

第3讲 冯.诺依曼计算机： 机器级程序及其执行

理解“程序和数据如何被存储、如何被执行的”，有助于构造和设计可执行的算法和程序，有助于复杂环境下程序执行机制的理解

(机器级)算法的概念

计算方法1	计算方法2
Step1: 取出数3至运算器中	Step1: 取出数3至运算器中
Step2: 乘以数3在运算器中	Step2: 乘以数8在运算器中
Step3: 加上数8在运算器中	Step3: 加上数2在运算器中
Step4: 存储结果 8×3 在存储器	Step4: 乘以数3在运算器中
Step5: 取出数3至运算器中	Step5: 加上数6在运算器中
Step6: 乘以数3在运算器中	
Step7: 加上 $(8 \times 3)^2$ 在运算器中	
Step8: 加上数6在运算器中	

机器指令		对应的功能
操作码	地址码	
000001	000000010100	将存储单元的数 取出送到运算器。
存数	数	将运算器中的数 存送到 R 号存储单元。
000010	000000101000	
加法	Y	将运算器中的数 加上 Y 号存储单元的数。结果送到运算器。
000011	000000101010	
乘法	数	将运算器中的数 乘以 R 号存储单元的数。结果送到运算器。
000012	000000101011	
打印		打印指令
001010	000000111000	
存数	00001010	打印指令

操作码	地址码
000001	0000001000
000100	0000001001
000011	0000001010
000100	0000001000
000011	0000001011
000010	0000001100
000101	0000001100
000110	

[illegible]

Figure 1 illustrates the block diagram of a memory array structure. It shows an address decoder (地址译码器) on the left, which takes address inputs A_0, A_1, \dots, A_{n-1} and outputs word lines W_0, W_1, \dots, W_{n-1} to a memory matrix (存储矩阵). The memory matrix consists of storage units (存储单元) containing data words (存储字). Each storage unit is connected to a word line and a bit line (存储位). The bit lines are connected to a tri-state buffer (三态缓冲器) which outputs data D_0, D_1, \dots, D_{n-1} . A tri-state control signal (三态控制) is also shown.

[illegible]

基本思维：机器级算法与程序→机器指令与指令系统→存储器→存储程序→运算器与控制器→机器级程序的执行；算法程序化→程序指令化→指令存储化→执行信号化



图灵机的思想与模型简介

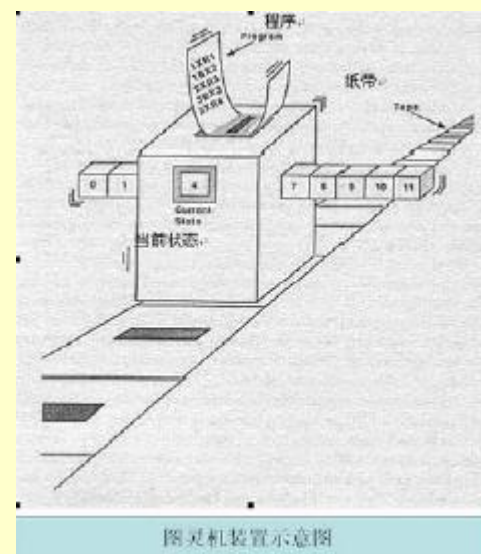
----图灵的贡献

----图灵机：计算机的理论模型

----指令、数据、程序与程序执行

图灵及其贡献

- ◆ **图灵** (Alan Turing, 1912~1954), 出生于英国伦敦, 19 岁入剑桥皇家学院, 22 岁当选为皇家学会会员。
- ◆ 1937 年, 发表了论文《论可计算数及其在判定问题中的应用》, 提出了 **图灵机模型**, 后来, 冯·诺依曼根据这个模型设计出历史上第一台电子计算机。
- ◆ 1950 年, 发表了划时代的文章: 《机器能思考吗?》, 成为了人工智能的开山之作。
- ◆ 计算机界于 1966 年设立了最高荣誉奖: **ACM 图灵奖**。

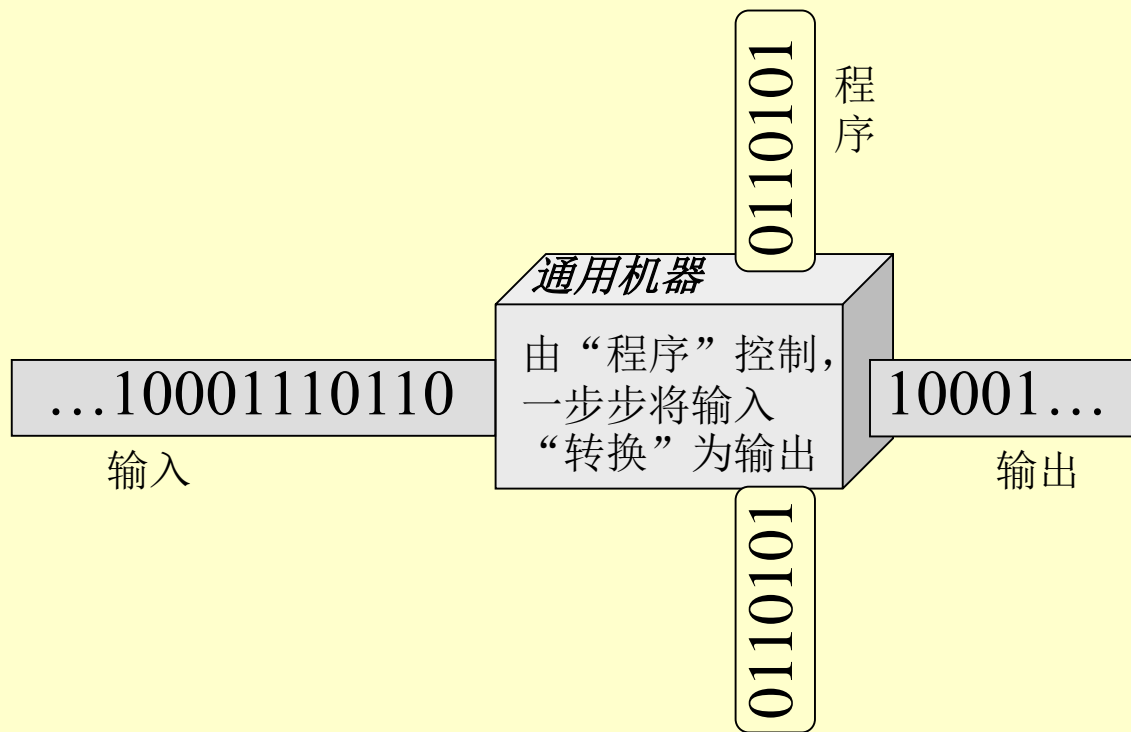


你能查阅一下哪些人获得图灵奖了吗？
因为什么贡献而获奖呢？



计算

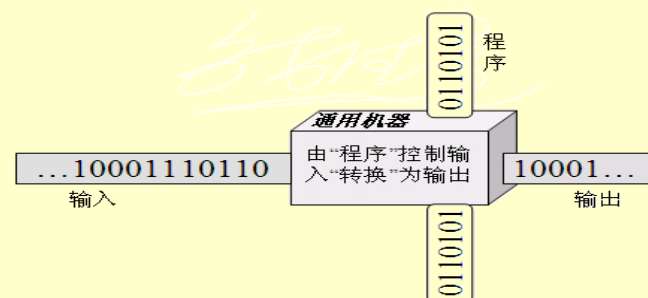
◆ 所谓**计算**就是计算者(人或机器)对一条两端可无限延长的纸带上的一串0或1，执行指令一步一步地改变纸带上的0或1，经过有限步骤最后得到一个满足预先规定的符号串的**变换过程**。



图灵机的思想

是关于数据、指令、程序及程序/指令自动执行的基本思想。

- ◆ 输入被制成一串0和1的纸带，送入机器中----**数据**。如00010000100011...
- ◆ 机器可对输入纸带执行的**基本动作**包括：“翻转0为1”，或“翻转1为0”，“前移一位”，“停止”。
- ◆ 对基本动作的控制----**指令**，机器是按照指令的控制选择执行哪一个动作，指令也可以用0和1来表示：**01**表示“翻转0为1”（当输入为1时不变），**10**表示“翻转1为0”（当输入0时不变），**11**表示“前移一位”，**00**表示“停止”。
- ◆ 输入如何变为输出的控制可以用指令编写一个**程序**来完成，如：011110110111011100...
- ◆ 机器能够读取程序，按程序中的指令顺序读取指令，读一条指令**执行**一条指令。由此实现**自动计算**。



图灵机模型

◆ 基本的**图灵机模型**为一个七元组,如右图示意

◆ 几点结论:

◆ (1) 图灵机是一种思想模型，它由一个控制器(有限状态转换器)，一条可无限延伸的带子和一个在带子上左右移动的读写头构成。

◆ (2) **程序是五元组 $\langle q, X, Y, R(\text{或}L\text{或}N), p \rangle$ 形式的指令集**。其定义了机器在一个特定状态 q 下从方格中读入一个特定字符 X 时所采取的动作作为在该方格中写入符号 Y , 然后向右移一格 R (或向左移一格 L 或不移动 N), 同时将机器状态设为 p 供下一条指令使用。

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

其中:

Q : 状态的有穷集合

q_0 : 开始状态

F : 终止状态集合

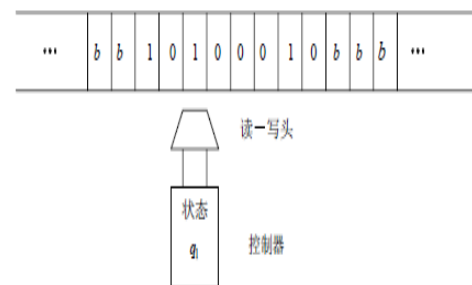
Γ : 带符号表

B : 空白符号

Σ : 输入字母表

δ : 移动函数, (1): $\delta(q, X) = (p, Y, R)$ 表示 M 在状态 q 读入符号 X , 将状态改为 p , 并在这个 X 所在的带方格中印刷符号 Y , 然后将读头向右移动一格。

