CMPE 200
Computer Architecture & Design

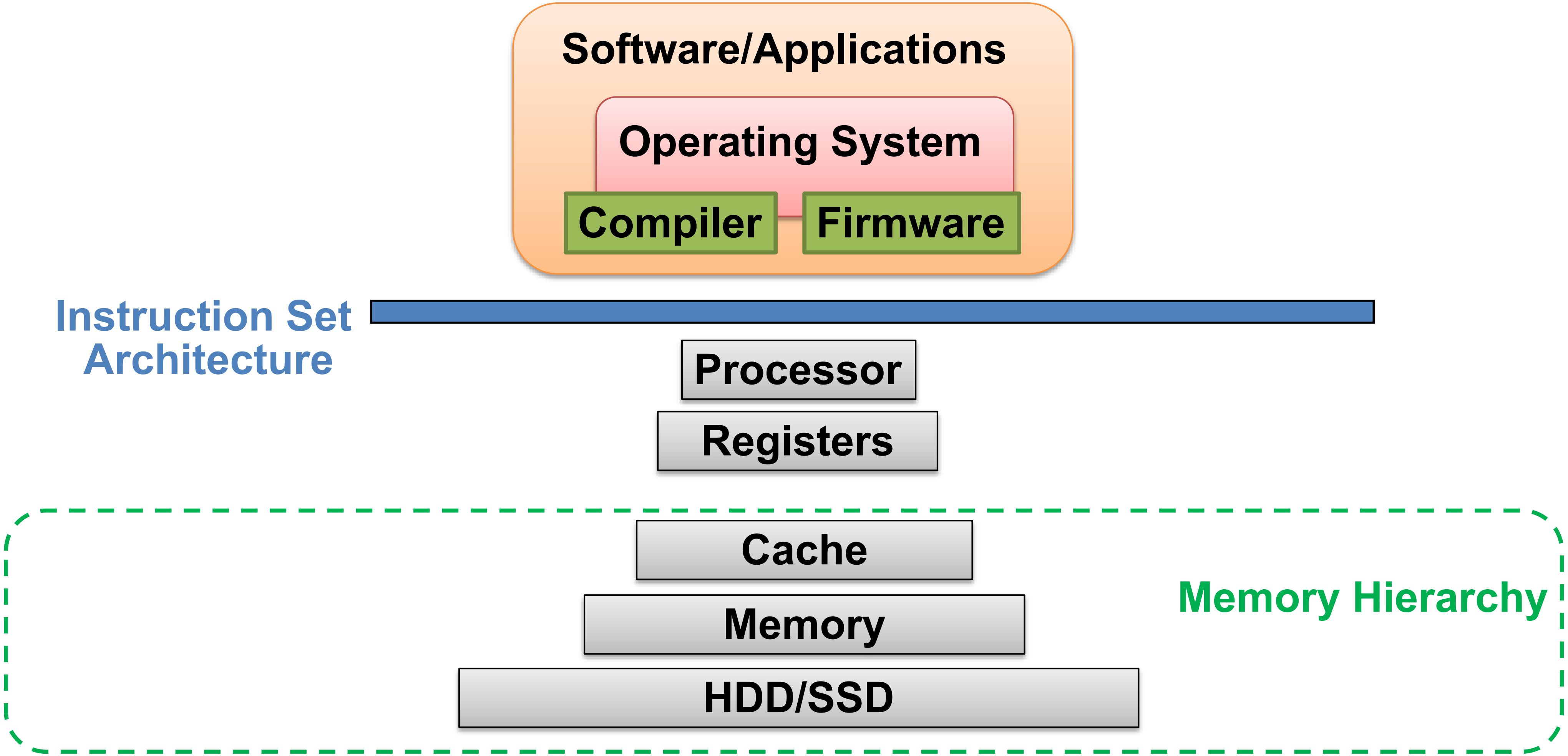
Lecture 4.
Memory Hierarchy (1)

Haonan Wang

SJSU

SAN JOSÉ STATE
UNIVERSITY

Computer Architecture Overview



Memory Hierarchy

Example: a C program that reads two integer values from “file.txt” file and prints the sum of them.

```
#include <stdio.h>
#include <string.h>

int numbers[2];

void myfunction(void)
{
   FILE *fp;
   int size = 2;
   int sum = 0;

   /* Open file for reading */
   fp = fopen(“mynumbers.txt", “r");

   /* Read and display data */
   fread(numbers, sizeof(int), size, fp);
   fclose(fp);
   sum = numbers[0] + numbers[1];
   printf(“Sum = %d\n”, sum);
}

int main (void) {
   myfunction();
   return(0);
}
```

Processor (CPU)

Understands and executes each line of the code.

