FU05 Computer Architecture

1. Introduction (概要)

Ben Abdallah Abderazek

E-mail: benab@u-aizu.ac.jp

Lectures of this course are adopted from the slides of the Book ''Computer Organization and Design - The Hardware/Software Interface, 5th Ed'' by David. A. Patterson & John L. Hennessy

Course Objectives

- Understand the main principles of computer architecture.
- Learn the fundamental of assembly programming.
- Use CAD tools and a hardware description language to design and simulate a simple MIPS processor.

Course Contents

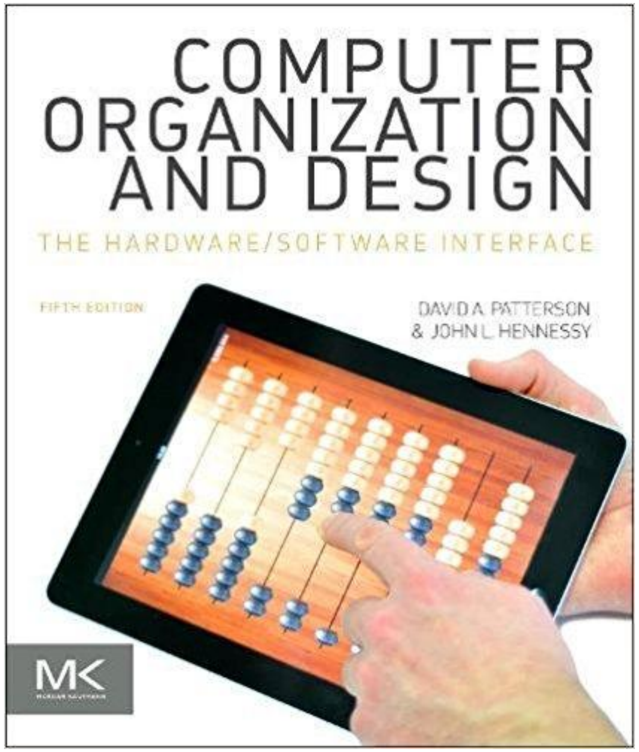
- Lecture 1 Introduction
- Lecture 2 Instruction set & Assembly Language I
- Lecture 3 Instruction set & Assembly Language II
- Lecture 4 Performance Evaluation
- Lecture 5 Arithmetic for Computer I
- Lecture 6 Arithmetic for Computer II
- Lecture 7 Datapath
- Lecture 8 Controlpath
- Lecture 9 Pipeline I
- Lecture 10 Pipeline II
- Lecture 11 Memory Hierarchy: Cache
- Lecture 12 Memory Hierarchy: Virtual Memory
- Lecture 13 Storage & other I/Os
- Lecture 14 Parallel Processor

Textbook(s)

- Computer Organization and Design MIPS Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design) Paperback – December 4, 2020
- Computer Organization and Design - The Hardware/Software Interface. David. A. Patterson and John L. Hennessy, 5th edition, Morgan Kaufmann Publishers, ISBN 0124077269

English Version

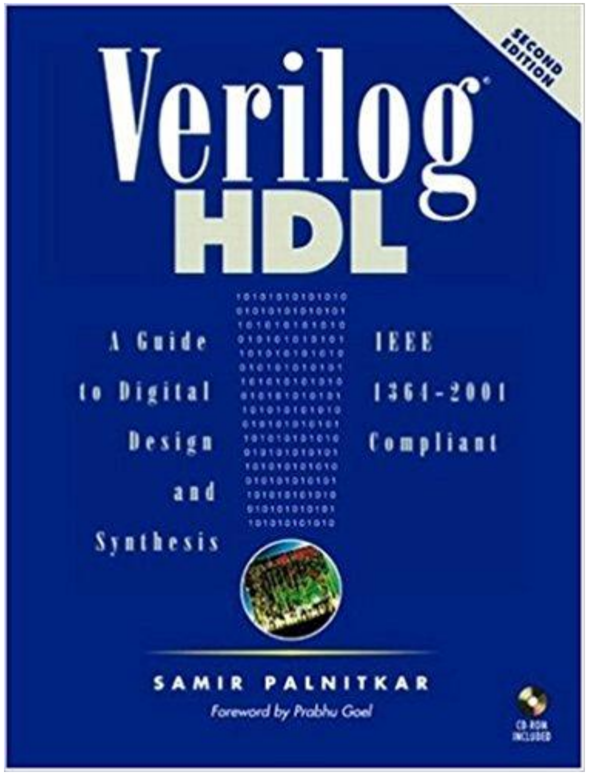
Japanese Version


COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
FIFTH EDITION
DAVID A. PATTERSON
& JOHN L. HENNESSY
MK

パターソン&ヘネシー
コンピュータの構成と設計
ハードウエアとソフトウエアのインタフェース
第5版
上
世界最高のコンピュータ教科書が
モバイル、クラウド対応で全面刷新