(2): $\delta(q, X) = (p, Y, L)$ 表示 M 在状态 q 读入符号 X , 将状态改为 p , 并在这个 X 所在的带方格中印刷符号 Y , 然后将读头向左移动一格。





几点结论(续):

◆(3)图灵机模型被认为是计算机的基本理论模型

----计算机是使用相应的程序来完成任何设定好的任务。图灵机是一种离散的、有穷的、构造性的问题求解思路，一个问题的求解可以通过构造其图灵机(即程序)来解决。

◆(4)图灵认为：凡是能用算法方法解决的问题也一定能用图灵机解决；凡是图灵机解决不了的问题任何算法也解决不了----图灵可计算性问题。



冯.诺依曼计算机：思想与构成

----存储程序原理

----冯.诺依曼计算机的构成



冯·诺依曼(Von.Neumann)计算机

◆1944~1945年间，冯·诺伊曼提出“存储程序”的计算机设计思想，并进行了实践，现代计算机普遍来讲属于冯·诺伊曼机体系。

◆冯·诺伊曼机的基本思想：

- 运算和存储分离

- 存储程序：指令和数据以同等地位事先存于存储器，可按地址寻访，连续自动执行。

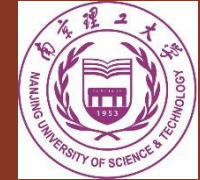
- 五大部件构成：运算器、控制器、存储器、输入设备和输出设备

- 指令和数据用二进制表示，指令由操作码和地址码组成

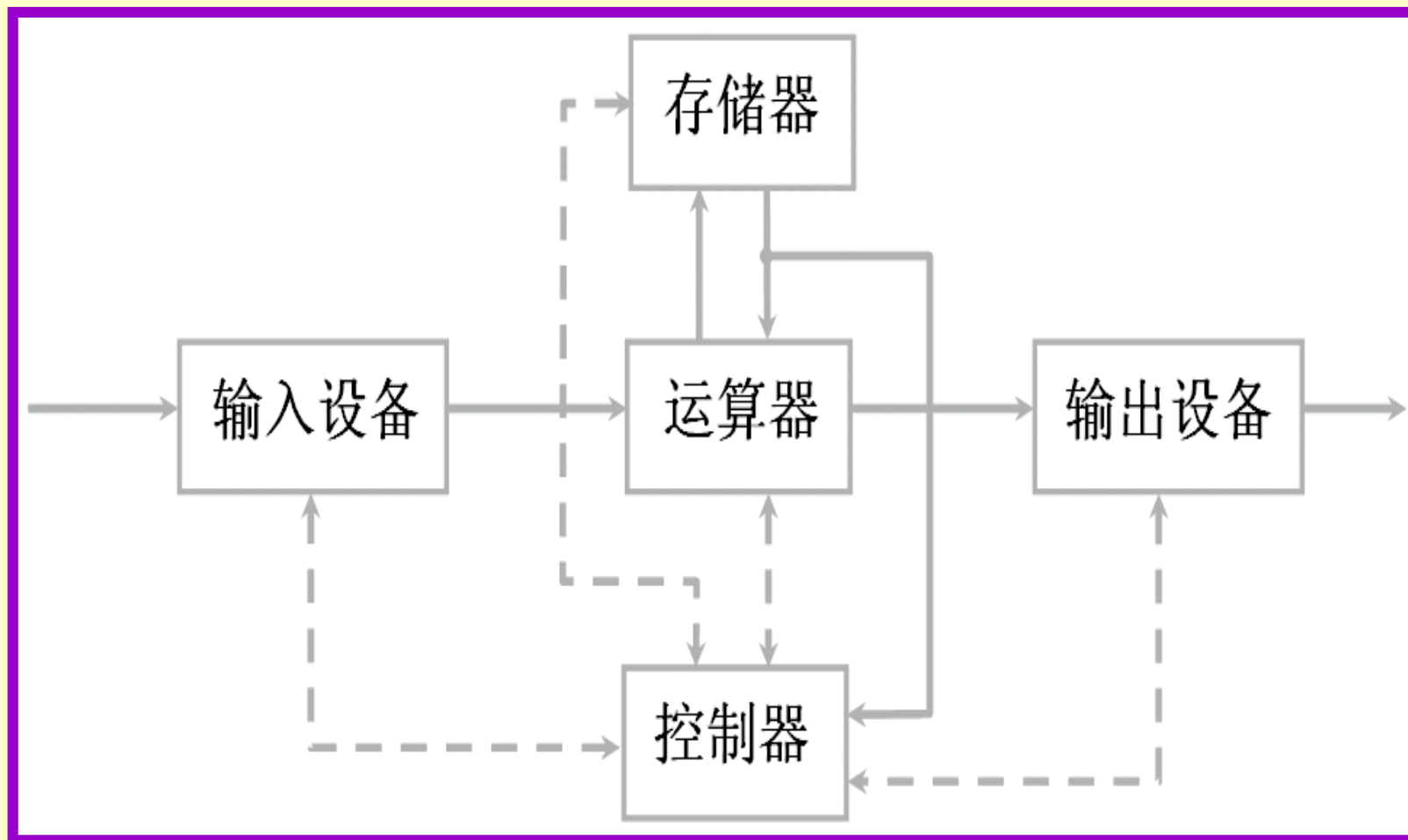
- 以运算器为中心，控制器负责解释指令，运算器负责执行指令

冯·诺依曼计算机的结构是怎样的？部件有哪些？
部件的关系怎样？

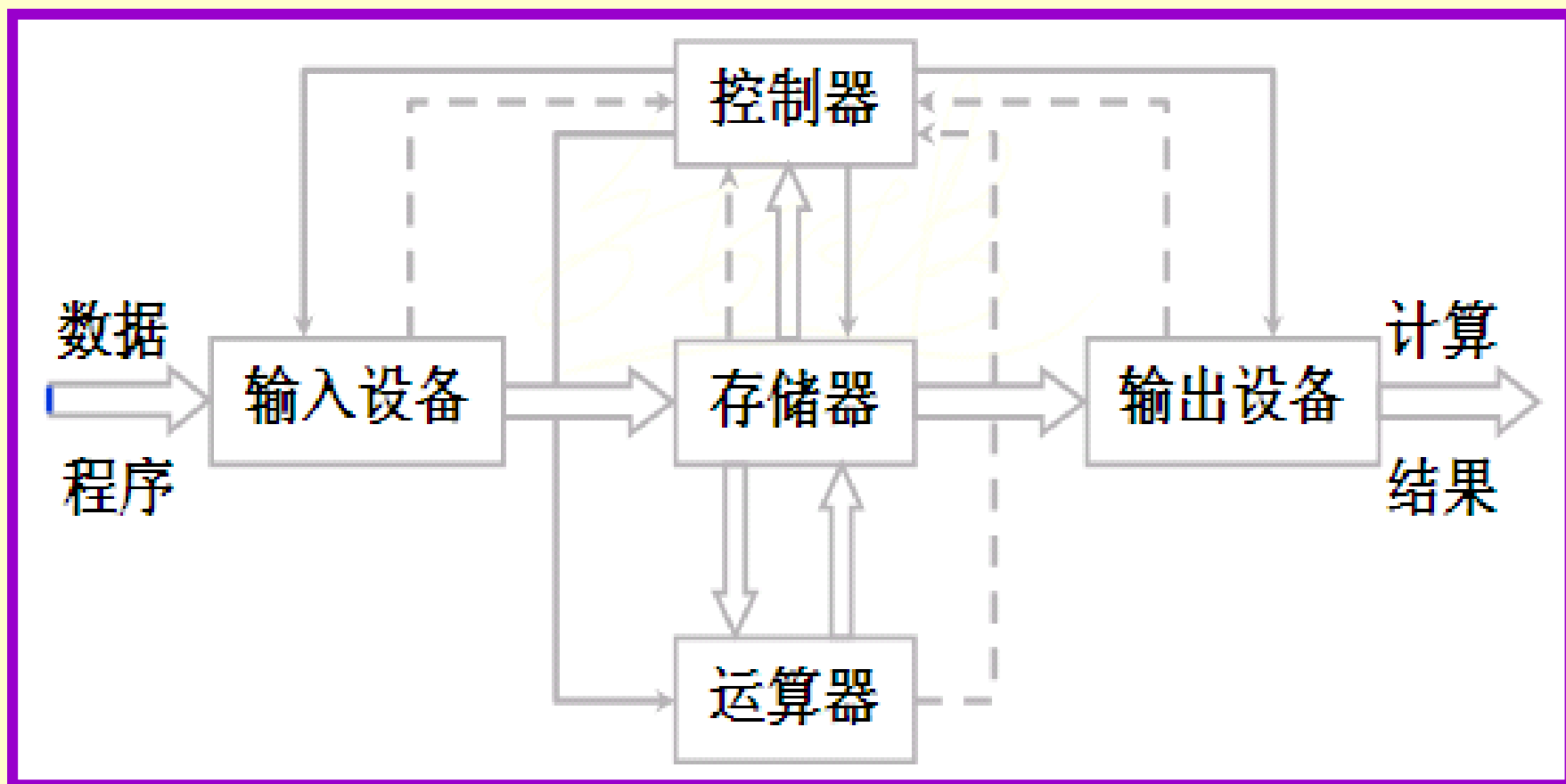
12/37



以运算器为中心的冯·诺依曼计算机构成图



