汇编语言程序设计

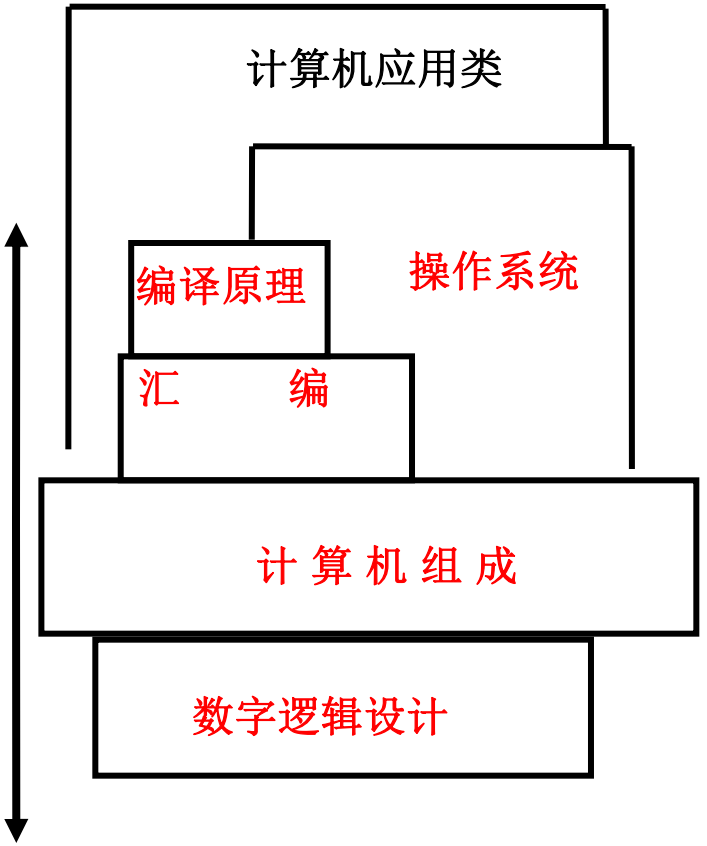
汇编语言与计算机系统结构

- 课程定位
- 主要授课内容
- 各类指令集简介

课程定位

与计算机组成原理、编译原理、操作系统、数字逻辑设计等组成计算机系统核心课程

- 汇编语言程序设计与计算机组成原理作为软硬件界面起到“承上启下”的作用



为后续课程打下指令集、汇编编程以及微体系结构入门的基础。

机器语言：依赖于机器的低级语言，书写格式为二进制代码

优点：执行速度快，效率高

缺点：表达的意义不直观，编写、阅读、调试较困难

汇编语言：是一种符号语言，与机器语言一一对应；它使用助记符表示相应的操作，并遵循一定的语法规则

优点：面向机器编程，在“时间”和“空间”上效率

缺点：涉及硬件细节，**要求熟悉计算机系统的内部结构**

高级语言：面向人的语言，表达形式接近自然语言

优点：便于阅读，易学易用，不涉及硬件，具有通用性

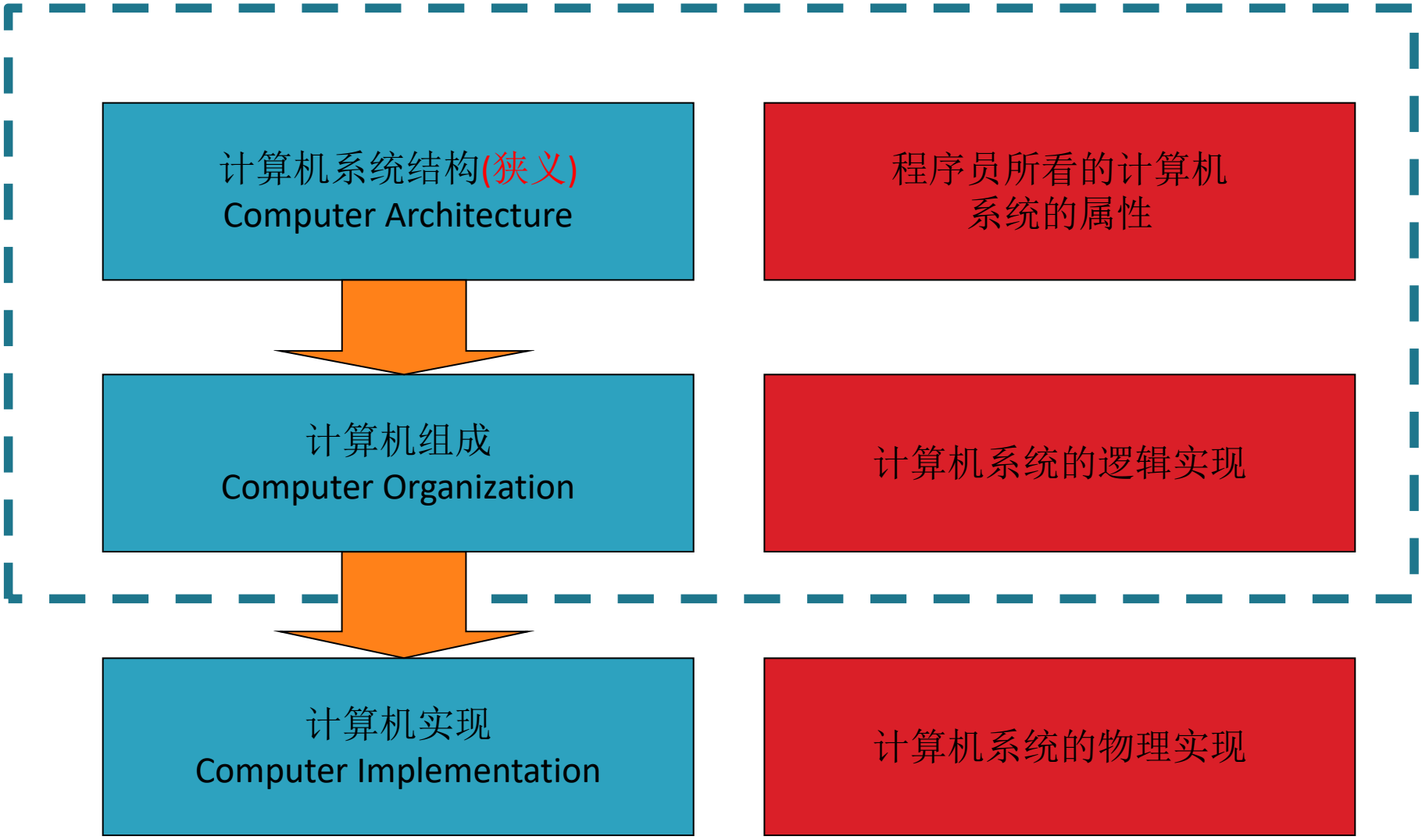
缺点：目标代码冗长，占用内存多，从而执行时间长，

，不能对某些硬件进行操作

- 编程语言是针对目标计算机系统结构的一种“抽象”
- 不同编程语言体现了对于系统结构的不同层面的“视角”

计算机系统结构

- 计算机系统结构（中国计算机科学技术百科全书的定义）
  - 计算机系统的物理或者硬件结构、各部分组成的属性以及这些部分的相互联系。
  - 系统软件开发人员看到的计算机系统的功能行为和概念结构
  - 计算机系统的结构与实现（计算机组成）



计算机系统结构是研究计算机系统自身的学科。

- 其它定义：对计算机系统中各级之间界面的划分和定义，以及对各级界面上、下的功能进行分配。
  - 1964年，IBM/360系列机的总设计工程师G.M.Amdahl、G.A.Blaauw、F.P.Brooks等人提出。也称体系结构。
  - 是从程序员的角度所看到的系统的属性，是概念上的结构和功能上的行为
    - 程序员：系统程序员（包括：**汇编语言**、机器语言、编译程序、操作系统）
    - 看到的：编写出能在机器上正确运行的程序所必须了解到的
  - 它不同于数据流程和控制的组织，不同于逻辑设计以及物理实现方法。

