

CS 110

Computer Architecture

Summary

Instructors:

Sören Schwertfeger & Chundong Wang

<https://robotics.shanghaitech.edu.cn/courses/ca/22s/>

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkley's CS61C

Final Exam

- Time (90 minutes + 10 minutes)
 - 9:00am – 10:40am, 18th June
 - Any submission past **10:40:00** would not be evaluated
- **Twelve groups**, not 9 for mid-term exam
 - To be announced with 12 Tencent Meeting rooms
 - Set your name in the Meeting Room as **StudentID-Name**
- Check the email on final exams received on 1st June
 - Dual devices (双设备)
 - Do not be late for the exam
- **NO cheating, NO plagiarism**

Final Exam

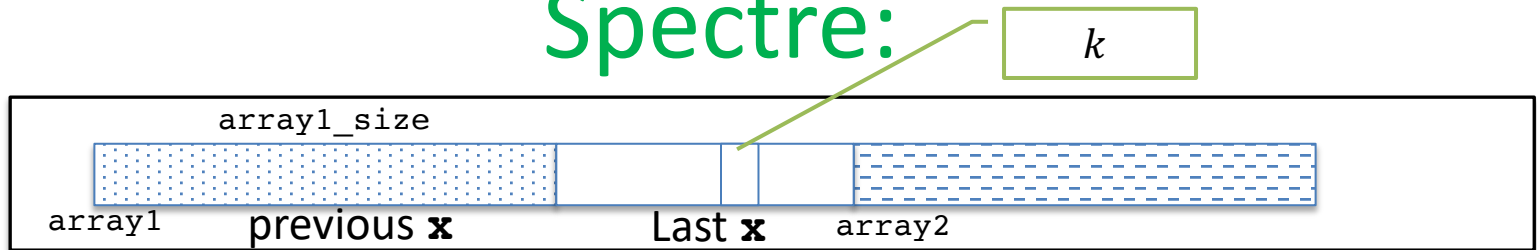
- Open-book
- Testing for final exam
 - Assitant inviligator would schedule, and pls attend Tencent Meetings
- QQ groups for online annonucement and clarification
 - Assitant inviligators create and pls join corresponding one
 - Inviligators may broadcast messages during the exam
 - If you got any question, do a private chat with the assitant inviligator
- One file to be submitted in the final exam
 - Three channels
 - Gradescope (preferred), ShanghaiTech Pan, email
 - Name your file in a format of Group x -StudentID-NamePinyin.pdf or Group x -StudentID-NamePinyin.zip
 - e.g., Group3-2020999999-ZhangSan3.pdf
- NO printing of exam papers
 - Handwrite your answers on blank sheet

Meltdown & Spectre

- Meltdown
 - Out of order execution

- Spectre
 - Speculative execution

Spectre:



- **Speculative execution**

- Example: branch prediction
- Covered in L13

Prerequisites:

- i. `array1[x]`, with an out-of-bound `x` larger than `array1_size`, resolves to a secret byte `k` that is cached;
- ii. `array1_size` and `array2` uncached.
- iii. Previous `x` values have been valid.

// `x` is controlled by attacker.

```
1. if (x < array1_size) ← cache miss, so run next line due to prediction history
2.  y = array2[array1[x] * 4096] ← array1[x] cache hit, as k is cached,
                                so load array2[k * 4096]
```

Regarding a misprediction with an illegal `x`, `array2[k * 4096]` will not be used, but has been loaded into CPU cache.

We can use Flush+Reload to guess `k` with `array2`.

The aim of Spectre:
to read out a victim's sensitive
information

The Impact of Spectre

- Processors can be tricked in speculative execution to modify cache state
 - Leaving attackers an exploitable opportunity
- Sensitive information of a victim program may be leaked
- Speculative Store Bypass
 - A newer variant of Spectre (v4) could allow an attacker to retrieve *older but stale values* in a CPU's stack or other memory locations.
 - <https://software.intel.com/security-software-guidance/software-guidance/speculative-store-bypass>

Meltdown and Spectre

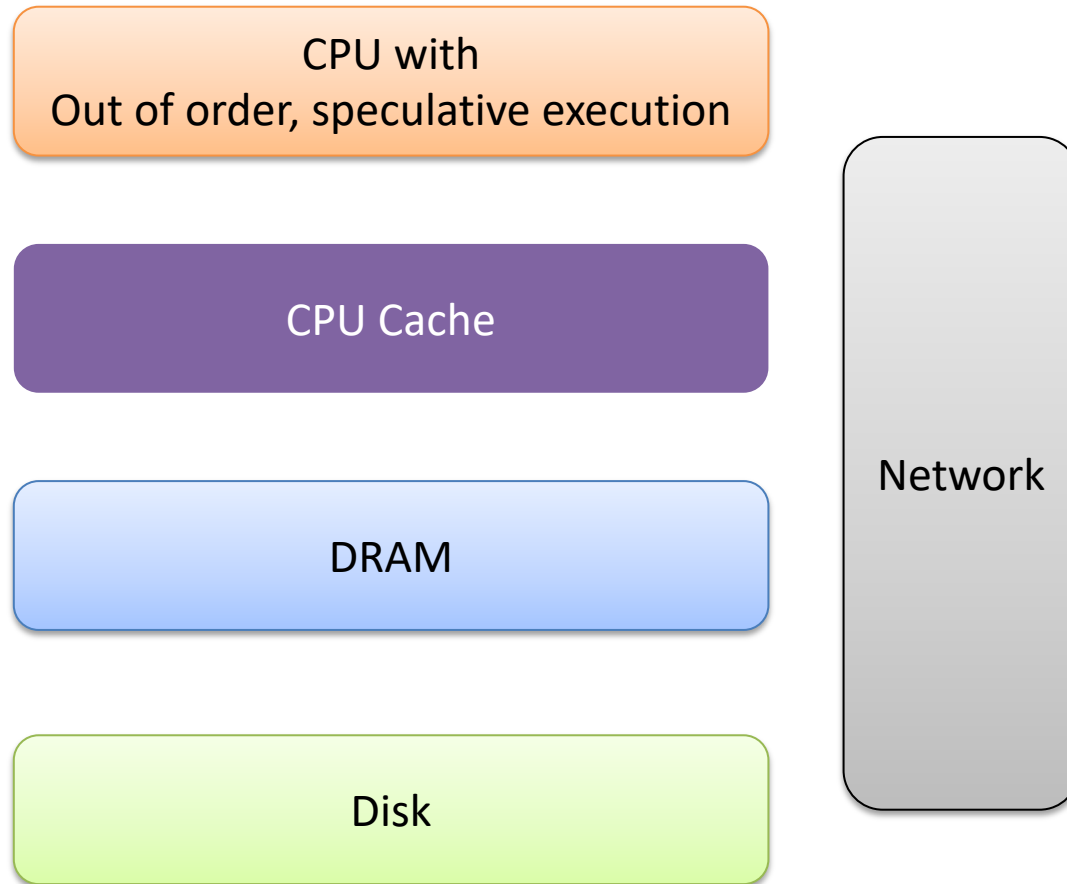
- More complicated than examples here
- Multiple variants today
- Many processors, OSes, applications affected
 - PC, mobile devices, cloud
- Many proposals to mitigate their impacts

No announced RISC-V silicon is susceptible, and the popular open-source RISC-V Rocket processor is unaffected as it does not perform memory accesses speculatively. <https://riscv.org/2018/01/more-secure-world-risc-v-isa/>

However, there is a workshop paper “*Replicating and Mitigating Spectre Attacks on a Open-Source RISC-V Microarchitecture*”

https://carrv.github.io/2019/papers/carrv2019_paper_5.pdf

Vulnerable Architecture



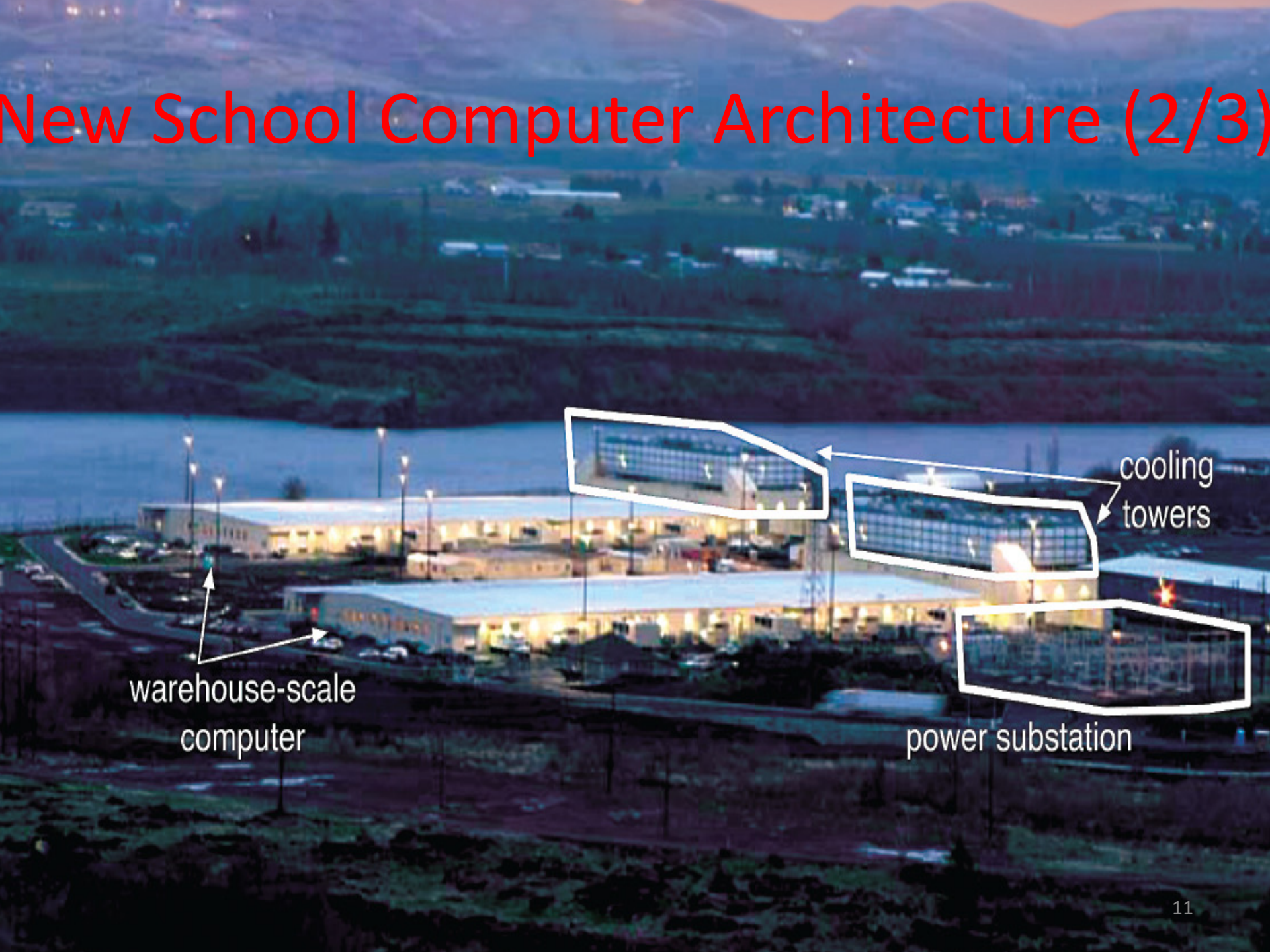
Let us review CA now.