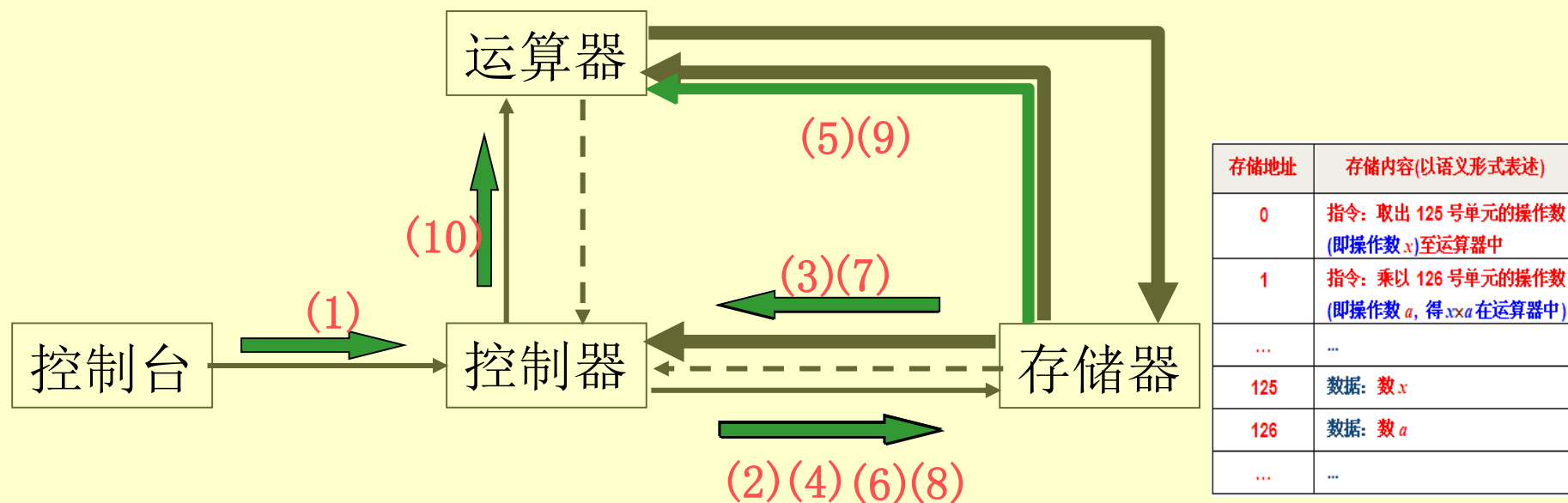
以存储器为中心的现代计算机构成图



同样是五个部件，以不同的结构来连接，便体现了不同的性能----这就是“系统”：强调“结构”，强调部件连接后的整体性、协同性



工作原理



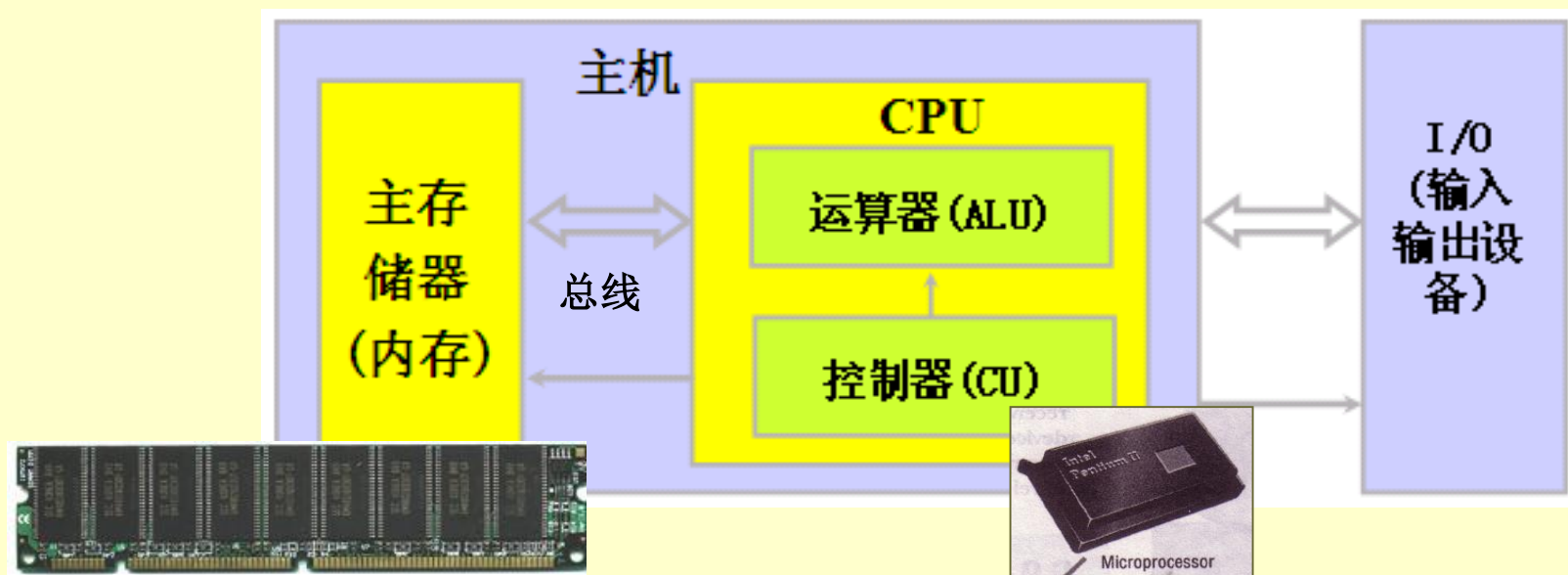
- (1) 启动控制器工作
- (2) 发送第1条指令地址
- (3) 取出指令并分析指令
- (4) 执行指令: 发送操作数 x 所在地址
- (5) 执行指令: 取出操作数 x

- (6) 发送下一条指令地址
- (7) 取出指令并分析指令
- (8) 执行指令: 发送操作数 a 所在地址
- (9) 执行指令: 取出操作数 a
- (10) 执行指令: 通知运算器计算 a 乘 x
- (11) 继续后续指令的取指、执行...

计算机的基本部件

◆**CPU**: 中央处理单元(Central Process Unit), 将运算器和控制器集成在一块芯片上, 形成微处理器。

◆**CPU、主存储器、I/O设备及总线**成为现代计算机的四大核心部件。



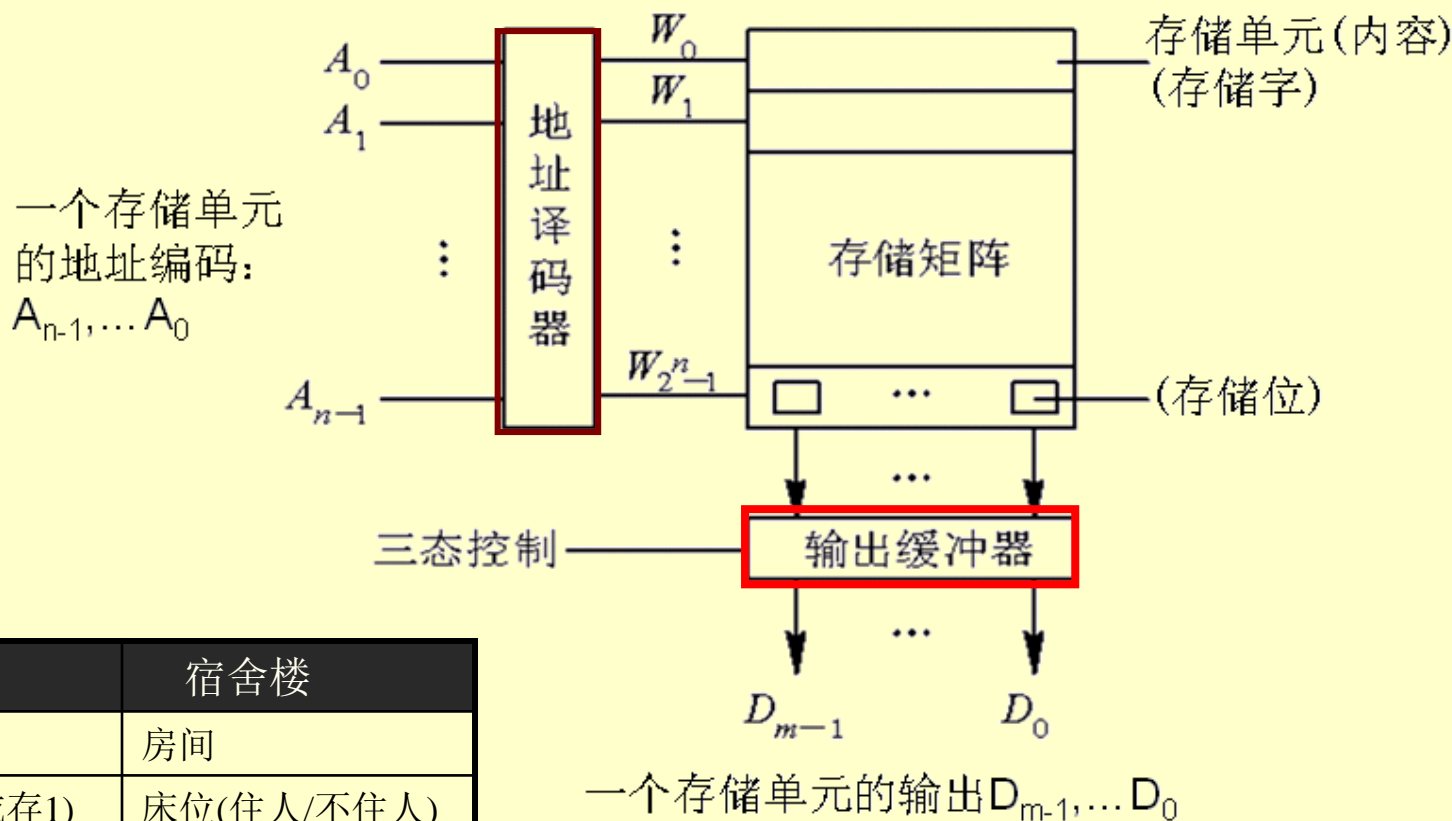
现代计算机里面, 一个微处理器(芯片)可能包含多个**CPU**, 即多核.



自动存取：存储器的工作原理

----存储器的自动读写

存储器的基本结构



存储器电路结构概念图

概念映射

存储器	宿舍楼
存储单元	房间
存储位(存0或存1)	床位(住人/不住人)
地址编码 $A_{n-1} \dots A_0$	房间号
单元控制线 W_i	房间钥匙
输出缓冲器	公共的走廊及大门
...	...

