w10-Lec

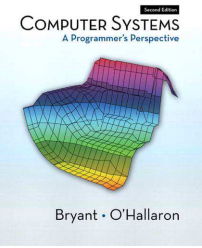
Basic Machine Organization

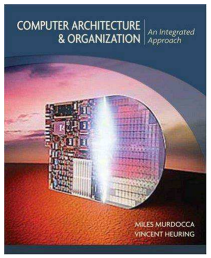
Assembled for 204111

by Ratsameetip Wita

Basic Machine Organization

- **History of Computer System**
- Hardware Organization
- Von Neumann Architecture
- Program Representation in Machine-Level

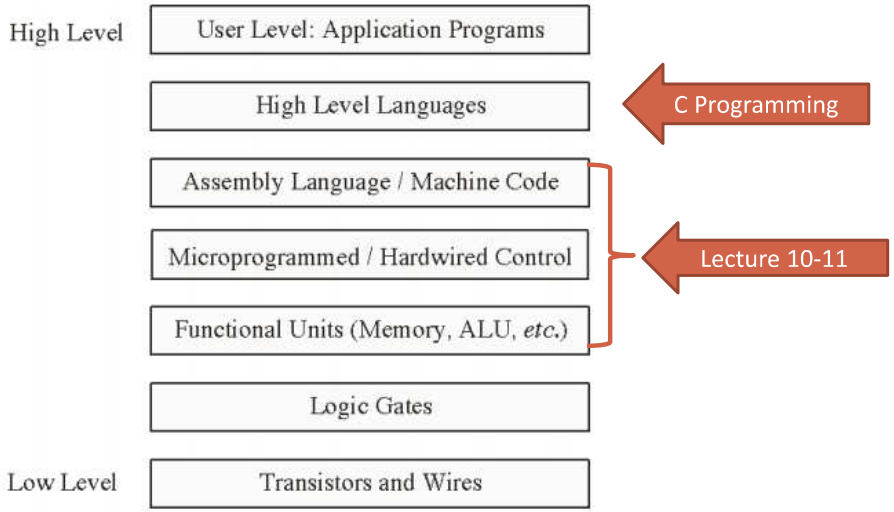




Computer Architecture and Organization: An Integrated Approach, 1st Edition, 2007 - Miles J. Murdocca and Vincent P. Heuring,

Computer Systems: A Programmer's Perspective - Bryant & O'Hallaron

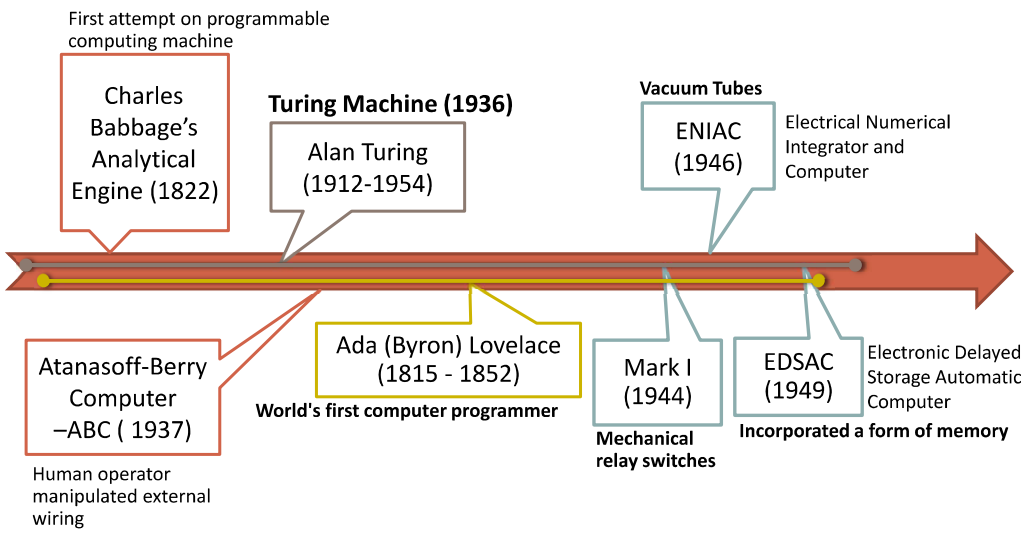
Levels of Machines



Computer System

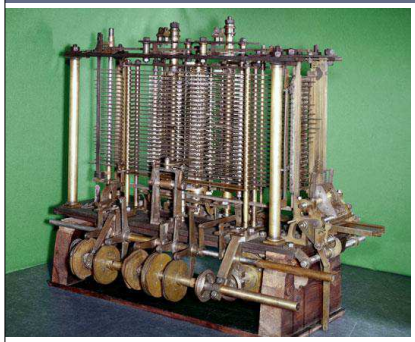
- ระบบคอมพิวเตอร์ประกอบด้วย **Hardware** และ **Software** ทำงานร่วมกัน
- รายละเอียดในการสร้างระบบ (จาก **design**) อาจเปลี่ยนแปลงไปตามยุคสมัย แต่หลักการจะไม่เปลี่ยน
- คอมพิวเตอร์โดยมากมีโครงสร้างของระบบ **Hardware** และ **Software** ที่คล้ายกัน และทำงานได้คล้ายคลึงกัน
- โปรแกรมเมอร์จำเป็นต้องรู้เกี่ยวกับ **Computer System** เพื่อ
  - เข้าใจการทำงานของส่วนต่าง ๆ ที่ประกอบกัน
  - เข้าใจผลกระทบของส่วนต่าง ๆ เหล่านี้ ต่อความถูกต้อง และประสิทธิภาพของโปรแกรมที่เขียน

Early History of Computer System



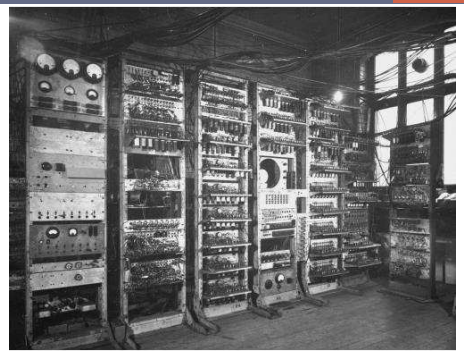
Milestone in Comp Arch.

