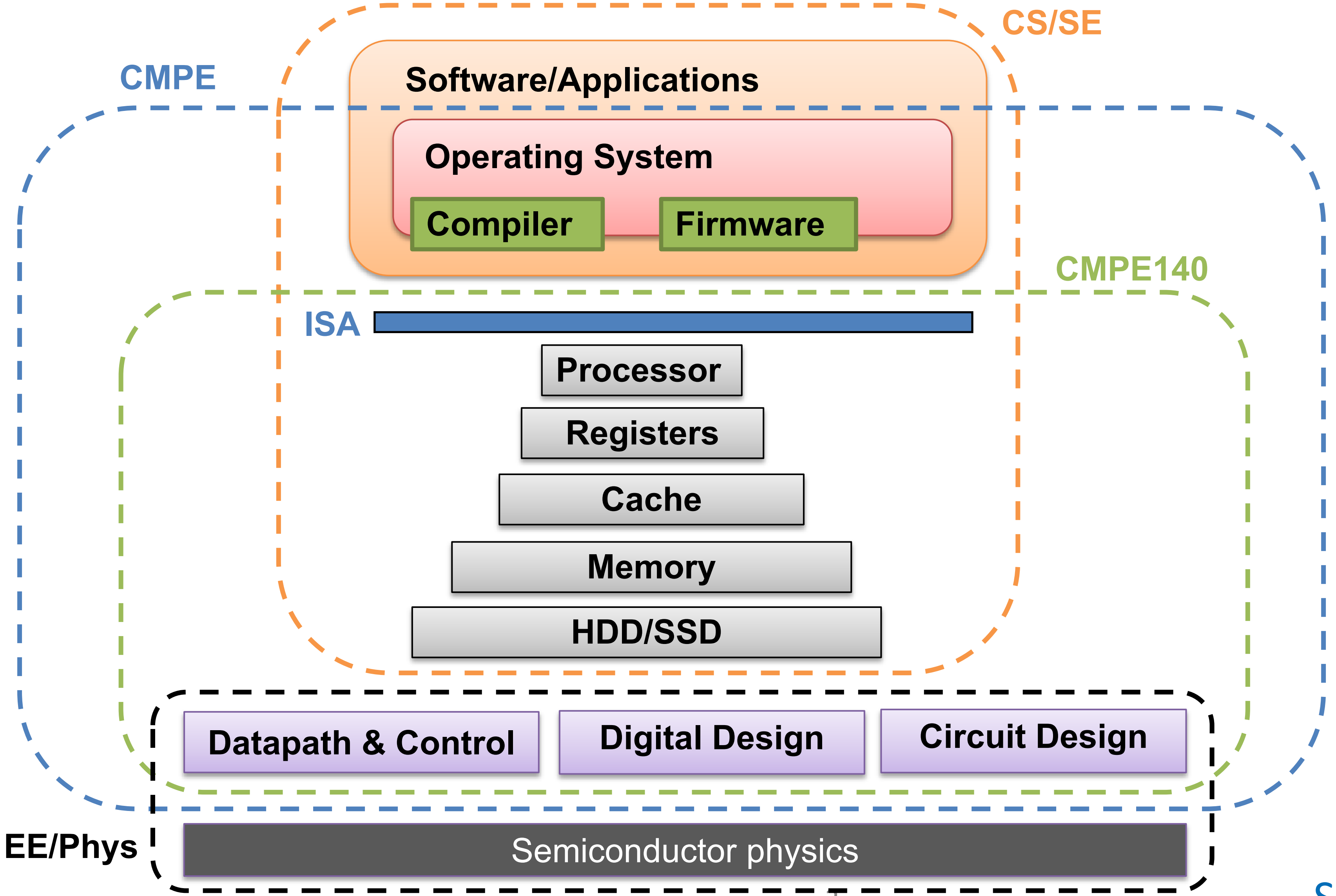
CMPE 200
Computer Architecture & Design

Lecture 1. Computer Architecture & Design Overview

Haonan Wang

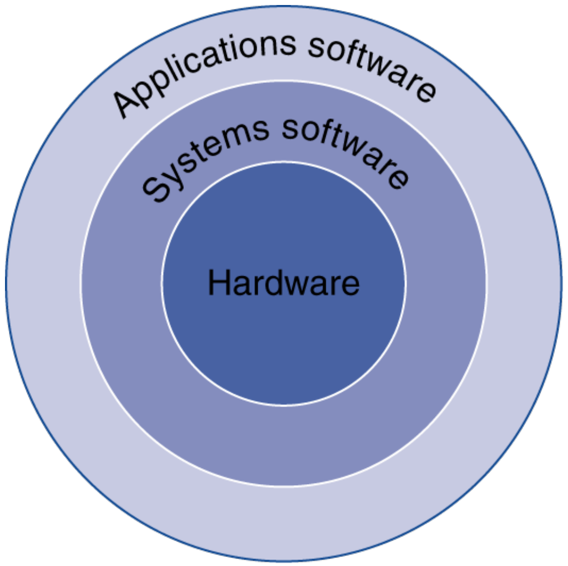
SJSU SAN JOSÉ STATE UNIVERSITY

Abstraction of Computers



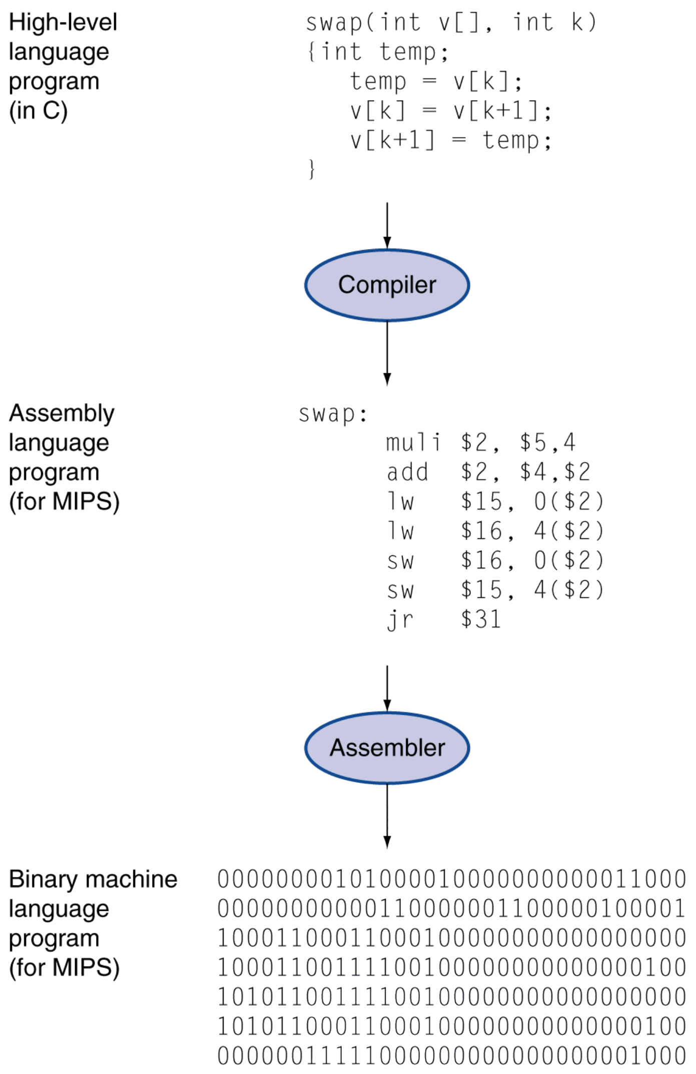
The Software's Point of View

- **Application software**
  - Written in high-level language
- **System software**
  - **Compiler:** translates HLL code to machine code