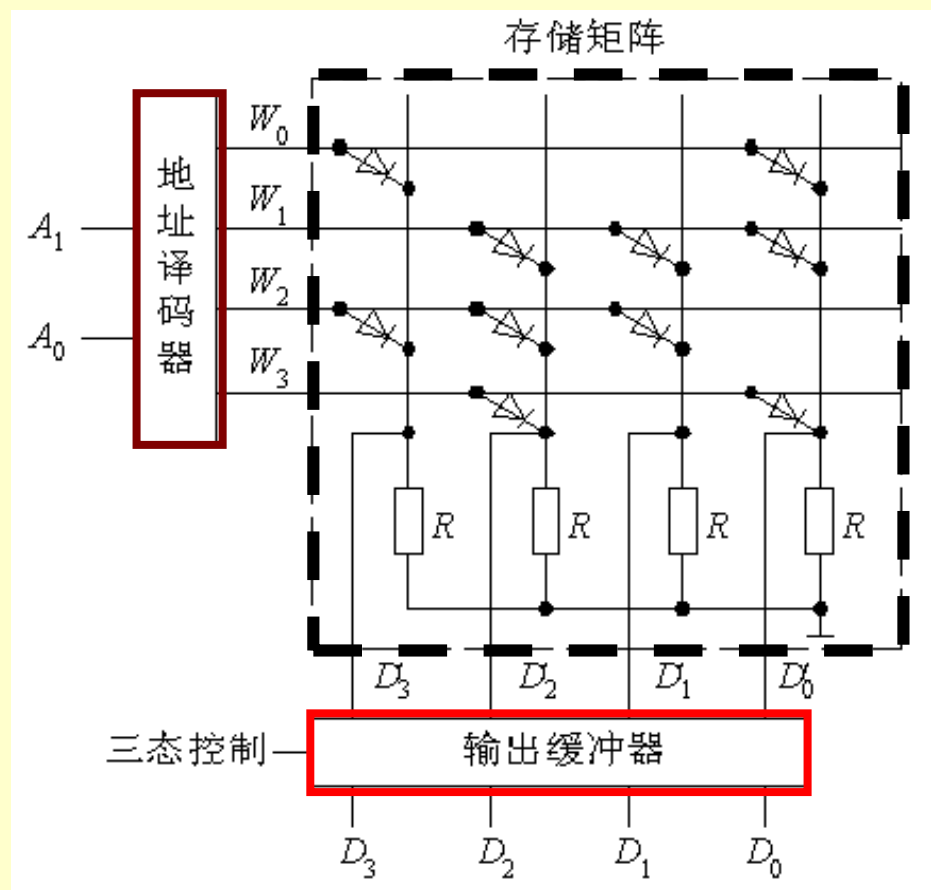
从存储器与宿舍楼的概念对比中，你能发现什么异同吗？

存储器内部的实现示例

◆当地址线和数据线间连接有二极管时，则存储的是1，否则，存储的是0

■当地址线和数据线间连接有二极管时，由地址线决定其是输出1或0，即：当地址线为高电平时，则输出1，而当地址线为低电平时，则输出0；

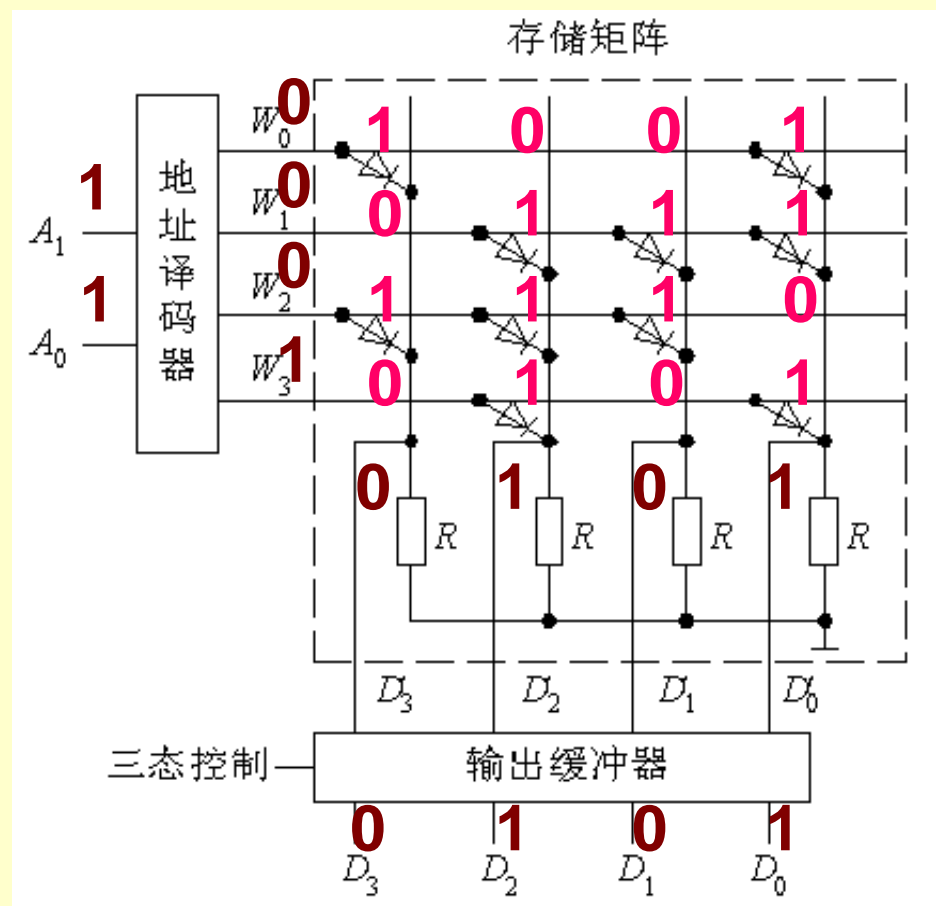
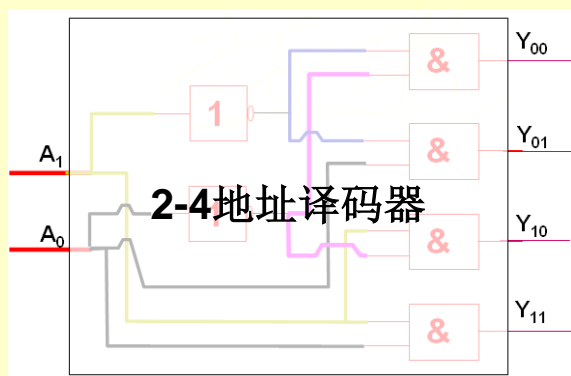
■没有连接的，则不受地址线影响，始终输出低电平0；



二极管ROM结构示例

(2位地址控制4个信息单元, 每个信息单元是4位0/1码)

存储器内部的实现示例



二极管ROM结构示例
(2位地址控制4个信息单元, 每个信息单元是4位0/1码)

存储矩阵的逻辑控制关系示例

$$\begin{aligned} W_0 &= (\text{NOT } A_0) \text{ AND } (\text{NOT } A_1) \\ W_1 &= A_0 \text{ AND } (\text{NOT } A_1) \\ W_2 &= (\text{NOT } A_0) \text{ AND } A_1 \\ W_3 &= A_0 \text{ AND } A_1 \end{aligned}$$

$$\begin{aligned} D_3 &= W_0 \text{ OR } W_2 \\ D_2 &= W_1 \text{ OR } W_2 \text{ OR } W_3 \\ D_1 &= W_1 \text{ OR } W_2 \\ D_0 &= W_0 \text{ OR } W_1 \text{ OR } W_3 \end{aligned}$$