- **The Zeroth Generation (1642-1945)**
  - **Mechanical Computers: Pascal, Charles Babbage's Machine**
- **The First Generation (1945-1955)** ← **Original Von Neumann Machine**
  - **Vacuum Tubes: ENIAC, EDVAC**
- **The Second Generation (1955-1965)**
  - **Transistors: 16-bit machine Whirlwind I, Minicomputer**
- **The Third Generation (1965-1980)**
  - **Integrated Circuits: IBM System/360 (32-bit Architecture)**
- **The Forth Generation (1980-)**
  - **VLSI : Personal Computer (Intel 8080)**

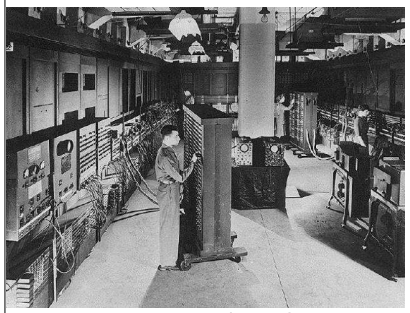


Babbage's analytical engine (1822)

http://www.sciencemuseum.org.uk/images/I032/10303265.aspx



Mark1 (1944)

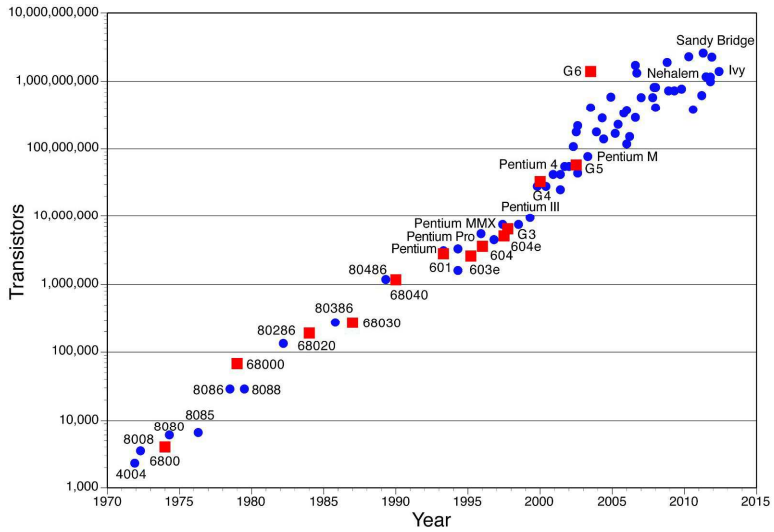


ENIAC (1946)

IBM PC 5150 (1981)

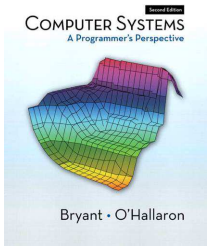
Moore's Law

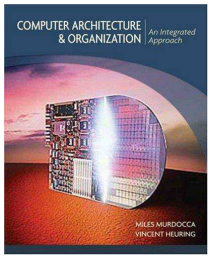
- **The number of transistors (Computing Power) doubling every 18 months.**



Basic Machine Organization

- History of Computer System
- **Hardware Organization**
- Von Neumann Architecture
- Program Representation in Machine-Level



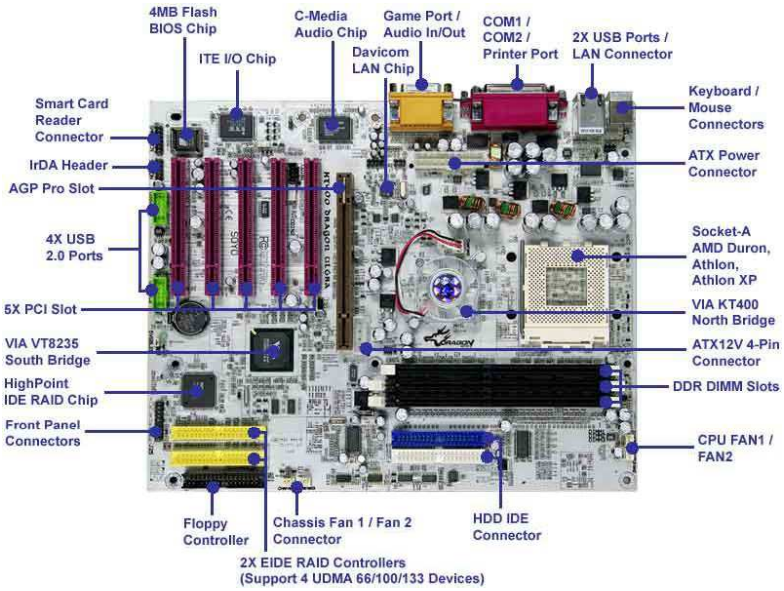


Computer Architecture and Organization: An Integrated Approach, 1st Edition, 2007 - Miles J. Murdocca and Vincent P. Heuring,

Computer Systems: A Programmer's Perspective - Bryant & O'Hallaron

Hardware Organization in a Typical System

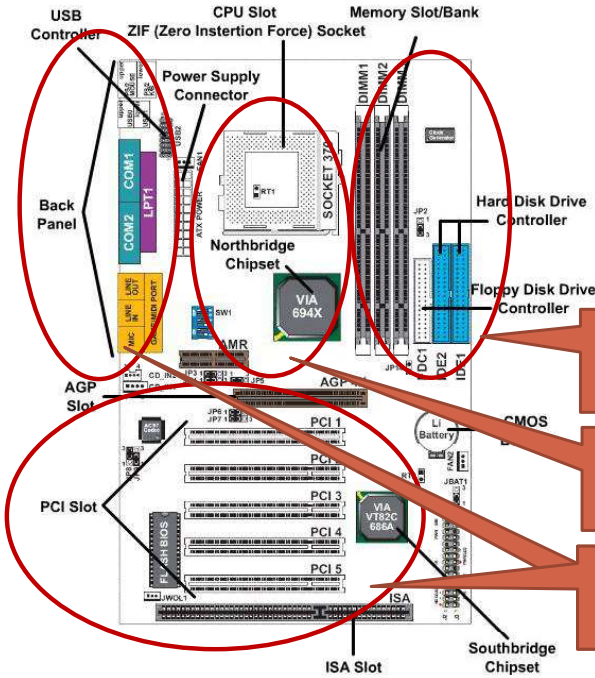
Mainboard/Motherboard



A Sample of AMD Athlon based motherboard.

