- **课程导学：**

“你管这破玩意叫计算机？”

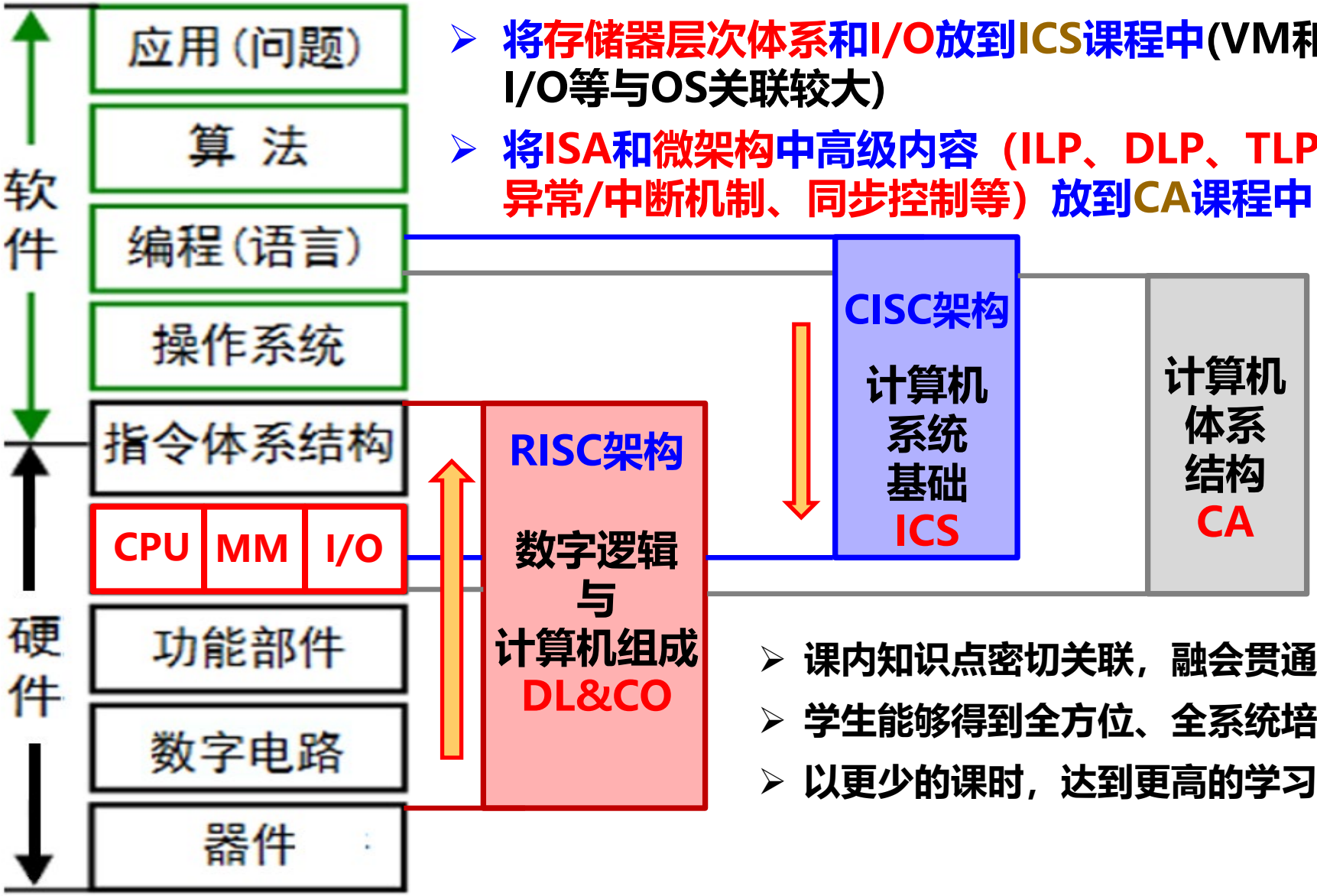
https://mp.weixin.qq.com/s/_Xc-KfyJumxCwHSJxgUpKw

你心里认为“计算机”是什么样子的？

你心里认为“学计算机”应该学什么？

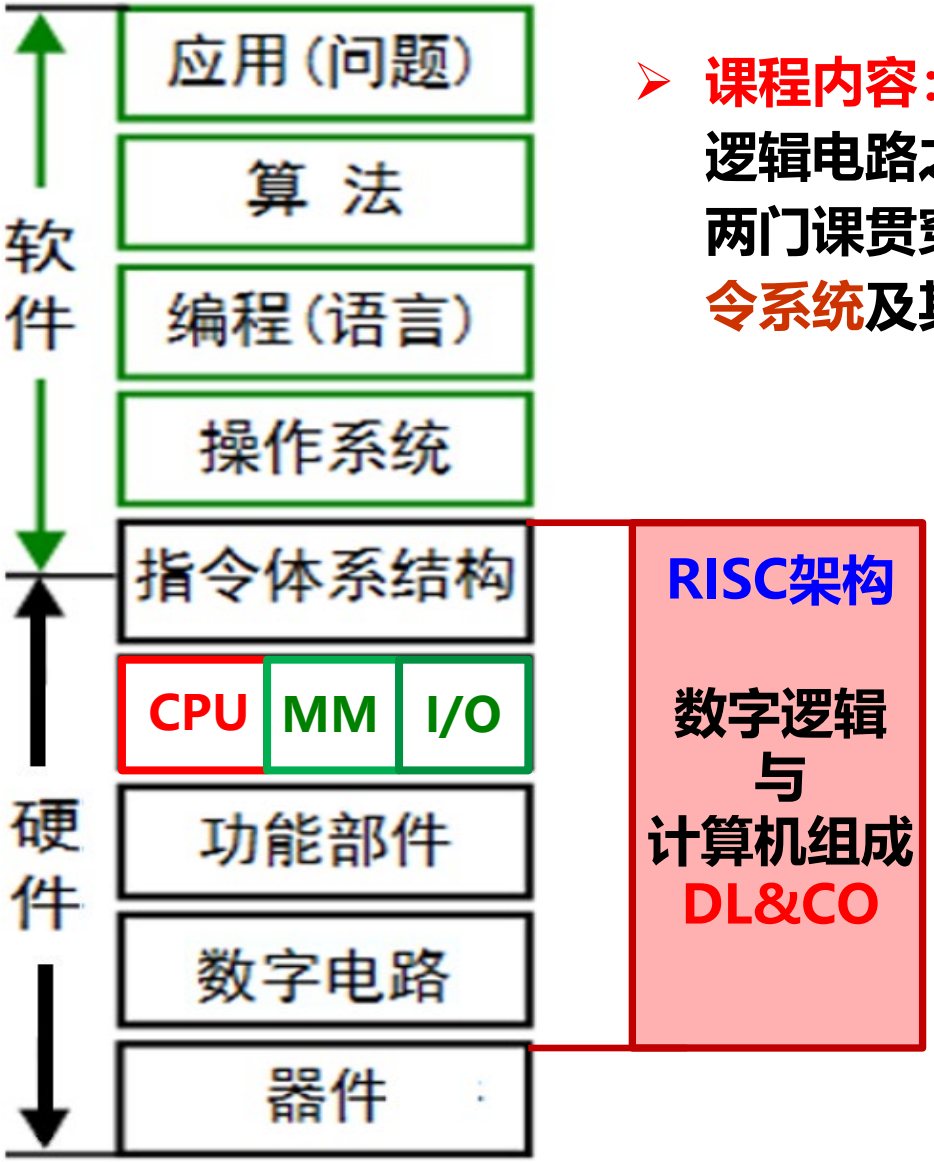
相关课程体系安排

- 将ISA、基本流水线CPU留在DL&CO课程中
- 将存储器层次体系和I/O放到ICS课程中(VM和I/O等与OS关联较大)
- 将ISA和微架构中高级内容（ILP、DLP、TLP、异常/中断机制、同步控制等）放到CA课程中



- 课内知识点密切关联，融会贯通
- 学生能够得到全方位、全系统培养
- 以更少的课时，达到更高的学习目标

本课程“数字逻辑与计算机组成”安排



- **课程内容：**将**数据通路和控制器**设计作为数字逻辑电路之后的数字系统设计案例，将传统的两门课贯穿起来；以**RISC-V**为模型机介绍**指令系统**及其**单周期CPU**和**流水线CPU**设计。

- **教材内容：**包含传统组原教材中的主要内容（本课程不涉及其中MM和IO等），若不开设ICS课程，也可使用DL&CO教材完成完整知识体系的教学

课程基本信息

- **课程名称**
  - **数字逻辑与计算机组成 Digital Logic and Computer Organization**
- **在线课程**
  - **中国大学MOOC平台有多门“数字逻辑电路”、“计算机组成原理”**
  - **超星大视频（袁春风，2011年，计算机组成原理），B站可以观看**
  - **爱课程中精品资源共享课（袁春风，2015年，计算机组成原理）**

  http://www.icourses.cn/sCourse/course_5884.html
- **教材：**
  - **《数字逻辑与计算机组成》，袁春风 武港山 吴海军 余子濠，机械工业出版社**
- **主要参考教材：**
  - 《数字逻辑与计算机组成习题解答与实验教程》
  - 《数字设计和计算机体系结构（原书第2版）》David Money Harris、Sarah. L. Harris 著，陈俊颖 译，机械工业出版
  - 《计算机组成与系统结构（第2版）》袁春风主编，清华大学出版社，2015 年
  - 《计算机组成与设计》，袁春风 余子濠，高等教育出版社，2020.10
  - 《计算机组成与系统结构习题解答与教学指导（第2版）》袁春风主编，清华大学出版社，2016年

实验及考核方式 ★★

- 关于本课程的配套实验，可以先阅读此文:
- https://mp.weixin.qq.com/s/V7QyK2VG7WFtrPTaCLQYFA
- 校外访问https://www.educoder.net/paths/t7lamrv6
- 校内访问https://cslab.nju.edu.cn/main.htm“在线教学-实践课程”
  - **实验1：基本逻辑部件设计**
  - **实验2：组合逻辑电路设计**
  - **实验3：同步时序电路设计**
  - **实验4：加法器和ALU设计**
  - **实验5：取指令部件设计**
  - **实验6：单周期CPU设计与测试**

- **考核方式**
  - **习题、小测验等平时成绩：15%**
  - **Lab实验成绩：35% （相当于期中考试成绩）**
  - **期末考试成绩：50%**

