The Principle of Computer System

Hardware/Software interface

楼学庆

浙江大学计算机学院

http://10.214.47.99/

Email:hzlou@163.com

玉泉校区曹光彪东楼507室

Principles of Computer Systems

Hardware/Software interface

楼学庆

浙江大学计算机学院

http://10.214.47.99/

Email:hzlou@163.com

玉泉校区曹光彪东楼507室

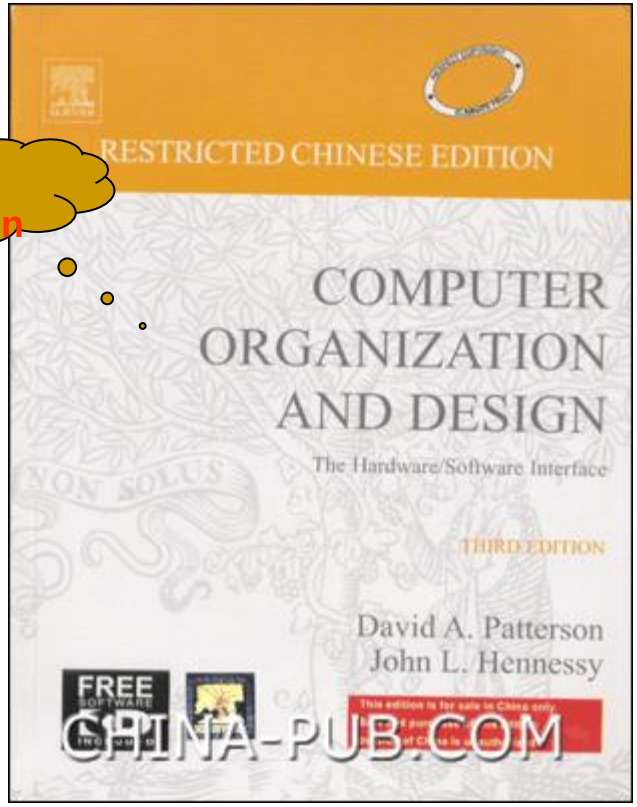
联系方式

- 网站：
  - http://10.214.47.99
- 邮箱：
  - hzlou@163.com(不收作业)

Textbook

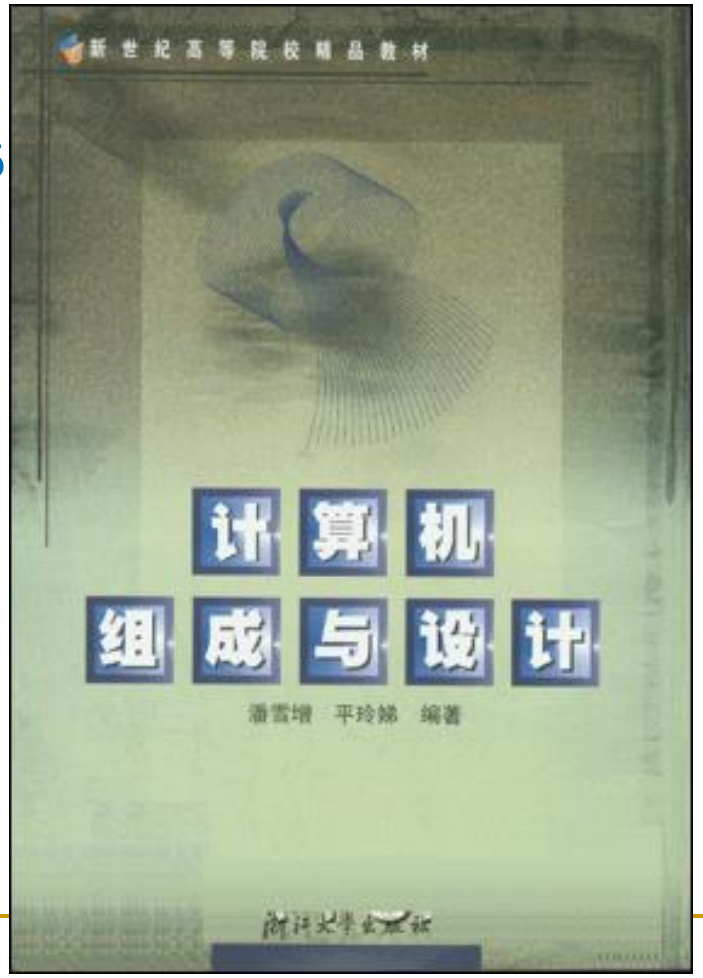
2nd Edition

3rd Edition



Reference

- 《计算机组成与设计》
  - 潘雪增、平玲娣
  - 浙江大学出版社
  - ISBN 7-308-03523-9/TP.25



Reference

- 《计算机组成和设计》硬件 / 软件接口（第2版）
  - 作者：郑纬民译
  - ISBN：9787302069010
  - 定价：76元

Course outline

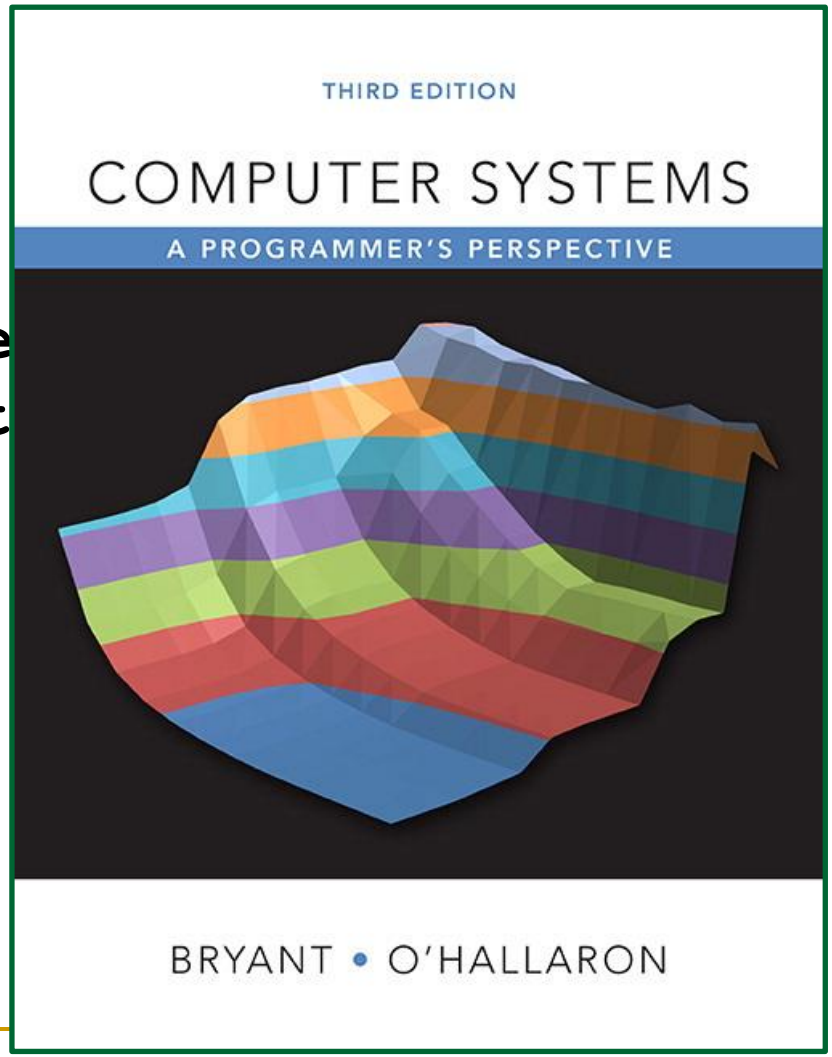
- **Name:**

  Computer Systems

- **Students:**

  Undergraduate stude
  some others department

  - Score : 4.5
  - Hours/week: 3.5-2
  - Total: 88 hours



教学内容研究性

- 没有必然，只有更好
  - 尽力寻求、实现各种可能的方法
  - 比较、评价各种不同的方法 (现行方法的优点、特点)
  - 相对完整实现所选择的方法
- 理论与实际相结合
  - 理论学习
  - 软件模拟 (C语言编程)
  - 硬件实验 (FPGA设计)

成绩（精研班）

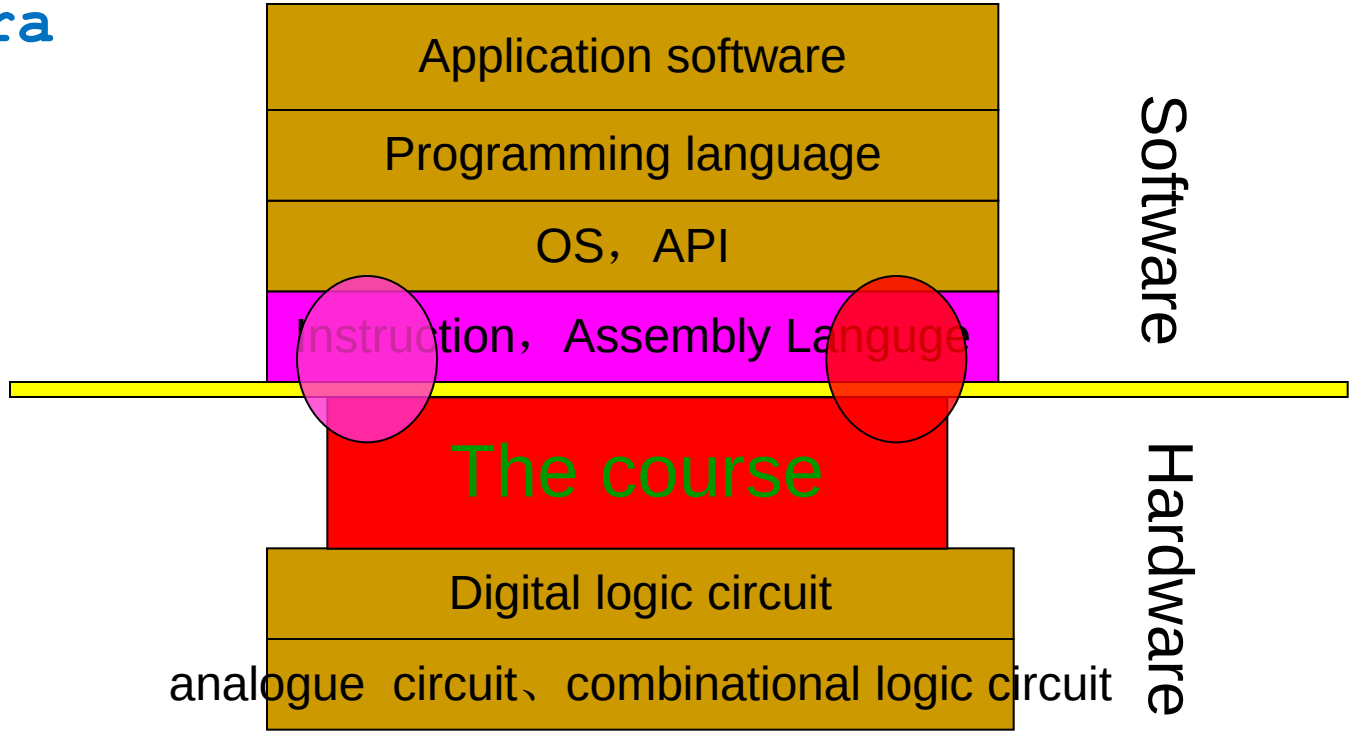
- 考试：30%(不低于30分)
  - 英文
  - 闭卷
- 平时，五部分14知识点：
  - 每个知识点：5%，总共70%(雷同**0**分)
- 额外：
  - 课堂练习(+5%)：每题1分
- QQ：25534460

小班化教学

- 分组学习制
  - 四人一组组长负责
  - 定期汇报
  - 随时交流
- 按知识点教学，每个知识点：
  - 课堂讲解(多媒体)
  - 讨论、布置练习
  - 演示、讲解
  - 相互评价

The Course

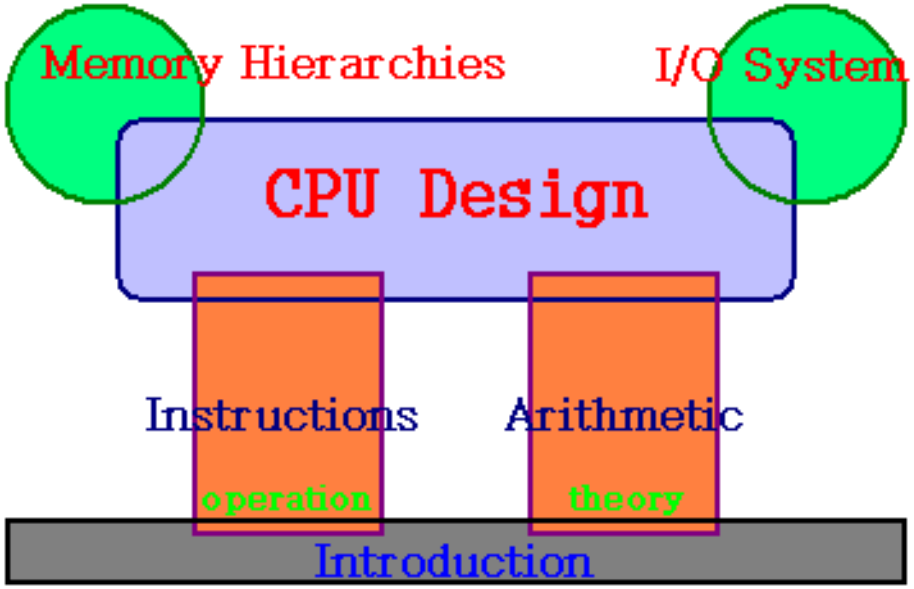
- **Prerequisites**
  - C Program Language
  - Digital Logic
  - Algebra