Uses fast on-chip memories

Memory (DRAM)

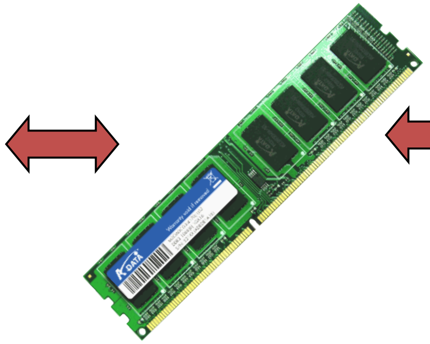
Provides operands to CPU

(*fp, size, sum, numbers[2])

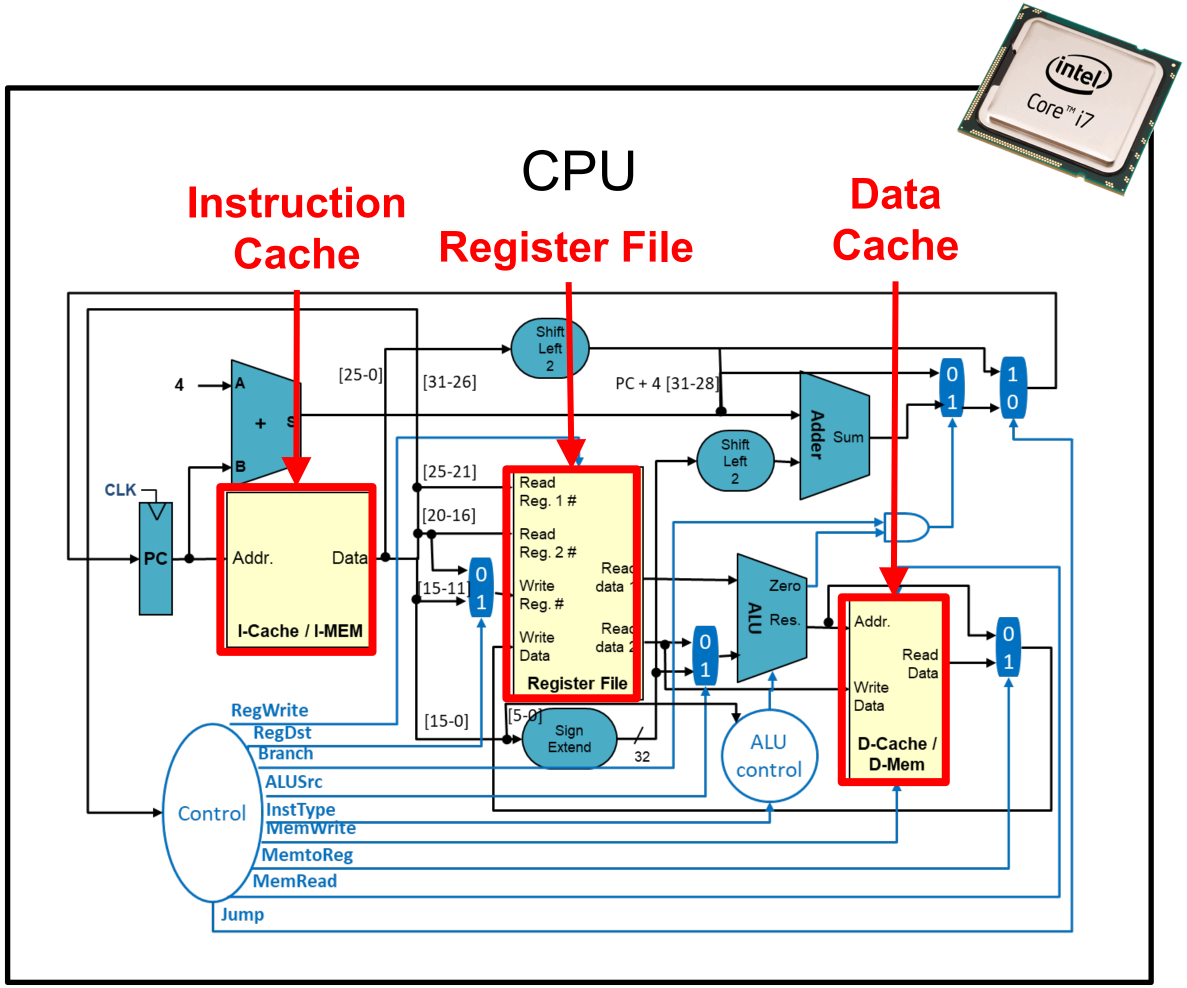
Storage (HDD)