一般认为这是一种狭义的定义。

范　围(广义)

- **指令系统（ Instruction set architecture, or ISA ）**
  - **机器语言，还包括机器字长、内存地址模式、处理器寄存器等程序员可见的系统状态、数据格式等。**
- 微体系结构（Micro-architecture）
  - 如何实现指令集。
  - 处理器内部的实现，包括流水线、处理部件、缓存等内容。
- 计算机系统
  - 计算机系统里的其它硬件，包括总线、交换开关(Switch)、内存控制器、DMA控制器等。
- 其他内容
  - 虚拟化、计算机集群、 NUMA。

- 指令系统（汇编语言可以看做是它的一种助记符）
  - 计算机处理器对外提供的主要接口与规格
  - 软硬件的分界
  - 系统程序员看到的计算机的主要属性

- 指令系统分类
  - CISC （复杂指令系统，Complex Set Instruction Computer ）
    - 面向高级语言，缩小机器指令系统与高级语言语义差距
    - 指令条数多，寻址方式多变
    - 单条指令能够完成多个操作
    - 代表：X86

  - RISC（精简指令系统， Reduced instruction set computer）
    - 通常只支持常用的能在一个周期内完成的操作（80：20原则）
    - 简单而统一格式的指令格式与译码
    - 只有LOAD和STORE指令可以访问存储器，简单的寻址方式
    - 较多的寄存器

    - 指令条数相对较少，依赖于编译器产生高效的代码；
    - 处理器微体系结构相对简单，运行频率高。
    - [redacted] / PowerPC

CISC与RISC走向融合

- X86处理器内部采用类似RISC的micro-op
  - 出于兼容性考虑，其指令集一直属于CISC
- 经典的RISC指令集也日益扩展、复杂化
  - PowerPC指令集(RISC)有超过230多条指令，较许多CISC指令集更为复杂

Load / store with (without) other operations?

- MIPS是一个经典的RISC指令集，兼具RISC设计的简洁优雅与不足
  - 代码密度较低
  - 应用于嵌入式领域，32bit指令有些“大材小用”
  - 因此注重扩展，包括提高代码密度（16位指令）以及多媒体、加密领域的指令扩展
- ARM指令集介于经典的RISC与CISC之间，相对复杂
  - 注重代码密度，降低功耗
  - 浮点较弱，逐步扩展强化

主要授课内容

- 基本知识
  - 各类指令集初步
  - 数制与整数表示
  - 浮点数表示
- X86汇编
  - 80x86计算机组织与保护模式
  - X86指令系统与寻址方式
  - C与X86汇编
  - X86汇编语言程序格式与基本编程
- MIPS汇编
  - MIPS计算机组织初步
  - 指令系统介绍
  - 汇编代码与异常处理

学习目标与要求：

- 了解汇编语言与计算机系统结构的关系及其起到的作用
- 了解以 X86系列微处理器为基础的PC机的编程结构，建立起“机器”与“程序”、“时间”与“空间”的概念
- 基本掌握 X86系列微处理器的指令系统及寻址方式，能够编写程序
- 掌握C语言基本代码段与汇编语言的对应关系

- 掌握MIPS指令集的指令类型，初步了解对应的系统结构
- 掌握撰写MIPS异常处理句柄的方法
- 初步了解MIPS指令集在操作系统中的作用*

参考书:

1. 《深入理解计算机系统》（Computer Systems： A Programmer‘s Perspective），第二版，2010
   - 第2、3章。
   - 课程ppt的部分素材也来源于该书作者的课程网站。

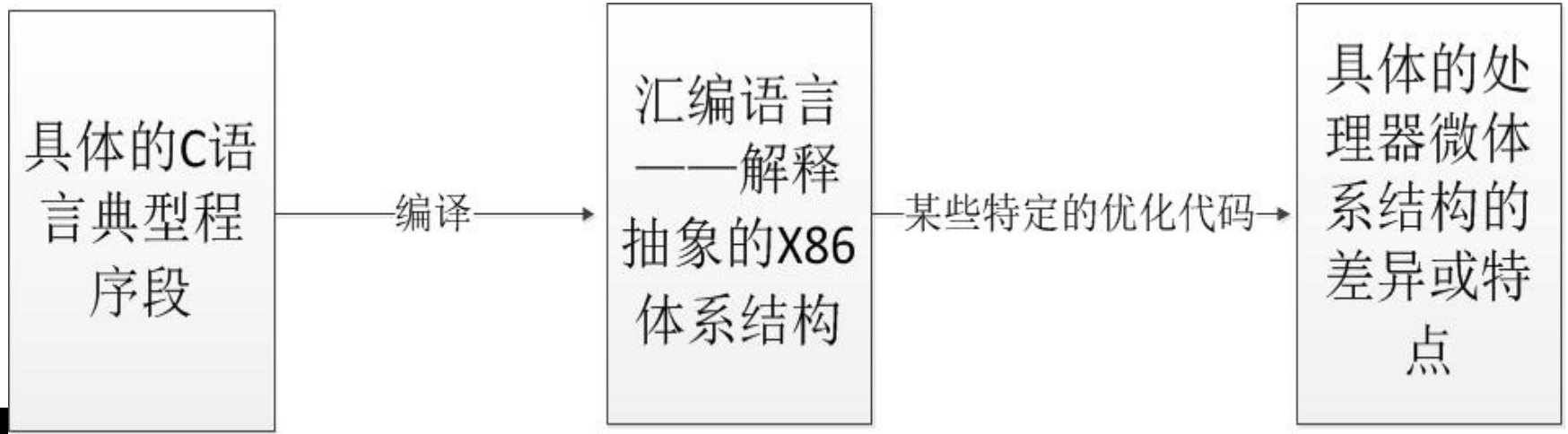
2. SEE MIPS RUN （MIPS体系结构透视）. 第二版，2008.

3. IBM-PC汇编语言程序设计 （第2版） 沈美明 温冬婵 编著 清华大学出版社

授课角度

- 突出了在课程组中的“承上启下”作用，在内容编排上突出了与相关课程的衔接
  - 强化与高级语言的联系，从典型的C语言代码段入手，通过编译成汇编代码来详细解释程序员角度的微体系结构（包括X86与MIPS结构）运行模型。
  - 汇编语言是高级语言在机器层面的表示，掌握这两种语言的对应可以将程序的执行与计算机的工作过程紧密联系起来.
  - 通过对不同汇编代码的解释来给出微体系结构方面的差异，为后续课程，如编译原理、计算机组成原理等提供一些先导知识。
- 开展多角度内容组织，系统角度与应用角度并存
  - 从C语言角度、处理器结构角度分别“自上而下”、“自下而上”的以纵向角度讲解汇编语言的语义及其某些新的汇编指令出现的原因。
  - [redacted]的特点以及“反汇编”应用。

- 从典型的C语言代码段入手，通过编译成汇编代码来详细解释程序员角度的X86结构运行模型
  - 可以给编译给出一些先导知识
- 再通过对不同汇编代码的解释来稍进一步的给出微体系结构方面的差异
  - 为后续的计原等课程给出一些先导知识



- 举例：条件移动指令

```
int absdiff(
    int x, int y)
{
    int result;
    if (x > y) {
        result = x-y;
    } else {
        result = y-x;
    }
    return result;
}
```



```
_absdiff:
        pushl   %ebp
        movl    %esp, %ebp
        movl    8(%ebp), %ecx
        pushl   %ebx
        movl    12(%ebp), %edx
        movl    %ecx, %eax
        movl    %edx, %ebx
        subl    %edx, %eax
        subl    %ecx, %ebx
        cmpl    %edx, %ecx
        cmovle  %ebx, %eax
        popl    %ebx
        popl    %ebp
        ret
```

march=i686

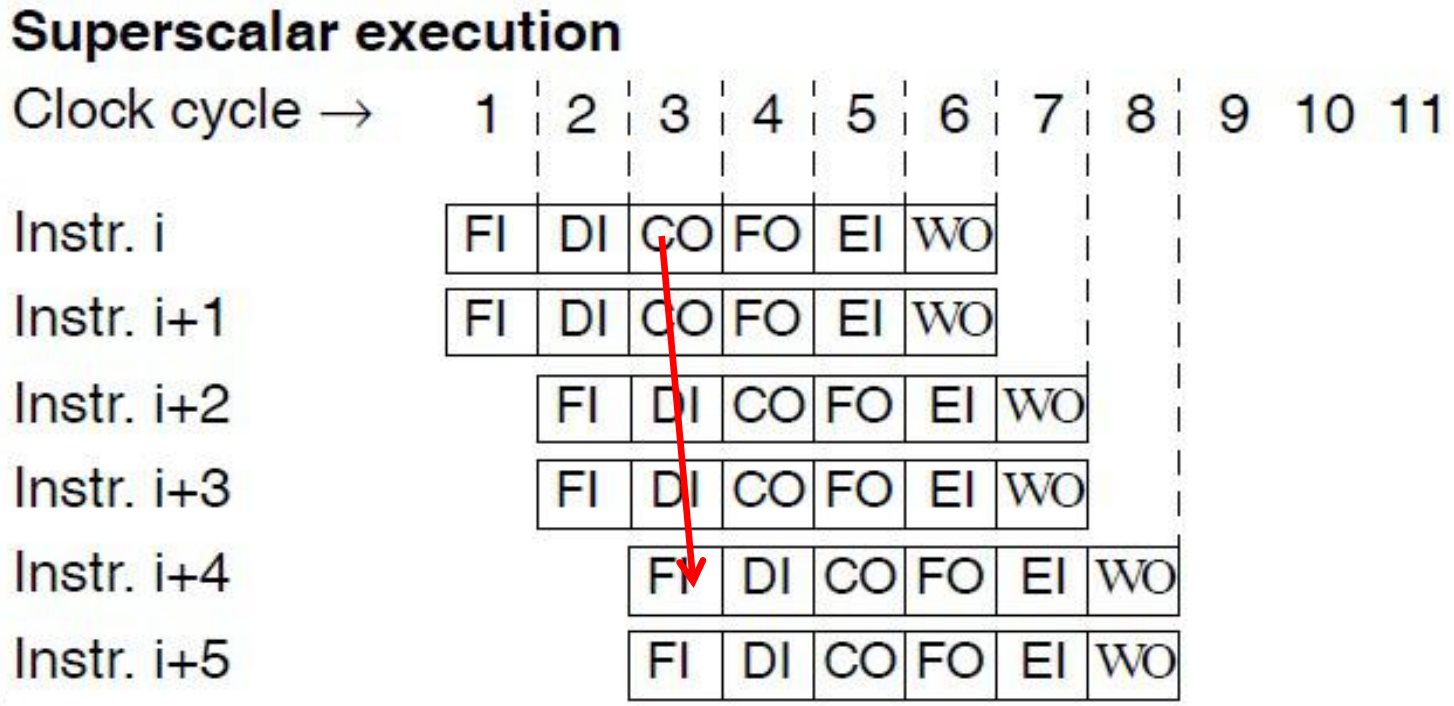
对照

```
_absdiff:
        pushl   %ebp
        movl    %esp, %ebp
        movl    8(%ebp), %edx
        movl    12(%ebp), %eax
        cmpl    %eax, %edx
        jle     L27
        popl    %ebp
        subl    %eax, %edx
        movl    %edx, %eax
        ret
        .p2align 4,,7
L27:
        popl    %ebp
        subl    %edx, %eax
```

- 从处理器角度来解释
  - 首先较新的X86处理器增加了条件执行（移动）指令
  - 为何增加
    - 从处理器的长流水线、多发射角度来解释条件跳转指令对性能可能带来的负面影响
    - 尽量消除条件跳转指令

Δ微体系结构背景

现代的通用处理器 支持深度流水线以及多发射结构，如
Pentium 4： >= 20 stages, up to 126 instructions on-fly



条件指令往往会引起一定的性能损失，因此需要尽量消除。