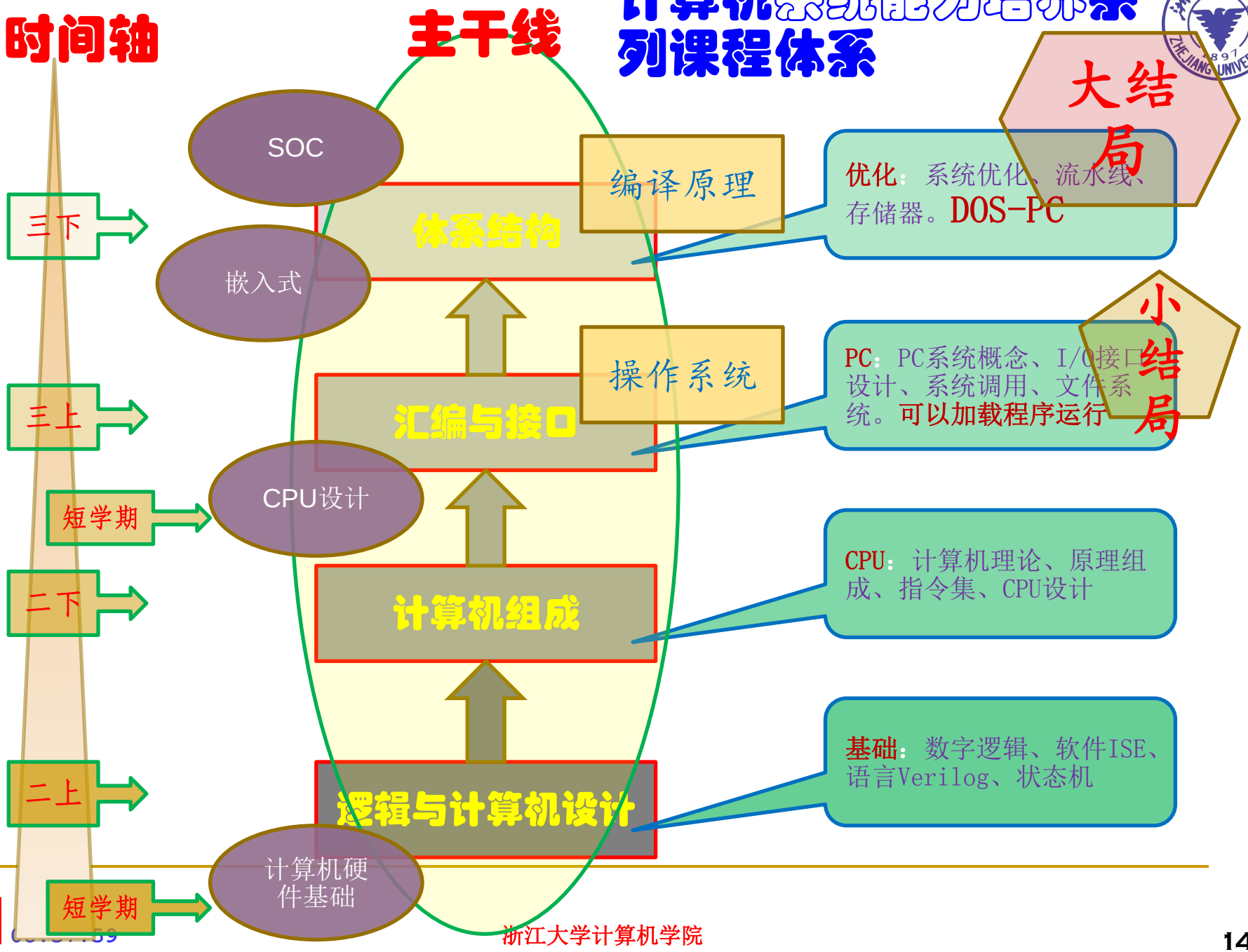
《计算机系统原理》课程目标

- 利用一个学期，覆盖《逻辑与计算机设计》、《计算机组成》、《体系结构》与《汇编与接口》等计算机硬件系统系列主要课程。作为非计算机技术方向，学习了解掌握计算机硬件、计算机系统方面知识的主要课程。
  - ☆前导课程：《C语言程序设计》
- 强烈建议不是只想玩软件的同学，改选 《计算机组成》！

课程结构


Memory Hierarchies
I/O System
CPU Design
Instructions
Arithmetic
operation
theory
Introduction

计算机系统能力培养系列课程体系



我有一壶酒
足以慰风尘
计组一堂课
汇编天下人

你我相约混一年，约吗？

课程计划

- 每学期提供课程计划，
- 确定计算机系统设计的总目标。
- 给出实现的路线图。

ZPC 设计路线图之春夏时节

*: 春夏《计算机组成》以设计运算器 ALU、CPU 为主。

周	章节	内容	理论		实验		说明	周
01	概论	吹牛	作 01	汉字键盘设计	实 01	运算器设计	1. 课堂实验当堂布置当堂完成，每次实验需画出逻辑图 2. 大设计时间、分工各组自行安排 3. 验收周提交设计大报告各自负责部分 4. 各大组汇总第 16 周由组里每人提交一个： 1) 运算器、 2) 单时钟、 3) 多时钟、 4) 模拟器(软件)	01
02	数据表达	进制、整数	作 02	移/原/补码算法	实 02			02
03		算术运算	作 03	乘除法算法推导	实 03			03
04		汉字、浮点	作 04	浮点处理	实 04			04
05	指令系统	指令	作 05	汇编器	实 05			05
06		子程序	作 06	反汇编	实 06			06
07		汇编编程	作 07	模拟执行	运算器验收			07
08	I/O 系统	显示原理	作 08	显示模拟	展示			08
中								中
09	CPU 设计	单时钟	作 09	模拟器设计	实 09	单时钟设计		09
10		控制器	作 10		实 10			10
11		多时钟设计	作 11		实 11			11
12	存储器	存储器	作 12		单时钟验收			12
13		虚拟存储	作 13	虚拟存储	实 13	多时钟设计		13
14		Cache	作 14		实 14			14
15	I/O 系统	数据传送	模拟器验收		多时钟验收			15
16	复习		综合展示		综合展示			16
期								期

系统模块表

非常天空 《计算机组成》2015 课程计划

ZPC 软硬件模块一览表(增减中)

总体分 3 大块：PC 端软件、MIPS 软件、硬件（基本组件、CPU、外设）。

序	PC 端	MIPS	硬件			备注
			基本组件	CPU	外设	
01	☆补码表示	指令执行	简单 I/O	寄存器组	数码管显示	☆：需理论证明
02	☆整数加减	VGA 模拟	多路选择器	单时钟	计数器	
03	☆整数乘法	图形模式	移位寄存器	多时钟	编码器	
04	☆整数除法	系统调用	译码器	微程序	LCD	
05	☆浮点表示	BIOS	逻辑运算	流水线	总线	
06	☆浮点运算	中断	加法器	总线 CPU	地址译码	
07	汉字显示	磁盘系统	先行进位		VGA	
08	汇编器	汇编	大小比较		中断	
09	反汇编	反汇编	乘法器		中断控制器	
10	程序汇编	编辑器	阵列乘法		定时器	
11	MIPS 模拟器		除法器		串口	
12	虚拟机		存储器		可编程串口	
13	虚拟存储		不对齐读写		并行输入输出	
14	串口通讯		地址译码		键盘	
15	虚拟盘				DDR	
16					浮点运算	
17						
18						

1. 以上组件并不全在春夏实现。
2. 硬件组件可分单独与联调两种。应该先单独能够连通，然后考虑接入你的系统。
3. 当你把这些组件一个个实现，一个个加入你的系统，你的 PC 就在眼前。

小班化教学

- 分组学习制：个人做研究，小组做产品
  - 四人一组，加强交流、分工与协作相结合。
    - 个人可以对专题作更广泛深入的研究，
    - 把相对成熟的、认为最佳的提交小组集成。
    - 避免简单重复工作。
  - 强调产品化的设计思路
- 教学方法：
  - 理论与实践相结合
  - 原理推导与程序模拟相结合
  - 逻辑设计与硬件实验相结合
  - 强调系统整体概念
  - 强调课程衔接

课程任务

- 理论：
  - 完成全书知识点学习。（详见大纲）
- 软件：
  - 完成各种算法证明，程序模拟；
  - 编写MIPS汇编、反汇编程序；
  - 完成一个MIPS模拟器，模拟调试MIPS指令执行。
  - 编写虚拟磁盘程序；
- 硬件：
  - 完成运算器设计，包括整数算术运算、逻辑运算，浮点算术运算；
  - 完成一个基于FPGA的有限指令CPU；
    - 单时钟、多时钟、微程序
  - 中断处理、总线结构设计研究。
  - VGA显示器、键盘的专用设计研究；

Contents

Part I Program Structure and Execution

Representing and Manipulating Information

- 2.2 Integer Representations 56
- 2.2.1 Integral Data Types 57
- 2.2.2 Unsigned Encodings 58
- 2.2.3 Two's-Complement Encodings 60
- 2.2.4 Conversions Between Signed and Unsigned 65
- 2.2.5 Signed vs. Unsigned in C 69
- 2.2.6 Expanding the Bit Representation of a Number 71
- 2.2.7 Truncating Numbers 75
- 2.2.8 Advice on Signed vs. Unsigned 76

- 3
- Machine-Level Representation of Programs 153
- 3.1 A Historical Perspective 156
- 3.2 Program Encodings 159
- 3.2.1 Machine-Level Code 160
- 3.2.2 Code Examples 162
- 3.2.3 Notes on Formatting 165
- 3.3 Data Formats 167

Processor Architecture

- 4.1 The Y86 Instruction Set Architecture 336
- 4.1.1 Programmer-Visible State 336
- 4.1.2 Y86 Instructions 337
- 4.1.3 Instruction Encoding 339
- 4.1.4 Y86 Exceptions 344
- 4.1.5 Y86 Programs 345
- 4.1.6 Some Y86 Instruction Details 350
- 4.2 Logic Design and the Hardware Control Language HCL 352
- 4.2.1 Logic Gates 353
- 4.2.2 Combinational Circuits and HCL Boolean Expressions 354
- 4.2.3 Word-Level Combinational Circuits and HCL Integer Expressions 355
- 4.2.4 Set Membership 360
- 4.2.5 Memory and Clocking 361

- 4.6 Summary 449
- 4.6.1 Y86 Simulators 450
- Bibliographic Notes 451
- Homework Problems 451
- Solutions to Practice Problems 457

Optimizing Program Performance

- 5.1 Capabilities and Limitations of Optimizing Compilers 476
- 5.2 Expressing Program Performance 480
- 5.3 Program Example 482
- 5.4 Eliminating Loop Inefficiencies 486
- 5.5 Reducing Procedure Calls 490
- 5.6 Eliminating Unneeded Memory References 491
- 5.7 Understanding Modern Processors 496
- 5.7.1 Overall Operation 497
- 5.7.2 Functional Unit Performance 500
- 5.7.3 An Abstract Model of Processor Operation 502

The Memory Hierarchy

- 6.1 Storage Technologies 561
- 6.1.1 Random-Access Memory 561
- 6.1.2 Disk Storage 570
- 6.1.3 Solid State Disks 581
- 6.1.4 Storage Technology Trends 583
- 6.2 Locality 586
- 6.2.1 Locality of References to Program Data 587
- 6.2.2 Locality of Instruction Fetches 588
- 6.2.3 Summary of Locality 589

- 6.5 Writing Cache-friendly Code 615
- 6.6 Putting It Together: The Impact of Caches on Program Performance 620
- 6.6.1 The Memory Mountain 621
- 6.6.2 Rearranging Loops to Increase Spatial Locality 625
- 6.6.3 Exploiting Locality in Your Programs 629
- 6.7 Summary 629
- Bibliographic Notes 630
- Homework Problems 631
- Solutions to Practice Problems 642

Part II Running Programs on a System

Linking

Exceptional Control Flow

- 8.4 Process Control 718
- 8.4.1 Obtaining Process IDs 719
- 8.4.2 Creating and Terminating Processes 719
- 8.4.3 Reaping Child Processes 723
- 8.4.4 Putting Processes to Sleep 729
- 8.4.5 Loading and Running Programs 730
- 8.4.6 Using fork and execve to Run Programs 733

- 8.5 Signals 736
- 8.5.1 Signal Terminology 738
- 8.5.2 Sending Signals 739
- 8.5.3 Receiving Signals 742
- 8.5.4 Signal Handling Issues 745
- 8.5.5 Portable Signal Handling 752
- 8.5.6 Explicitly Blocking and Unblocking Signals 753
- 8.5.7 Synchronizing Flows to Avoid Nasty Concurrency Bugs 755
- 8.6 Nonlocal Jumps 759
- 8.7 Tools for Manipulating Processes 762
- 8.8 Summary 763
- Bibliographic Notes 763
- Homework Problems 764
- Solutions to Practice Problems 771

Virtual Memory

- 9.1 Physical and Virtual Addressing 777
- 9.2 Address Spaces 778
- 9.3 VM as a Tool for Caching 779
- 9.3.1 DRAM Cache Organization 780
- 9.3.2 Page Tables 780
- 9.3.3 Page Hits 782
- 9.3.4 Page Faults 782
- 9.3.5 Allocating Pages 783
- 9.3.6 Locality to the Rescue Again 784

- 9.4 VM as a Tool for Memory Management 785
- 9.5 VM as a Tool for Memory Protection 786
- 9.6 Address Translation 787
- 9.6.1 Integrating Caches and VM 791
- 9.6.2 Speeding up Address Translation with a TLB 791
- 9.6.3 Multi-Level Page Tables 792
- 9.6.4 Putting It Together: End-to-end Address Translation 794

- 9.7 Case Study: The Intel Core i7/Linux Memory System 799
- 9.7.1 Core i7 Address Translation 800
- 9.7.2 Linux Virtual Memory System 803
- 9.8 Memory Mapping 807
- 9.8.1 Shared Objects Revisited 807
- 9.8.2 The fork Function Revisited 809
- 9.8.3 The execve Function Revisited 810
- 9.8.4 User-level Memory Mapping with the mmap Function 810

- 9.9 Dynamic Memory Allocation 812
- 9.9.1 The malloc and free Functions 814
- 9.9.2 Why Dynamic Memory Allocation? 816
- 9.9.3 Allocator Requirements and Goals 817
- 9.9.4 Fragmentation 819
- 9.9.5 Implementation Issues 820
- 9.9.6 Implicit Free Lists 820
- 9.9.7 Placing Allocated Blocks 822
- 9.9.8 Splitting Free Blocks 823

- 9.9.9 Getting Additional Heap Memory 823
- 9.9.10 Coalescing Free Blocks 824
- 9.9.11 Coalescing with Boundary Tags 824
- 9.9.12 Putting It Together: Implementing a Simple Allocator 827
- 9.9.13 Explicit Free Lists 835
- 9.9.14 Segregated Free Lists 836
- 9.10 Garbage Collection 838
- 9.10.1 Garbage Collector Basics 839
- 9.10.2 Mark&Sweep Garbage Collectors 840
- 9.10.3 Conservative Mark&Sweep for C Programs 842

- 9.11 Common Memory-Related Bugs in C Programs 843
- 9.11.1 Dereferencing Bad Pointers 843
- 9.11.2 Reading Uninitialized Memory 843
- 9.11.3 Allowing Stack Buffer Overflows 844
- 9.11.4 Assuming that Pointers and the Objects They Point to Are the Same Size 844
- 9.11.5 Making Off-by-One Errors 845
- 9.11.6 Referencing a Pointer Instead of the Object It Points to 845
- 9.11.7 Misunderstanding Pointer Arithmetic 846
- 9.11.8 Referencing Nonexistent Variables 846
- 9.11.9 Referencing Data in Free Heap Blocks 847
- 9.11.10 Introducing Memory Leaks 847

- 9.12 Summary 848
- Bibliographic Notes 848
- Homework Problems 849
- Solutions to Practice Problems 853

Part III Interaction and Communication Between Programs

System-Level I/O

- 10.1 Unix I/O 862
- 10.2 Opening and Closing Files 863
- 10.3 Reading and Writing Files 865
- 10.4 Robust Reading and Writing with the Rio Package 867
- 10.4.1 Rio Unbuffered Input and Output Functions 867
- 10.4.2 Rio Buffered Input Functions 868

- 10.5 Reading File Metadata 873
- 10.6 Sharing Files 875
- 10.7 I/O Redirection 877
- 10.8 Standard I/O 879
- 10.9 Putting It Together: Which I/O Functions Should I Use? 880
- 10.10 Summary 881
- Bibliographic Notes 882
- Homework Problems 882
- Solutions to Practice Problems 883

Network Programming

- 11.1 The Client-Server Programming Model 886
- 11.2 Networks 887
- 11.3 The Global IP Internet 891
- 11.3.1 IP Addresses 893
- 11.3.2 Internet Domain Names 895
- 11.3.3 Internet Connections 899

- 11.4 The Sockets Interface 900
- 11.4.1 Socket Address Structures 901
- 11.4.2 The socket Function 902
- 11.4.3 The connect Function 903
- 11.4.4 The open_clientfd Function 903
- 11.4.5 The bind Function 904
- 11.4.6 The listen Function 905
- 11.4.7 The open_listenfd Function 905
- 11.4.8 The accept Function 907
- 11.4.9 Example Echo Client and Server 908

