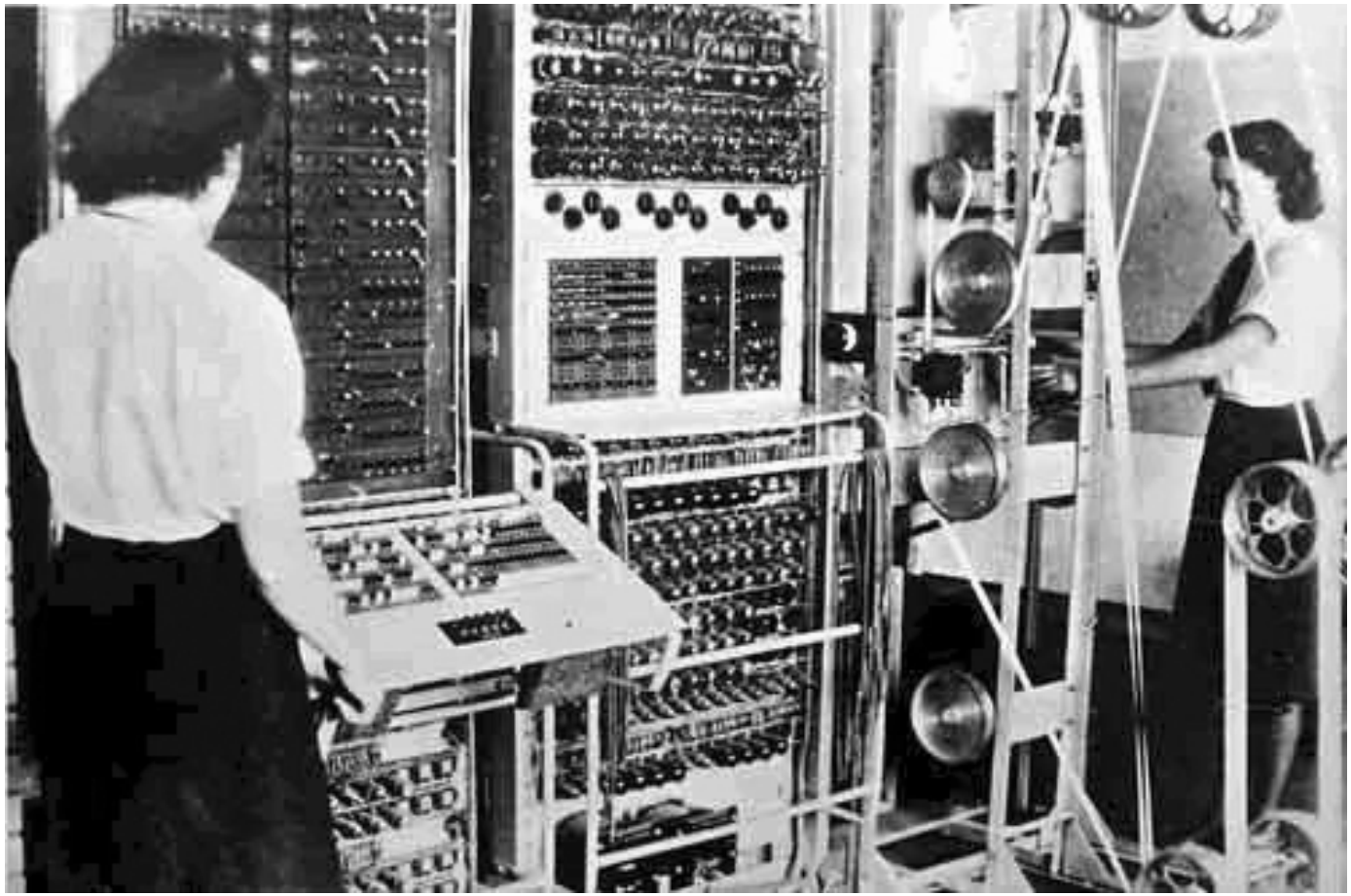
CS3350B Computer Architecture
Introduction

Marc Moreno Maza

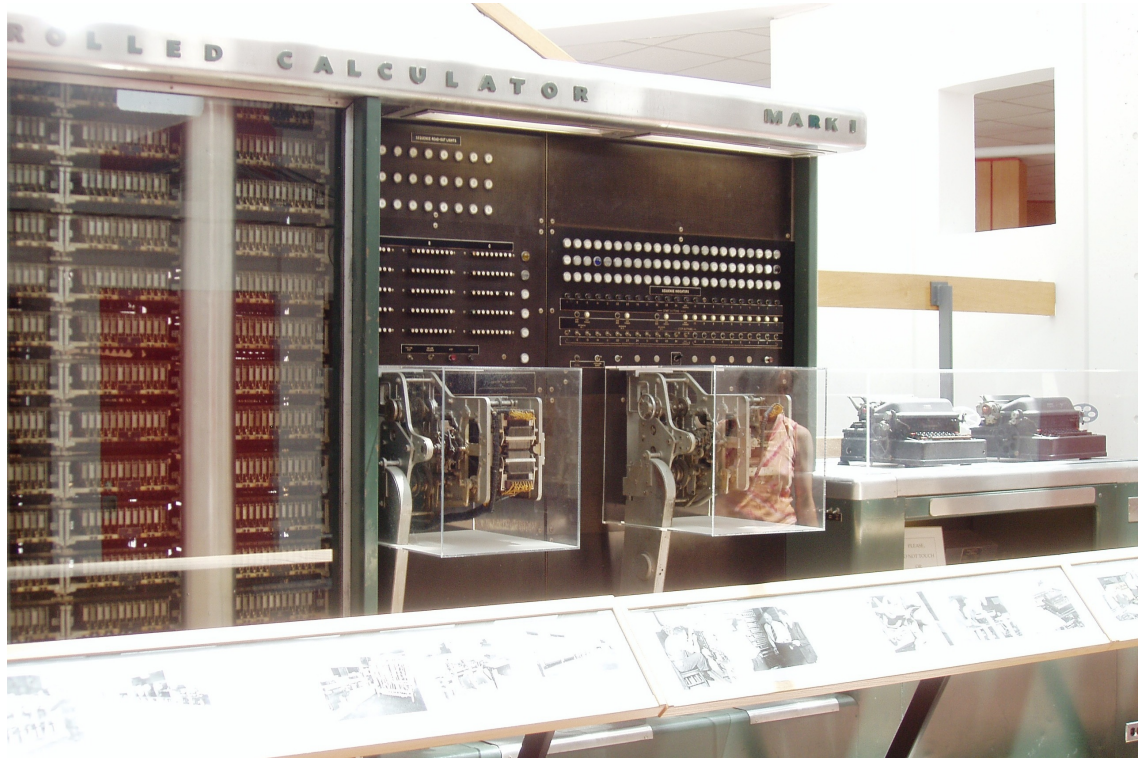
http://www.csd.uwo.ca/~moreno/cs3350_moreno/index.html
Department of Computer Science
University of Western Ontario, Canada

Thursday January 5, 2017

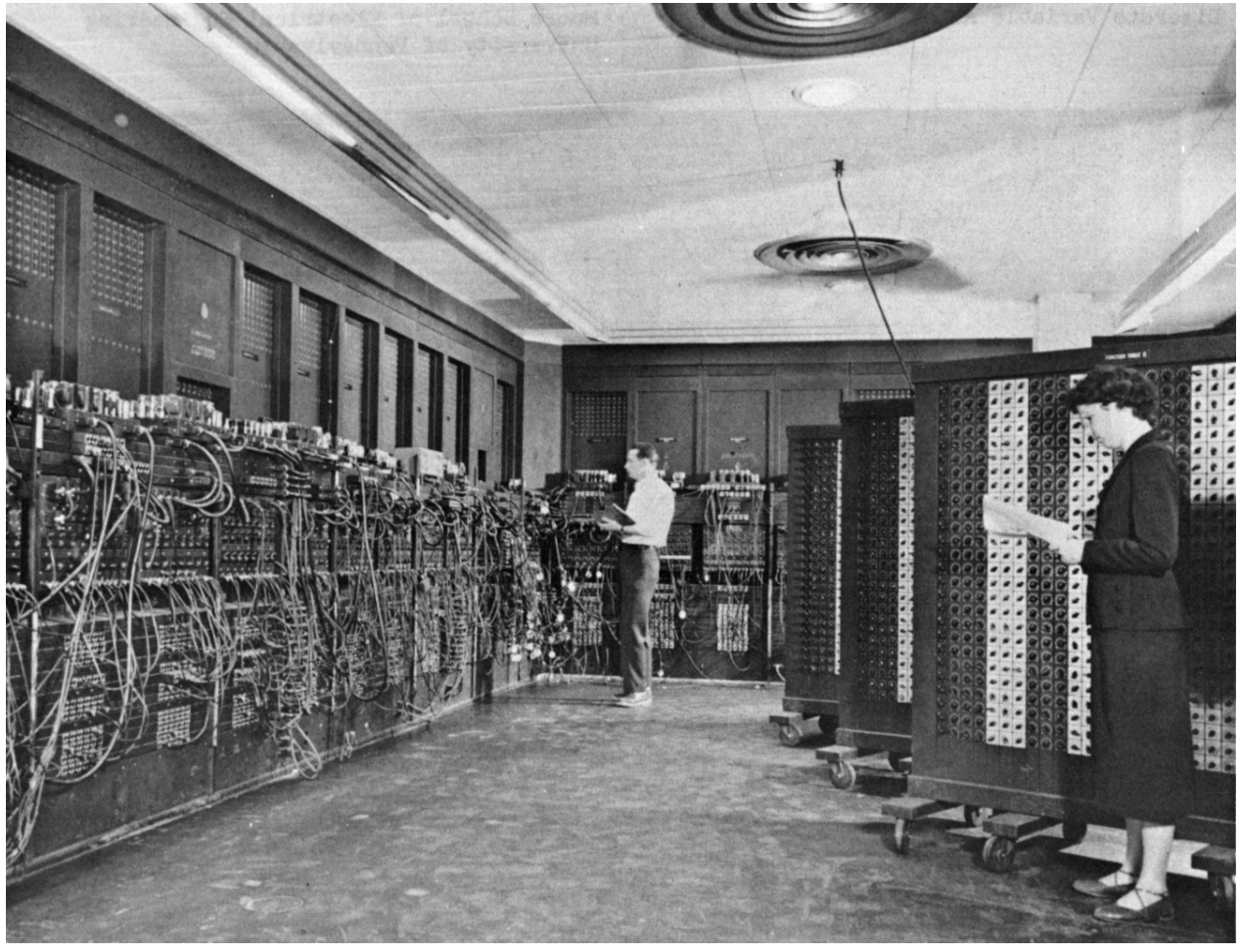
Konrad Zuse's Z3 electro-mechanical computer (1941, Germany).
Turing complete, though conditional jumps were missing.