パターソン&ヘネシー
コンピュータの構成と設計
ハードウエアとソフトウエアのインタフェース
第5版
下
世界最高のコンピュータ教科書が
モバイル、クラウド対応で全面刷新

Reference (Verilog HDL)



Verilog HDL (2nd Edition) 2nd Edition
by Samir Palnitkar (Author)

ISBN-13: 978-0130449115
ISBN-10: 0130449113

HDLによるVLSI設計―VerilogHDLと
VHDLによるCPU設計 第２版
深山 正幸，他 著
出版社: 共立出版

ISBN-13: 978-4320120273 ; (2002/01)

Course Website & Evaluation Method

- **Course Website**

http://www.u-aizu.ac.jp/~benab/classes/ca/

- **Regulation**
  - Every student is required to print out handouts before coming to the classroom.
  - If a student is late for more than 30 minutes, he/she cannot attend the lecture of the day.
- **Evaluation Method**
  - Final examination (50%), Exercise reports (50%).
  - No makeup examination.

The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore’s Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive (everywhere)

Classes of Computers

- Personal computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff

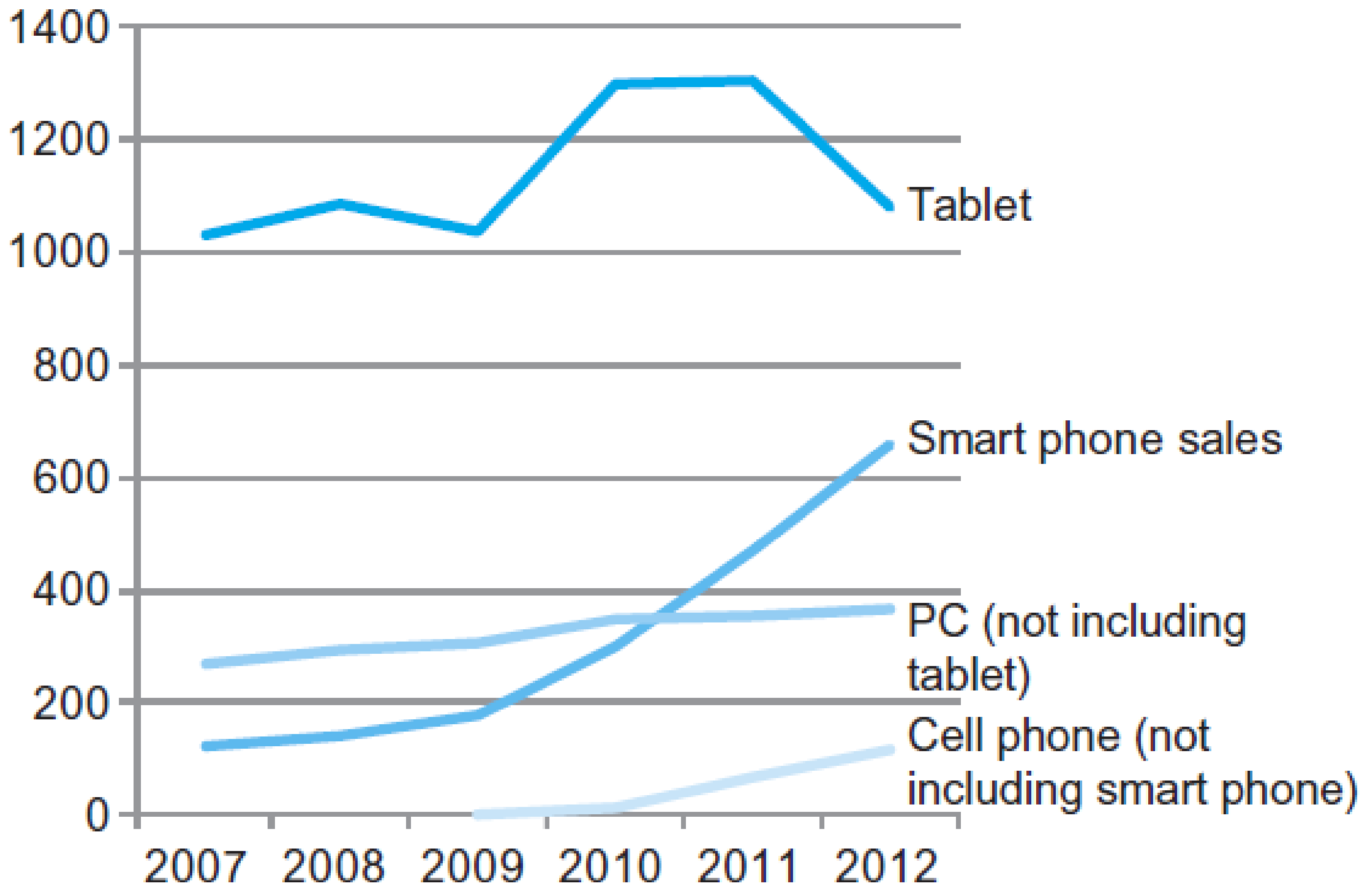
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