- 11.5 Web Servers 911
- 11.5.1 Web Basics 911
- 11.5.2 Web Content 912
- 11.5.3 HTTP Transactions 914
- 11.5.4 Serving Dynamic Content 916
- 11.6 Putting It Together: The TinyWeb Server 919
- 11.7 Summary 927
- Bibliographic Notes 928
- Homework Problems 928
- Solutions to Practice Problems 929

Concurrent Programming

- A
- Error Handling 999
- A.1 Error Handling in Unix Systems 1000
- A.2 Error-Handling Wrappers 1001
- References 1005
- Index 1011

课程安排

周	章节	知识点	分	实验方式	
1	概论				汉字键盘设计，熟悉C语言编程
2	数据表达	数据表示加减	5	程序模拟	用移码表示整数，实现加减运算
3		乘除	5		补码乘除法设计模拟
4		浮点数	5		浮点处理或汉字
5	汇编语言	汇编	5		MIPS汇编到机器码
6		反汇编	5		机器码到MIPS汇编
7		模拟CPU	5		MIPS CPU模拟
8	CPU设计	单个组件	5	硬件实验	交硬件实验报告
9		单时钟	5		交硬件实验报告
10		多时钟	5		交硬件实验报告
11	存储系统	存储器	5	平时	（期中考试）
12		Cache	5		（平时机动）
13		虚拟存储	5	程序模拟	程序模拟虚拟存储
14	I/O系统	I/O	5		显示模拟
15		综合	5		综合演示
16	复习				

实验设计

- 软件实验：
  - MIPS汇编
  - MIPS模拟器
- 硬件实验：
  - 运算器
  - 单时钟
  - 多时钟（组合逻辑与微程序）
- 小实验：每周作业，课堂实验
  - 软件：算法模拟
  - 硬件：部件实验

实验考核

- 大设计与课堂实验相结合。
  - 每次实验灵活布置与设计有关的小实验，
  - 大设计由小组自行安排时间。
- 个人分工与小组集成相结合
- 验收清单

学期提交

- 刻写光盘可便于后续课程

春夏学期终极提交

*：平时注意积累资料，期末按 组 统一刻盘。

号	材质	名称	时间	说明	备注	号
01	纸质	运算器使用手册	第 06 周	设计彩色封面（宣传册页可选）		01
02		CPU 使用手册	第 14 周	设计彩色封面		02
03		CPU 宣传册页	第 14 周	单页彩色 A4 双面		03
04	光盘 组 目录	运算器	第 06 周	A. 实验报告 B. 所有代码 C. 使用手册 D. 演示 PPT E. 其它（如视频）	全组汇总，每人负责一个。	04
05		单时钟	第 11 周			05
06		多时钟	第 14 周			06
07		虚拟机	第 14 周			07
08	光盘 个人 目录	基本资料	第 14 周	A. 个人介绍 B. 周记 C. 照片 D. 作业		08
09		课堂实验	第 14 周	个人课堂实验代码		09
10		运算器	第 06 周	A. 个人部分所有代码 B. 个人部分实验报告		10
11		单时钟	第 11 周			11
12		多时钟	第 14 周			12
13		虚拟机	第 14 周			13
14						14


非常天空


ZPC 2015
ZHEJIANG UNIVERSITY
浙江大学计算机学院


ZPC 2016
ZHEJIANG UNIVERSITY
浙江大学计算机学院


ZPC 2014
design
浙江大学计算机学院


ZPC 2017
ZHEJIANG UNIVERSITY
浙江大学计算机学院

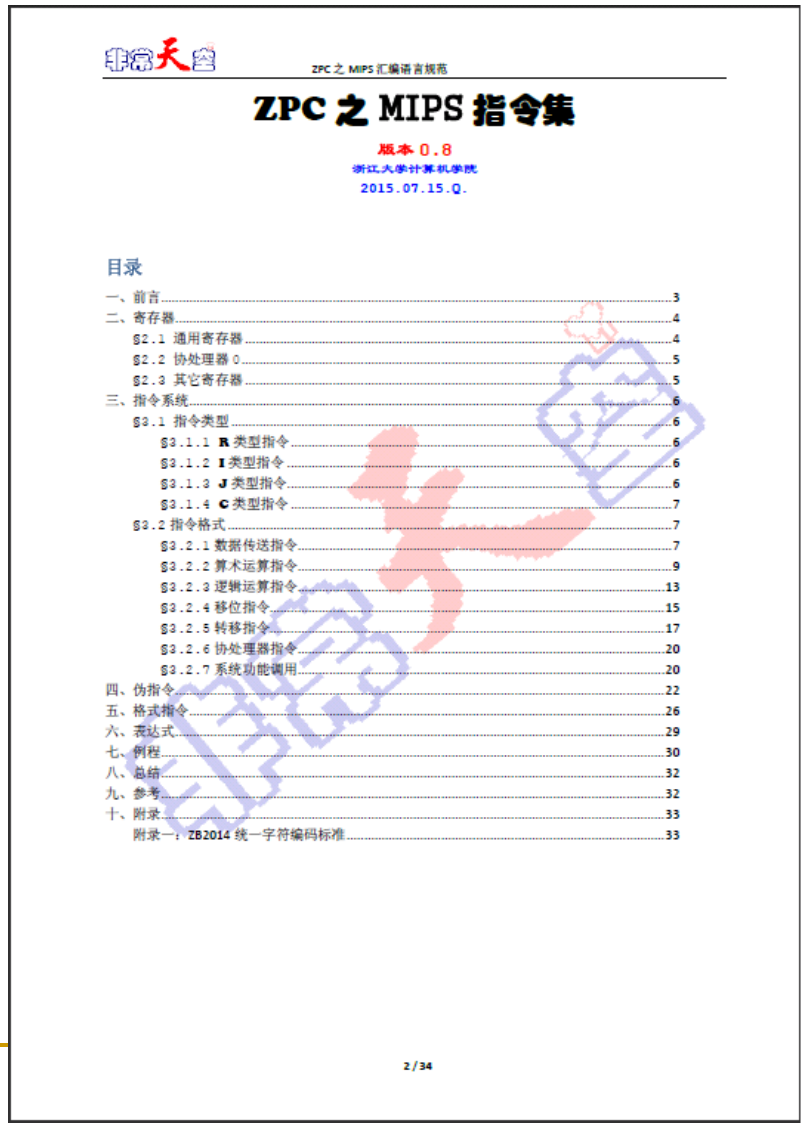

CPU 2013
design
浙江大学计算机学院


CPU 2012
design
浙江大学计算机学院


CPU 2011
design
浙江大学计算机学院

三、ZPC之MIPS指令集

- 目录
- 一、前言
- 二、寄存器
- §2.1 通用寄存器
- §2.2 协处理器0
- §2.3 其它寄存器
- 三、指令系统
- §3.1 指令类型
- §3.1.1 R类型指令
- §3.1.2 I类型指令
- §3.1.3 J类型指令
- §3.1.4 C类型指令
- §3.2指令格式
- §3.2.1数据传送指令
- §3.2.2算术运算指令
- §3.2.3逻辑运算指令
- §3.2.4移位指令
- §3.2.5转移指令
- §3.2.6协处理器指令
- §3.2.7系统功能调用



非常天空 ZPC 之 MIPS 汇编语言规范

ZPC 之 MIPS 指令集

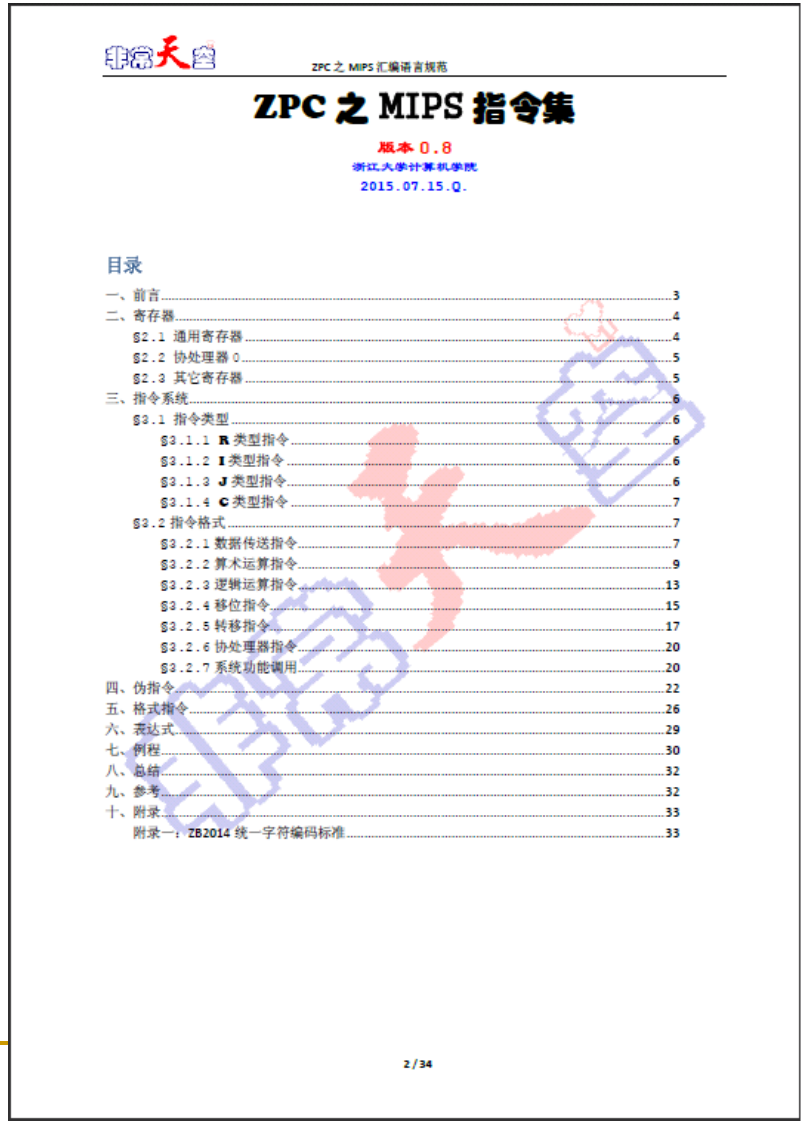
版本 0.8

浙江大学计算机学院

2015.07.15.Q.

目录

2 / 34



非常天空 ZPC 之 MIPS 汇编语言规范

ZPC 之 MIPS 指令集

版本 0.8

浙江大学计算机学院

2015.07.15.Q.

目录

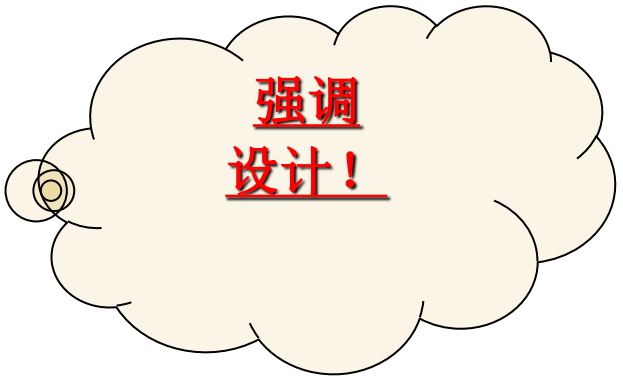
2 / 34

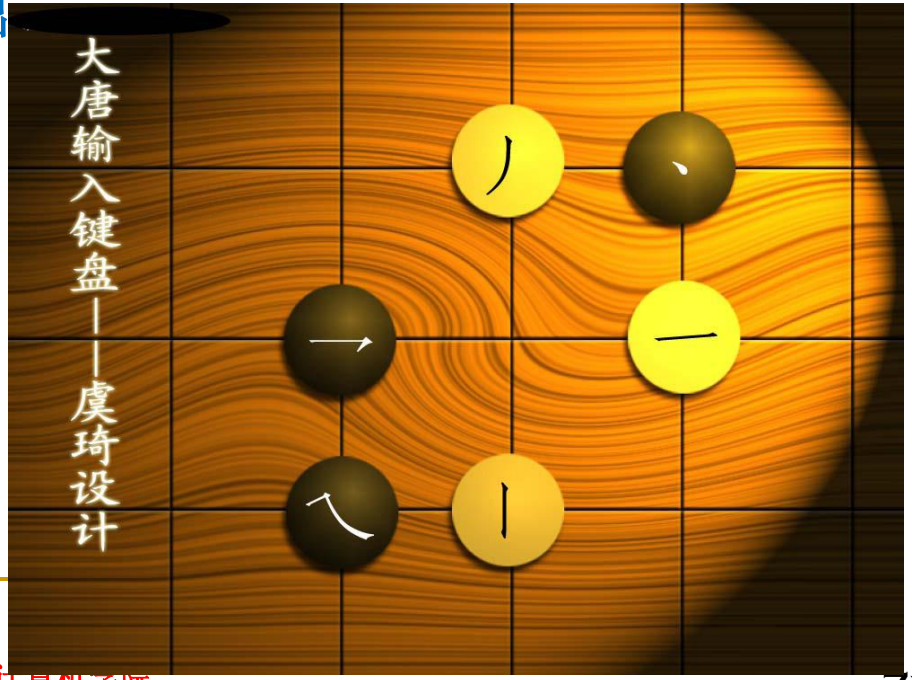
网站建设

- 课程资料
- 参考资料下载
- 作业提交、批改

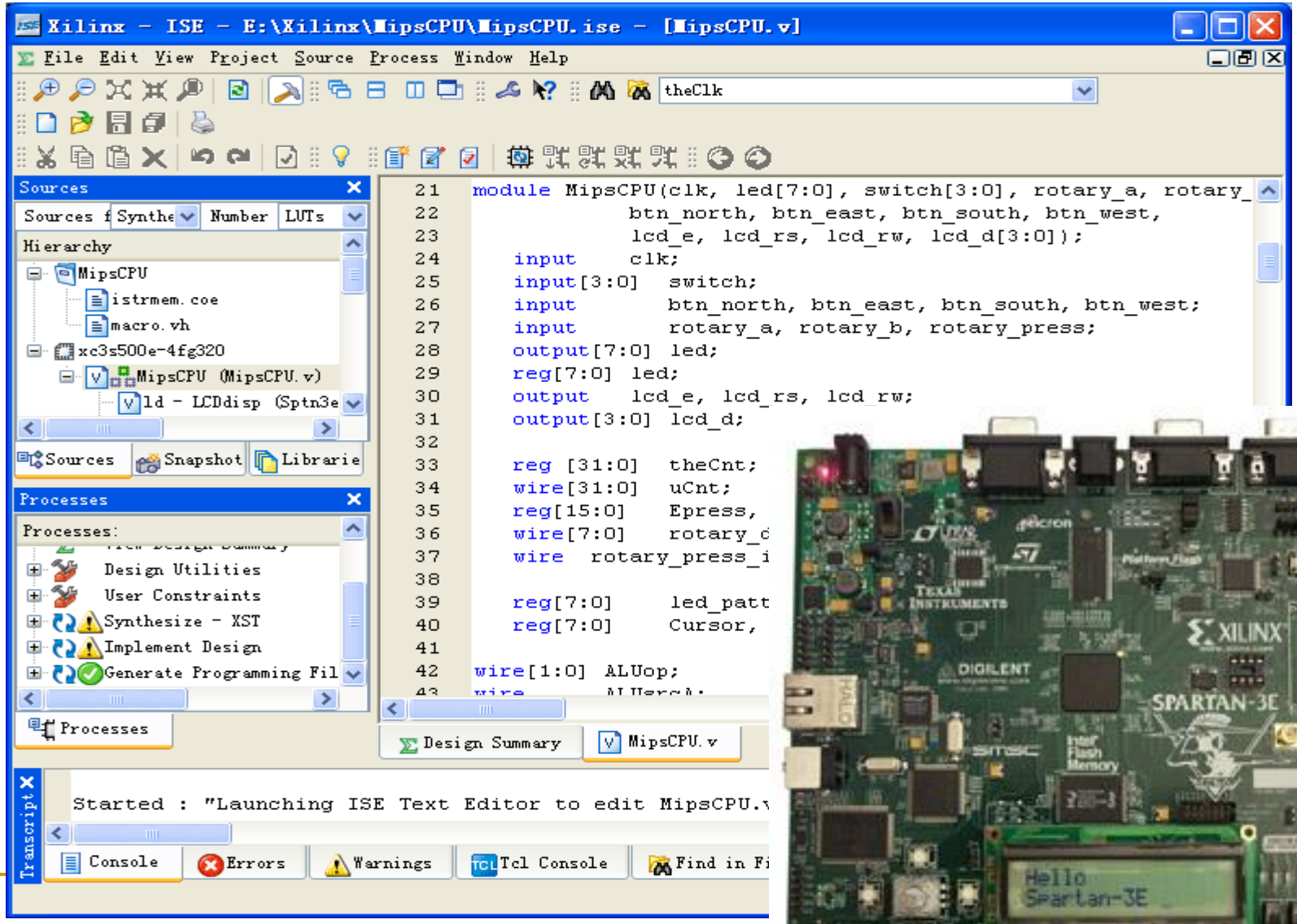
作业实例1：大汉键盘

- 创意来源：
  - 围棋、田字格、笔画
- 输入方法：
  - 1.根据需要输入的汉字结构在田字格中摆放棋子。
  - 2.每摆放一次则屏幕响应位置出现该笔画。
  - 3.笔画组合到一定程度后，出现符合要求的待选汉字。
  - 4.选择你需要的汉字。
- 实现条件：
  - 传感棋盘、
  - 识别软件、
  - 汉字字库