Provides file inputs and program code

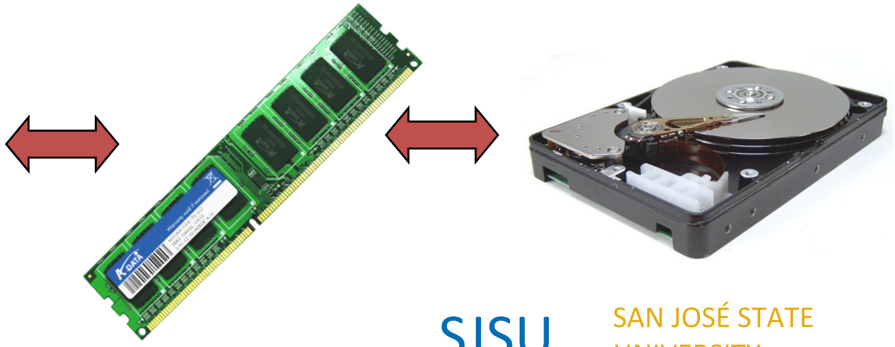
(file.txt)



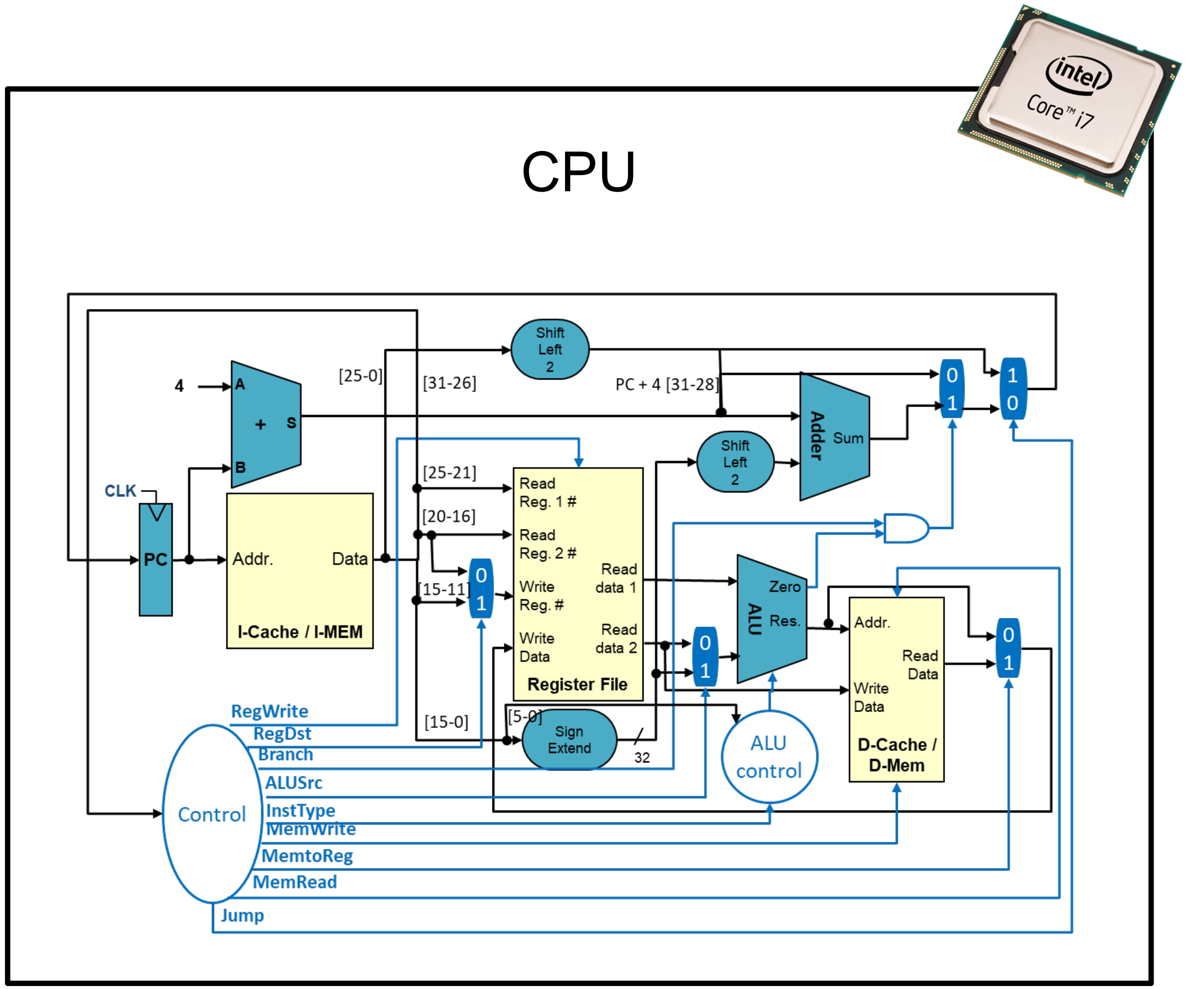
Memory Hierarchy