  - **Operating System:** service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- **Hardware**
  - Processor, memory, I/O controllers, …



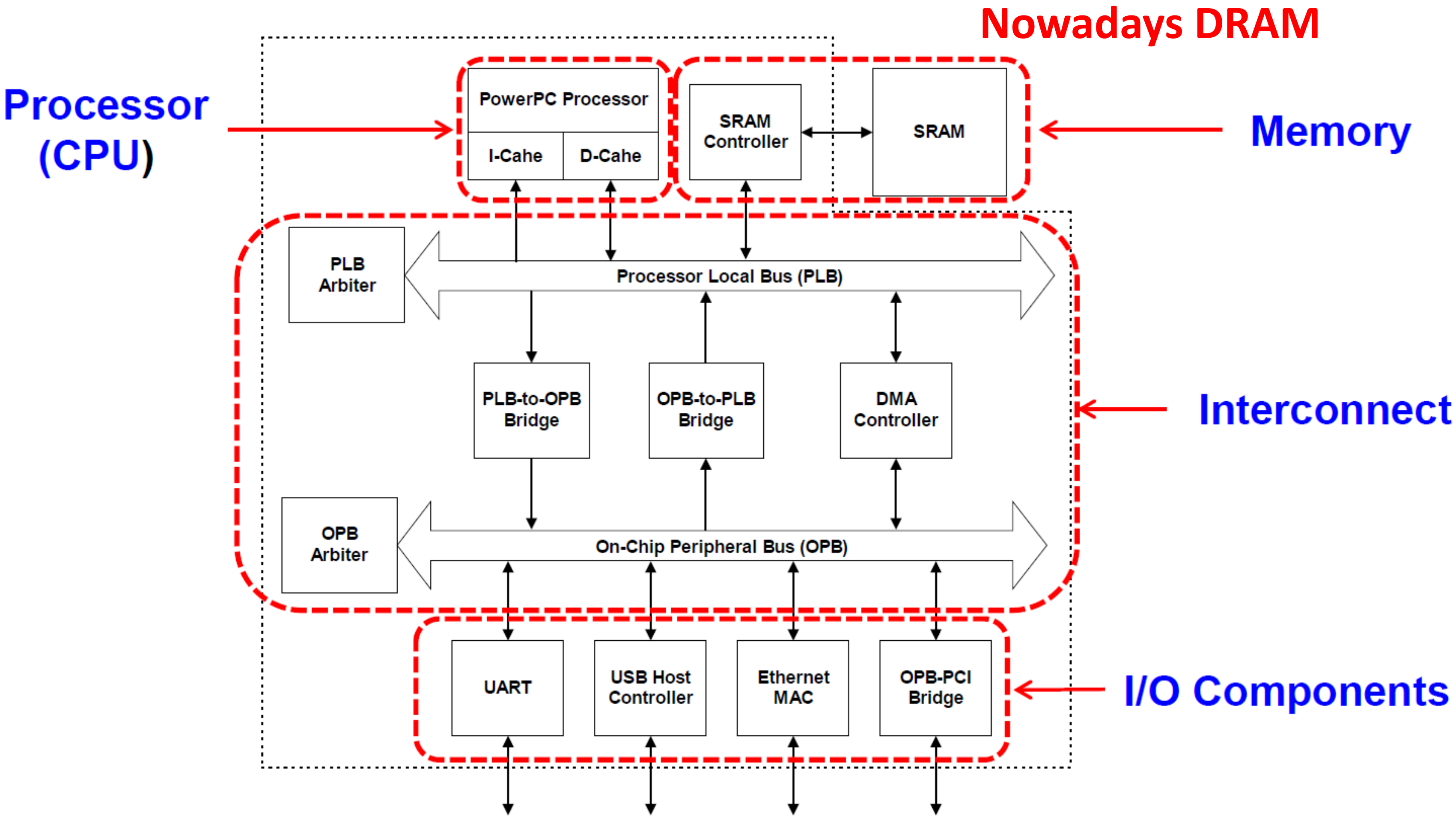
The Codes’ Point of View

- **High-level language**
  - Abstraction Level closer to problem domain
  - Provides for productivity and portability
- **Assembly language**
  - Textual representation of instructions
- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data



Hardware's Point of View

Computer Organization:



Your Thoughts

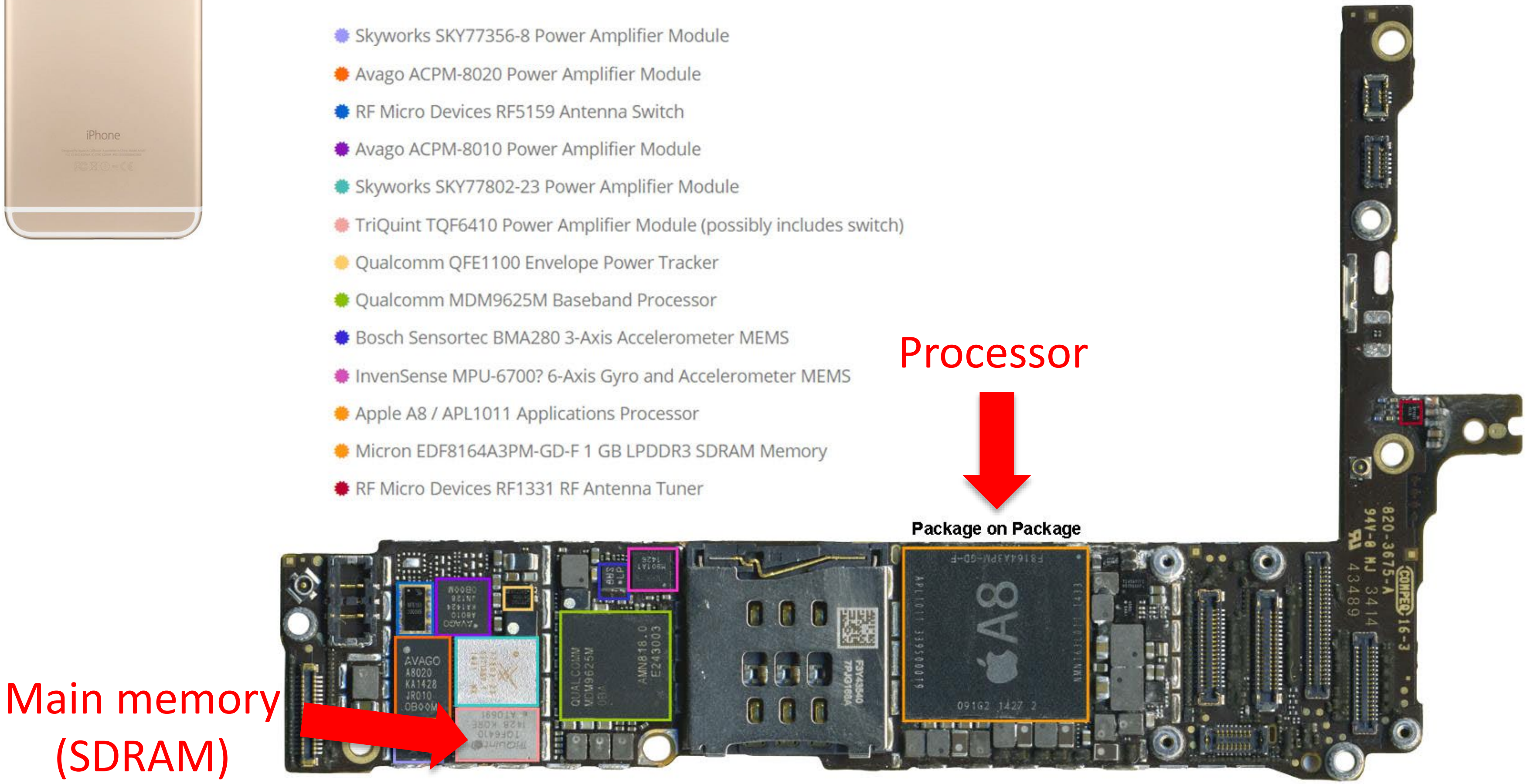
Try to conclude with as few words as possible:

Computer Organization vs. Computer Architecture

– What is the key difference?

Hardware Examples

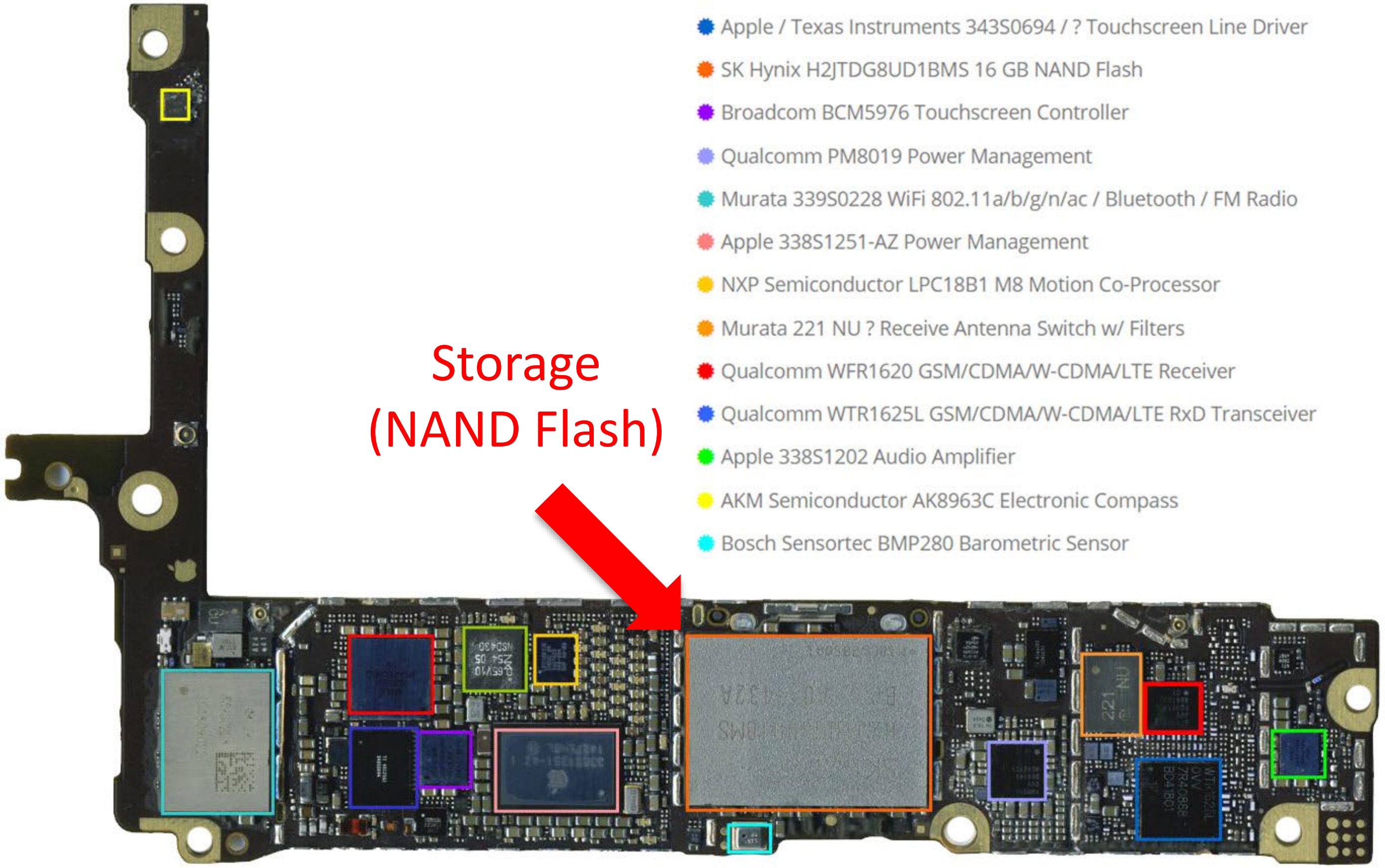
- Skyworks SKY77356-8 Power Amplifier Module
- Avago ACPM-8020 Power Amplifier Module
- RF Micro Devices RF5159 Antenna Switch
- Avago ACPM-8010 Power Amplifier Module
- Skyworks SKY77802-23 Power Amplifier Module
- TriQuint TQF6410 Power Amplifier Module (possibly includes switch)
- Qualcomm QFE1100 Envelope Power Tracker
- Qualcomm MDM9625M Baseband Processor
- Bosch Sensortec BMA280 3-Axis Accelerometer MEMS
- InvenSense MPU-6700? 6-Axis Gyro and Accelerometer MEMS
- Apple A8 / APL1011 Applications Processor
- Micron EDF8164A3PM-GD-F 1 GB LPDDR3 SDRAM Memory
- RF Micro Devices RF1331 RF Antenna Tuner