Image Source:http://www.pchardware.co.uk/siteimages/soyodgondiagram.jpg

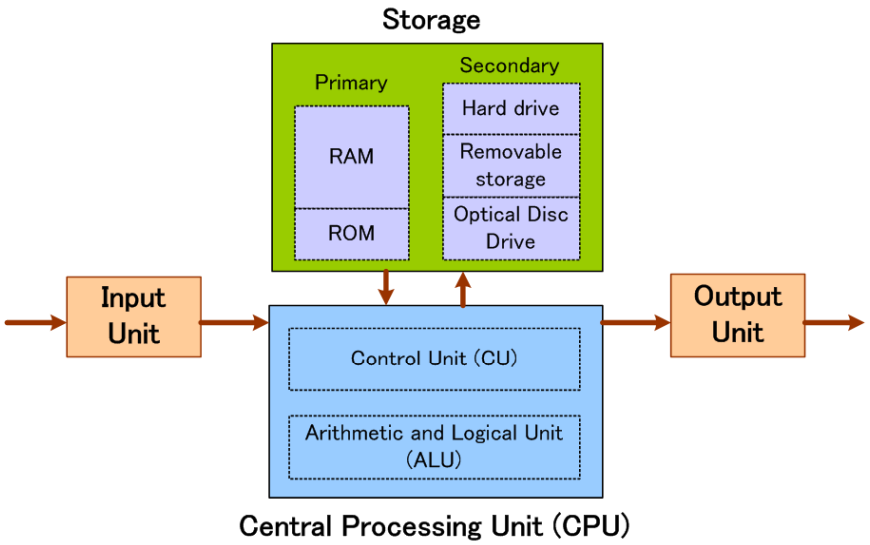
Simplified Motherboard Model*



*Sample of Socket370 CPU Model

Image Source: http://triunfandostereo.org/tag/part-of-motherboard-with-image

Hardware Organization in a Typical System [2]



Hardware Organization in a Typical System [3]

- หน่วยรับข้อมูล (Input Unit)
- หน่วยแสดงผล (Output Unit)
- หน่วยประมวลผลกลาง (Central Processing Unit)
- หน่วยความจำ (Memory, Storage)
- หน่วยส่งสัญญาณข้อมูล(Buses)

Input/Output

- Input เป็นหน่วยรับข้อมูลเข้าสู่ระบบคอมพิวเตอร์
  - Keyboard, Mouse, Barcode reader, OCR, Scanner, Fingerprint Reader, Smartcard Reader, Microphone, etc.
- Output เป็นหน่วยแสดงผลของการประมวลข้อมูลจากคอมพิวเตอร์
  - Monitor, Speaker,Printer, Plotter, etc.

CPU-Central Processing Unit

- ศูนย์กลางการประมวลผลตามชุดคำสั่ง (Instruction set)
- ควบคุมระบบต่าง ๆของคอมพิวเตอร์ให้ทุกหน่วยทำงานสอดคล้องกัน
- ย่อลงบนแผ่นวงจรเล็ก ๆเรียกว่า ชิพ (Chip) หรือ ไมโครโพรเซสเซอร์ (Microprocessor)
  - วงจรรวม (Integrated circuit: IC) บางประเภท อาจประกอบด้วยหลาย CPU ในชิพหนึ่งตัว (เรียก IC ดังกล่าวว่า multi-core processor)

CPU-Central Processing Unit [2]

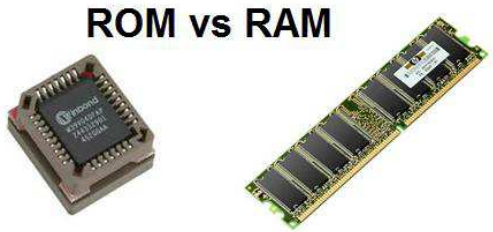
- Control Unit (CU)
  - ทำหน้าที่ควบคุมการทำงานของหน่วยประมวลผล
  - ควบคุมการเคลื่อนย้ายข้อมูลระหว่าง Register, ALU และ Memory
- Arithmetic Logic Unit (ALU)
  - ดึงข้อมูลจาก register เพื่อทำการคำนวณผลหรือเปรียบเทียบ แล้วจึงส่งผลลัพธ์เก็บไว้ใน register
- Registers
  - หน่วยความจำขนาดเล็ก ใช้เก็บผลลัพธ์แบบชั่วคราวจากการคำนวณใน ALU

Registers

- หน่วยเก็บข้อมูลที่มีความเร็วในการเขียน/อ่าน สูงสุด
- ถือเป็นส่วนหนึ่งของ CPU
- รีจีสเตอร์ที่สำคัญได้แก่
  - Accumulator ใช้ทำคำสั่งเกี่ยวกับการคำนวณ
  - Storage Register เก็บข้อมูลและคำสั่งชั่วคราว
  - Address Register เก็บตำแหน่งของข้อมูลและคำสั่งในหน่วยความจำ
  - General Purpose Register ใช้ช่วยในฟังก์ชันต่าง ๆ
  - Floating-point Register ใช้เก็บตัวแปรแบบ floating point
  - Special purpose Register เช่น program counter, stack register

Memory

- Main/Primary Memory เป็นหน่วยความจำหลักของระบบ
  - Read-only memory (ROM): programmed during production
  - Programmable ROM (PROM): can be programmed once
  - Eraseable PROM (EPROM): can be bulk erased (UV, X-Ray)

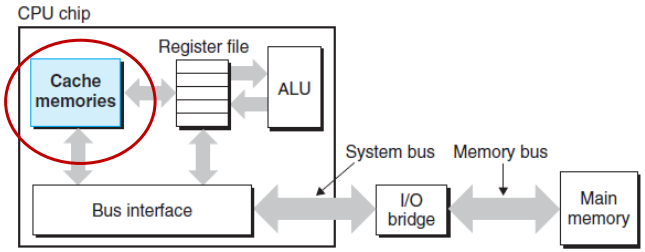


Memory [2]

