NOT FOR EXAM

Advanced Topics

2023 Fall COMP3230A

Great! We know a lot about OS!

More we DON’T know about OS…

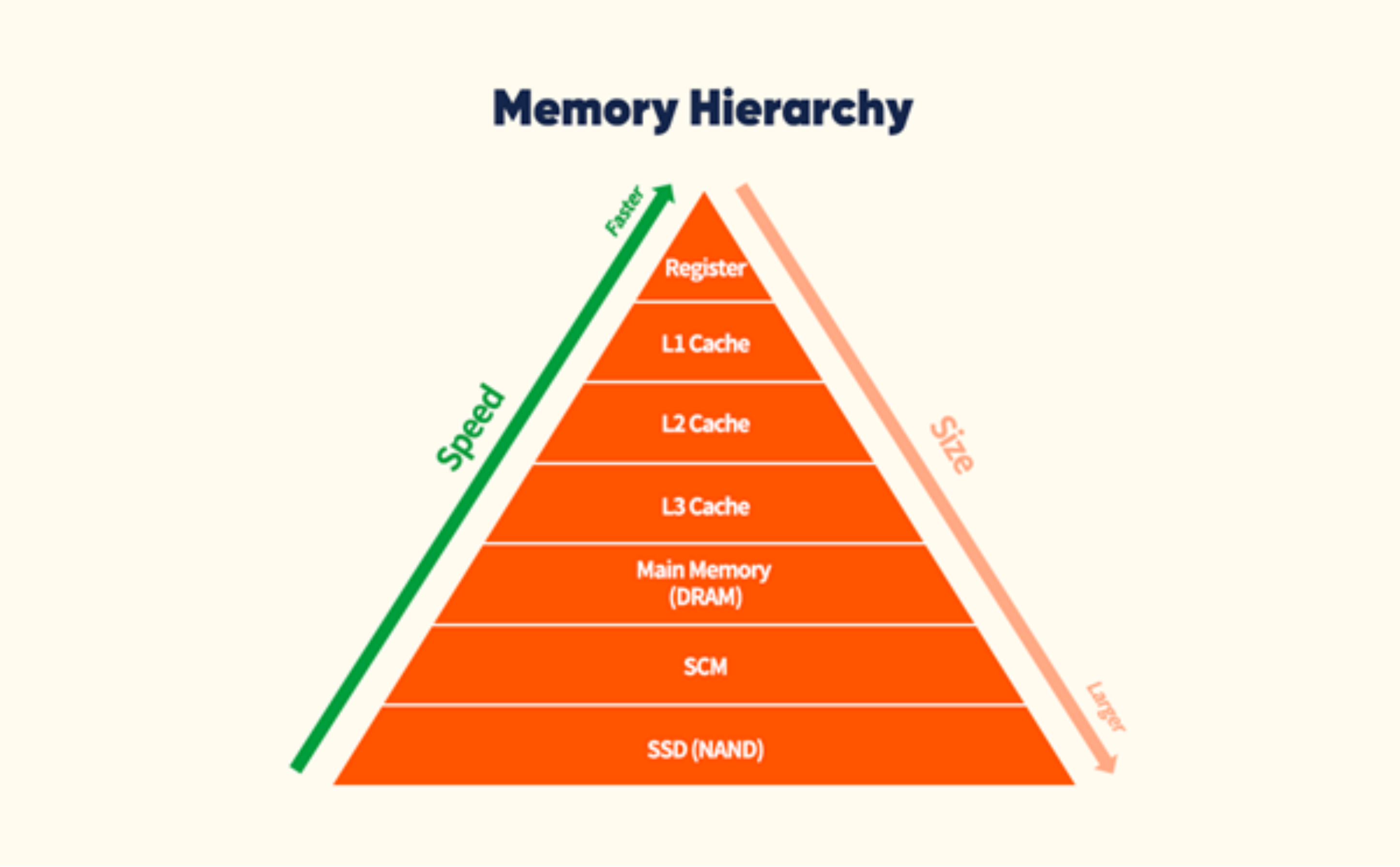
Education is not the filling of a pail, but the lighting of a fire.