New School Computer Architecture (1/3)



Personal
Mobile
Devices

New School Computer Architecture (2/3)



warehouse-scale
computer

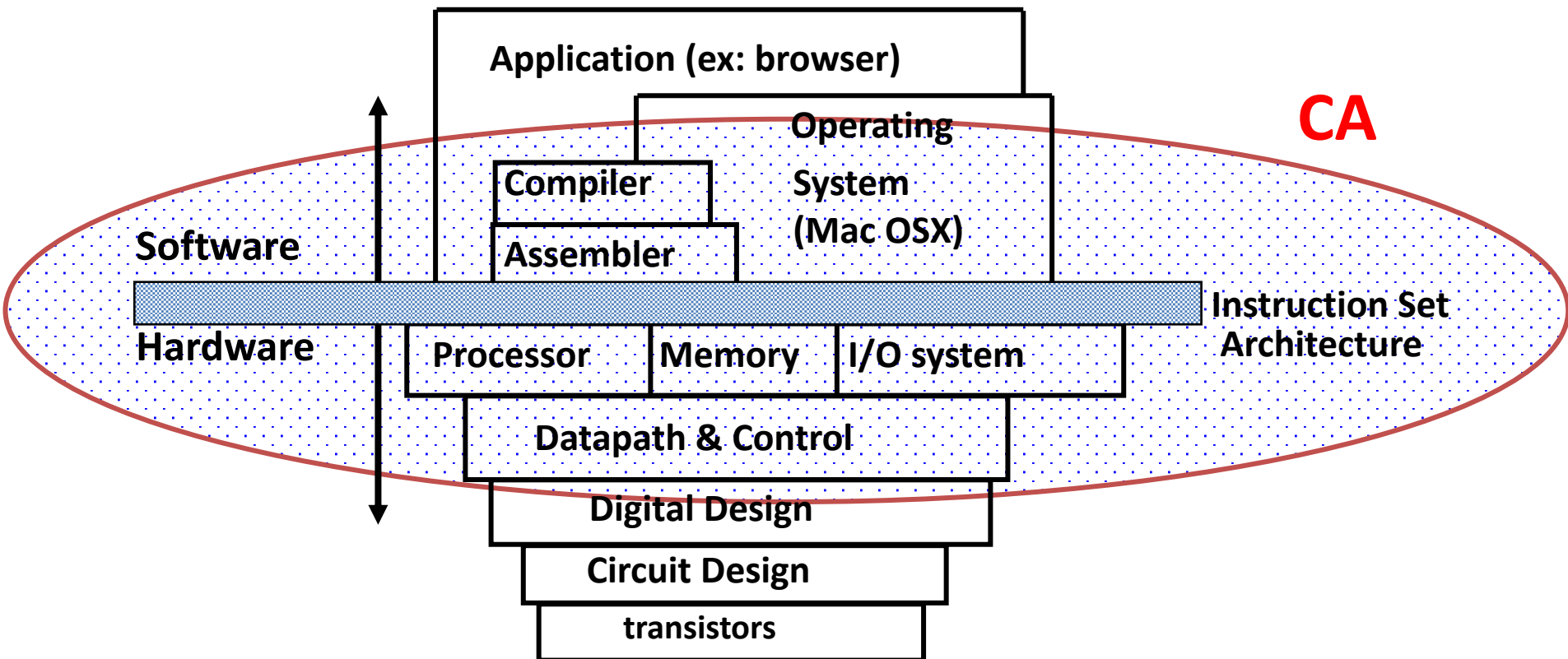
cooling
towers

power substation

New School Computer Architecture (3/3)



Old Machine Structures



New-School Machine Structures (It's a bit more complicated!)

Software

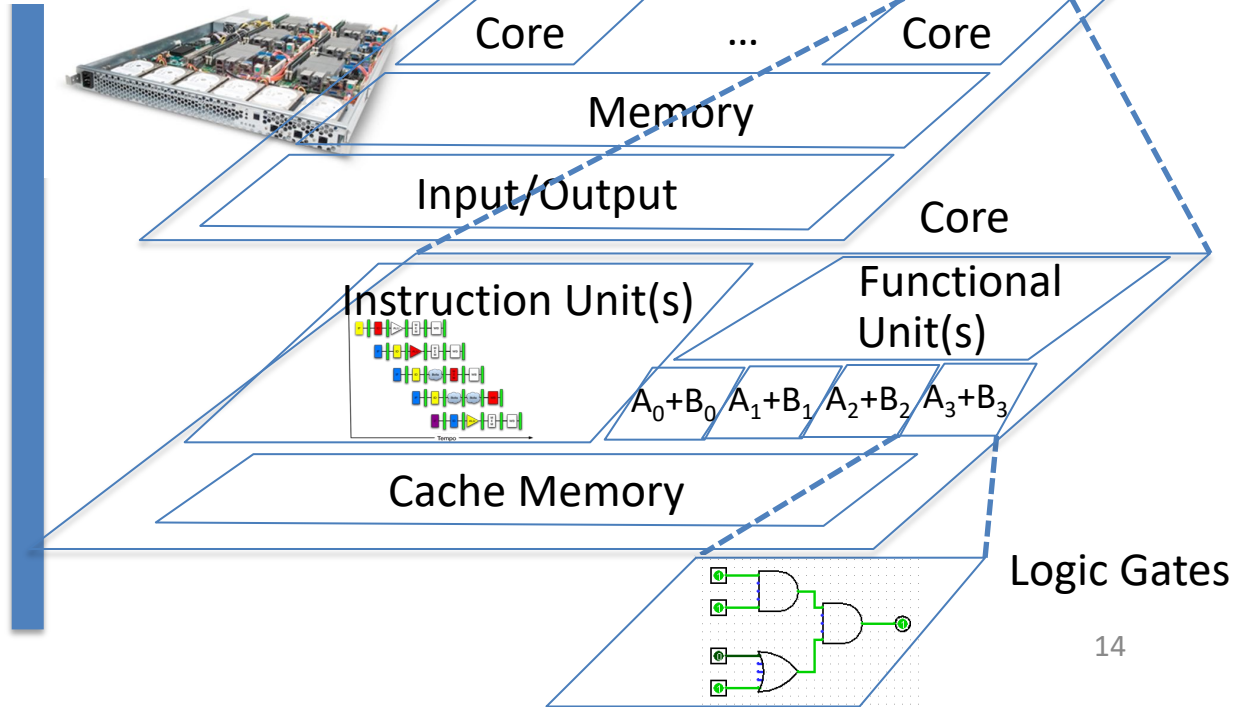
Hardware

Warehouse
Scale
Computer

Smart
Phone



*Leverage
Parallelism &
Achieve High
Performance*



- **Parallel Requests**
Assigned to computer
e.g., Search "Avatar 2"
- **Parallel Threads**
Assigned to core
e.g., Lookup, 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 functioning in parallel at same time
- **Programming Languages**

Great Ideas in Computer Architecture

1. Design for Moore's Law
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
5. Memory Hierarchy
6. Performance via
Parallelism/Pipelining/Prediction

Powers of Ten inspired CA Overview

Going Top-Down cover 3 Views

1. Architecture (when possible)
 2. Physical Implementation of that architecture
 3. Programming system for that architecture and implementation (when possible)
- See <http://www.powersof10.com/film>