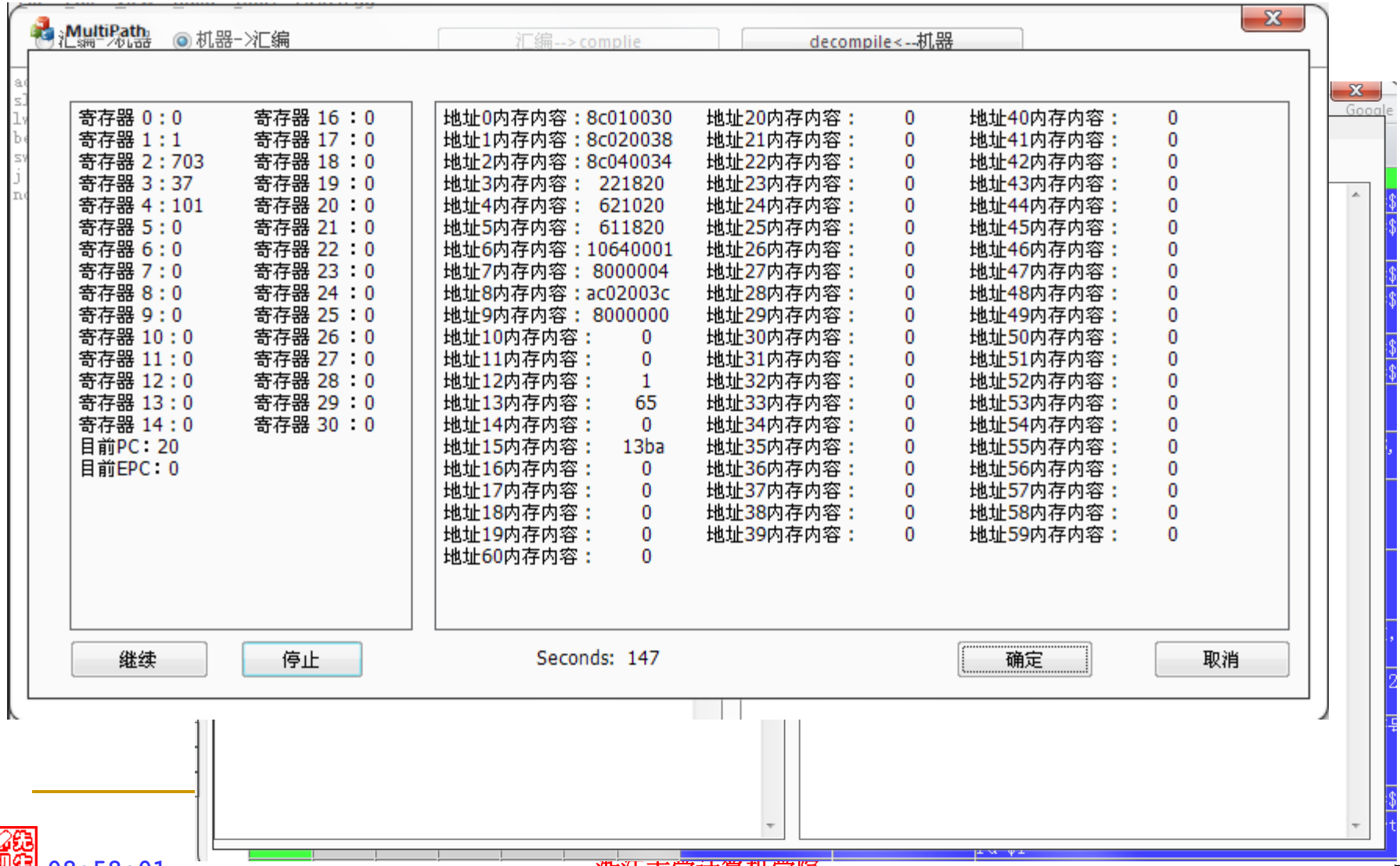
作业实例2：ISE


Xilinx - ISE - E:\Xilinx\MipsCPU\MipsCPU.ise - [MipsCPU.v]
File Edit View Project Source Process Window Help
theClk
Sources
Sources f Synthe Number LUTs
Hierarchy
MipsCPU
istrmem.coe
macro.vh
xc3s500e-4fg320
MipsCPU (MipsCPU.v)
ld - LCDdisp (Sptn3e
Sources Snapshot Librarie
Processes
Processes:
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming Fil
Processes
21 module MipsCPU(clk, led[7:0], switch[3:0], rotary_a, rotary_
22 btn_north, btn_east, btn_south, btn_west,
23 lcd_e, lcd_rs, lcd_rw, lcd_d[3:0]);
24 input clk;
25 input[3:0] switch;
26 input btn_north, btn_east, btn_south, btn_west;
27 input rotary_a, rotary_b, rotary_press;
28 output[7:0] led;
29 reg[7:0] led;
30 output lcd_e, lcd_rs, lcd_rw;
31 output[3:0] lcd_d;
32
33 reg [31:0] theCnt;
34 wire[31:0] uCnt;
35 reg[15:0] Epress,
36 wire[7:0] rotary_d
37 wire rotary_press_i
38
39 reg[7:0] led_patt
40 reg[7:0] Cursor,
41
42 wire[1:0] ALUop;
Design Summary
MipsCPU.v
Transcript
Started : "Launching ISE Text Editor to edit MipsCPU.v
Console
Errors
Warnings
Tcl Console
Find in Fi
XILINX
DIGILENT
SPARTAN-3E
Hello
Spartan-3E

作业实例3：模拟器

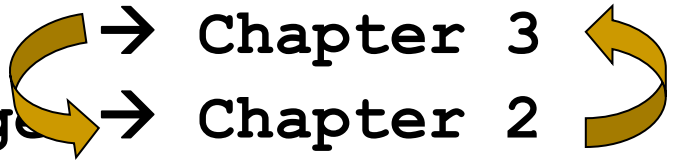

MIPS模拟器
MIPS
机器码
addi $t0, $t0, 3
addi $t6, $zero, 1
addi $s0, $s0, 4
add $t1, $t0, $t6
sub $t2, $zero, $t1
or $t3, $t0, $s0
ori $t4, $t0, 12
nor $t5, $t0, $t1
and $s1, $t0, $t1
slt $s2, $t0, $t1
addi $s4, $s4, 65
sw $s4, 5116 ($s3)
lw $s5, 5116 ($s3)
00100001000010000000000000000011
00100000000011100000000000000001
00100010000100000000000000000100
00000001000011100100100000100000
00000000000001001010100000100010
00000001000100000101100000100101
00110101000011000000000000001100
00000001000010010110100000100111
00000001000010011000100000100100
00000001000010011001000000101010
00100010100101000000000001000001
10100110011101000001001111111100
10001110011101010001001111111100
寄存器
$zero: 0
$at: 0
$v0: 0
$v1: 0
$a0: 0
$a1: 0
$a2: 0
$a3: 0
$t0: 3
$t1: 4
$t2: -4
$t3: 7
$t4: 0
$t5: -8
$t6: 1
$t7: 0
$s0: 4
$s1: 0
$s2: 1
$s3: 0
$s4: 65
$s5: 65
$s6: 0
$s7: 0
$t8: 0
$t9: 0
$k0: 0
$k1: 0
$gp: 0
$sp: 0
$fp: 0
$ra: 0
1279
1311
1343
1375
1407
1439
1471
1503
1535
1567
1599
1631
1663
1695
1727
1759
1791
1823
1855
1887
1919
1951
1983
2015
A
显示器
执行
退出
隐藏<<
清空

作业实例4：模拟器

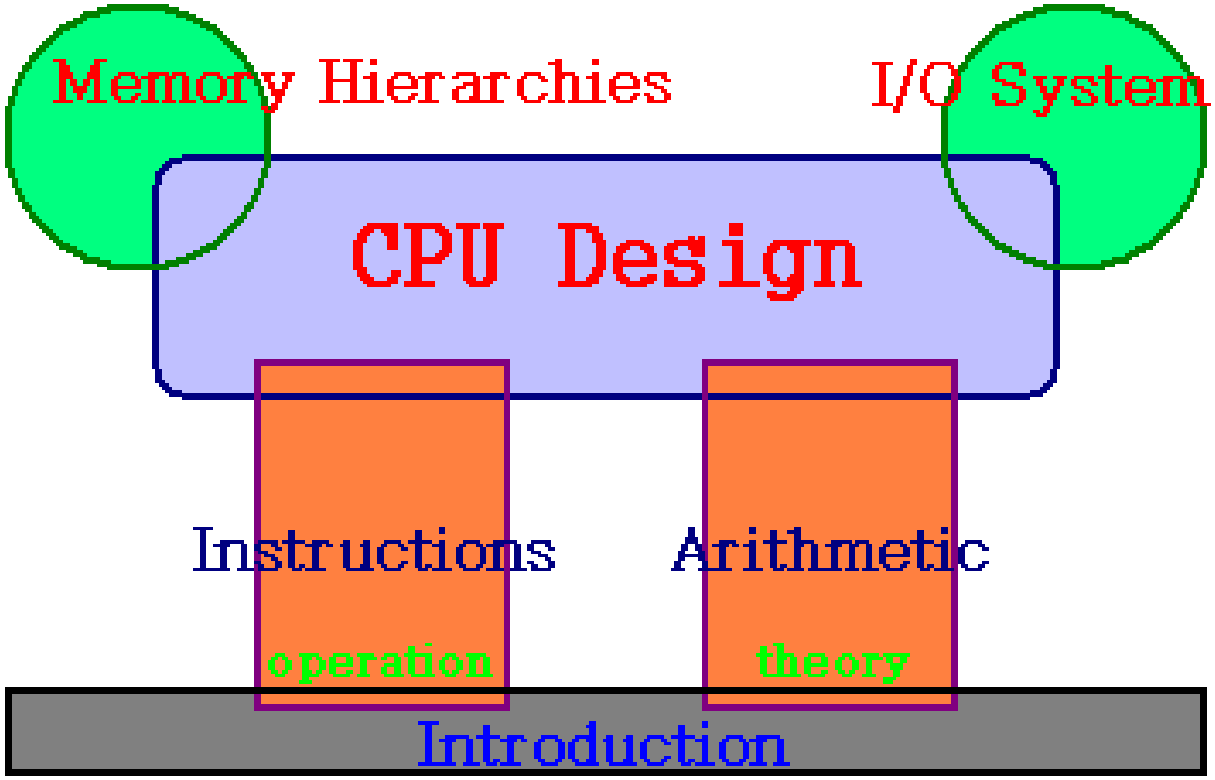

MultiPath
机器->汇编
汇编-->complie
decompile<--机器
寄存器 0：0
寄存器 1：1
寄存器 2：703
寄存器 3：37
寄存器 4：101
寄存器 5：0
寄存器 6：0
寄存器 7：0
寄存器 8：0
寄存器 9：0
寄存器 10：0
寄存器 11：0
寄存器 12：0
寄存器 13：0
寄存器 14：0
寄存器 16：0
寄存器 17：0
寄存器 18：0
寄存器 19：0
寄存器 20：0
寄存器 21：0
寄存器 22：0
寄存器 23：0
寄存器 24：0
寄存器 25：0
寄存器 26：0
寄存器 27：0
寄存器 28：0
寄存器 29：0
寄存器 30：0
目前PC：20
目前EPC：0
地址0内存内容：8c010030
地址1内存内容：8c020038
地址2内存内容：8c040034
地址3内存内容： 221820
地址4内存内容： 621020
地址5内存内容： 611820
地址6内存内容：10640001
地址7内存内容： 8000004
地址8内存内容：ac02003c
地址9内存内容： 8000000
地址10内存内容： 0
地址11内存内容： 0
地址12内存内容： 1
地址13内存内容： 65
地址14内存内容： 0
地址15内存内容： 13ba
地址16内存内容： 0
地址17内存内容： 0
地址18内存内容： 0
地址19内存内容： 0
地址60内存内容： 0
地址20内存内容： 0
地址21内存内容： 0
地址22内存内容： 0
地址23内存内容： 0
地址24内存内容： 0
地址25内存内容： 0
地址26内存内容： 0
地址27内存内容： 0
地址28内存内容： 0
地址29内存内容： 0
地址30内存内容： 0
地址31内存内容： 0
地址32内存内容： 0
地址33内存内容： 0
地址34内存内容： 0
地址35内存内容： 0
地址36内存内容： 0
地址37内存内容： 0
地址38内存内容： 0
地址39内存内容： 0
地址40内存内容： 0
地址41内存内容： 0
地址42内存内容： 0
地址43内存内容： 0
地址44内存内容： 0
地址45内存内容： 0
地址46内存内容： 0
地址47内存内容： 0
地址48内存内容： 0
地址49内存内容： 0
地址50内存内容： 0
地址51内存内容： 0
地址52内存内容： 0
地址53内存内容： 0
地址54内存内容： 0
地址55内存内容： 0
地址56内存内容： 0
地址57内存内容： 0
地址58内存内容： 0
地址59内存内容： 0
继续
停止
Seconds: 147
确定
取消

Course

- Introduction → Chapter 1 & 4
- Arithmetic → Chapter 3
- MIPS assembly language → Chapter 2
- CPU design → Chapter 5
- Memory Hierarchy → Chapter 7
- I/OInterface & Peripherals → Chapter 8



Course Structure



课程安排（精研班）

周	章节	知识点	分	实验方式	
1	概论				熟悉C语言编程
2	数据表达	数据表示加减	5	程序模拟	选择一种表示方式，实行加减运算
3		乘除	5		乘除法设计模拟
4		浮点数	5		浮点处理或汉字
5	汇编语言	汇编	5		MIPS汇编到机器码
6		反汇编	5		机器码到MIPS汇编
7		模拟CPU	5		MIPS CPU模拟
8	CPU设计	单个组件	5	硬件实验	交硬件实验报告
9		单时钟	5		交硬件实验报告
10		多时钟	5		交硬件实验报告
11	存储系统	存储器	5		（期中考试）
12		Cache	5		（平时机动）
13		虚拟存储	5	程序模拟	程序模拟虚拟存储
14	I/O系统	I/O	5	硬件实验	显示模拟
15		综合	5	软件/硬件	综合演示
16	复习				

Computer Organization & Design

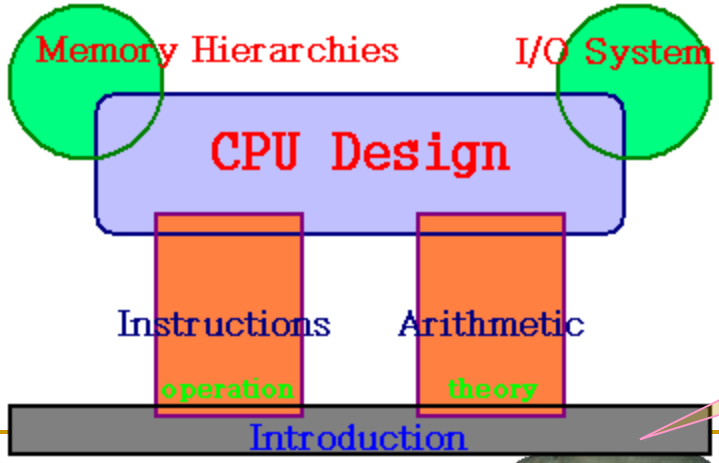
第01章：Introduction

楼学庆

http://10.214.47.99/

Email:hzlou@163.com

玉泉校区曹光彪东楼507室


Memory Hierarchies
I/O System
CPU Design
Instructions
operation
Arithmetic
theory
Introduction

Today

von Neumann' Computer

- Today's computers are built on 2 key principles:
  - ①Instruction are represented as numbers.
  - ②Programs can be stored in memory to be read or written just like numbers.