——Yeats, Plutarch, or Anonymous

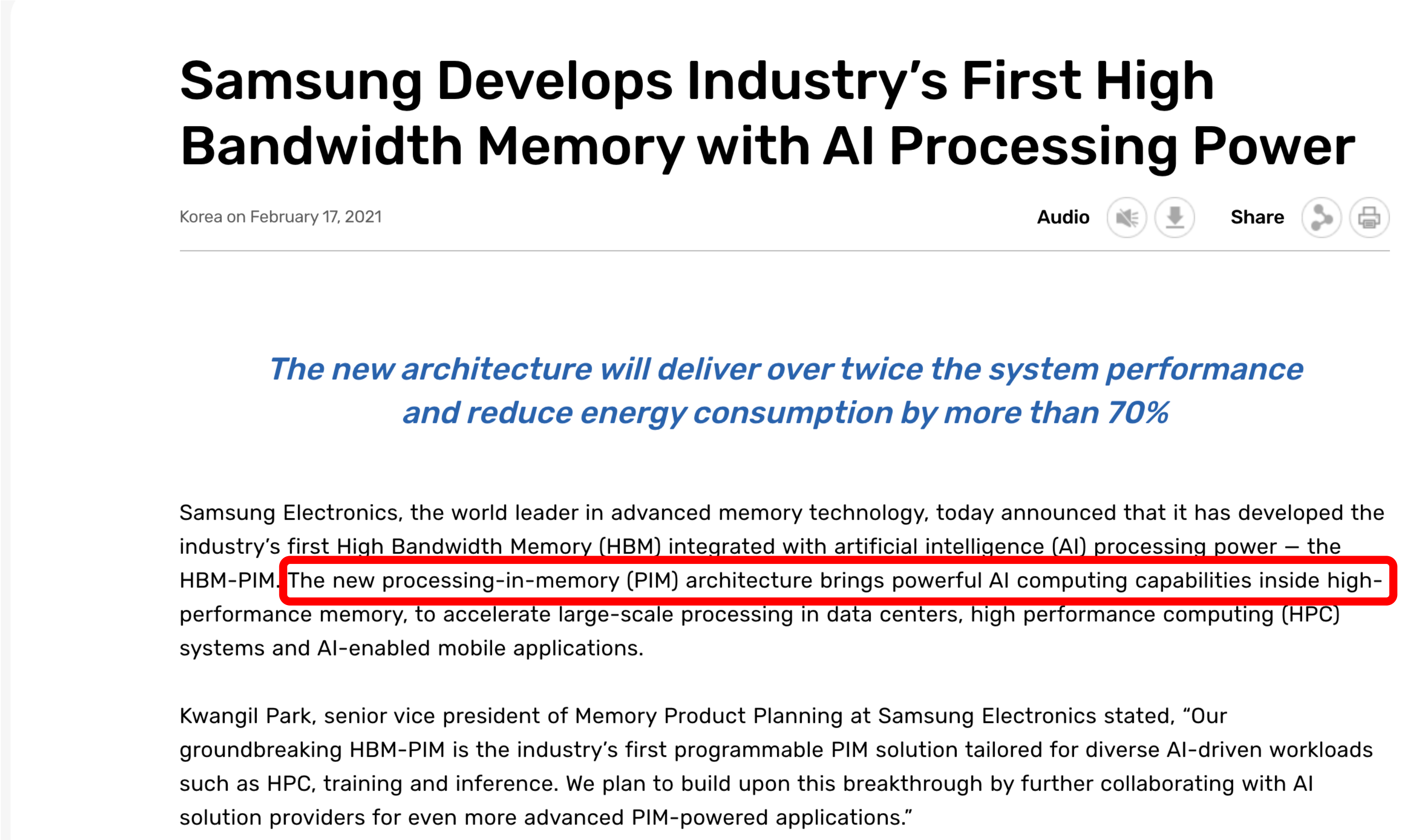
Contents

- Processing-in-Memory (PIM)
- OS Security
- Mobile and Embedded OSs

We’ve tried really hard to make memory access faster



Samsung Function-in-Memory DRAM (2021)



Samsung Develops Industry’s First High Bandwidth Memory with AI Processing Power

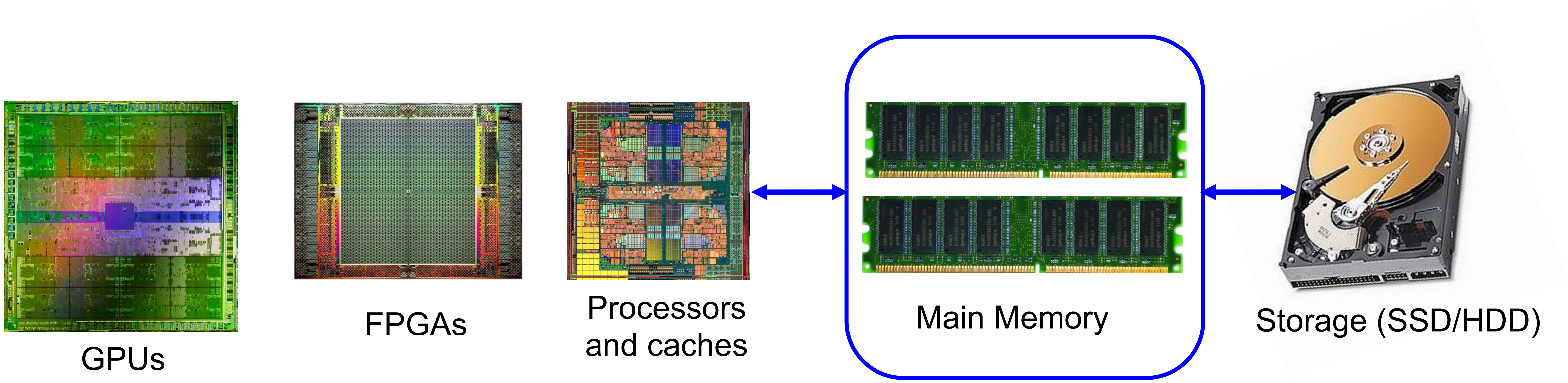
Korea on February 17, 2021

The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry’s first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power – the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, “Our groundbreaking HBM-PIM is the industry’s first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications.”

The Main Memory System



GPUs

FPGAs

Processors and caches

Main Memory

Storage (SSD/HDD)

- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor

- Main memory system must scale (in *size*, *technology*, *efficiency*, *cost*, and *management algorithms*) to maintain performance growth and technology scaling benefits