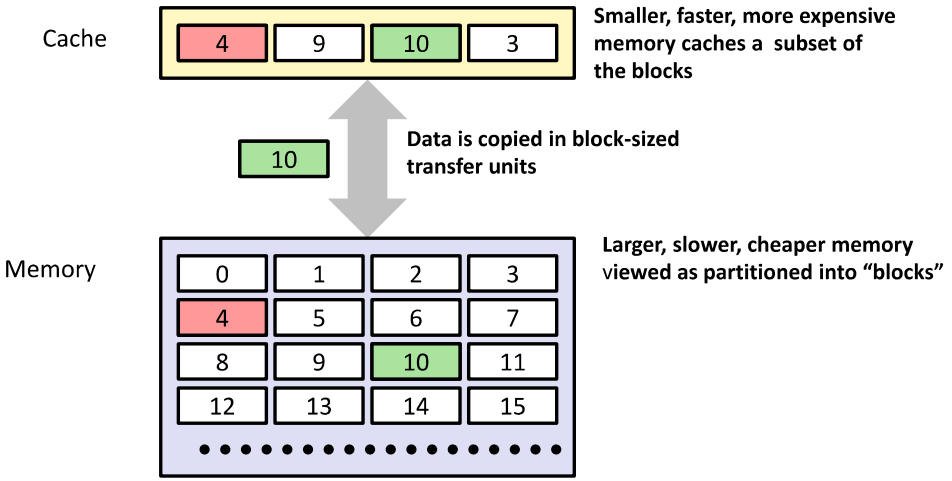
- RAM: Random Access Memory หน่วยความจำหลักเขียนและอ่านข้อมูลได้ เก็บข้อมูลได้เฉพาะเวลาที่มีไฟฟ้าหล่อเลี้ยง (volatile memories)
  - SRAM - Static RAM (Cache)
  - DRAM – Dynamic RAM

Cache

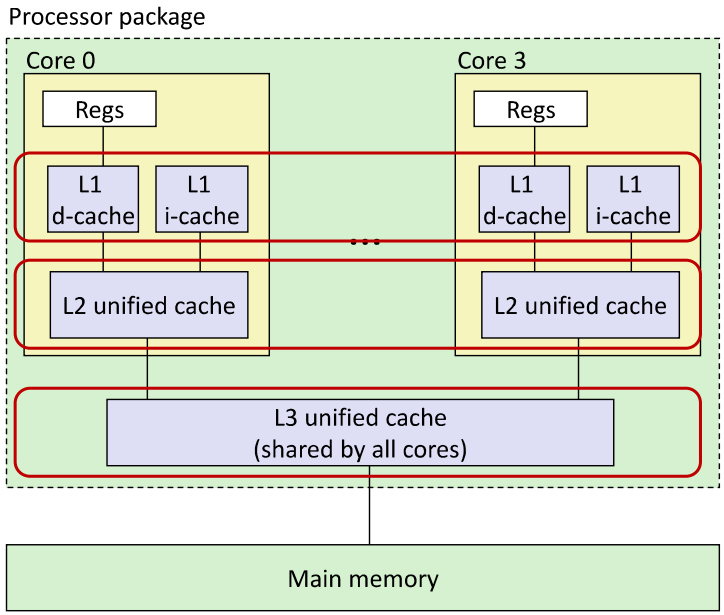
- เป็นหน่วยความจำที่มีความเร็วในการอ่านและเขียนสูง ใช้ทำหน้าที่พักข้อมูล ระหว่าง CPU กับหน่วยความจำหลัก
- ในบางสถาปัตยกรรมจะมีการออกแบบเป็นระดับชั้นของ Cache เช่น L1Cache, L2Cache, L3Cache ตามลำดับความเร็วในการอ่าน/เขียนข้อมูล



General Cache Concept



Intel Core i7 Cache Hierarchy



L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 11 cycles

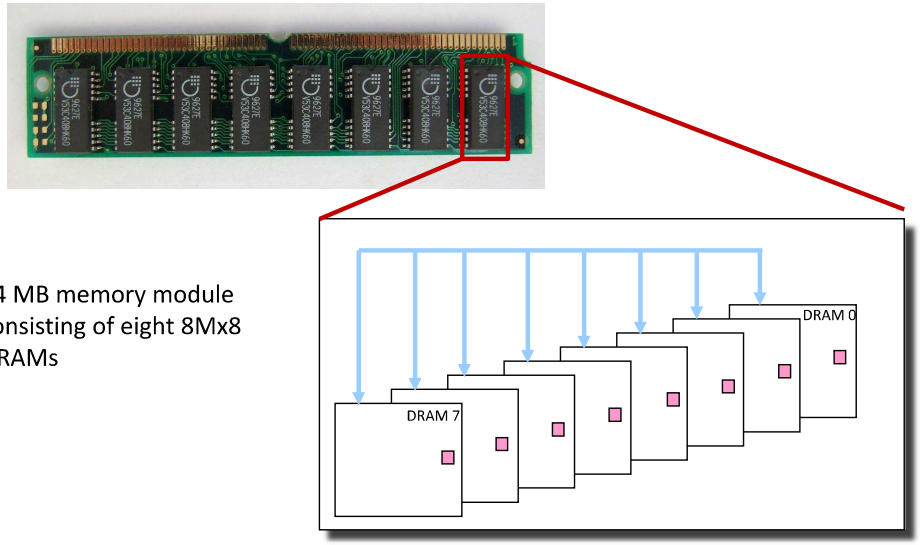
L3 unified cache:
8 MB, 16-way,
Access: 30-40 cycles

Block size: 64 bytes for all caches.