Figures from http://www.techinsights.com/teardown.com/apple-iphone-6/

Hardware Examples

Input/output device
(touch screen)



Storage
(NAND Flash)

- NXP Semiconductor PN548 NFC Controller w/ Secure Element Chip
- Apple / Texas Instruments 343S0694 / ? Touchscreen Line Driver
- SK Hynix H2JTDG8UD1BMS 16 GB NAND Flash
- Broadcom BCM5976 Touchscreen Controller
- Qualcomm PM8019 Power Management
- Murata 339S0228 WiFi 802.11a/b/g/n/ac / Bluetooth / FM Radio
- Apple 338S1251-AZ Power Management
- NXP Semiconductor LPC18B1 M8 Motion Co-Processor
- Murata 221 NU ? Receive Antenna Switch w/ Filters
- Qualcomm WFR1620 GSM/CDMA/W-CDMA/LTE Receiver
- Qualcomm WTR1625L GSM/CDMA/W-CDMA/LTE RxD Transceiver
- Apple 338S1202 Audio Amplifier
- AKM Semiconductor AK8963C Electronic Compass
- Bosch Sensortec BMP280 Barometric Sensor

Figures from http://www.techinsights.com/teardown.com/apple-iphone-6/

Hardware Examples

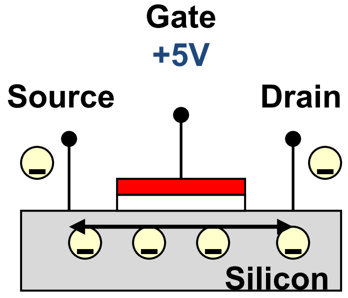
What is the advantage of having them in the same package?

Figures from http://www.anandtech.com/show/8562/chipworks-a8

Electronic Components' Point of view

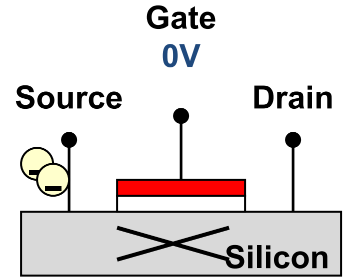
- **Transistors**
  - 3-terminal device
  - Gate input: the control input; its voltage determines whether current can flow
  - Source & Drain: terminals that current flows from/to

- **Many transistors can be fabricated on one piece of silicon** (i.e. an integrated chip, IC)



Transistor is 'on'

High voltage at gate allows current to flow between source and drain



Transistor is 'off'

Low voltage at gate prevents current from flowing between drain and source

Integrated Circuit

Actual silicon wafer is quite small but can contain several billion transistors

Silicon wafer is then packaged to form the chips we are familiar with

Moore’s Law (1965)

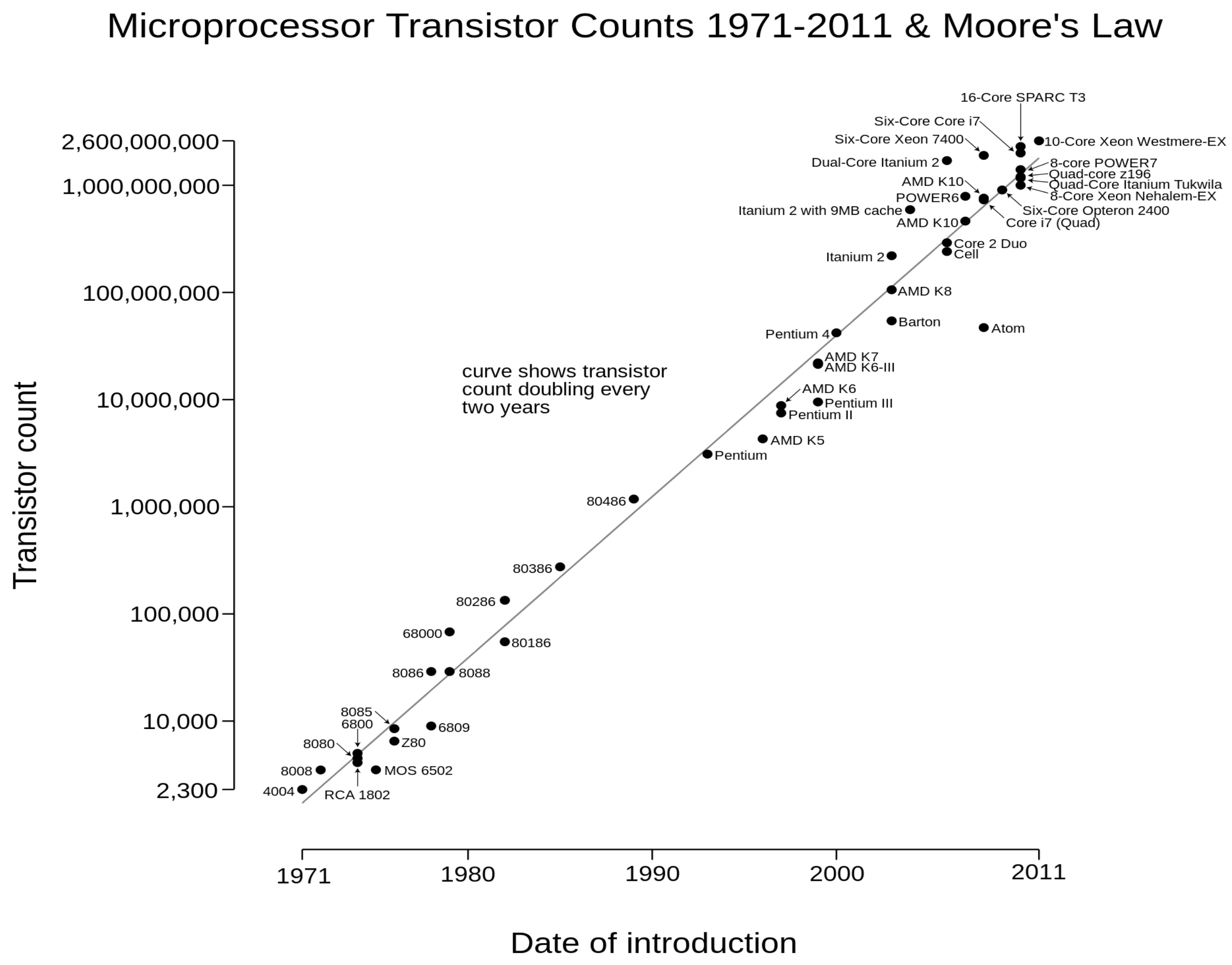
In 1975, he recalibrated it to every two years. Later, it is recalibrated again to 18 months, as is widely known as Moore’s Law.