Classes of Computers

- 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

The PostPC Era



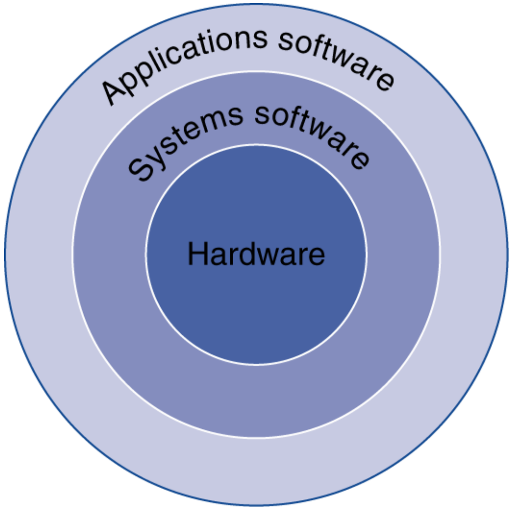
What You Will Learn

- 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

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

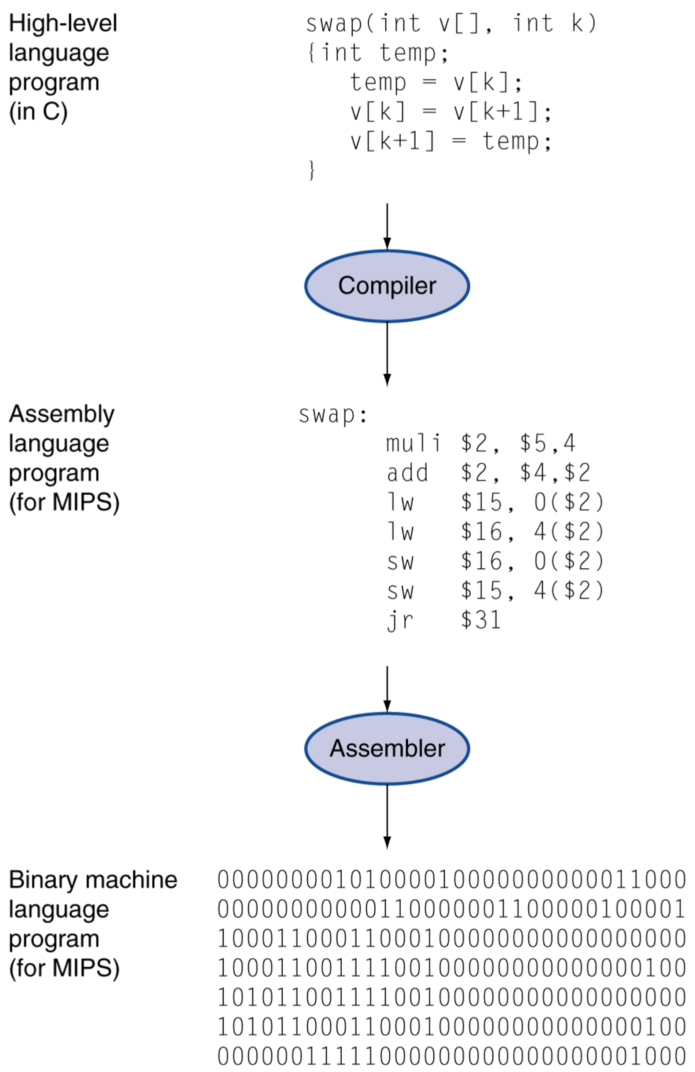
Below Your Program



- 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

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



High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Assembly language program (for MIPS)

