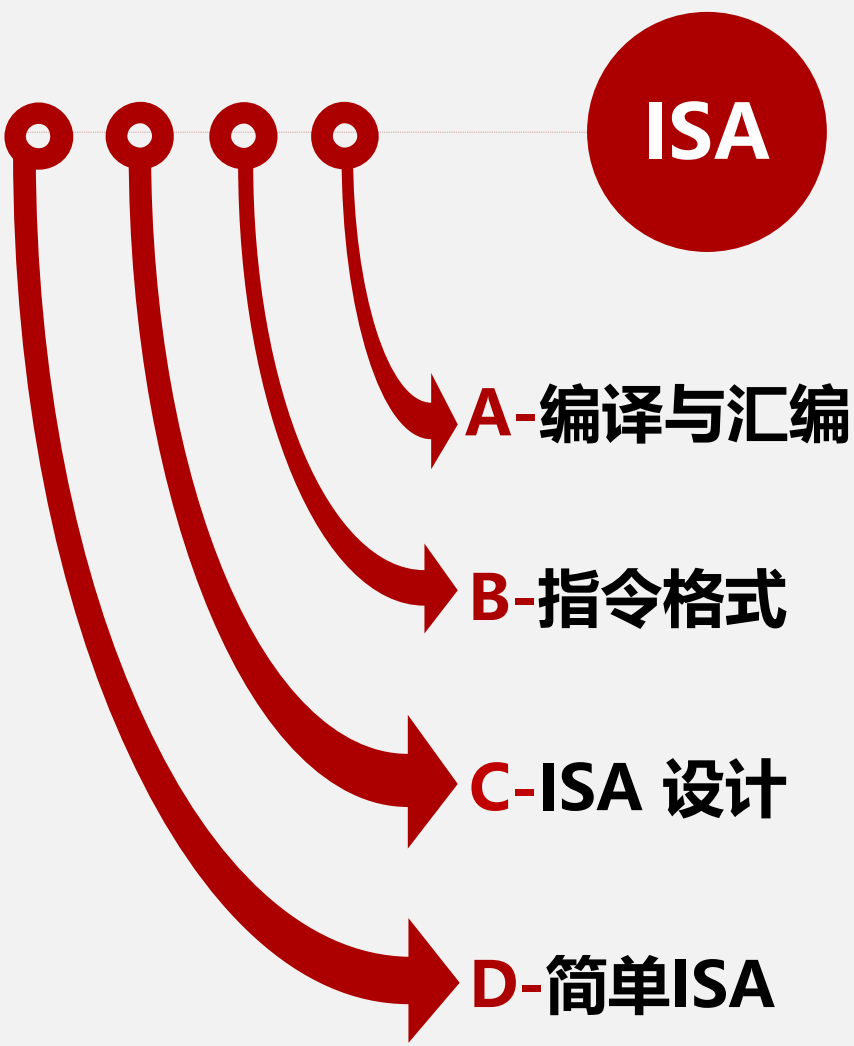
《计算机系统》

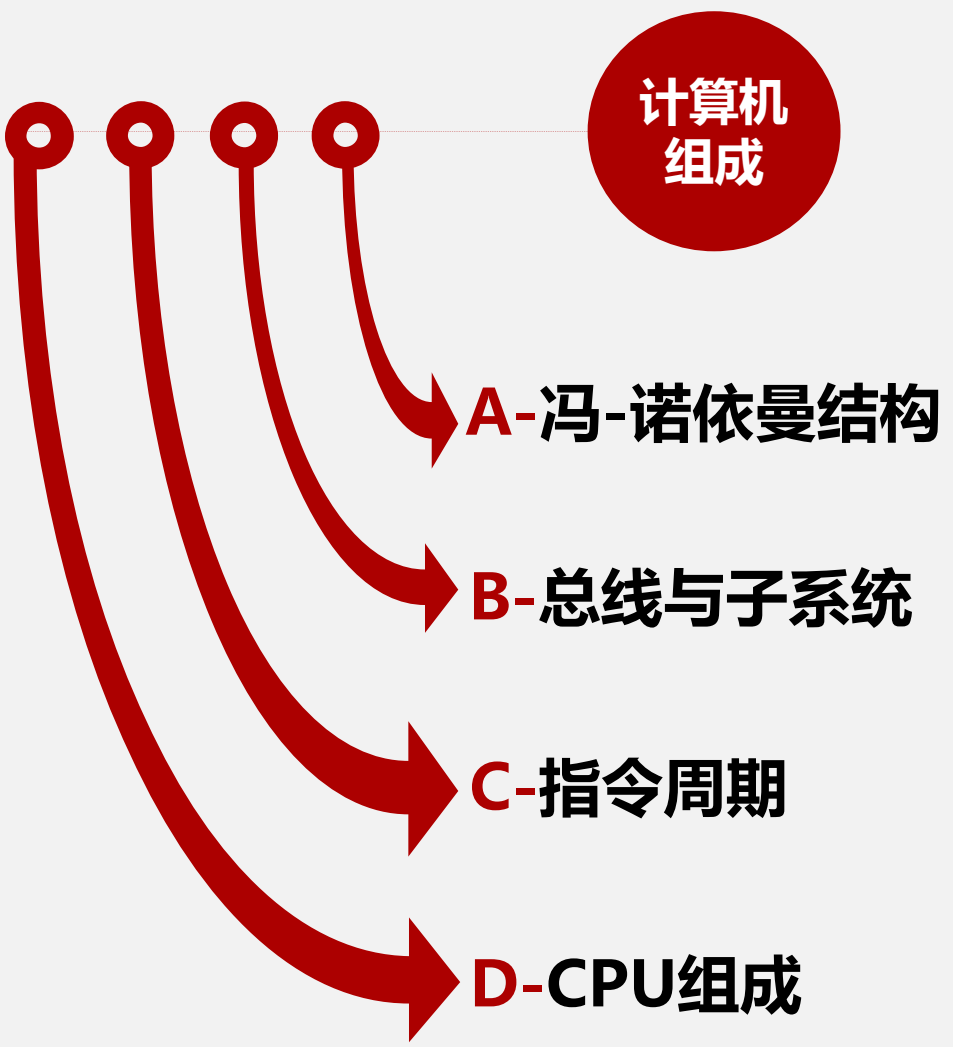
指令集结构与计算机组成

湖南大学

《计算机系统》课程教学组

提要


ISA
A-编译与汇编
B-指令格式
C-ISA 设计
D-简单ISA


计算机组成
A-冯-诺依曼结构
B-总线与子系统
C-指令周期
D-CPU组成

指令集结构

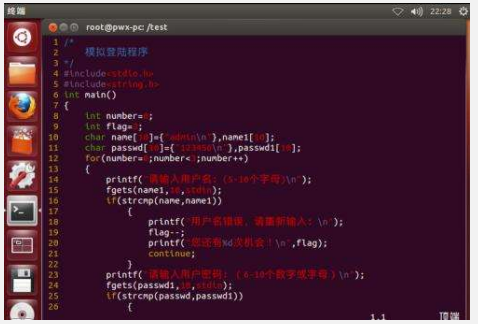
指令集结构(ISA, Instruction Set Architecture)，是微处理器的接口，包含了与该微处理器进行交互所需要的信息，但并不涉及微处理器自身如何设计和实现的细节。

ISA包括：

- 微处理器的指令集
- 程序员可直接访问的寄存器的细节
- 访问内存所需的信息
- 微处理器如何响应中断

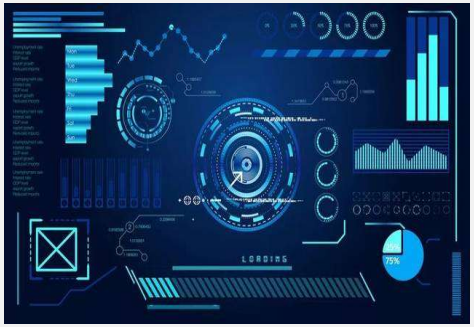
计算机语言

人与计算机的交流，或者说人给计算机布置任务，是通过计算机语言来实现的。

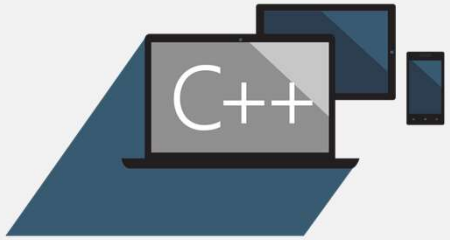


基本分类：

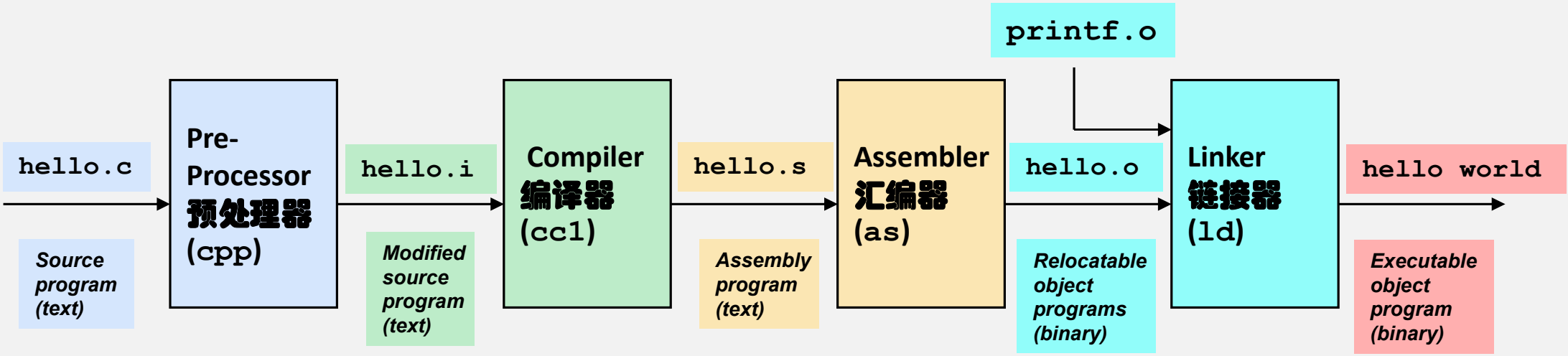
- **高级语言**-与平台无关
- **汇编语言**-因处理器而异
- **机器语言**-因指令集而异



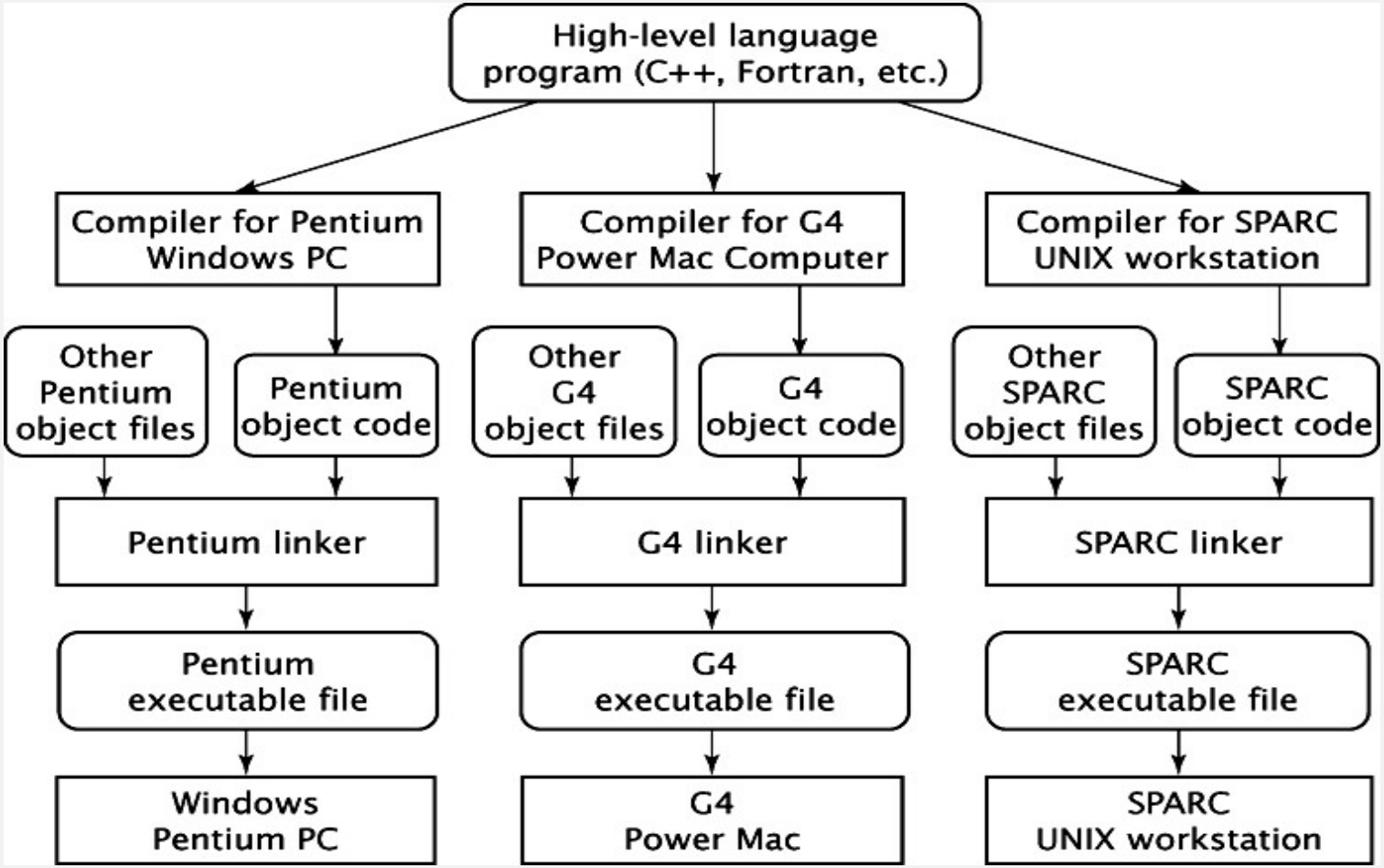
编写程序需要遵循一定的字符及语法规则



编译与汇编

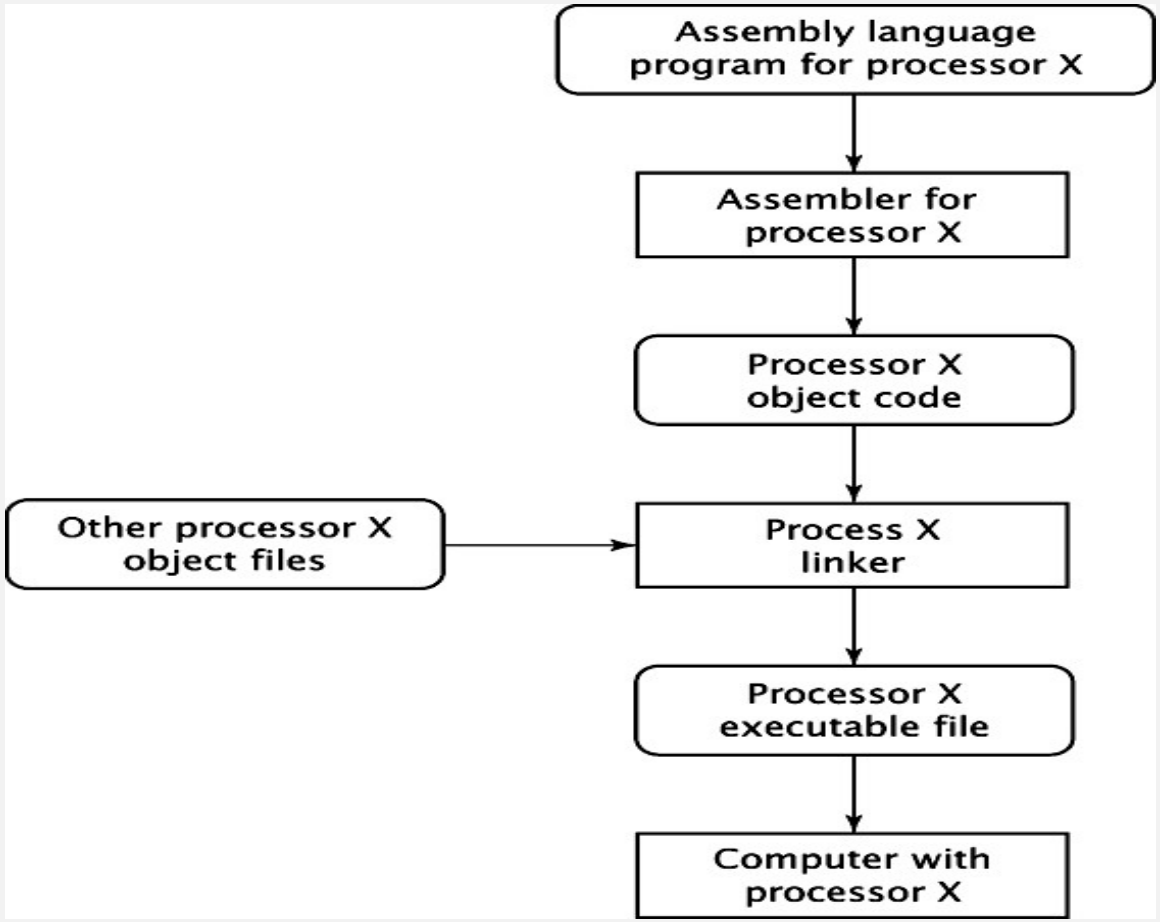

hello.c
Pre-Processor
预处理器
(cpp)
hello.i
Compiler
编译器
(cc1)
hello.s
Assembler
汇编器
(as)
printf.o
hello.o
Linker
链接器
(ld)
hello world
Source program (text)
Modified source program (text)
Assembly program (text)
Relocatable object programs (binary)
Executable object program (binary)

编译过程



同一高级语言源代码可以经过编译在不同的微处理器和操作系统或者计算平台上运行。

汇编过程



每一种汇编语言对应一种微处理器，不需要针对不同平台的汇编器

高级语言与汇编语言

二者的链接与装载过程相同

汇编器比编译器简单

- 高级语言一条语句可能有多种有效的转换
- 汇编语言指令都唯一对应一种机器代码指令

二者的使用

- 个人计算机上的软件大多采用高级语言编写
- 二者也常常一起形成最优化代码
- 小型计算机系统的代码，例如微波炉控制器，一般采用汇编语言编写

汇编语言

运算类指令举例

```
addl %ebx, %eax
sarl $3, %ebx
xorl $1, %eax
```

传送类指令举例

```
movl $4, %eax                    movl $value, %eax
movl $8(%edx, %ebx, 2), %eax
push %eax                        pop %ebx
```

控制类指令举例

```
jmp .L4
jne .L8
loop .L6  (相当于：%edi+1, cmp %edi, maxvalue, jne .L6)
```

数据类型	
数值数据	无符号整型数、有符号整型数、浮点数据
布尔数据	数据值常以0表示FALSE，以非0表示TRUE
字符数据	字符编码标准（ASCII、EBCDIC、UNICODE、或其它）

指令格式

- 指令代码（Instruction Code）：汇编语言指令转换成对应的机器代码，以二进制数值的形式表现。
- 位的不同分组代表着指令的不同部分。某一组可能代表要执行的操作（操作码），而其它的组则选取操作的操作数。

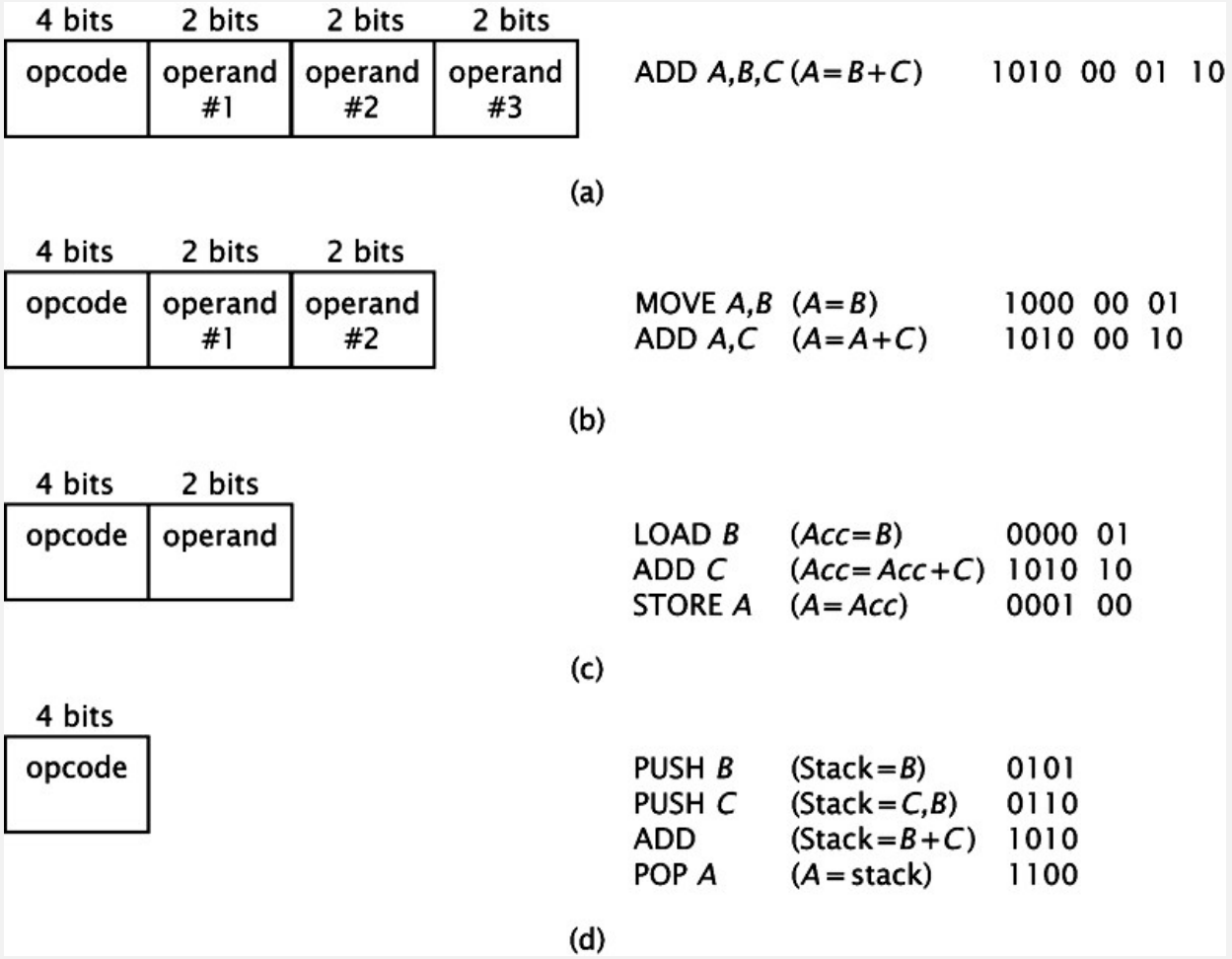
指令格式

考虑一个简单的例子：A = B + C。

- 该指令有一个操作——加法，两个源操作数——B和C，和一个目的操作数——A。
- 如果微处理器可以执行16种不同的操作，加法是其中的一种，那么它需要4位来代表其中的操作（因为$2^4 = 16$）。这里，我们假设位模式1010代表加法。
- 而且假定对于这种操作仅有4种可能的操作数——A，B，C和D。微处理器需要两位来表示每一种操作数。对于本例，00—A，01—B，10—C，11—的D。

指令格式

微处理器可以设计成能运行具有3、2、1或0个操作数的指令。



(a)三操作数，(b)二操作数，(c)单操作数，(d)零操作数

指令格式

短指令

优点：
- 指令译码电路简单
- 每条指令可在一个存储单元或存储字内
- 一次内存访问完成取指令操作

缺点：
- 一条指令中包含的信息量小
- 源程序和目标代码较长
- 程序设计难度大

长指令

缺点：
- 指令译码电路复杂
- 每条指令需要多个存储单元或存储字
- 需多次内存访问完成取指令操作

优点：
- 一条指令中包含的信息量大
- 源程序和目标代码较短
- 程序设计难度小

指令集结构设计

设计指令集结构并没有万能公式可言。相同的需求会产生不同的ISA设计

在设计ISA规范时，必须在性能、大小和成本限制间有个评估折衷。

完整性问题

- 该指令集是否有足够的指令可以让程序完成它必须的任务?

正交性问题

- 如果指令不重叠或者不执行同样的操作，那么它们就是正交的。
- 在最小的指令集中为程序员提供必需的操作。

寄存器组

- 设计者可以通过指定寄存器用途等方式优化ISA。

指令集结构设计

要考虑的其它问题

- 处理器必须向下兼容其它的微处理器吗?
- 微处理器将处理何种类型和大小的数据?
- 需要中断吗? 中断向量
- 需要条件指令吗? 标志（flags)

简单指令集

相对简单的指令集结构举例

存储器模型

该微处理器可以访问64K（$=2^{16}$）字节的存储器（每字节8位）即64K×8的存储器。从外部设备输入数据和输出数据到外部设备，都可以被看作是访问内存，即存储器统一I/O。

寄存器

累加器AC、寄存器R、1位零标志Z。

16条指令

每一条都是8位指令代码，见表3.1（下页）。

简单指令集

Table 3.1

Instruction set for a Relatively Simple CPU （Γ为内存地址）

Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	$AC = M[\Gamma]$
STAC	0000 0010 Γ	$M[\Gamma] = AC$
MVAC	0000 0011	$R = AC$
MOVR	0000 0100	$AC = R$
JUMP	0000 0101 Γ	GOTO Γ
JMPZ	0000 0110 Γ	IF (Z=1) THEN GOTO Γ
JPNZ	0000 0111 Γ	IF (Z=0) THEN GOTO Γ
ADD	0000 1000	$AC = AC + R$, If $(AC + R = 0)$ Then $Z = 1$ Else $Z = 0$
SUB	0000 1001	$AC = AC - R$, If $(AC - R = 0)$ Then $Z = 1$ Else $Z = 0$
INAC	0000 1010	$AC = AC + 1$, If $(AC + 1 = 0)$ Then $Z = 1$ Else $Z = 0$
CLAC	0000 1011	$AC = 0$, $Z = 1$
AND	0000 1100	$AC = AC \wedge R$, If $(AC \wedge R = 0)$ Then $Z = 1$ Else $Z = 0$
OR	0000 1101	$AC = AC \vee R$, If $(AC \vee R = 0)$ Then $Z = 1$ Else $Z = 0$
XOR	0000 1110	$AC = AC \oplus R$, If $(AC \oplus R = 0)$ Then $Z = 1$ Else $Z = 0$
NOT	0000 1111	$AC = AC'$, If $(AC' = 0)$ Then $Z = 1$ Else $Z = 0$

简单指令集

- LDAC、STAC、JUMP、JMPZ和JPNZ指令都需要16位的存储地址作为操作数，此处用Γ表示。这些指令在存储器中每个都需要3字节。例如：

  25：JUMP 1234H

- 该指令以如下形式存储在存储器中：

  25：0000 0101（JUMP）

  26：0011 0100（34H）

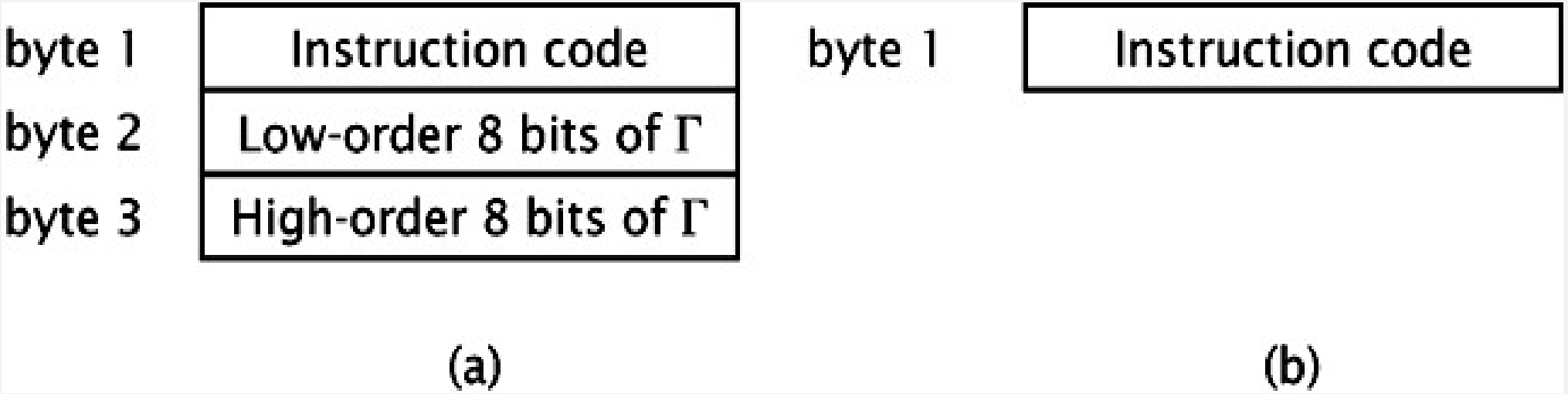
  27：0001 0010（12H）

	存储器
25：	0X05
26：	0X34
27：	0X12

注意：第二字节为低8位，第三字节为高8位

- 其他指令都只有操作码部分，在内存中只需一个字节。

简单指令集


byte 1
Instruction code
byte 2
Low-order 8 bits of Γ
byte 3
High-order 8 bits of Γ
(a)
三字节格式
byte 1
Instruction code
(b)
一字节格式

简单指令集

全部16条指令

数据传送：NOP、LDAC、STAC、MVAC 和 MOVR

程序控制：JUMP、JMPZ 和 JPNZ

数据运算：

算术指令—ADD、SUB、INAC 和 CLAC

逻辑指令—AND、OR、XOR 和 NOT

简单指令集

用法举例

编程计算1 + 2 + ... + n

```
total  = 0;
for i = 1 to n
do {total = total + i};
```

假设:

数值 n → 内存单元 10

求和次数 i → 内存单元 100

结果total → 内存单元 300

确定运算步骤如下:

```
1: total = 0, i = 0
2: i = i + 1
3: total = total + i
4: IF i≠n THEN GOTO 2
```

简单指令集

实现这一算法的相对简单CPU的代码如下：

```
        CLAC
        STAC    300
        STAC    100

Loop:   LDAC    100
        INAC
        STAC    100

        MVAC
        LDAC    300
        ADD
        STAC    300

        LDAC    10
        SUB
        JPNZ    Loop
```

total = 0， i = 0

(设 i 变量放在[100], total变量放在[300])

i = i + 1

total = total + i

if i ≠ n then goto Loop

(设 n 在 [10])

简单指令集

对于简单的应用程序来说，指令集完整。
如果一个应用程序需要使用浮点型数据，
它就不是十分完整。

04

寄存器组和寻址方式是它最大的弱点。

03

02

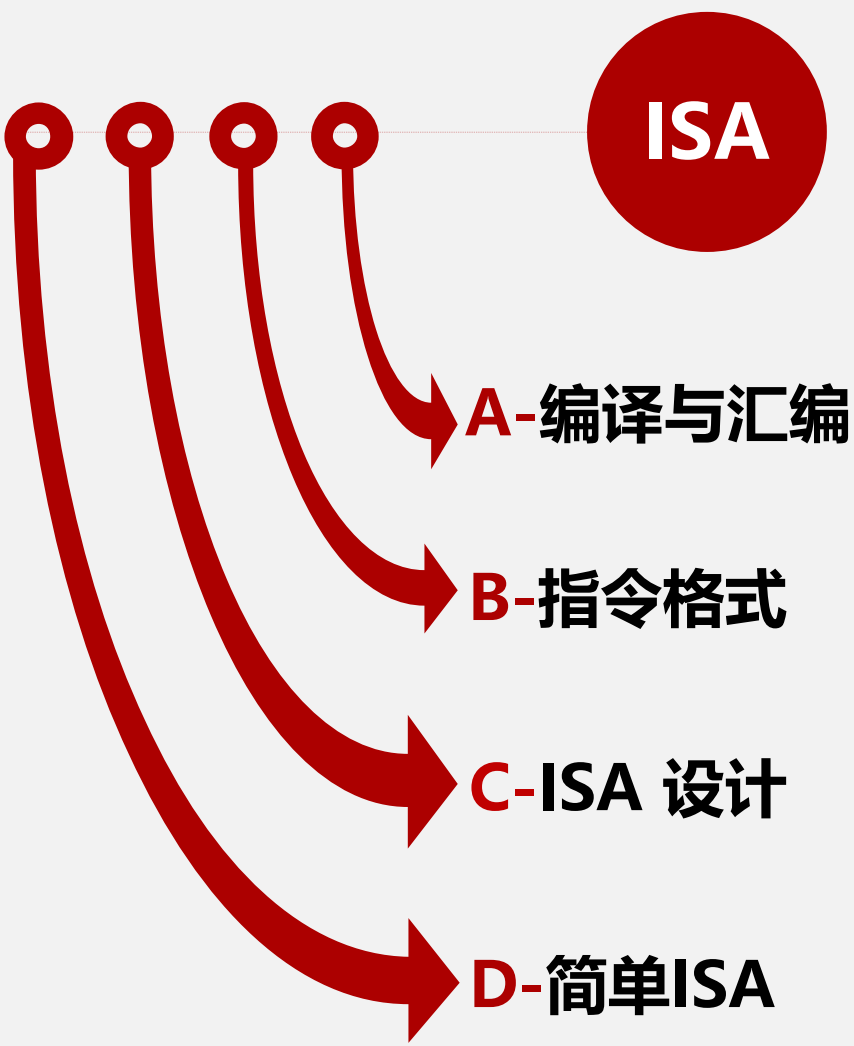
该指令集是相当于正交的。只是多了OR指令，但有时CPU的指令集不是完全正交更好一些。

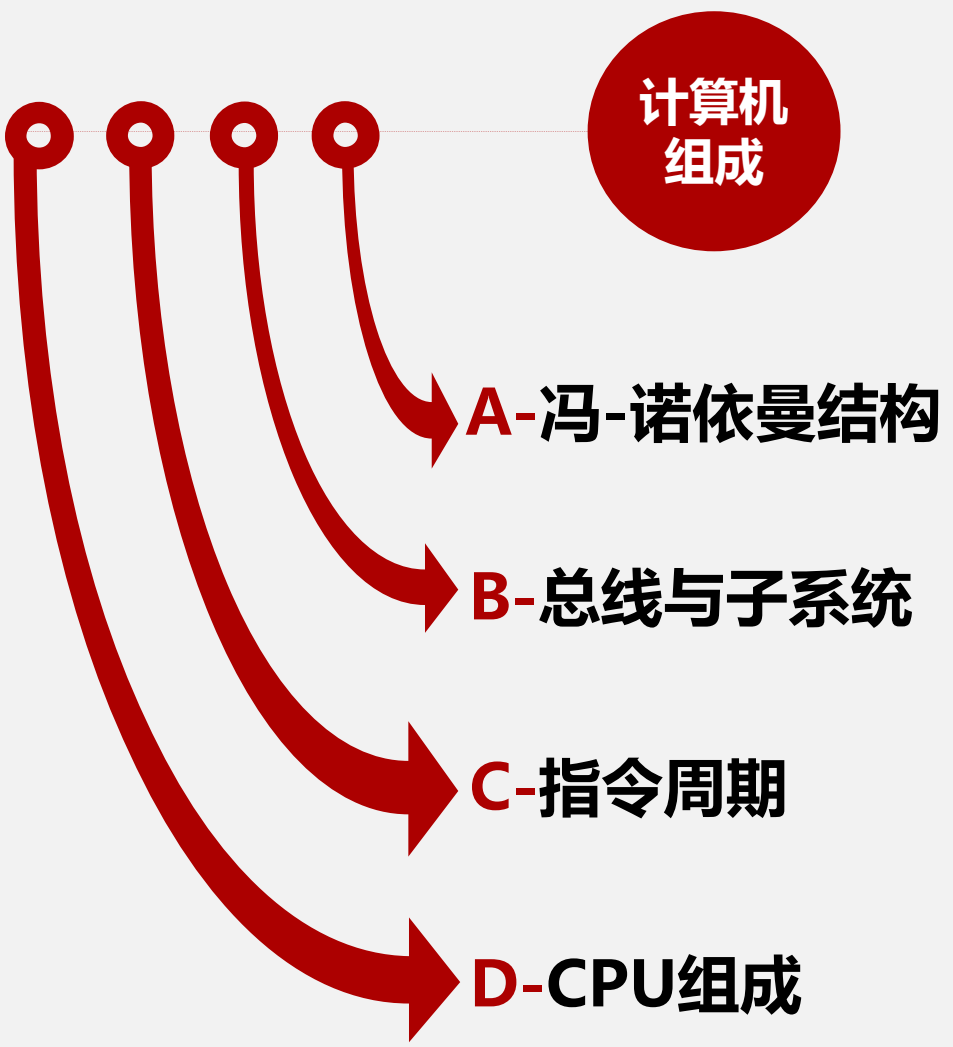
01

它满足了以教学为目的的设计目标。

怎样看待这个指令集结构?

提要

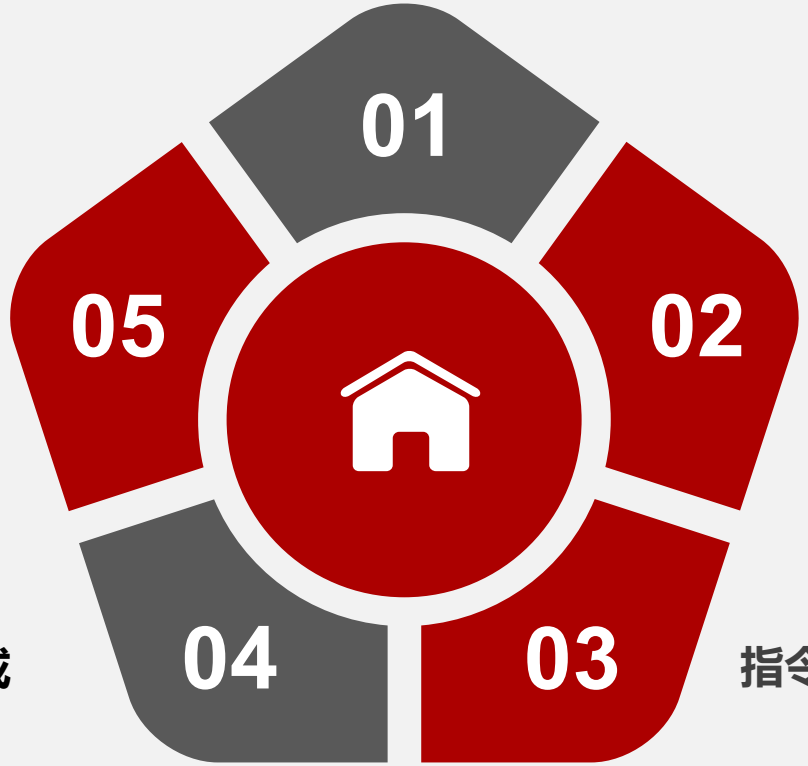

ISA
A-编译与汇编
B-指令格式
C-ISA 设计
D-简单ISA


计算机组成
A-冯-诺依曼结构
B-总线与子系统
C-指令周期
D-CPU组成

硬件基本组成

计算机由五大部件组成

运算器、控制器、存储器、输入设备、输出设备



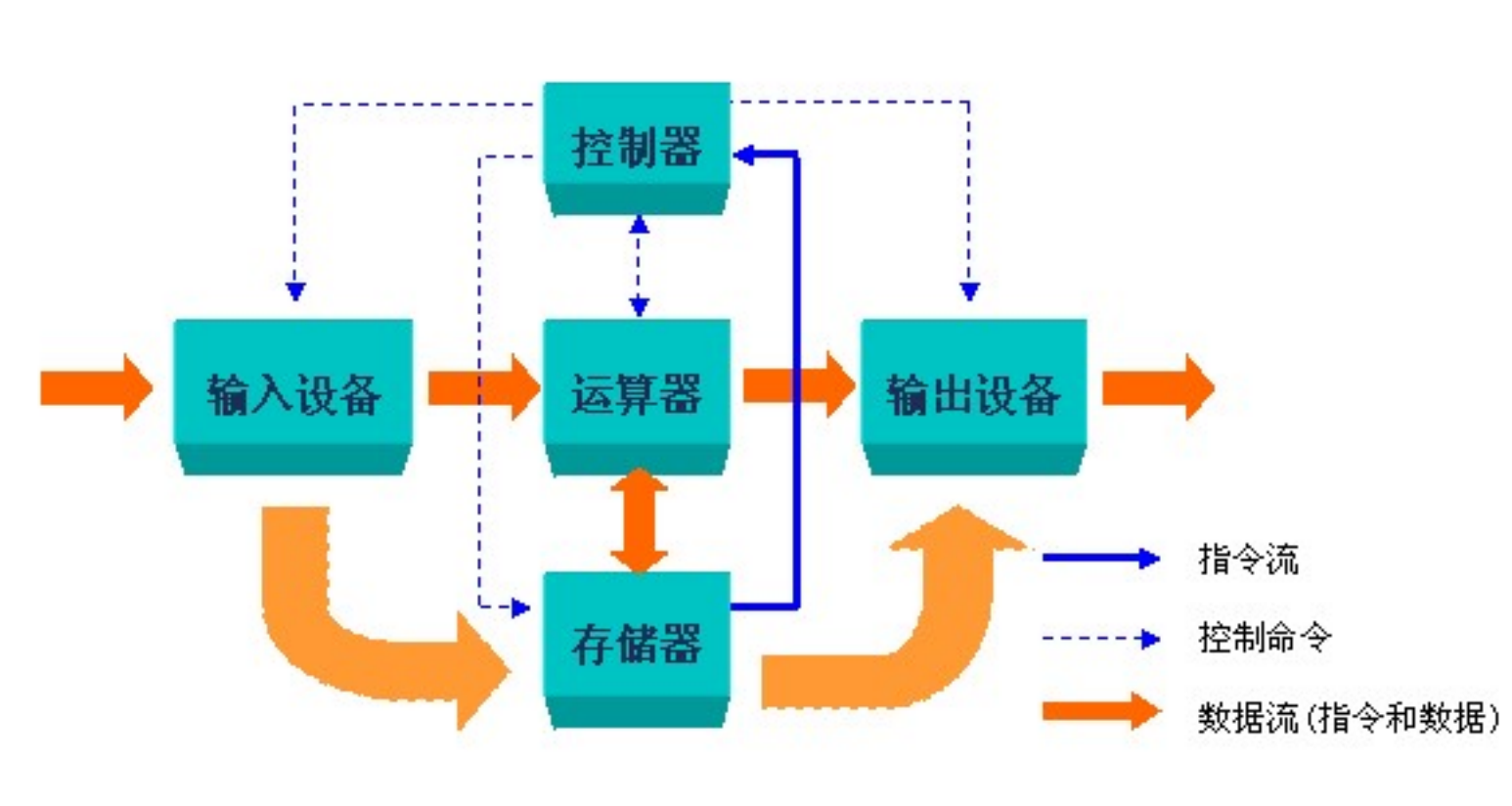
指令和**数据**以同等地位存于存储器，可**按地址寻访**。

指令和数据用**二进制**表示

指令由**操作码**和**地址码**组成

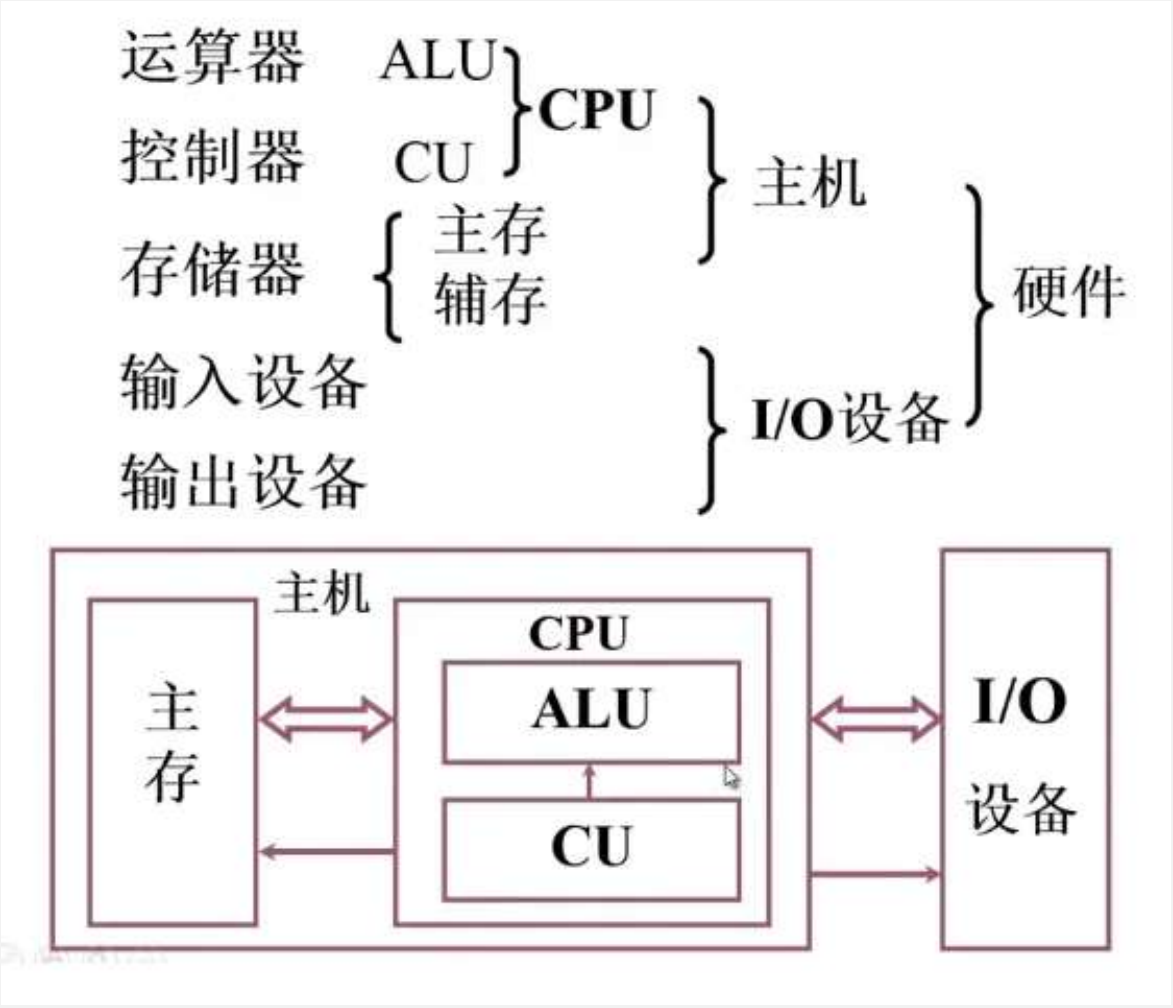
存储程序和**程序控制**

硬件基本组成



冯·诺依曼计算机硬件框图

基本工作原理



ALU 与 CU 共同构成CPU的核心部分。

存储器分为主存和辅存（缓存cache）

输入输出设备通过向CPU发送请求获得读写数据的权限以及相应的数据。

总线(BUS)

总线

一组传递数据的导线，是连接各个部件的信息传输线，是各个部件共享的传输介质。

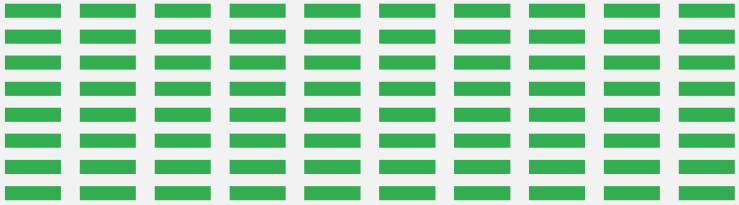
总线优势

较少的连接数量、较少的空间、更低的功耗、较少的引脚

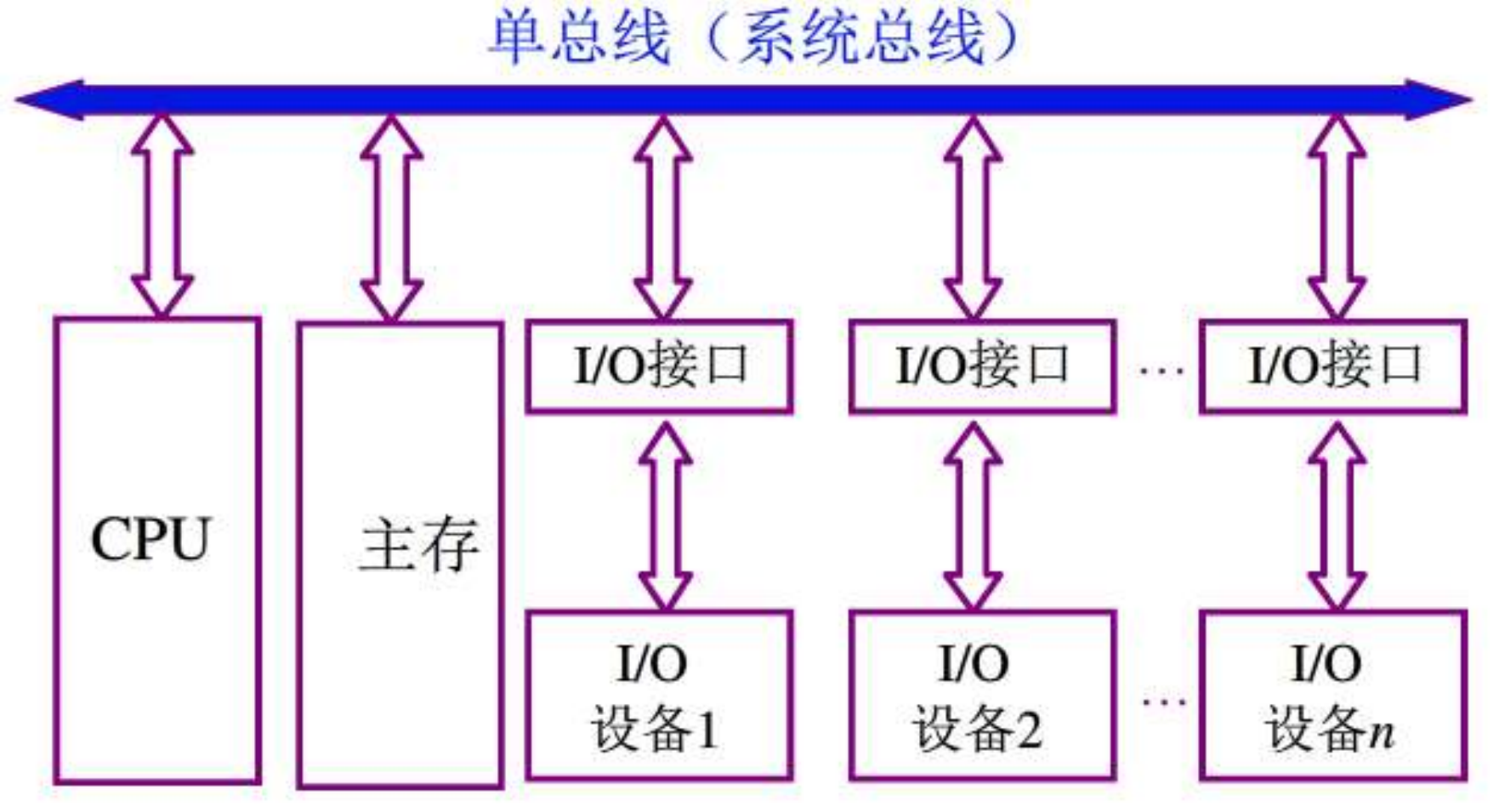
总线上信息的传送

串行

并行



总线(BUS)



总线(BUS)

总线分类

1. 片内总线 —— 芯片内部的总线

2. 系统总线

（计算机各部件之间的信息传输线）

- **数据总线**：双向 与机器字长、存储字长有关
- **地址总线**：单向 与存储地址、I/O地址有关
- **控制总线**：出（中断与总线请求）
  入（读写内存、总线允许、中断确认）

子系统

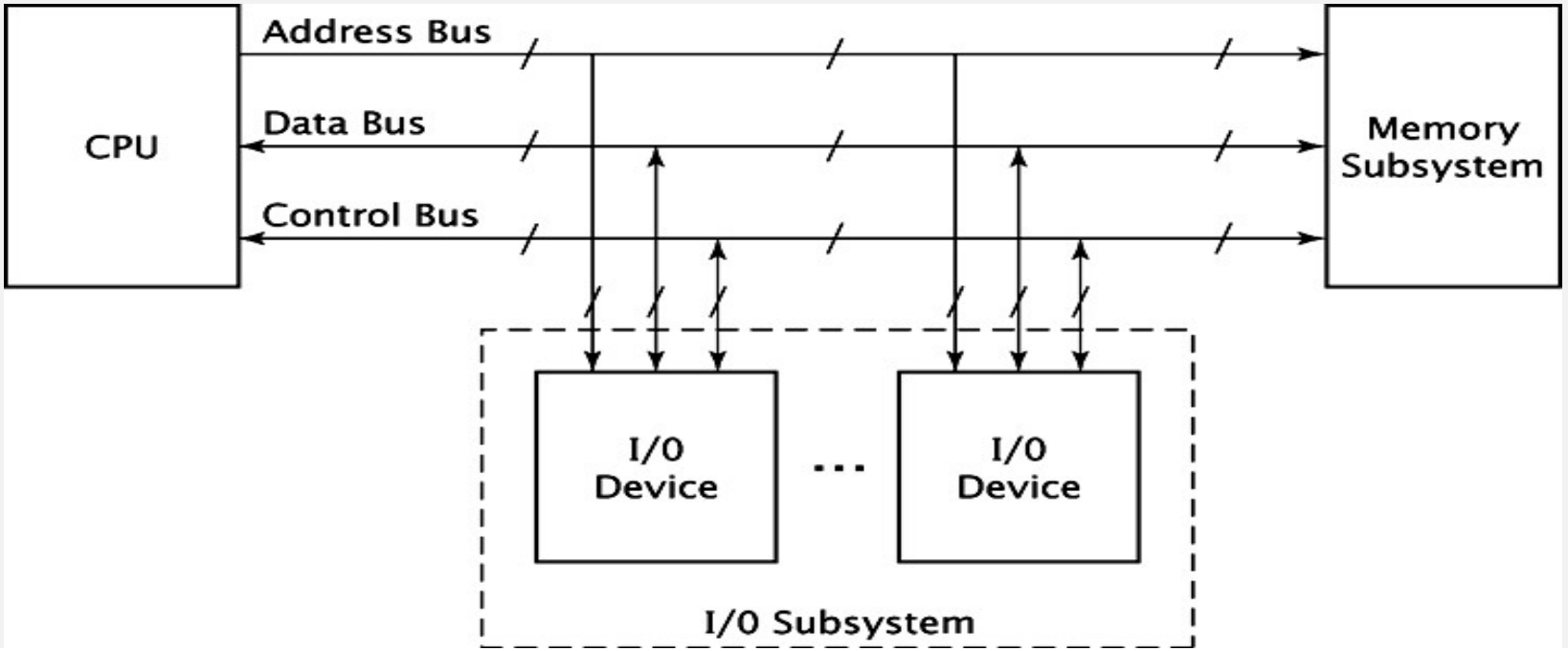
一个简单的计算机通常包括三个主要的子系统：

中央处理单元（CPU）	执行多种操作并控制整个计算机，微处理器（Microprocessor）通常作为微机的CPU。
存储器	用来存储CPU正在执行的程序和数据。
输入输出系统	允许CPU与输入输出设备交互。比如个人计算机的键盘和显示器，或者微波炉的面板和数字显示等。

子系统

抽象化的计算机组成——三大主要部件

- CPU子系统
- 存储器子系统
- I/O子系统



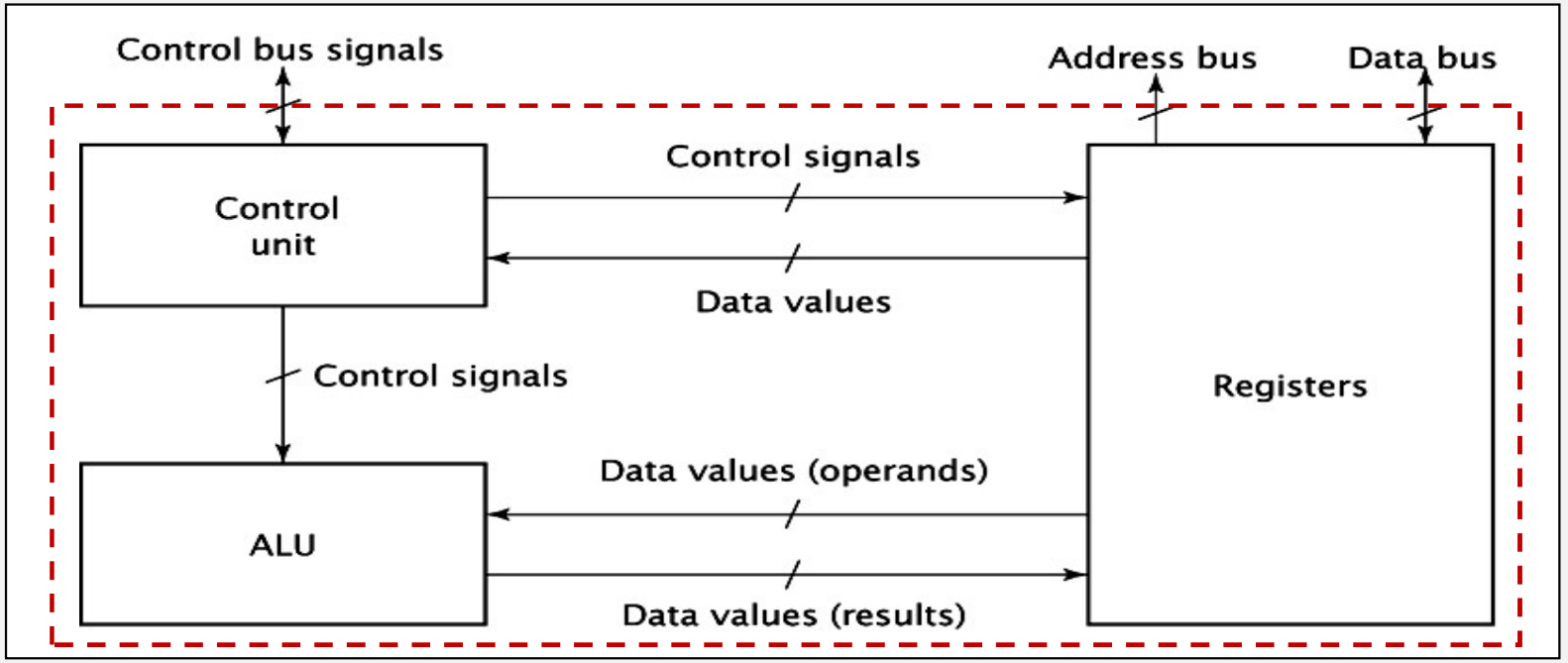
指令周期

指令周期（Instruction Cycle）是微处理器完成一条指令处理的过程。
包括**取指**（Fetch），**译码**（Decode），**执行**（Execute）三个阶段。

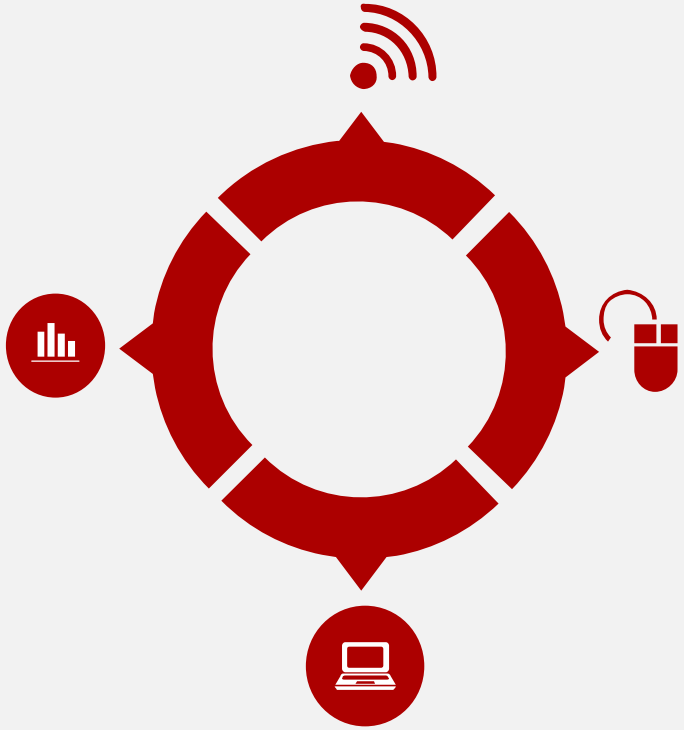
CPU

CPU控制整个计算机，内部有三大部分：

- 寄存器部分（Register Section）、
- 算术/逻辑单元（Arithmetic/Logic Unit，也叫ALU）、
- 控制单元（Control Unit）。如图所示。



CPU



CPU的重要组成部分

程序计数器

Program Counter (PC) 或 Instruction Pointer (IP)，用来存放下一条要提取（执行）的指令的地址。

指令寄存器

Instruction Register (IR) 用来存储从系统数据总线上读取到的指令 。

算术逻辑单元

执行大部分的算术和逻辑运算，如加、与等操作。从CPU的寄存器部分取得操作数，运算结果再存回到寄存器部分。由于必须在一个时钟周期内完成操作，只采用组合逻辑构造而成。

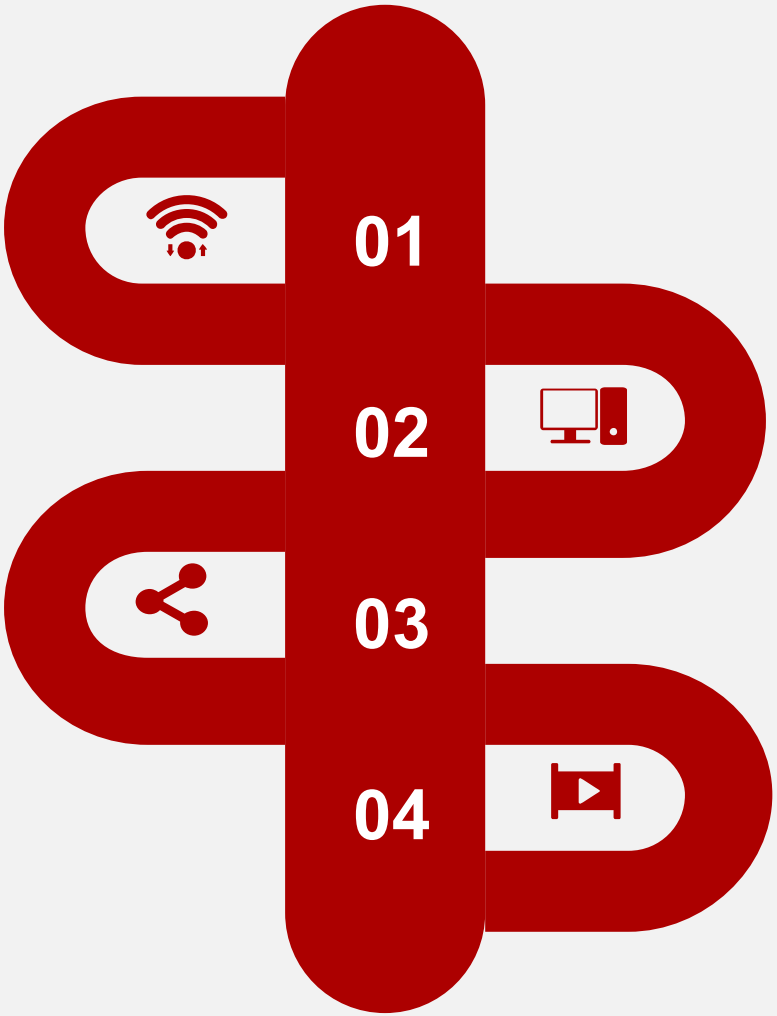
控制单元

同CPU控制整个计算机一样，控制单元控制着CPU。

小结

ISA是处理器所能完成的所有操作（指令）的集合，包含了与该处理器进行交互所需的信息。由编译与汇编过程完成从高级代码到具体指令的转换。

冯-诺依曼结构的计算机包含五大部分及连接它们的总线。



指令格式包括操作码与操作数；在设计ISA时需要考虑完整性与正交性问题。

指令在CPU内部有一定的周期（取指、译码、执行）；CPU内部主要有三大部分（控制单元，算术逻辑单元，寄存器部分）。

下一节：RTL语言

湖南大学

《计算机系统》课程教学组