Computer Generations

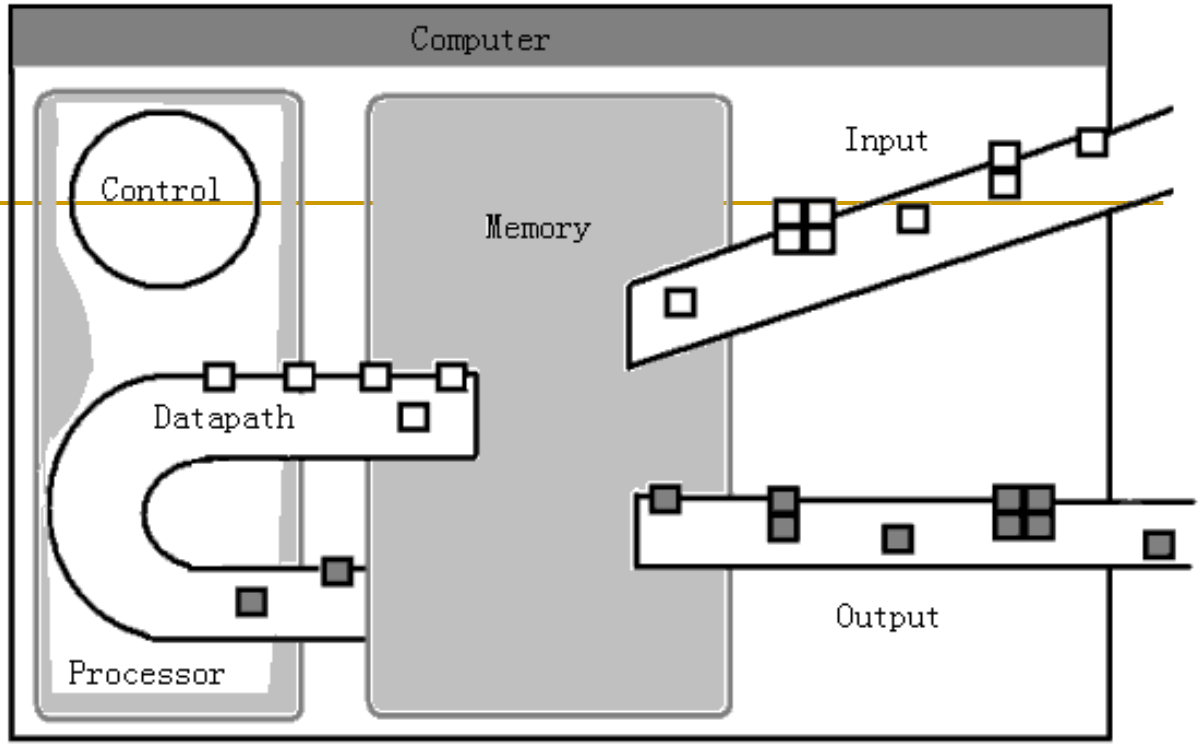
- First generation
  - 1950-1959, vacuum tubes, commercial electronic computer
- Second generation
  - 1960-1968, transistors, cheaper computers
- Third generation
  - 1969-1977, integrated circuit, minicomputer
- Fourth generation
  - 1978-1997, LSI and VLSI, PCs and workstations
- Fifth generation
  - 1998-?, micromation and hugeness

Four Design Principles

- 1. Simplicity favors regularity
- 2. Smaller is faster
- 3. Good design demands good compromises
- 4. Make the common case fast

Part 1:

Hardware



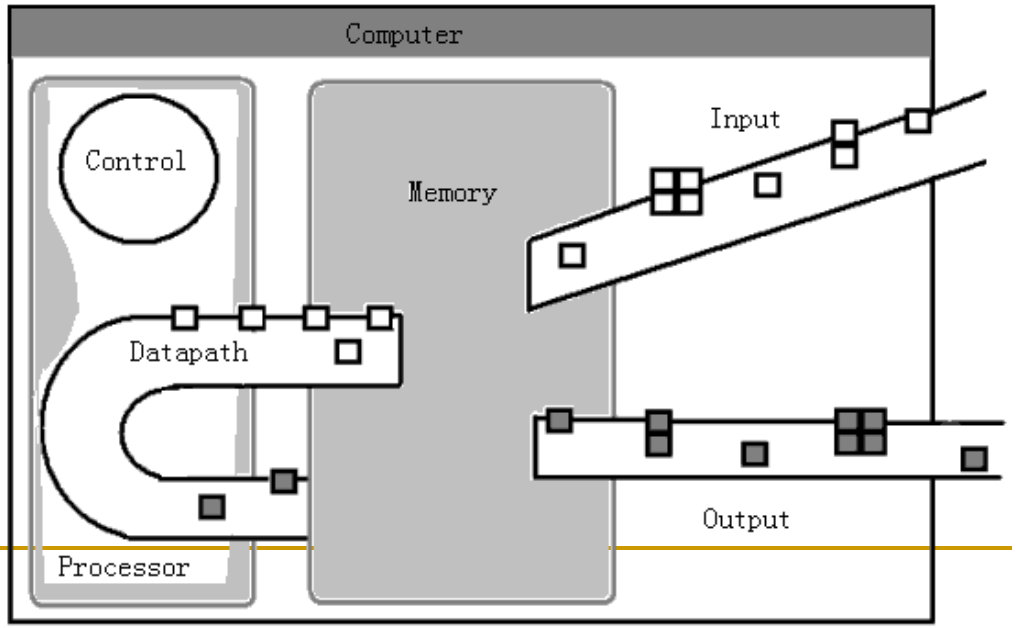
Five Parts

- Control (CU)
- Datapath (EU)
- Memory
- Input
- Output

CPU

RAM

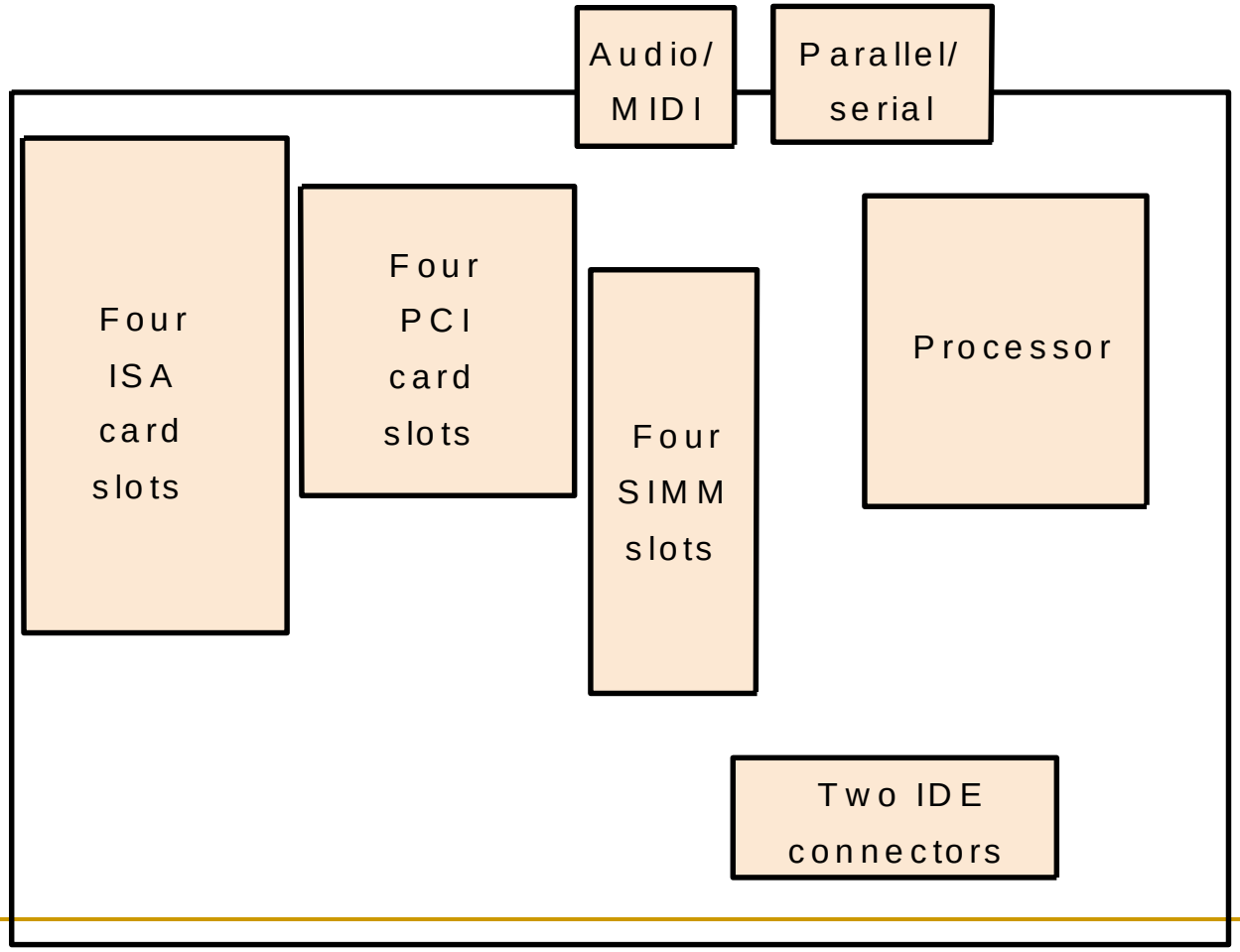
I/O



Lecture 1: Introduction

- §1.1 Below Your Program
- §1.2 Under the Covers
- §1.3 CPU Performance and Its Factors
- §1.4 Evaluating Performance

Close-up of PC motherboard



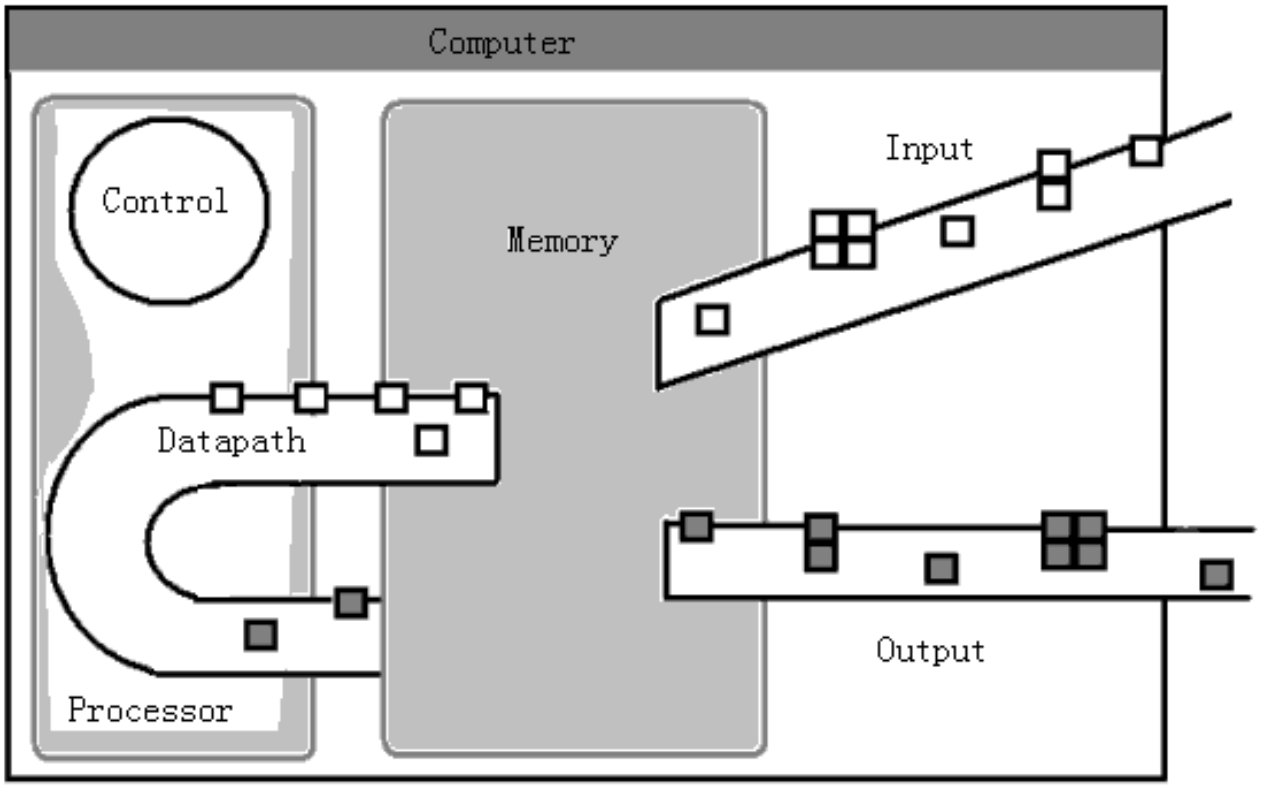
Must a Programmer Care About Hardware?

- Memory management: if we understand how/where data is placed, we can help ensure that relevant data is nearby

- Thread management: if we understand how threads interact, we can write smarter multi-threaded programs

  → Why do we care about multi-threaded programs?