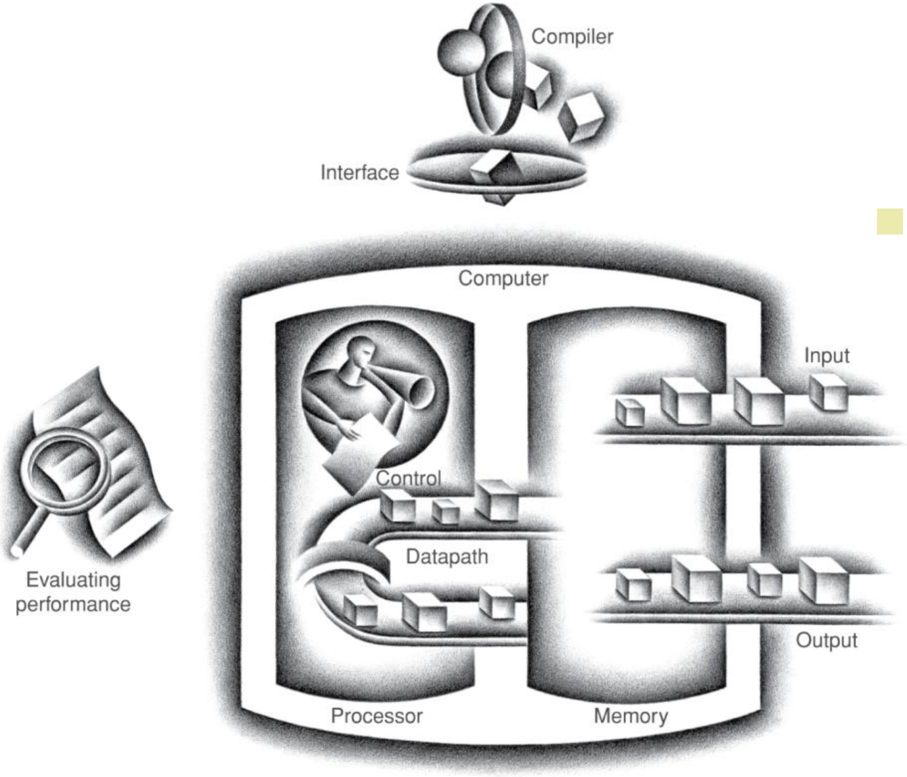
```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Components of a Computer

The BIG Picture



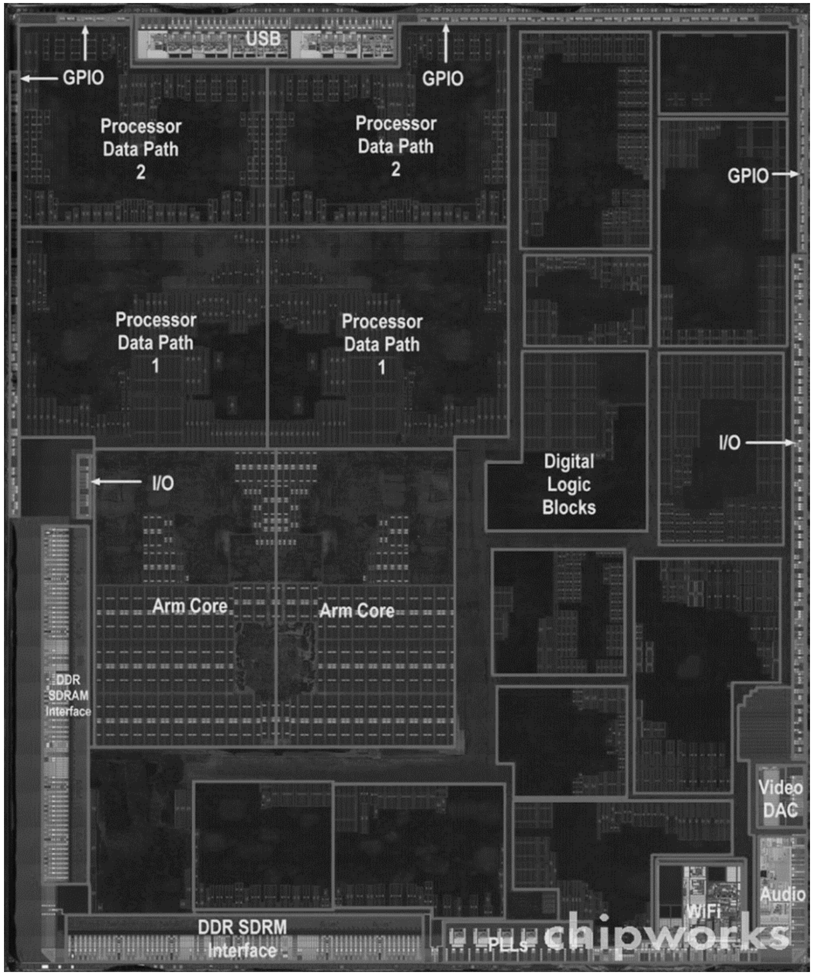
- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
    - Small fast SRAM memory for immediate access to data

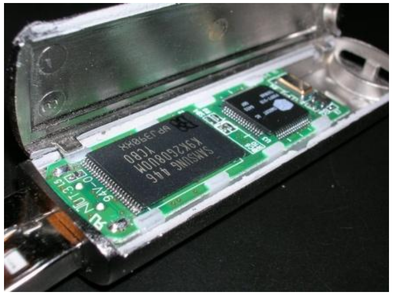
Inside the Processor

- Apple A5



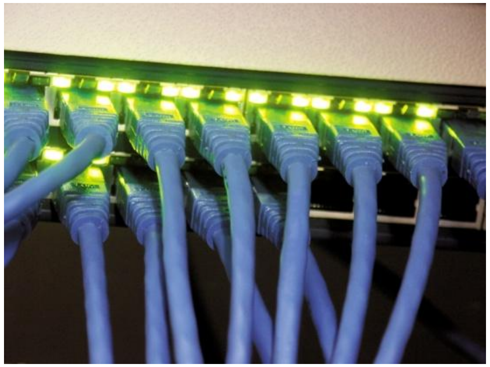
A Safe Place for Data

- Volatile main memory
  - Loses instructions and data when power off
- Non-volatile secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)



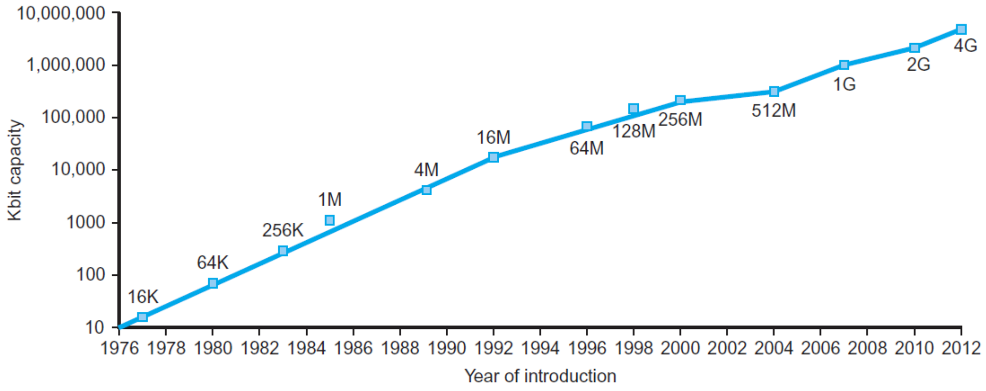
Networks

- Communication, resource sharing, nonlocal access
- Local area network (LAN): Ethernet
- Wide area network (WAN): the Internet
- Wireless network: WiFi, Bluetooth



Technology Trends

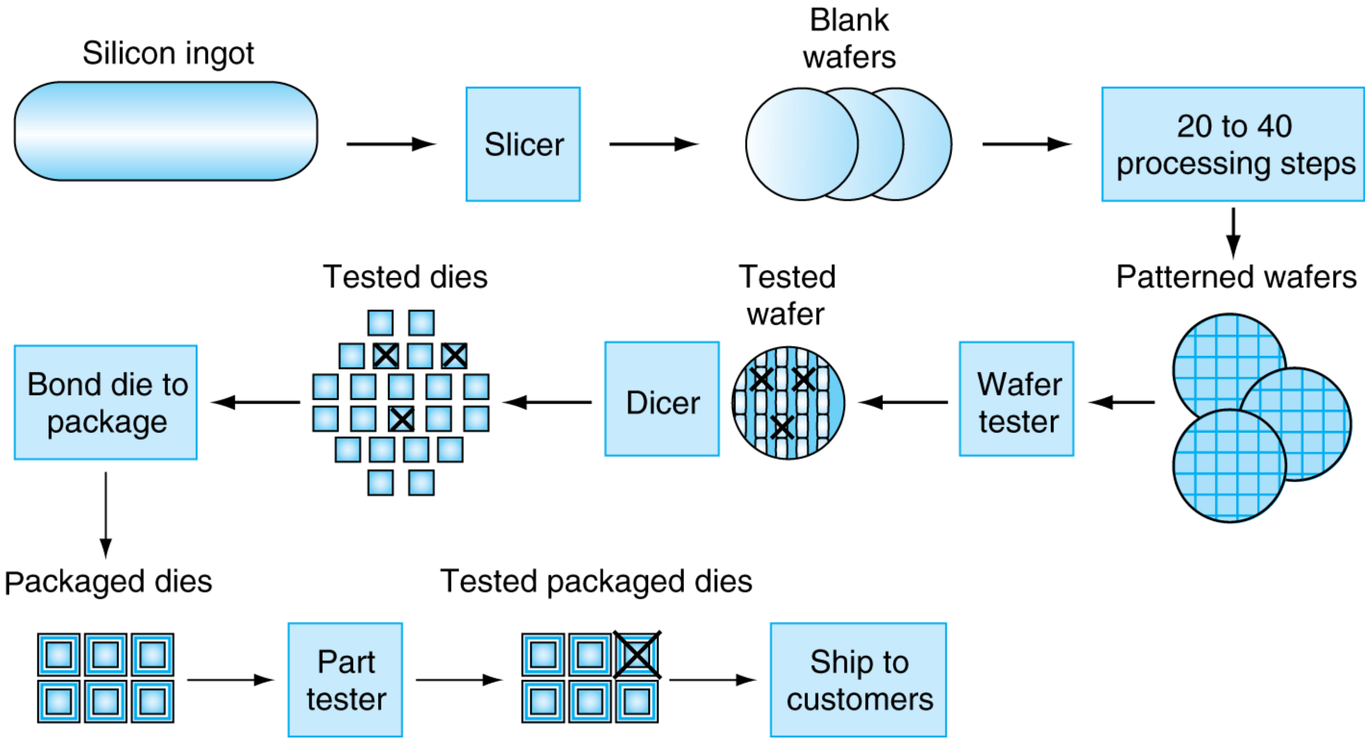
- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost



DRAM capacity

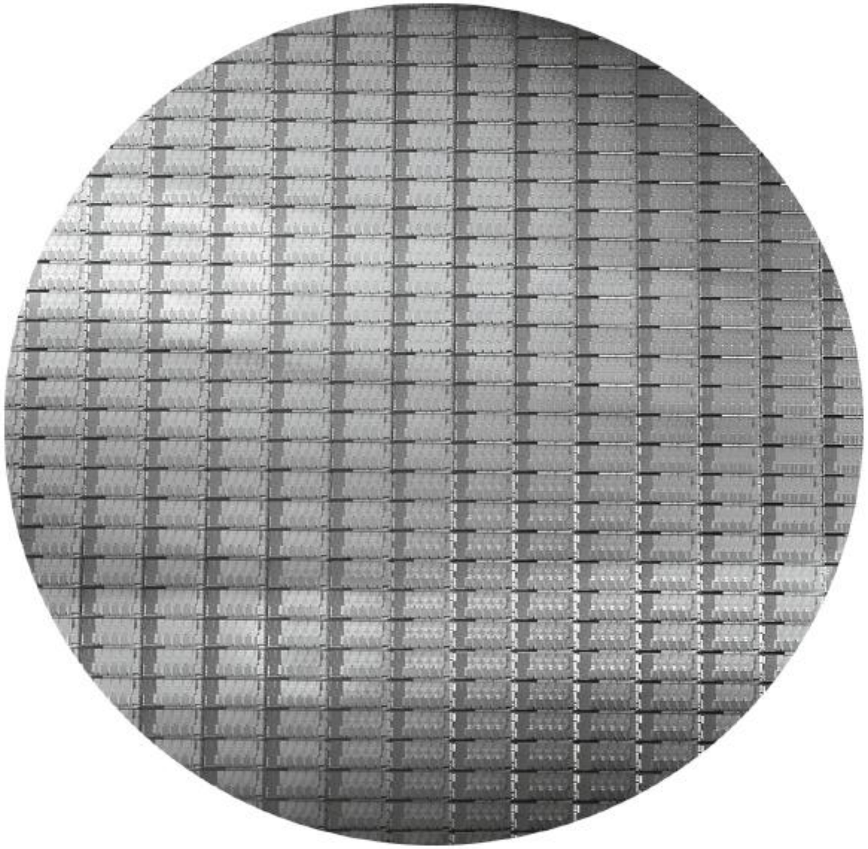
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

Manufacturing ICs



- Yield: proportion of working dies per wafer

Intel Core i7 Wafer



- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area} \big/ \text{Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area/2}))^2}$$

- Nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

Typical Instruction Execution Stages

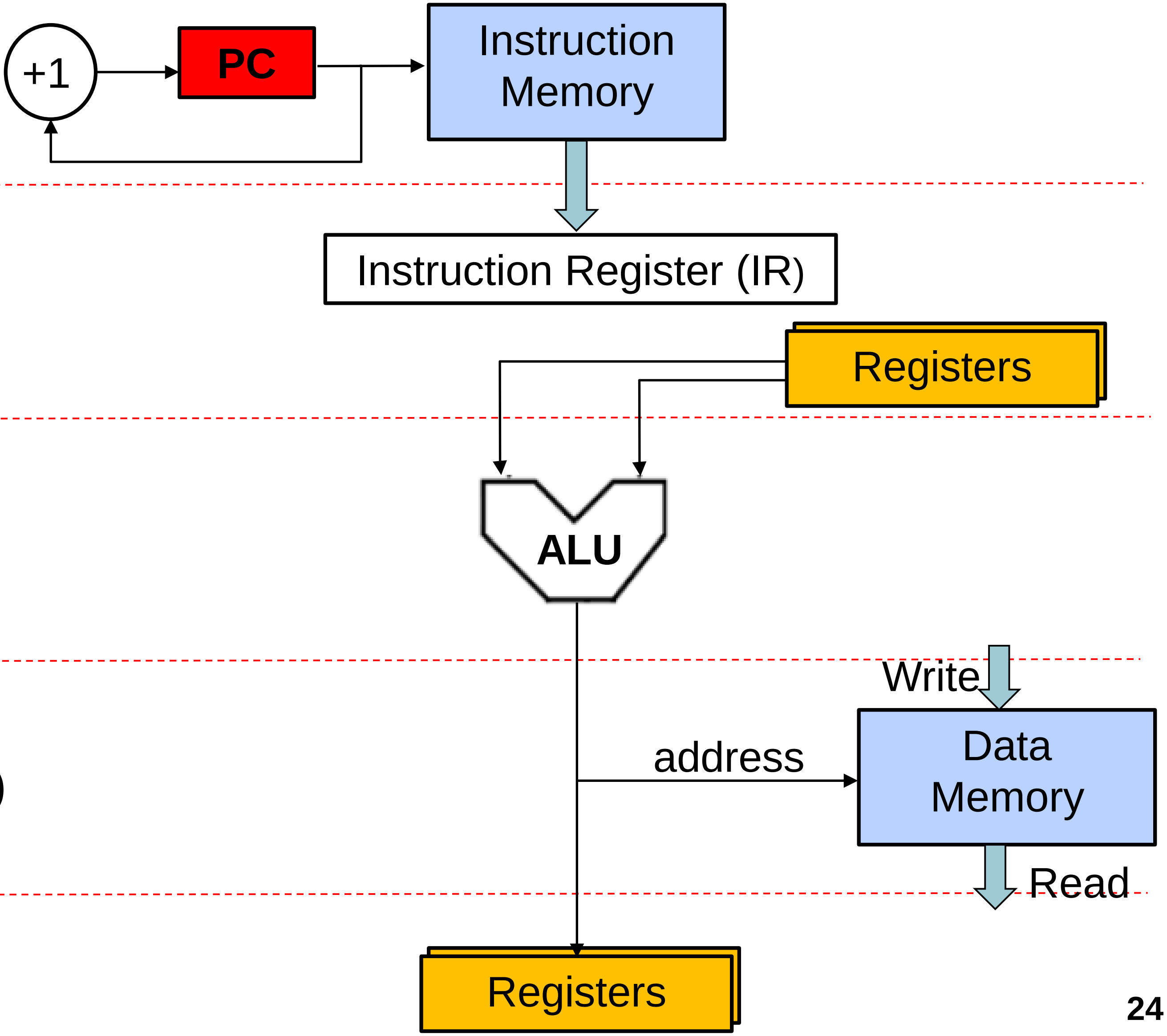
Instruction Fetch (IF)

Instruction Decode (ID)

Execution (EX)

Memory Access (MEM)

Write-Back (WB)



+1

PC

Instruction Memory

Instruction Register (IR)

Registers

ALU

Write

address

Data Memory

Read

Registers

Instruction Fetch (IF)

- The instruction Fetch (IF) stage is responsible for obtaining the requested instruction from the instruction memory.