Dynamic Random-Access Memory (DRAM)

- เป็นหน่วยความจำที่อยู่ในรูปแบบชิพเซต
- มีลักษณะการอ้างถึงในรูปแบบ Cell
- จัดเรียงเป็นชิพเซตซ้อนกัน (เรียงแถว)
  - ในการอ้างถึงตำแหน่งใน DRAM จะใช้แบบสองมิติ (rows and columns)
- Value must be refreshed every 10-100 ms

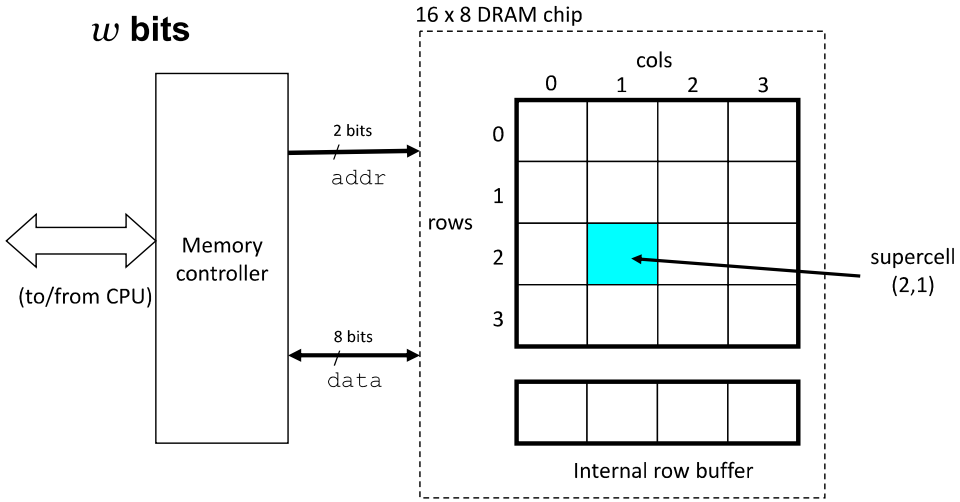
DRAMs Addressing



64 MB memory module consisting of eight 8Mx8 DRAMs

Conventional DRAM Organization

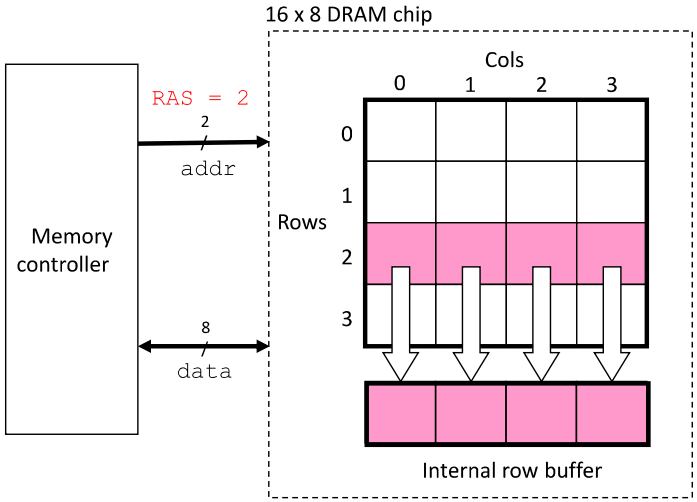
- $d \times w$ **DRAM:**
  - dw **total bits organized as** d **supercells of size** w **bits**



Reading DRAM Supercell (2,1)

Step 1(a): Row access strobe (RAS) selects row 2.

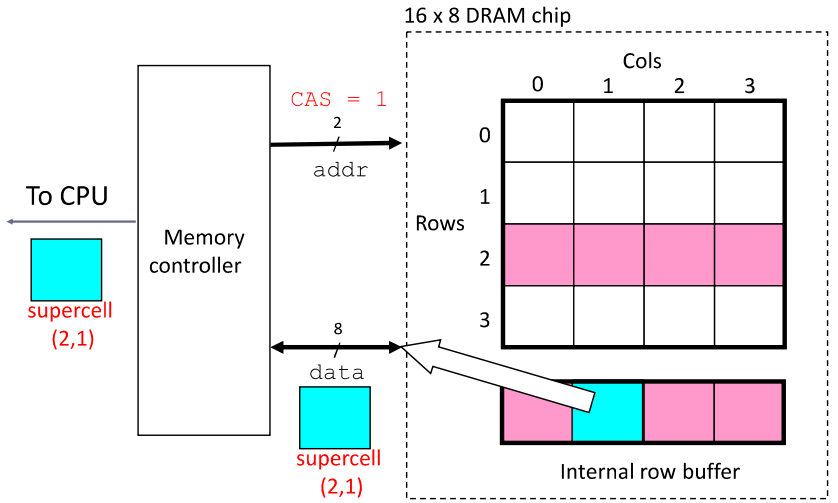
Step 1(b): Row 2 copied from DRAM array to row buffer.



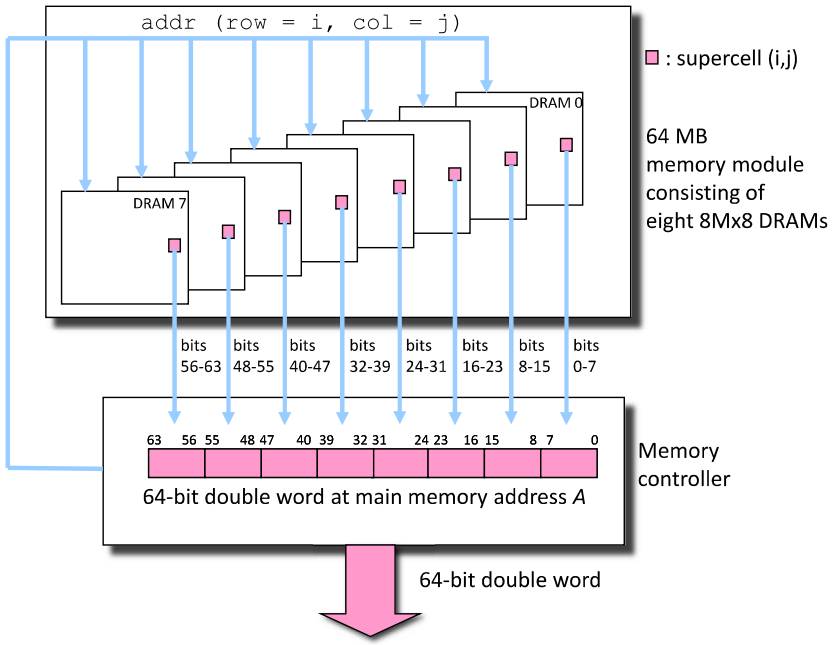
Reading DRAM Supercell (2,1)

Step 2(a): Column access strobe (CAS) selects column 1.