Colossus (UK, 1941) was the world's first totally electronic programmable computing device. But not Turing complete.



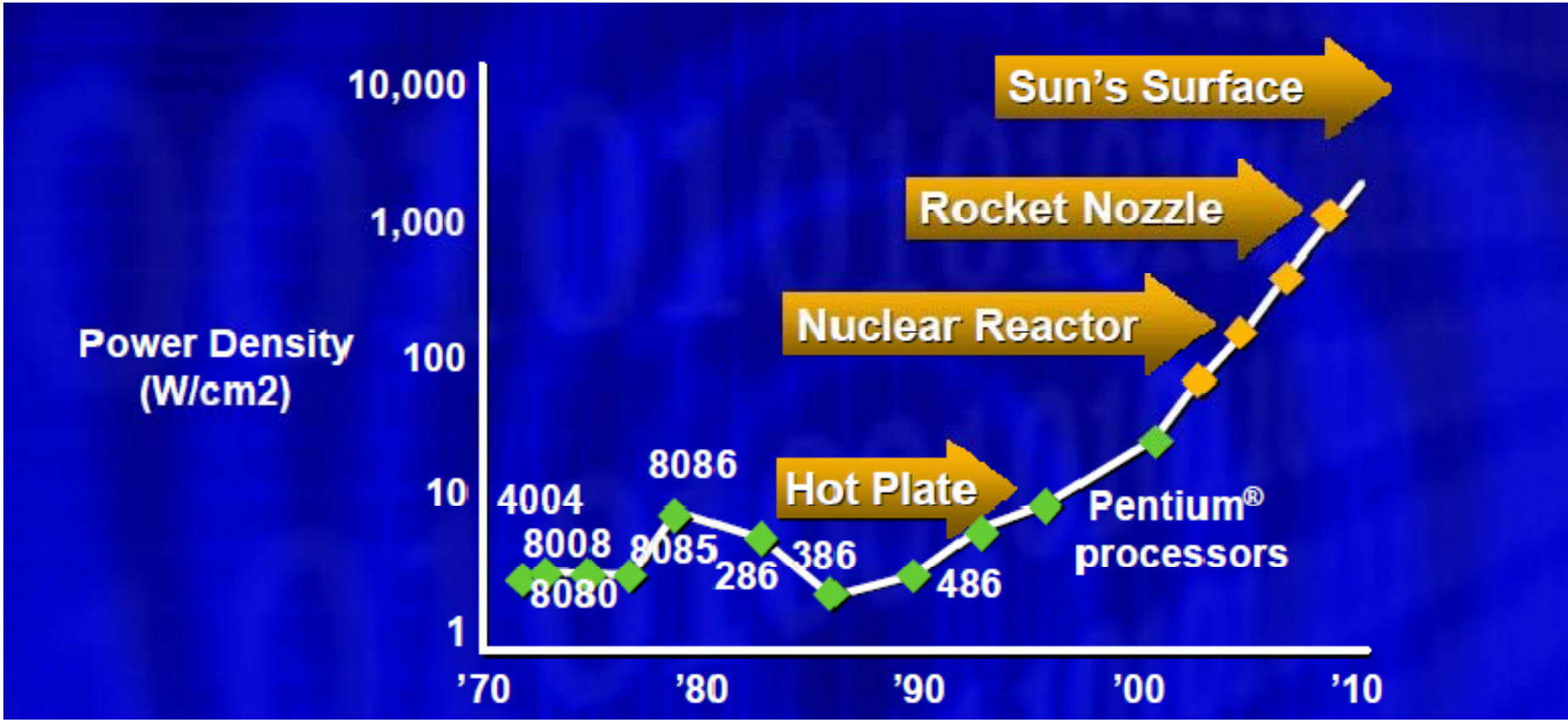
Harvard Mark I – IBM ASCC (1944, US). Electro-mechanical computer (no conditional jumps and not Turing complete). It could store 72 numbers, each 23 decimal digits long. It could do three additions or subtractions in a second. A multiplication took six seconds, a division took 15.3 seconds, and a logarithm or a trigonometric function took over one minute. A loop was accomplished by joining the end of the paper tape containing the program back to the beginning of the tape (literally creating a loop).