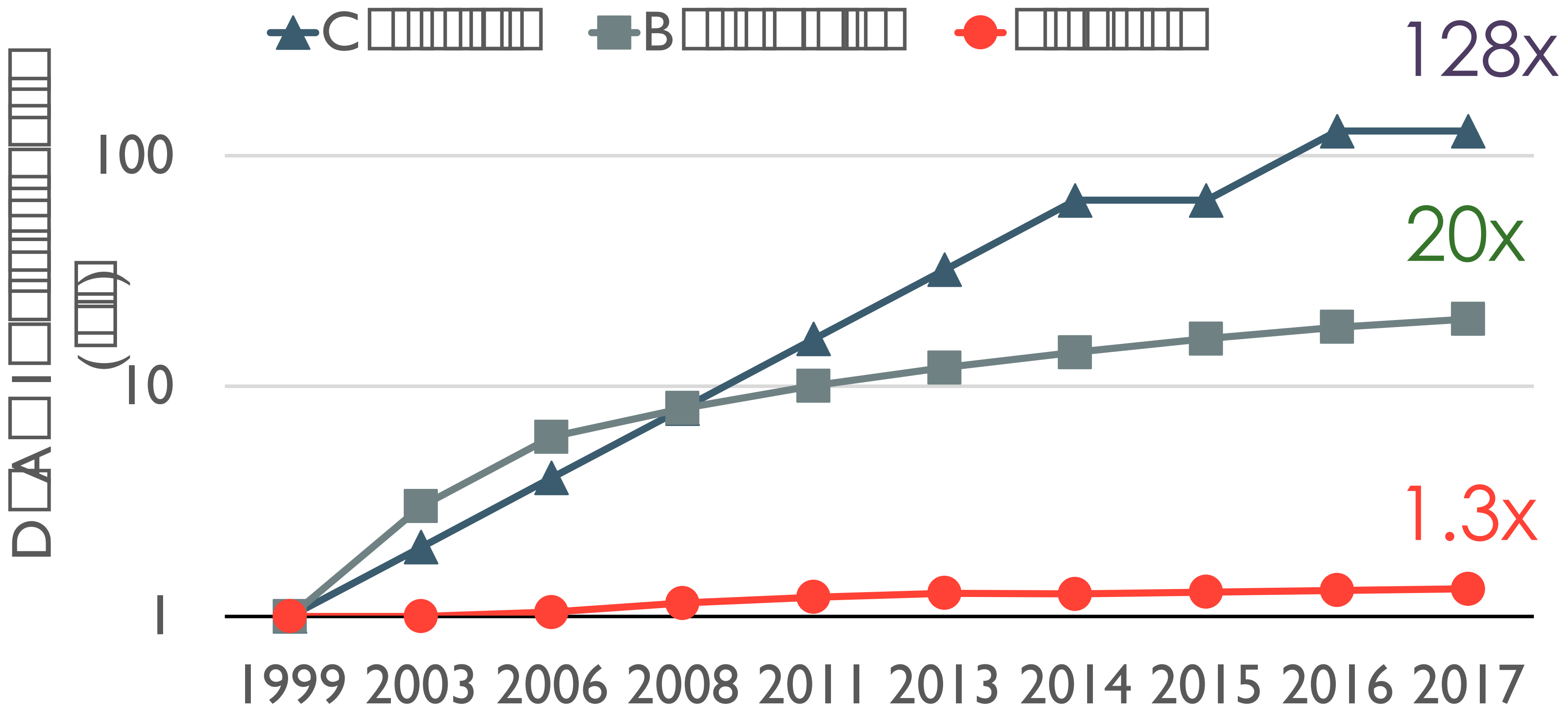
Three Key Systems Trends

1. Data access is a major bottleneck
   - Applications are increasingly data hungry

2. Energy consumption is a key limiter

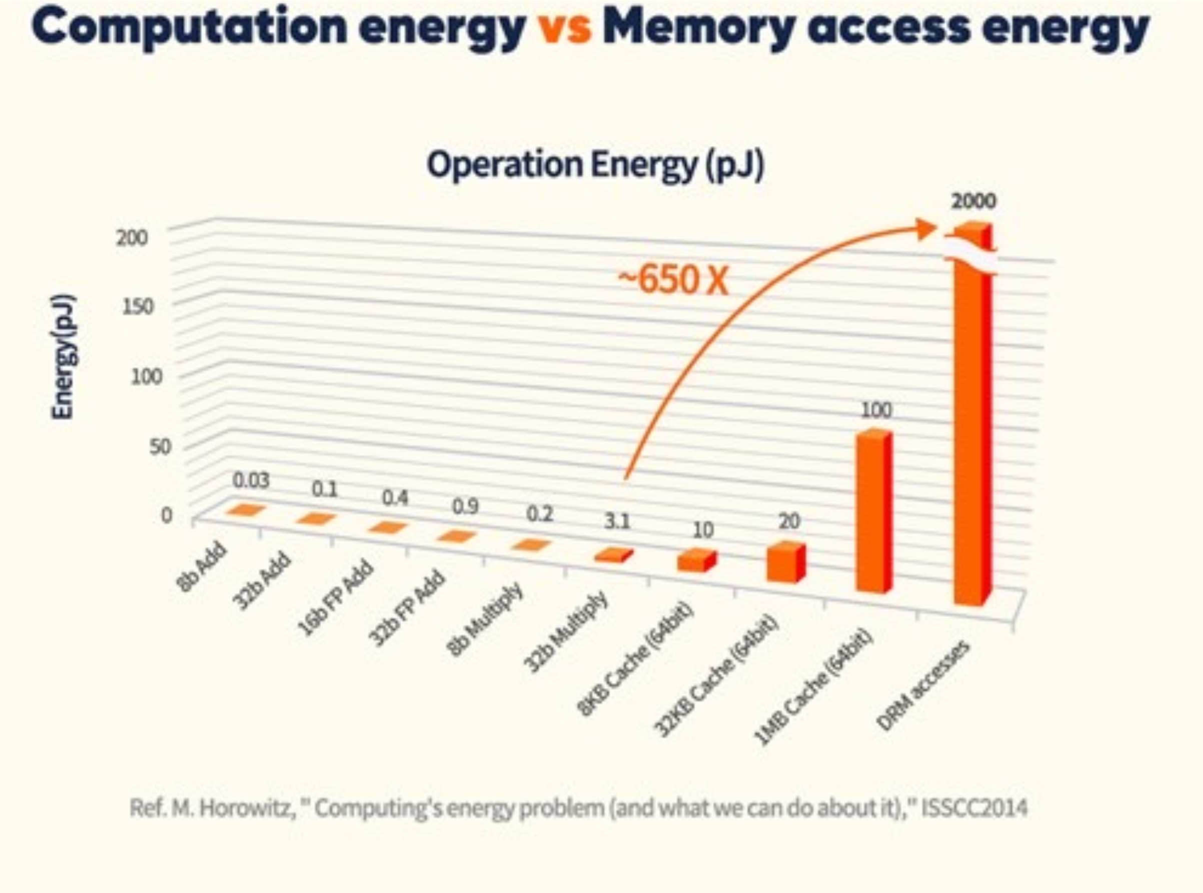
3. Data movement energy dominates compute
   - Especially true for off-chip to on-chip movement

Example: Capacity, Bandwidth & Latency



Memory latency remains almost constant

Data Movement vs. Computation Energy



Computation energy vs Memory access energy

Ref. M. Horowitz, " Computing's energy problem (and what we can do about it)," ISSCC2014

A memory access consumes ~1000X the energy of a complex addition

62.7% of the total system energy is spent on data movement

Energy consumption is a first-class concern in consumer devices

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks, ASPLOS 2018

Why Processing-in-Memory (PIM)?