Step 2(b): Supercell (2,1) copied from buffer to data lines, and eventually back to the CPU.



Memory Modules



Secondary Storage

- **Secondary Storage** ใช้เก็บข้อมูลทั่ว ๆไป ข้อมูลไม่หายเมื่อไม่มีไฟเลี้ยง(nonvolatile) แบ่งตามลักษณะของวิธีการเก็บข้อมูลได้ดังนี้
  - **Solid State**
  - **Magnetic**
  - **Optical**

- **Virtual Memory** เป็นการใช้พื้นที่ของ **Secondary Storage** มาเป็นส่วนขยายของหน่วยความจำหลักของระบบ

What's Inside A Disk Drive?

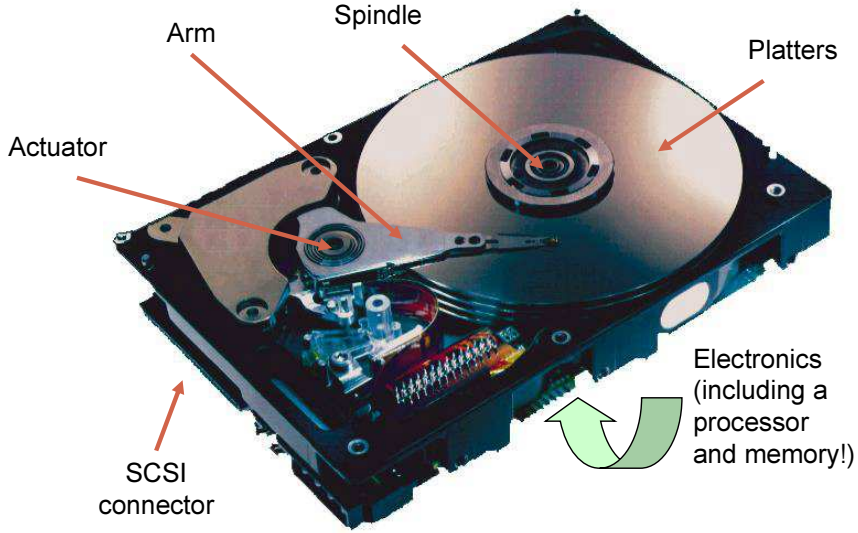
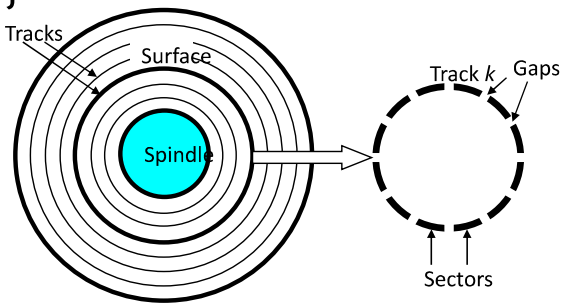


Image courtesy of Seagate Technology

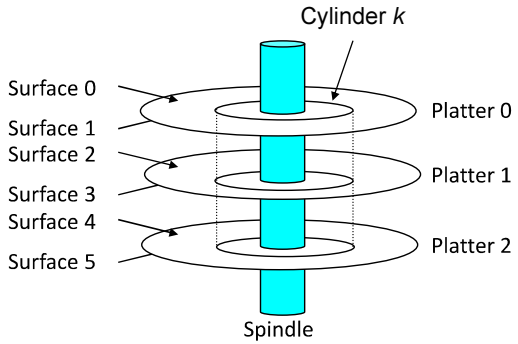
Disk Geometry

- **Disks** ประกอบด้วยแผ่น **Platter** โดยจะบันทึกได้สองด้าน **(Surface)**
- แต่ละด้านจะมีวงแหวนเรียงตัวอยู่เรียกว่า **Track**
- และในแต่ละ **Track** จะมีหน่วยย่อยสุดเรียกว่า **Sector** เป็นส่วนเก็บข้อมูลเป็นส่วน ๆ



Disk Geometry (Muliple-Platter View)

- **Aligned tracks form a cylinder.**



Capacity = (# bytes/sector) x (avg. # sectors/track) x (# tracks/surface) x (# surfaces/platter) x (# platters/disk)

Computing Disk Capacity

Example:

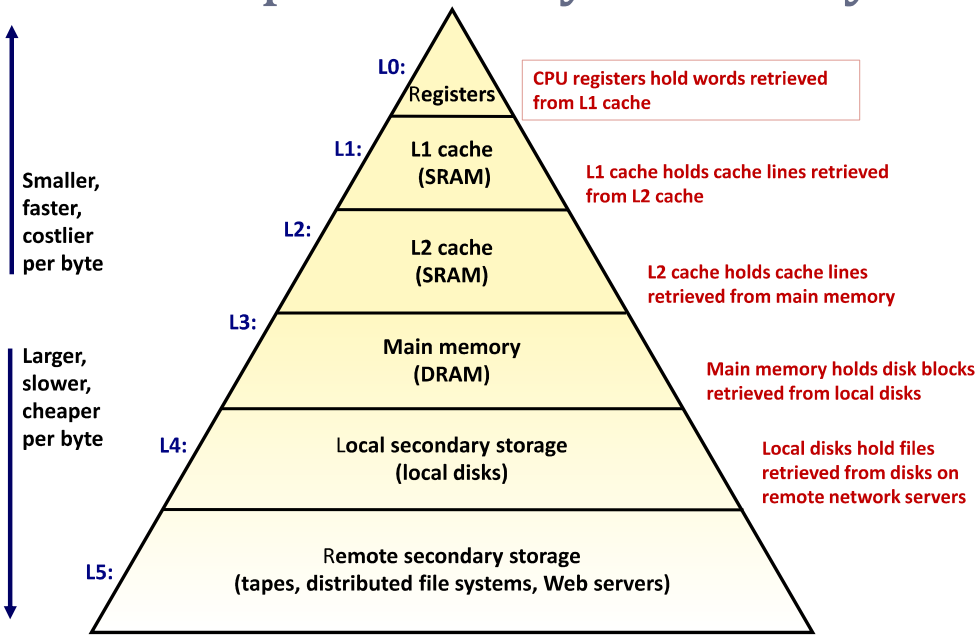
- **512 bytes/sector**
- **300 sectors/track (on average)**
- **20,000 tracks/surface**
- **2 surfaces/platter**
- **5 platters/disk**