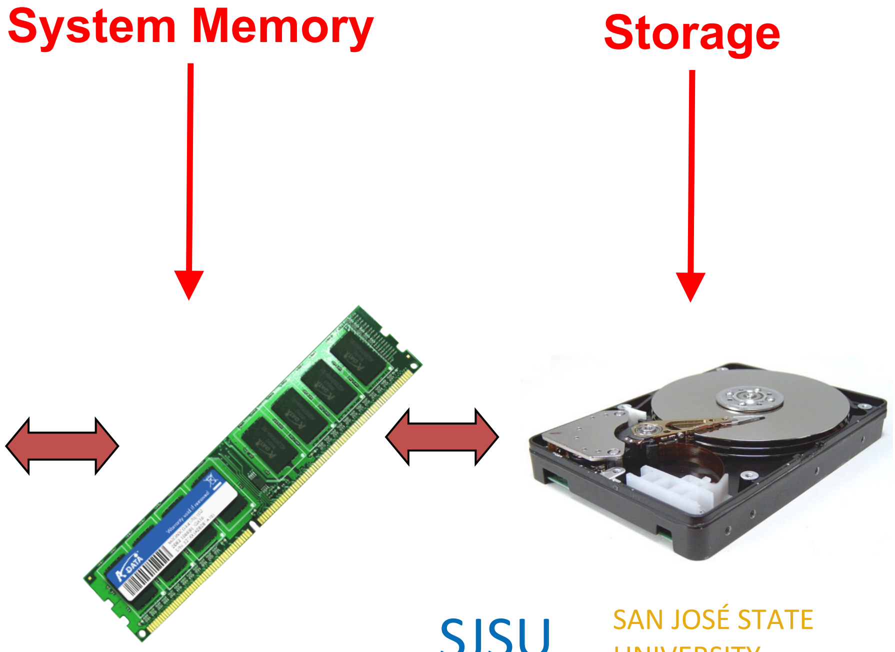
- **On-chip Memories (Memories inside of CPU)**
  - Register file, Caches
  - Small but Fast