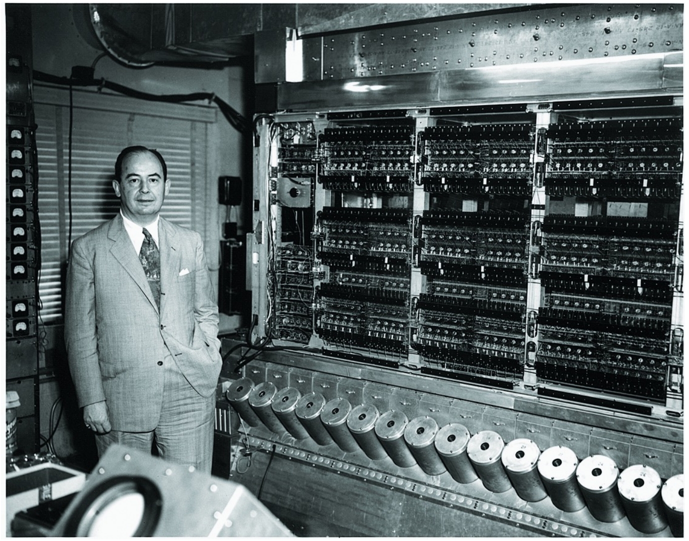
- More data-intensive computation
  - ML/DL: "Big/foundation models", BERT, GPT-3, DALL-E (2), billions of parameters
  - Graph processing, databases, video analytics, scientific computing, data center workloads……
  - Large datasets may exceed the main memory size

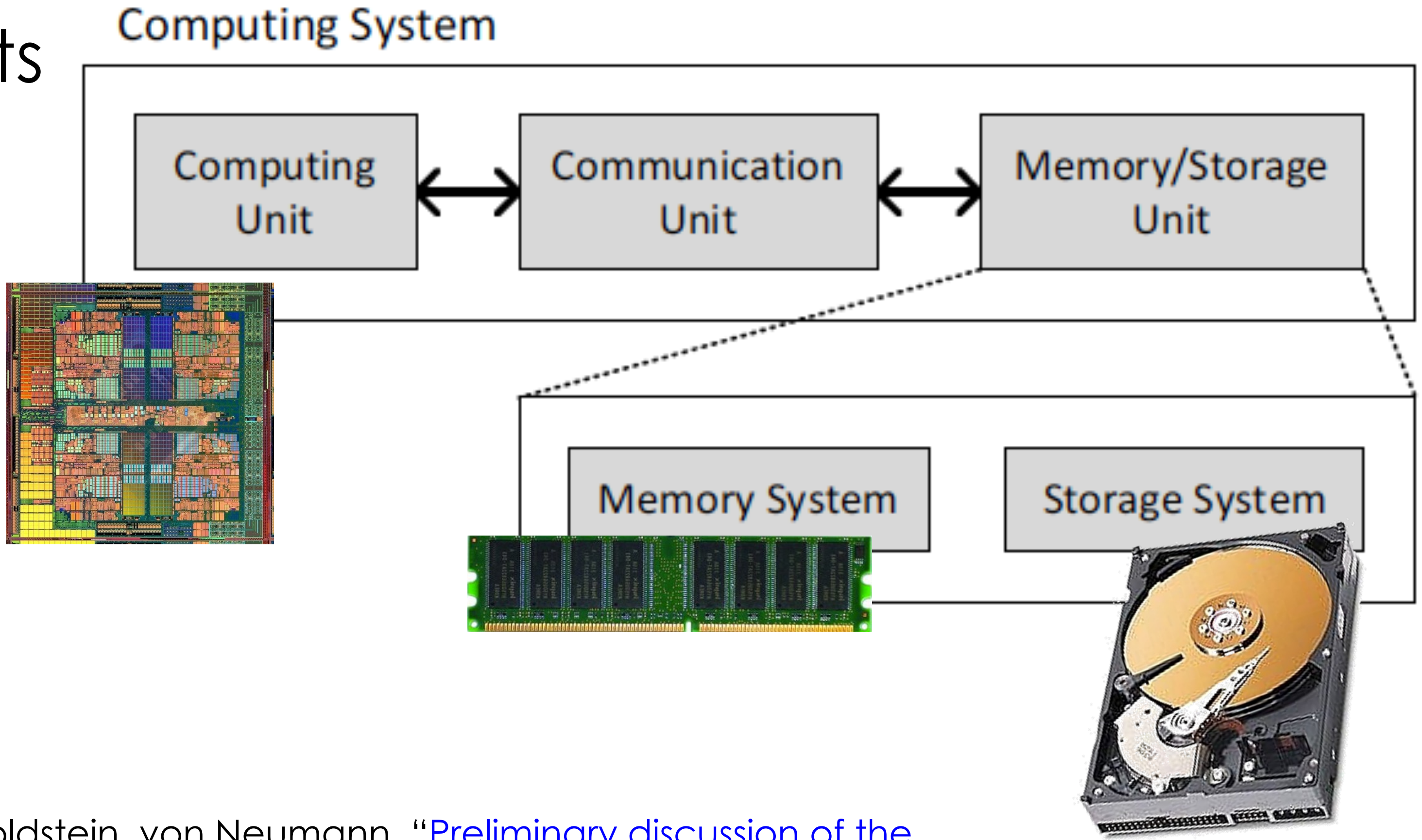
Why Processing-in-Memory (PIM)?