第1章 二进制编码

第一讲 计算机系统概述

第二讲 二进制数的表示

第三讲 数值数据的编码表示

第四讲 非数值数据的编码表示及数据的宽度和存储排列

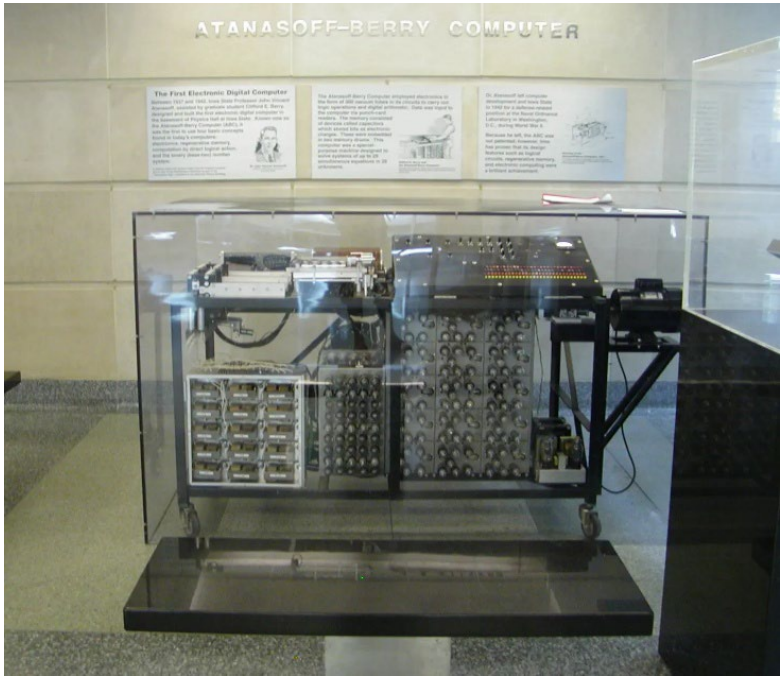
第一讲 计算机系统概述

- 冯.诺依曼结构计算机
  - 冯.诺依曼结构基本思想
  - 计算机硬件的基本组成
- 程序的表示和执行过程
  - 机器级语言和高级编程语言
  - 翻译程序：汇编、编译、解释
- 计算机系统抽象层
  - 计算机硬件和软件的接口：指令系统
  - 本课程内容在计算机系统中的位置

世界上第一台电子计算机ABC（非通用）

- 阿塔纳索夫-贝瑞计算机（Atanasoff-Berry Computer，简称ABC）——爱荷华州立大学的约翰·文特森·阿塔纳索夫（John Vincent Atanasoff）和他的研究生克利福特·贝瑞（Clifford Berry）在1937年设计，不可编程，仅仅设计用于求解线性方程组，并在1942年成功进行了测试。

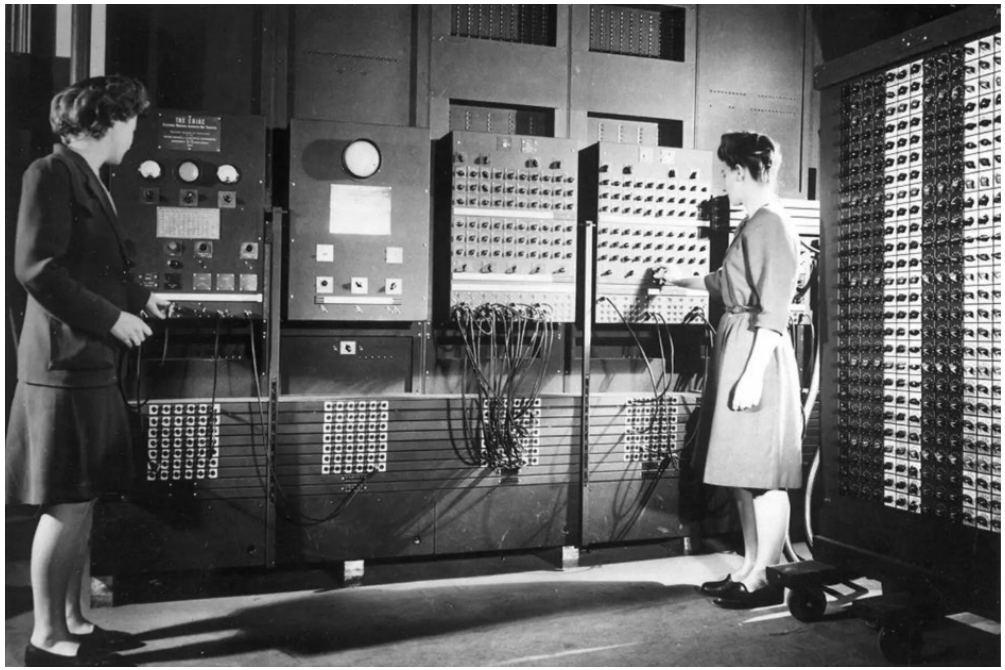
- 这台计算机是电子与电器的结合，电路系统中装有300个电子真空管执行数字计算与逻辑运算，机器使用电容器来进行数值存储，数据输入采用打孔读卡方法，采用二进制



世界上第一台通用电子计算机ENIAC

- 1945年，电子数字积分计算机，是世界上第一台通用计算机，也是继ABC之后的第二台电子计算机，它是图灵完全的电子计算机，能编程，解决不同的计算问题。

- 一开始是采用十进制
- 非存储程序
- 非冯诺依曼结构



冯·诺依曼的故事

- 1944年，冯•诺依曼参加原子弹的研制工作，涉及到极为困难的计算。
- 1944年夏的一天，诺依曼巧遇美国弹道实验室的军方负责人戈尔斯坦，他正参与ENIAC的研制工作。
- 冯•诺依曼被戈尔斯坦介绍加入ENIAC研制组，1945年，在共同讨论的基础上，冯•诺依曼以“关于EDVAC的报告草案”为题，起草了长达101页的总结报告，发表了全新的“**存储程序通用电子计算机方案**”。

Electronic
Discrete
Variable
Automatic
Computer

现代计算机的原型

1946年，普林斯顿高等研究院（the Institute for Advance Study at Princeton，IAS）让冯·诺依曼设计“存储程序”计算机，其依据就是这份报告。被称为IAS计算机（1951年建成，即EDVAC）。

世界上第一台现代、存储程序式、通用、冯诺依曼结构、电子计算机?

- 1948年6月的曼彻斯特小型机（Manchester Baby） 是第一。
- EDSAC——电子延迟存储自动计算器（Electronic delay storage automatic calculator）是第二，由英国剑桥大学在1949年5月建成，
- EDVAC本身——冯诺依曼的草案启发了全世界，自己却由于某些技术和非技术原因，直到1951年才建成。

冯·诺依曼结构的核心？

- 在那个报告中提出的计算机结构被称为冯·诺依曼结构。

- 冯·诺依曼结构最重要的思想是什么？

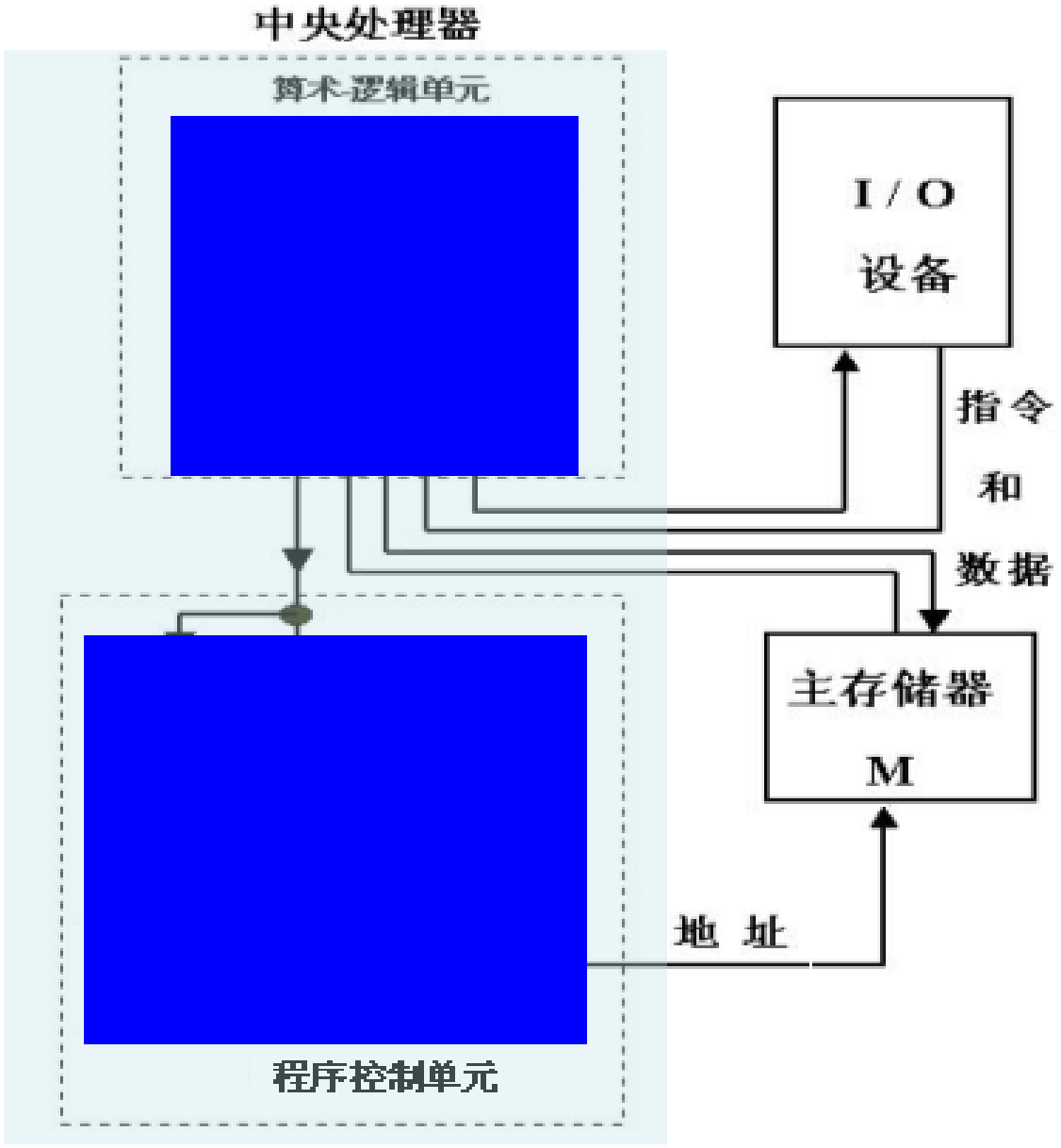
  “存储程序(Stored-program)” 工作方式：

  任何要计算机完成的工作都要先被编写成程序，然后将程序和原始数据送入主存并启动执行。一旦程序被启动，计算机应能在不需操作人员干预下，自动完成逐条取出指令和执行指令的任务。