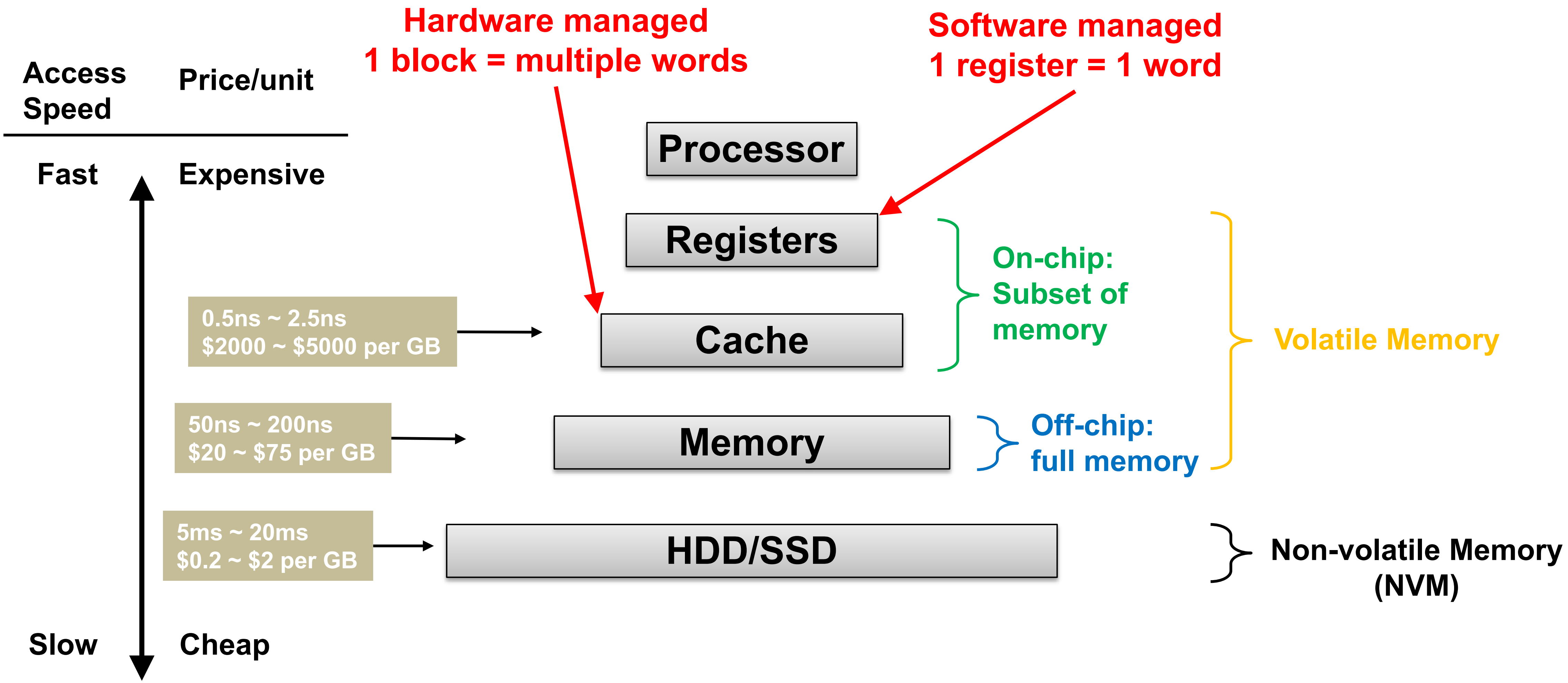
Memory Hierarchy



- **Off-chip Memories (Memories outside CPU)**
  - System Memory, Storage
  - Large but Slow



Memory Hierarchy



Access Speed

Price/unit

Fast

Expensive

Slow

Cheap

Hardware managed
1 block = multiple words

Software managed
1 register = 1 word

Processor

Registers

Cache

Memory

HDD/SSD

0.5ns ~ 2.5ns
$2000 ~ $5000 per GB

50ns ~ 200ns
$20 ~ $75 per GB

5ms ~ 20ms
$0.2 ~ $2 per GB

On-chip:
Subset of
memory

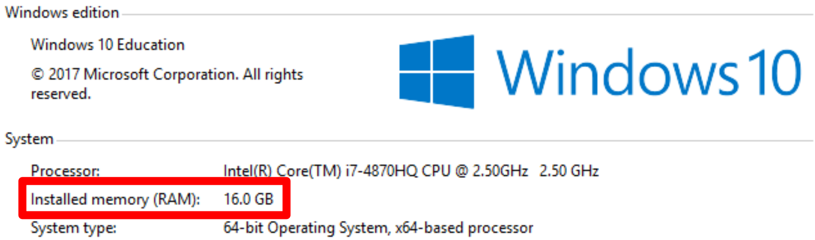
Off-chip:
full memory

Volatile Memory

Non-volatile Memory
(NVM)

Memories in Your PC

- **Windows**
  - This PC → Properties
  - cmd window → wmic
  - 3rd party tool like CPU-Z



Windows edition

Windows 10 Education

© 2017 Microsoft Corporation. All rights reserved.