高/低电平信号，即0,1，
通过连接点相互传递

W_i 是地址线

同一地址线上各连接点之间是“与”关系

A_k 是地址编码线

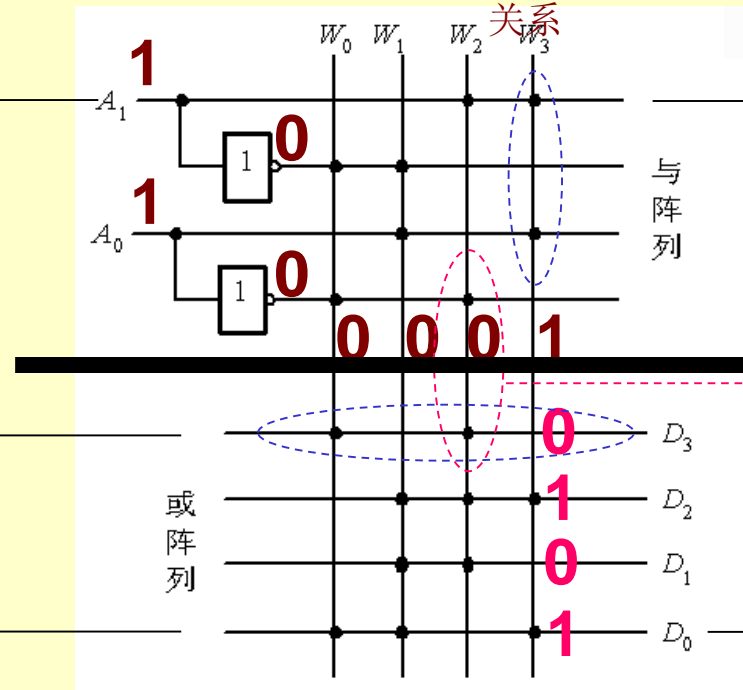
地址编码线与地址线有点连接，
无点不连接

同一条数据线上各连接点之间是“或”关系

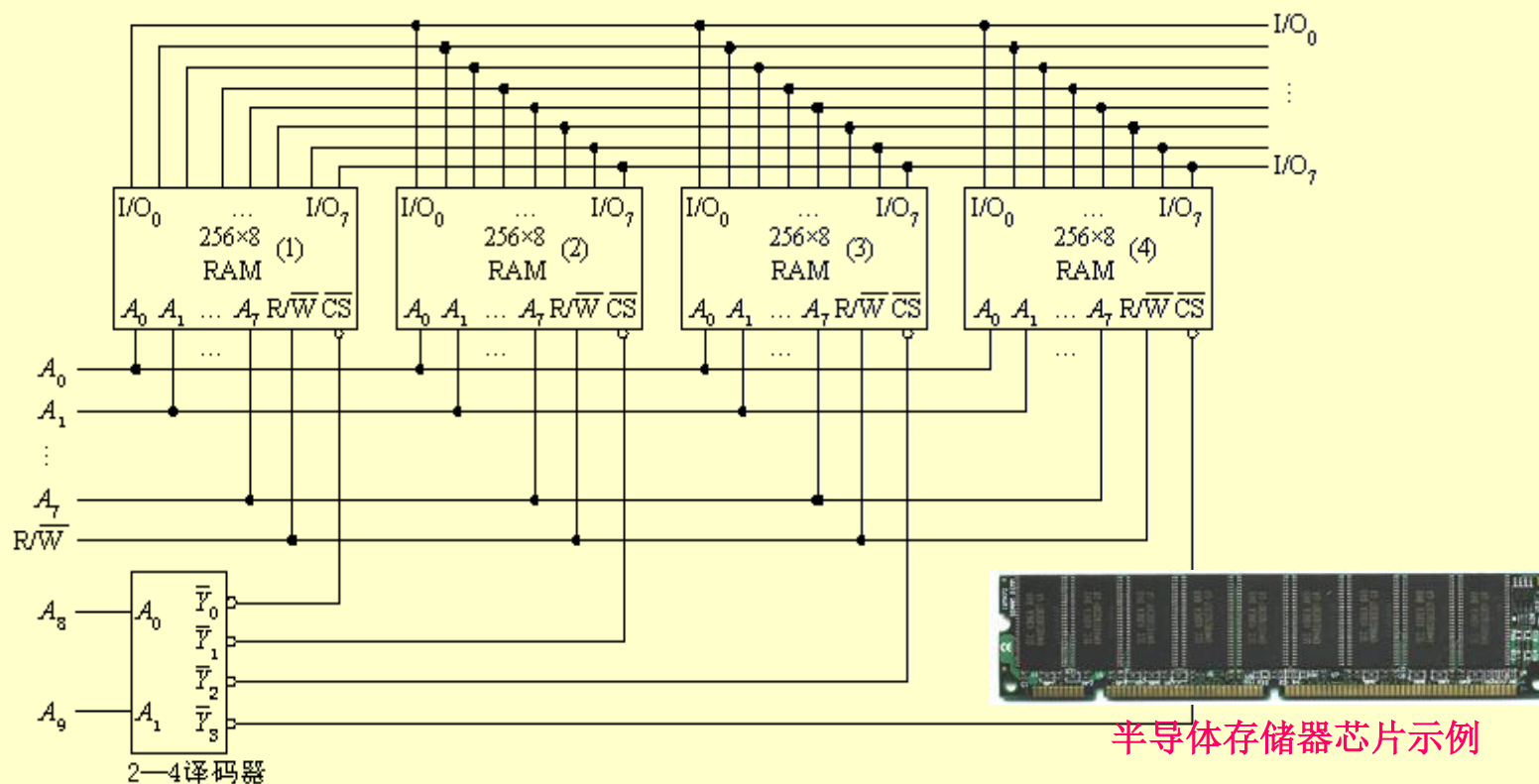
D_j 是数据线

上半区通过“与”关系产生地址线上的最终信号传递到下半区

地址线与数据线有点连接，无点不连接



用多个存储器芯片可搭建容量更大的存储器



利用4个256x8存储器芯片扩展出1024x8存储器的电路图

问：从概念的角度，你能说说存储器扩展要解决什么问题吗？
提示：地址编码空间, 存储字长.



2.2.4 机器级程序

机器级程序

----机器指令

----机器级算法与程序



算法---从冯.诺依曼计算机的角度

◆可在机器上执行的求解问题的操作规则及步骤, 被称为可执行的算法。

$$\text{计算 } 8 \times 3^2 + 2 \times 3 + 6 = ((8 \times 3) + 2) \times 3 + 6$$

计算方法1

- Step1:** 取出数3至运算器中
- Step2:** 乘以数3在运算器中
- Step3:** 乘以数8在运算器中
- Step4:** 存结果 8×3^2 在存储器中
- Step5:** 取出数2至运算器中
- Step6:** 乘以数3在运算器中
- Step7:** 加上 (8×3^2) 在运算器中
- Step8:** 加上数6在运算器中

计算方法2

- Step1:** 取出数3至运算器中
- Step2:** 乘以数8在运算器中
- Step3:** 加上数2在运算器中
- Step4:** 乘以数3在运算器中
- Step5:** 加上数6至运算器中

问: 怎么看待算法节省的步数? ---算法需要“优化”



机器指令

◆ 机器指令是CPU可以直接分析并执行的指令，一般由0和1的编码表示。

◆ 指令 \approx 操作码 + 地址码；

机器语言

000001 0000000100

000001 0000001100

000001 0000001000

机器指令		对应的功能
操作码	地址码	
取数	α	α 号存储单元的数 取出送到运算器；
000001	0000000100	
存数	β	运算器中的数 存储到 β 号存储单元；
000010	0000010000	
加法	γ	运算器中的数 加上 γ 号存储单元的数，结果保留在运算器；
000011	0000001010	
乘法	δ	运算器中的数 乘以 δ 号存储单元的数，结果保留在运算器；
000100	0000001001	
打印		打印指令
000101	0000001100	
停机		停机指令
000110		



机器级程序

◆ $8 \times 3^2 + 2 \times 3 + 6;$

◆ $ax^2 + bx + c.$

对应的十进制地址	存储单元的地址	存储单元的内容		说明
		操作码	地址码	
0	00000000 00000000	000001	0000001000	指令：取出 8 号存储单元的数(即 3)至运算器中
1	00000000 00000001	000100	0000001001	指令：乘以 9 号存储单元的数(即 8)得 8×3 在运算器中
2	00000000 00000010	000011	0000001010	指令：加上 10 号存储单元的数(即 2)得 $8 \times 3 + 2$ 在运算器中
3	00000000 00000011	000100	0000001000	指令：乘以 8 号存储单元的数(即 3)得 $(8 \times 3 + 2) \times 3$ 在运算器中
4	00000000 00000100	000011	0000001011	指令：加上 11 号存储单元的数(即 6)得 $8 \times 3^2 + 2 \times 3 + 6$ 至运算器中
5	00000000 00000101	000010	0000001100	指令：将上述运算器中结果存于 12 号存储单元
6	00000000 00000110	000101	0000001100	指令：打印
7	00000000 00000111	000110		指令：停机
8	00000000 00001000	000000 0000000011		数据：数 3 存于 8 号单元
9	00000000 00001001	000000 0000001000		数据：数 8 存于 9 号单元
10	00000000 00001010	000000 0000000010		数据：数 2 存于 10 号单元
11	00000000 00001011	000000 0000000011		数据：数 6 存于 11 号单元
12	00000000 00001100			数据：存放结果



高级语言程序的示例

计算 ax^2+bx+c

其中 a, x, b, c 是变量。

变量的地址是由编译程序在编译过程中自动分配的，也即是说编译器根据当时编译的情况，分配 a, x, b, c 为8号，9号，10号，11号存储单元，并产生上述的机器指令程序

```
Main() {  
    int result;      //定义变量 result  
    int  x;          //定义变量 x  
    int  a;          //定义变量 a  
    int  b;          //定义变量 b  
    int  c;          //定义变量 c  
    x = 3;           //将 3 赋值给 x  
    //数据赋值过程中也可在运行过程中进行  
    a = 8;           //将 8 赋值给 a  
    b = 2;           //将 2 赋值给 b  
    c = 6;           //将 6 赋值给 c  
    result = a * x * x + b * x + c;  
    //计算 a * x * x + b * x + c 并赋值给 result  
    print  result;   //打印 result 的值  
}
```