- The instruction and the PC (program counter), which is incremented to the next instruction, are stored in the IF/ID pipeline register as temporary storage so that may be used in the next stage at the start of the next clock cycle.

Instruction Decode

- The Instruction Decode (ID) stage is responsible for decoding the instruction and sending out the various control lines to the other parts of the processor.
- The instruction is sent to the control unit where it is decoded and the registers are fetched from the register file.

Execute (EX)

- The Execution (EX) stage is where any calculations are performed.
- The main component in this stage is the ALU. The ALU is made up of arithmetic, logic and capabilities.

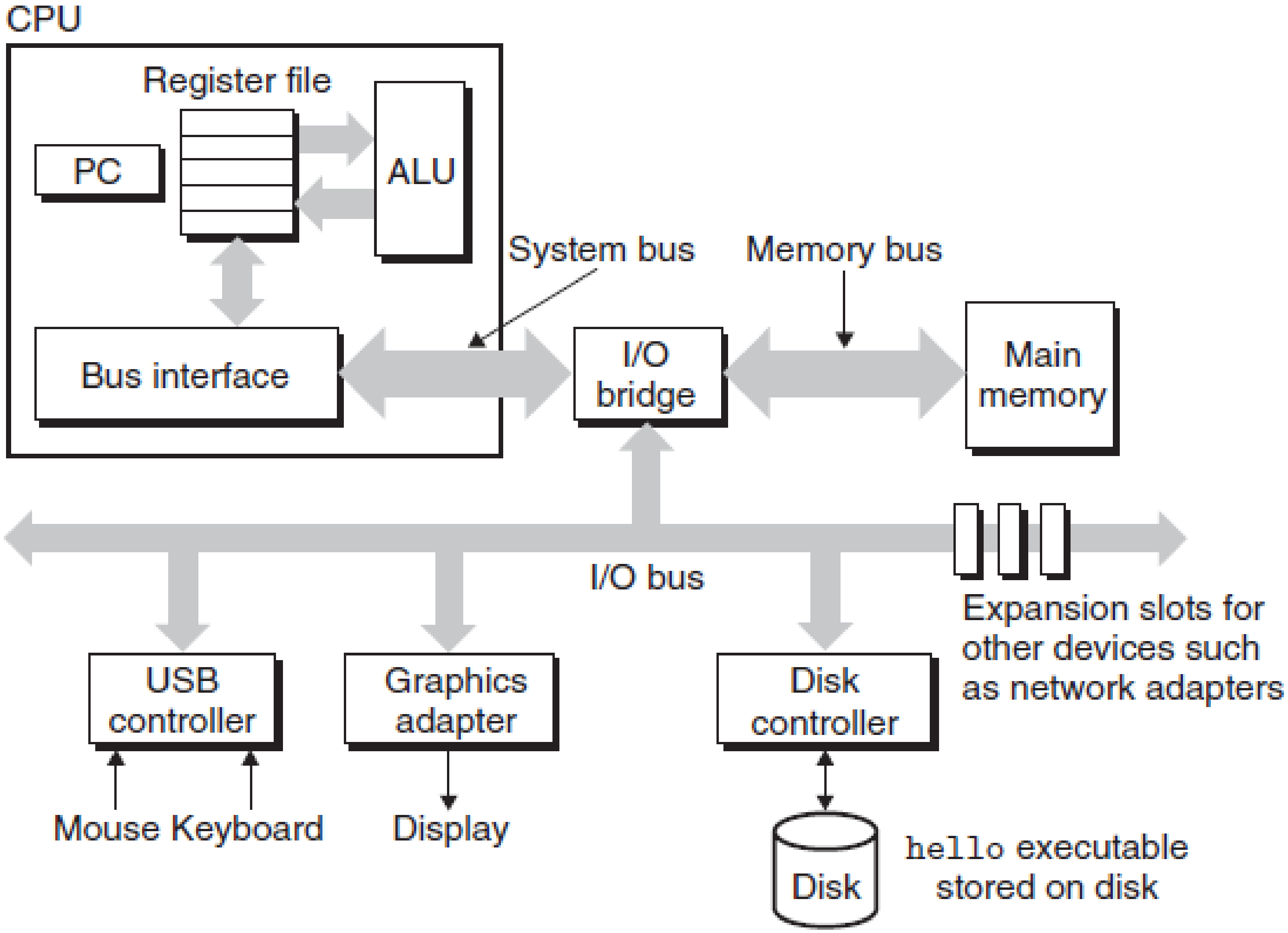
Memory Access (MEM)

- The Memory and IO (MEM) stage is responsible for storing and loading values to and from memory.