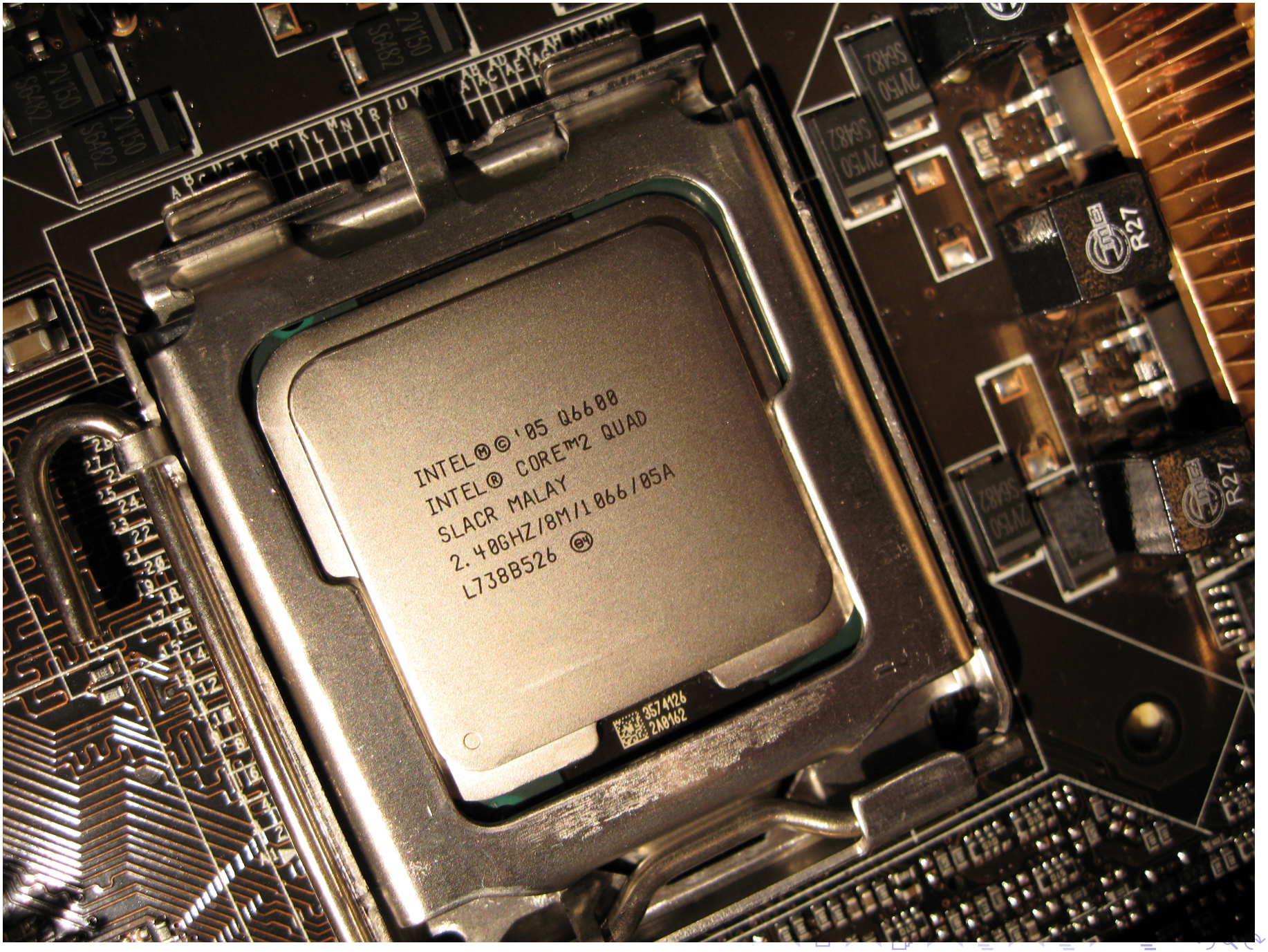


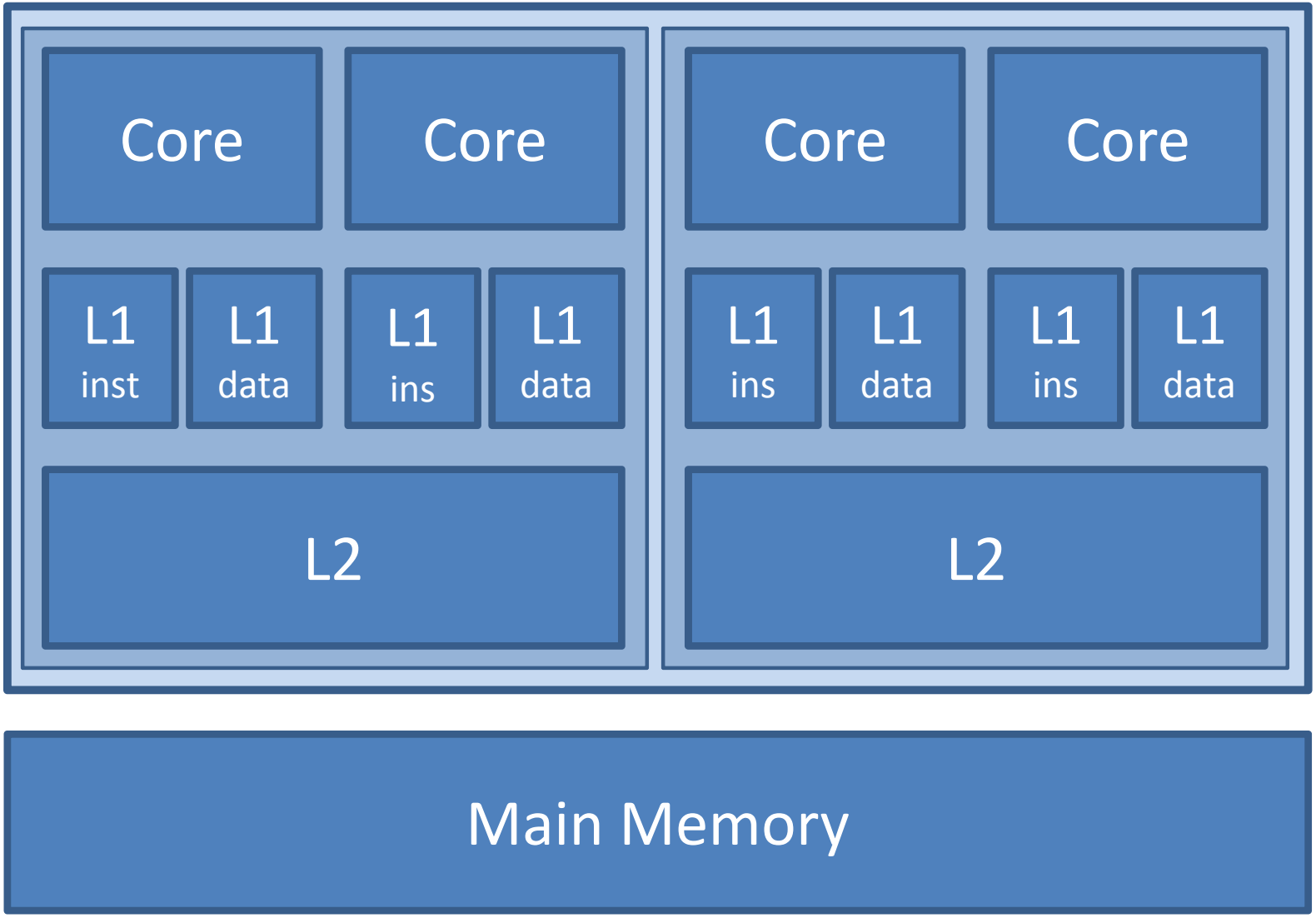
Electronic Numerical Integrator And Computer (ENIAC). The first general-purpose, electronic computer. It was a Turing-complete, digital computer capable of being reprogrammed and was running at 5,000 cycles per second for operations on the 10-digit numbers.

The IBM Personal Computer, commonly known as the IBM PC (Introduced on August 12, 1981).

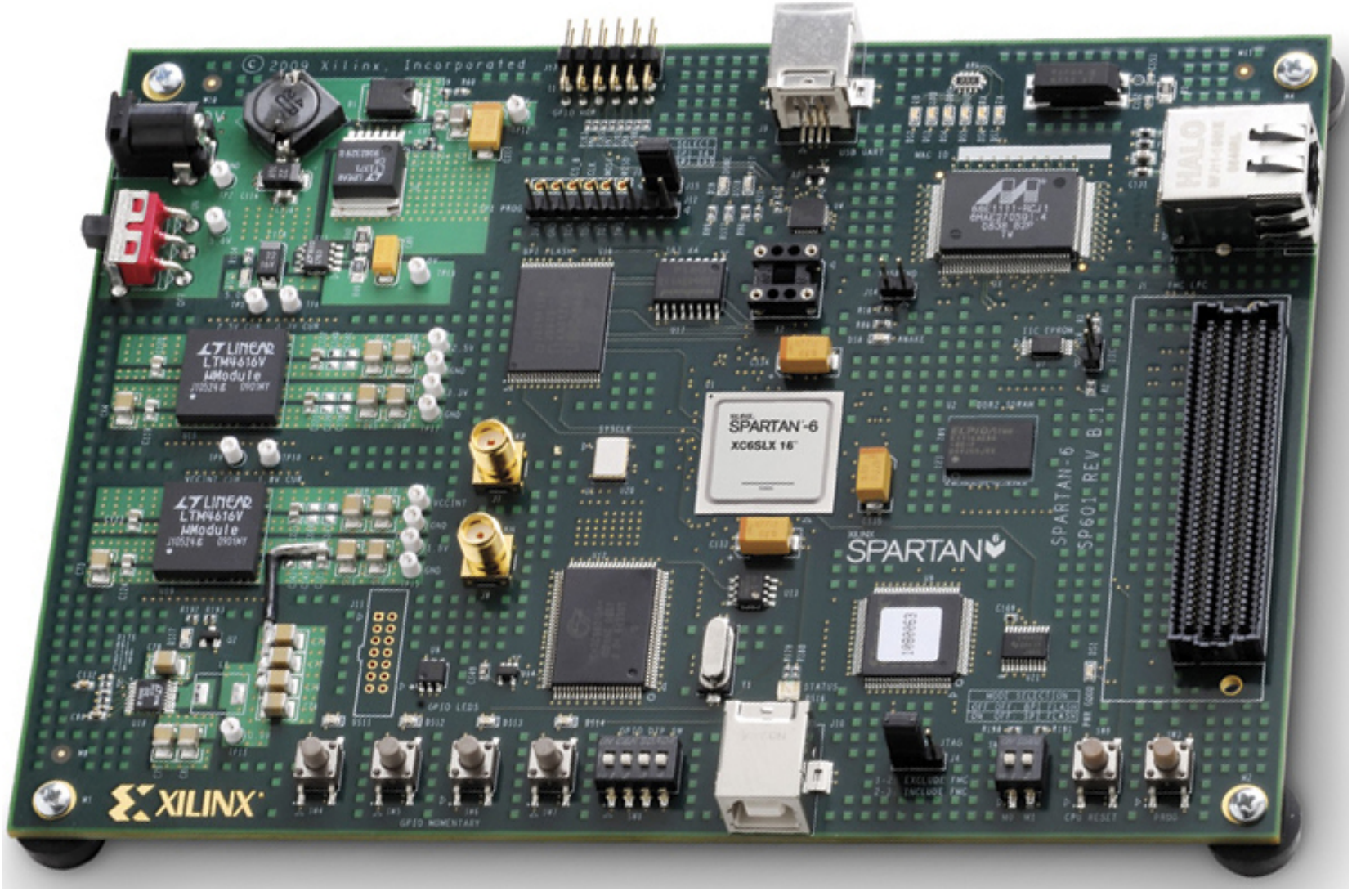
The Pentium Family.