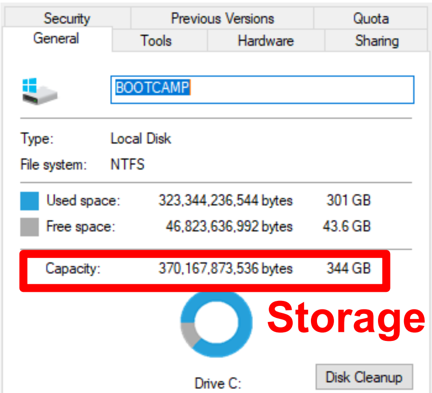
System

Processor: Intel(R) Core(TM) i7-4870HQ CPU @ 2.50GHz 2.50 GHz

Installed memory (RAM): 16.0 GB

System type: 64-bit Operating System, x64-based processor

System Memory



BOOTCAMP

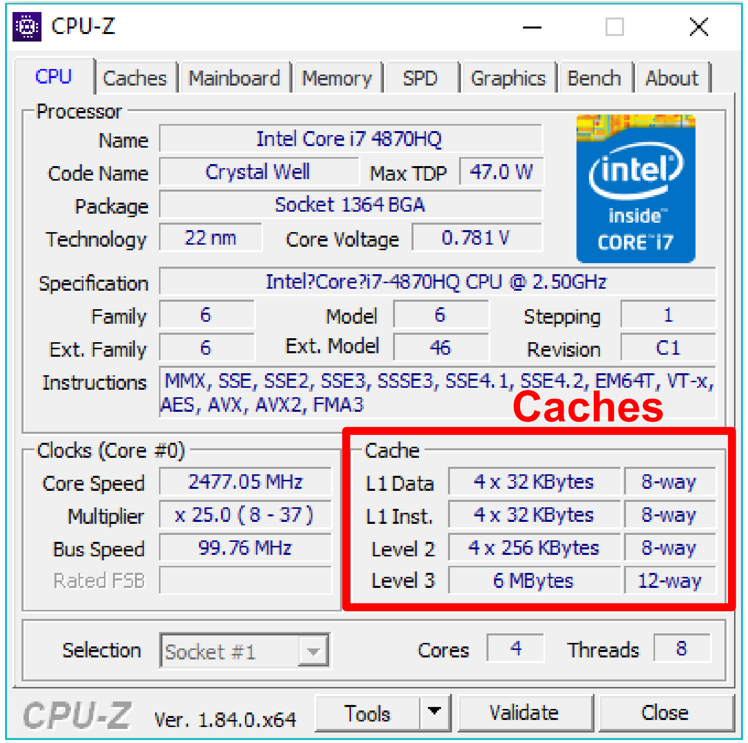
Type: Local Disk

File system: NTFS

Used space:	323,344,236,544 bytes	301 GB
Free space:	46,823,636,992 bytes	43.6 GB
Capacity:	370,167,873,536 bytes	344 GB

Drive C:

Storage



Processor

Name: Intel Core i7 4870HQ

Code Name: Crystal Well | Max TDP: 47.0 W

Package: Socket 1364 BGA

Technology: 22 nm | Core Voltage: 0.781 V

Specification: Intel?Core?i7-4870HQ CPU @ 2.50GHz

Family: 6 | Model: 6 | Stepping: 1

Ext. Family: 6 | Ext. Model: 46 | Revision: C1

Instructions: MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX, AVX2, FMA3

Clocks (Core #0)

Core Speed: 2477.05 MHz

Multiplier: x 25.0 (8 - 37)

Bus Speed: 99.76 MHz

Rated FSB:

Cache

L1 Data	4 x 32 KBytes	8-way
L1 Inst.	4 x 32 KBytes	8-way
Level 2	4 x 256 KBytes	8-way
Level 3	6 MBytes	12-way

Caches

Selection: Socket #1 | Cores: 4 | Threads: 8

CPU-Z Ver. 1.84.0.x64

- **Linux**
  - lscpu
  - cat /proc/cpuinfo
  - etc.