Earth

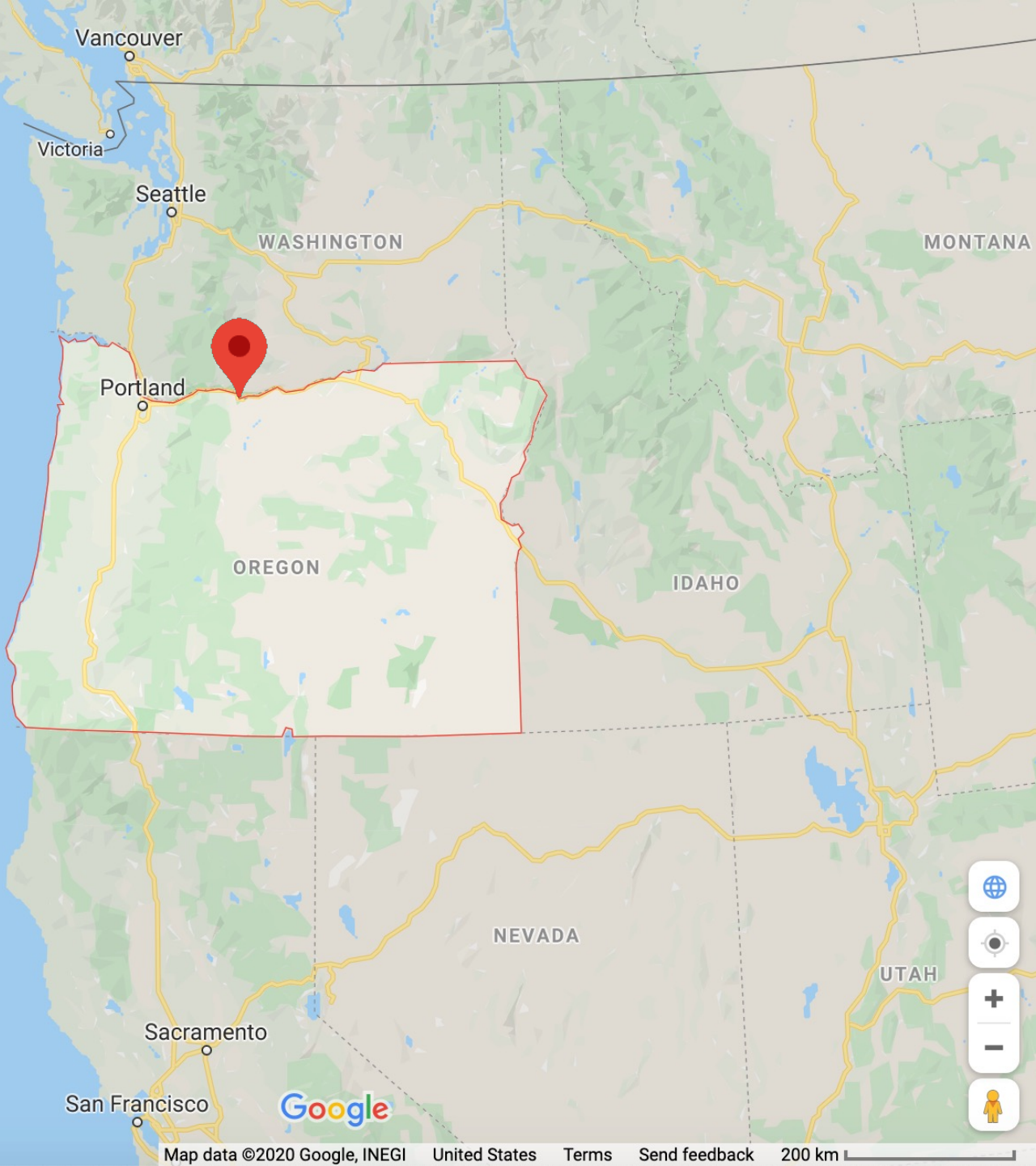
10^7 meters

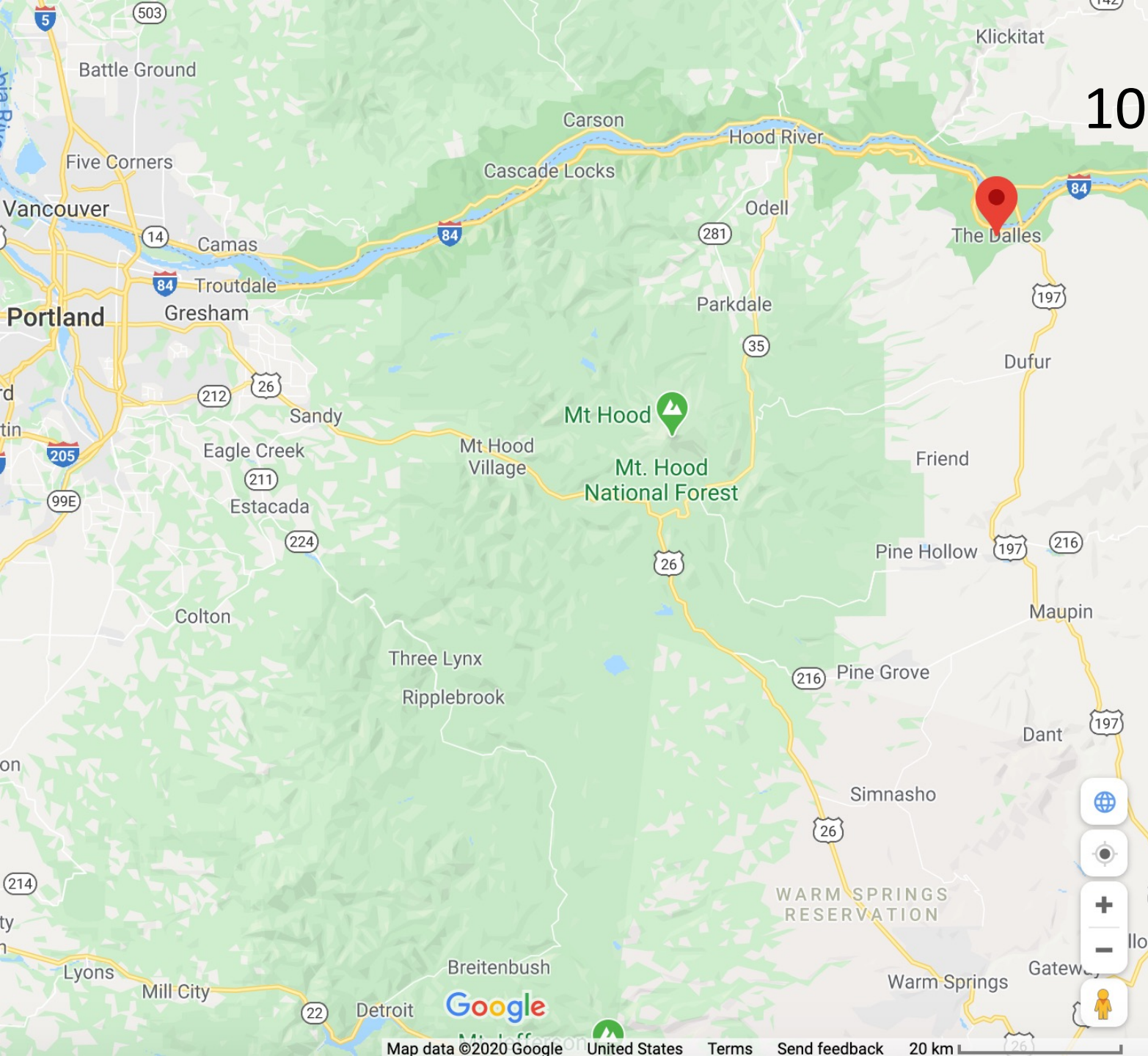


10⁷ meters



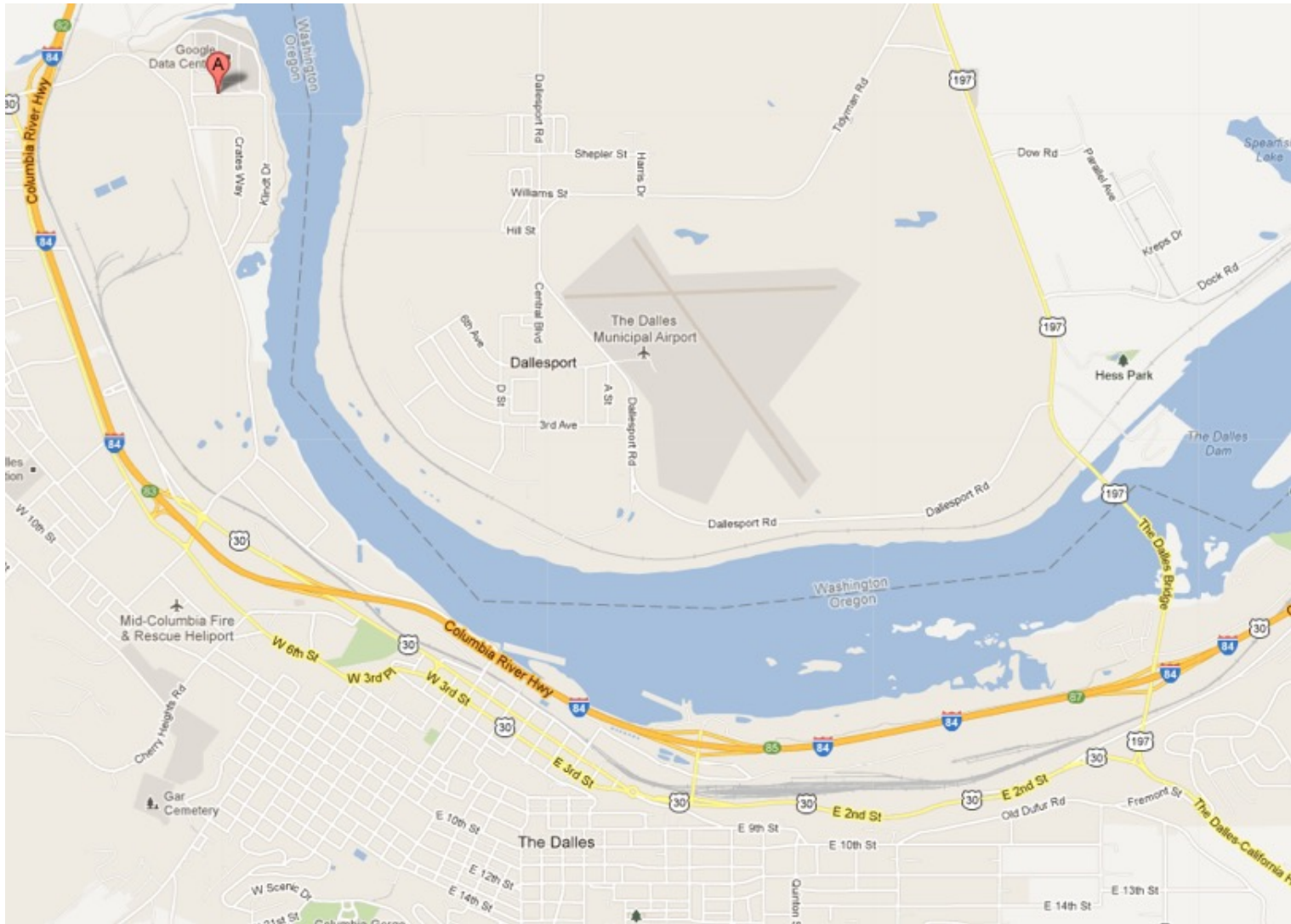
10^6 meters





10⁵ meters

The Dalles, Oregon 10^4 meters



The Dalles, Oregon 10^4 meters



Google's Oregon WSC 10^3 meters



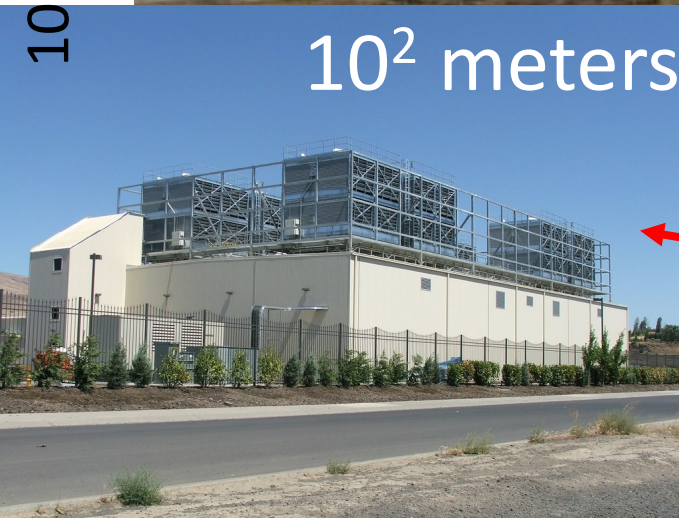
10^4 meters

Google's Oregon WSC

10 kilometers



10^2 meters



10^3 meters



Google Warehouse

- 90 meters by 75 meters, 10 Megawatts
- Contains 40,000 servers, 190,000 disks
- Power Utilization Effectiveness: 1.23
 - 85% of 0.23 overhead goes to cooling losses
 - 15% of 0.23 overhead goes to power losses
- Contains 45, 40-foot long containers
 - 8 feet × 9.5 feet × 40 feet
- 30 stacked as double layer, 15 as single layer

Containers in WSCs

10² meters



100 meters

Google Container

10¹ meters

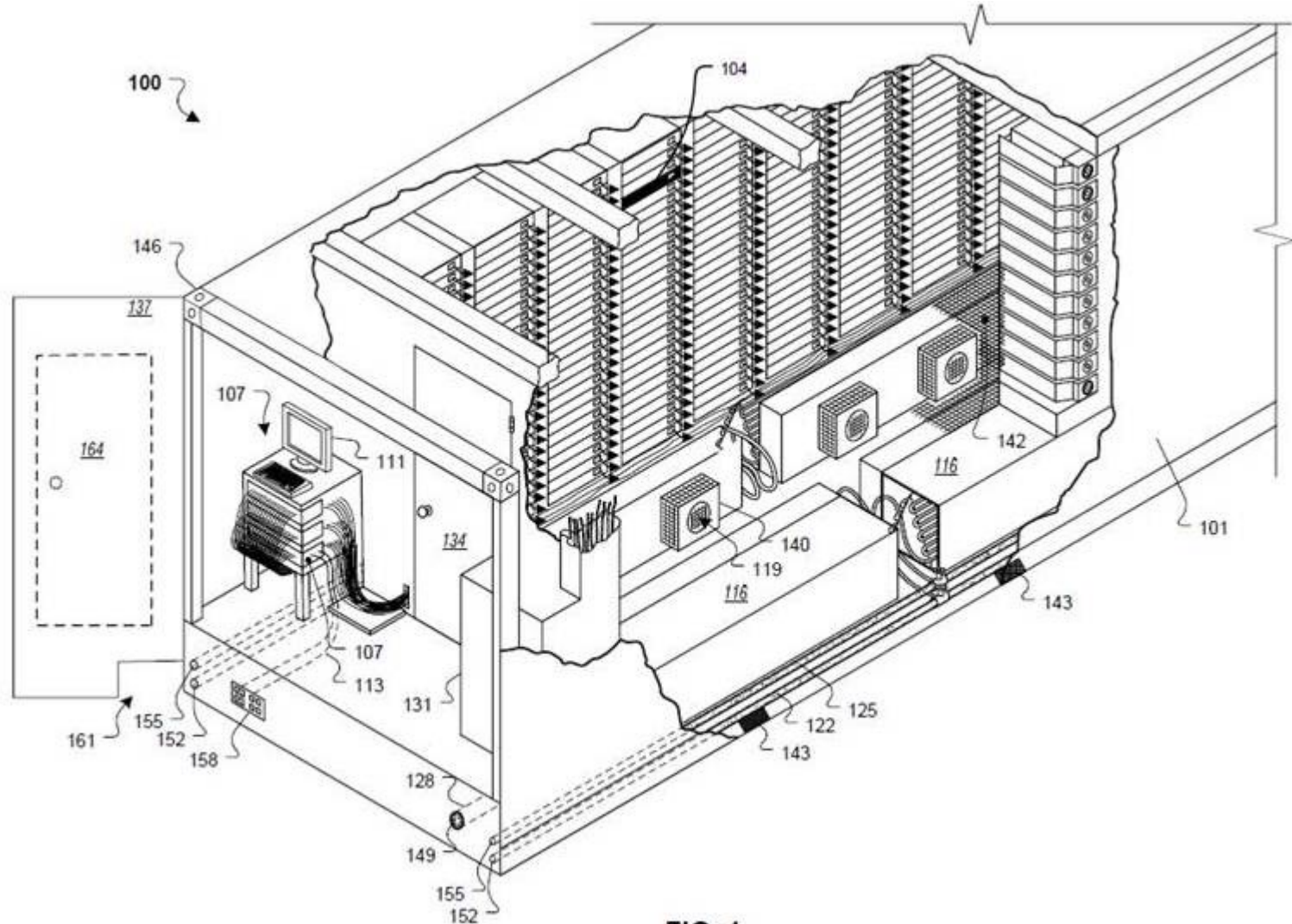
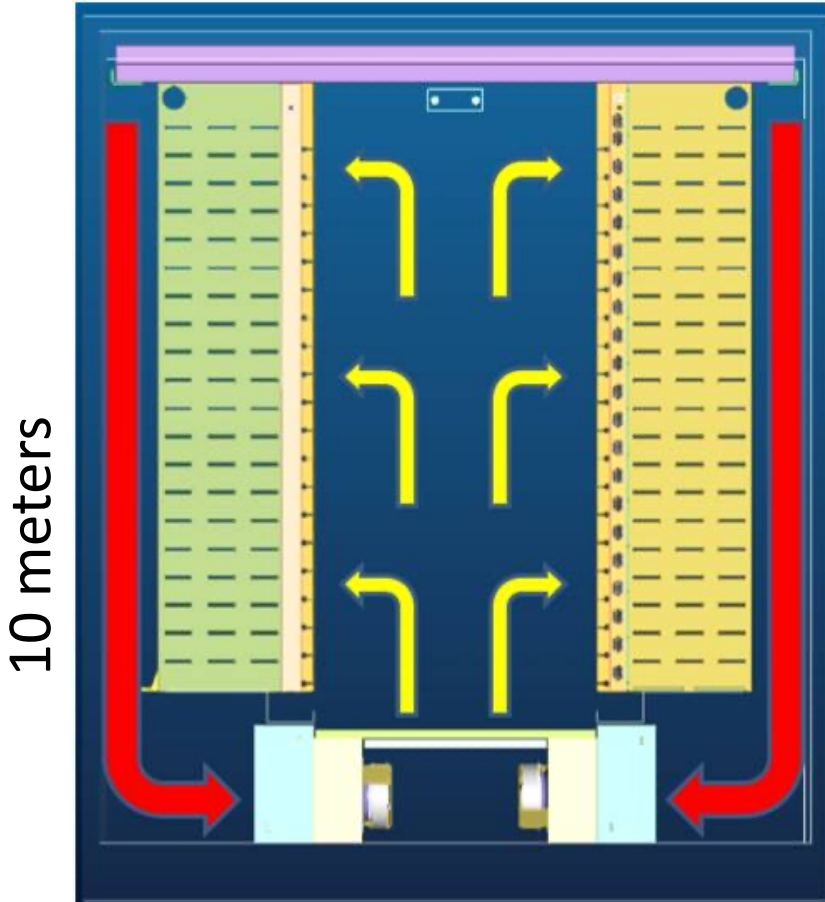