- The Problem
  - Data access is the major performance and energy bottleneck
  - Our current design principles cause great energy waste (and great performance loss)
- Processing of data is performed far away from the data

A Computing System

- Three key components
  - Computation
  - Communication
  - Storage/memory



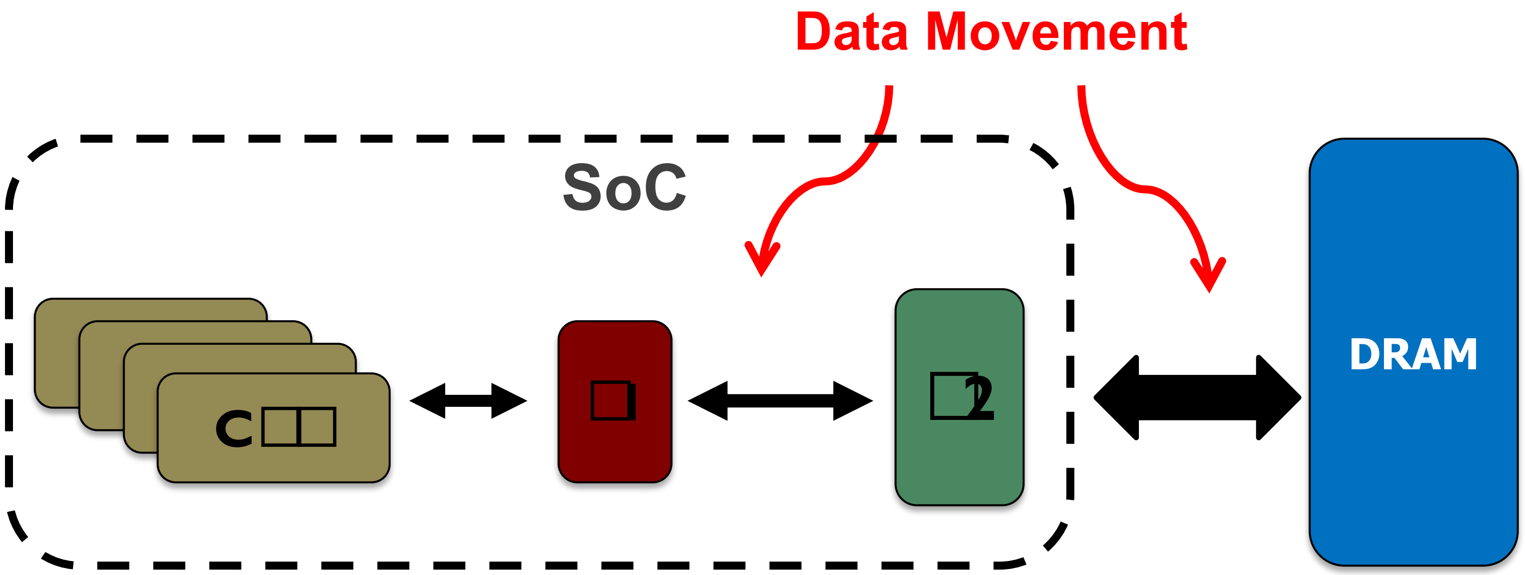


Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.

Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

Data Movement in Computing Systems

- Data movement dominates performance and is a major system energy bottleneck
  - Comprises 41% of mobile system energy during web browsing*
  - Costs 115x as much energy as an ADD operation**



Compute systems should be more data-centric

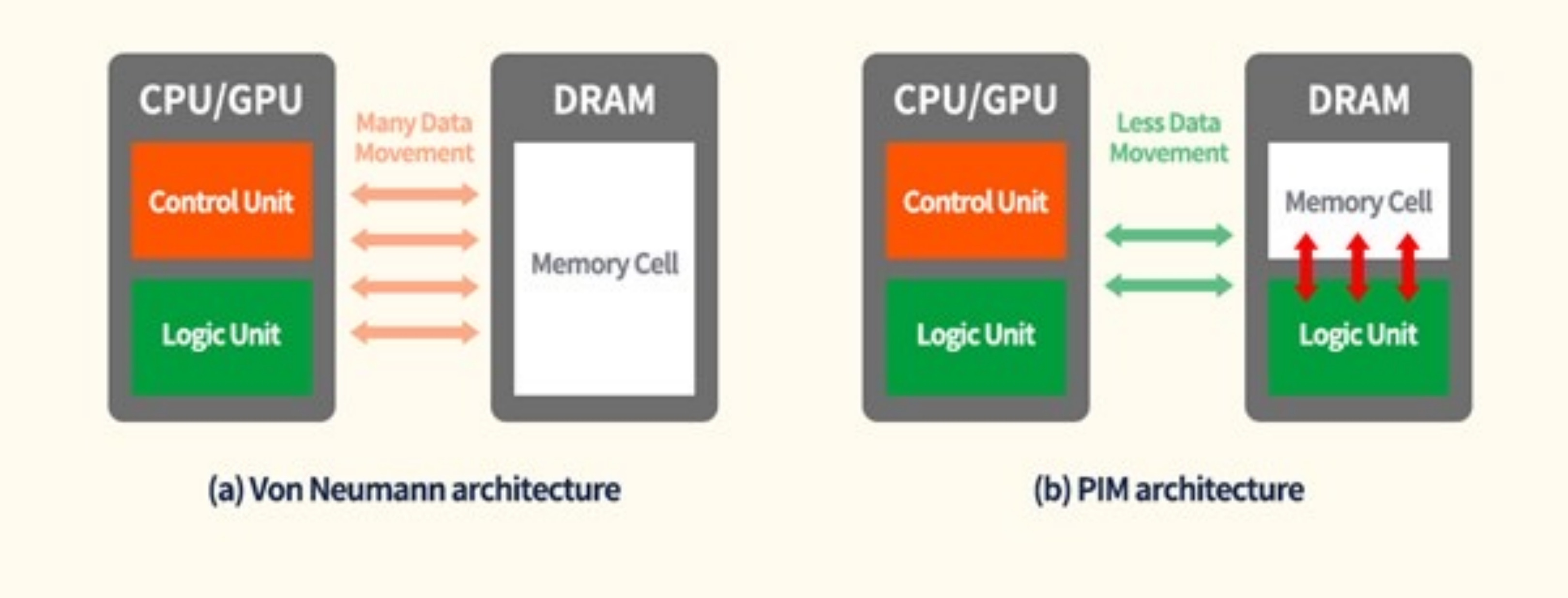
Processing-In-Memory proposes computing where it makes sense (where data resides)