Computer Organization



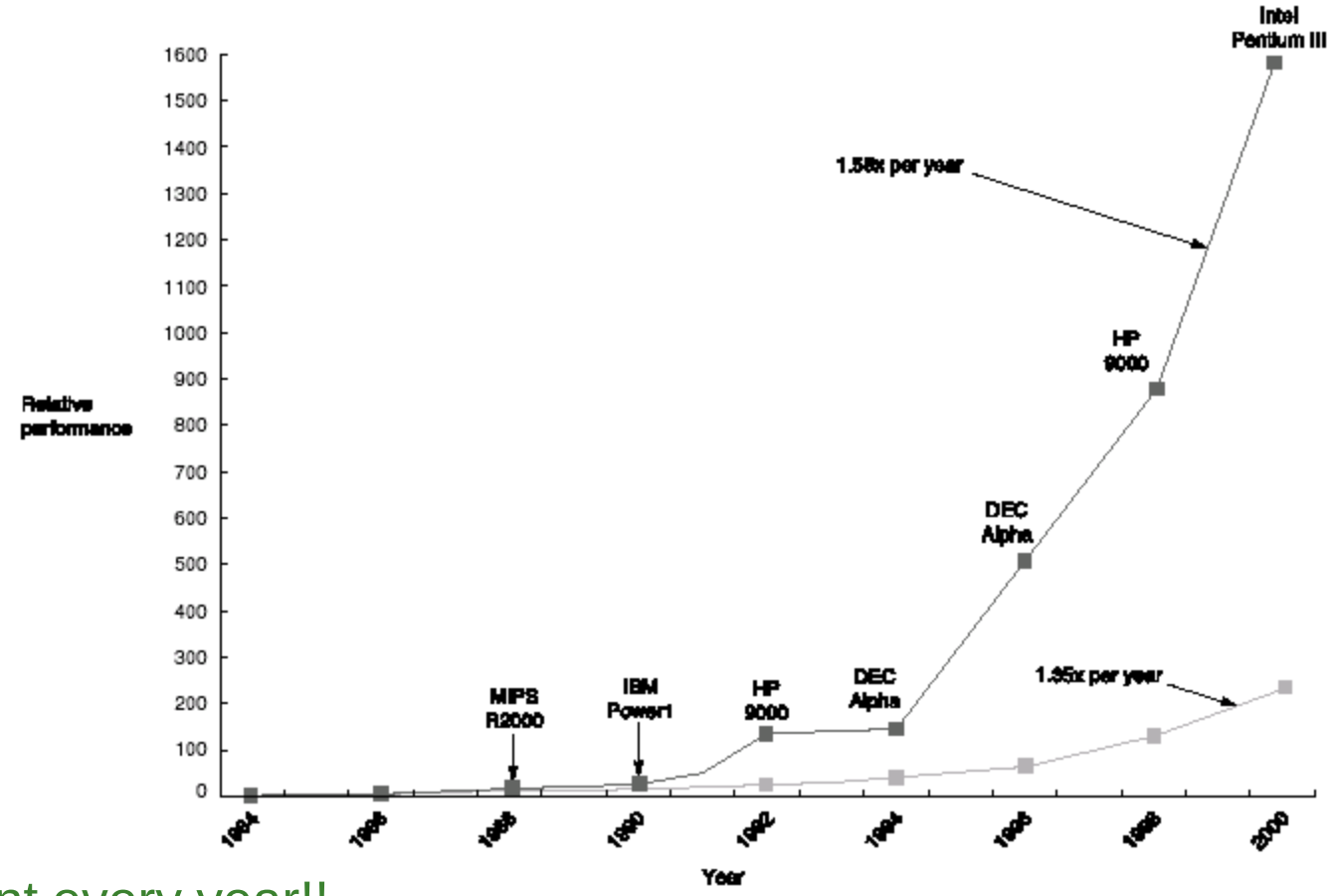
Control Process Unit → CPU

- CPU
  - Control (CU)
  - Datapath (EU)
- Specialties
  - Main Frequency
    - IBM-PC: 4.77MHz
    - Now: 3.8 GHz
  - Machine Word
    - IBM-PC: 8086/8088: 16bits
    - Now: 32bits

Control Process Unit → CPU

- Instruction Set
  - CISC: Complex Instruction Set Computer
  - RISC: Reduced Instruction Set Computer
- Performance
  - CPI: Clock Cycles per Instruction
  - MIPS: Millions Instructions per Second
  - MFLOPS: millions of floating-point operations per second

Microprocessor Performance



50% improvement every year!!

What contributes to this improvement?

Modern Trends

- Historical contributions to performance:
  - Better processes (faster devices) ~20%
  - Better circuits/pipelines ~15%
  - Better organization/architecture ~15%
- In the future, bullet-2 will help little and bullet-3 will not help much for a single core!

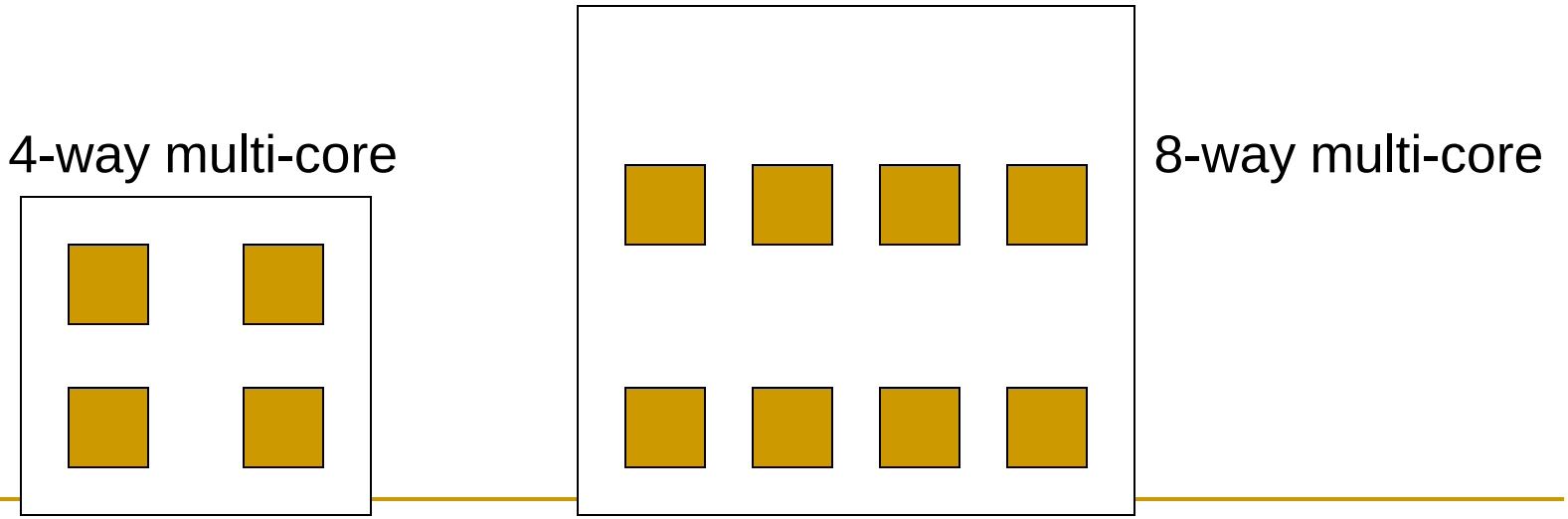
	Pentium	P-Pro	P-II	P-III	P-4	Itanium	Montecito
Year	1993	95	97	99	2000	2002	2005
Transistors	3.1M	5.5M	7.5M	9.5M	42M	300M	1720M
Clock	60M	200M	300M	500M	750M	800M	1800M

Moore's Law in action

At this point, adding transistors to a core yields little benefit

What Does This Mean to a Programmer?

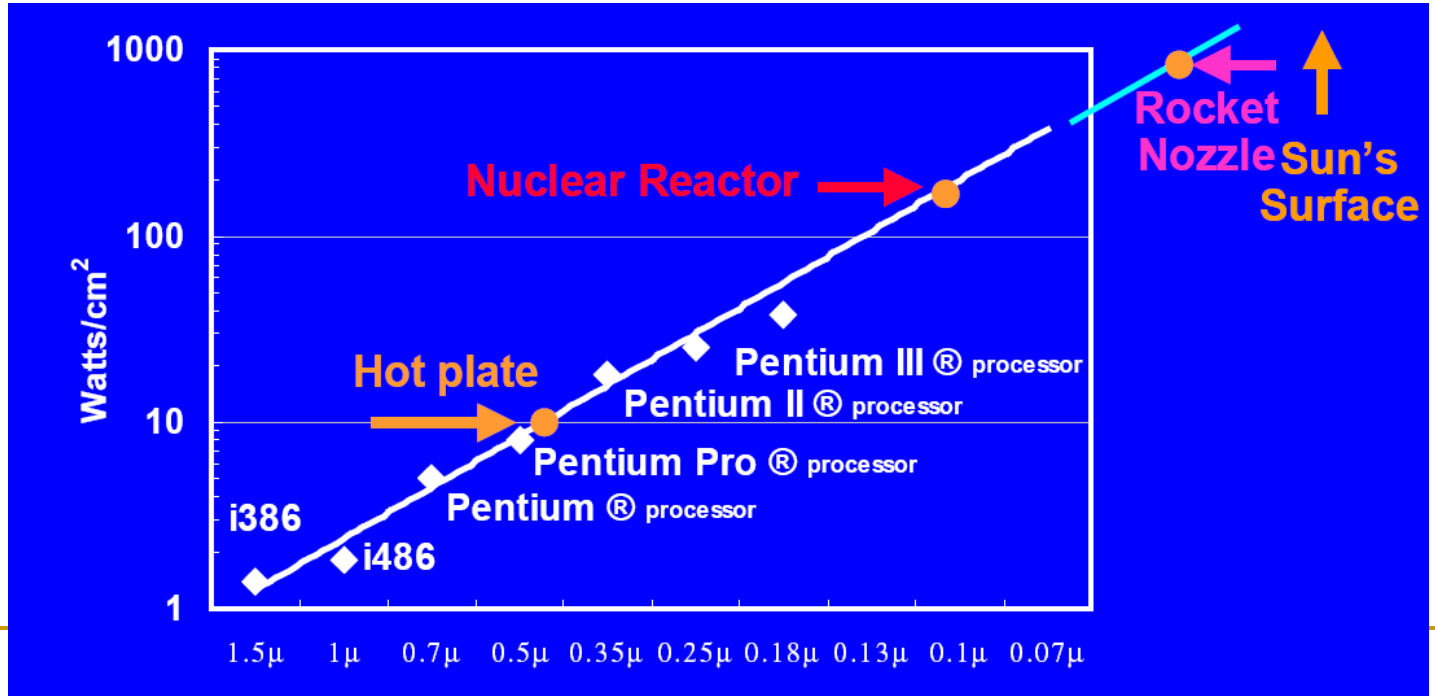
- In the past, a new chip directly meant 50% higher performance for a program
- Today, one can expect only a 20% improvement, unless…
  the program can be broken up into multiple threads
- Expect #threads to emerge as a major metric for software quality



Challenges for the Hardware Designers

Major concerns:

- The performance problem (especially scientific workloads)
- The power dissipation problem (especially embedded processors)
- The temperature problem
- The reliability problem



The HW/SW Interface

Application software

Systems software
(OS, compiler)

Hardware

```
a[i] = b[i] + c;
```

Compiler

```
lw    $15, 0($2)
add   $16, $15, $14
add   $17, $15, $13
lw    $18, 0($12)
lw    $19, 0($17)
add   $20, $18, $19
sw    $20, 0($16)
```

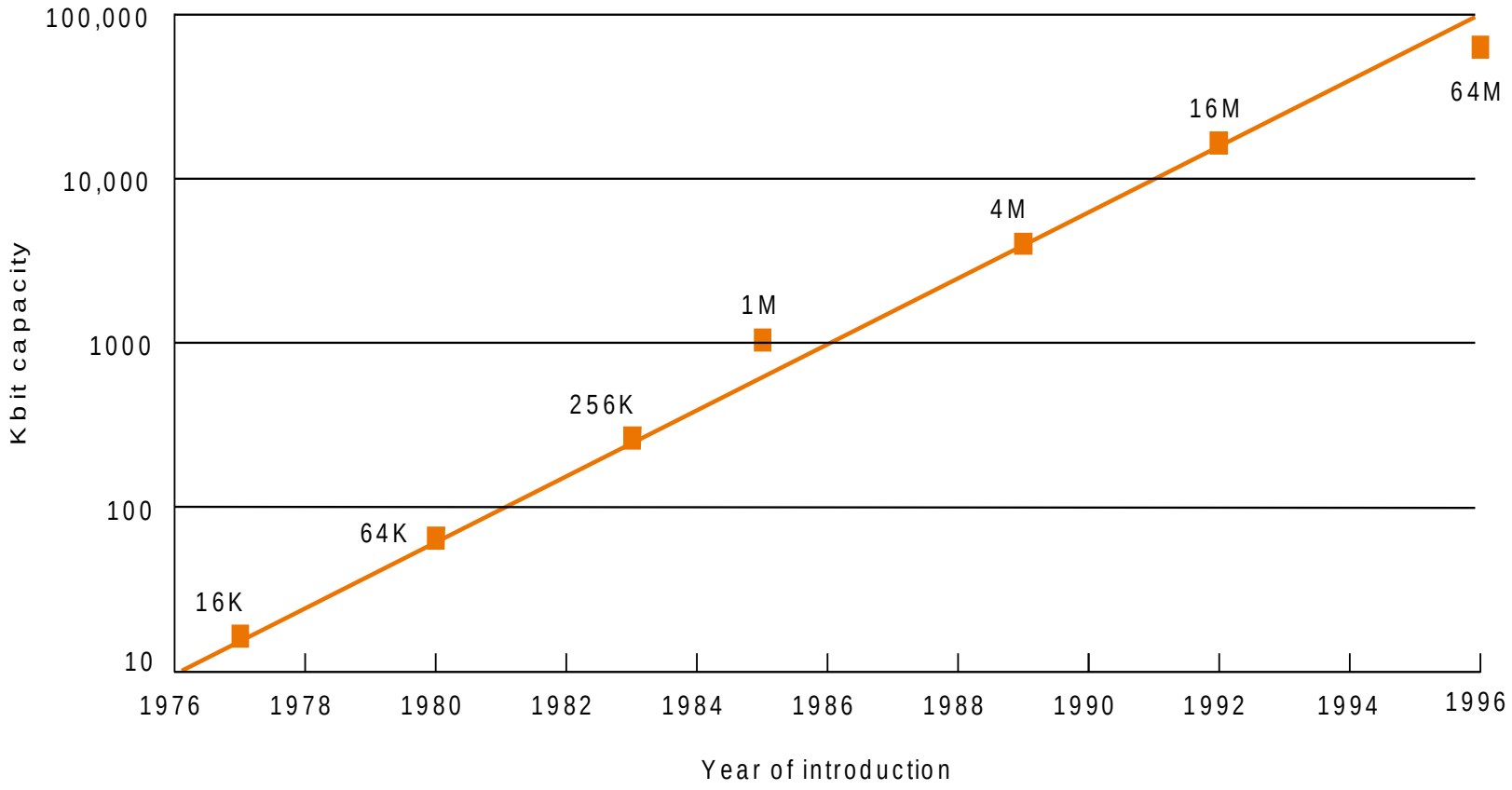
Assembler

```
00000010110000
11010000100010
```