2.2.5 机器级程序的执行机制

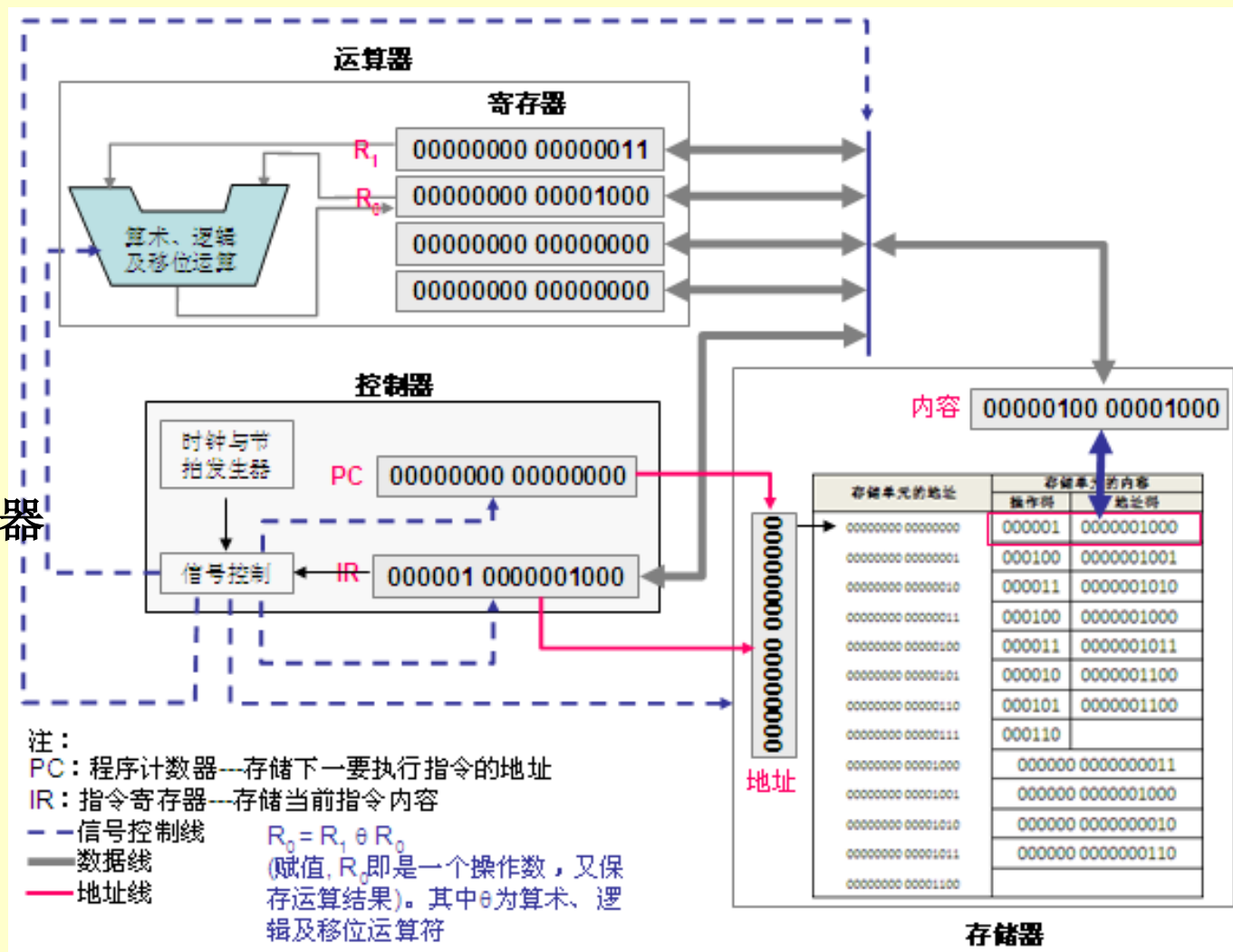
机器级程序的执行机制

----运算部件与控制部件

----指令的信号化与节拍化

计算机各部件内部的简单构成关系

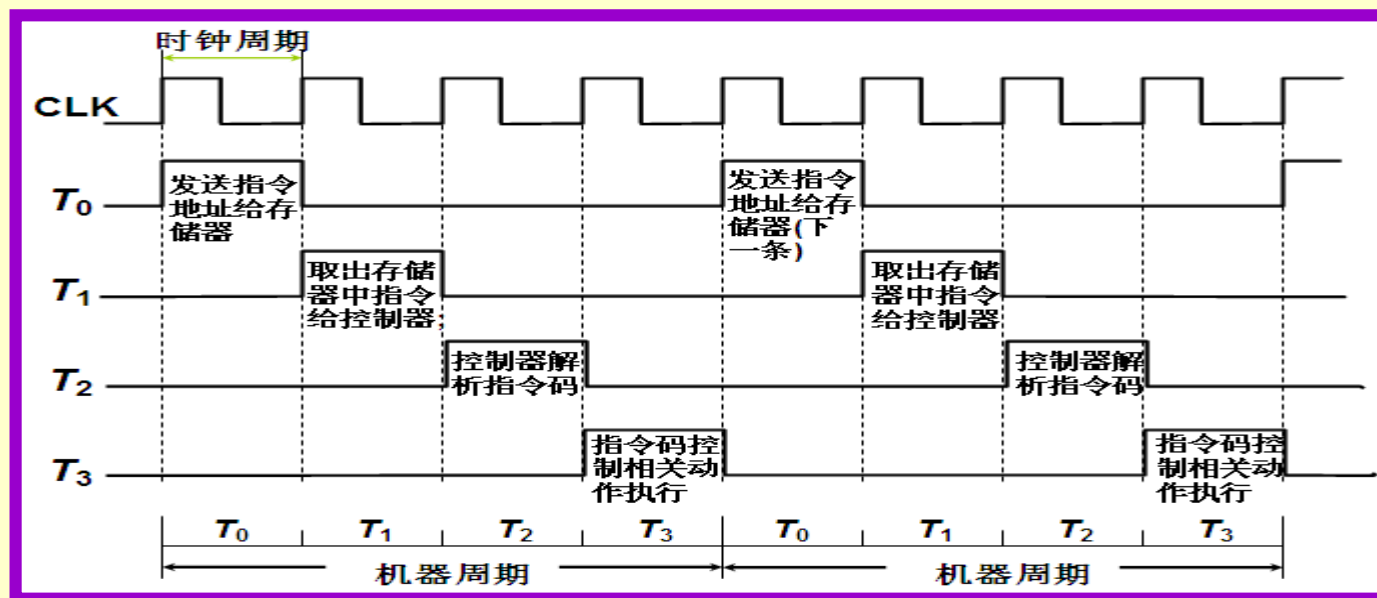
- 寄存器
- 算术逻辑部件
- 程序计数器PC
- 指令寄存器
- 信号控制器
- 时钟与信号发生器
- 存储单元地址
- 存储单元内容





指令执行

- ◆不同的指令，由一组不同的电信号构成
- ◆同一指令的电信号在时钟与节拍的控制下按次序产生与传输
- ◆一条指令占用一个或多个机器周期，一个机器周期又分为多个节拍
- ◆最小的时间区隔单位--时钟周期