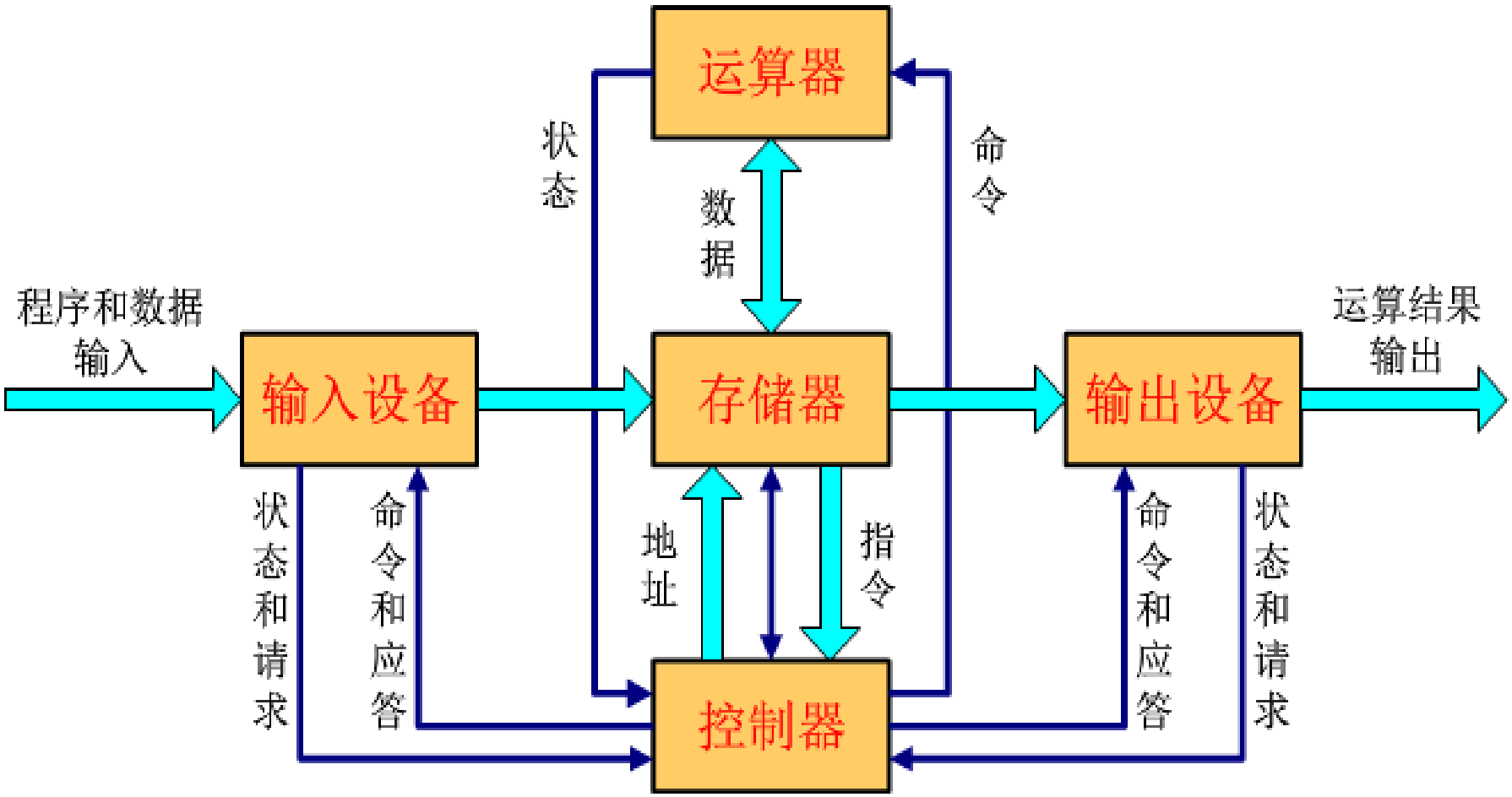
- 冯·诺依曼结构计算机也称为冯·诺依曼机器（Von Neumann Machine）。
- 几乎现代所有的通用计算机大都采用冯·诺依曼结构。

冯·诺依曼结构计算机应该是怎样的？

- 应该有个主存，用来存放程序和数据
- 应该有一个自动逐条取出指令的部件
- 还应该有具体执行指令（即运算）的部件
- 程序由指令构成
- 指令描述如何对数据进行处理
- 应该有将程序和原始数据输入计算机的部件
- 应该有将运算结果输出计算机的部件



冯.诺依曼结构计算机模型



早期，部件之间用分散方式相连

现在，部件之间大多用总线方式相连

趋势，点对点（分散方式）高速连接

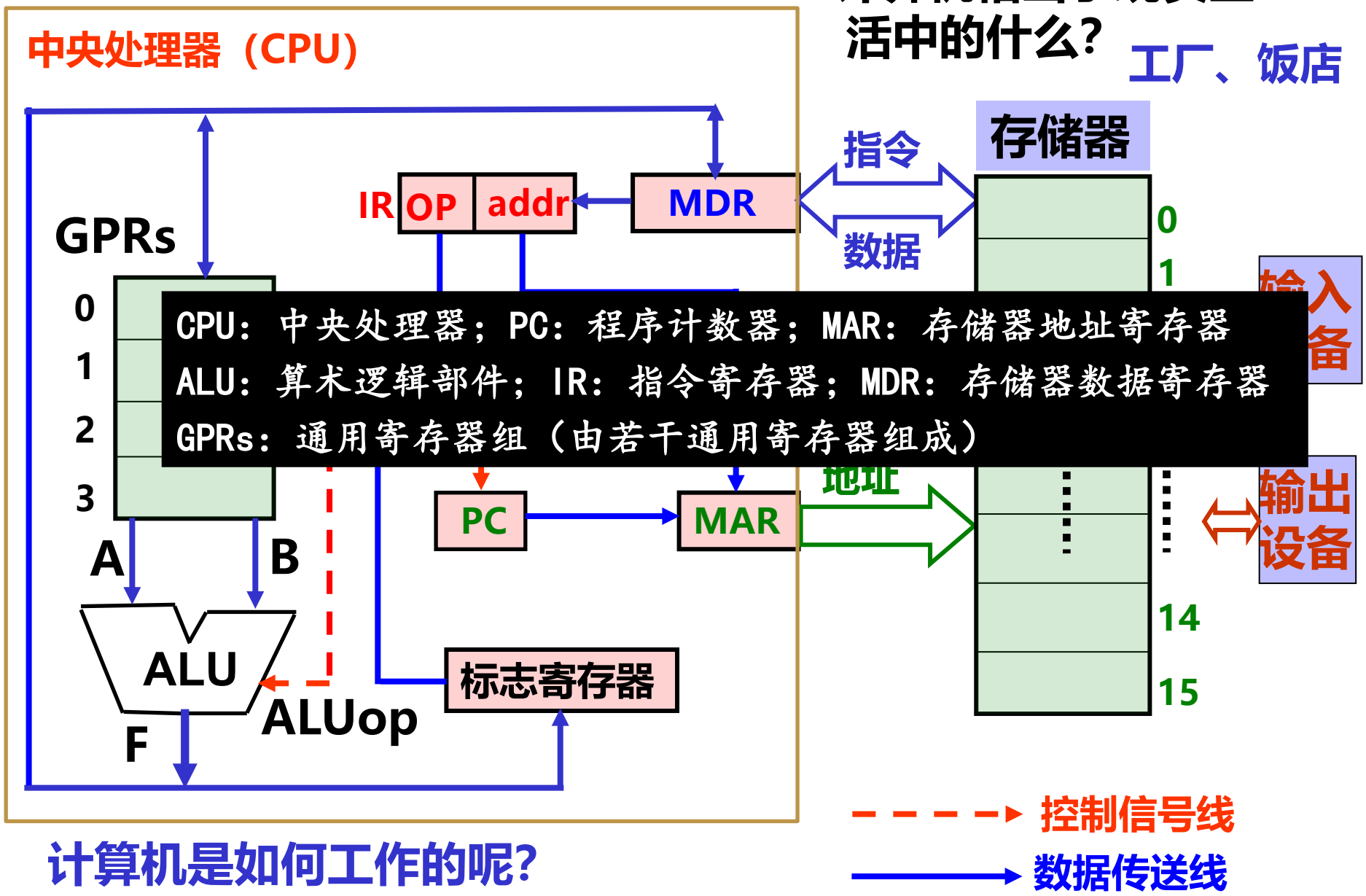
冯·诺依曼结构的主要思想

综上所述，归纳一下

1. 计算机应由运算器、控制器、存储器、输入设备和输出设备五个基本部件组成。
2. 各基本部件的功能是：
   - 存储器不仅能存放数据，而且也能存放指令，形式上两者没有区别，但计算机应能区分数据还是指令；
   - 控制器应能自动取出指令来执行；
   - 运算器应能进行加/减/乘/除四种基本算术运算，并且也能进行一些逻辑运算和附加运算；
   - 操作人员可以通过输入设备、输出设备和主机进行通信。
3. 内部以二进制表示指令和数据。每条指令由操作码和地址码两部分组成。操作码指出操作类型，地址码指出操作数的地址。由一串指令组成程序。
4. 采用“存储程序”工作方式。