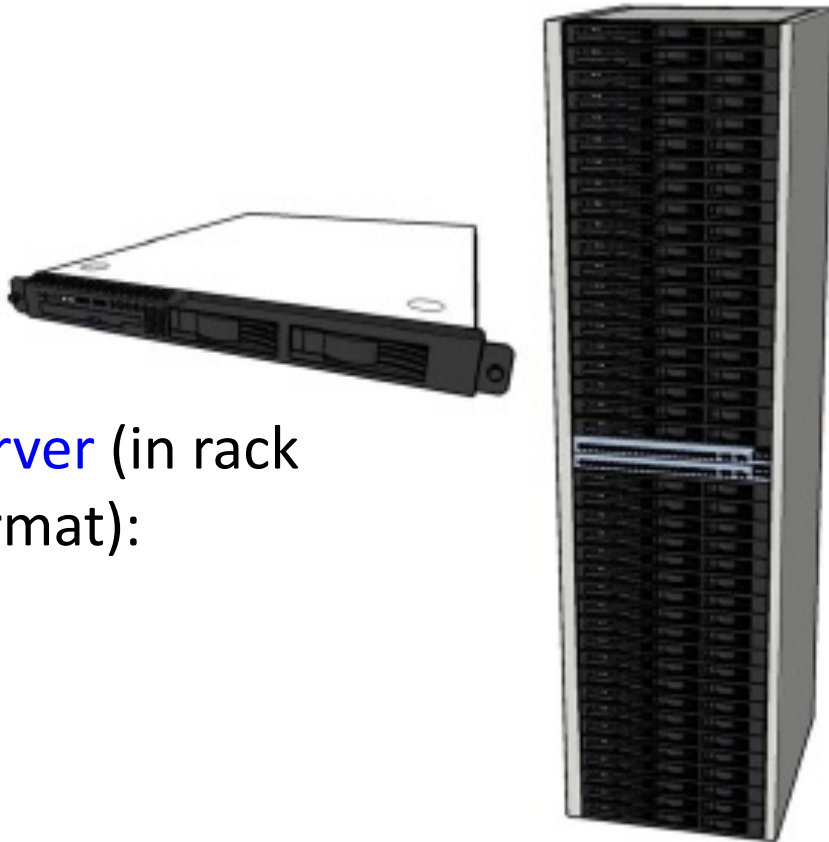
FIG. 1

Google Container 10^0 meters



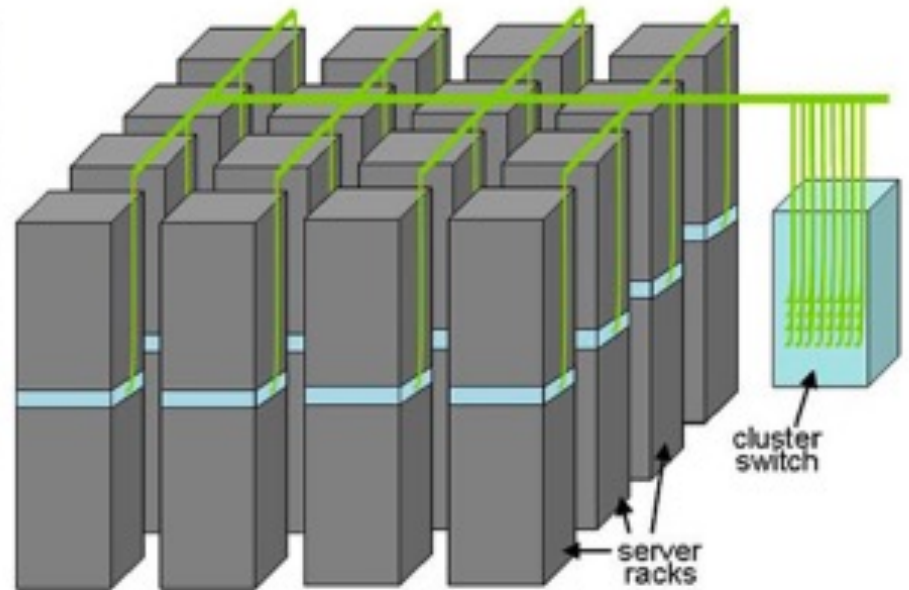
- 2 long rows, each with 29 racks
- Cooling below raised floor
- Hot air returned behind racks

Equipment Inside a Container



Server (in rack format):

7 foot Rack: servers + Ethernet local area network switch in middle (“rack switch”)



Array (aka cluster):
server racks + larger local area network switch (“array switch”) 10X faster => cost 100X: cost $f(N^2)$

Great Ideas in Computer Architecture

1. *Design for Moore's Law*
 - *WSC, Container, Rack*
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. *Dependability via Redundancy*
 - *Multiple WSCs, Multiple Racks, Multiple Switches*
5. Memory Hierarchy
6. *Performance via Parallelism/Pipelining/Prediction*
 - *Task level Parallelism, Data Level Parallelism*

Google Server Internals 10^{-1} meters

10 centimeters



Facebook Datacenter

facebook



Software: Often uses MapReduce

- Simple data-parallel ***programming model*** and ***implementation*** for processing large datasets
- Users specify the computation in terms of
 - a ***map*** function, and
 - a ***reduce*** function
- Underlying runtime system
 - Automatically ***parallelize*** the computation across large scale clusters of machines
 - ***Handles*** machine ***failure***
 - ***Schedule*** inter-machine communication to make efficient use of the networks

Programming Multicore Microprocessor: OpenMP

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000;
int value[num_steps];
int reduce()
{
    int i;
    int sum = 0;
    #pragma omp parallel for private(x) reduction(+:sum)
    for (i=1; i<= num_steps; i++){
        sum = sum + value[i];
    }
}
```

Great Ideas in Computer Architecture

1. *Design for Moore's Law*
 - *More transistors = Multicore + SIMD*
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
5. *Memory Hierarchy*
 - *More transistors = Cache Memories*
6. *Performance via Parallelism/Pipelining/Prediction*
 - *Thread-level Parallelism*

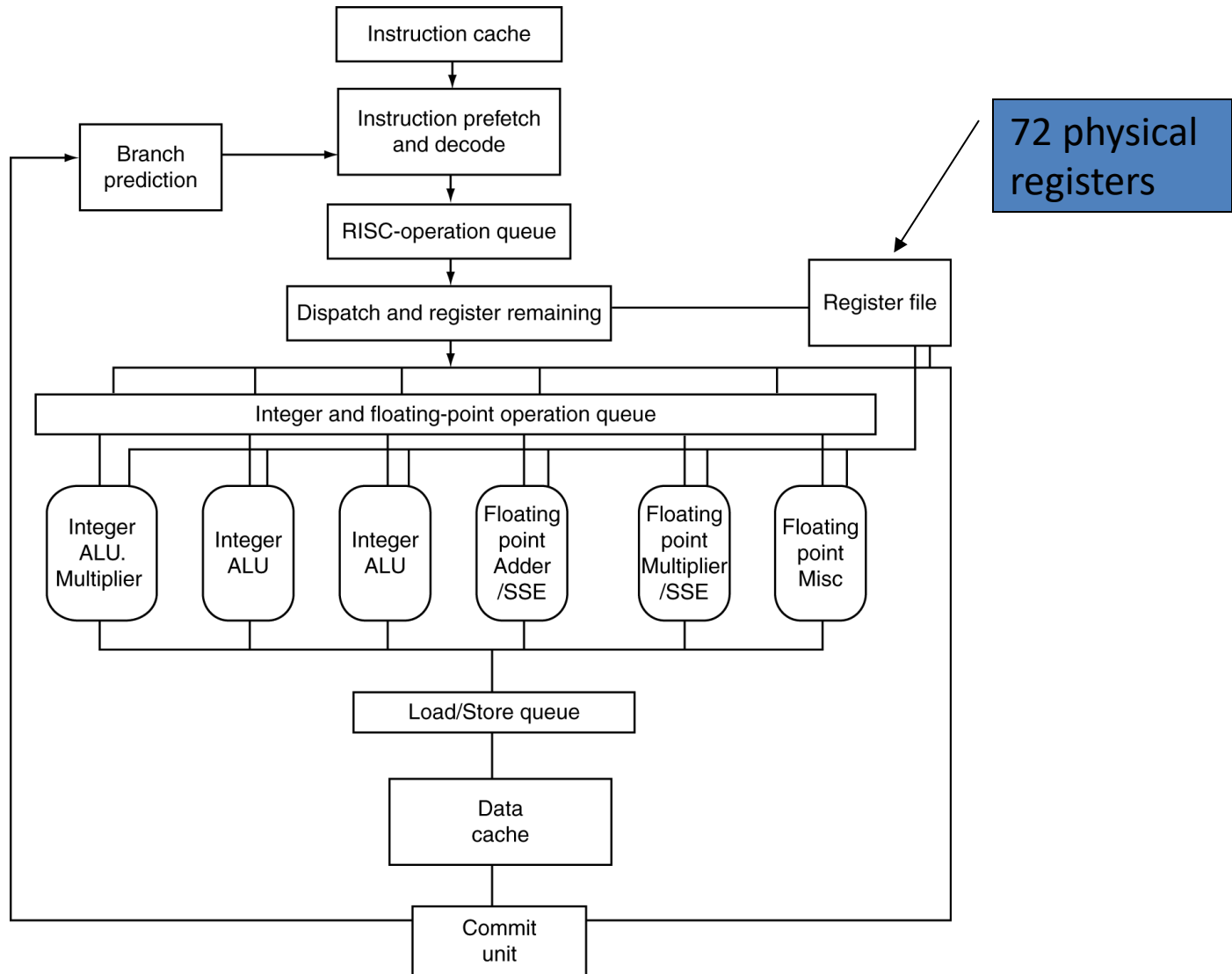
10^{-2} meters

AMD Opteron Microprocessor

centimeters

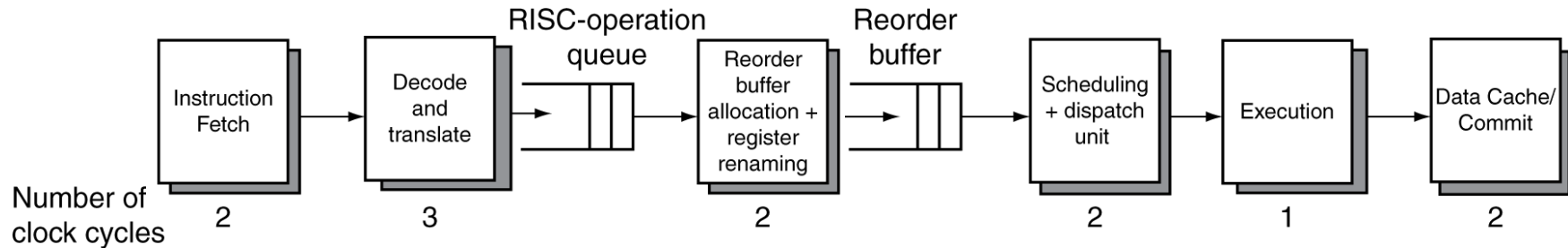


AMD Opteron Microarchitecture



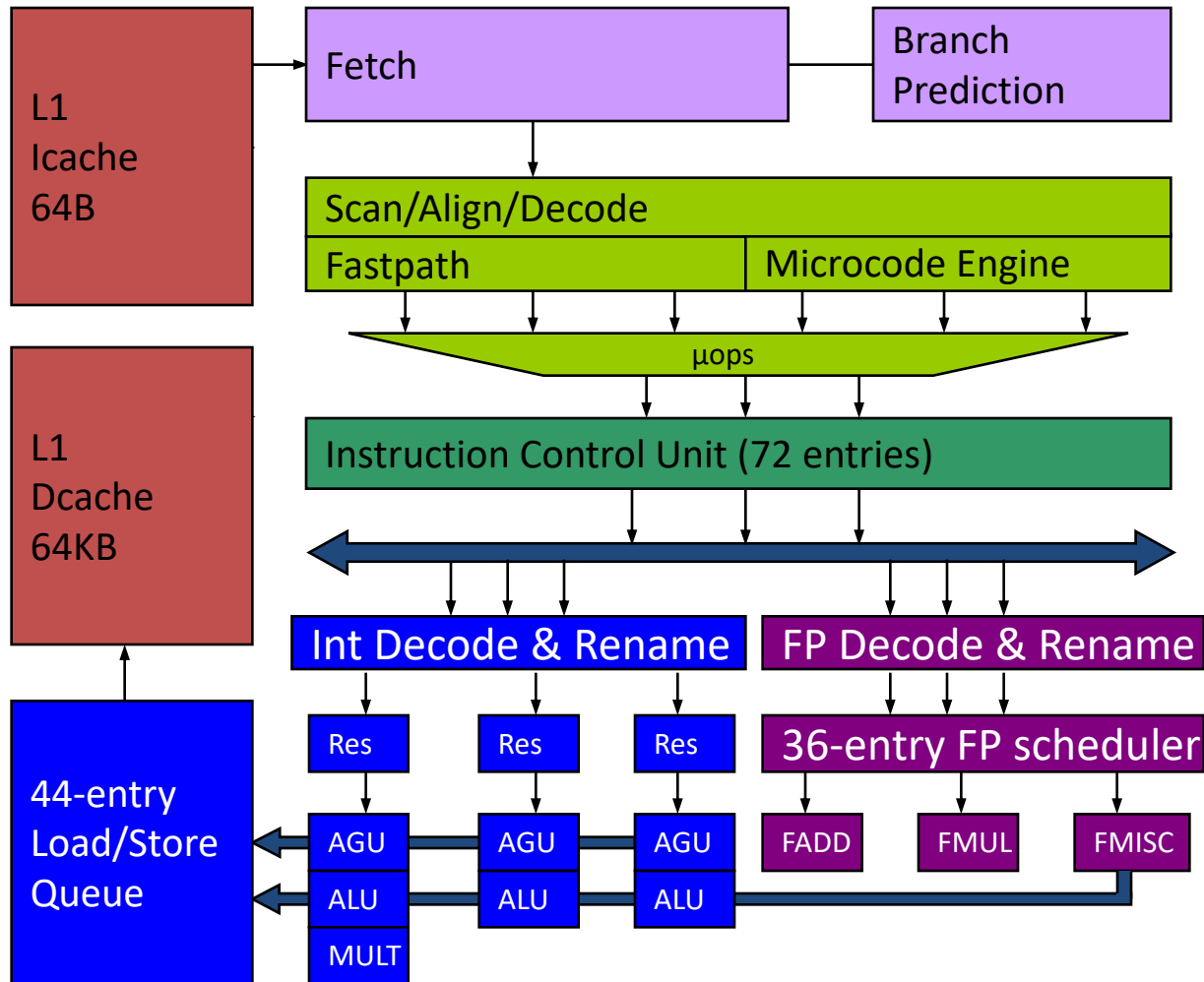
AMD Opteron Pipeline Flow

- For integer operations



- 12 stages (Floating Point is 17 stages)
- Up to 106 RISC-ops in progress

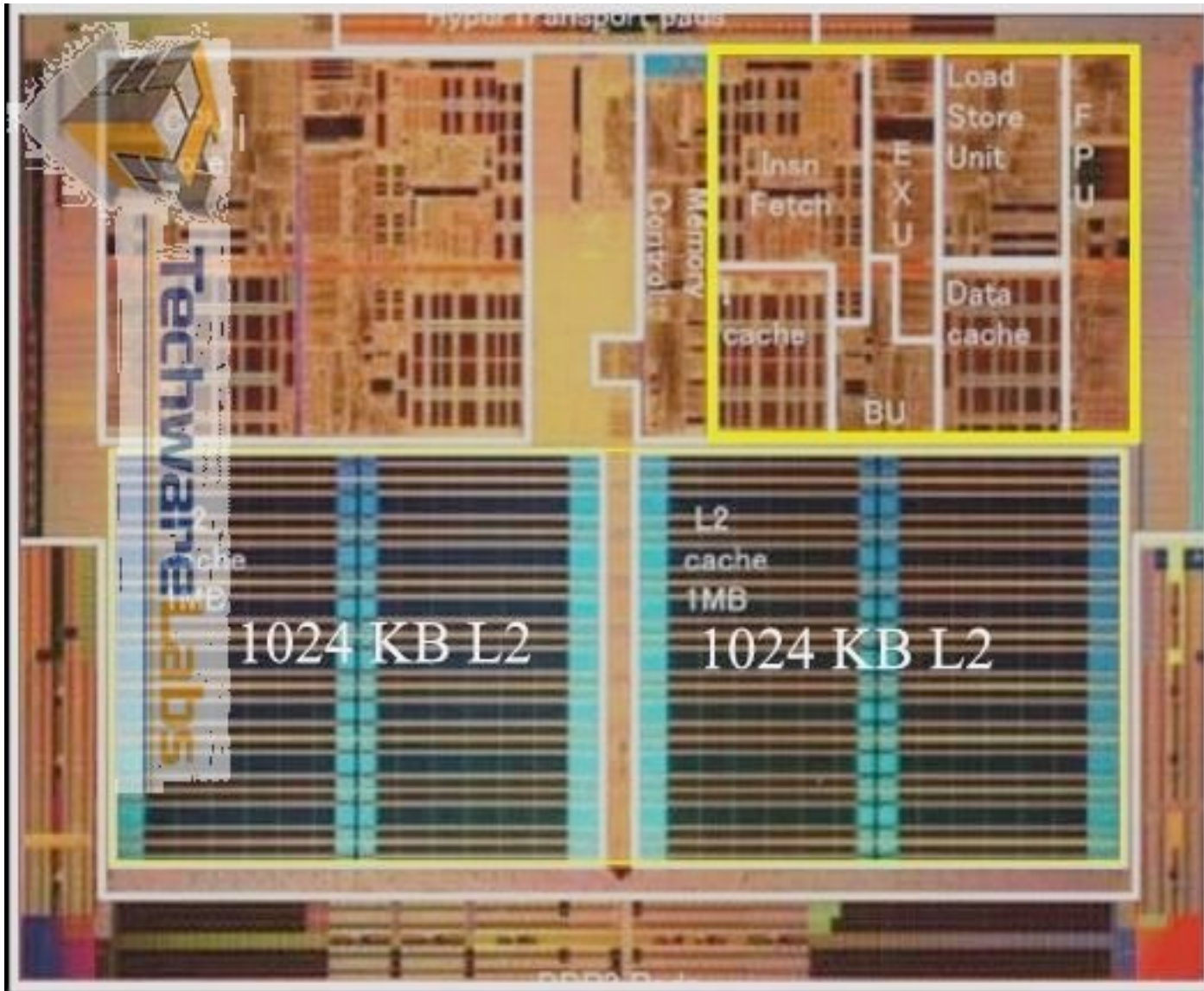
AMD Opteron Block Diagram



10⁻² meters

AMD Opteron Microprocessor

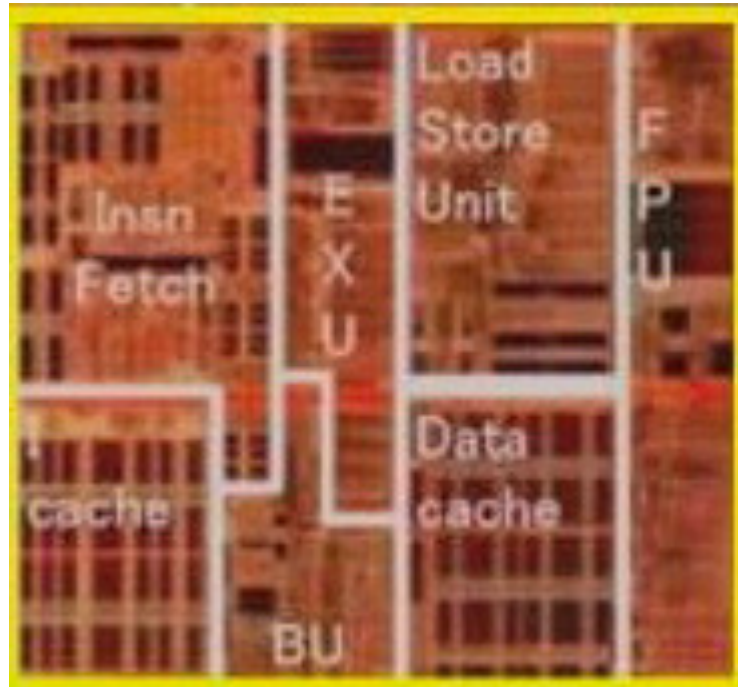
centimeters



10⁻³ meters

AMD Opteron Core

millimeters



Zoom into a Microchip

Zoom into a Microchip

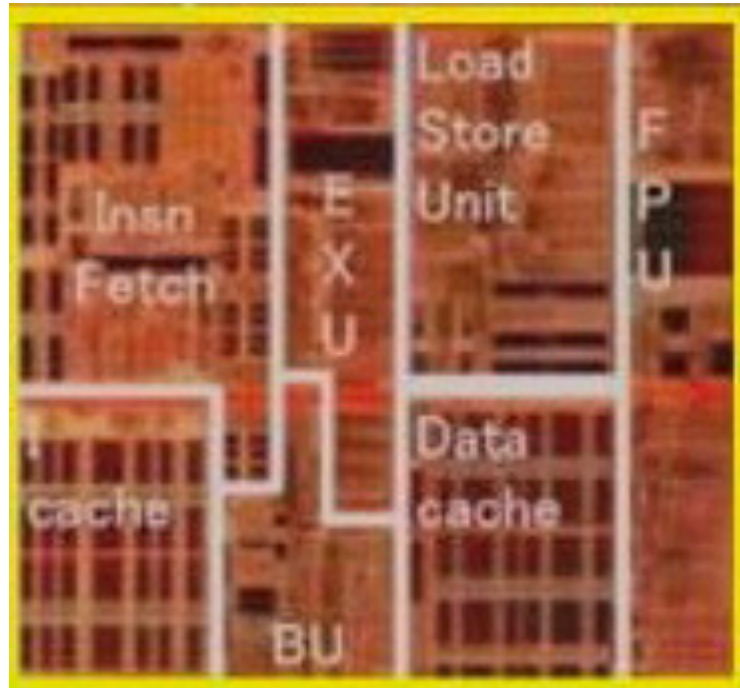
From a Digital Camera
To a Scanning Electron
Microscope

Produced by
NISE Net

10⁻³ meters

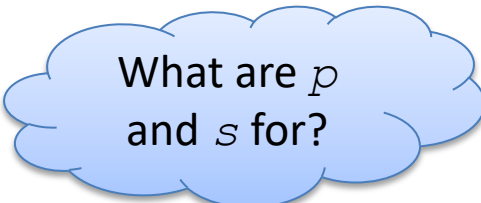
AMD Opteron Core

millimeters



Programming One Core: C with Intrinsics

```
void mmult(int n, float *A, float *B, float *C)
{
    for ( int i = 0; i < n; i+=4 )
        for ( int j = 0; j < n; j++ )
        {
            __m128 c0 = _mm_load_ps(C+i+j*n);
            for( int k = 0; k < n; k++ )
                c0 = _mm_add_ps(c0,
                    _mm_mul_ps(_mm_load_ps(A+i+k*n),
                               _mm_load1_ps(B+k+j*n)));
            _mm_store_ps(C+i+j*n, c0);
        }
}
```



What are p
and s for?

Inner loop from gcc -O -S

Assembly snippet from innermost loop:

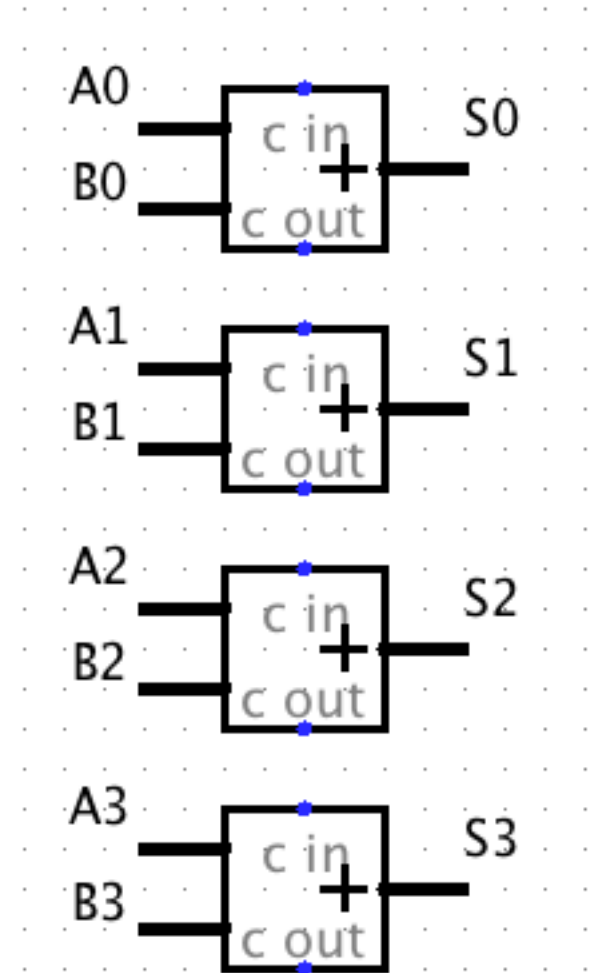
```
movaps (%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm8
movaps 16(%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm7
movaps 32(%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm6
movaps 48(%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm5
```

Great Ideas in Computer Architecture

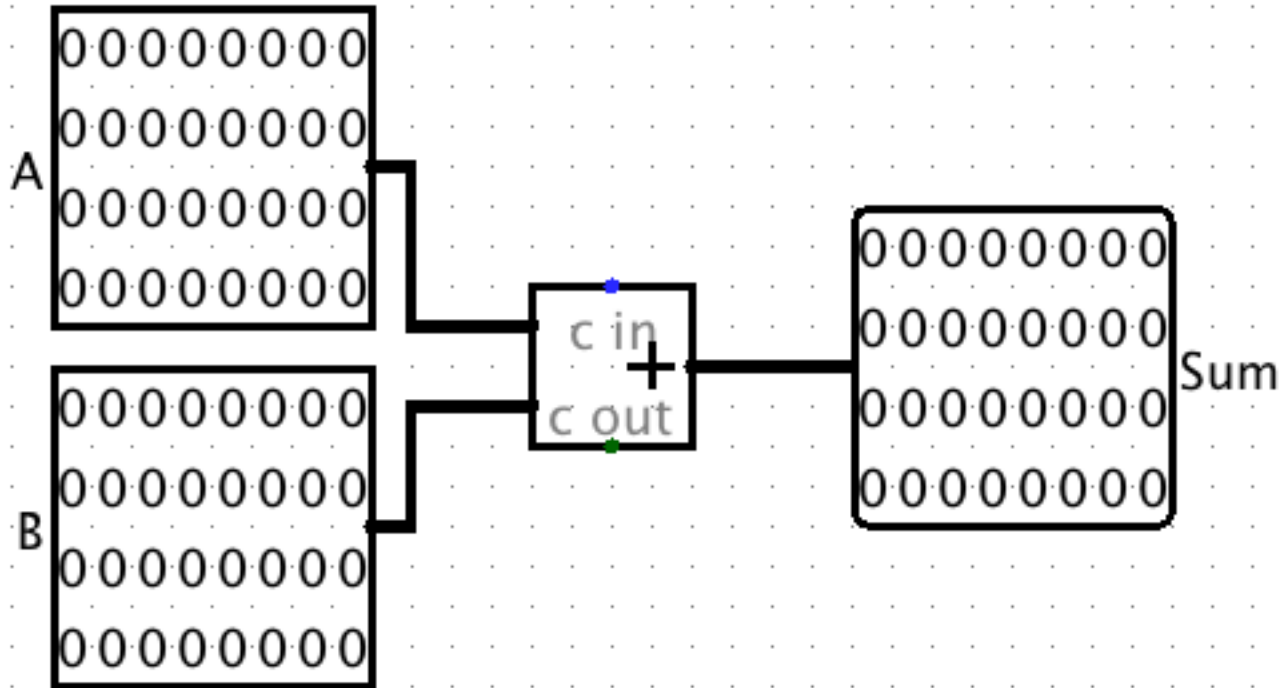
1. Design for Moore's Law
2. *Abstraction to Simplify Design*
 - *Instruction Set Architecture, Micro-operations*
3. Make the Common Case Fast
4. Dependability via Redundancy
5. Memory Hierarchy
6. *Performance via Parallelism/Pipelining/Prediction*
 - *Instruction-level Parallelism (superscalar, pipelining)*
 - *Data-level Parallelism*

SIMD Adder

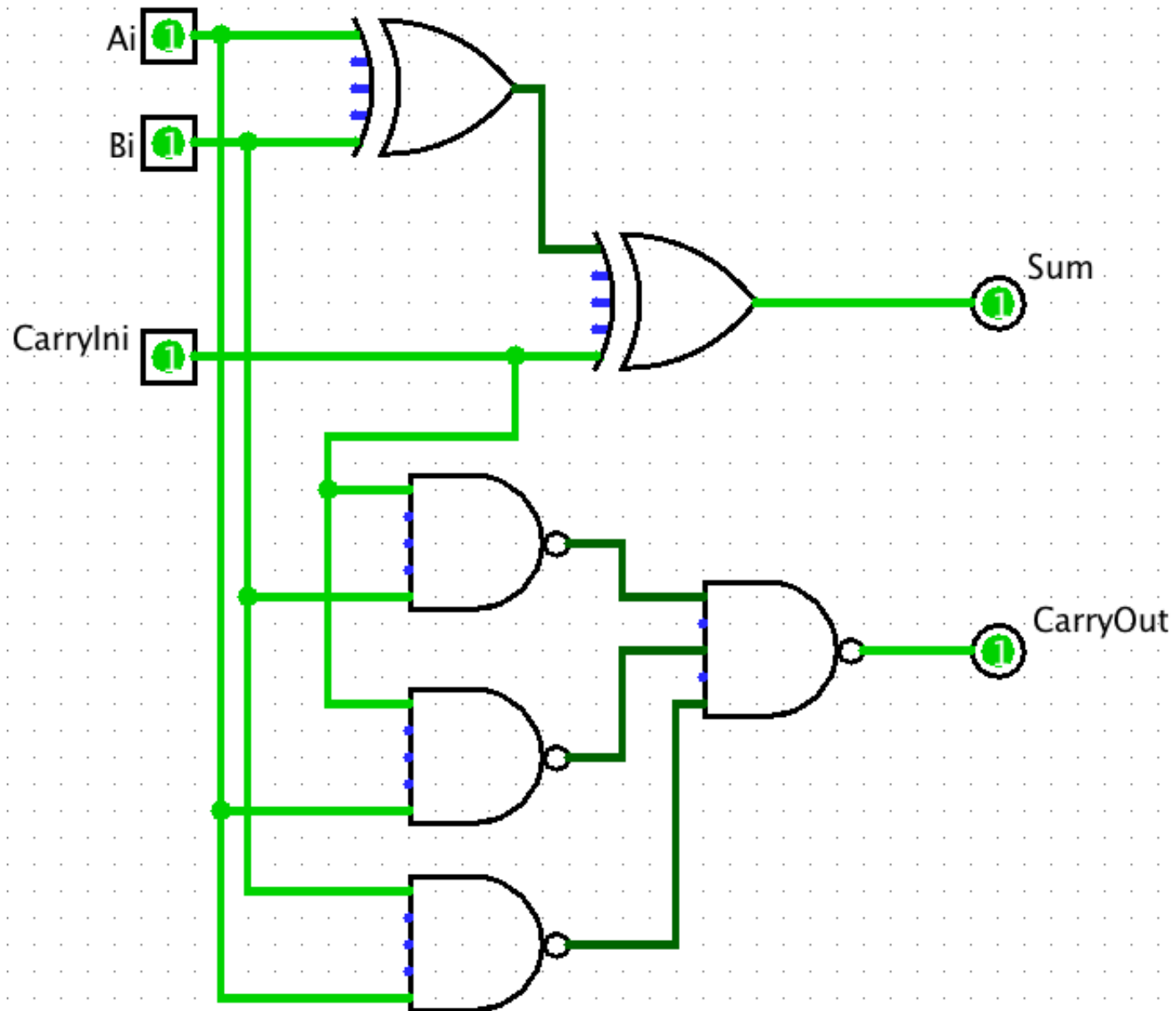
- Four 32-bit adders that operate in parallel
 - Data Level Parallelism



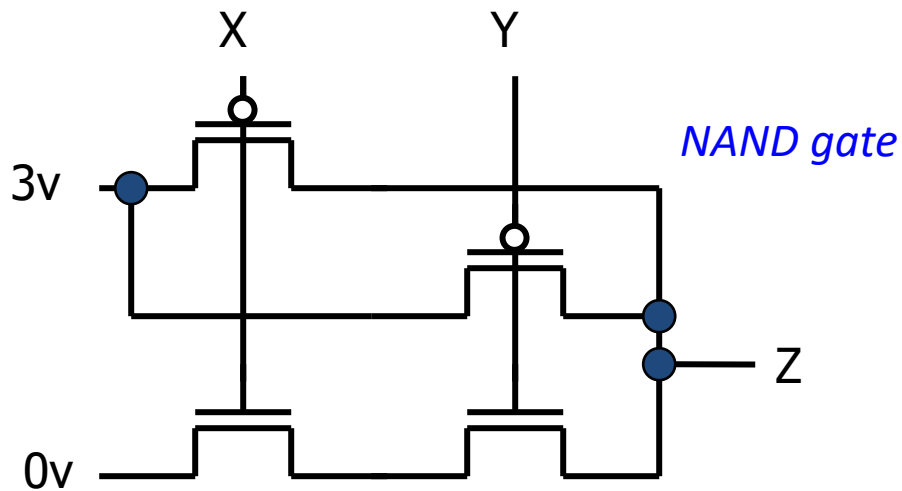
One 32-bit Adder



1 bit of 32-bit Adder



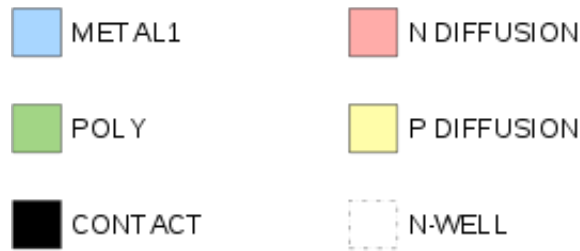