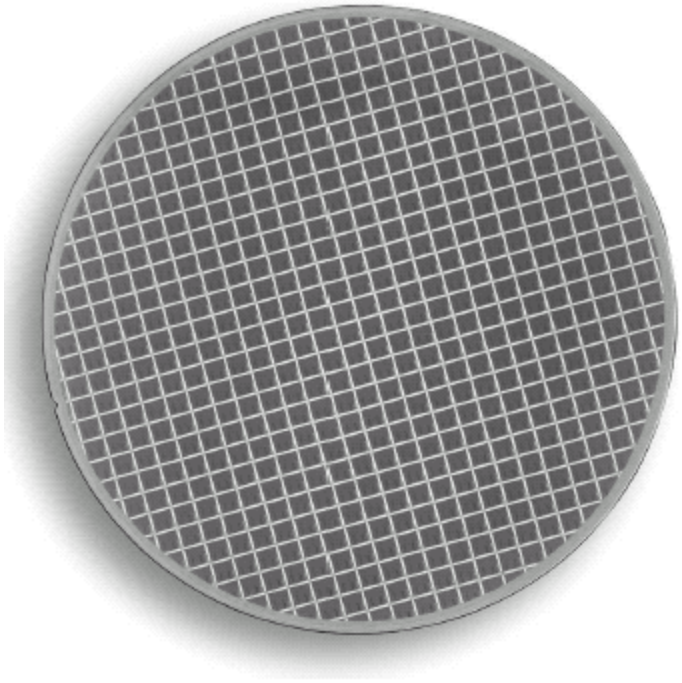
Computer Components

- Input/output devices
- Secondary storage: non-volatile, slower, cheaper
- Primary storage: volatile, faster, costlier
- CPU/processor

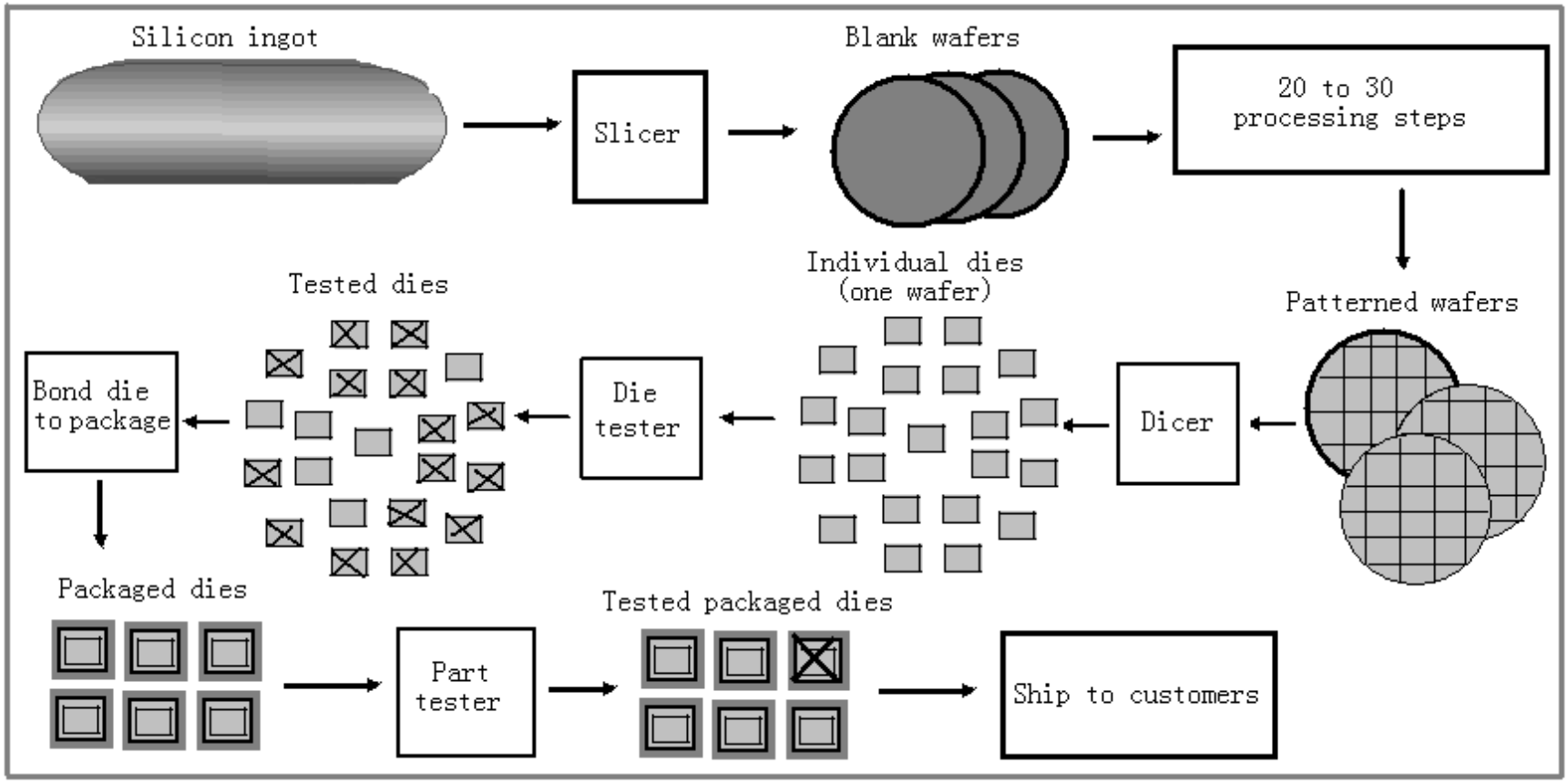
Growth of capacity per DRAM chip over time



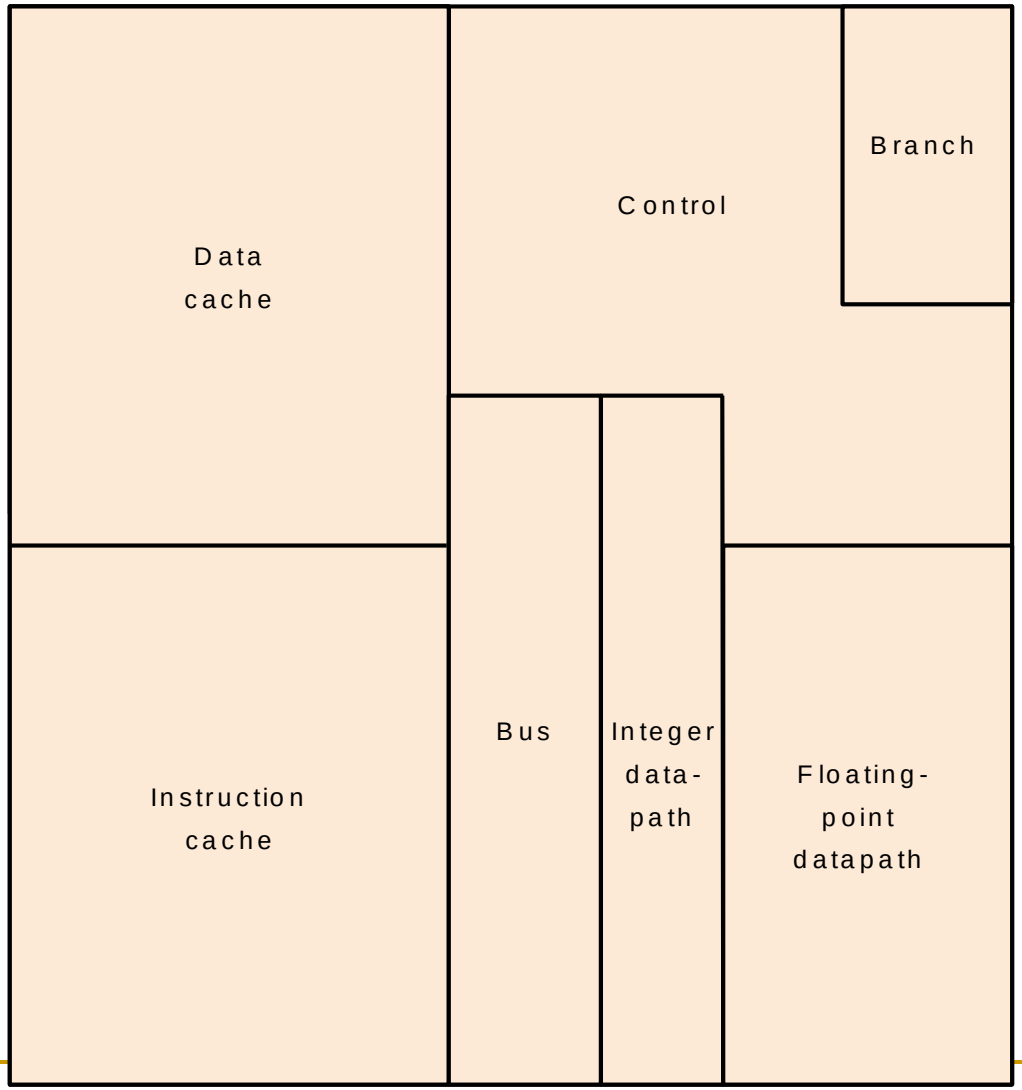
Wafers and Dies



The semiconductor silicon and the chip manufacturing process



Inside the processor chip



Manufacturing Process

- Silicon wafers undergo many processing steps so that different parts of the wafer behave as insulators, conductors, and transistors (switches)
- Multiple metal layers on the silicon enable connections between transistors
- The wafer is chopped into many dies – the size of the die determines yield and cost

Processor Technology Trends

- Shrinking of transistor sizes: 250nm (1997) → 130nm (2002) → 70nm (2008) → 35nm (2014)
- Transistor density increases by 35% per year and die size increases by 10-20% per year… functionality improvements!
- Transistor speed improves linearly with size (complex equation involving voltages, resistances, capacitances)
- Wire delays do not scale down at the same rate as transistor delays

Memory

- RAM: volatile, faster, costlier
  - IBM-PC: 640KB
  - Now: 16GB or …
- Disk: non-volatile, slower, cheaper
  - IBM-PC:
    - Hard-disk: 10MB, Tape → 1TB+128GB
    - Floppy-disk: 5¼″(360KB～1.2MB), 3½″(1.4MB)
  - Now:
    - Hard-disk: …
    - Flash-disk: 1 GB
    - Flash

Memory and I/O Technology Trends

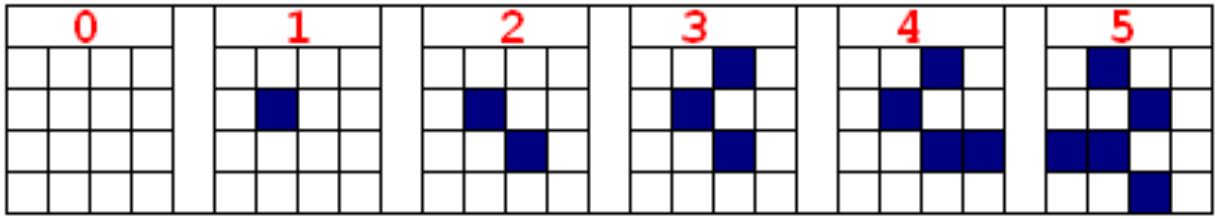
- DRAM density increases by 40-60% per year, latency has reduced by 33% in 10 years (the memory wall!), bandwidth improves twice as fast as latency decreases

- Disk density improves by 100% every year, latency improvement similar to DRAM

- Networks: primary focus on bandwidth; 10Mb → 100Mb in 10 years; 100Mb → 1Gb in 5 years

Input & Output

- Input
  - Keyboard
  - Mouse
  - Scanner, …
- Output
  - Display
  - Printer, …
- Input & Output
  - Disk, …

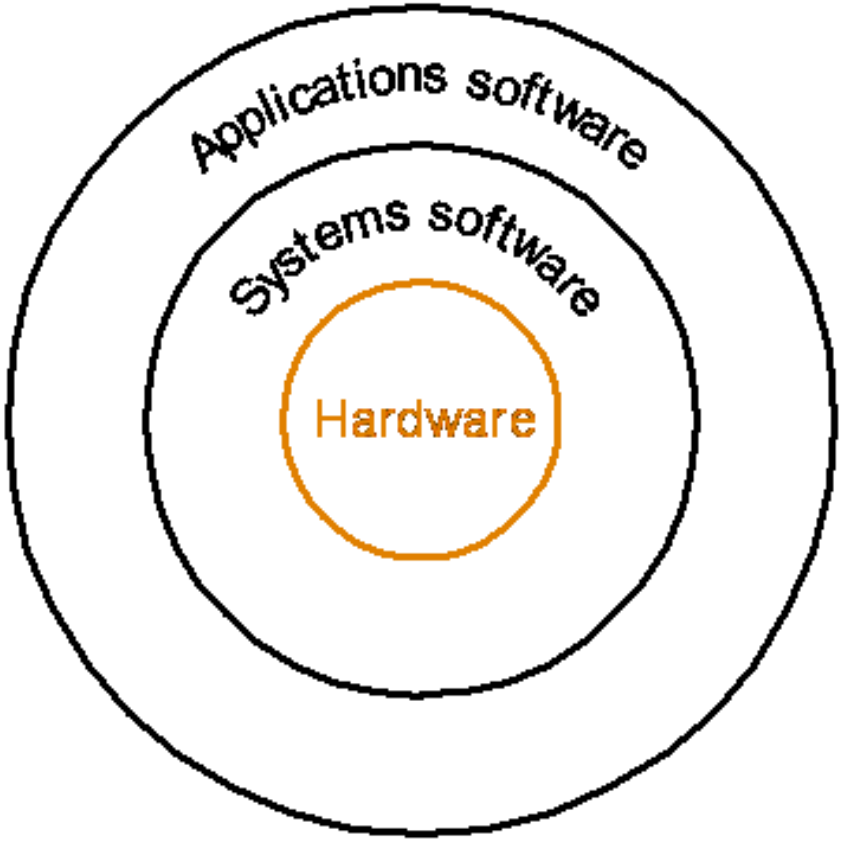


The diversity of I/O devices

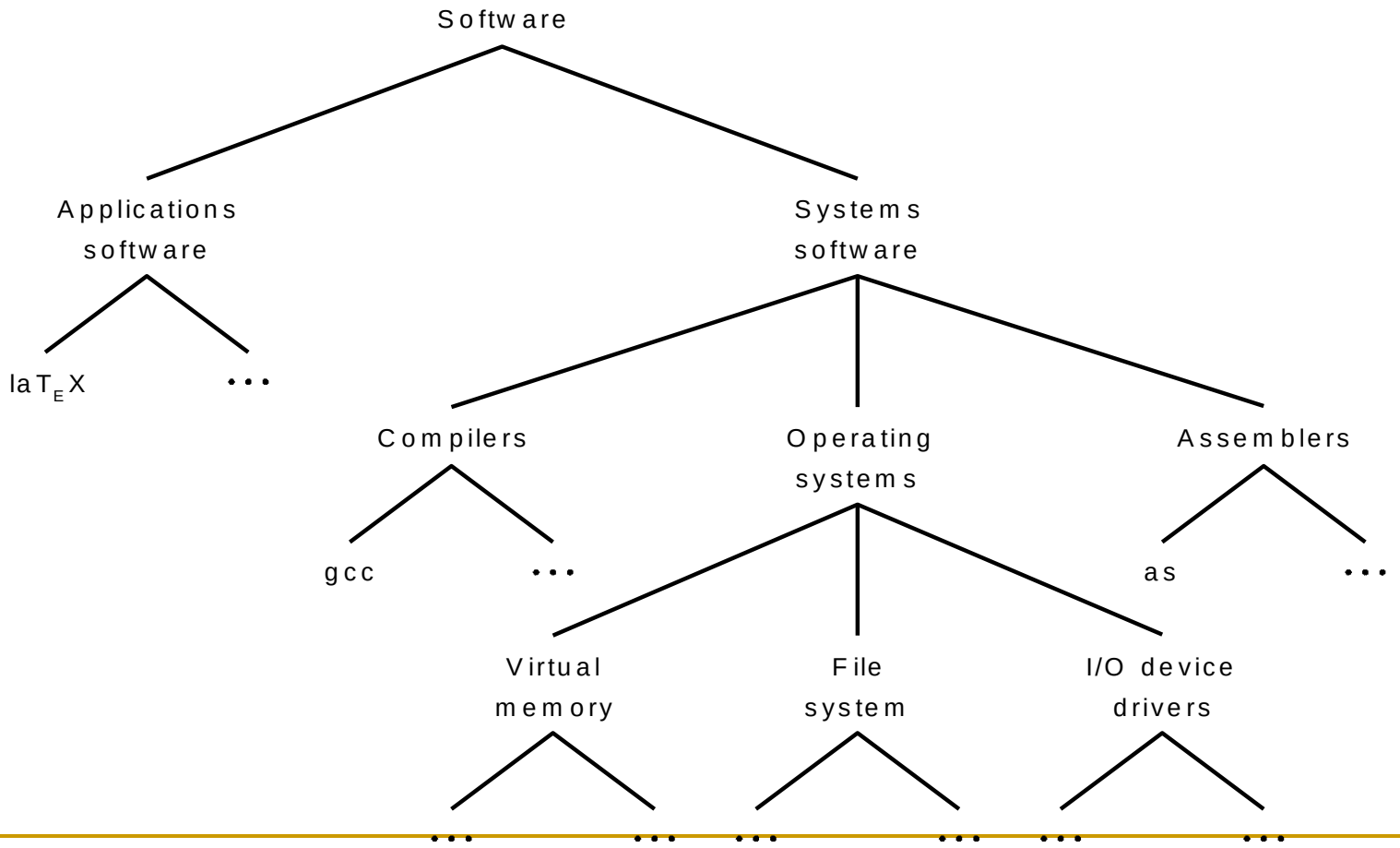
Device	Behavior	Partner	Data rate (KB/sec)
Keyboard	input	human	0.01
Mouse	input	human	0.02
Voice input	input	human	0.02
Scanner	input	human	400.00
Voice output	output	human	0.60
Line printer	output	human	1.00
Laser printer	output	human	200.00
Graphics display	output	human	60,000.00
Modem	input or output	machine	2.00-8.00
Network/LAN	input or output	machine	500.00-6000.00
Floppy disk	storage	machine	100.00
Optical disk	storage	machine	1000.00
Magnetic tape	storage	machine	2000.00
Magnetic disk	storage	machine	2000.00-10,000.00

Part 2:

Software



An example of the decomposability of computer systems



The HW/SW Interface

Application software

Systems software
(OS, compiler)

Hardware

```
a[i] = b[i] + c;
```

Compiler

```
lw    $15, 0($2)
add   $16, $15, $14
add   $17, $15, $13
lw    $18, 0($12)
lw    $19, 0($17)
add   $20, $18, $19
sw    $20, 0($16)
```

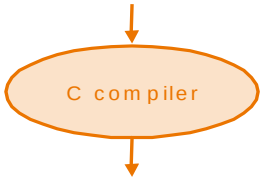
Assembler

```
00000010110000
01101000010010
```

The process of compiling and assembling

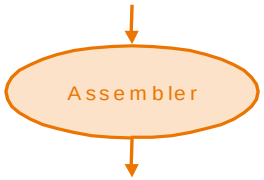
High-level language program (in C)

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



Assembly language program (for MIPS)

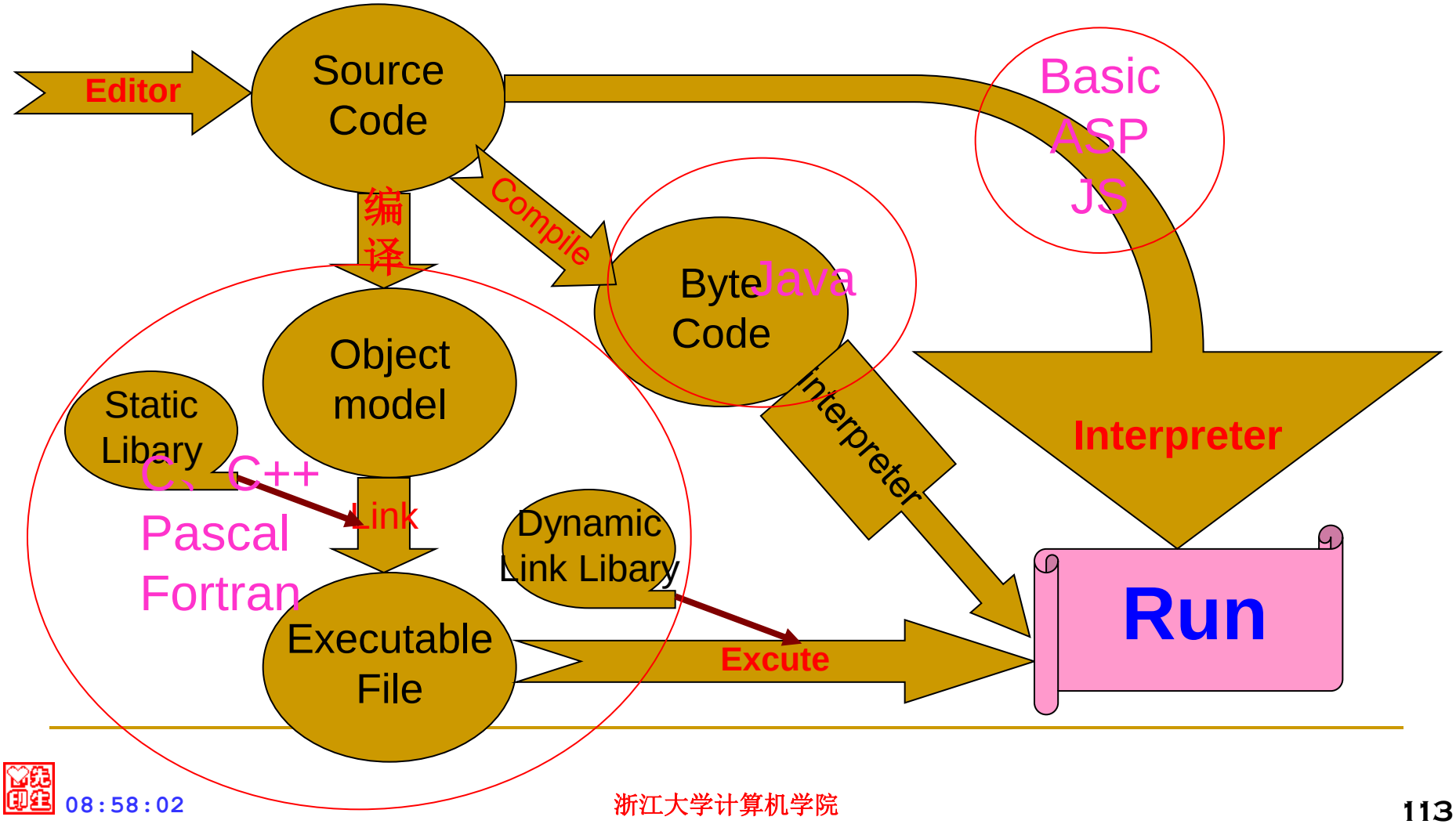
```
swap:
    muli $2, $5,4
    add  $2, $4,$2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```



Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

程序执行


Editor
Source Code
编译
Compile
Basic
ASP
JS
Byte Code
Java
Object model
Static Libary
C, C++
Pascal
Fortran
Link
Interpreter
Interpreter
Dynamic Link Libary
Executable File
Excute
Run

Java

```
public class Fruit
{
  public static void main(String args[])
  {
    // Declare and initialize three variables
    int numOranges = 5;                    // Count of oranges
    int numApples = 10;                    // Count of apples
    int numFruit = 0;                      // Count of fruit

    numFruit = numOranges + numApples;     // Calculate the total fruit

    // Display the result
    System.out.println("A totally fruity program");
    System.out.println("Total fruit is " + (numFruit +22));
  }
}
```

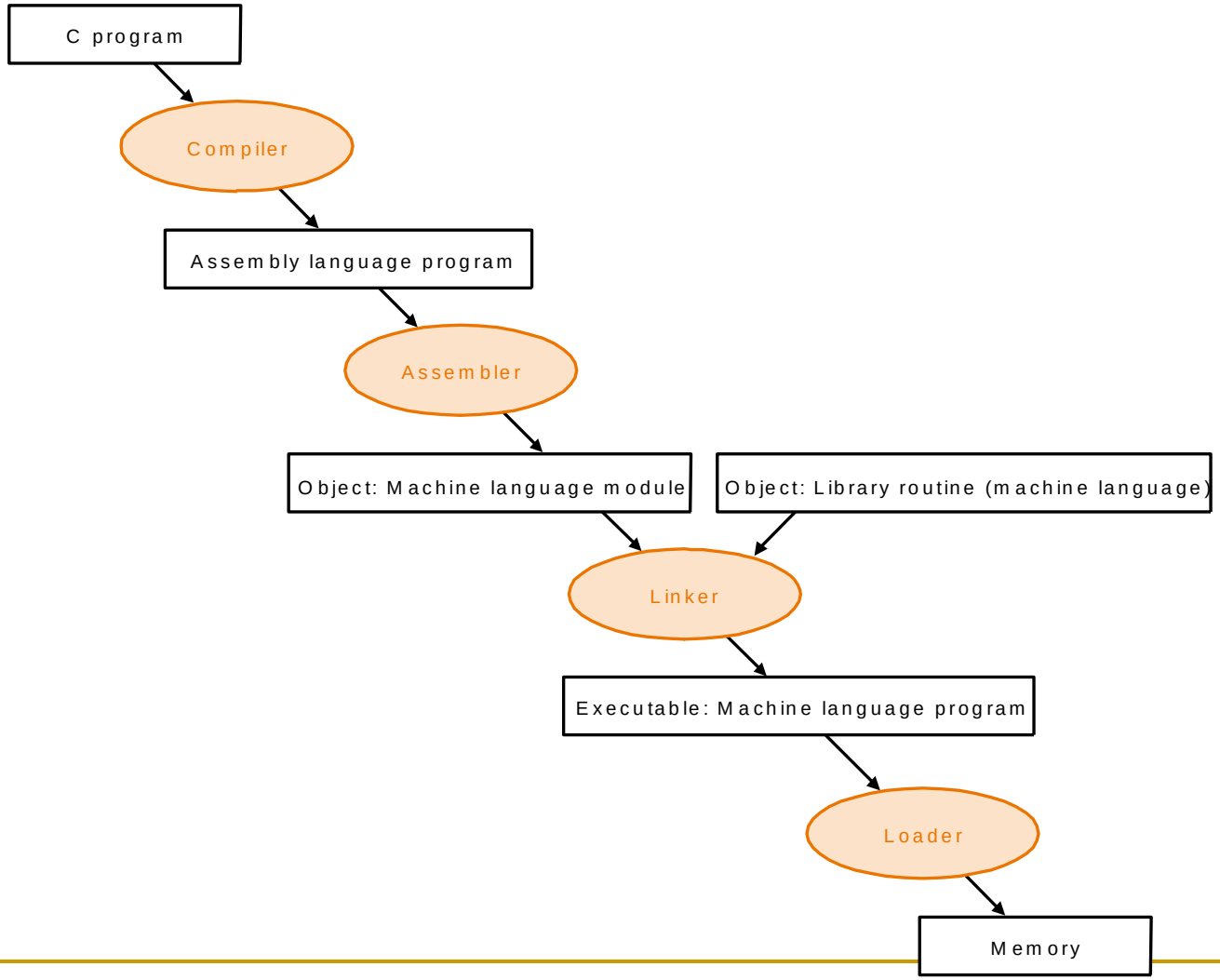
Start a C program in a file on disk to run 1

- Compiling
  - C program → assembly language program
- Assembling
  - Assembly language program → machine language module
- Linking
  - Object modules (including library routine) → executable program
  - 6 pieces of the object file for Unix systems:
    - object file header
    - text segment
    - data segment
    - relocation information
    - symbol table
    - debugging information
  - Place code and data modules symbolically in memory
  - Determine the addresses of data and instruction labels
  - Patch both the internal and external references

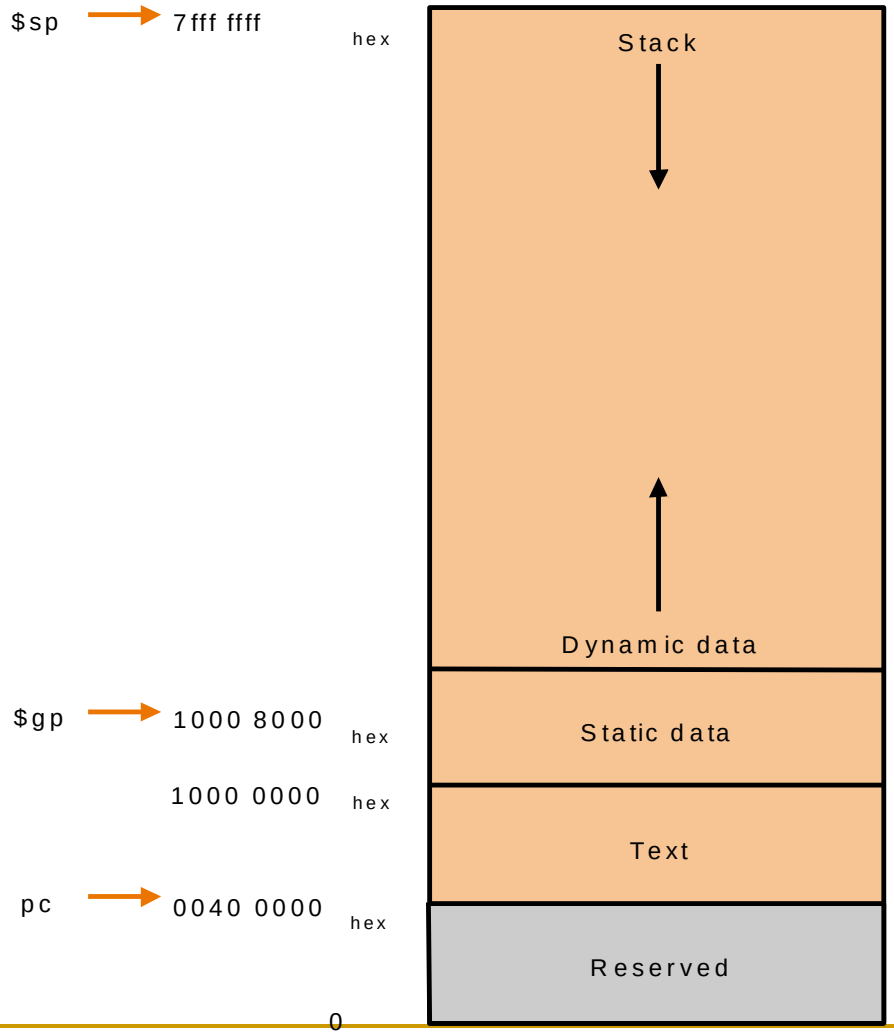
Start a C program in a file on disk to run 2

- loading
  - Determine size of text and data segments
  - Create an address space large enough
  - Copy instructions and data from executable file to memory
  - Copy parameters (if any) to the main program onto the stack
  - Initialize registers and set $sp to the first free location
  - Jump to a start-up routine

A translation hierarchy



MIPS memory allocation for program and data




Thank You!
非常天空
浙江大学计算机学院

成绩

- 考试：70%(不低于40分)
  - 英文
  - 闭卷
- 平时：30%
  - 作业
  - 期中
  - 点名
- 额外：
  - 课堂练习(+5%)：每题1分