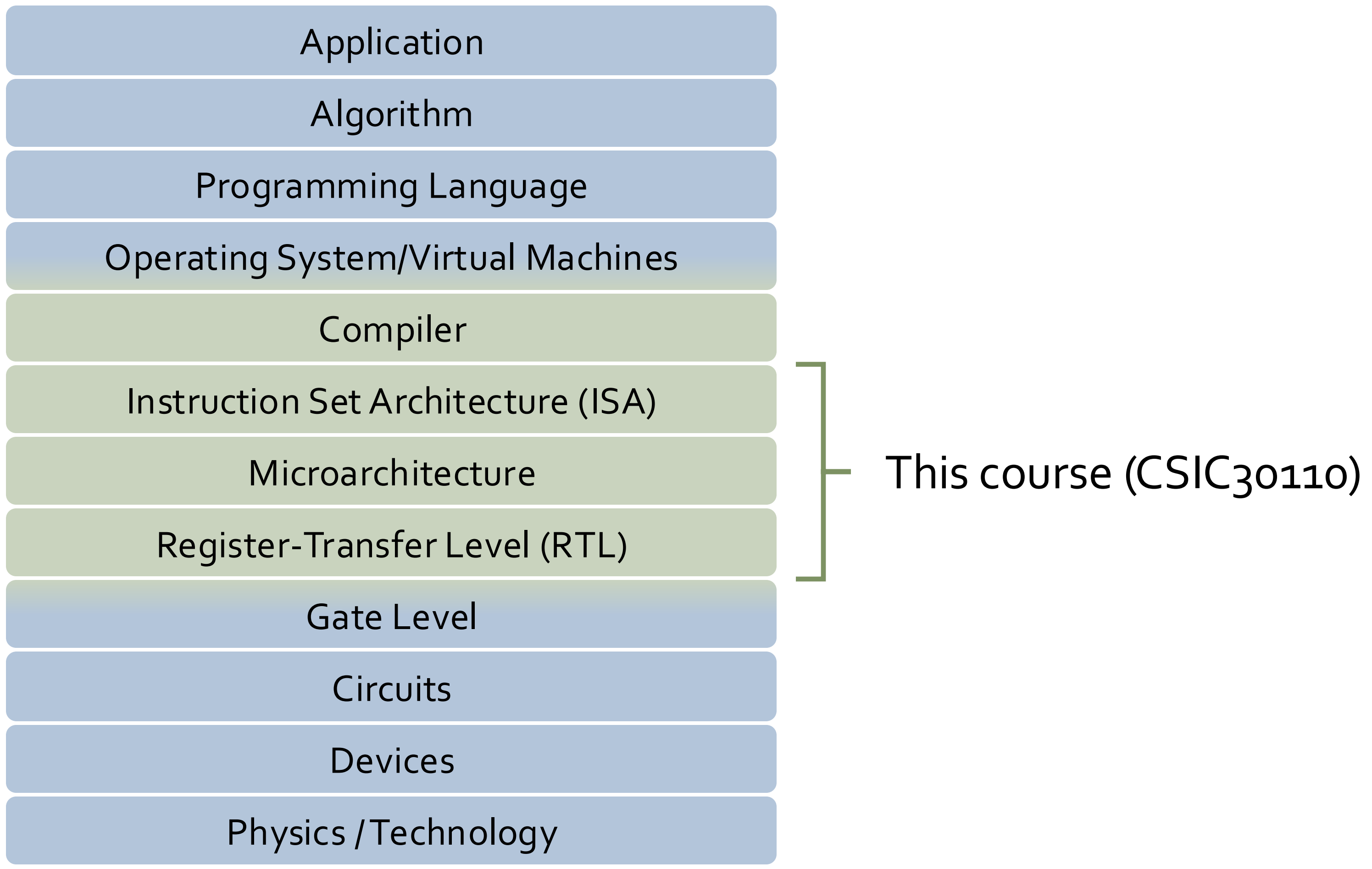
- Technology influenced ISA
  - Storage is expensive, tight encoding space
  - Reduced Instruction Set Computer
    - Remove instuctions until whole computer fits on die
  - Multicore/Manycore

- Application Influenced ISA
  - Instructions for applications
    - DSP instructions
  - Compiler technology has improved

Recap

- What is Computer Architecture?
- Course Admin
- ISA Review

Recap


Application
Algorithm
Programming Language
Operating System/Virtual Machines
Compiler
Instruction Set Architecture (ISA)
Microarchitecture
Register-Transfer Level (RTL)
Gate Level
Circuits
Devices
Physics / Technology
This course (CSIC30110)

Acknowledgements

- These slides contain material developed and copyright by:
  - Arvind (MIT)
  - Krste Asanovic (MIT/UCB)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)
  - Christopher Batten (Cornell)
  - David Wentzlaff (Princeton)
- MIT material derived from course 6.823
- UCB material derived from course CS252
- Cornell material derived from course ECE 4750
- Princeton material derived from course ECE 475