10,000
1,000
100
10
1
Power Density
(W/cm2)
Sun's Surface
Rocket Nozzle
Nuclear Reactor
Hot Plate
4004
8008
8080
8085
8086
286
386
486
Pentium®
processors
'70
'80
'90
'00
'10


INTEL©'05 Q6600
INTEL® CORE™2 QUAD
SLACR MALAY
2.40GHZ/8M/1066/05A
L738B526
3S74126
2A0162
R27
R27


Core
Core
Core
Core
L1
inst
L1
data
L1
ins
L1
data
L1
ins
L1
data
L1
ins
L1
data
L2
L2
Main Memory

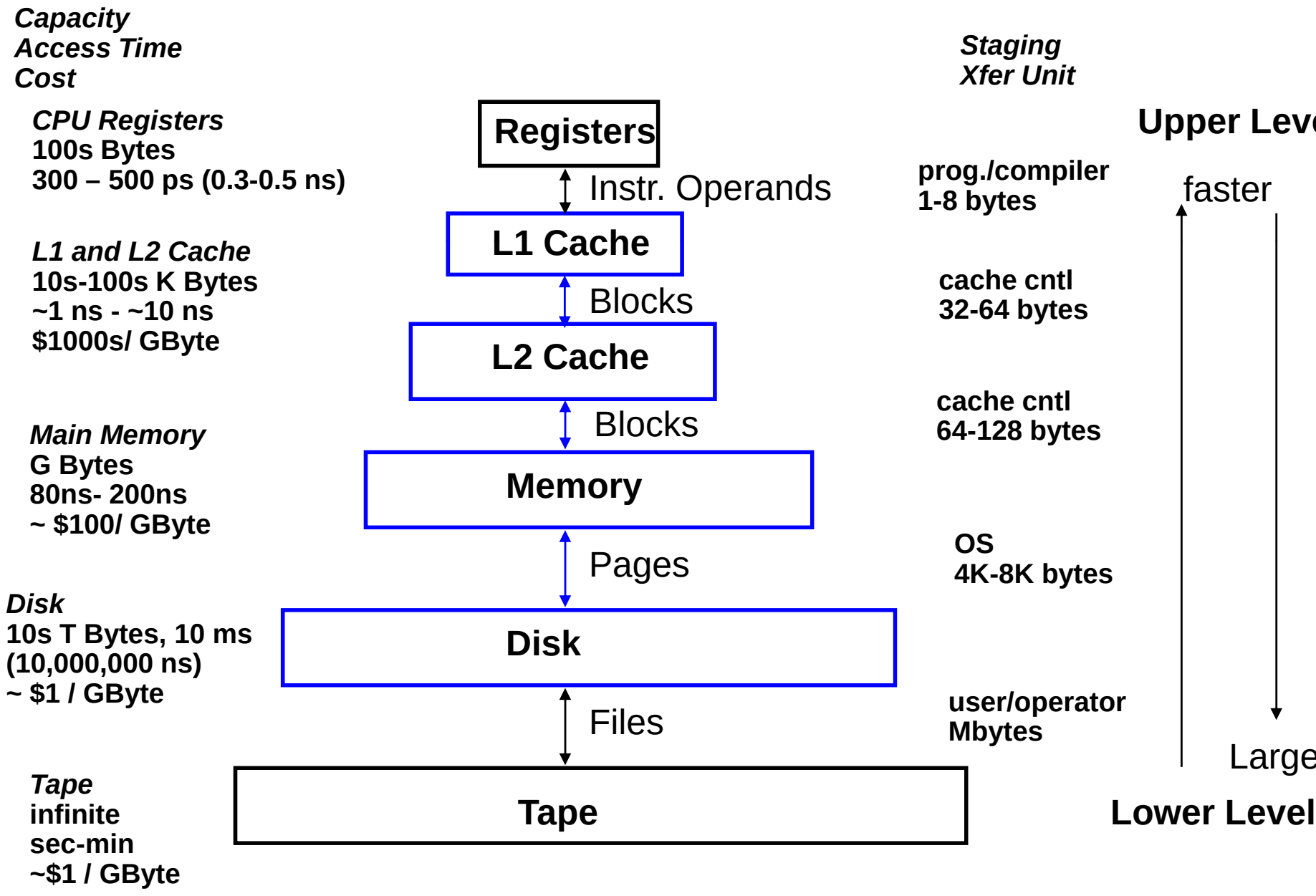

94V-0
MADE IN CHINA
NVIDIA
L7118760 0746A2
S TAIWAN
NF2043.00P
G92-270-A2
RoHS
COMPLIANT


© 2009 Xilinx, Incorporated
LT LINEAR
LTM4616V
µModule
SPARTAN-6
XC6SLX 16
SPARTAN
SP601 REV B1
XILINX

L1 Data Cache			
Size	Line Size	Latency	Associativty
32 KB	64 bytes	3 cycles	8-way
L1 Instruction Cache			
Size	Line Size	Latency	Associativty
32 KB	64 bytes	3 cycles	8-way
L2 Cache			
Size	Line Size	Latency	Associativty
6 MB	64 bytes	14 cycles	24-way

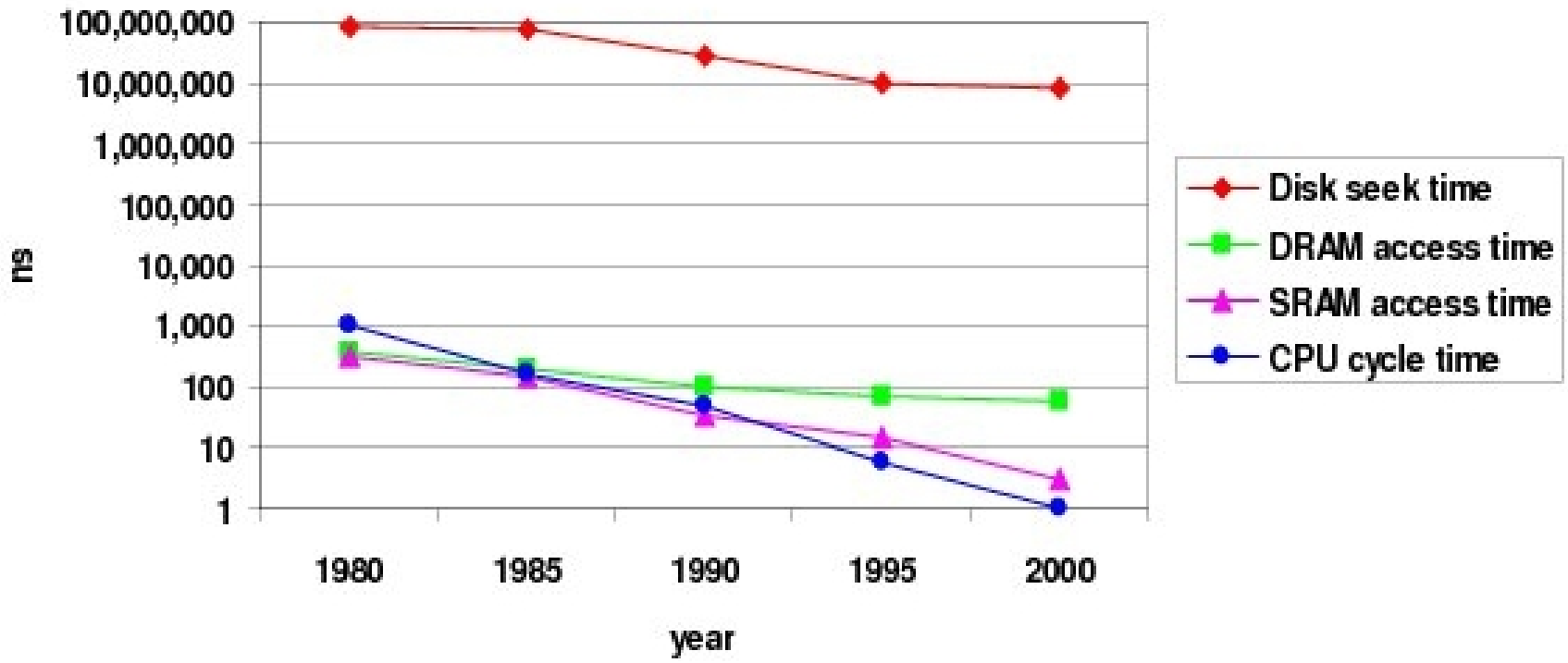
Typical cache specifications of a multicore in 2008.


cooling
towers
warehouse-scale
computer
power substation


Capacity
Access Time
Cost
CPU Registers
100s Bytes
300 – 500 ps (0.3-0.5 ns)
L1 and L2 Cache
10s-100s K Bytes
~1 ns - ~10 ns
$1000s/ GByte
Main Memory
G Bytes
80ns- 200ns
~ $100/ GByte
Disk
10s T Bytes, 10 ms
(10,000,000 ns)
~ $1 / GByte
Tape
infinite
sec-min
~$1 / GByte
Registers
Instr. Operands
L1 Cache
Blocks
L2 Cache
Blocks
Memory
Pages
Disk
Files
Tape
Staging
Xfer Unit
prog./compiler
1-8 bytes
cache cntl
32-64 bytes
cache cntl
64-128 bytes
OS
4K-8K bytes
user/operator
Mbytes
Upper Leve
faster
Large
Lower Level

The CPU-Memory Gap

The increasing gap between DRAM, disk, and CPU speeds.

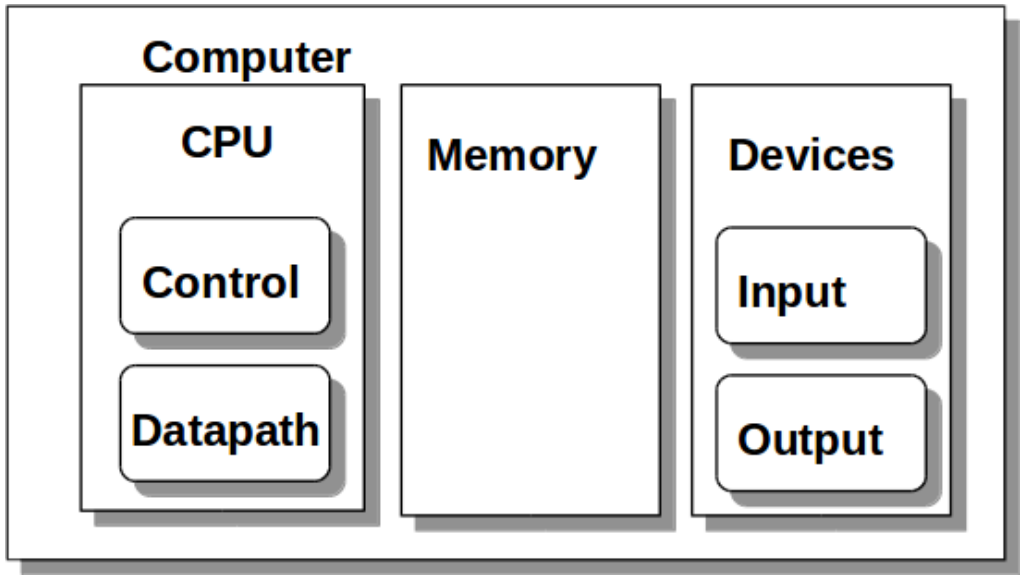


Once upon a time, every thing was slow in a computer . . .