*Reducing data Movement Energy via Online Data Clustering and Encoding (MICRO'16)

**Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC'14)

We Need A Paradigm Shift To …

- Enable computation with minimal data movement
- Compute where it makes sense (where data resides)
- Make computing architectures more **data-centric**



(a) Von Neumann architecture

(b) PIM architecture

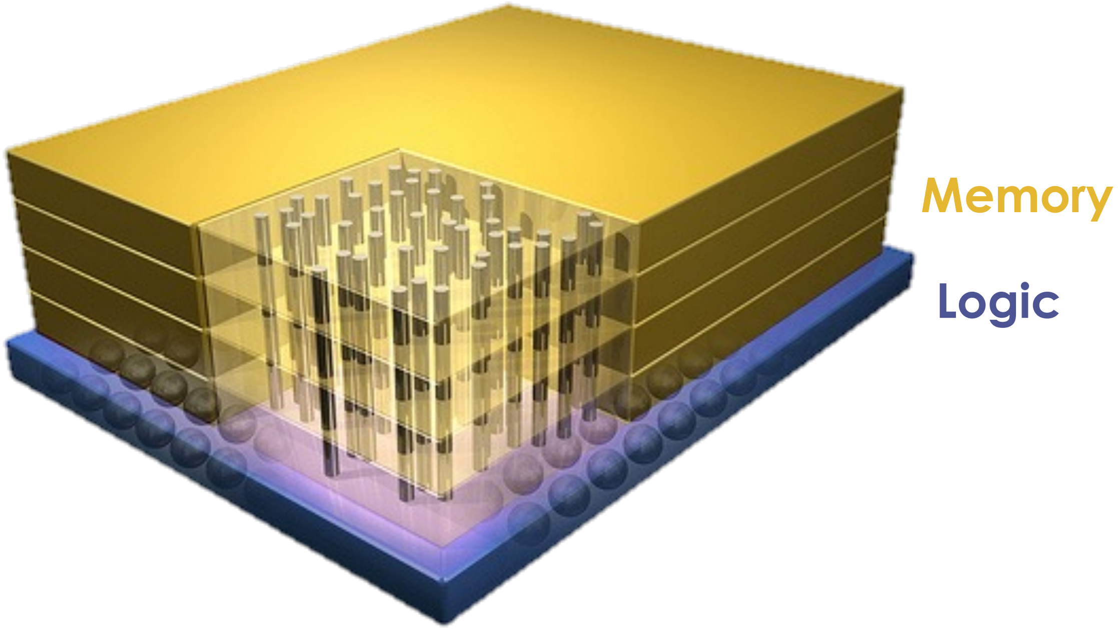
- **High-Performance**
- **Energy-Efficient**
- **Low-Latency**

Processing in Memory: Two Directions

Minimally Changing Memory Chips

- DRAM has great capability to perform bulk data movement and computation internally with small changes
  - Can exploit internal bandwidth to move data
  - Can exploit analog computation capability
  - …
- Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM
  - RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
  - Fast Bulk Bitwise AND and OR in DRAM (Seshadri et al., IEEE CAL 2015)
  - Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
  - "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology" (Seshadri et al., MICRO 2017)

Exploiting 3D-Stacked Memory



OS Security

- Virtualization provides protection for OS, but far from enough.
  - And we don’t fully isolate processes anyway.
- Common threats faced by OS
  - Malware: viruses, worms, trojan horses…
  - DDoS (Distributed Denial of Service):
    - overwhelm a system’s resources and stops serving legitimate requests.
  - Network intrusion
    - occurs when an individual gains access to a system for improper use
    - Careless insiders, Malicious insiders, Masqueraders, Clandestine users
  - Buffer overflow

Security Goals and Policies

- **Confidentiality**
  - If some piece of information is supposed to be hidden from others, don't allow them to find it out
- **Integrity**
  - If some piece of information or component of a system is supposed to be in a particular state, don't allow an adversary to change it.
  - Authenticity: the information was created by a particular party and not by an adversary
- **Availability**
  - If some information or service is supposed to be available for your own or others' use, make sure an attacker cannot prevent its use.