- It also responsible for input or output from the processor. If the current instruction is not of Memory or IO type than the result from the ALU is passed through to the write back stage.

Write Back (WB)

- The Write Back (WB) stage is responsible for writing the result of a calculation, memory access or input into the register file.

Hardware Organization of a Typical Computer System



Exercises

Ex1. Aside from the smart cell phones used by billion people, list and describe four other types of computers.

Ex2. Describe the steps that transform a program written in a high-level language such as C into a representation that is directly executed by a computer processor.

Ex3. List the main components of a CPU.

Ex4. List and briefly describe the main components of a computer system.

Solutions

Ex1.

- **Personal computer (includes workstation and laptop):** Personal computers emphasize delivery of good performance to single users at low cost and usually execute third-party soft ware.
- **Personal mobile device (PMD, includes tablets):** PMDs are battery operated with wireless connectivity to the Internet and typically cost hundreds of dollars, and, like PCs, users can download soft ware (“apps”) to run on them. Unlike PCs, they no longer have a keyboard and mouse, and are more likely to rely on a touch-sensitive screen or even speech input.
- **Server:** Computer used to run large problems and usually accessed via a network.
- **Warehouse scale computer:** Thousands of processors forming a large cluster.
- **Supercomputer:** Computer composed of hundreds to thousands of processors and terabytes of memory.
- **Embedded computer:** Computer designed to run one application or one set of related applications and integrated into a single system.

Solutions

Ex.2.

- The program is compiled into an assembly language program, which is then assembled into a machine language program.