Capacity = 512 x 300 x 20000 x 2 x 5

= 30,720,000,000

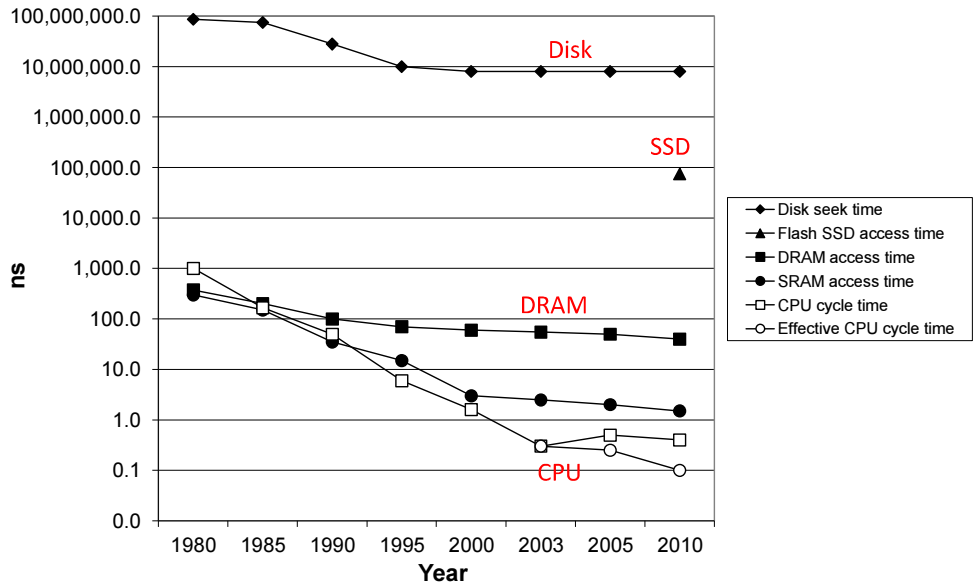
= 30.72 GB

An Example Memory Hierarchy



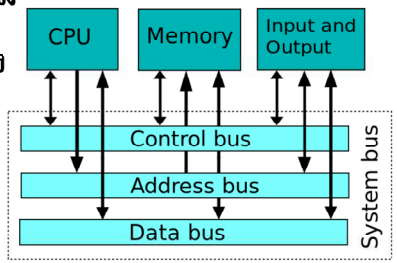
The CPU-Memory Gap

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



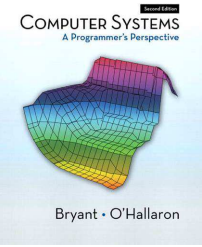
System Buses

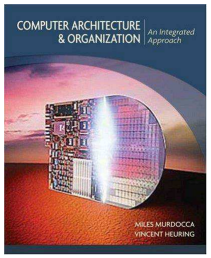
- เป็นตัวเชื่อมระหว่างส่วนประกอบต่าง ๆที่ทำงานร่วมกันในระบบ ใช้เพื่อส่งข้อมูลในรูปแบบ **Fixed-size chunk** ในหน่วย **word (1 word = 4 bytes (32bit) or 8 bytes (64bit))**
- **System Buses** ประกอบด้วย
  - **Control Bus** ใช้ส่งสัญญาณควบคุม
  - **Address Bus** ใช้ส่งข้อมูลตำแหน่ง
  - ในหน่วยความจำ
  - **Data Bus** ใช้ส่งข้อมูลต่าง ๆ



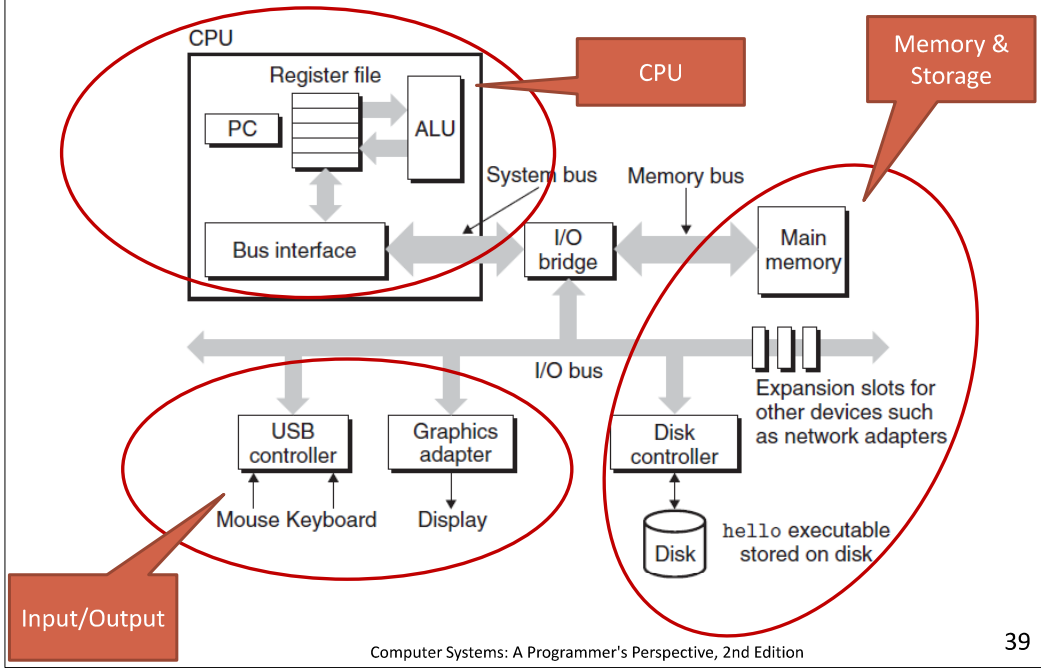
Basic Machine Organization

- History of Computer System
- Hardware Organization
- **Von Neumann Architecture**
- Program Representation in Machine-Level