Classes of Computers

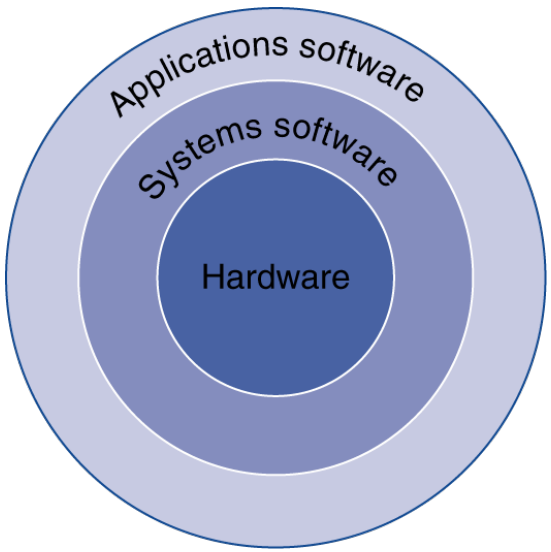
- **Personal computers**
  - General purpose, variety of software
  - Subject to cost/performance trade-off
- **Server computers**
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- **Supercomputers**
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market
- **Embedded computers**
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Components of a computer



- Same components for all kinds of computer
  - desktop, server, embedded

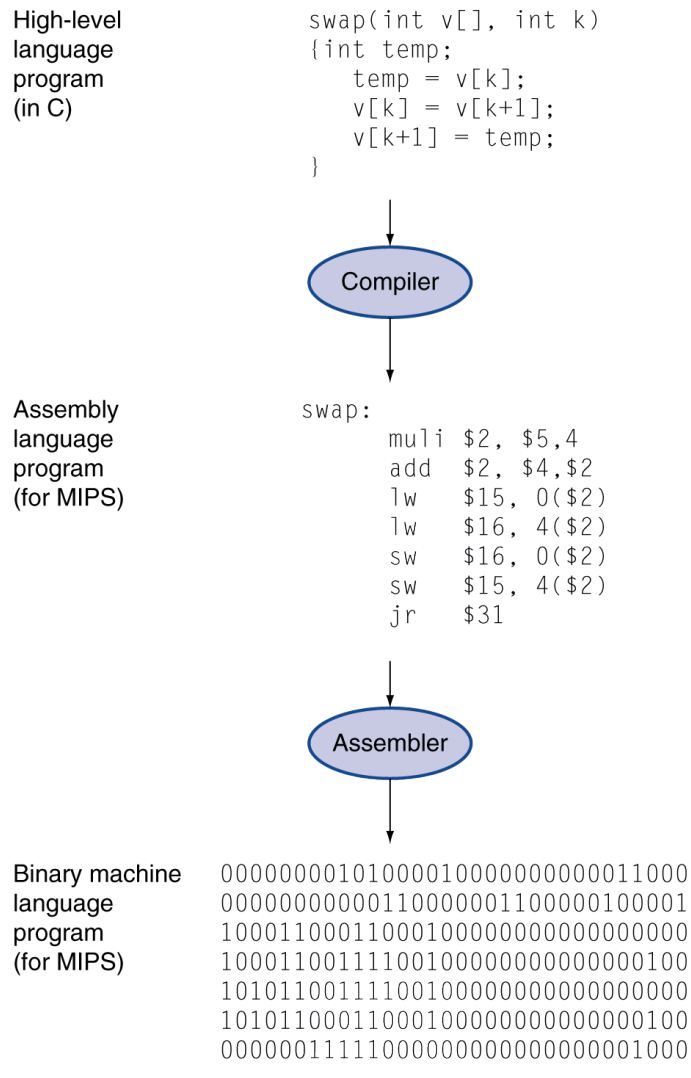
Below your program



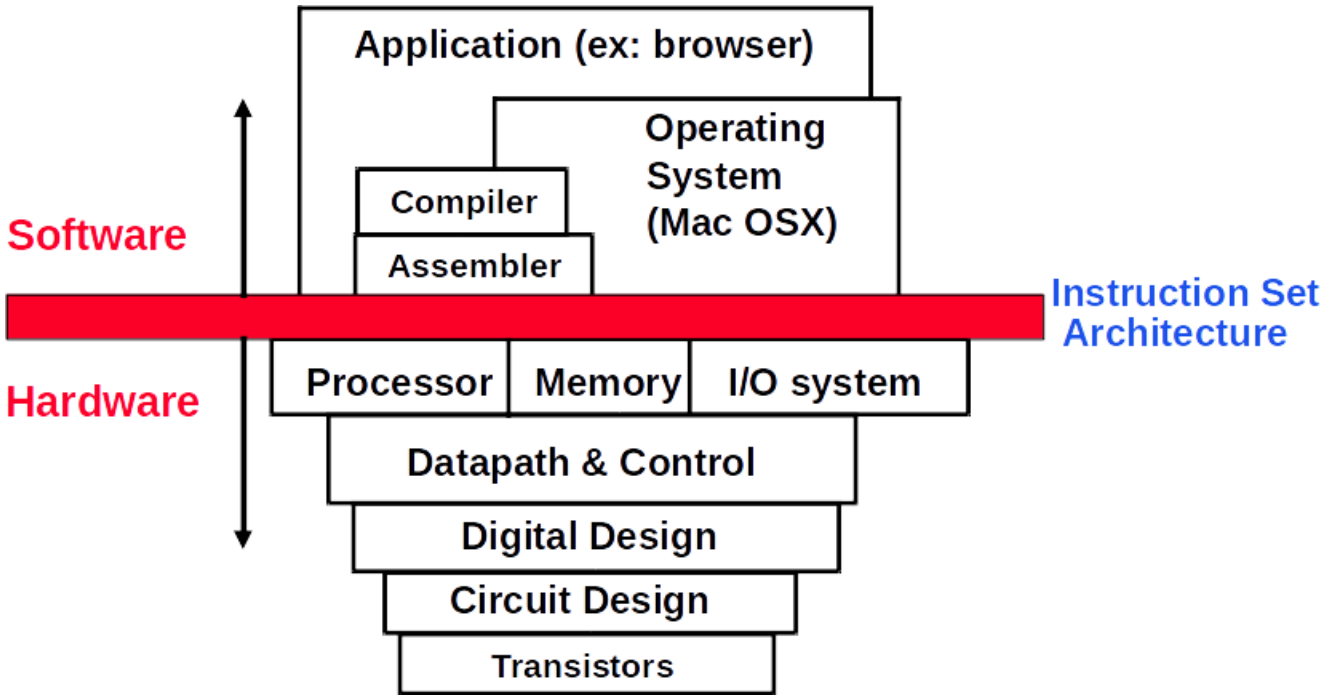
- **Application software**
  - Written in a high-level language
- **System software**
  - Compiler: translates HLL code to machine code
  - Operating system: service code
  - Handling input/output
  - Managing memory and storage
  - Scheduling tasks & sharing resources
- **Hardware**
  - Processor, memory,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

Levels of program code

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



Old-school machine structures (layers of abstraction)



New-school machine structures

Software

- Parallel Requests

  Assigned to computer
  e.g., Search “Katz”

- Parallel Threads

  Assigned to core
  e.g., Look-up, Ads

- Parallel Instructions

  >1 instruction @ one time
  e.g., 5 pipelined instructions

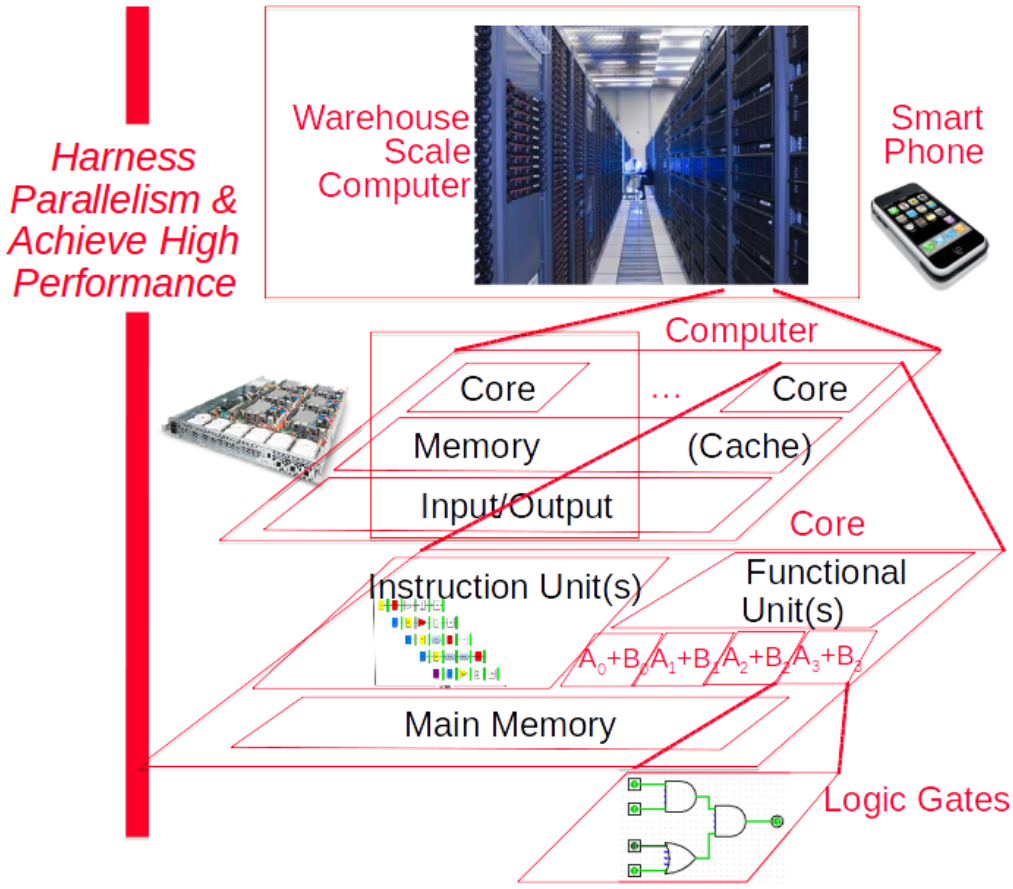
- Parallel Data

  >1 data item @ one time
  e.g., Add of 4 pairs of words

- Hardware descriptions

  All gates working in parallel at same time

Hardware

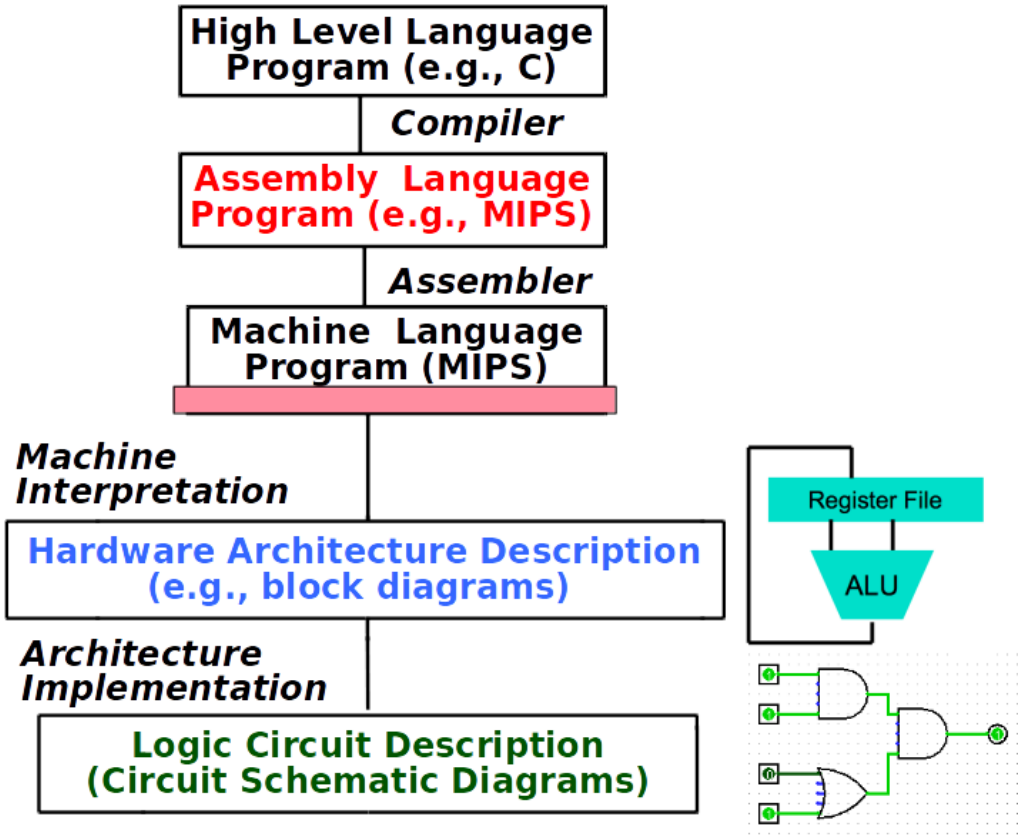


Why do computers become so complicated?

Pursuing performance!

- Eight great ideas
  - Use abstraction to simplify design
  - Design for Moore's Law
  - Make the common case fast
  - Performance via parallelism
  - Performance via pipelining
  - Performance via prediction
  - Hierarchy of memories
  - Dependability via redundancy

Great Idea #1: Abstraction




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw $t0, 0($2)
lw  $t1, 4($2)
sw  $t1, 0($2)
sw  $t0, 4($2)
```

```
# Anything can be
represented as a
# number, i.e., data or
instructions
```

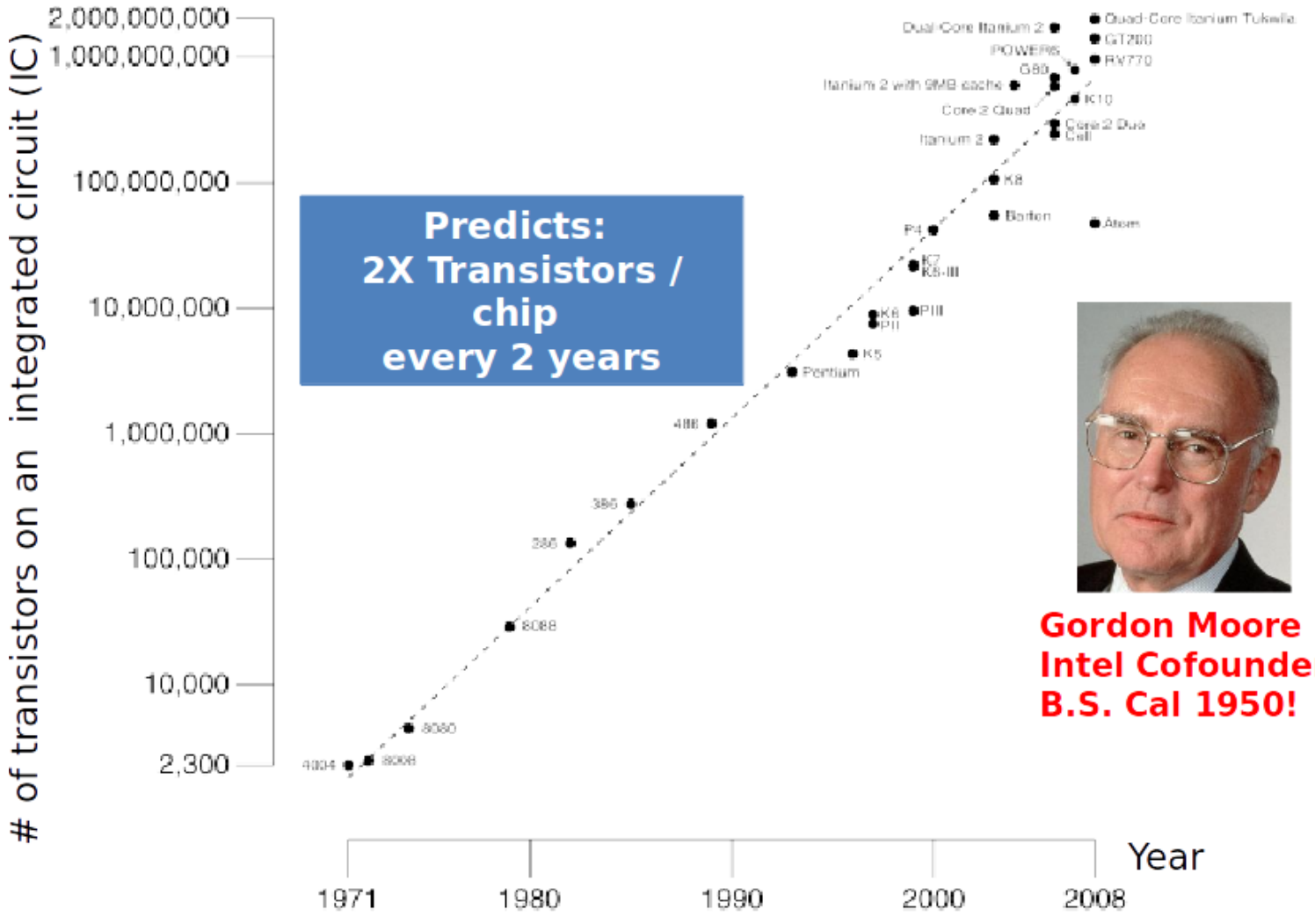
0000 1001 1100 0110 1010 1111

0101 1000

1010 1111 0101 1000 0000 1001

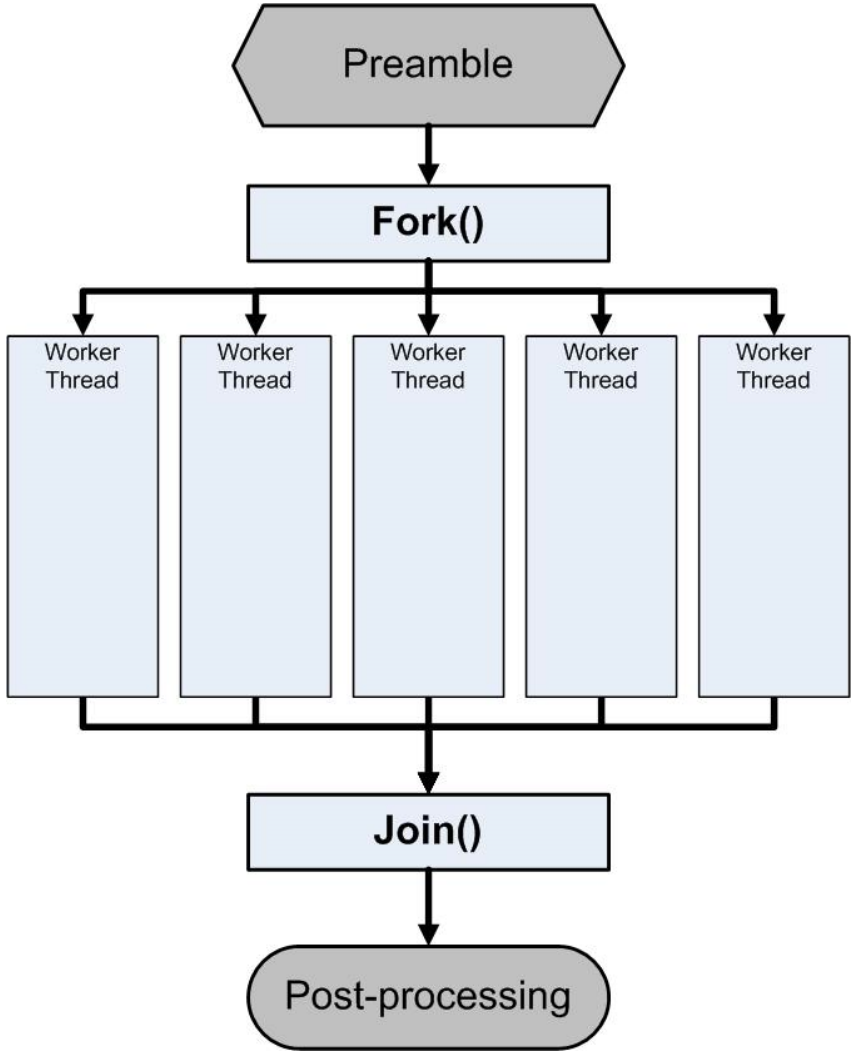
1100 0110

Great idea #2: Moore's Law

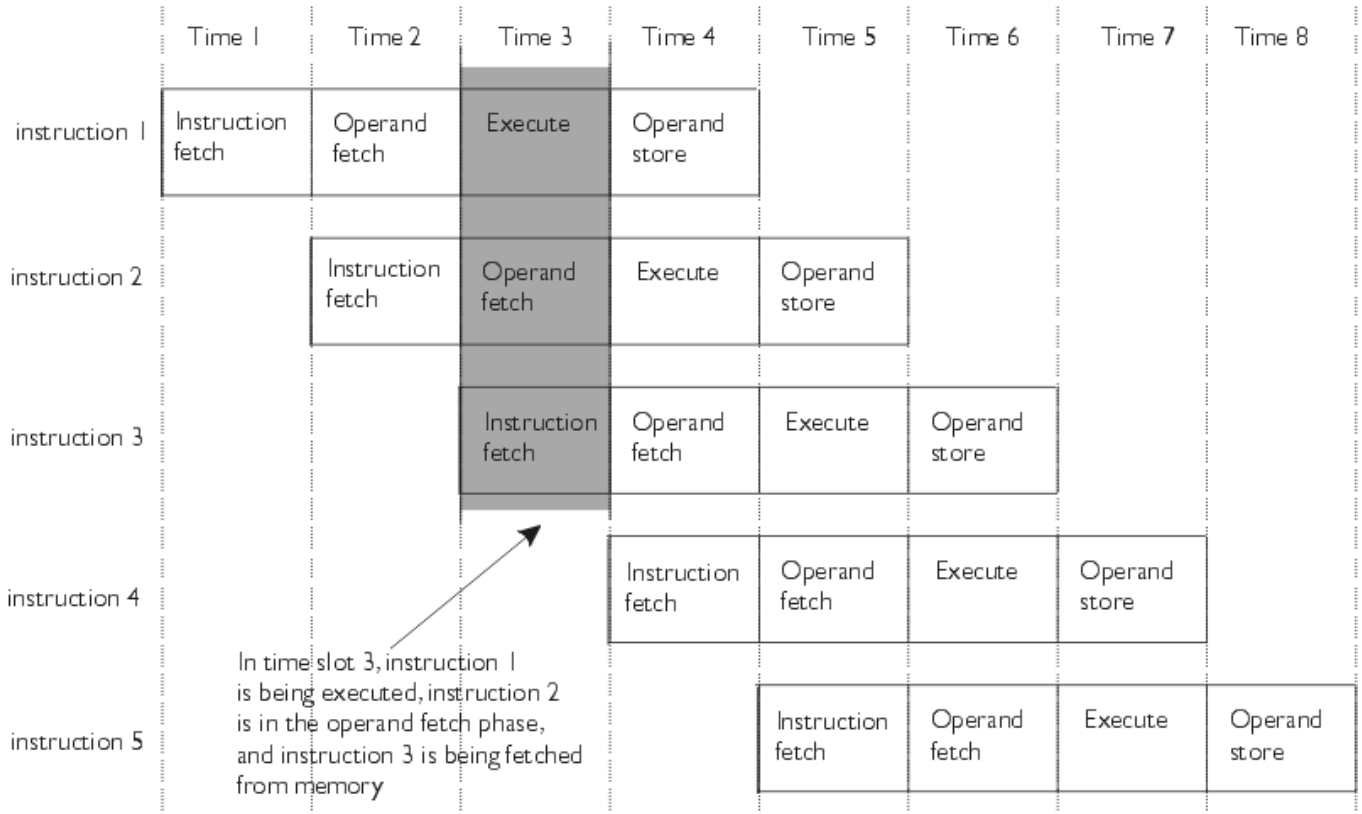


Gordon Moore
Intel Cofounder
B.S. Cal 1950!

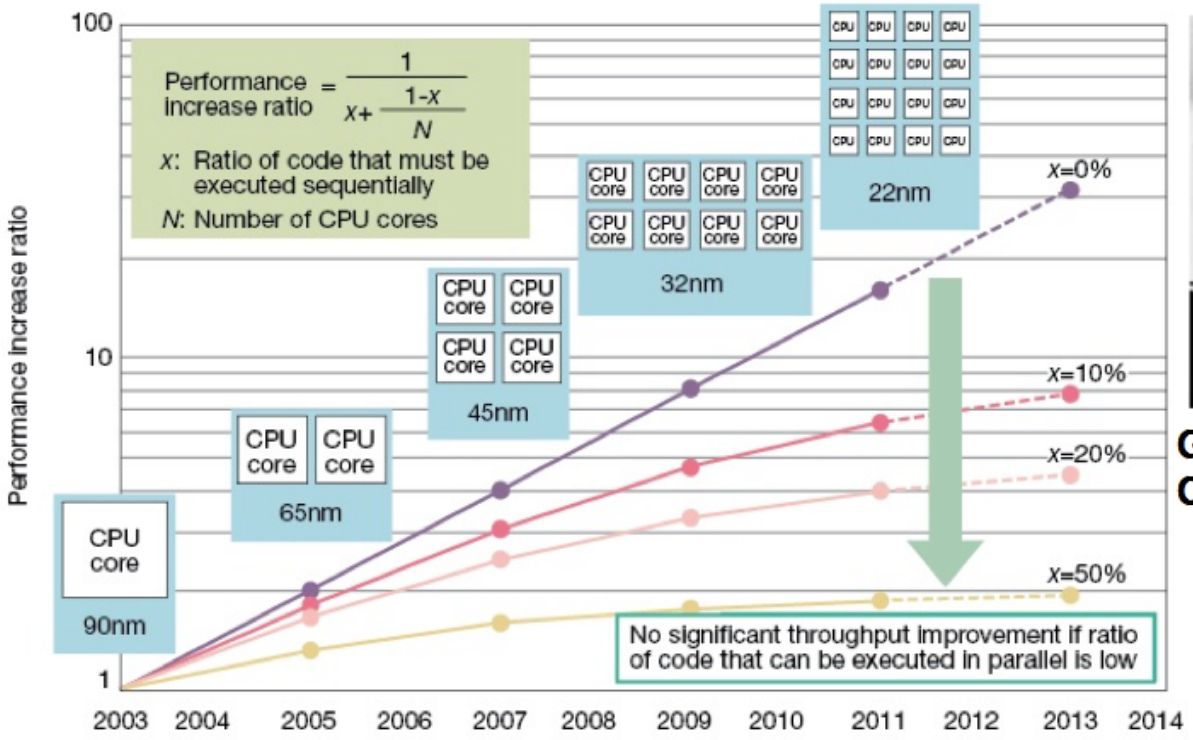
Great idea #4: Performance via parallelism



Great idea #5: Performance via pipelining



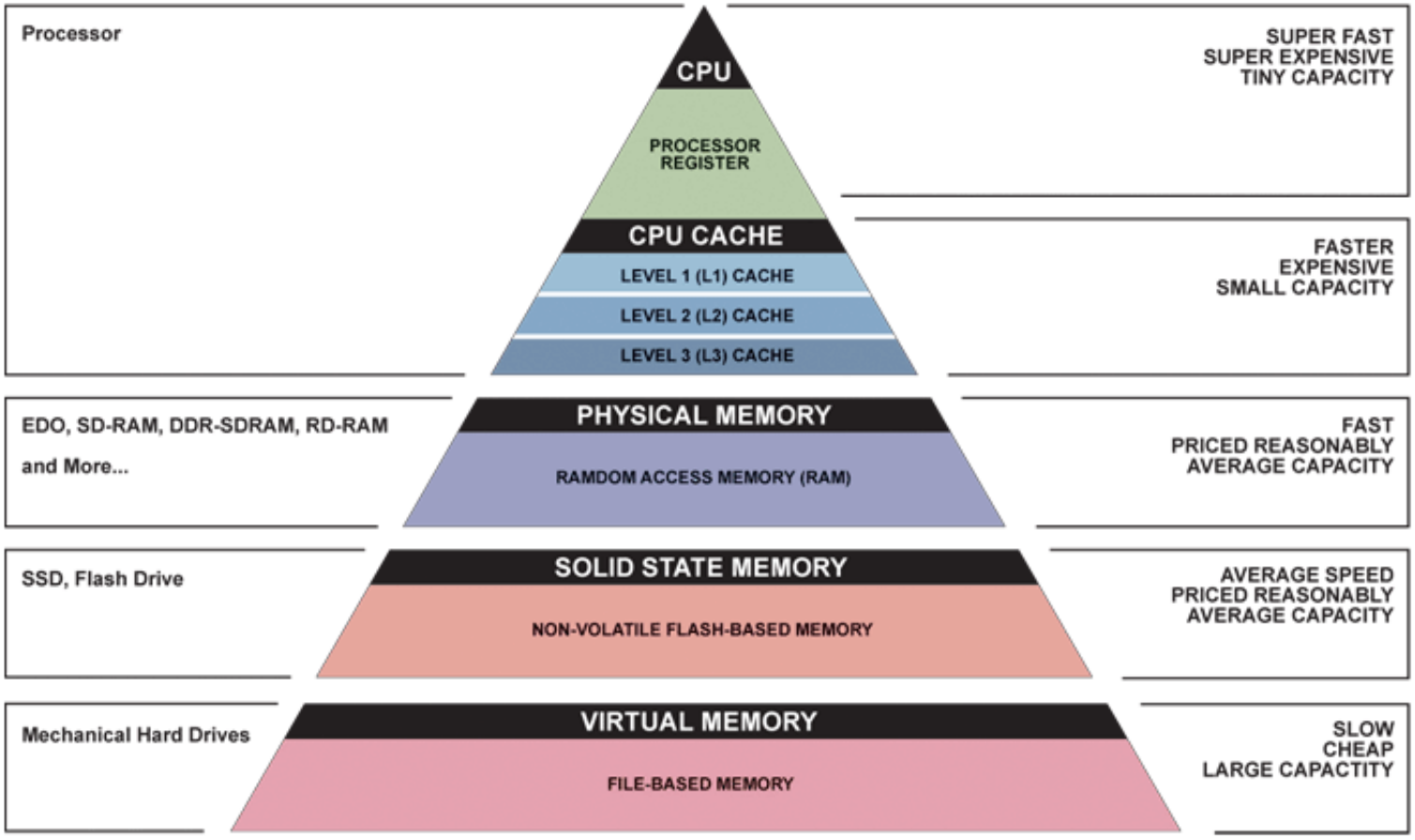
Caveat: Amdahl's Law



Gene Amdahl
Computer Pioneer

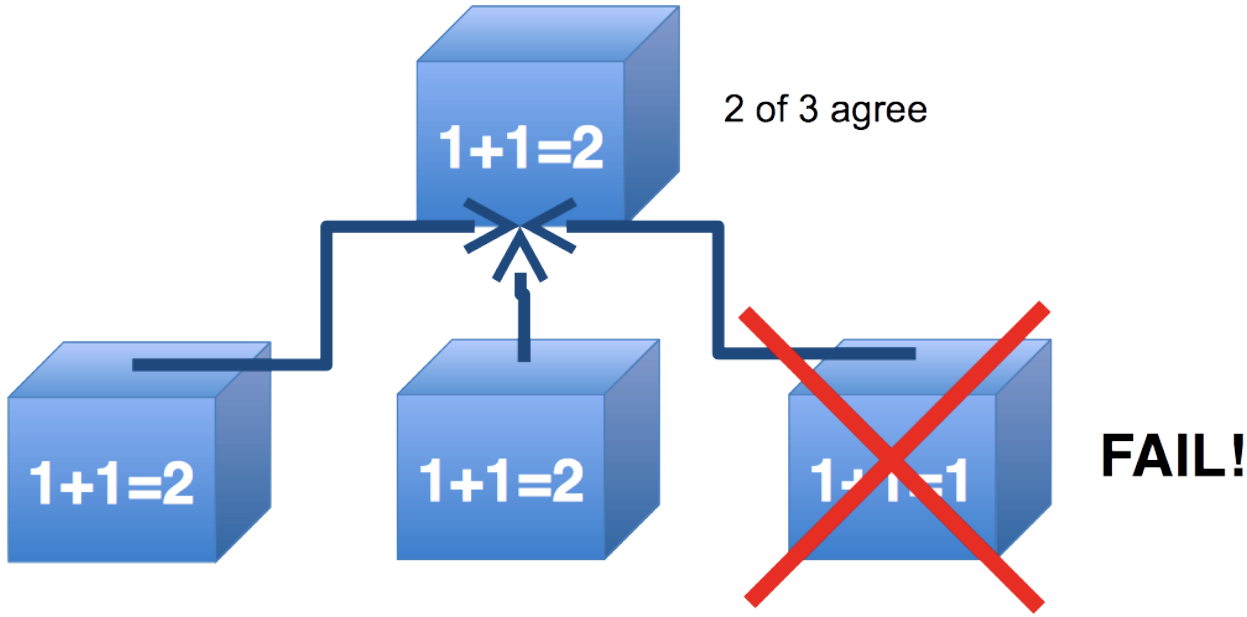
Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.

Great idea #7: Memory hierarchy (principle of locality)



Great Idea #8: Dependability via redundancy

- Redundancy so that a failing piece doesn't make the whole system fail



- Increasing transistor density reduces the cost of redundancy

Great Idea #8: Dependability via redundancy

Applies to everything from data centers to storage to memory to instructors

- Redundant **data centers** so that can lose 1 datacenter but Internet service stays online
- Redundant **disks** so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
- Redundant **memory bits** of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)

Understanding 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

What you will learn

- How programs are translated into the machine language, and
- how the hardware executes them
- The hardware/software interface
- What determines program performance, and
- how it can be improved
- How hardware designers improve performance
- What is parallel processing

Course Topics

1. Introduction
   - Machine structures: layers of abstraction
   - Eight great ideas
2. Performance Metrics I
   - CPU performance
   - perf, a profiling tool
3. Memory Hierarchy
   - The principle of locality
   - DRAM and cache
   - Cache misses
   - Performance metrics II: memory performance and profiling
   - Cache design and cache mapping techniques
4. MIPS Instruction Set Architecture (ISA)
   - MIPS number representation
   - MIPS instruction format, addressing modes and procedures
   - SPIM assembler and simulator

Course Topics (cont'd)

5. Introduction to Logic Circuit Design
   - Switches and transistors
   - State circuits
   - Combinational logic circuits
   - Combinational logic blocks
   - MIPS single cycle and multiple cycle CPU data-path and control
6. Instruction Level Parallelism
   - Pipelining the MIPS ISA
   - Pipelining hazards and solutions
   - Multiple issue processors
   - Loop unrolling, SSE
7. Multicore Architecture
   - Multicore organization
   - Memory consistency and cache coherence
   - Thread level parallelism
8. GPU Architecture
   - Memory model
   - Execution model: scheduling and synchronization

Student evaluation

- Four assignments, each worth 10% of the final mark
  - Assignment 1 (CPU performance and memory hierarchy), due Friday, Jan. 27
  - Assignment 2 (MIPS and logic circuits), due Friday, Feb. 17
  - Assignment 3 (Circuits and data-path), due Friday, March 10,
  - Assignment 4 (ILP and multicore), due Friday, March 31.
- Four quizzes (key concepts, 30-minute in class), each worth 5% of the final mark
  - Quiz 1 (CPU/memory performance metrics and hierarchical memory), Thursday, Jan. 26
  - Quiz 2 (MIPS), Thursday, Feb. 16
  - Quiz 3 (Circuits and data-path), Thursday, March 9
  - Quiz 4 (ILP and multicore), Thursday, March 30
- One final exam (covering all the course contents), worth 40% of the final mark

Recommended (but not required) textbook

Patterson & Hennessy (2011), "Computer Organization and Design: The Hardware/Software Interface“, revised 4th edition or 5th edition. ISBN: 978-0-12-374750-1

Instructor: Marc Moreno Maza

- Email: `moreno@csd.uwo.ca`
- Office room: MC327
- Office hours:

  Tuesdays 2:30pm - 3:15pm
  Thursdays 1:30pm - 3:15pm
  Otherwise by appointment

Teaching Assistants: Davood Mohajerani and Egor Chesakov

- Emails: `dmohajer@uwo.ca` and `echesako@uwo.ca`
- Office room: MC327
- Office hours:

  Mondays 1:30pm - 3:30pm
  Wednesdays 1:30pm - 3:30pm

Acknowledgements

The lecturing slides of this course are adapted from the slides accompanied with the text book and the teaching materials posted on the `www` by other instructors who are teaching Computer Architecture courses.