时钟周期、
节拍与机
器周期

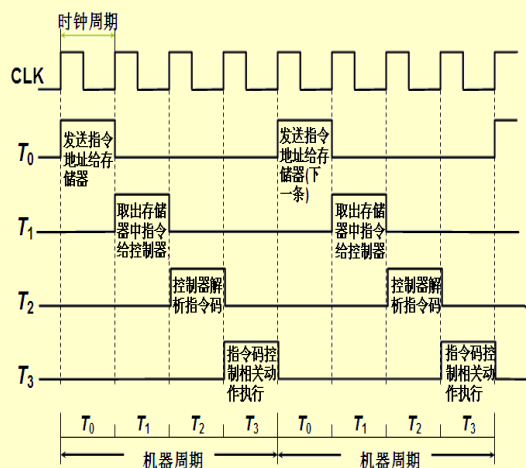
问：机器的“主频”指的是什么？



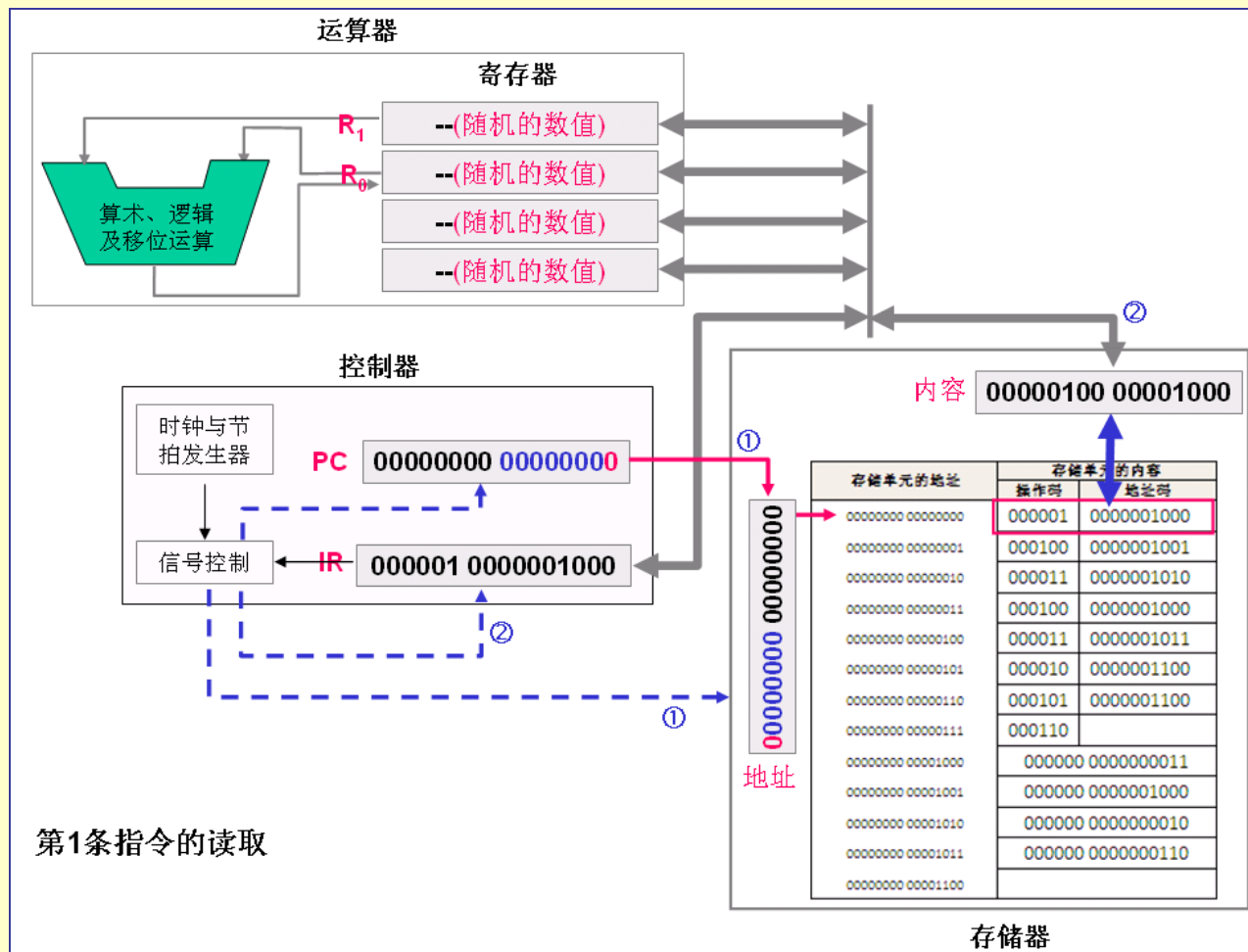
机器级程序的执行过程模拟

----程序的硬件执行过程

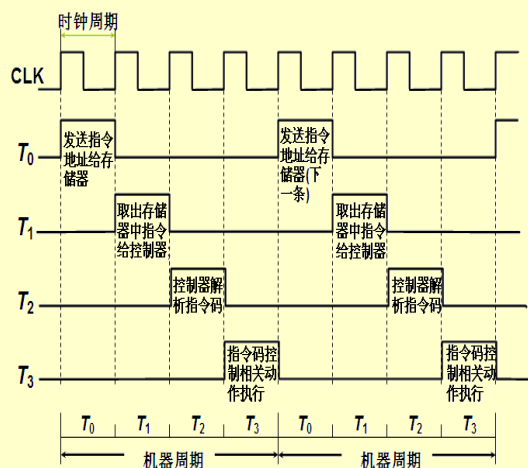
一条指令的执行



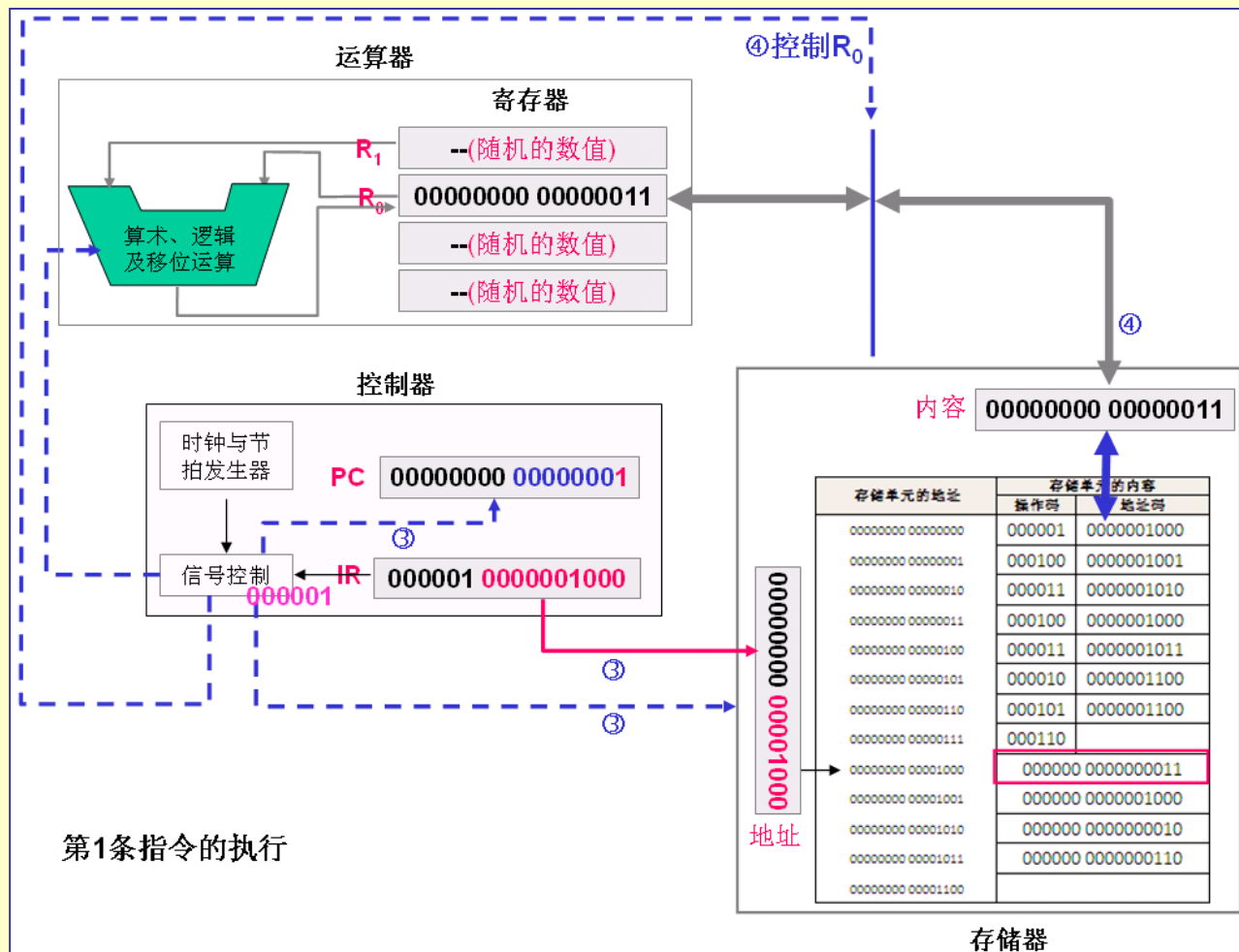
第1条指令在一个机器周期内完成。其中读取①②和执行③④分别在 T_0 、 T_1 和 T_2 、 T_3 节拍内完成。(示意)