OS Security Mechanisms

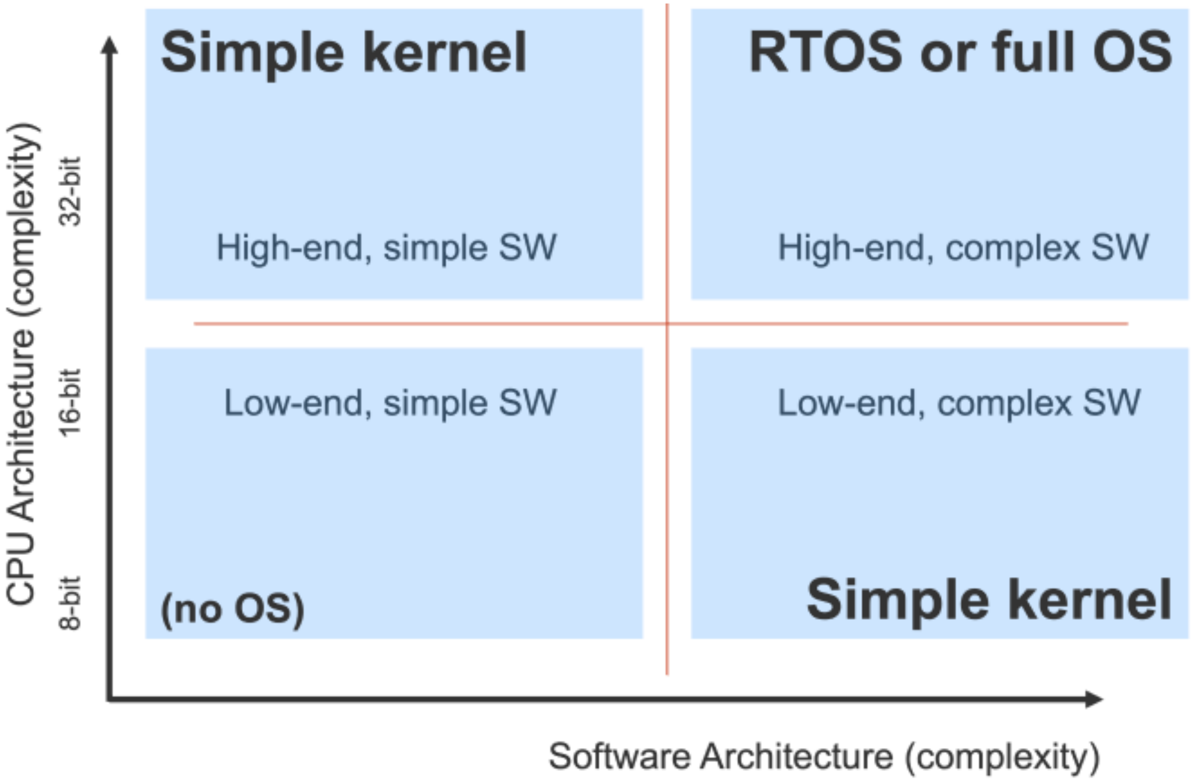
- Authentication
  - Username and password
  - User attribution identification/Biometrics, e.g., fingerprints, face recognition.
  - One-time password
  - Tokens: a physical access card or key
- Cryptography
  - e.g., ssh, encrypted information
- Access Control
  - Decide if a particular request made by a particular process belonging to a particular user at some given moment should or should not be granted
- OS Virtualization: VMs, containers
- TEE: Trusted Execution Environment

Beyond General-Purpose OSs

- So far, we focus on OSs for general-purpose computers
  - Windows, MacOS, Linux, etc
  - Run on a non-embedded device, which is a computer that works on its own and is the end product itself.
  - Desktops, laptops, servers, etc
- IoT Era: A wide variety of devices, embedded systems everywhere
  - Wireless Routers
  - Wearables (e.g., smart wristband/watches)
  - Smart appliances (e.g., smart speakers, security cameras, smart TV, smart plugs, etc)
  - In-car/in-flight control systems and entertainment systems
  - Point of sale (POS) terminals
  - Traffic lights, ATMs, Sensors, Printers, GPS navigation systems, digital cameras, many more electronic devices

Do we need an OS at all?

- It might require an OS if the computer's functionality is complex, but not necessarily.



- Abstract hardware
- Enable low power operation
- Manage concurrency
- Manage scheduling
- Provide shared libraries
- Virtualize hardware resources
- Meet resource constraints

Mobile and Embedded OS

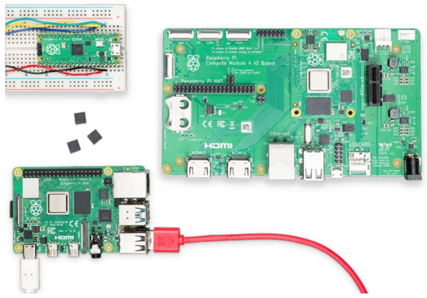
- Mobile OSs are often categorized as Embedded OSs.

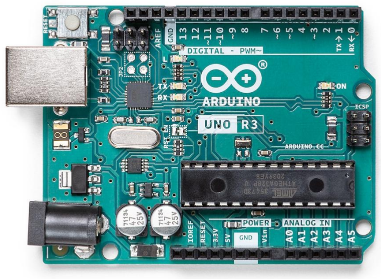
iOS

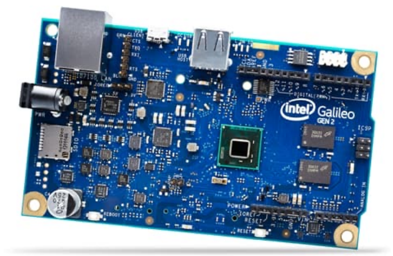
android

symbian

Mobile and Embedded OS







- Embedded System’s OS
  - Integrated into a device, limited in capabilities
  - Designed for some particular operations
- Run on resource (CPU/MCU, memory, storage, size, power, etc)-limited devices
  - An SD card contains all the code, or even reside on-chip
  - Applications and OS are distributed as a single image
- Many embedded devices run ARM-based Linux kernel on Arduino/Raspberry PI

Embedded OS: Examples

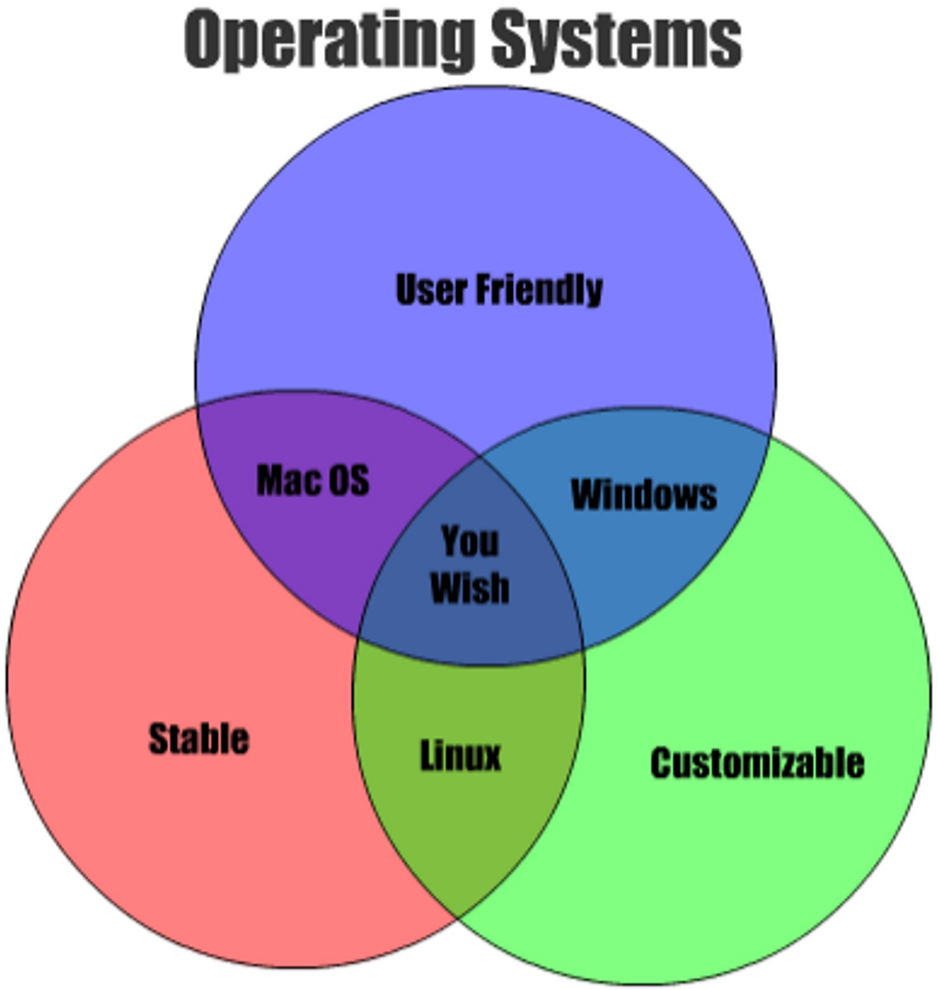
- Embedded Linux, VxWorks, QNX, INTEGRITY OS, etc
  - Embedded Linux: A general-purpose embedded OS
  - Yocto: a tool to build a customized Linux image
- Wireless Routers: OpenWrt
- IoT: TinyOS, Contiki, HarmonyOS, etc.
- RTOS: Real-Time Operating System
  - OS for real-time applications that processes data and events that have critically defined time constraints
  - Event-driven: priority-based switching
  - E.g., In-car Airbags is time-critical, while in-car entertainment is not
- Robot Operating System (ROS)
  - For robots. Not really an OS, more like middleware (sets of tools/libs)

Characteristics

- Power efficient
- Fewer storage capabilities
- Smaller processing power
- Fast and lightweight
- I/O device flexibility
- Real-time operation
- Tailored to the intended use case

Hardware of embedded devices

- 2018: ~256k RAM, ~512k flash, ~ 80 MHz
- 2008: ~10k RAM, ~48k flash, ~8 MHz
- 2 uA sleep, 10 mA active


Operating Systems
User Friendly
Mac OS
Windows
You Wish
Stable
Linux
Customizable

Future OS

- Where are OSs heading from here over the next decade?
- …More secure, reliable, trustworthy
  - Imagine computer systems taking more control of our lives in power grid, telephone networks, medical devices, automobiles, etc…
- Evolving with OS's old friend, the hardware…
  - Very large data centers
  - Very large scale multicore systems
  - Billions of "Things" - IoT devices
  - Very heterogeneous systems
  - Very large scale storage
- Maybe, we'll have "*Dust OS*" as well as "*Galaxy OS*" in decades…

Operating Systems

- **Virtualization**
    - CPU Virtualization
        - Process Abstract
            - Address space
            - Process states
            - Process control block
            - Process operations API
            - Signals
        - Limited Direct Execution
            - System calls
            - Context switch
            - Interrupts
        - Scheduling
            - Scheduling metrics
            - FIFO, SJF, HRRN, STCF, RR, MLFQ
            - Multi-core scheduling, Linux CFS
    - Memory Virtualization
        - Address space
        - Address translation: dynamic relocation
        - Segmentation
        - Paging
        - TLB
        - Multi-level paging
        - Inverted page table
        - Swap space
        - Page replacement policy: FIFO, LFR, LRU, Clock
        - Thrashing

- **Concurrency**
    - Thread
        - POSIX threads (pthreads)
        - Race conditions, critical sections, mutual exclusion, atomic operations, synchronization
    - Locks
        - Atomic instructions: test-and-set, compare-and-swap
        - Mutex locks
    - Condition Variables
        - Pthread CVs
        - Producer-Consumer problem
    - Semaphores
        - Binary Semaphores
        - Counting Semaphores
        - Ordering
        - Readers-Writers problem
    - Deadlock
        - Dining philosophers' problem
        - Four necessary conditions
        - Deadlock prevention, avoidance, detection&recovery

- **Persistence**
    - I/O devices (HDD, SSD)
    - Files and Directories
        - Inode
        - File descriptor
        - Hard/Symbolic links
    - File System Implementation
        - On-disk data structure
            - Superblock, Bitmap, Inodes, Data blocks
        - Free space management
            - Bitmap, linked-list, block-list
        - Caching and buffering
        - Access control and protection
        - Journaling file system
            - Data journaling
            - Metadata journaling
- ***Advanced Topics***
    - *Processing-In-Memory*
    - *OS Security*
    - *Mobile and embedded OS*