现代计算机结构模型

计算机相当于现实生活中的什么？ 工厂、饭店



计算机是如何工作的呢？

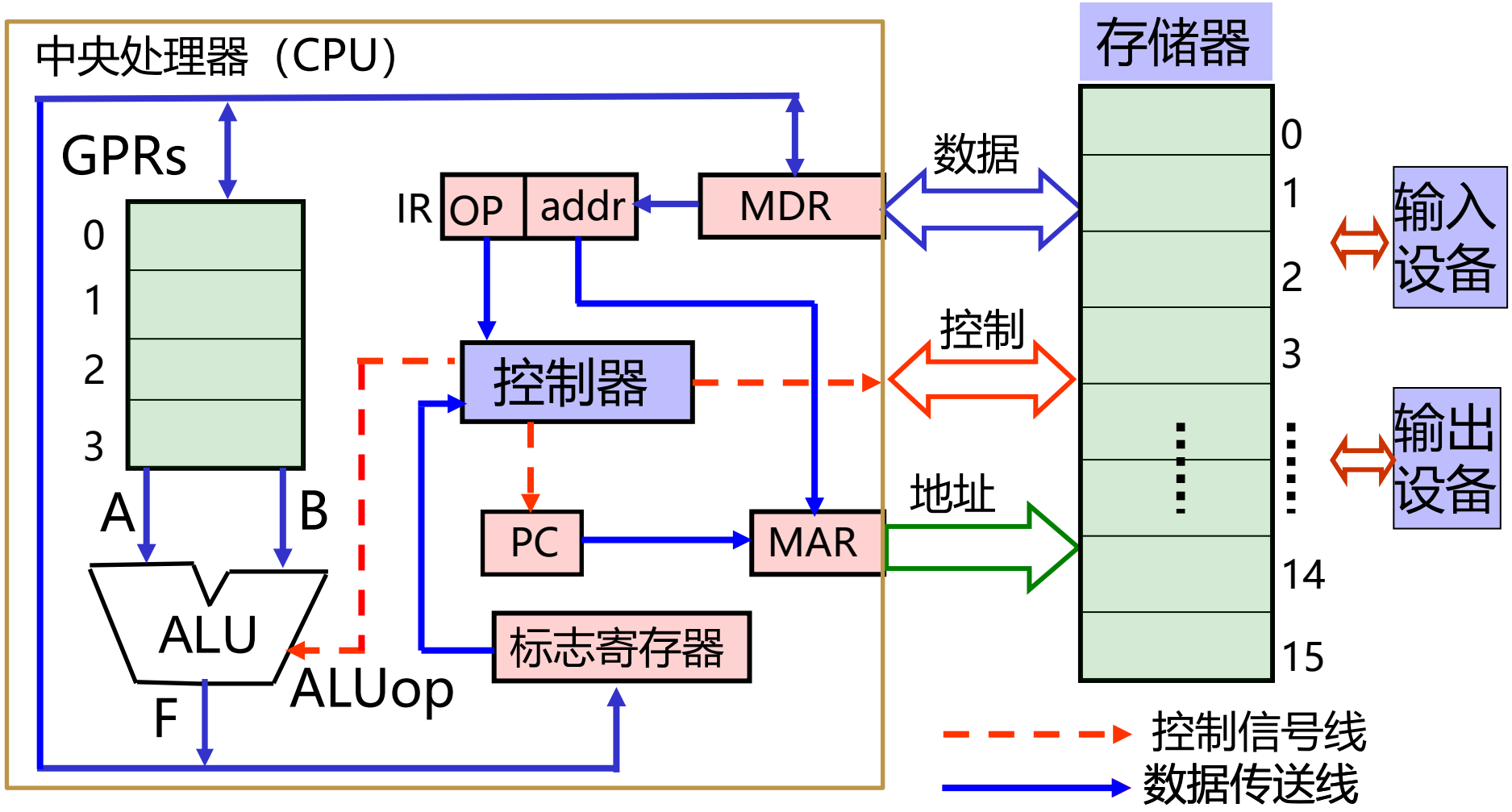
第一讲 计算机系统概述

- 冯.诺依曼结构计算机
  - 冯.诺依曼结构基本思想
  - 计算机硬件的基本组成
- 程序的表示和执行过程
  - 机器级语言和高级编程语言
  - 翻译程序：汇编、编译、解释
- 计算机系统抽象层
  - 计算机硬件和软件的接口：指令系统
  - 本课程内容在计算机系统中的位置

计算机是如何工作的？

先想象一下你爸妈是怎样做一桌你指定的菜的?

厨房-CPU，爸妈-控制器，锅灶-ALU ，盘碟-GPRs，冰箱桌柜-存储器



计算机是如何工作的：一个比喻

类似“存储程序”工作方式

- 做菜前

原材料（数据）和菜谱（指令）都按序放在冰箱桌柜（存储器）里， 每个存放位置都给一个编号（存储单元地址）。每个盘碟也有编号（GPRs里的具体寄存器指定）。

菜谱上信息：原料的存放位置、做法、做好的菜放在哪里等

例如，把10、11号位置的原料一起炒，并装入3号盘

首先，我告诉爸妈——从存放在5号位置（起始PC=5）的菜谱开始顺着做

- 开始做菜（在爸妈的控制下来完成以下步骤）

第一步：从5号位置取菜谱（根据PC取指令）

第二步：看菜谱（指令译码）

第三步：从冰箱橱柜 或 盘碟中取原材料（取操作数）

第四步：洗、切、炒等具体操作（指令执行，用到ALU也就是锅灶等）

第五步：装盘 或 送桌（回写结果）

第六步：算出下一菜谱所在位置6=5+1（修改PC的值）

回到第一步，继续做下一道菜（执行下一条指令）

计算机是如何工作的？ “存储程序”工作方式

程序由指令组成，若所有指令执行完，则程序执行结束

- 程序在执行前
  数据和指令事先存放在存储器中：形式上没有差别，都是0/1序列
  每条指令和每个数据都有地址
  指令按序存放
  程序起始地址置PC

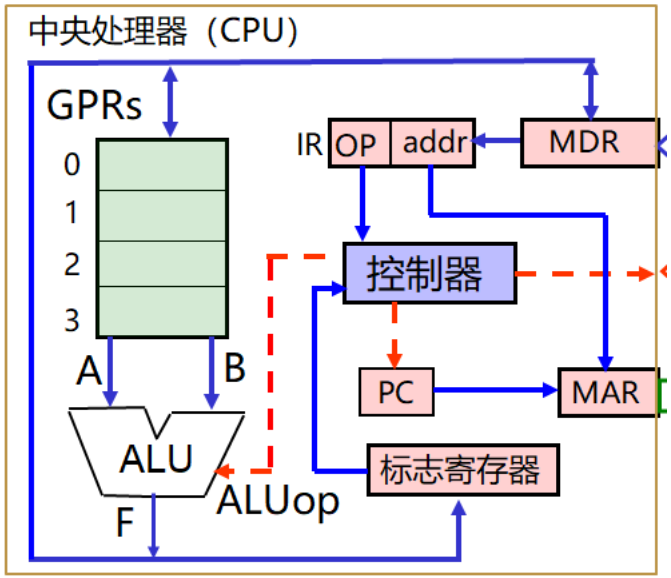
- 开始执行程序后：计算机能自动取出一条一条指令执行，在执行过程中无需人的干预。以下步骤在控制器的协调下完成。

  第一步：根据PC取指令　　第二步：指令译码
  第三步：取操作数　　第四步：指令执行
  第五步：回写结果　　第六步：修改PC的值
  第七步：按新的PC，回到第一步，继续执行下一条指令

计算机是如何工作的？——指令的执行

- **指令执行过程中，指令和数据被从存储器取到CPU，存放在CPU内的寄存器中，指令在IR中，数据在GPR中。**

- **指令由OP、ADDR等字段组成**



指令中需给出的信息：

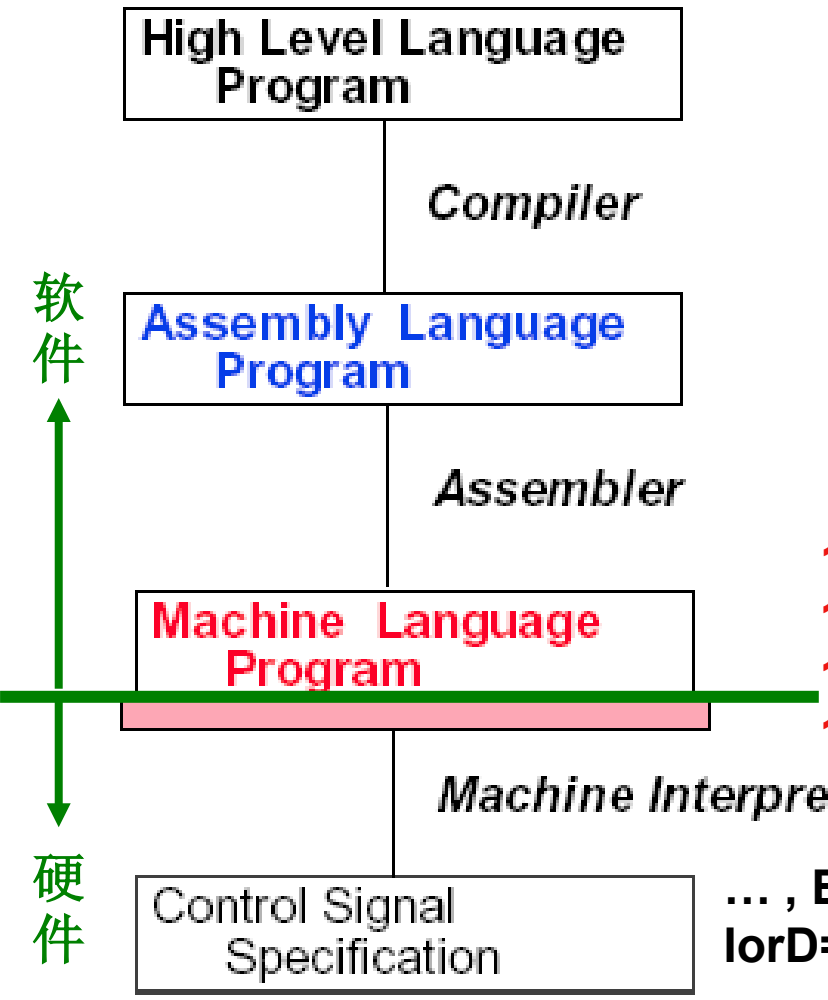
操作性质（操作码）

源操作数1 或/和 源操作数2　（立即数、寄存器编号、存储地址）

目的操作数地址　（寄存器编号、存储地址）

至此，我们一直说的好像都是“硬件”，还缺了点什么？

程序（指令）的表示和执行：软硬件层次




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

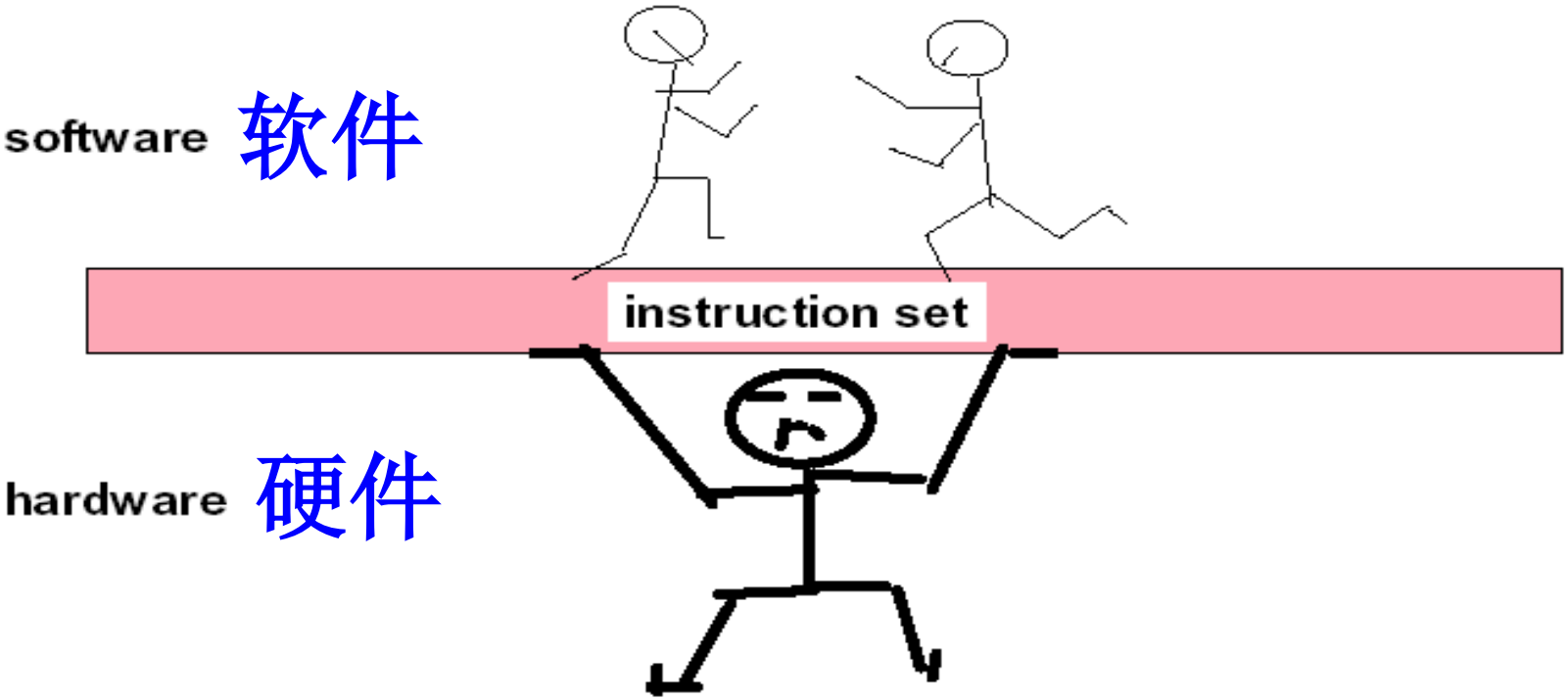
```
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 4($2)
```

```
1000 1100 0100 1111 0000 0000 0000 0000
1000 1100 0101 0000 0000 0000 0000 0100
1010 1100 0101 0000 0000 0000 0000 0000
1010 1100 0100 1111 0000 0000 0000 0100
```