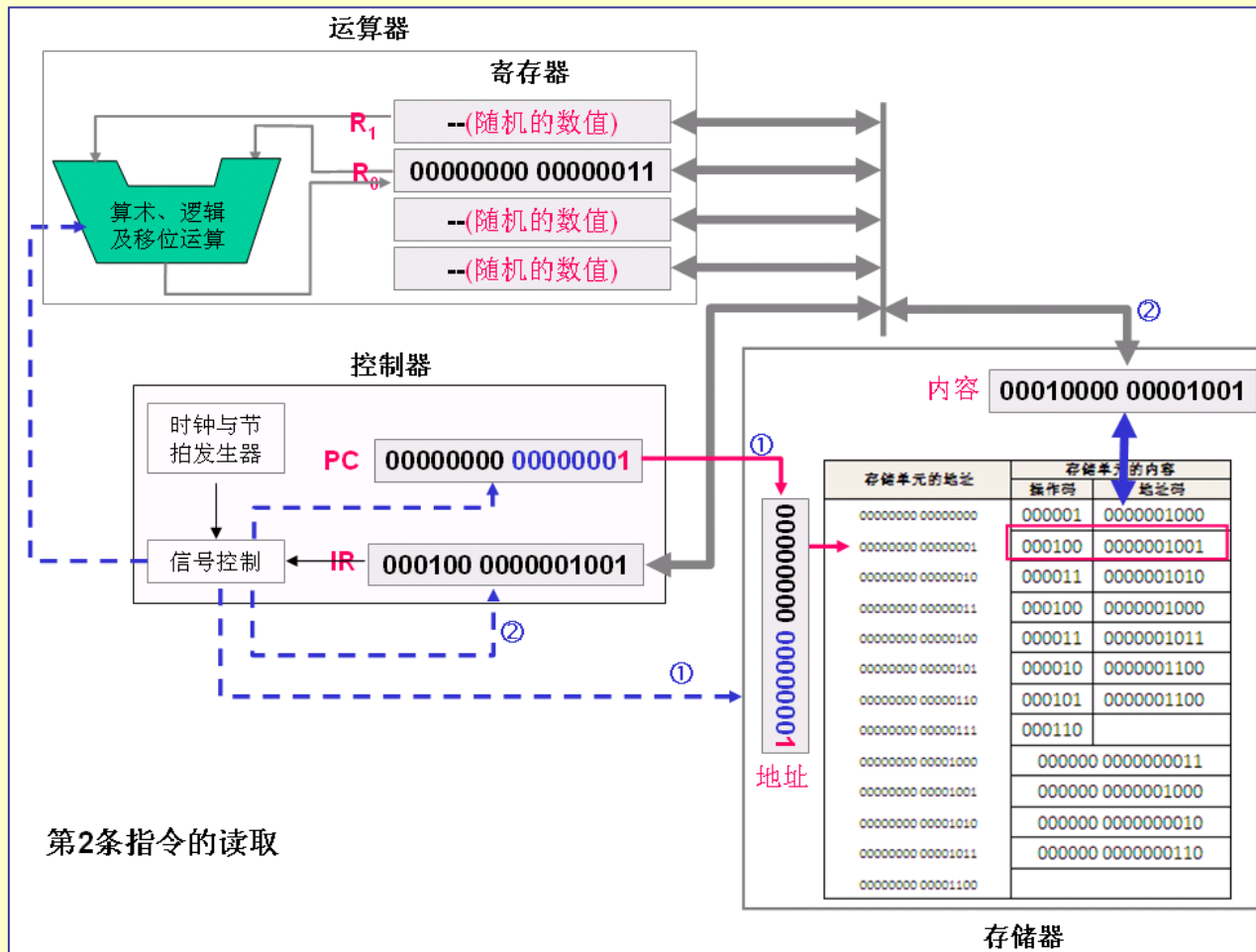
一条指令的执行



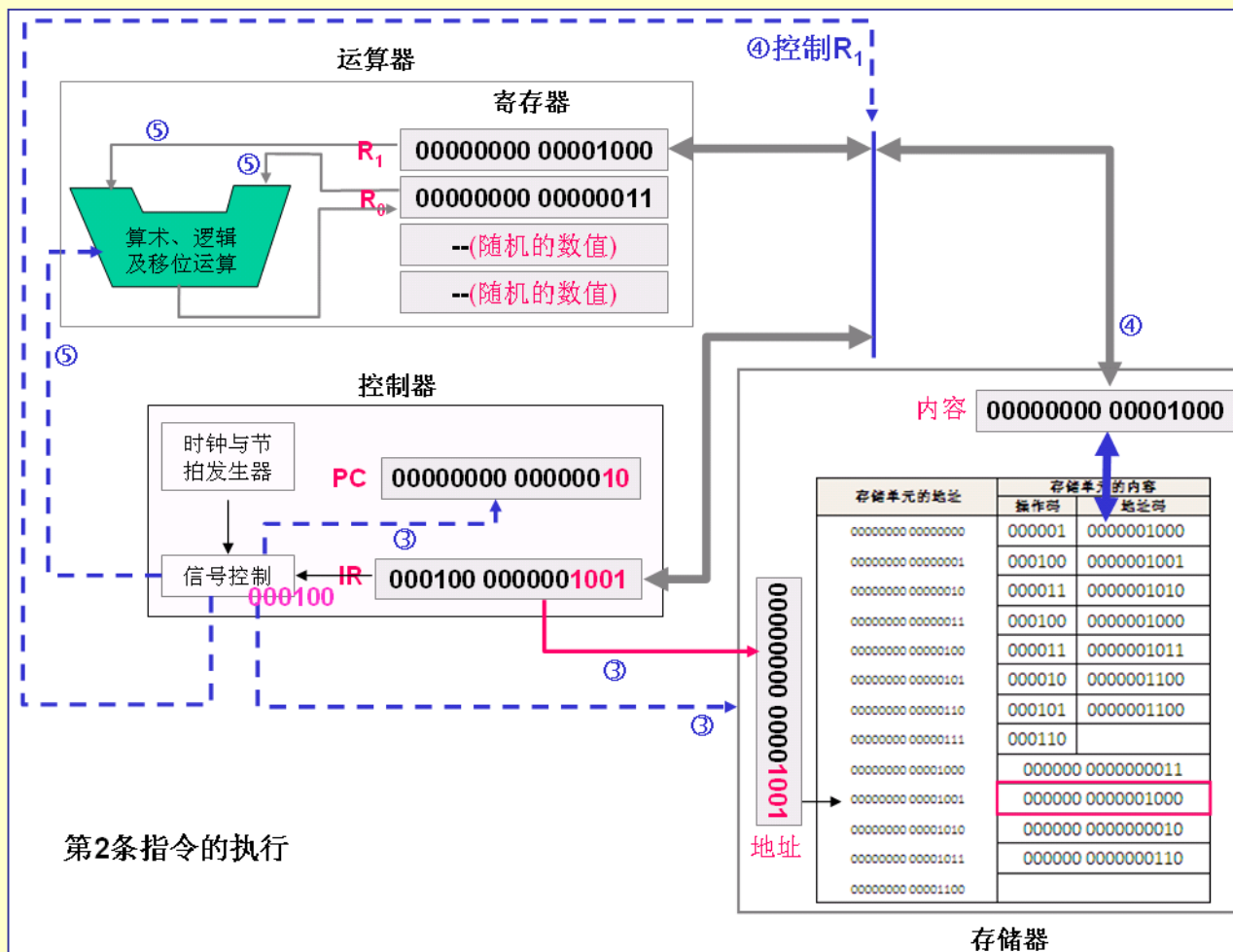
第1条指令在一个机器周期内完成。其中读取①②和执行③④分别在 T_0 、 T_1 和 T_2 、 T_3 节拍内完成。(示意)



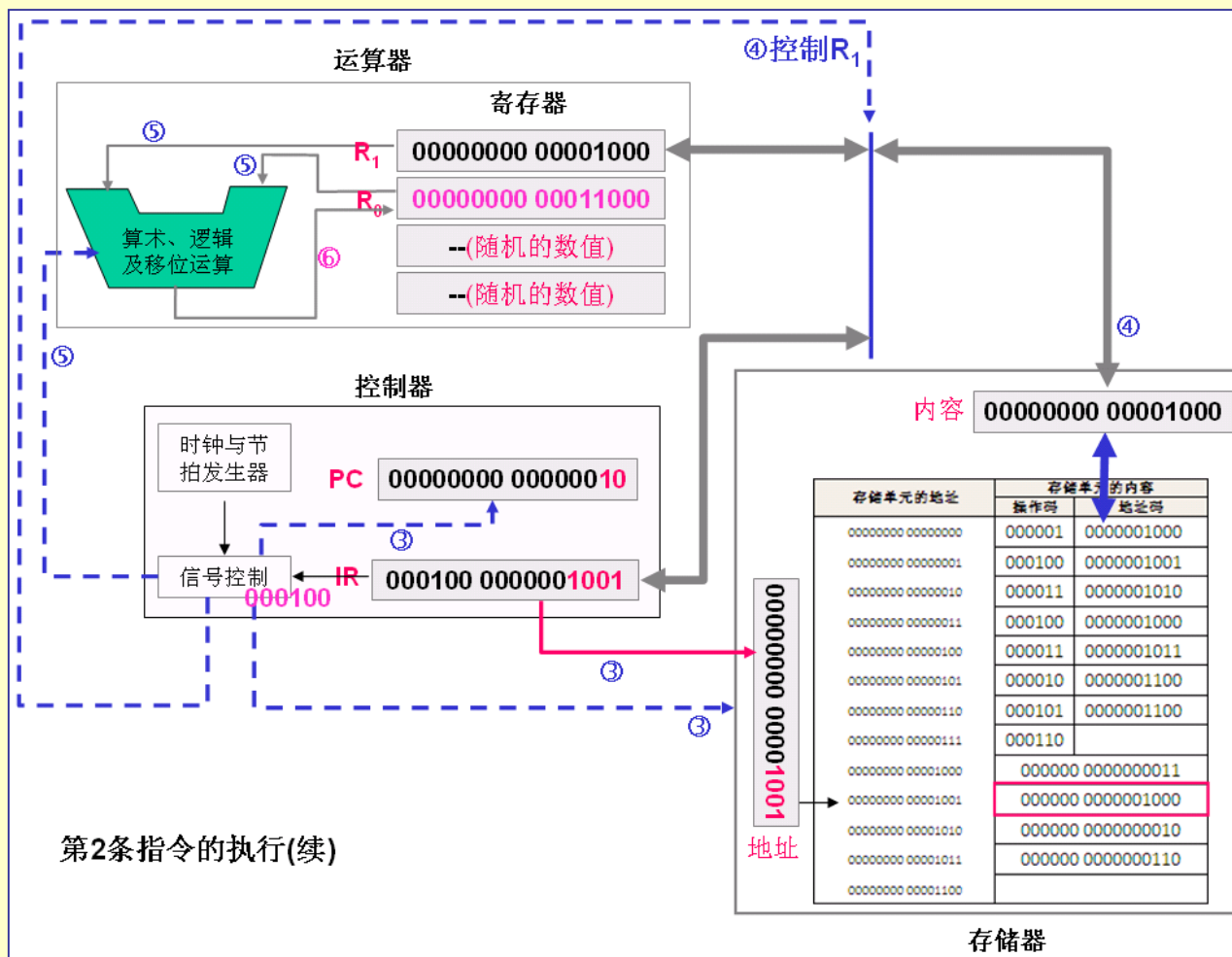
程序中指令的逐条执行



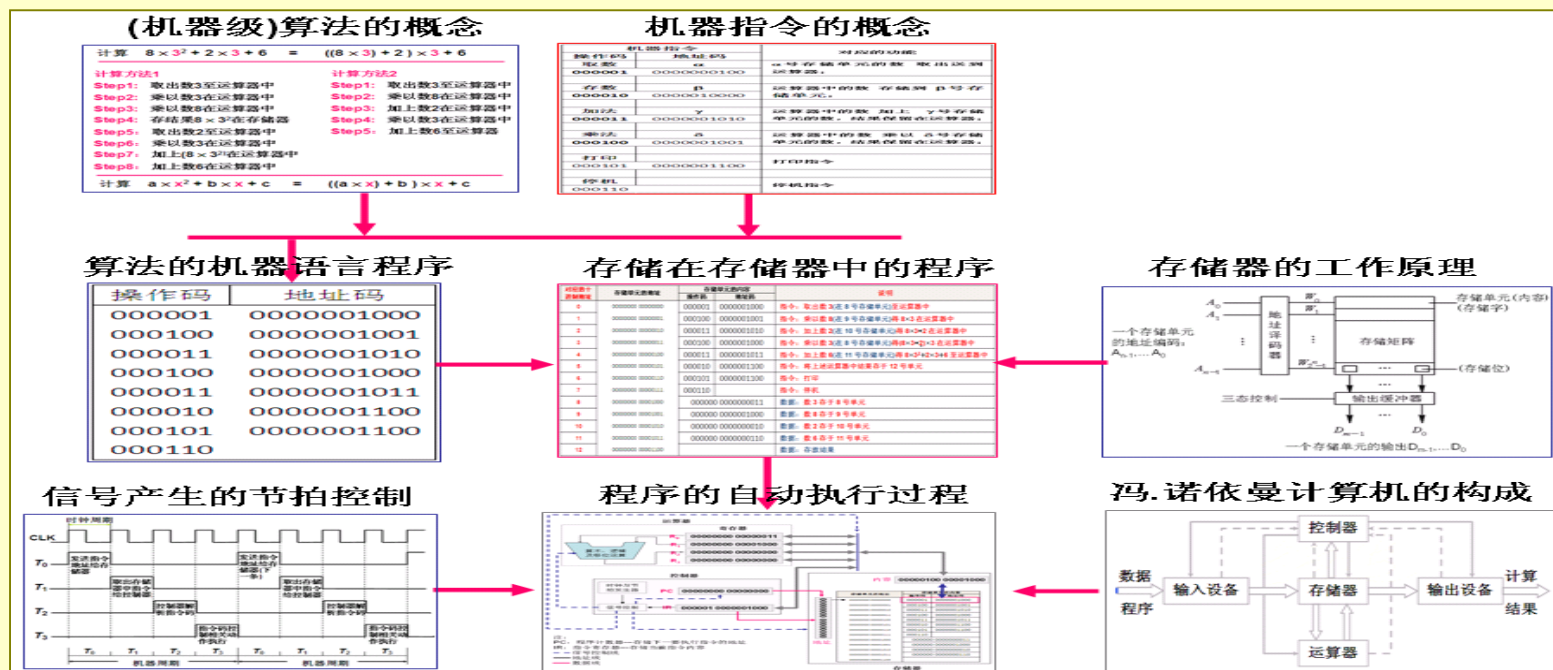
程序中指令的逐条执行



程序中指令的逐条执行



基本目标: 理解程序及其硬件实现思维



基本思维: 机器级算法与程序 → 机器指令与指令系统 → 存储器 → 存储程序 → 运算器与控制器 → 机器级程序的执行; 算法程序化 → 程序指令化 → 指令存储化 → 执行信号化

第3讲 冯.诺依曼计算机： 机器级程序及其执行

Questions & Discussion?