Discussion

How would you design the memory system?

Single piece of memory that does everything

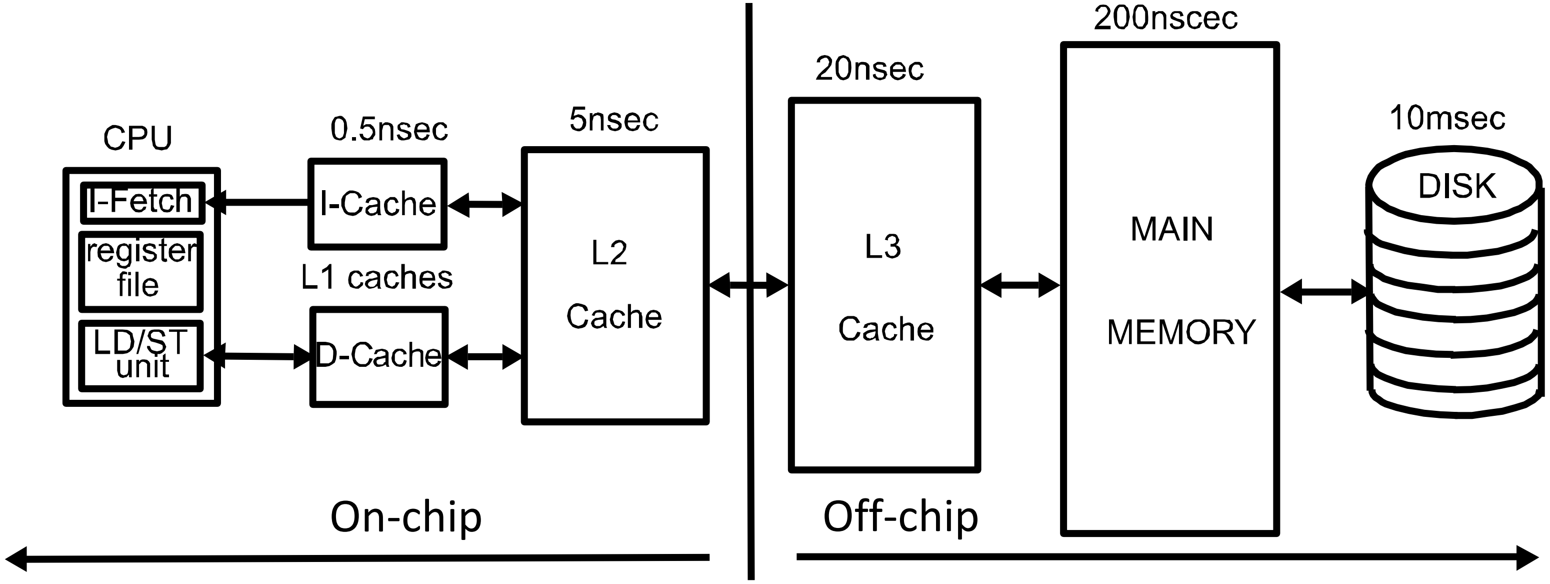
Vs.