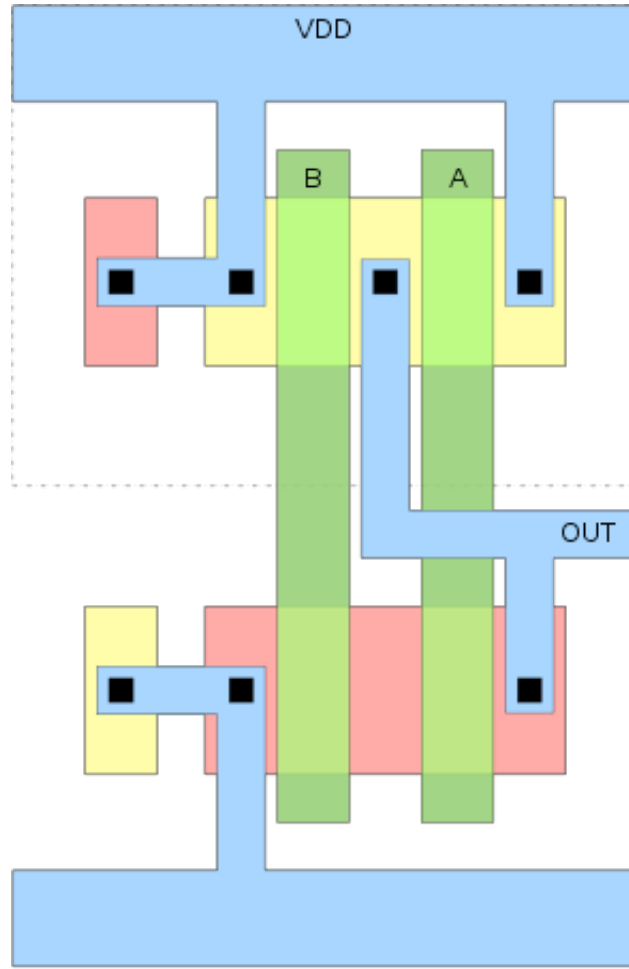
Complementary MOS Transistors (NMOS and PMOS) of NAND Gate



x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	3 volts
3 volts	0 volts	3 volts
3 volts	3 volts	0 volts

Physical Layout of NAND Gate 10^{-7} meters

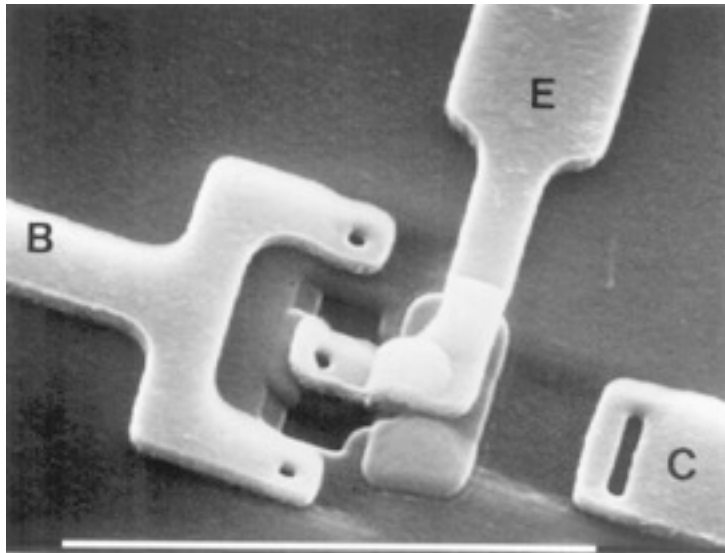
100 nanometers



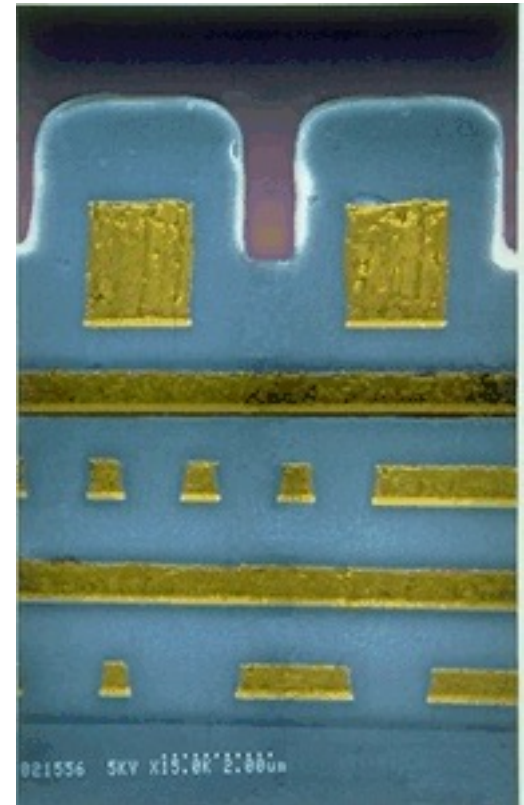
10^{-7} meters

Scanning Electron Microscope

100 nanometers



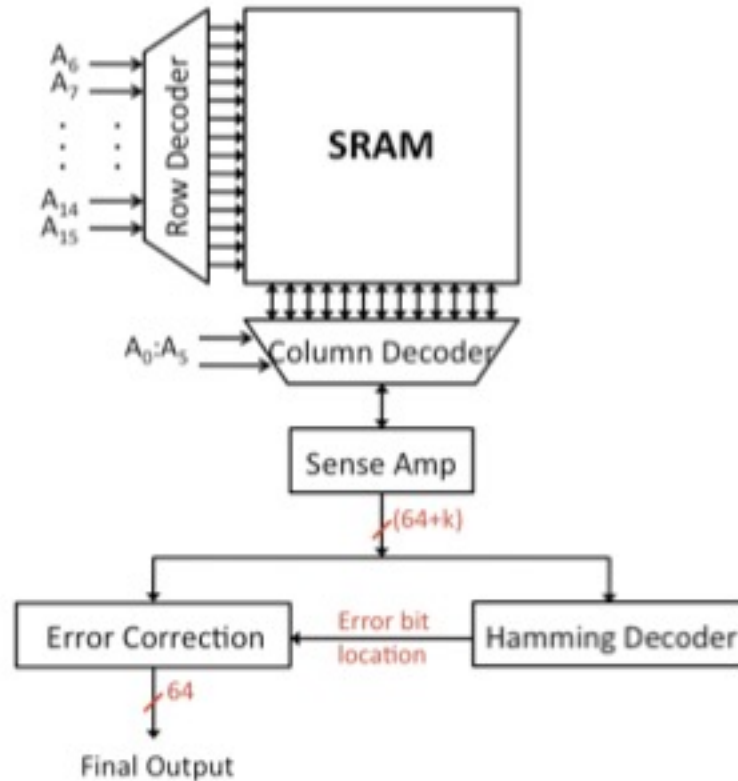
Top View



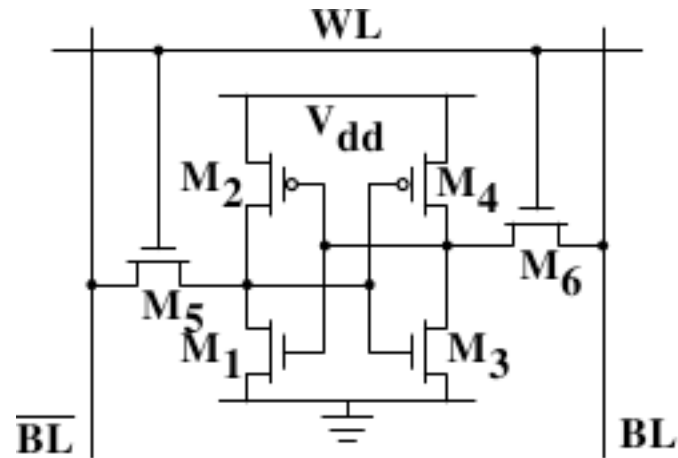
Cross Section

How to make a CMOS chip?

Block Diagram of Static RAM



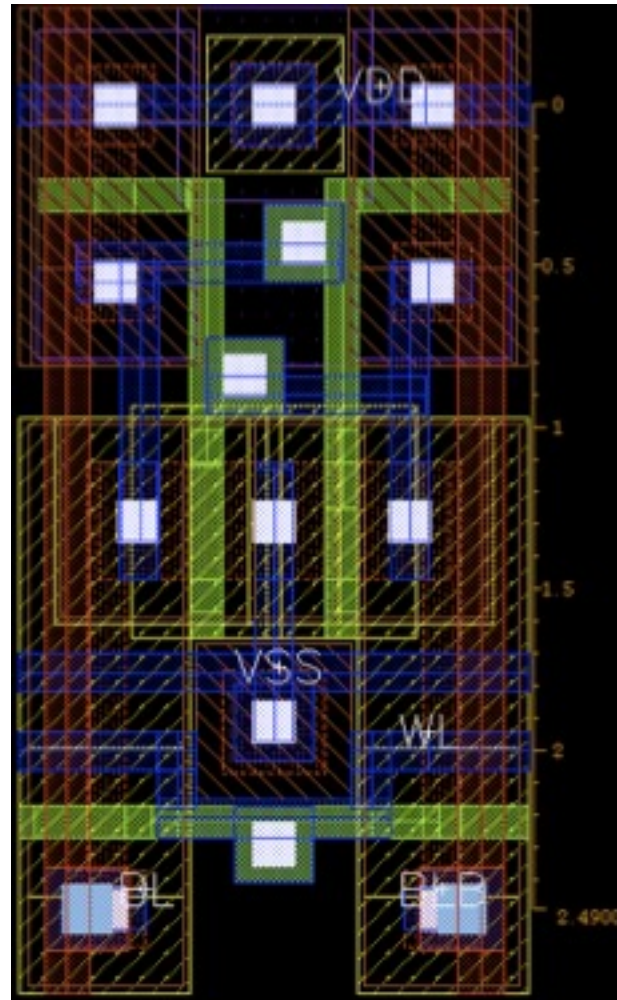
1 Bit SRAM in 6 Transistors



10^{-7} meters

Physical Layout of SRAM Bit

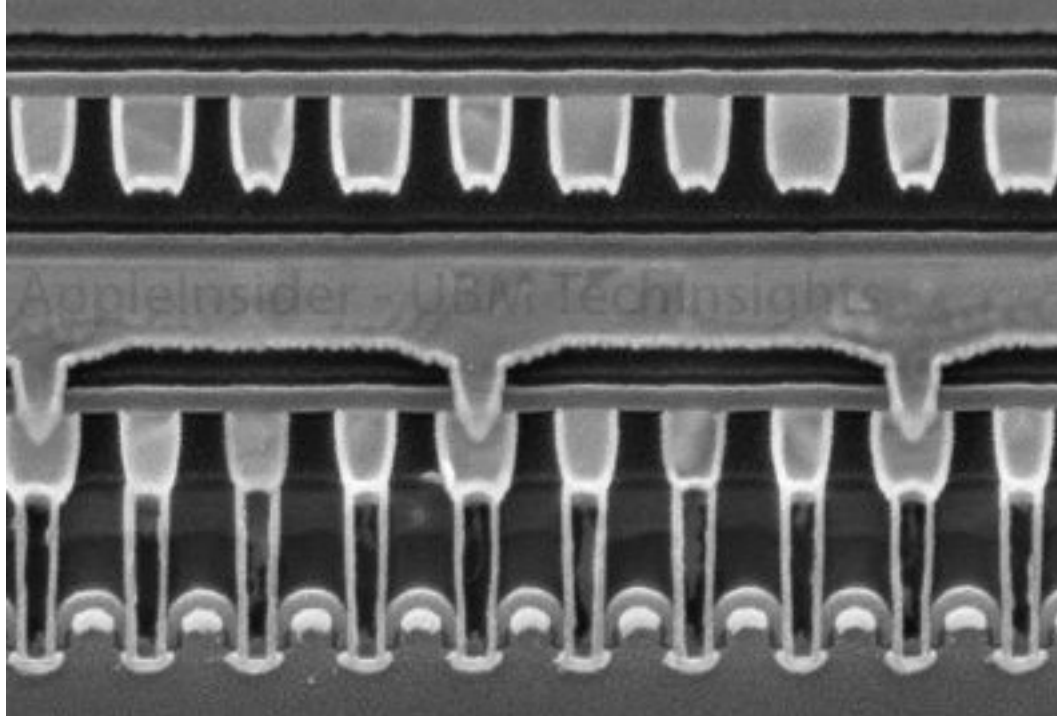
100 nanometers



10^{-7} meters

SRAM Cross Section

100 nanometers



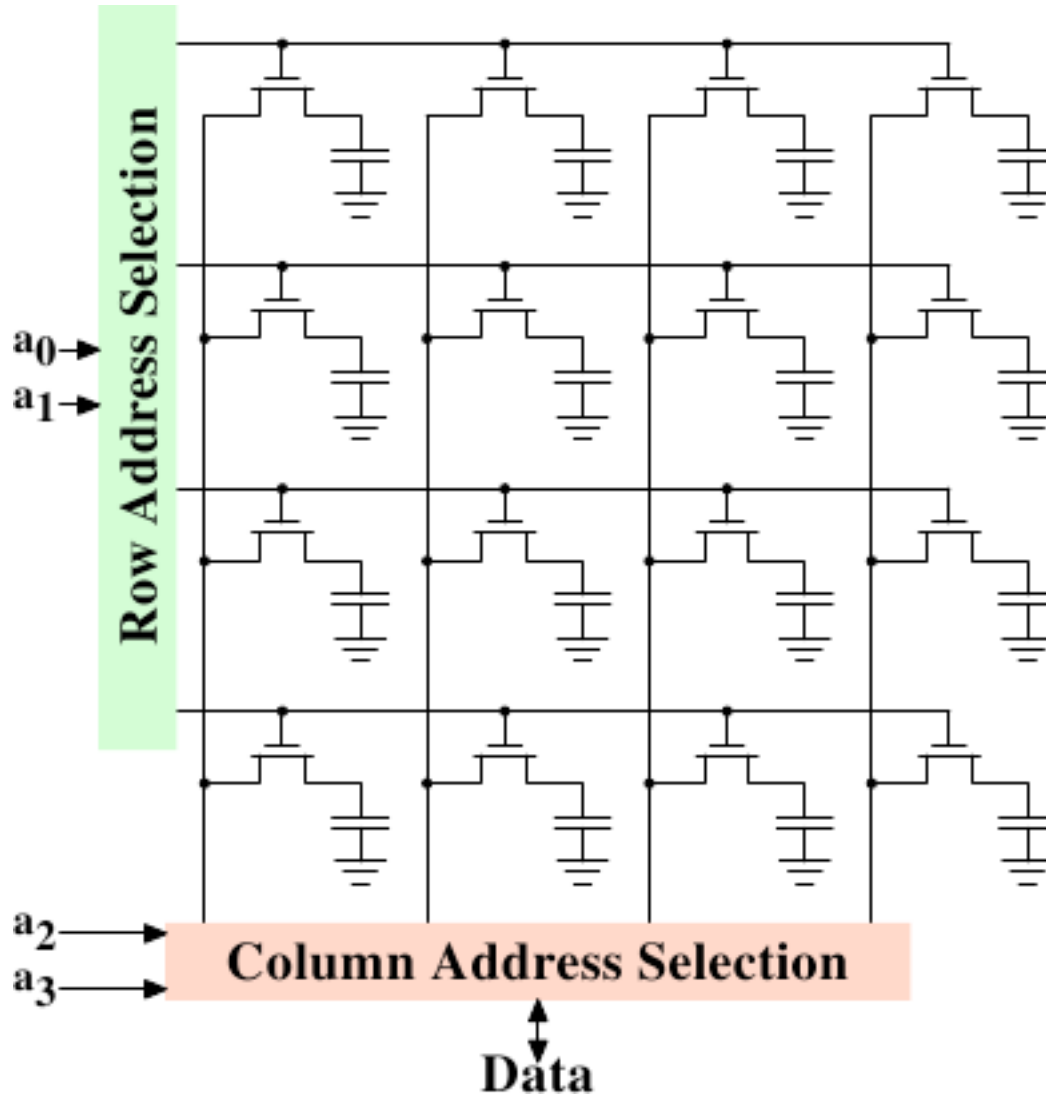
DIMM Module

- DDR = Double Data Rate
 - Transfers bits on Falling AND Rising Clock Edge
- Has Single Error Correcting, Double Error Detecting Redundancy (SEC/DED)
 - 72 bits to store 64 bits of data
 - Uses “Chip kill” organization so that if single DRAM chip fails can still detect failure
- Average server has 22,000 correctable errors and 1 uncorrectable error per year

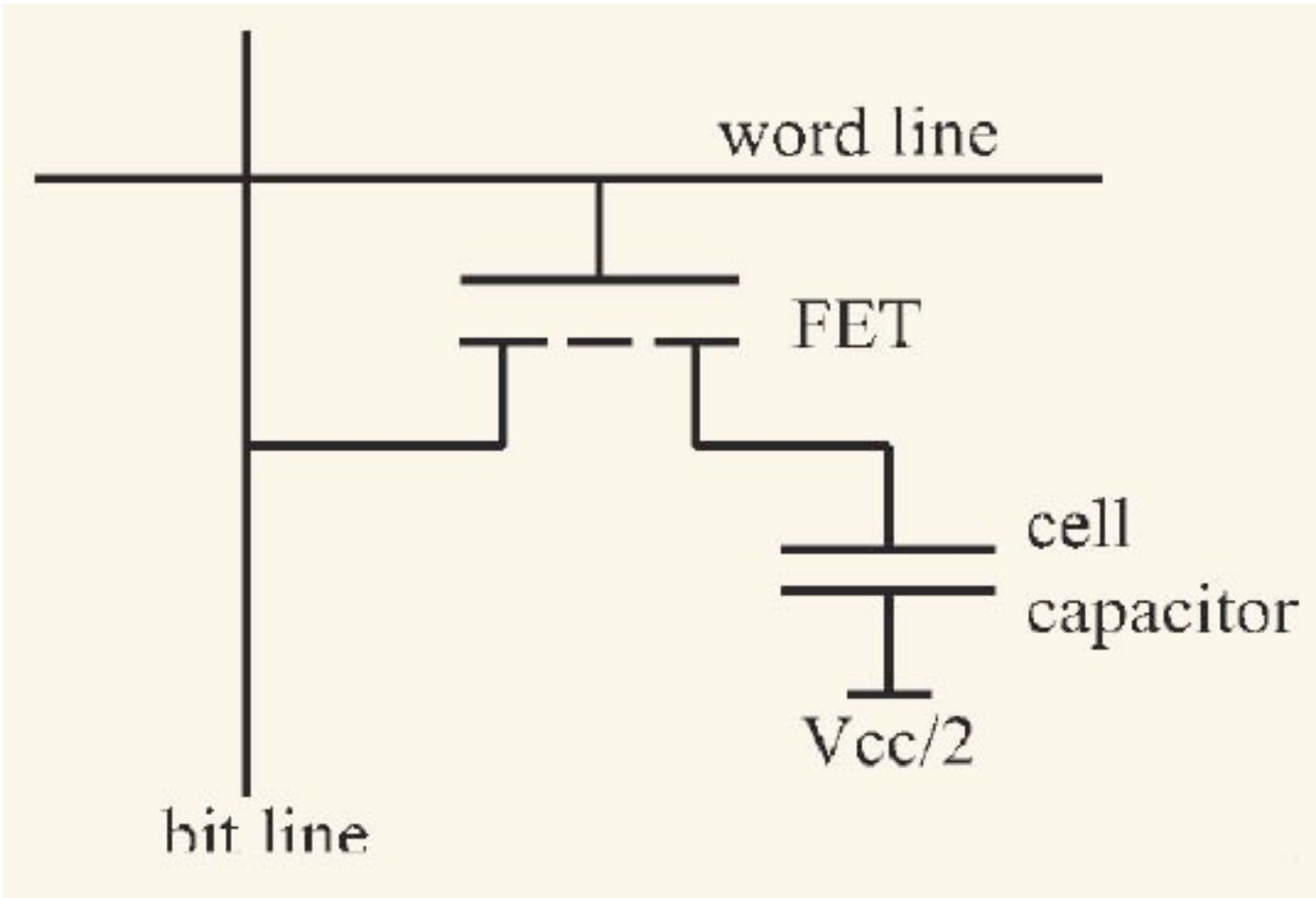
10^{-6} meters

DRAM Bits

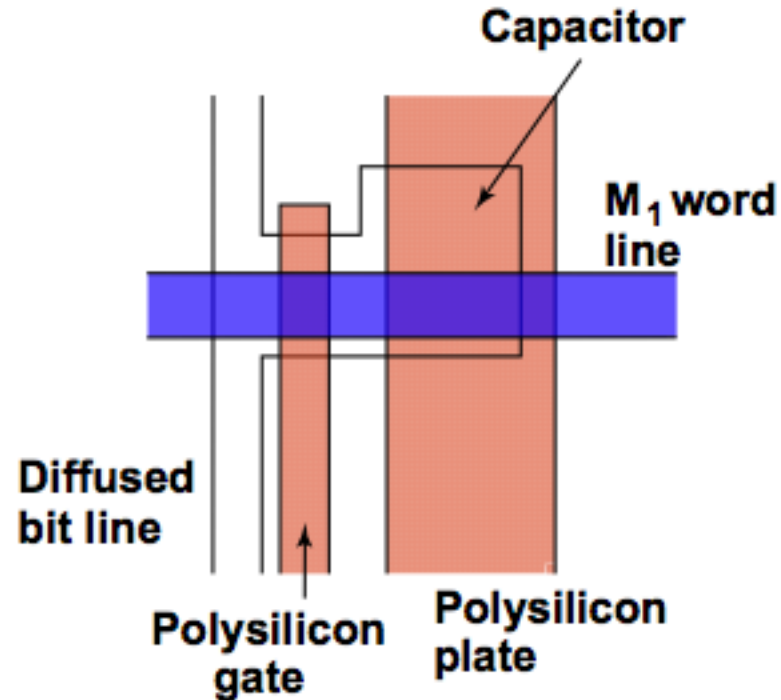
1 micron



DRAM Cell in Transistors



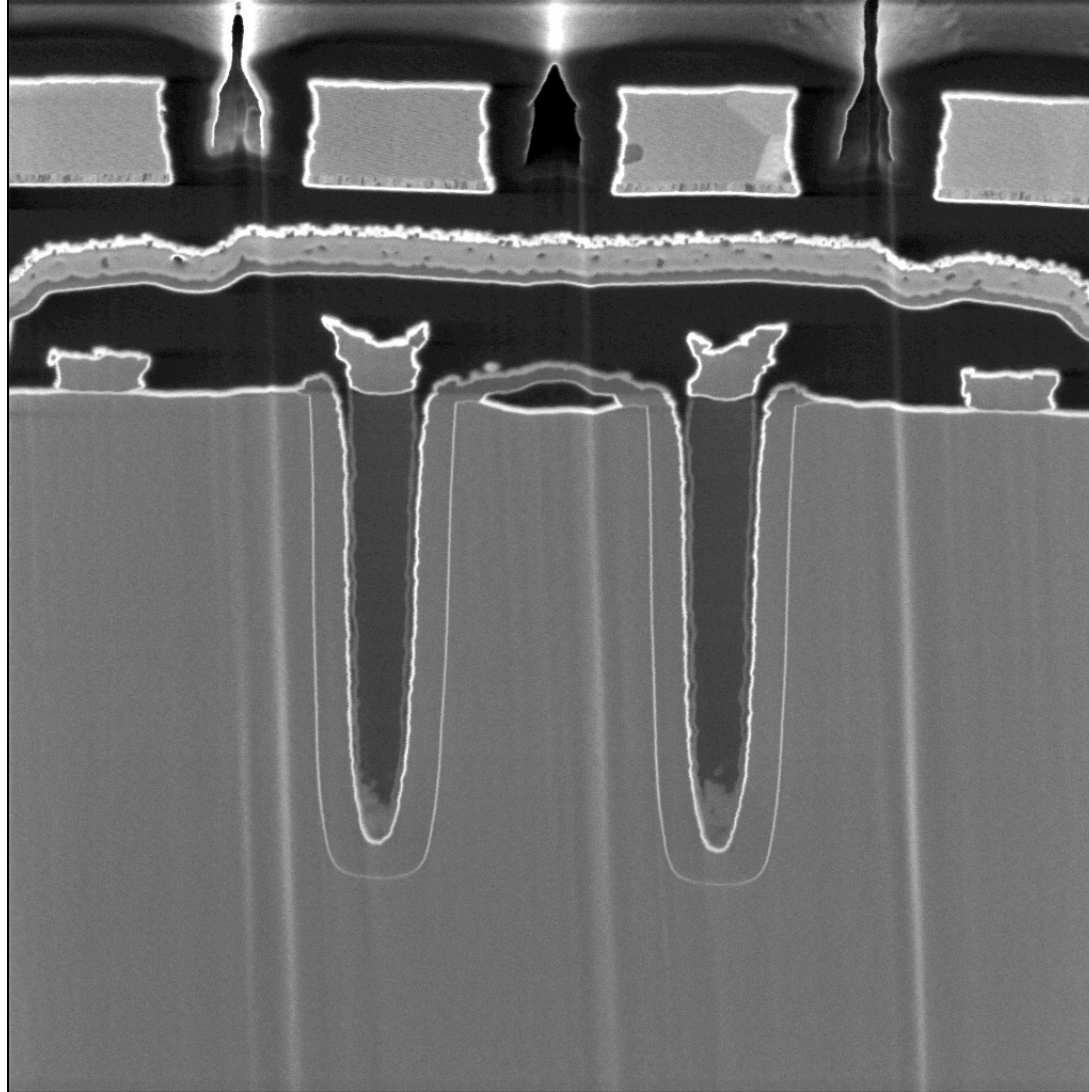
Physical Layout of DRAM Bit



10^{-7} meters

Cross Section of DRAM Bits