“The number of transistors per square inch on integrated circuits had doubled every year since the integrated circuit was invented and this trend would continue for the foreseeable future.”

-- Gordon E. Moore, “Cramming More Components onto Integrated Circuits,” Electronics, pp. 114–117, April 19, 1965.

Moore's Law (1965)

The law has been preserved over four decades..

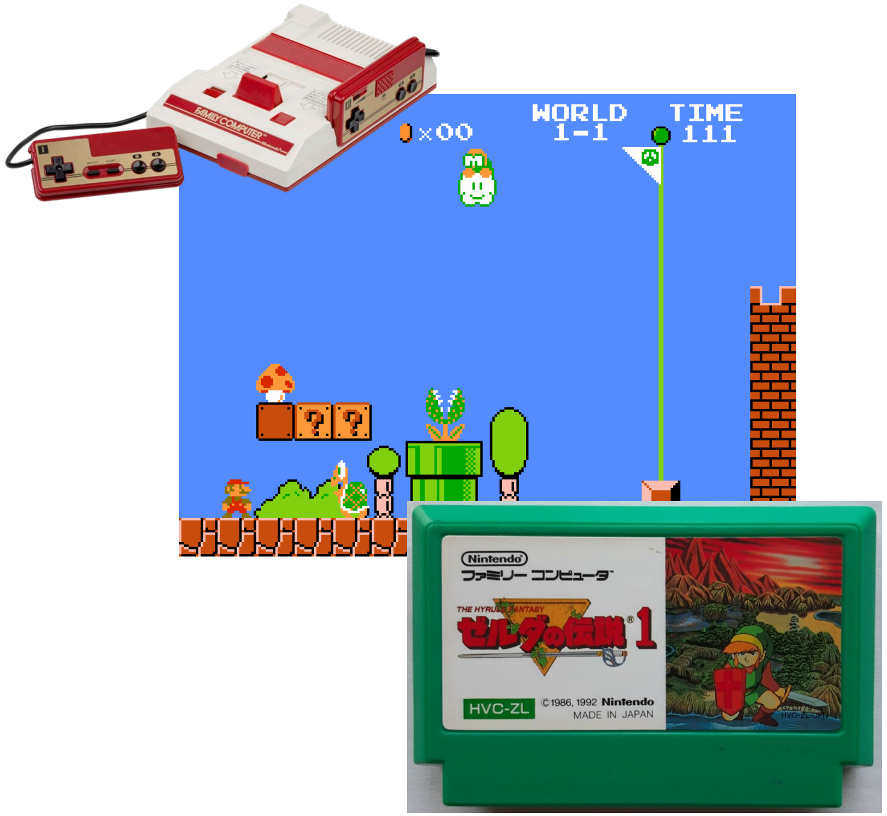


SJSU SAN JOSÉ STATE UNIVERSITY

What Does It Enable?

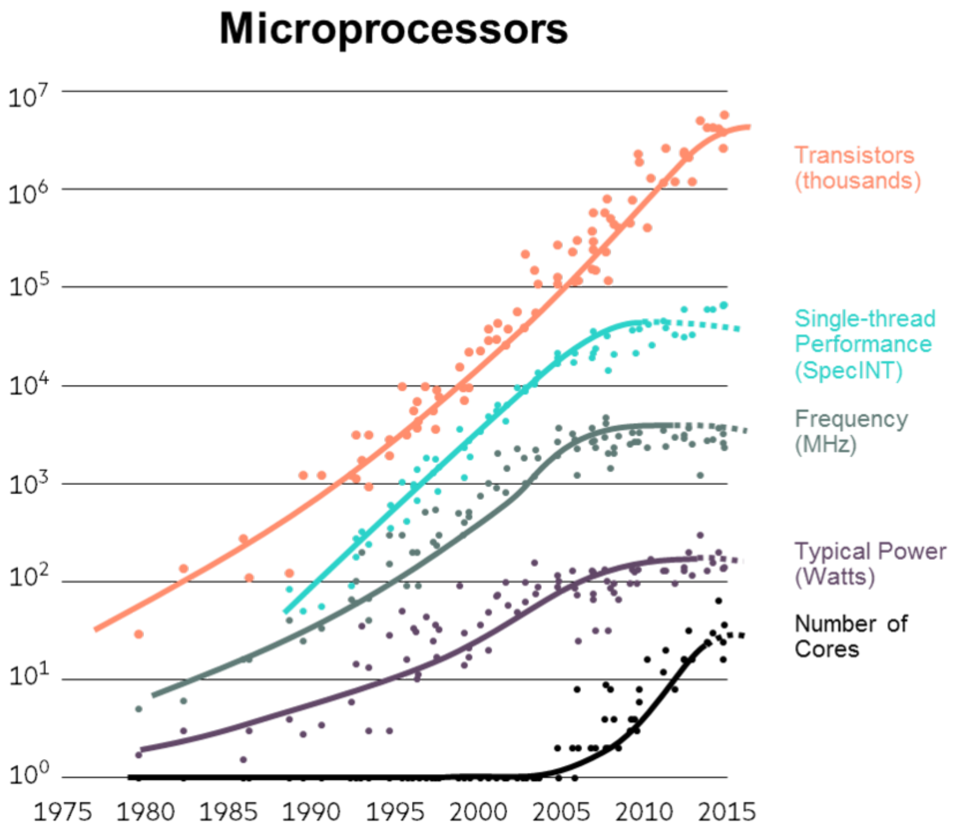
Applications that were impossible to run on the computers before

– 3D games, Augmented reality, Virtual reality, Deep learning…



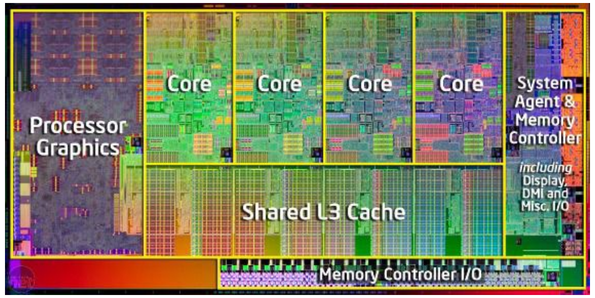
Moore's Law (1965)

Will it last forever?