Multiple levels of memories

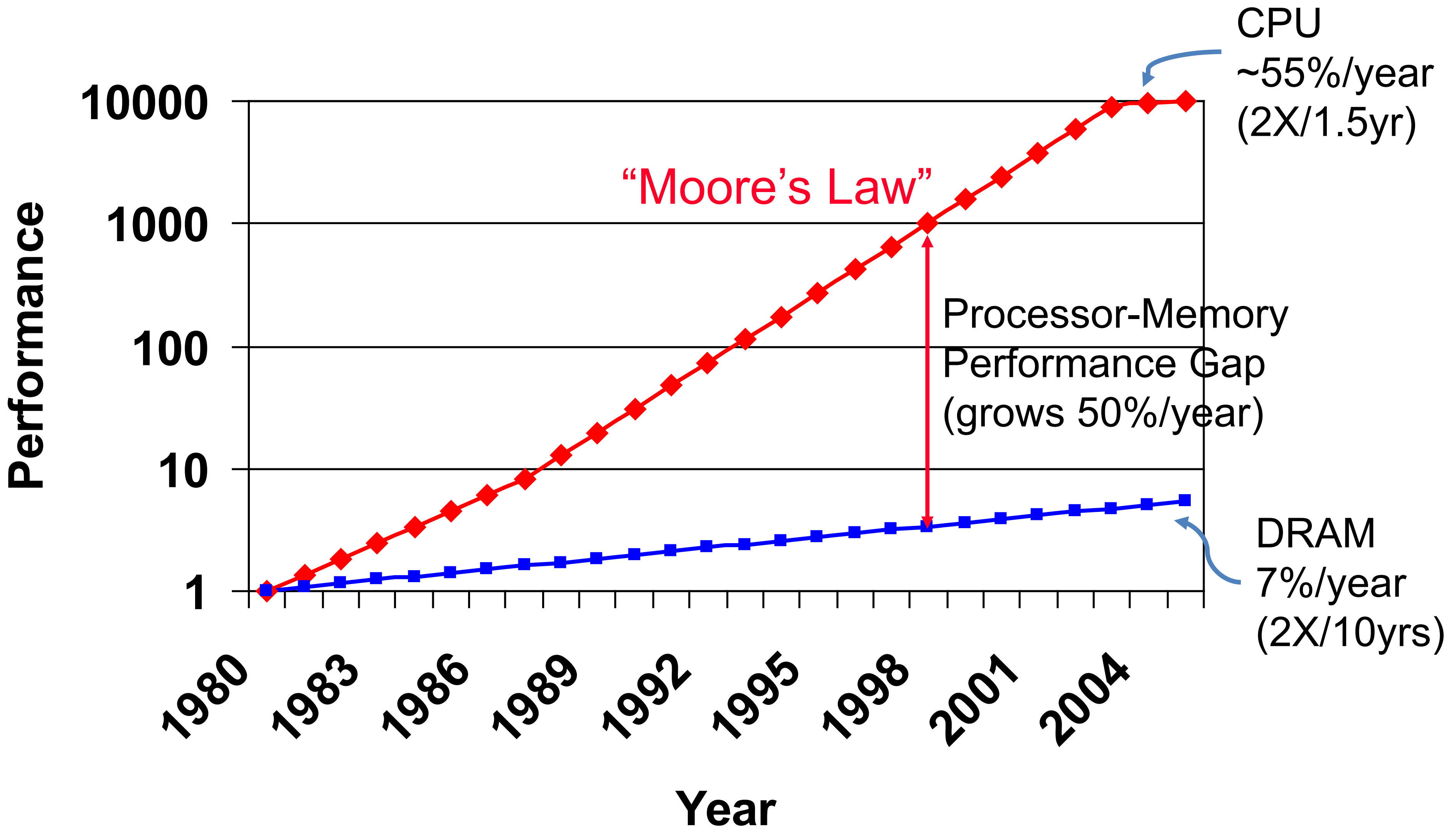
Why should we use this hierarchy design?

Why?

- **Two Types of Locality:**
  - **Temporal Locality** (Locality in Time): If an address is referenced, it tends to be referenced again (e.g., loops, variable reuse)
  - **Spatial Locality** (Locality in Space): If an address is referenced, neighboring addresses tend to be referenced (e.g., array, stack, etc.)



The “Memory Wall”



The Memory Hierarchy Goal

- **How do we create a memory system that gives the illusion of being large, cheap and fast (most of the time)?**
  - With hierarchy
    - try the fast parts first -- most of the time, this works well
    - if not, move the data so it works well the next time
  - With parallelism
    - use multiple identical parts operating simultaneously
    - for large quantities of data, this works well

- **Example – keep a subset of the data in fast memory**
- **Example – 1-byte-wide memory ➔ 4 × 1-byte-wide memory ➔ 4-byte-wide memory**
  - load word takes 4 memory cycles vs. 1 memory cycle

Let us Conclude

What is cache?

Fast memory

What are the two types of data localities?

Temporal & Spatial

What are the two memory design directions inspired by the principle of locality?

Hierarchy & parallelism

SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY