厦門大學
XIAMEN UNIVERSITY

电子系统设计 Part1

计算机系统基础知识

信息与通信工程系 陈凌宇

chenly@xmu.edu.cn

导言

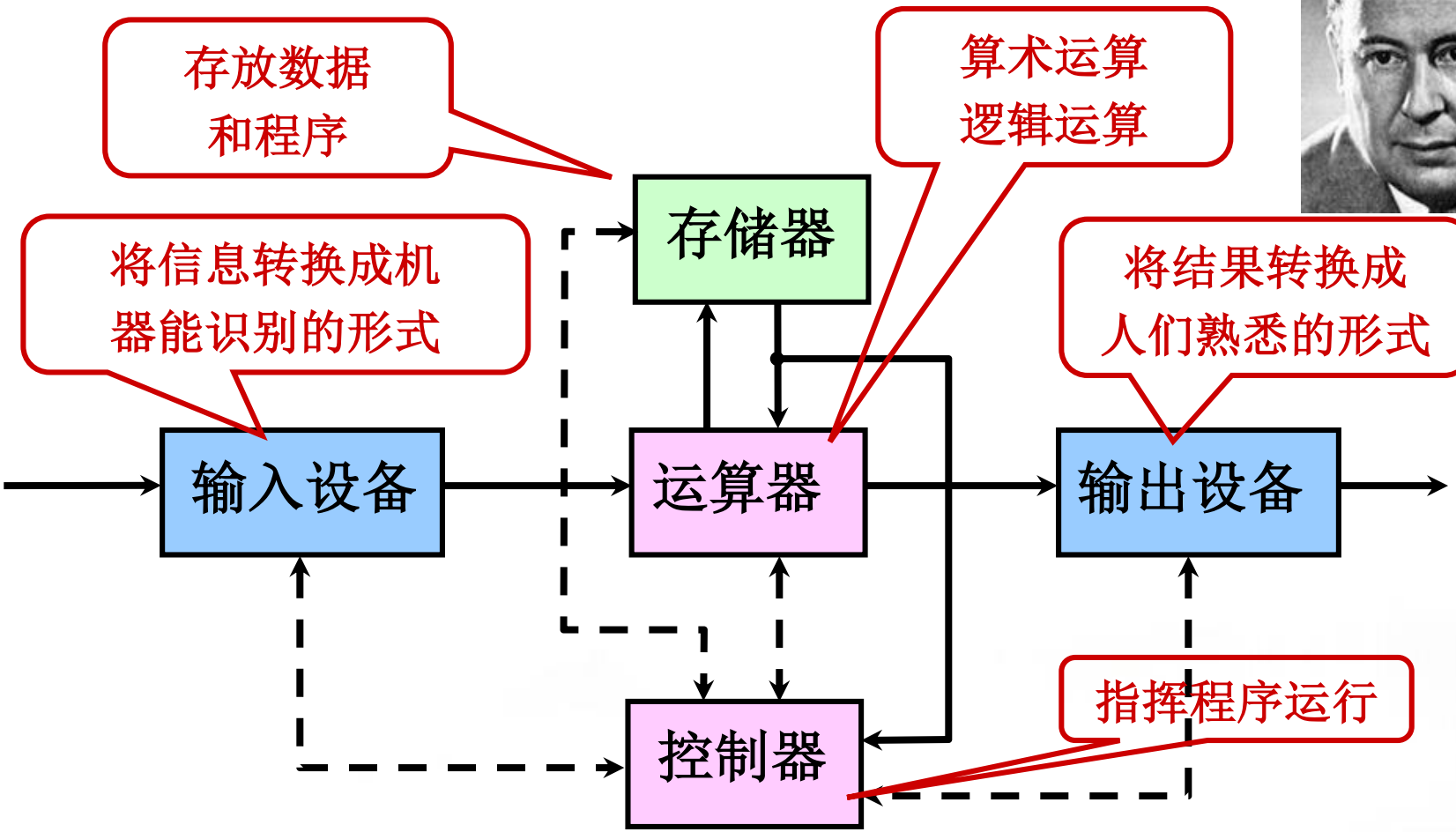
本部分介绍了计算机系统的基础知识，包括处理器的组成与工作原理，指令集的概念等，同时复习计算机中数的表示方法。

目录

- 计算机基本组件
- 机器语言与高级语言
- 数的表示方法

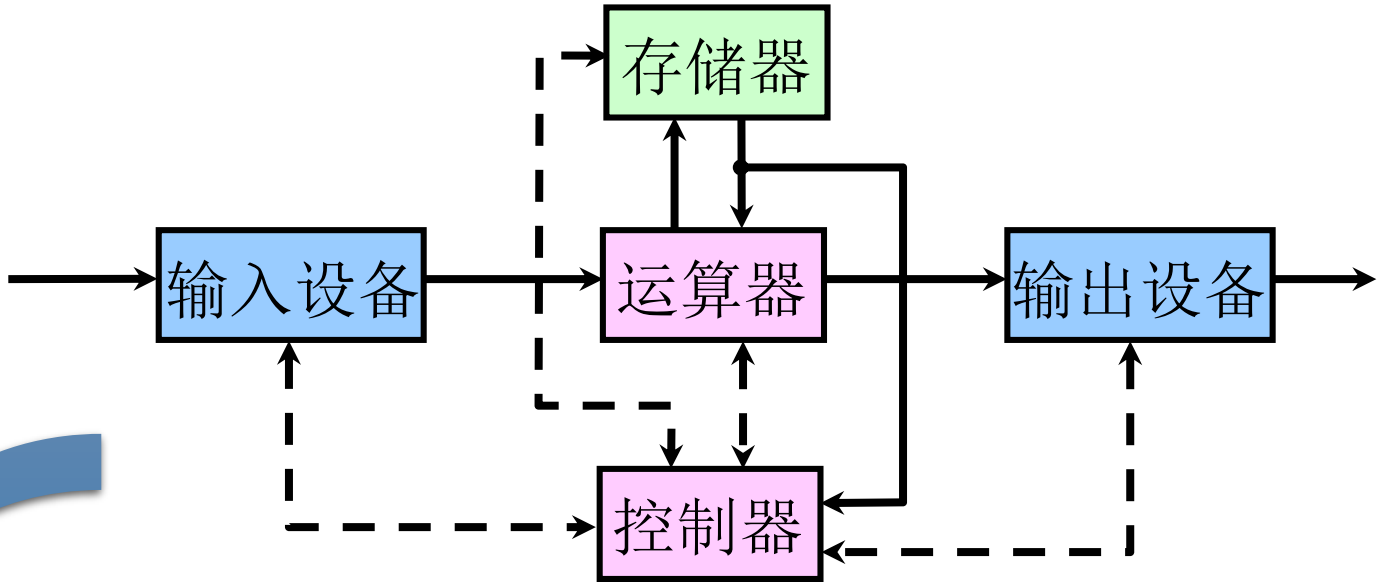
计算机系统的组成

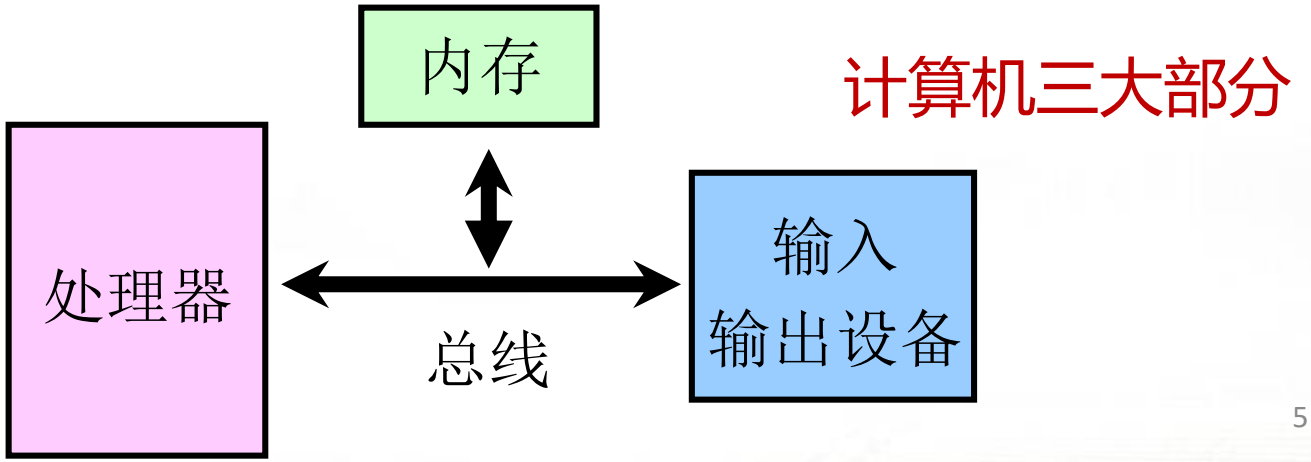
- 计算机具有相同的基本结构（量子计算机等除外）



冯·诺依曼计算机硬件框图

计算机系统的组成

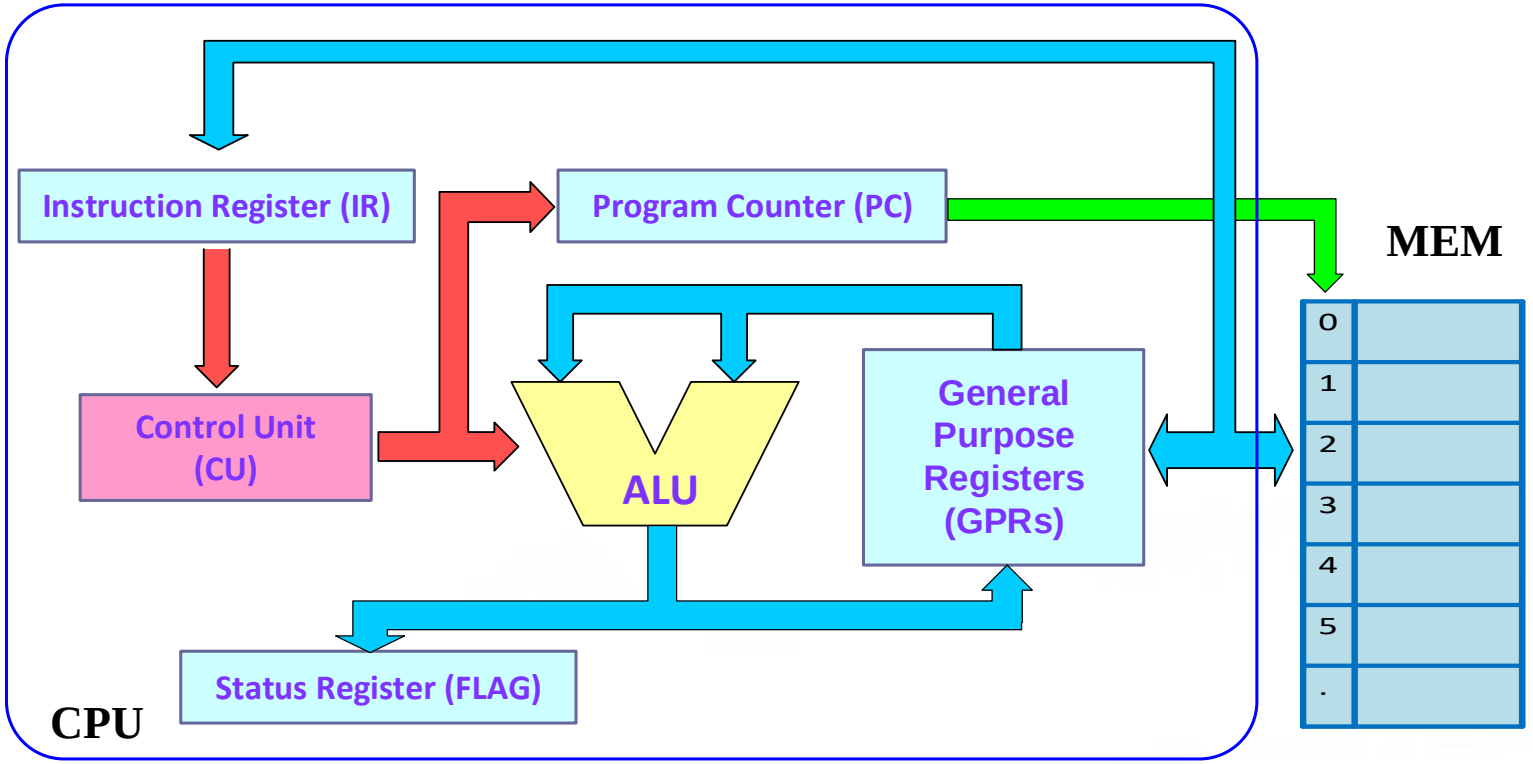




计算机三大部分

处理器

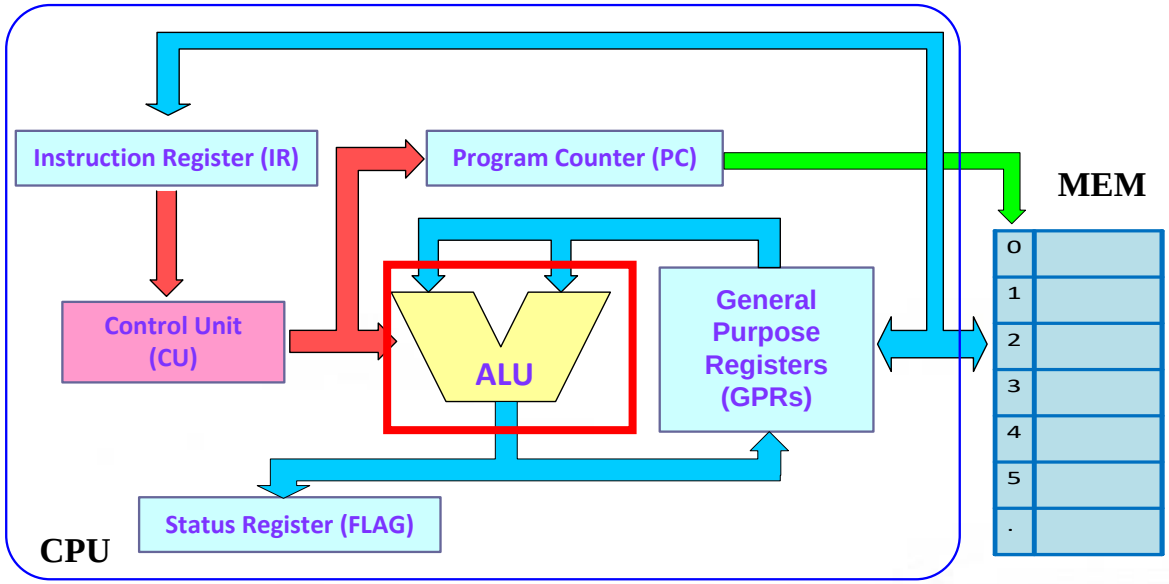
- 计算机的核心
- 包括算术逻辑单元（ALU），控制单元（CU），寄存器组（Register）



处理器

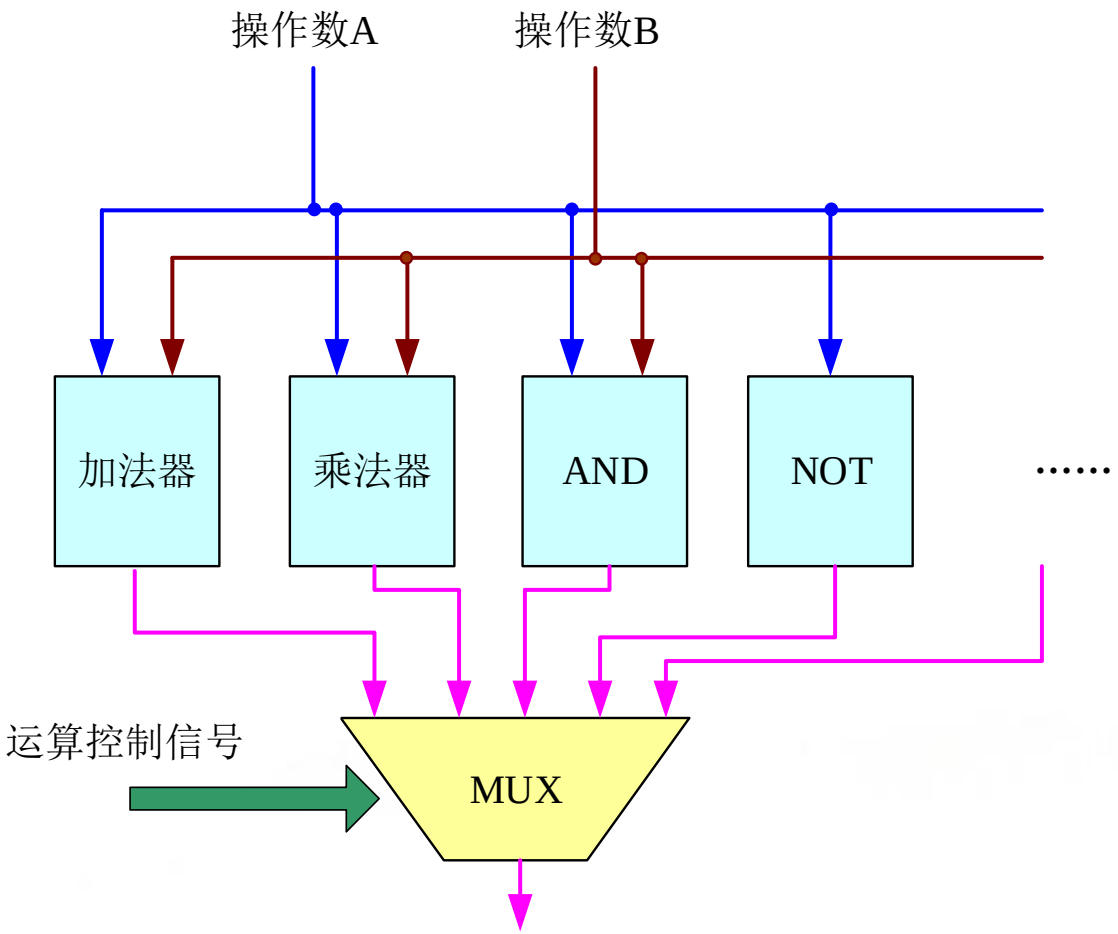
- 算术逻辑单元（Arithmetic Logic Unit, ALU）
  - 执行所有的算术运算及逻辑运算，如：

    +，-，*，AND，OR，NOT，>，< 等
  - 运算器所进行的全部操作都是由控制器发出的控制信号来指挥的，它是执行部件。



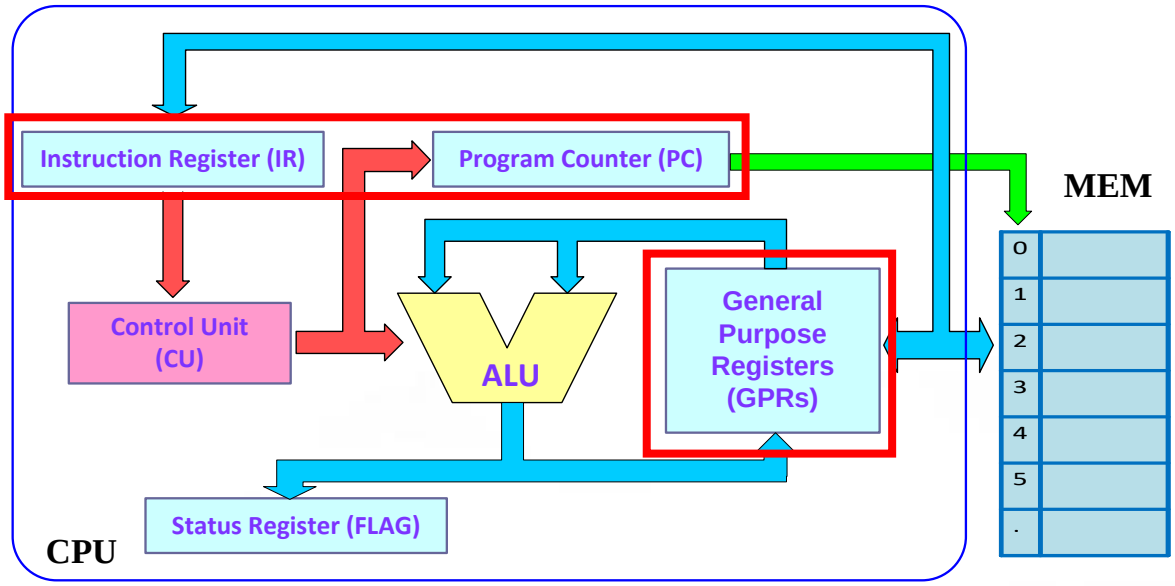
处理器

- ALU的原理



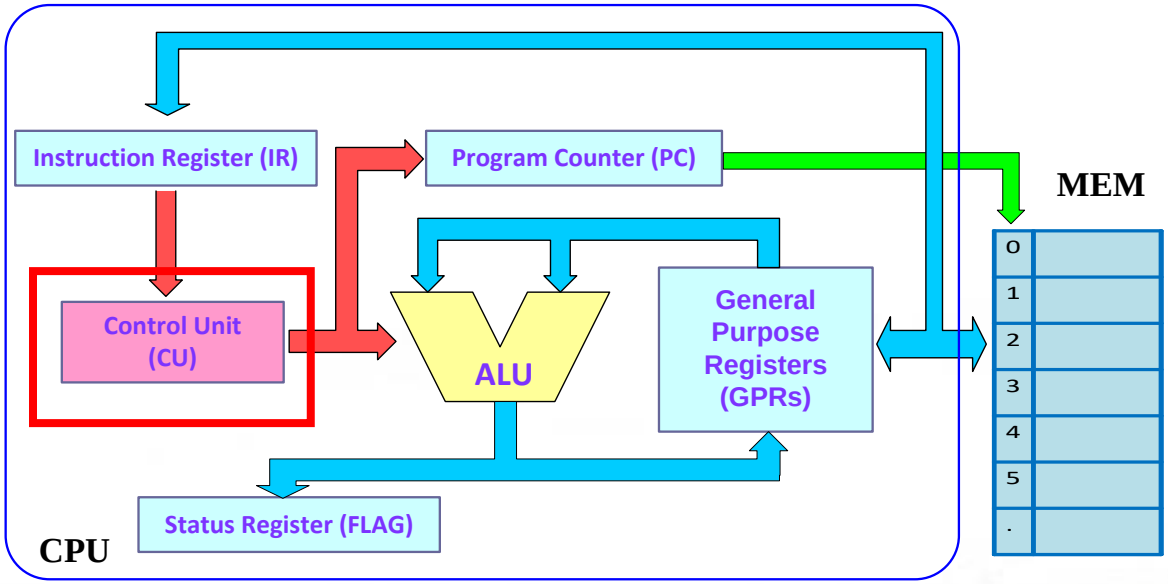
处理器

- 寄存器组（Register）
  - 记录CPU运算的结果；
  - 保存CPU当前的状态；
  - CU根据相应寄存器的值做出决策；



处理器

- 控制单元（CU）
  - “决策机构”，完成协调和指挥整个系统的操作；
  - 从内存中取出一条指令，对指令进行译码，并产生相应的操作控制信号，以便启动响应的动作；
  - 控制ALU计算，并将结果回写寄存器或者内存；



处理器

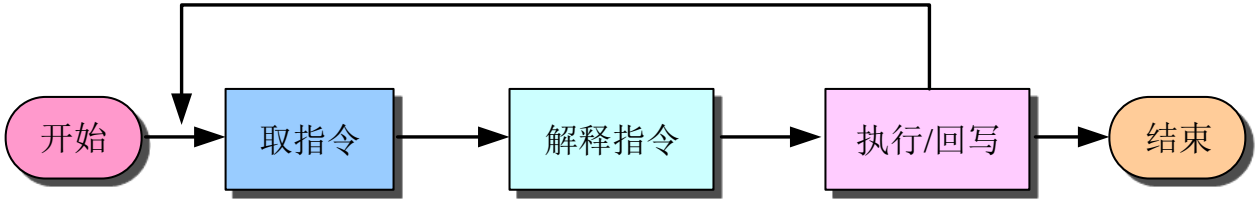
- 例子： 计算4的阶乘，结果写入内存

```
int x = 4, y = 1;
unsigned int *p;
while (x > 1) {
    y = y * x;
    x --;
}
p = 0x100;
*p = y;
```

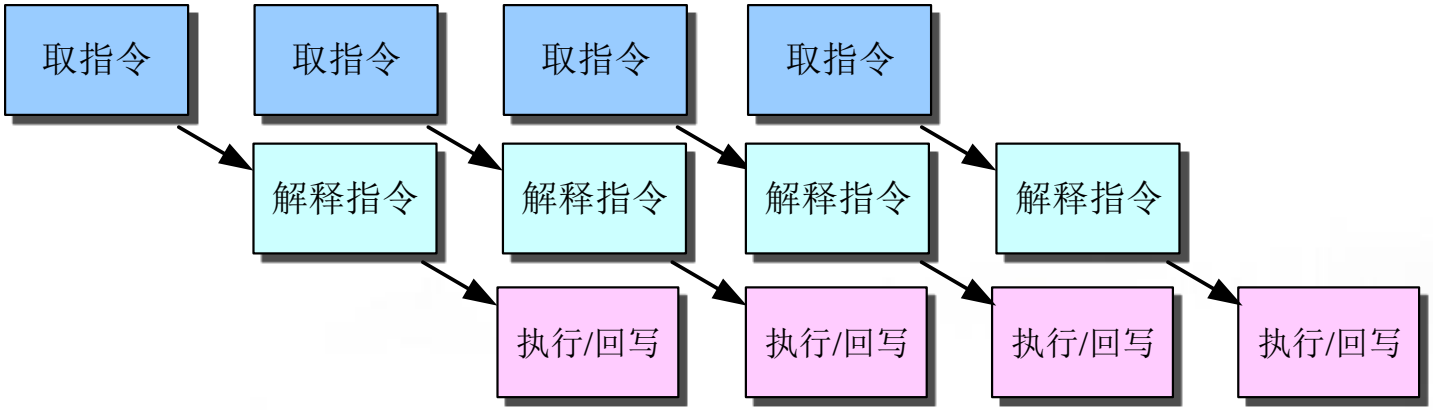
#1 设置R1 = 4;
#2 设置R2 = 1;
#3 判断R1是否大于1，结果存入R3寄存器
#4 若R3寄存器为0跳转至#
#5 计算R2*R1，结果写回R2
#6 计算R1-1，结果写回R1
#7 跳转至#4
#8 设置R4为0x100
#9 结果写回R4指向的内存

处理器

- CPU的工作基本流程



- 流水技术

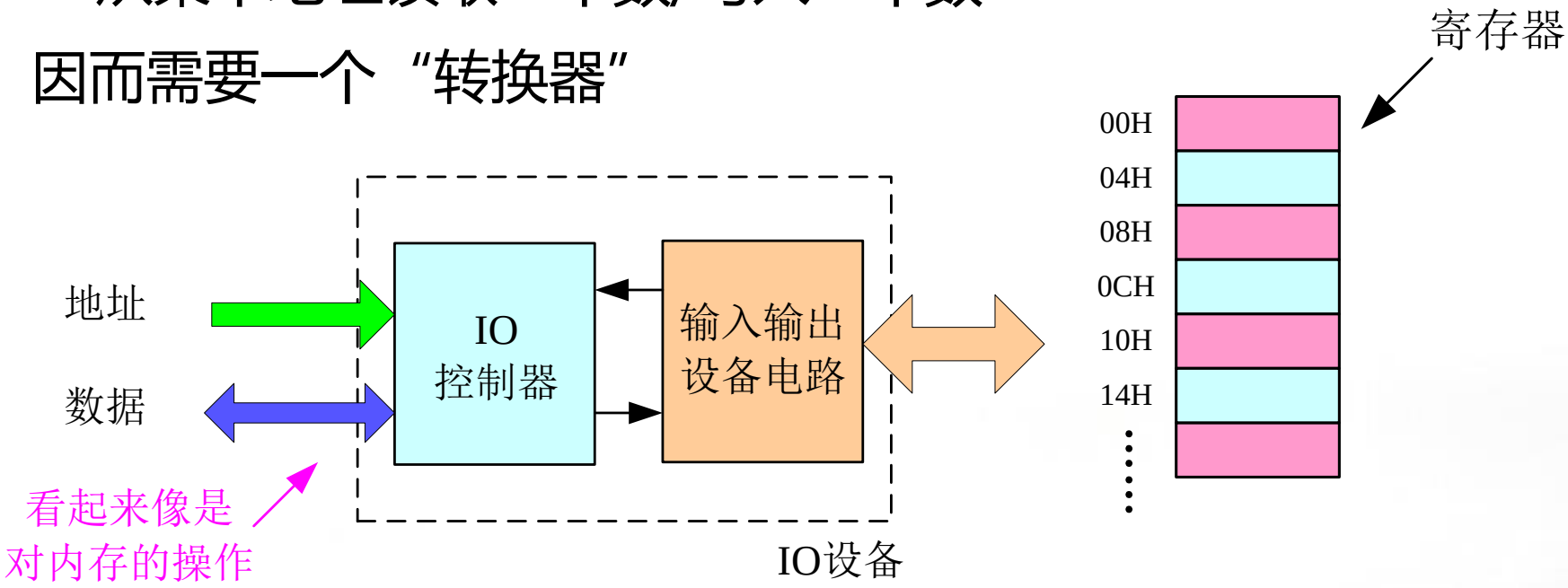


内存

- 一个能够线性存储数据的元件
- 存储指令与数据
- 类型
  - 可读写： SRAM， DRAM
  - 非易失： EEPROM，FLASH

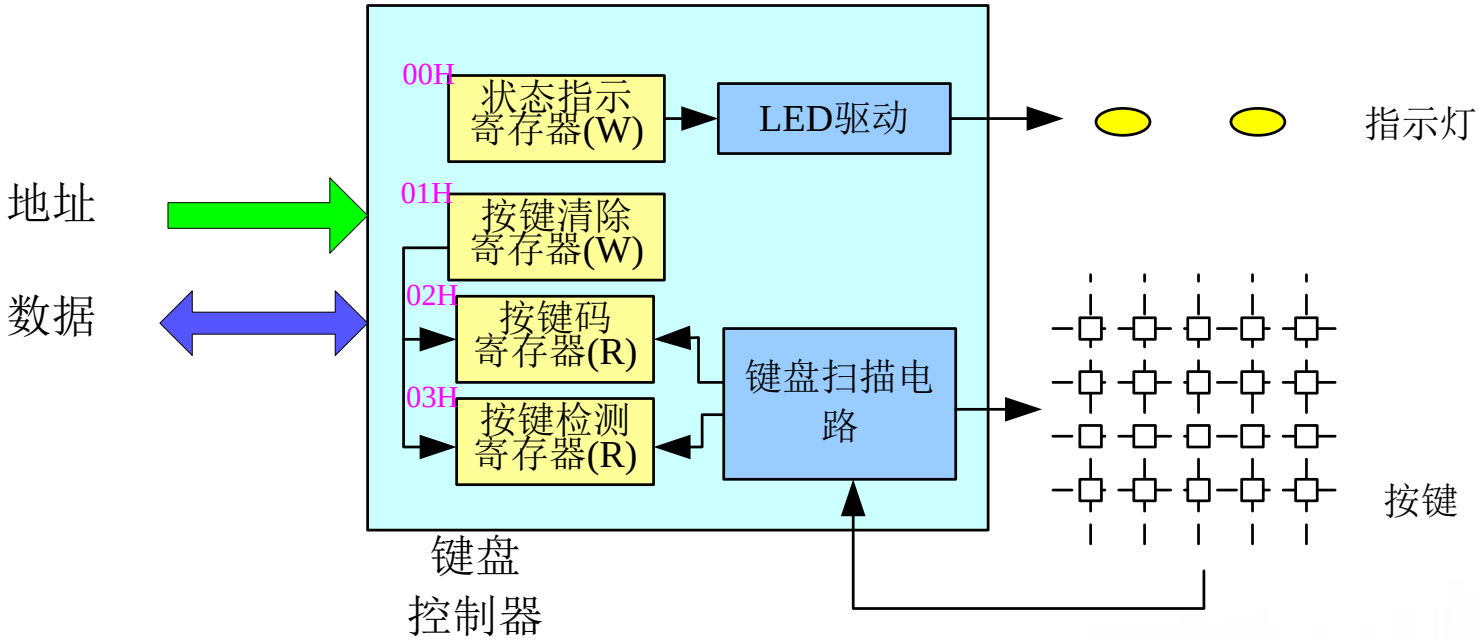
输入输出设备

- 形式多种多样
  - 键盘、鼠标、显示设备、音频设备……
- CPU来仅能够通过以下方式访问外设
  - 从某个地址读取一个数/写入一个数
- 因而需要一个“转换器”



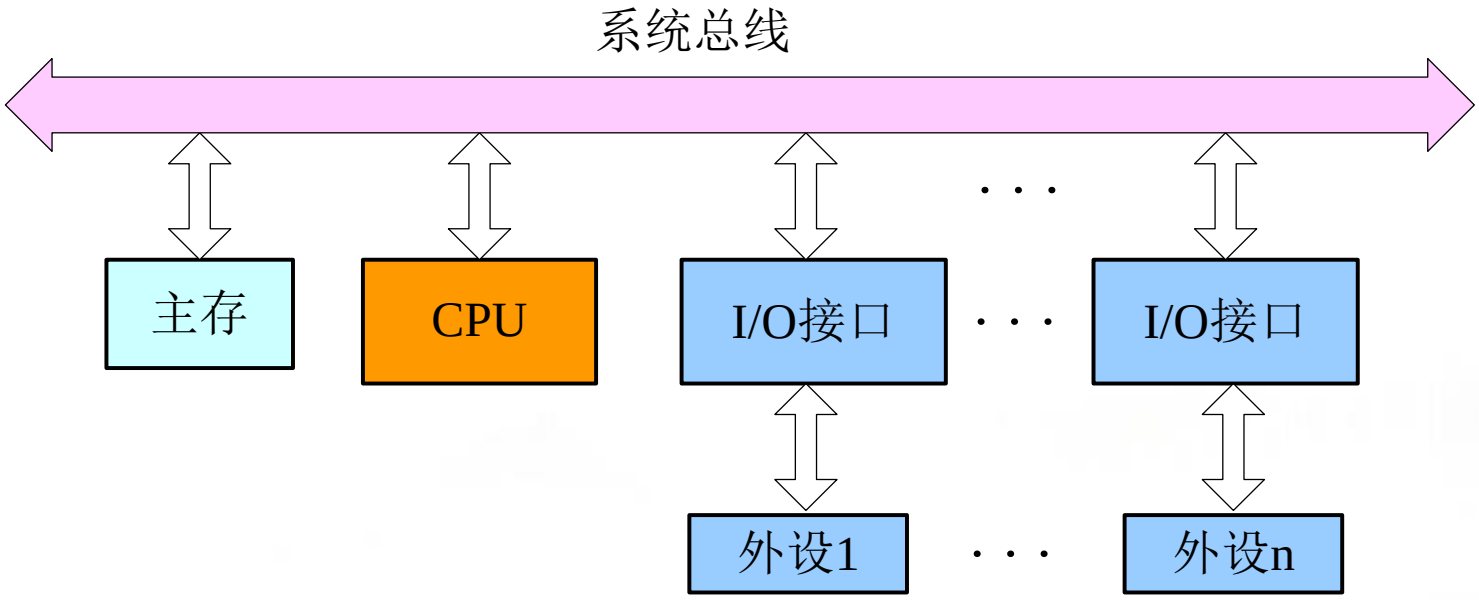
输入输出设备

- 例子：简单的键盘



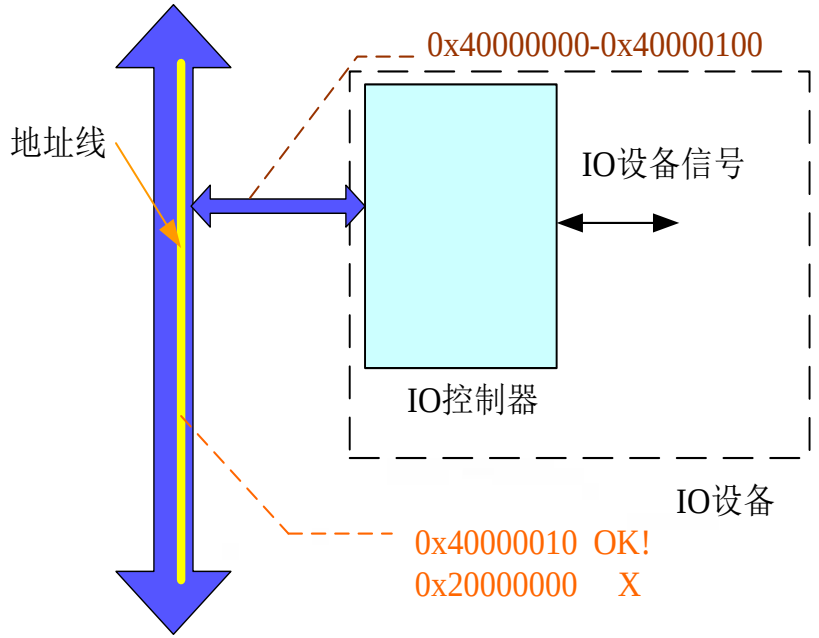
总线

- 总线指通过分时共享的方式，将信息以一个或多个源部件传送到一个或多个目的部件的一组传输线，是计算机中传输数据的公共通道。
- 分时和共享是总线的两个基本特性。



总线

- 总线基本信号：ADDR, WE, RD, WDATA, RDATA
- 地址分配
  - 如何区分总线上的不同设备？为每个设备分配固定的地址空间。



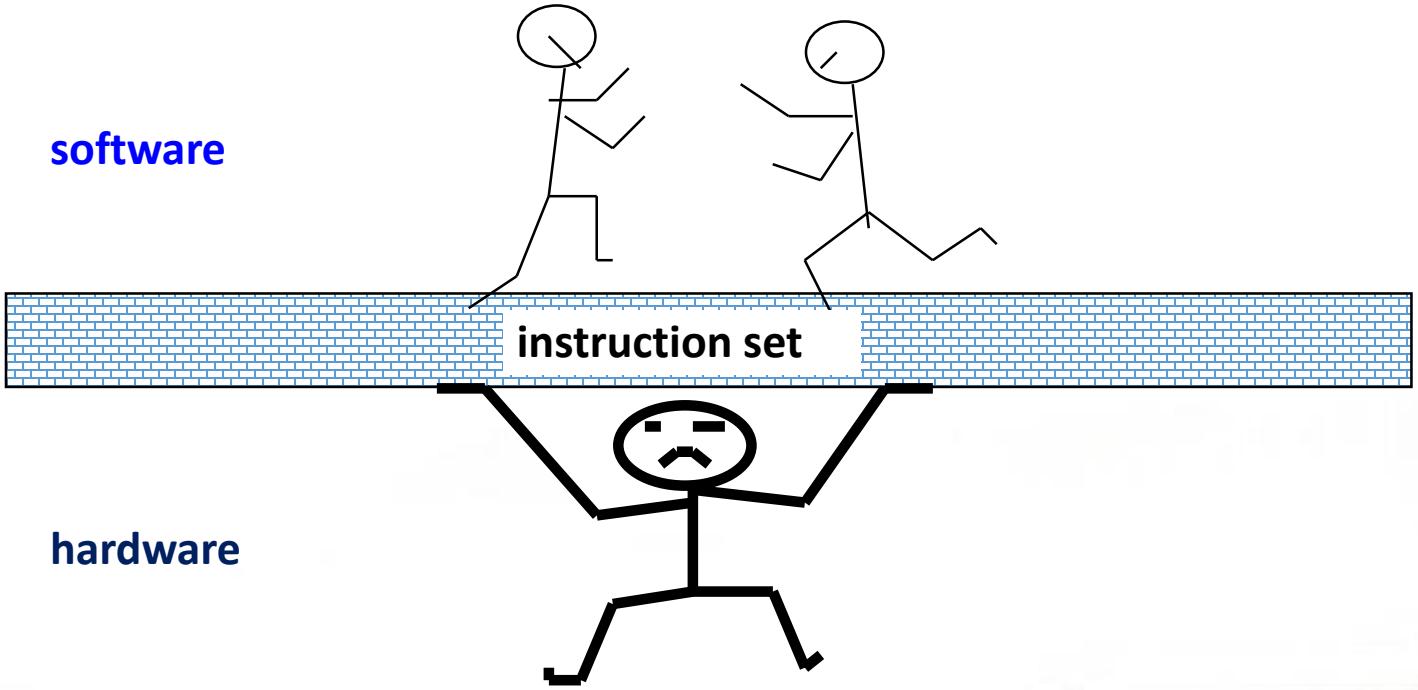
考察总线地址是否落在Base Address至Base Address + Size中

目录

- 计算机基本组件
- 机器语言与高级语言
- 数的表示方法

指令集

- 所有CPU指令的集合
- 一条指令对应了CPU中的某种硬件操作
- 软件最终由一条条指令构成：一条指令即为一个基本步骤
- 指令集是软件与硬件的接口



汇编语言

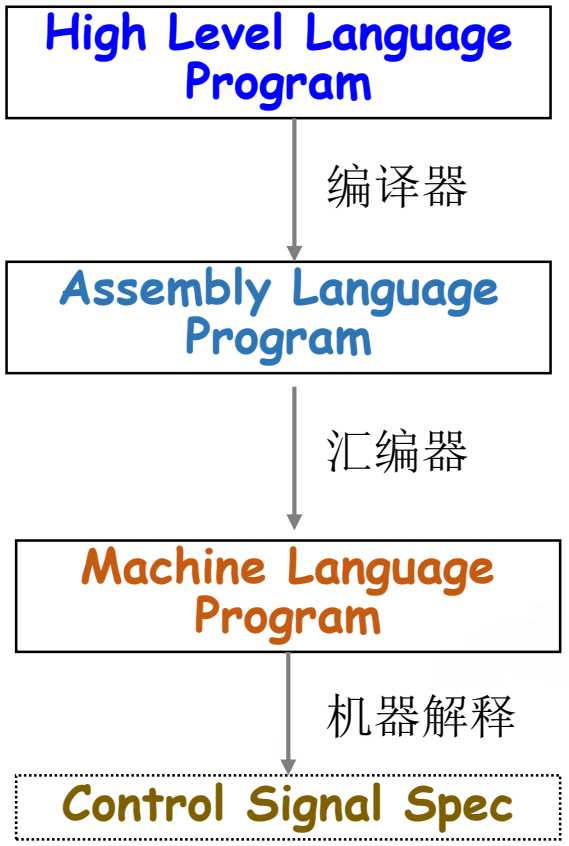
- 汇编语言
  - 机器语言很难记忆，利用助记符

```
01010000  ←→  addi  r1, r0, 1
20000008  ←→  jal   0x8
……
```

  - 这种助记符号来表示计算机指令的语言称为符号语言，也称汇编语言。在汇编语言中，每一条用符号来表示的汇编指令与计算机机器指令一一对应；
  - 用汇编器实现翻译。

高级语言

- 贴近人的思维，C/C++/Java....
- 使用编译器把高级语言转为汇编语言




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw    $15,   0($2)
lw    $16,   4($2)
sw    $16,   0($2)
sw    $15,   4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

```
ALUOP[0:3] <= InstReg[9:11] & MASK
```

目录

- 计算机基本组件
- 机器语言与高级语言
- 数的表示方法

数的表示

- 在计算机中，使用0/1的组合来代表一个数
- 概念辨析：
  - Bit, Bytes, Word, Half-Word, Double-Word
  - MSB(Most Significant Bit), LSB(Least Significant Bit)
- 无符号数和有符号数
  - 无符号数表示范围是0~2^N-1，N为位数；
  - 有符号数通常用补码方式表示，表示范围为-2^{N-1}~2^{N-1}-1；
  - 二进制数是看成无符号数还是有符号数由上下文确定；

数的表示

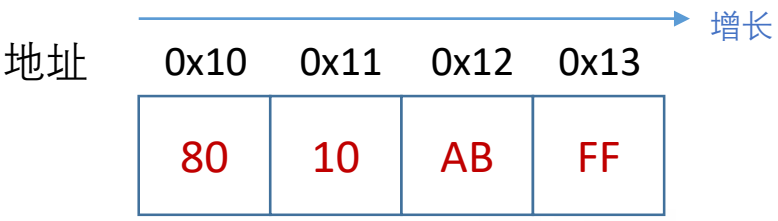
- 以下的变量取值是多少？内存存放的二进制数是多少？
  - b, (*p1), (*p2), (*p3)

```
unsigned int a = 0xFFAB1080;
int b = (int) a;
int *p1 = (int *) &a;
unsigned short int *p2 = (unsigned short int *) &a;
char *p3 = (char *) &a;
```

数的表示

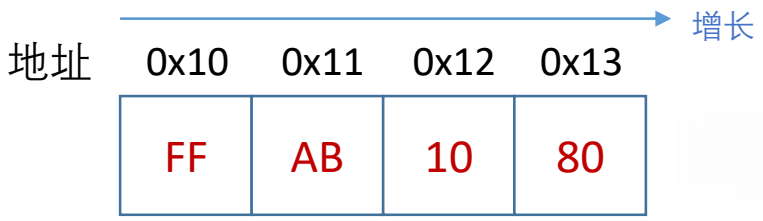
- 以下的变量取值是多少？内存存放的二进制数是多少？
  - b, (*p1), (*p2), (*p3)

```
unsigned int a = 0xFFAB1080;
int b = (int) a;
int *p1 = (int *) &a;
unsigned short int *p2 = (unsigned short int *) &a;
char *p3 = (char *) &a;
```



小端序（Little-Endian）

数据低位字节存放在内存的低位地址



大端序（Big-Endian）

数据低位字节存放在内存的高位地址

数的扩展

- 经常需要将短长度的数扩展成更多位数的数
  - 如8bit的 “11111111” 扩展成16位;
  - 又如以下代码

```
char a0 = -1;                int a1 = a0;
unsigned char b0=0xff; unsigned int b1 = b0;
```

- 扩展方法有两种
  - 零扩展： 高位补0，主要用于无符号数;
  - 符号扩展： 高位补原来的MSB（保留符号位)，主要用于有符号数;

Question ?