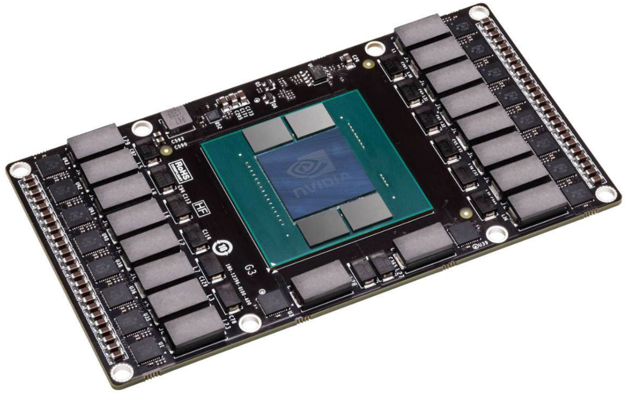


Dead?

Current Trend of Computer Systems



- **Multi- or Many-core Processors**
  - Embedding multiple cores in one CPU
- **Complex microarchitecture**
  - Out-of-order execution, branch predictor, memory prefetcher, etc..



- **Accelerators**
  - GPUs, FPGAs, etc..
  - Application specific designs
- **Novel architectures**
  - Dark silicon, 3D-stacking, neuromorphic computing, quantum, etc..

Metrics: How Do We Evaluate a System?

- **Performance (speed)**
  – Response time (Latency):
    - How long it takes to do a task
  – Throughput :
    - Total work done per unit time

- **Power efficiency**
  – How much power it consumes to do a task

- **Cost**

What Determines Performance?

- **Algorithm**
  - Determines number of operations executed
- **Programming language, compiler, architecture**
  - Determine number of machine instructions executed per operation
- **Processor and memory system**
  - Determine how fast instructions are executed
- **I/O system (including OS)**
  - Determines how fast I/O operations are executed

Relative Performance

- **Define Performance = 1/Execution Time**

- **"X is n times faster than Y"**

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

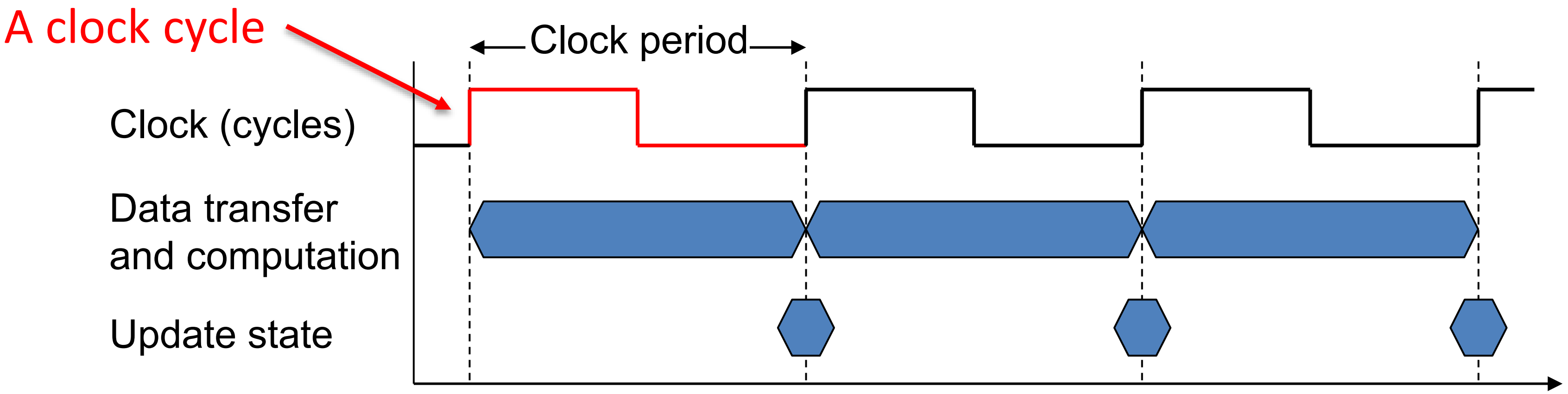
Measuring Execution Time

- **Elapsed time**
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- **CPU time**
  - Time spent processing a given job on CPU
    - Discounts I/O time, other jobs' shares
  - Estimating CPU time is relatively easy based on the number of lines of code and processing speed of the CPU

CPU Clocking

- **Operation of digital hardware governed by a constant-rate clock**

- **Different system components may have different clock rates**



Clock period: duration of a clock cycle

e.g., 250ps = 0.25ns = 250×10^{-12}s

Clock frequency (rate): cycles per second

e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time Estimation

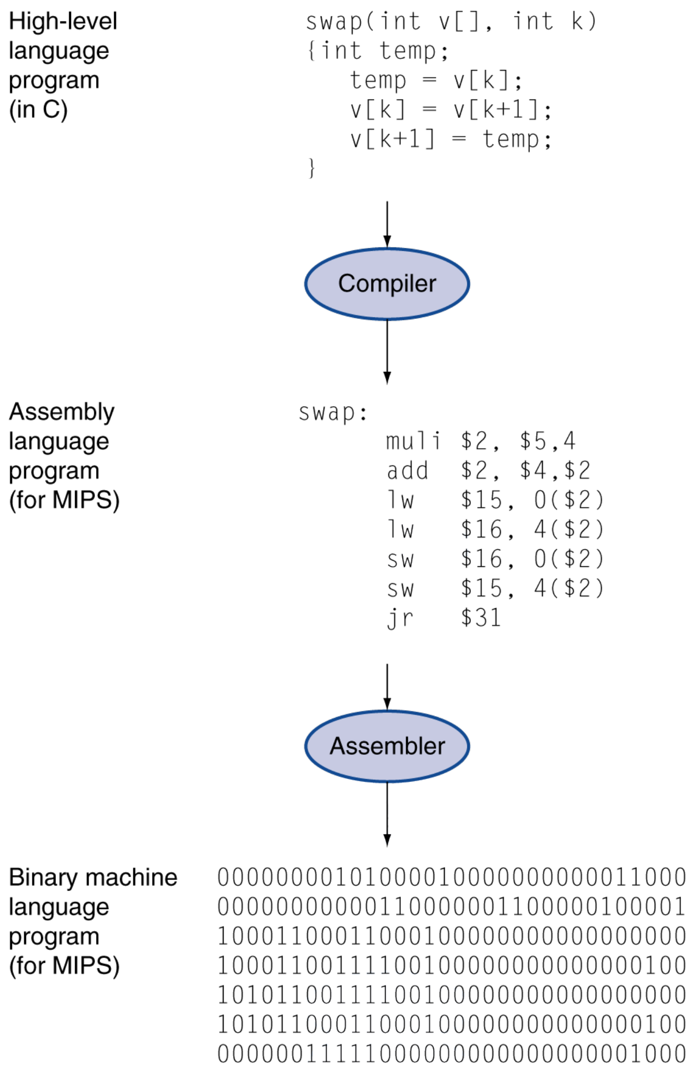
- A code line of high-level language program can be translated to **one or multiple** assembly code lines
- CPU processes a piece of machine code for each assembly code line, one by one
- **Example:** Application A has 1000 assembly lines. CPU has 250 ps clock period.