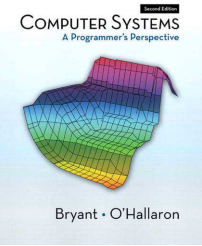


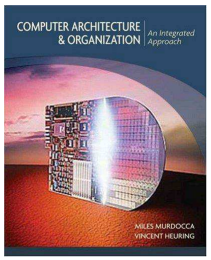
Von Neumann Architecture



Basic Machine Organization

- History of Computer System
- Hardware Organization
- Von Neumann Architecture
- **Program Representation in Machine-Level**



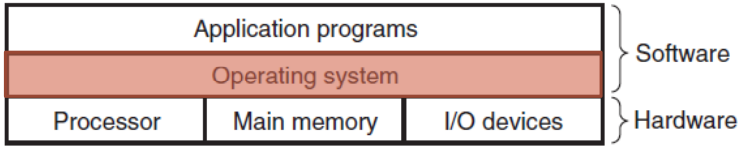


Machine-Level Representation of Programs

- โปรแกรมที่เขียนขึ้น จะมีการแปลงอยู่ในรูป machine code โดย compiler และ assembler
- machine code เป็นโค้ดที่รวมทั้งลำดับการทำงานของโปรแกรม และคำสั่งเพิ่มเติมในการติดต่อส่วนต่าง ๆของคอมพิวเตอร์ เพื่อให้โปรแกรมทำงานได้
- machine code จะมีความแตกต่างกันตาม Instruction Set Architecture (ISA) (32 bit, 64 bit)
- Programmer จำเป็นต้องเข้าใจหลักการของ machine code ในแง่ของประสิทธิภาพในการเขียนโปรแกรม

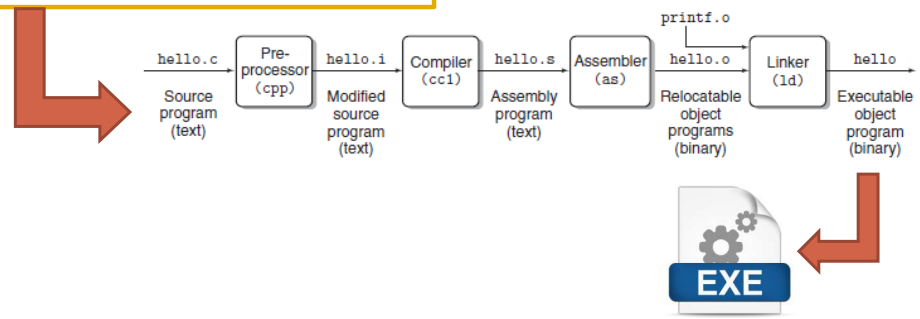
Runtime Environment

- ในการรันโปรแกรม hello.c ตัวโปรแกรมไม่ได้เป็นผู้ติดต่อ CPU และ Memory โดยตรง
- Operating System หรือ OS ทำหน้าที่เป็นตัวกลางระหว่าง Software และ Hardware
- executable program (หรือ machine code) จะต้องมีรูปแบบการเก็บข้อมูลความสอดคล้องกับสถาปัตยกรรมของ OS ที่รัน (32bit, 64 bit)



Compiling "hello.c" (Revisited)

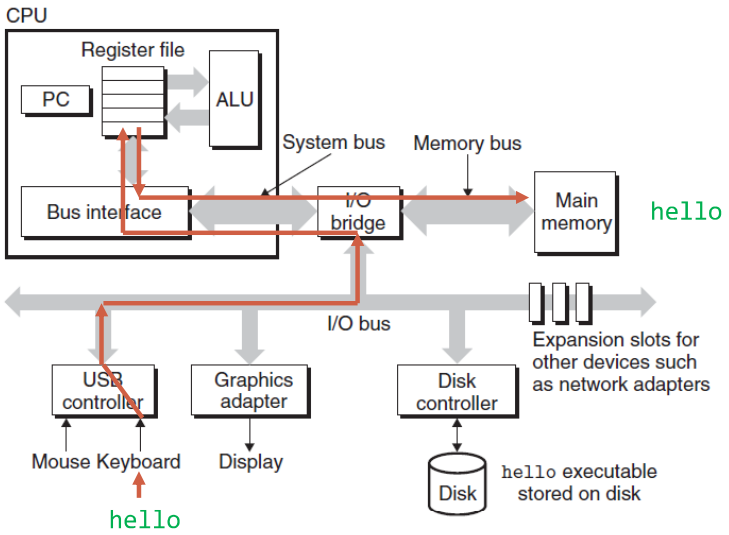
```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    printf("Hello world!\n");
    return 0;
}
```



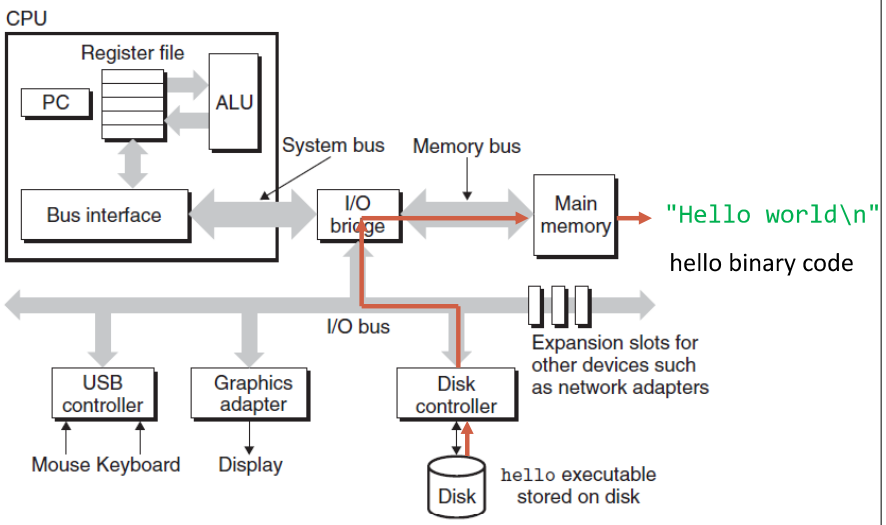
Running "hello.c"

1. Reading the hello command from the keyboard.
2. Loading the executable from disk into main memory
3. Writing the output string from memory to the display.

1. Read "hello" Command from Keyboard



2. Load "hello" from Disk to Memory



3. Writing "Hello World" on Screen