100 nanometers



AMD Opteron Dependability

- L1 cache data is SEC/DED protected
- L2 cache and tags are SEC/DED protected
- DRAM is SEC/DED protected with chipkill
- On-chip and off-chip ECC protected arrays include autonomous, background hardware scrubbers
- Remaining arrays are parity protected
 - Instruction cache, tags and TLBs
 - Data tags and TLBs
 - Generally read only data that can be recovered from lower levels

Programming Memory Hierarchy: Cache Blocked Algorithm

- The blocked version of the i-j-k algorithm is written simply as (A,B,C are submatrices of a, b, c)

```
for (i=0;i<N/r;i++)  
  for (j=0;j<N/r;j++)  
    for (k=0;k<N/r;k++)  
      C[i][j] += A[i][k]*B[k][j]
```

- r = block (sub-matrix) size (Assume r divides N)
- $X[i][j]$ = a sub-matrix of X , defined by block row i and block column j

Great Ideas in Computer Architecture

1. *Design for Moore's Law*
 - Higher capacities caches and DRAM
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. *Dependability via Redundancy*
 - Parity, SEC/DEC
5. *Memory Hierarchy*
 - Caches, TLBs
6. *Performance via Parallelism/Pipelining/Prediction*
 - Data-level Parallelism

Course Summary

- As the field changes, Computer Architecture courses change, too!
- It is still about the software-hardware interface
 - Programming for performance!
 - Parallelism: Task-, Thread-, Instruction-, and Data-MapReduce, OpenMP, C, SSE Intrinsics
 - Understanding the memory hierarchy and its impact on application performance

All the best!