Clock cycles for Application A

= 1000 cycles

CPU time for Application A

= 1000 cycles x 250 ps = 250×10^{-9}s



CPU Time

CPU Time = Clock Cycles x Clock Period
= Clock Cycles / Clock Frequency

Instruction Count and CPI

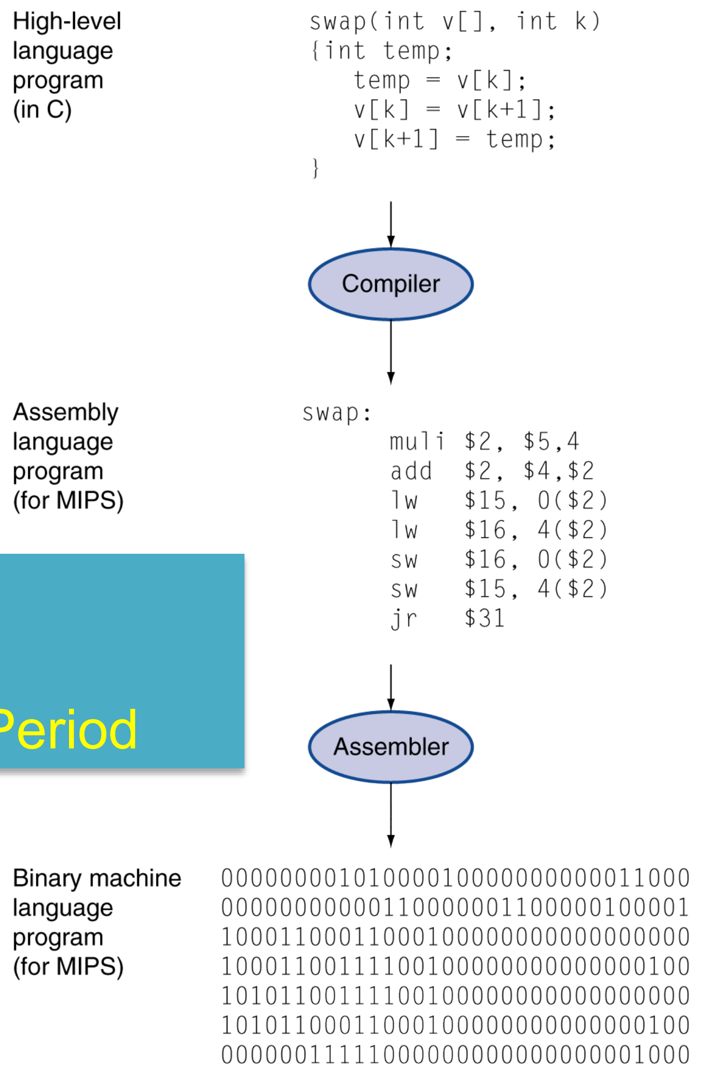
- An assembly line is also called one **Instruction**

- In some CPUs, an instruction may take **multiple clock cycles**

- **CPU Time equation can be rewritten**

CPU Time
= Clock Cycles x Clock Period
= Instruction Count x Cycles Per Instruction x Clock Period

- **Cycles Per Instruction** is also called **CPI**



CPI Example

- A program takes 33 billion instructions to run
- CPU processes instructions at 2 cycles per instruction
- Clock speed is 3GHz

What is estimated CPU Time for this program?

CPU Time = Instruction Count x CPI x Clock Period

$= 33 \times 10^9 \times 2 \times 1/(3 \times 10^9)$

$= 22$ seconds

CPI and Average CPI

- **In some CPUs, different types of instructions** (i.e. integer instruction vs. floating point instruction) **may take different number of cycles**

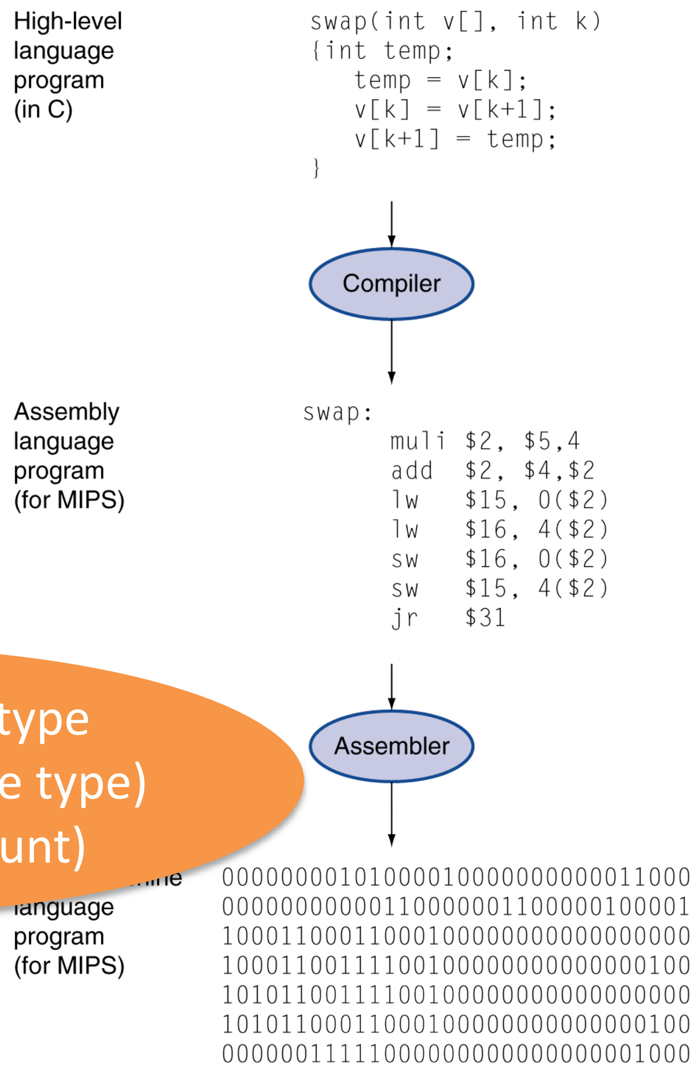
- **CPU Time equation can be rewritten**

$$\text{CPU Time} = \text{Clock Cycles} \times \text{Clock Period}$$
$$= \sum \left(IC_i \times CPI_i \right) \times \text{Clock Period}$$