回忆一下刚才做菜的比喻？

```
… , EXTop=1,ALUSelA=1,ALUSelB=11,ALUop=add,
IorD=1,Read,MemtoReg=1,RegWr=1,.....
```

软硬件交界面：ISA



软、硬件界面：指令集体系结构（Instruction Set Architecture，ISA）

有时简称系统结构、体系结构，指令系统，甚至简称“架构”

机器语言能被硬件直接执行。

指令集体系结构（ISA）

- ISA指Instruction Set Architecture，即指令集体系结构
- ISA是一种规约（Specification），它规定了如何使用硬件
  - 可执行的指令的集合，包括指令格式、操作种类以及每种操作对应的操作数的相应规定；
  - 指令可以接受的操作数的类型；
  - 操作数所能存放的寄存器组的结构，包括每个寄存器的名称、编号、长度和用途；
  - 操作数所能存放的存储空间的大小和编址方式；
  - 操作数在存储空间存放时按照大端还是小端方式存放；
  - 指令获取操作数的方式，即寻址方式；
  - 指令执行过程的控制方式，包括程序计数器、条件码定义等。
- ISA在计算机系统中是必不可少的一个抽象层，Why?
  - 没有它，软件无法使用计算机硬件！
  - 没有它，一台计算机不能称为“通用计算机”

软件（Software）

- System software(系统软件) - 简化编程，并使硬件资源被有效利用
  - 操作系统（Operating System）：硬件资源管理，用户接口
  - 语言处理系统：翻译程序+ Linker, Debug, etc …
    - 翻译程序(Translator)有三类：

      汇编程序(Assembler)：汇编语言源程序→机器目标程序

      编译程序(Complier)：高级语言源程序→汇编/机器目标程序

      解释程序(Interpreter)：将高级语言语句逐条翻译成机器指令并立即执行,不生成目标文件。
  - 其他实用程序: 如：磁盘碎片整理程序、备份程序等
- Application software(应用软件) - 解决具体应用问题/完成具体应用
  - 各类媒体处理程序：Word/ Image/ Graphics/…
  - 管理信息系统 (MIS)
  - Game，…

第一讲 计算机系统概述

- 冯.诺依曼结构计算机
  - 冯.诺依曼结构基本思想
  - 计算机硬件的基本组成
- 程序的表示和执行过程
  - 机器级语言和高级编程语言
  - 翻译程序：汇编、编译、解释
- 计算机系统抽象层
  - 计算机硬件和软件的接口：指令系统
  - 本课程内容在计算机系统中的位置

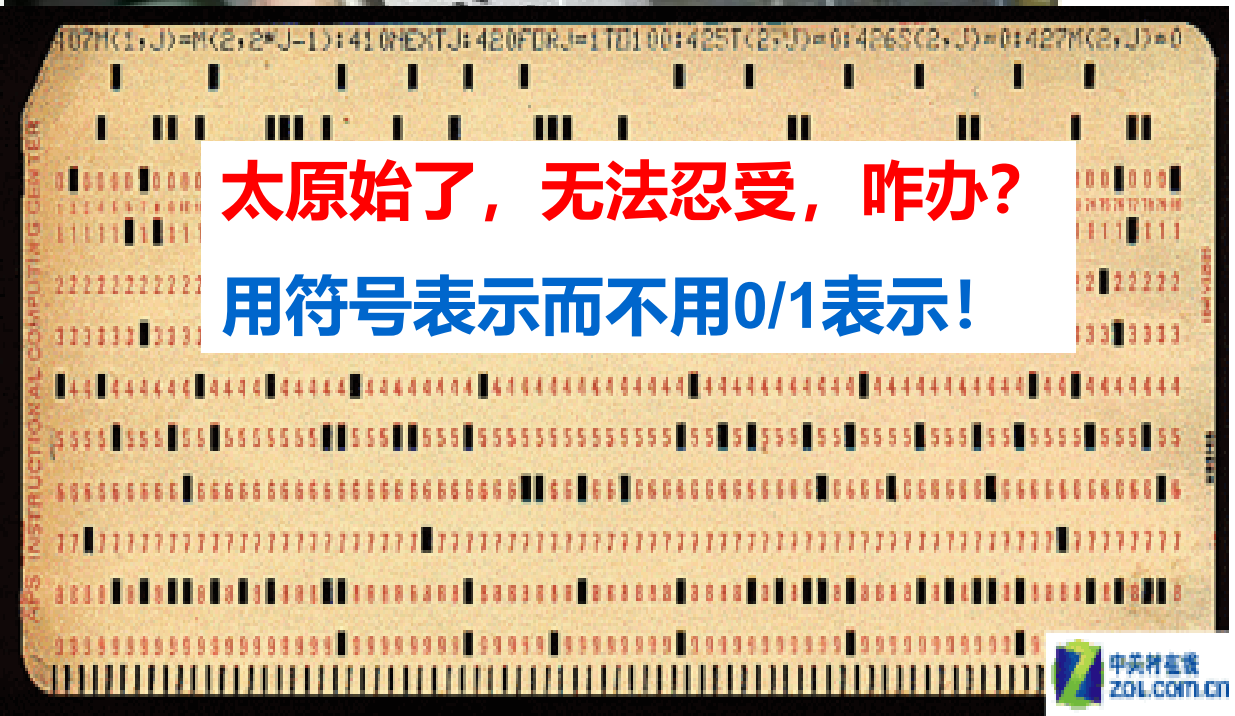
最早的程序开发过程

- **用机器语言编写程序，并记录在纸带或卡片上**

输入：按钮、开关；所有信息都
输出：指示灯等　　是0/1序列！

假设：0010-jxx　转移指令

```
0:  0101 0110
1:  0010 0100
2:  ......
3:  ......
4:  0110 0111
5:  ......
6:  ......
```

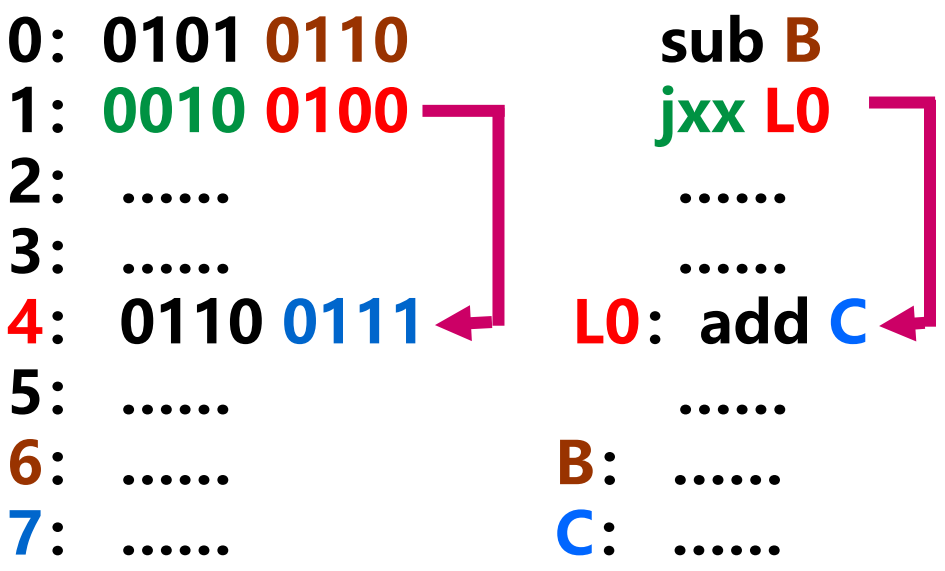


若在第4条指令前加入指令，则需重新计算地址码（如jxx的目标地址），然后重新打孔。不灵活！

书写、阅读困难！

用汇编语言开发程序

- 若用符号表示跳转位置和变量位置，是否简化了问题?
- 于是，汇编语言出现
  - 用助记符表示操作码
  - 用标号表示位置
  - 用助记符表示寄存器
  - .....



你认为用汇编语言编写的优点是:

不会因为增减指令而需要修改其他指令

不需记忆指令码，写程序比机器语言方便

可读性也比机器语言强

不过，这带来新的问题，是什么呢?

人容易了，可机器不认识这些指令了!

需将汇编语言转换为机器语言!

用汇编程序转换

在第4条指令前加指令时不用改变sub、jnz和add指令中的地址码!

进一步认识机器级语言

- 汇编语言源程序由汇编指令构成
- 你能用一句话描述什么是汇编指令吗?
  - 用助记符和标号来表示的指令（与机器指令一一对应）
- 指令又是什么呢?
  - 包含操作码和操作数或其地址码
    （机器指令用二进制表示，汇编指令用符号表示）
  - 可以描述：取（或存一个数）
    两个数加（或减、乘、除、与、或等）
    根据运算结果判断是否转移执行
- 想象用汇编语言编写复杂程序是怎样的情形?
  （例如，用汇编语言实现排序（sort）、矩阵相乘）
  - 需要描述的细节太多了！程序会很长很长！而且在不同结构的机器上就不能运行!

```
    sub B
    jxx L0
    ......
    ......
L0: add C
    ......
B:  ......
C:  ......
```

机器语言和汇编语言都是面向机器结构的语言，故它们统称为机器级语言

SKIP

结论：用汇编语言比机器语言好，但是，还是很麻烦!

指令所能描述的功能

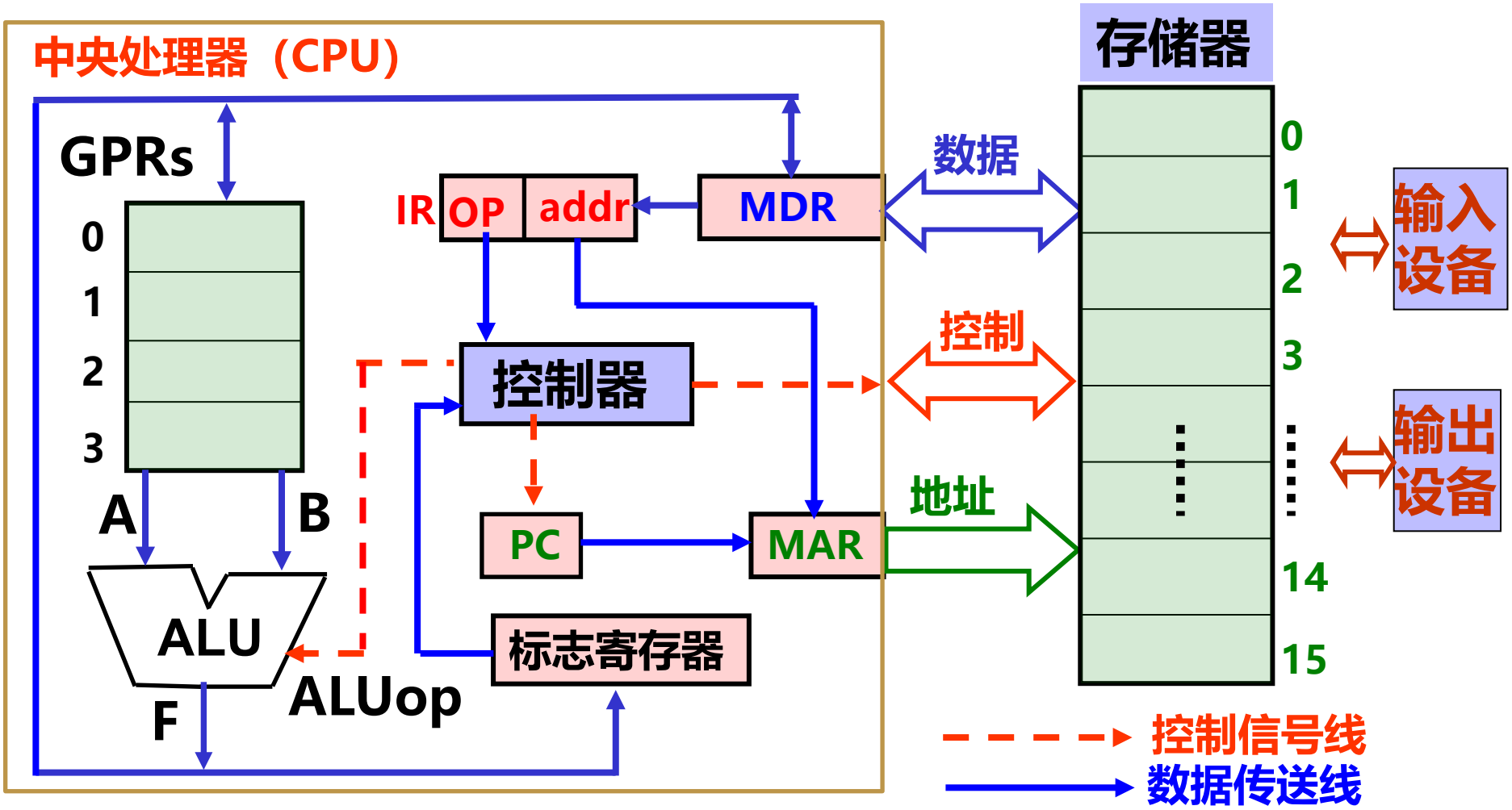
对于以下结构的机器，你能设计出几条指令吗?

Load M#，R#　　(将存储单元内容装入寄存器)

Store R#，M#　　(将寄存器内容装入存储单元)

Add R#，R#　(类似的还有Sub，Mul等；操作数还可“R#，M#”等)

BACK



用高级语言开发程序

- **随着技术的发展，出现了许多高级编程语言**
  - 它们与具体机器结构无关
  - 面向算法描述，比机器级语言描述能力强得多
  - **高级语言中一条语句对应几条、几十条甚至几百条指令**
  - 有“面向过程”和“面向对象”的语言之分
  - **处理逻辑分为三种结构**
    - **顺序结构、选择结构、循环结构**

现在，几乎所有程序员都用高级语言编程，但最终要将高级语言转换为机器语言程序

  - **有两种转换方式：“编译”和“解释”**
    - 编译程序(Complier)：将高级语言源程序转换为机器级目标程序，执行时只要启动目标程序即可
    - 解释程序(Interpreter)：将高级语言语句逐条翻译成机器指令并立即执行，不生成目标文件。

不同层次语言之间的等价转换


```
tmp = a[i];
a[i] = a[i+1];
a[i+1] = tmp;
```

```
lw $8, 0($2)
lw $9, 4($2)
sw $9, 0($2)
sw $8, 4($2)
```

每条指令由操作码和若干地址码组成

100011	00010	01000	0000000000000000
100011	00010	01001	0000000000000100
101011	00010	01001	0000000000000000
101011	00010	01000	0000000000000100

…, EXTop=1,ALUSelA=1,ALUSelB=11,ALUop=add, IorD=1,Read,MemtoReg=1,RegWr=1, …

任何高级语言程序最终通过执行若干条机器指令来完成！

开发和运行程序需什么支撑？

- 最早的程序开发很简单（怎样简单？）
  - 直接输入指令和数据，启动后把第一条指令地址送PC开始执行
- 用高级语言开发程序需要复杂的支撑环境（怎样的环境？）
  - 需要编辑器编写源程序
  - 需要一套翻译转换软件处理各类源程序
    - 编译方式：预处理程序、编译器、汇编器、链接器
    - 解释方式：解释程序
  - 需要一个可以执行程序的界面（环境）
    - GUI方式：图形用户界面
    - CUI方式：命令行用户界面

语言处理程序

语言处理系统 ＋

人机接口 ＋ 操作系统

语言的运行时系统

操作系统内核

指令集体系结构

计算机硬件

支撑程序开发和运行的环境由系统软件提供

最重要的系统软件是操作系统和语言处理系统

语言处理系统运行在操作系统之上，操作系统利用指令管理硬件

现代（传统）计算机系统的层次

- **现代计算机用高级语言编程**
  - 过程式语言
  - 非过程化语言

语言处理系统包括：各种语言处理程序（如编译、汇编、链接）、运行时系统（如库函数、调试、优化等功能）

操作系统包括人机交互界面、提供服务功能的内核例程

计算机系统抽象层的转换

功能**转换**：上层是下层的**抽象**，下层是上层的**实现**

底层为上层提供支撑环境！

程序执行效果（结果和性能等）

不仅取决于

算法、程序编写

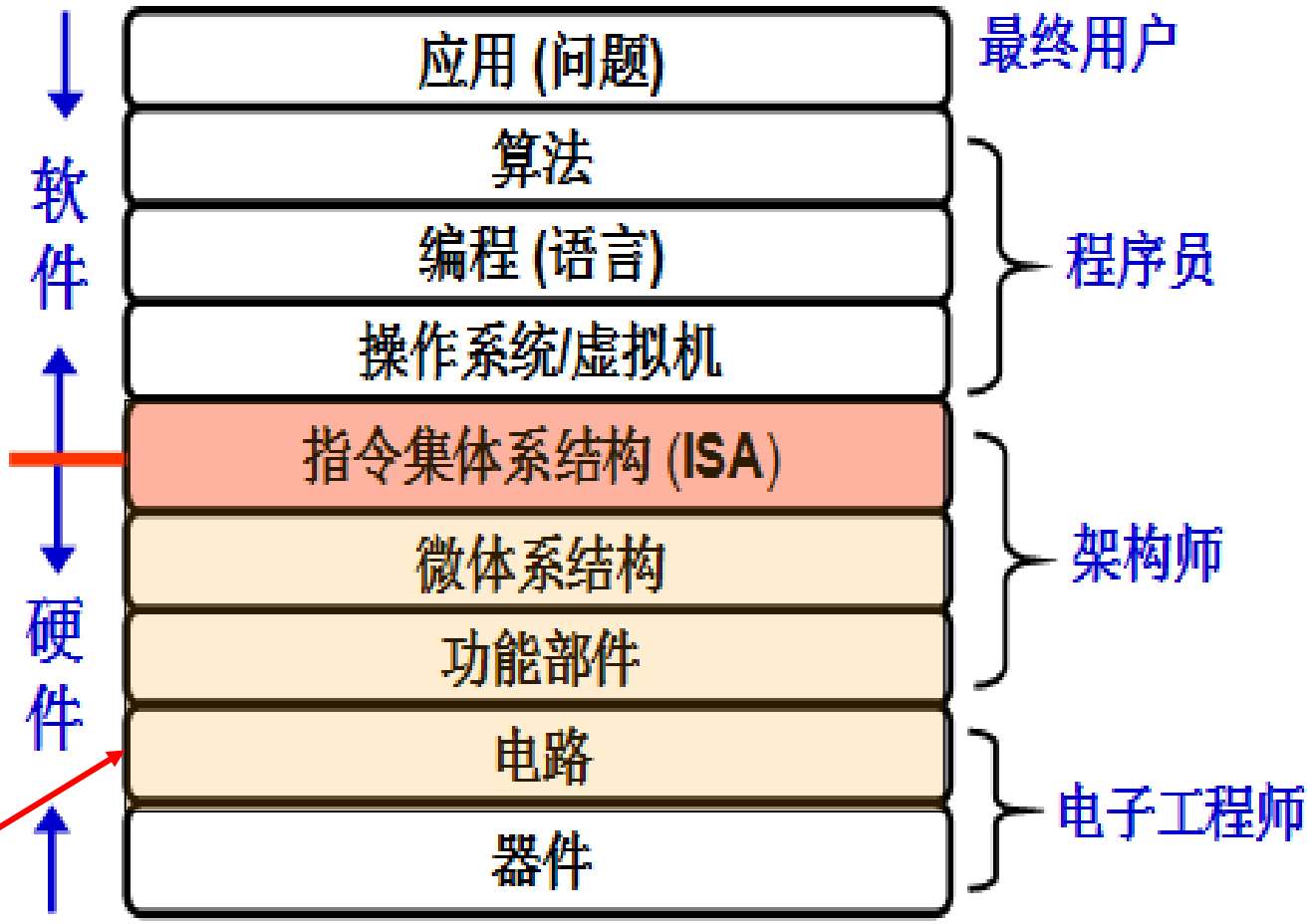
而且取决于

语言处理系统

操作系统

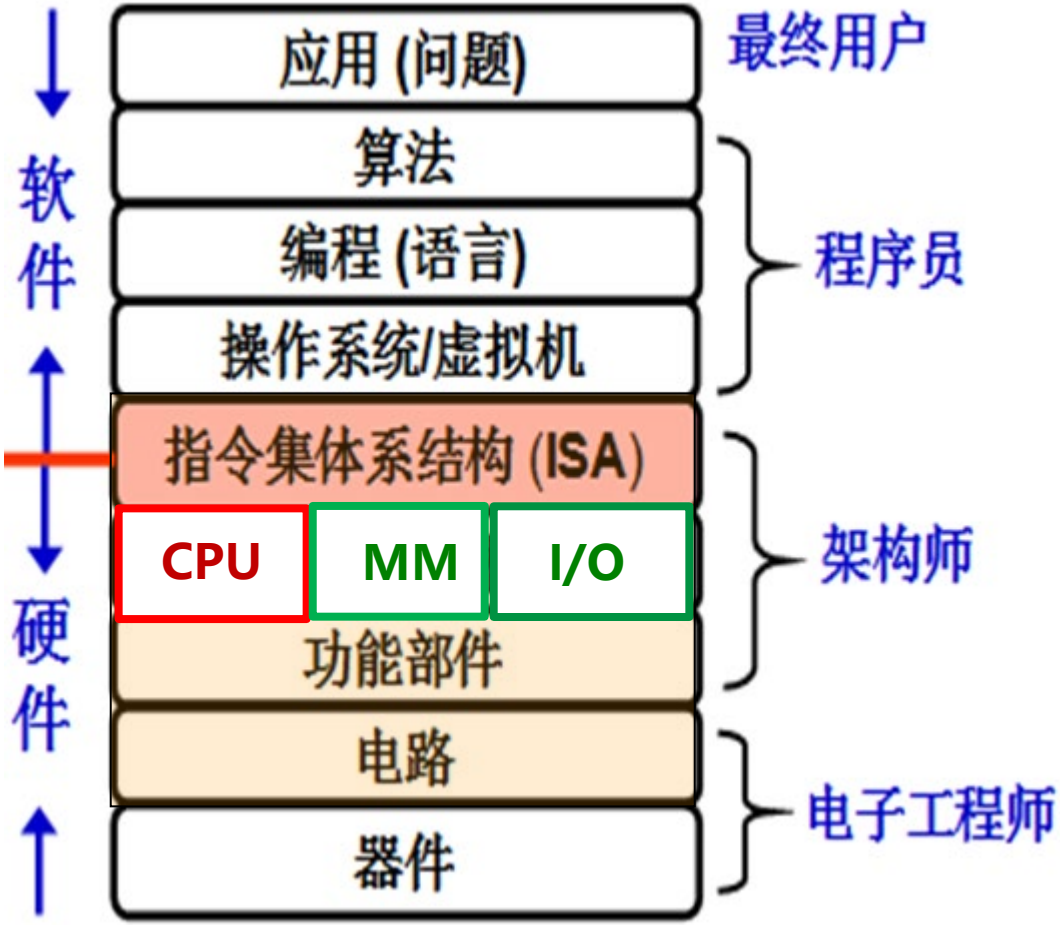
ISA

微体系结构



本课程教学内容：数字电路→功能部件→ISA →微架构（CPU、存储器、I/O）

本课程教学内容安排



主要内容

二进制编码

数字逻辑电路

硬件描述语言（实验课）

运算功能部件

指令集体系结构

中央处理器（CPU）

存储器层次结构（ICS）

系统互连与输入/出（ICS）

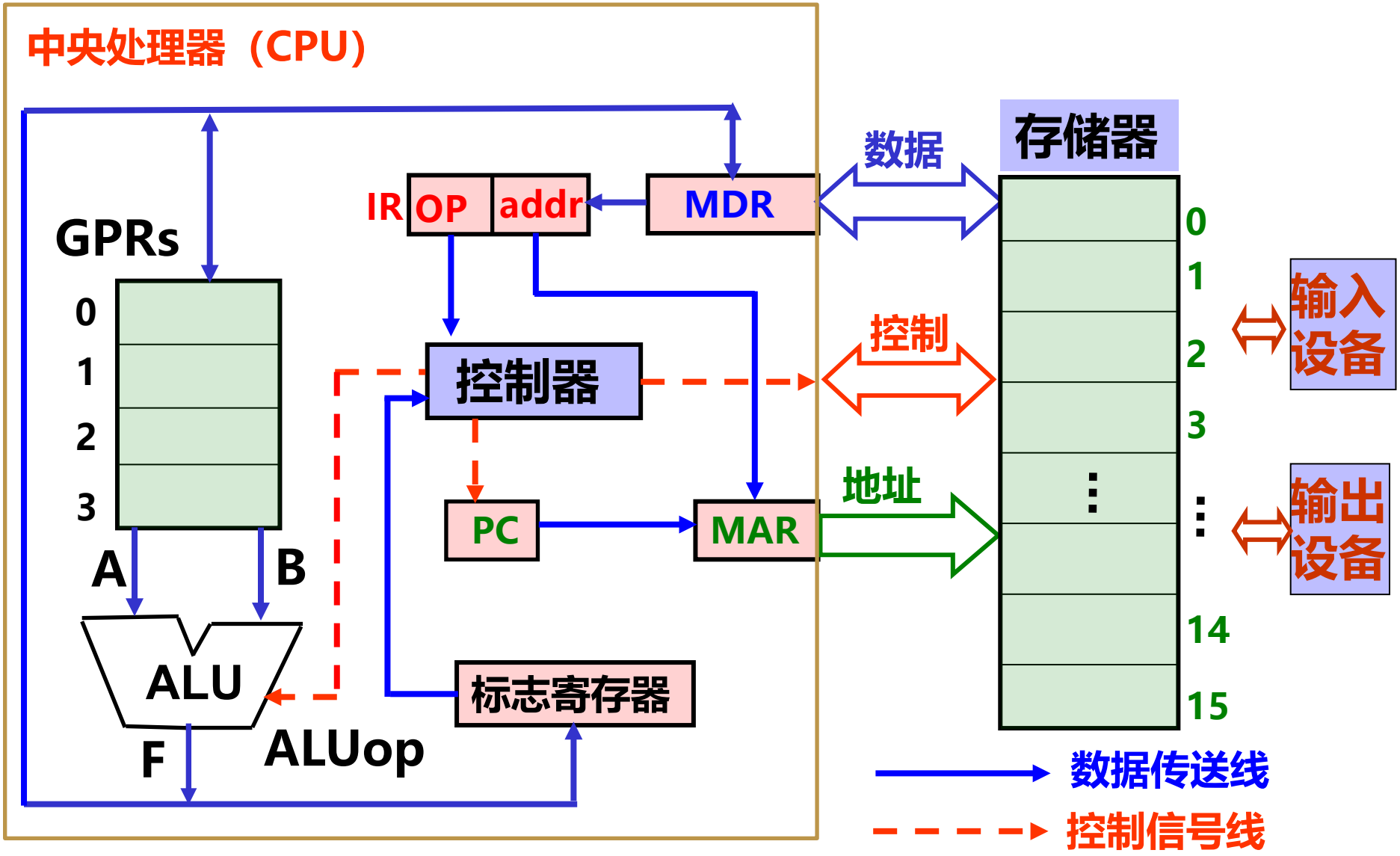
下次课开始学习二进制数的表示

在此之前先举一个程序和指令执行的简单例子

（请同学们先预习后面9张幻灯片中的内容）

程序和指令执行过程举例

8位模型机M：8位定长指令字，4个GPR，16个主存单元



程序和指令执行过程举例

假设模型机M中8位指令，格式有两种：R型、M型

R--寄存器（Register）M—存储器（Memory）

格式	4位	2位	2位	功能说明
R型	op	rt	rs	R[rt] ← R[rt] op R[rs] 或 R[rt] ← R[rs]
M型	op	addr		R[0] ← M[addr] 或 M[addr] ← R[0]

rs和rt为通用寄存器编号；addr为主存单元地址

R型：op=0000，寄存器间传送（mov）； op=0001，加（add）

M型：op=1110，取数（load）；op=1111，存数（store）

问题：指令 1110 0111的功能是什么?

答：因为op=1110，故是M型load指令，功能为：

R[0] ←M[0111]，即：将主存地址0111（7号单元）中的8位数据装入到0号寄存器中。

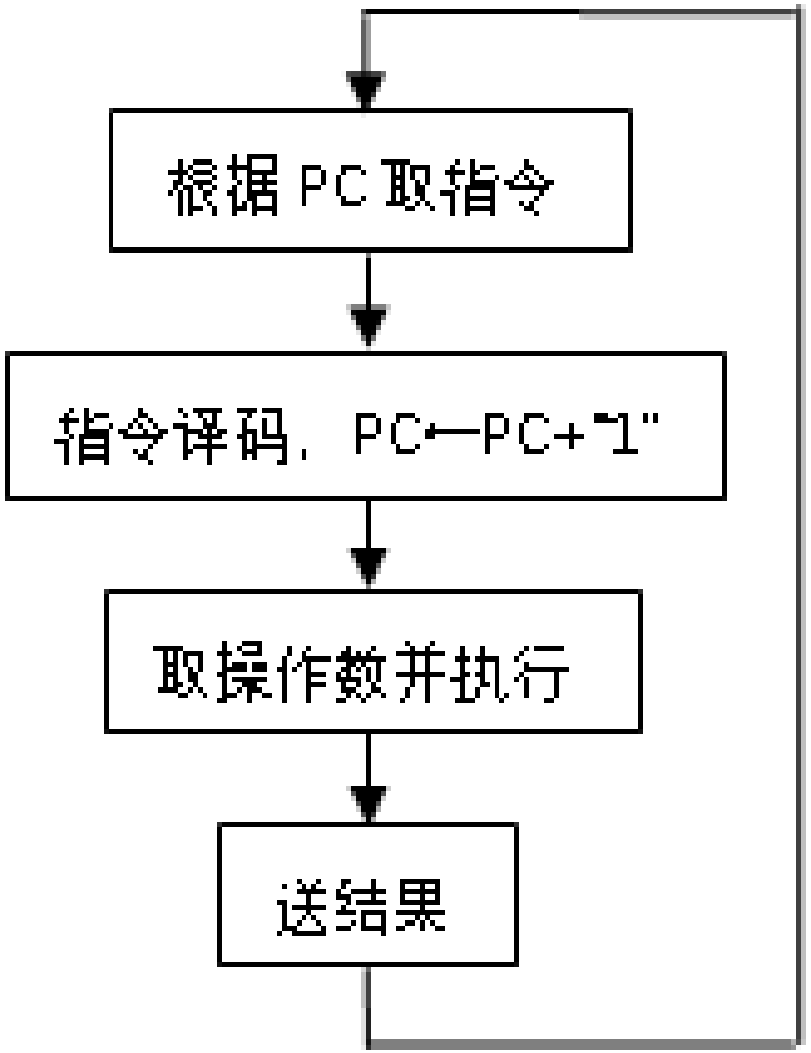
程序和指令执行过程举例

若在M上实现“z=x+y”，x和y分别存放在主存5和6号单元中，结果z存放在7号单元中，则程序在主存单元中的初始内容为：

主存地址	主存单元内容	内容说明（Ii表示第i条指令）	指令的符号表示
0	1110 0110	I1： R[0] ← M[6]；op=1110：取数操作	load r0, 6#
1	0000 0100	I2： R[1] ← R[0]；op=0000：传送操作	mov r1, r0
2	1110 0101	I3： R[0] ← M[5]；op=1110：取数操作	load r0, 5#
3	0001 0001	I4： R[0] ← R[0] + R[1]；op=0001：加操作	add r0, r1
4	1111 0111	I5： M[7]← R[0]；op=1111：存数操作	store 7#, r0
5	0001 0000	操作数x，值为16	
6	0010 0001	操作数y，值为33	
7	0000 0000	结果z，初始值为0	

程序执行过程及其结果是什么?

程序和指令执行过程举例



程序执行过程

指令I1（PC=0）的执行过程

	I1：1110 0110
取指令	IR←M[0000]
指令译码	op=1110，取数
PC增量	PC←0000+1
取数并执行	MDR←M[0110]
送结果	R[0]←MDR
执行结果	R[0]=33

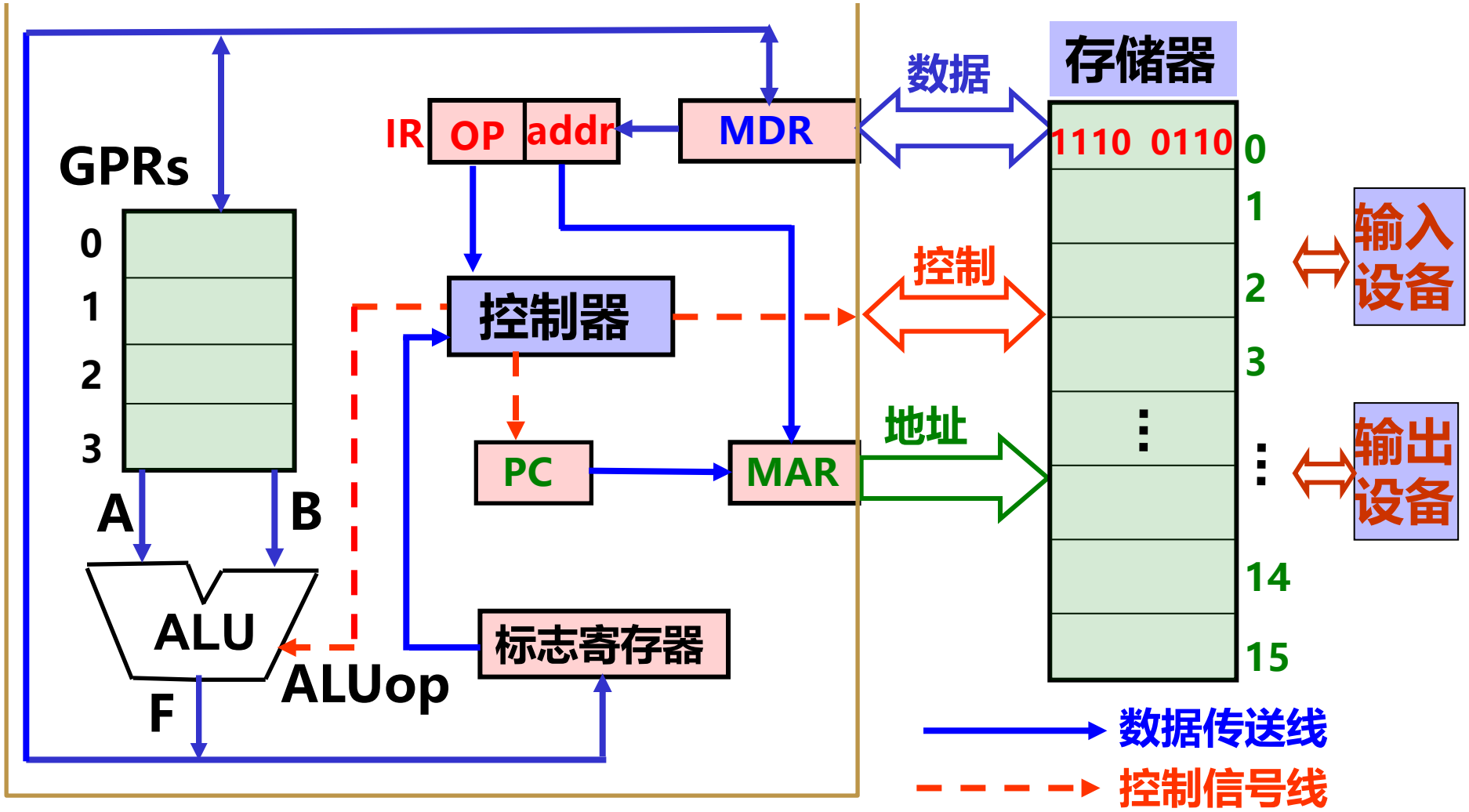
随后执行PC=1中的指令I2

程序和指令执行过程举例

指令 1110 0110功能为R[0] ←M[0110]，指令执行过程如下：

取指IR←M[PC]：MAR←PC；控制线←Read；IR←MDR

取数R[0]←M[addr]：MAR←addr；控制线←Read；R[0]←MDR



程序和指令执行过程举例

	I2：0000 0100	I3：1110 0101
取指令	IR←M[0001]	IR←M[0010]
指令译码	op=0000，传送	op=1110，取数
PC增量	PC←0001+1	PC←0010+1
取数并执行	A←R[0]、mov	MDR←M[0101]
送结果	R[1]←F	R[0]←MDR
执行结果	R[1]=33	R[0]=16

程序和指令执行过程举例

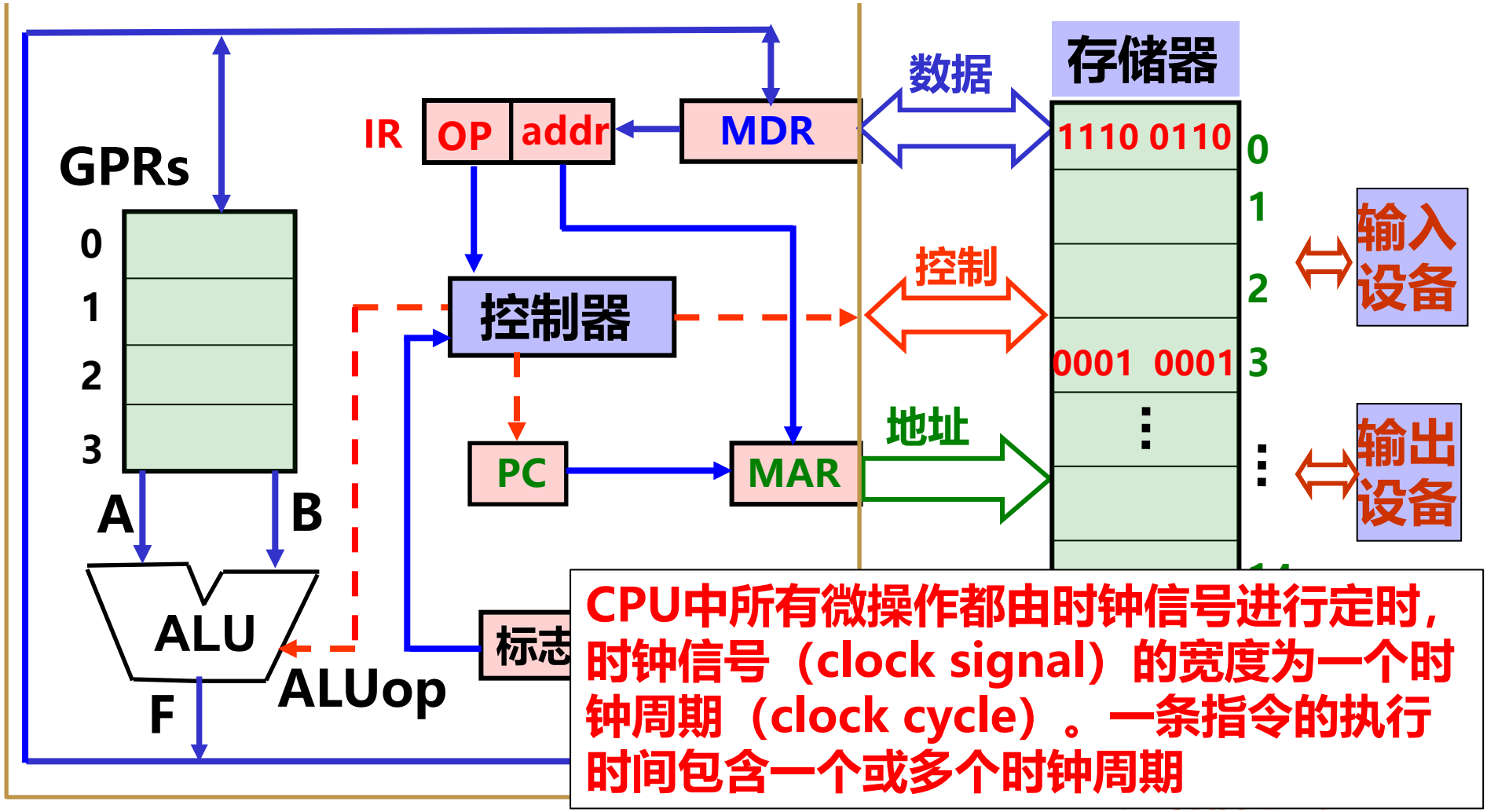
	I4：0001 0001	I5：1111 0111
取指令	IR←M[0011]	IR←M[0100]
指令译码	op=0001，加	op=1111，存数
PC增量	PC←0011+1	PC←0100+1
取数并执行	A←R[0]、B←R[1]、add	MDR←R[0]
送结果	R[0]←F	M[0111]←MDR
执行结果	R[0]=16+33=49	M[7]=49

程序和指令执行过程举例

指令 0001 0001功能为R[0]←R[0]+R[1]，指令执行过程如下

ALU运算R[0]←R[0]+R[1]的微操作（在控制信号的控制下完成）：

A←R[0]；B←R[1]；ALUop←add；R[0]←F



CPU中所有微操作都由时钟信号进行定时，时钟信号（clock signal）的宽度为一个时钟周期（clock cycle）。一条指令的执行时间包含一个或多个时钟周期

程序和指令执行过程举例

若在M上实现“z=x+y”，x和y分别存放在主存5和6号单元中，结果z存放在7号单元中，则程序在主存单元中的初始内容为：

主存地址	主存单元内容	内容说明（Ii表示第i条指令）	指令的符号表示
0	1110 0110	I1： R[0] ← M[6]；op=1110：取数操作	load r0, 6#
1	0000 0100	I2： R[1] ← R[0]；op=0000：传送操作	mov r1, r0
2	1110 0101	I3： R[0] ← M[5]；op=1110：取数操作	load r0, 5#
3	0001 0001	I4： R[0] ← R[0] + R[1]；op=0001：加操作	add r0, r1
4	1111 0111	I5： M[7]← R[0]；op=1111：存数操作	store 7#, r0
5	0001 0000	操作数x，值为16	
6	0010 0001	操作数y，值为33	
7	0000 0000	结果z，初始值为0	

问题：在计算机中，源码(如z=x+y;）、地址、主存中的指令和数据、汇编代码（如load r0,6#）等如何表示呢?

用二进制进行编码表示！