Sum of (IC of each type instruction x CPI of the type)
(IC = instruction count)

- **Average CPI**

$$= \text{Total Clock Cycles} / \text{Total IC}$$
$$= \sum \left(IC_i \times CPI_i \right) / \text{Total IC}$$



Example: Average CPI

Instruction Type	Integer	Floating point	Branch	Load/ Store
CPI for type	1	2	4	3
Instruction Count in program A	50	10	10	10
Instruction Count in program B	20	0	10	10

- **Program A: Total Instructions = 80**

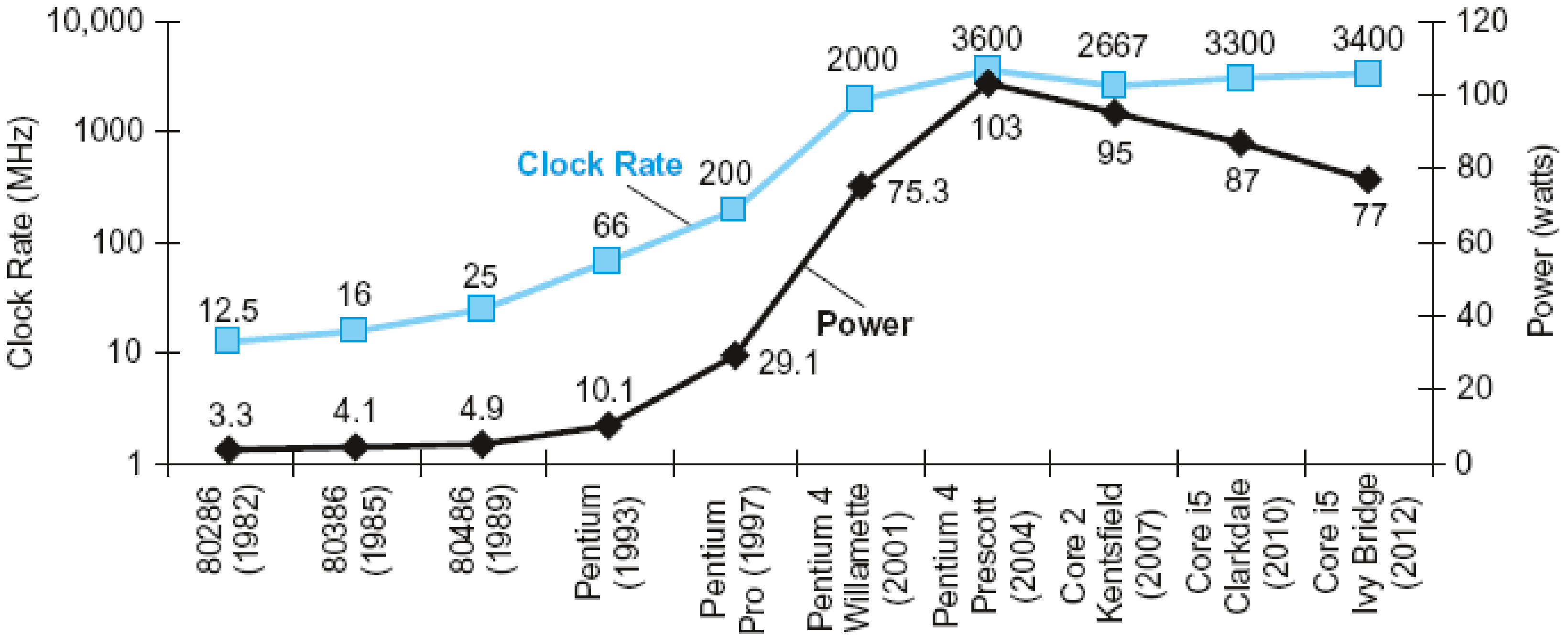
Clock Cycles = 50×1 + 10×2 + 10×4 + 10x3 = 140

Avg. CPI = 140/80 = 1.75

CPU time (if Clock period = 100 ps)

= 1.75 x 80 x 100ps = 14000 ps = 14ns

Power and Performance



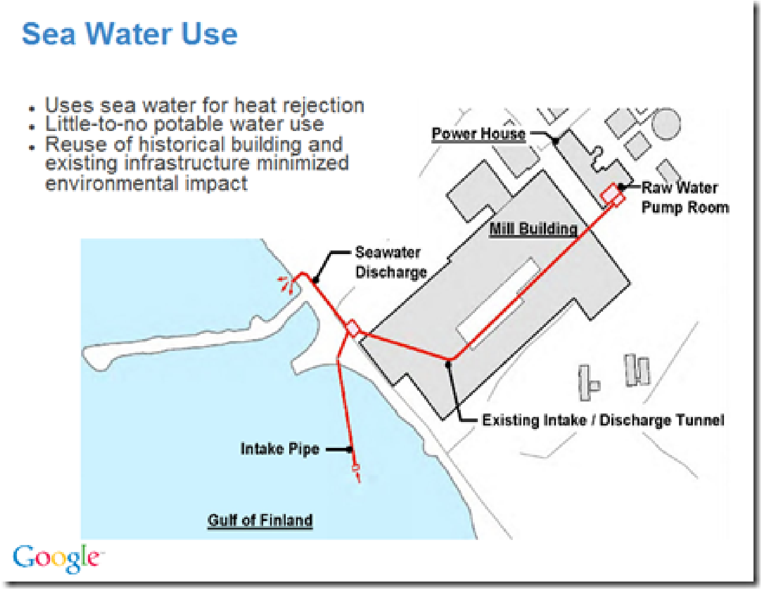
In CMOS IC technology

Power = Capacitive load x Voltage^2 x Frequency

Issue with Power Scaling

- **Increasing core frequency may burn your processor!**
  - Almost all energy consumption turns into heat

- **Cooling is costly**



- **Design more efficient system instead**
  - Multicore, etc..

SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY