



计算机系统概述

CS359 计算机系统结构 邓倩妮



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

目标与要求



- **教学目标：** 概要了解计算机系统的全貌，掌握如何评价计算机系统的性能
- **学习要求：**
 - 能概述冯诺依曼结构计算机的特点，以及计算机硬件的基本组成和各部件的功能；
 - 能描述“存储程序”工作方式的主要特点，以及程序在计算机上的执行过程；
 - 能罗列计算机性能评价的基本指标和性能参数，以及各性能参数之间的关系；
 - 能正确使用阿姆达尔定律（Amdahl's law）评估计算机系统设计时的性能改进程度，并了解阿姆达尔定律的适用范围和扩展方向。
- **参考教材章节**
 - Chapter 1, David A. Patterson, John L.Hennessy. Computer Organization & Design: A Hardware/Software Interface, 4th edition. 计算机组成与设计：硬件/软件接口（第4版），机械工业出版社。

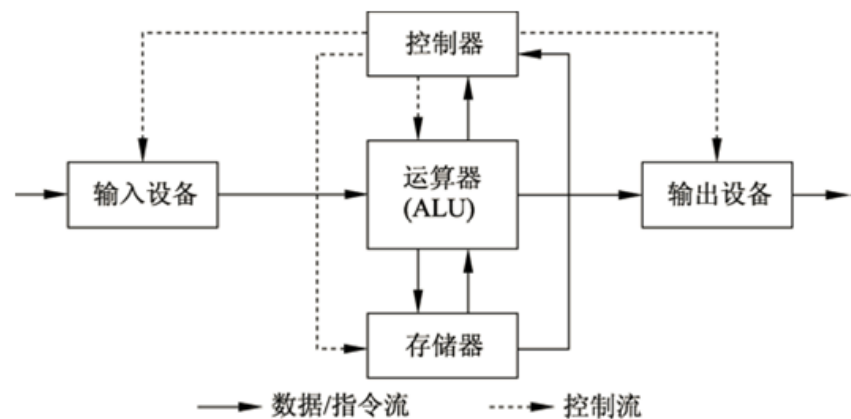
1.1 计算机的基本组成





冯·诺依曼结构

- 计算机由**五大部件**组成
- **存储程序**：把程序本身当作数据来对待，程序和该程序处理的数据用同样的方式储存，用**二进制**表示
- 指令和数据以同等地位存于**存储器**，按**址寻访**
- 按照**程序顺序执行**：计算机在工作时自动地从存储器中取出指令加以执行
- **以运算器为中心**：**I/O设备与存储器间的数据传送都要经过运算器**





冯·诺依曼结构-以**运算器**为中心

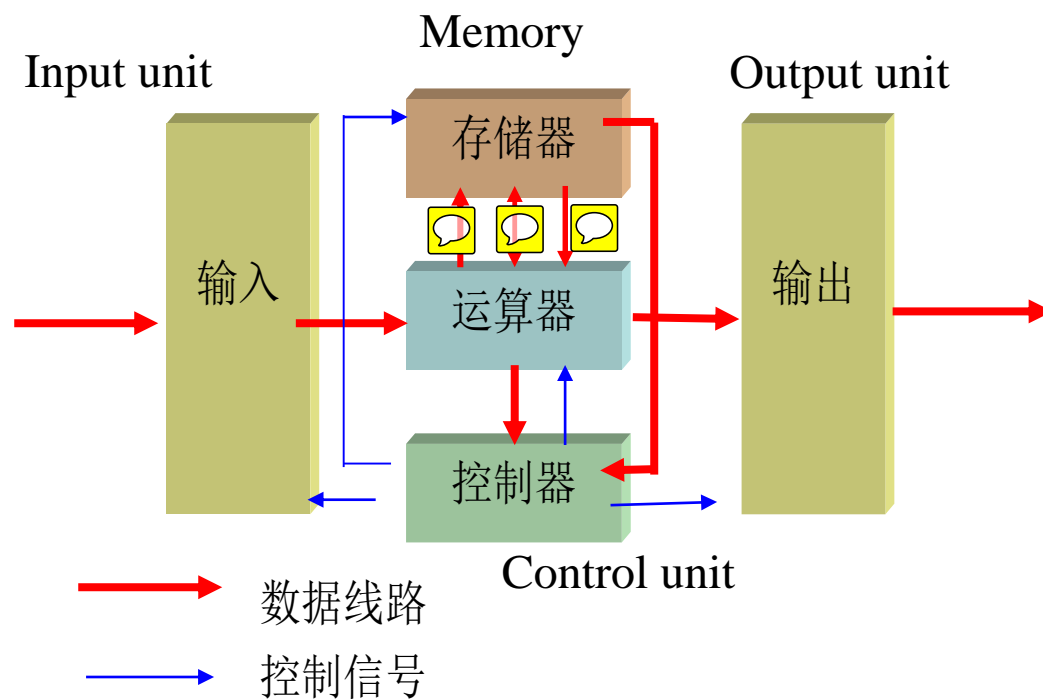


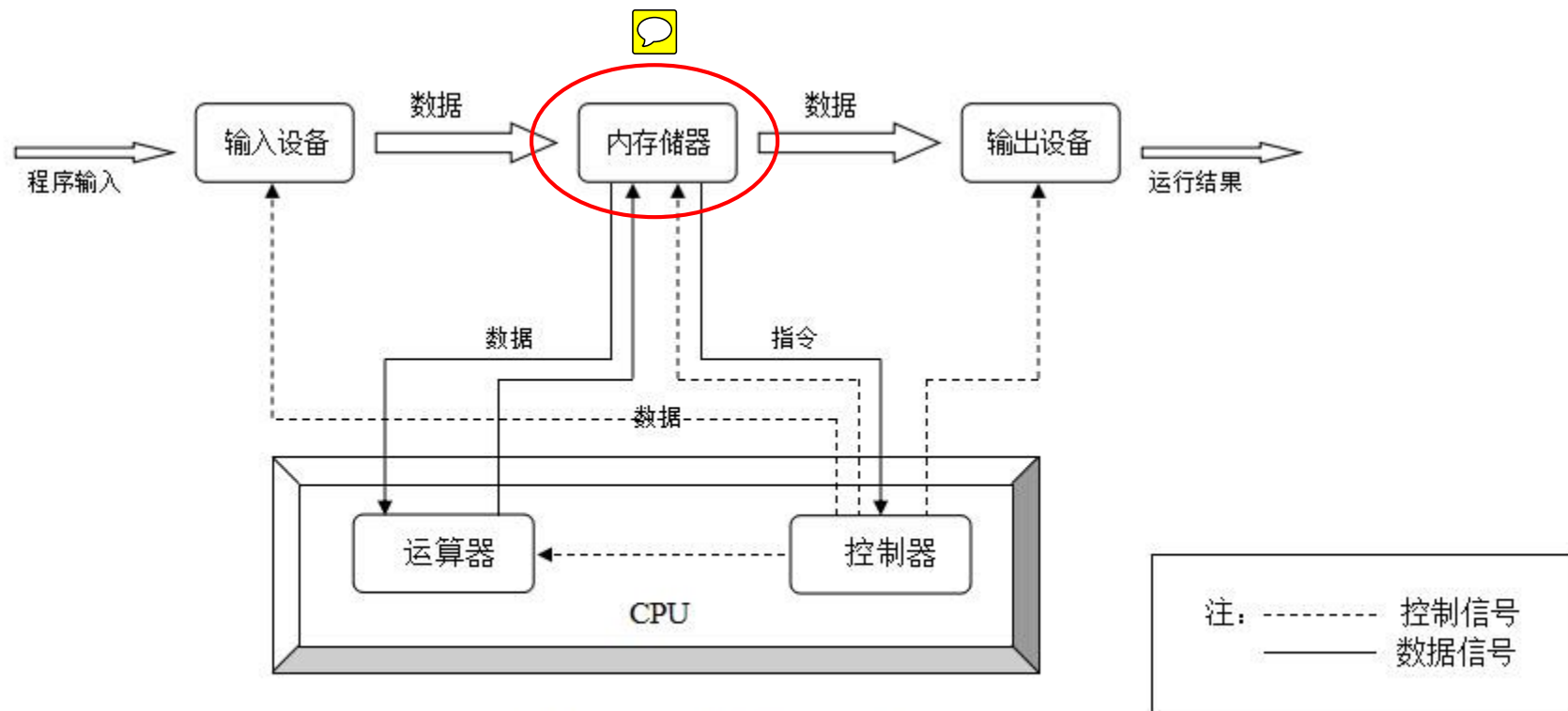
图 1-1 计算机的基本结构



约翰·冯·诺依曼
1903~1957



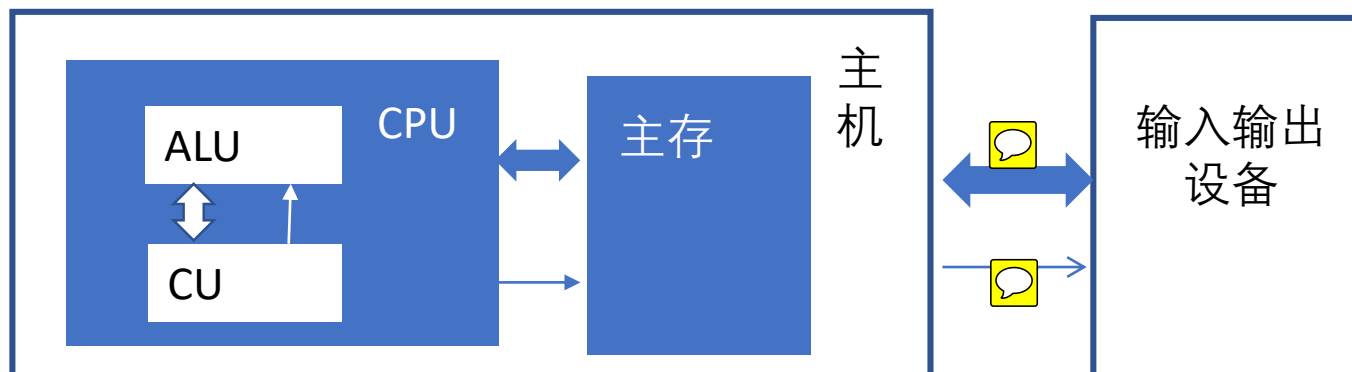
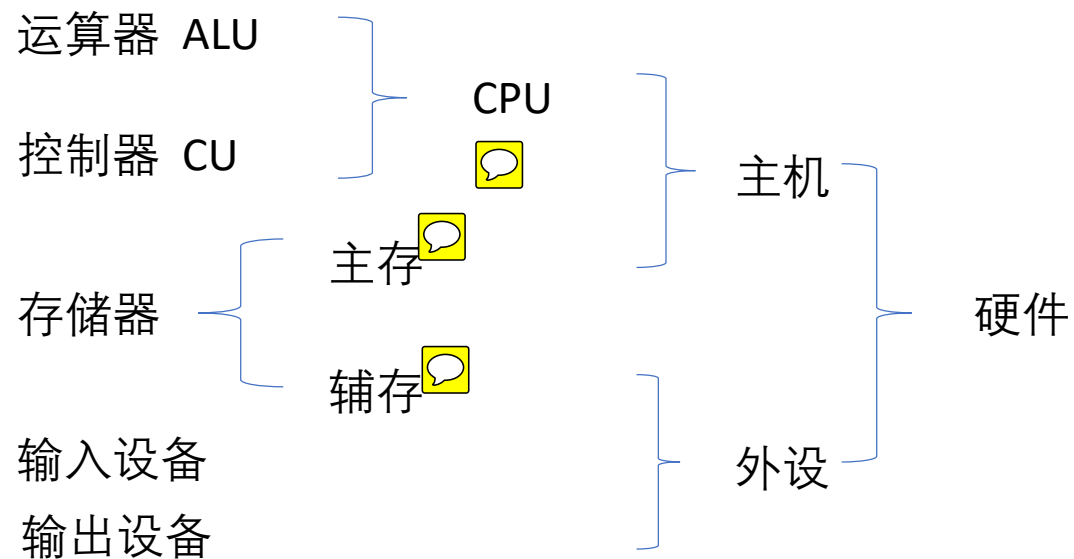
以存储为中心的计算机硬件结构



计算机的工作原理示意图



层次化的计算机硬件系统





小结

- 存储程序式计算机
- 以运算器为中心 -> 以存储器为中心
- 层次化的硬件结构
- 计算机系统结构的发展和变化

1.2 计算机的执行过程



计算机是怎样执行程序的呢？



$5 \times 30 = ?$



```
int z;  
int x=5;  
int y=30;  
z=x*y;  
cout<<"5×30="<<z;
```

高级语言程序

```
swap( int v[ ], int k)
{ int temp;
  temp=v[k];
  v[k]=v[k+1];
  v[k+1]=temp;
}
```

机器语言程序

```
00A20084
00821020
8DE20000
8E020004
AE020000
ADE20004
03E00008
```

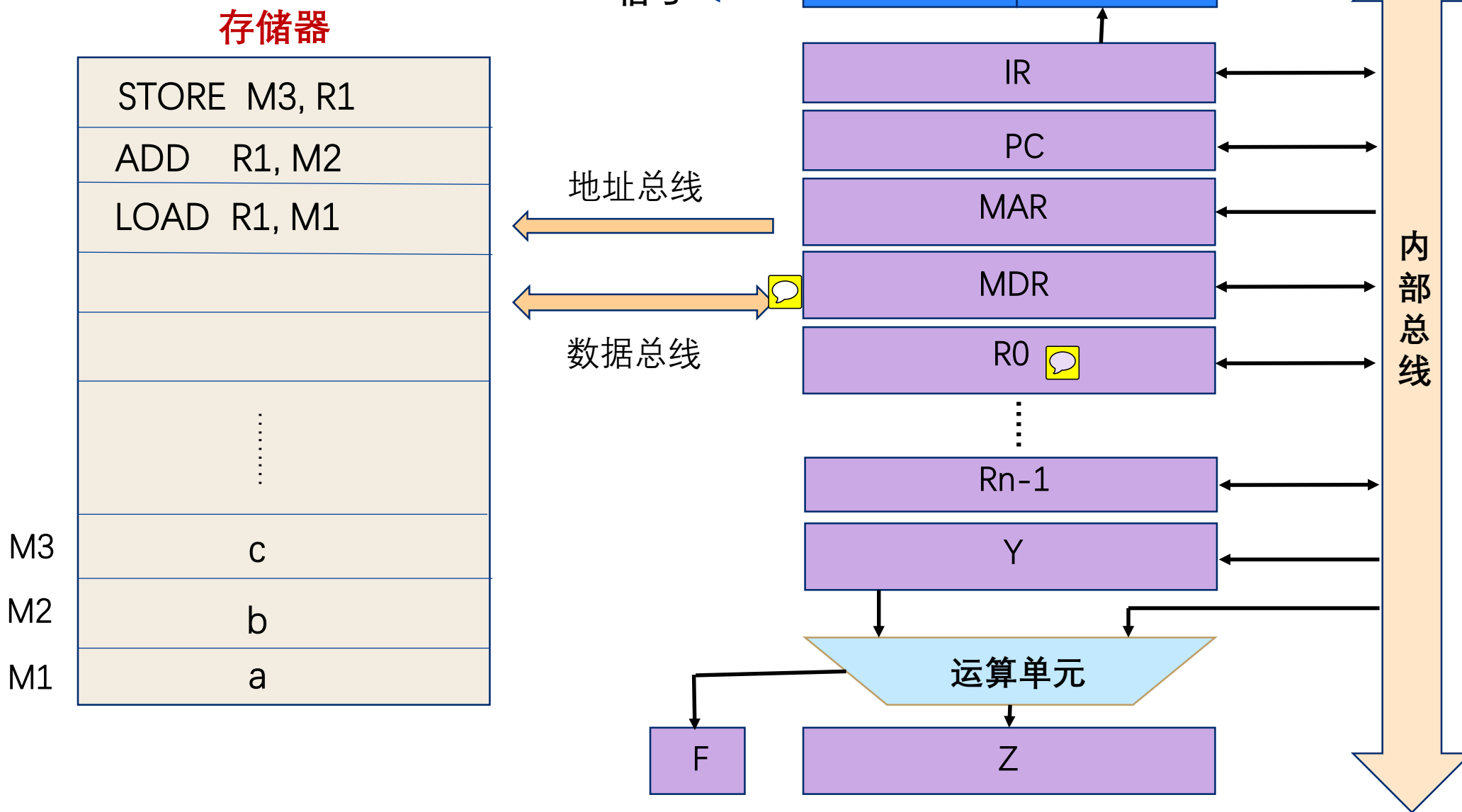
汇编语言程序

```
swap:
    sll    $2, $5, 2
    add    $2, $4, $2
    lw     $15, 0($2)
    lw     $16, 4($2)
    sw     $16, 0($2)
    sw     $15, 4($2)
    jr     $31
```

编译器

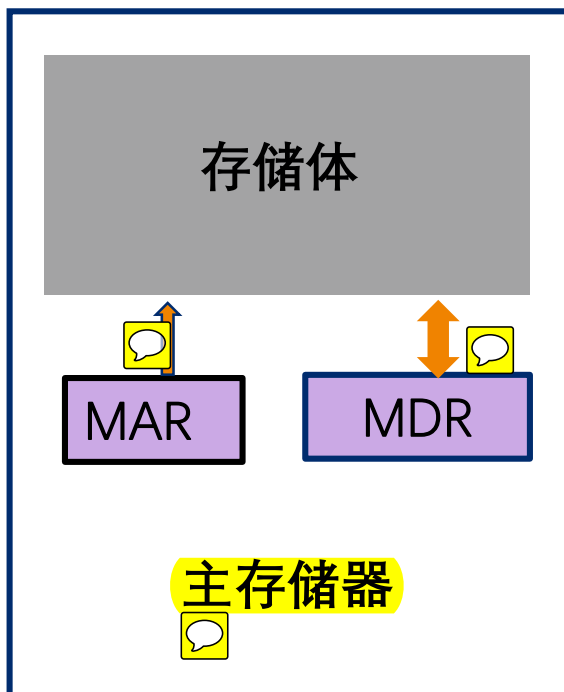
汇编器

计算机的内部结构





存储器专用寄存器



MAR (Memory Address Register)

存储器地址寄存器



MDR (Memory Data Register)

存储器数据寄存器

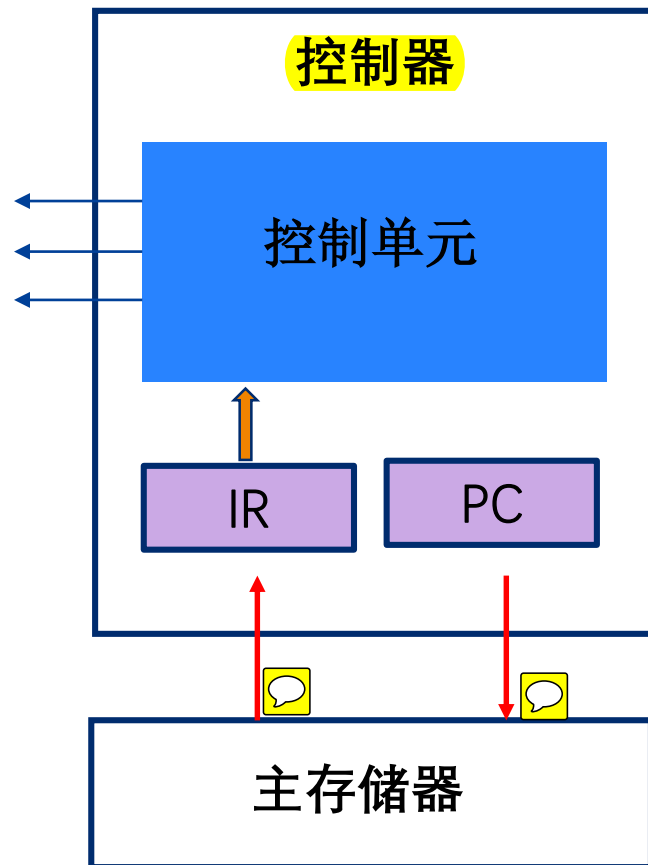


存储字长:

每次访问存储器获得的数据的位数



控制器专用寄存器



PC (Program Counter)

程序计数器:

存放当前即将执行的指令的地址

IR (Instruction Register)

指令寄存器:

存放当前即将执行的指令



指令执行过程举例

汇编语言程序	功能
LOAD R1, M1	将存储器M1的内容读入寄存器R1
ADD R1, M2	将存储器M2的内容和寄存器R1的内容相加， 运算结果保留在R1中
STORE M3, R1	将R1的内容存入M3中



指令执行过程举例

汇编语言程序	机器语言程序
LOAD R1, M1	0000 0001 00000010
ADD R1, M2	0001 0001 00000100
STORE M3, R1	0010 0001 00000110

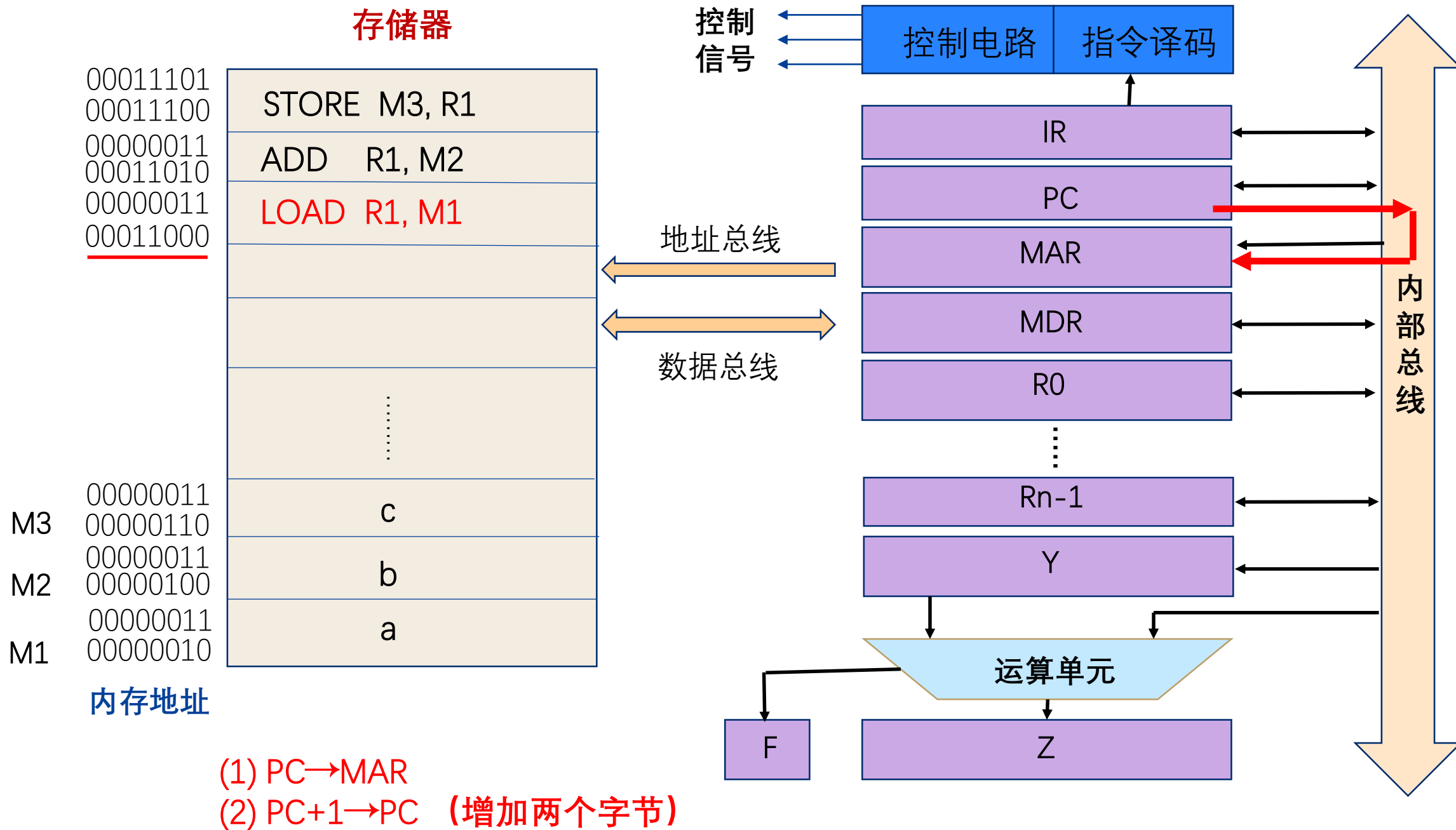


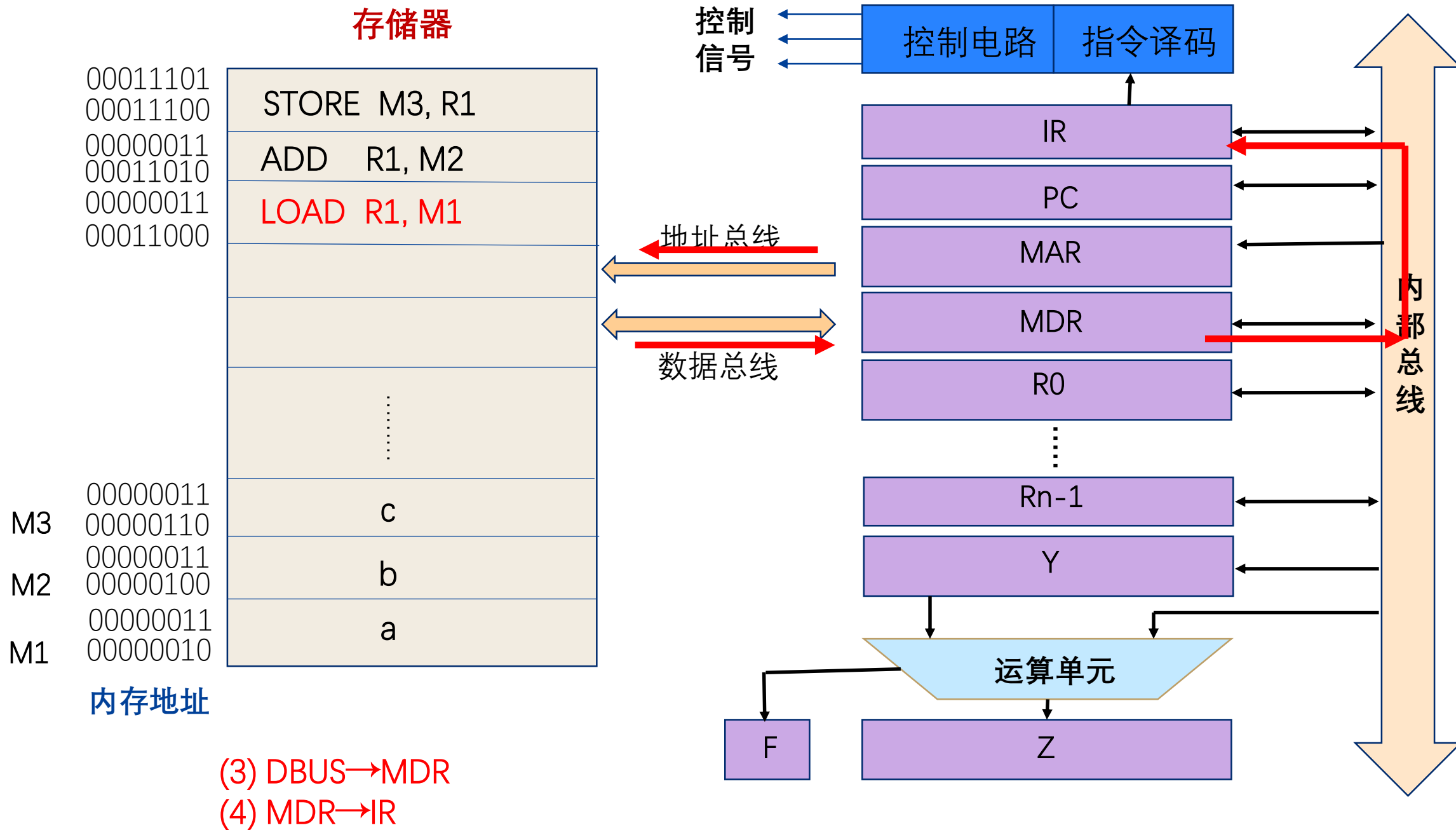
汇编语言程序	机器语言程序
LOAD R1, M1	0000 0001 00000010
ADD R1, M2	0001 0001 00000100
STORE M3, R1	0010 0001 00000110

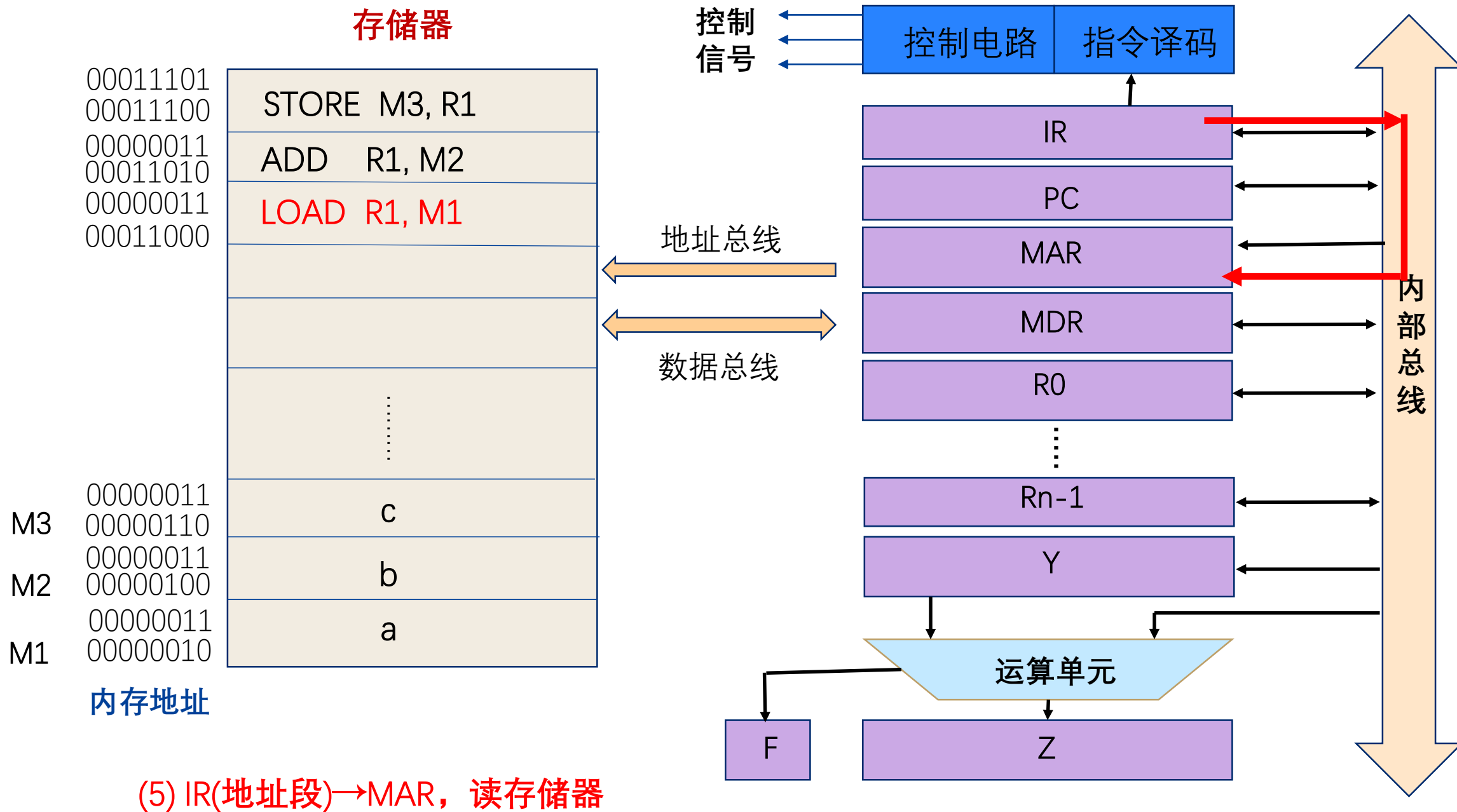
存储器

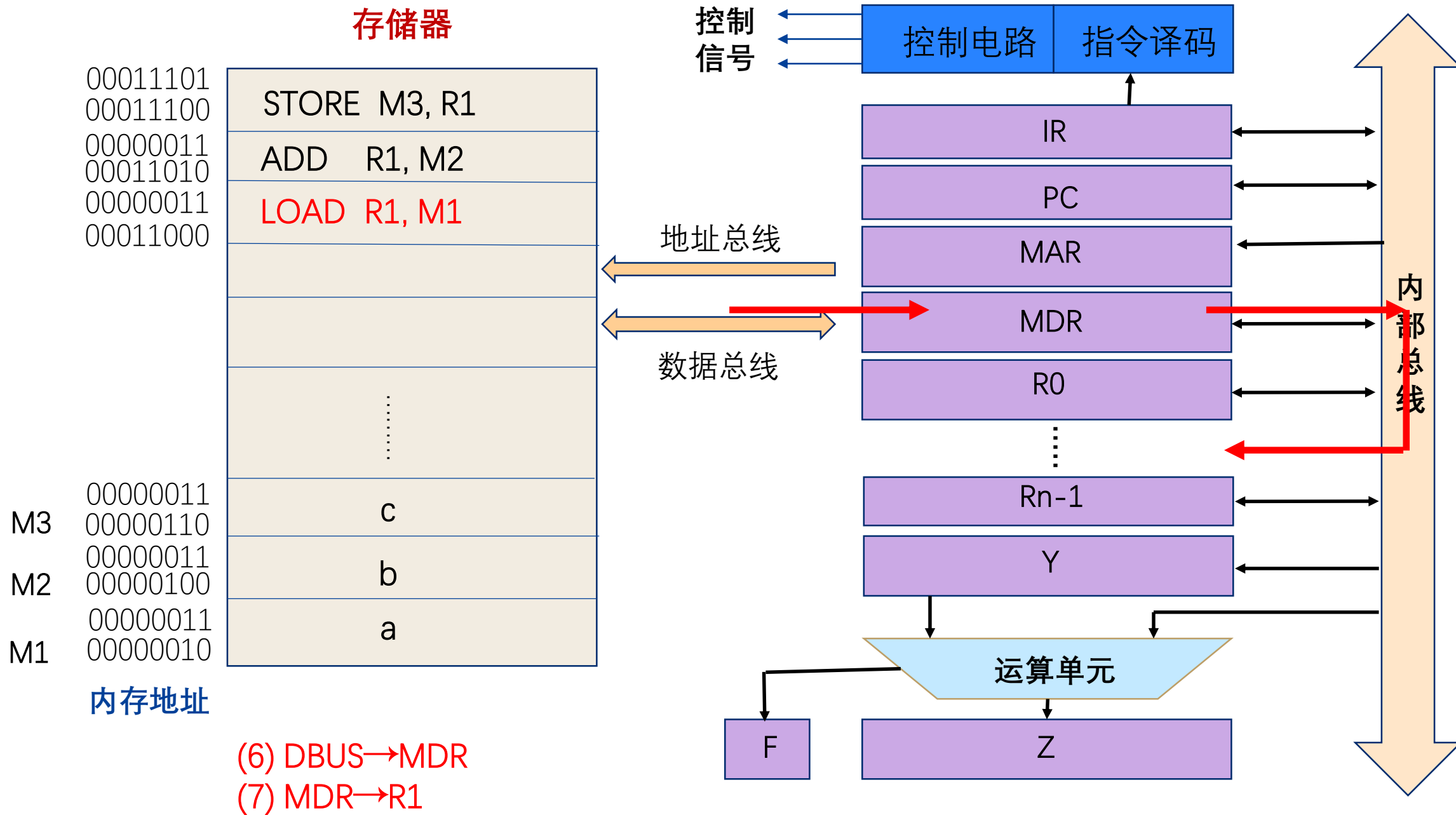
00011101	STORE M3, R1	
00011100		
00000011	ADD R1, M2	
00011010		
00000011	LOAD R1, M1	
00011000		
	⋮	
00000011	c	M3
00000110		
00000011	b	M2
00000100		
00000011	a	M1
00000010		

内存地址











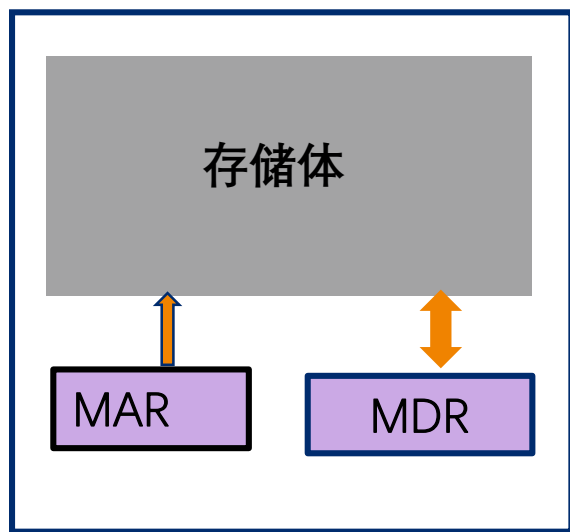
指令执行的步骤

- step1. 取指令
- step2. 解码指令：
 - 将指令解码成一系列的控制信号
- step3. 执行指令：
 - 将控制信号发送给相关部件，执行相应的运算
- 重复step1-step3

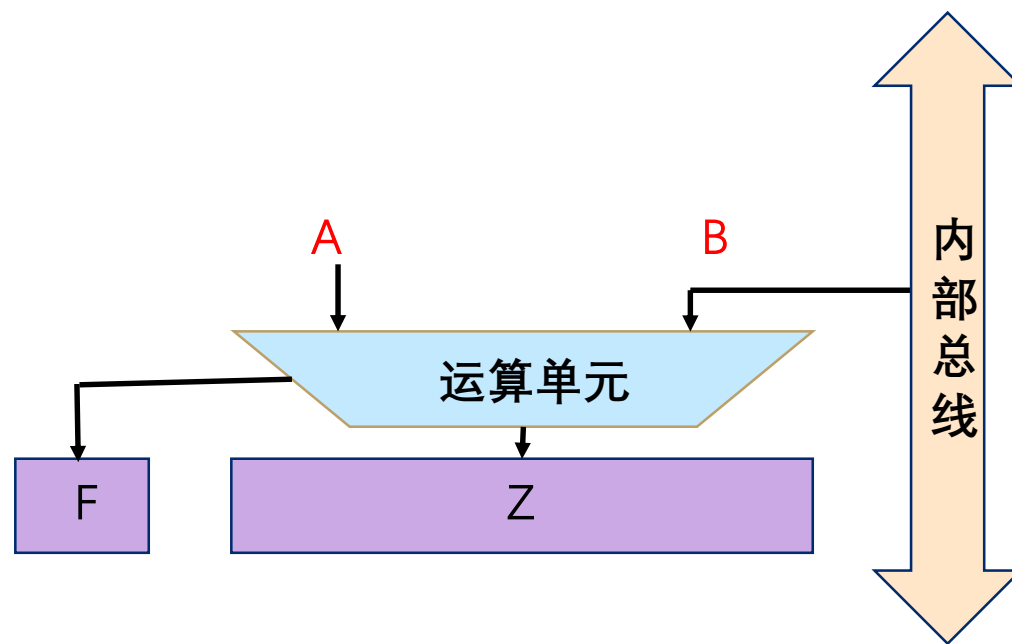


术语

- 指令字长： 指令的长度
- 存储字长： 每次访问存储器获得的数据的位数， MDR宽度
- 机器字长： 处理器能同时处理的数据位数




MDR 的宽度： 存储字长



A 和 B 的宽度： 机器字长



小结

- 机器指令的执行过程
- 一些术语：
 - 寄存器：中央处理器内的组成部分 
 - 字节：8位二进制代码
 - 存储单元：具有特定存储地址的存储单位
 - 存储容量：存储器中可存二进制代码的总量，单位：KB，MB，GB
 - 指令字长、存储字长、机器字长

1.3 计算机的性能指标





性能与需求相关



■ 飞机的性能指标

- 巡航速度
- 最大航程
- 载客容量

使用者的需求角度不同，性能的优劣对比就不同

计算机的性能— 两个常用的指标

1. 执行时间 (Execution Time)

2. 吞吐量 (Throughput)

执行时间



也被称为**响应时间 (response time)**

计算机完成某任务的总时间，包括硬盘访问、内存访问、CPU执行时间、操作系统开销、和输入/输出活动等所有从开始执行到执行结束的总时间。

又被称为：**墙上时钟时间**

性能 = 1 / 执行时间

性能加速比

$$\text{加速比} = \frac{\text{性能}_X}{\text{性能}_Y} = \frac{\frac{1}{\text{性能}_Y}}{\frac{1}{\text{性能}_X}} = \frac{\text{执行时间}_Y}{\text{执行时间}_X}$$

- 系统X用10秒钟执行某个程序，系统Y执行同一个程序花费15秒
 - 系统X 比 系统Y 快 1.5 倍、
 - 系统X 对 系统Y 的加速比 (speedup) 是 1.5
 - X比Y性能提升了 50%

吞吐量 (Throughput)

- 又称：带宽 (bandwidth)
- 单位时间内完成的任务数量
- 例如：
 - 多媒体应用
 - 吞吐率高：音频/视频播放流畅
 - Web 服务器：
 - 吞吐率高：在单位时间完成的任务越多越好（管理员角度）
 - 执行时间短：提交的任务越快完成越好（使用者角度）

执行时间与吞吐量

- 互相关联

- 例如：

- 使用更快的处理器

- 既能缩短一个程序的执行时间，又能提高系统整体的吞吐量

- 将一个处理器增加为多个处理器

- 只增加吞吐量，无法缩短单个任务的执行时间

执行时间 vs. CPU时间

- 一个程序在输入输出等待时，处理器会切换去执行另外一个程序，以提高系统的运行效率，但这样会延长单个程序的执行时间。
- CPU时间
 - 更合适用于衡量一个程序在硬件系统中的性能
- CPU时间
 - 给定程序任务占用处理器的时间。

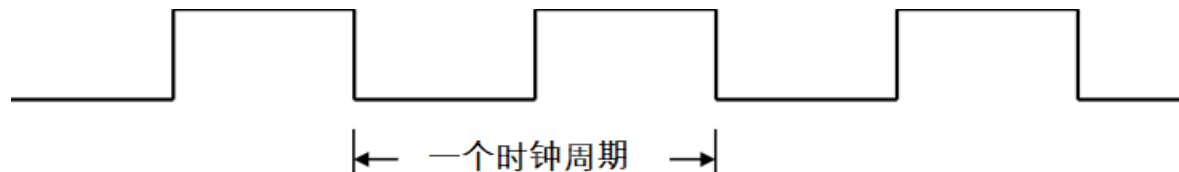
影响CPU时间的因素？

CPU执行时间 = 指令条数 (IC) × 平均每条指令的执行时间

平均每条指令的执行时间=

平均每条指令所花的时钟周期 (CPI) × 一个时钟周期长度 (CT)

- IC: instruction count
- CPI: Cycle Per Instruction
- CT: cycle time



平均CPI

$$\text{平均 CPI} = \sum_{i=1}^n (\text{CPI}_i \times \text{IC}_i)$$

- IC_i : 第*i*类指令在所有指令中所占的比例
- CPI_i : 第*i*类指令的平均CPI (clock cycles per instruction)
- n : 指令的类别数

CPU执行时间 — 用于性能比较

- 例如：
- 假定计算机M1和M2具有相同的指令集体系结构，主频分别为1.5GHz和1.2GHz。在M1和M2上运行某基准程序P，平均CPI分别为2和1，那么程序在M1和M2上运行时间的比值是多少？

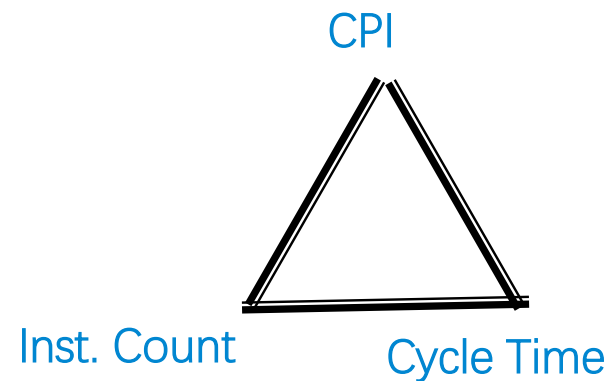
CPU执行时间 = 指令数目 \times CPI \times 一个时钟周期长度

$$2 * (1/1.5) : 1 * (1/1.2) = 1.6$$


CPU时间三要素

CPU执行时间 = 指令数目 \times CPI \times 一个时钟周期长度

- **影响CPU时间的三要素**
 - 计算机系统结构的性能优化方向
- 任何一个因子都不能独立地影响性能
- 处理器的性能优化，本质上就是在这三个要素之间折衷



CPU吞吐量 — 性能指标

- MIPS —— Million Instruction Per Second
 - 每秒执行百万条指令数 
- FLOPS —— Floating Point Operation Per Second
 - 每秒浮点运算次数
 - MFLOPS 、 GFLOPS、 TFLOPS

- 举例：某程序在两台计算机上的性能测量结果为：

测量内容	计算机A	计算机B
指令数	100亿条	80亿条
时钟频率	4GHz	4GHz
CPI	1.0	1.1

哪台计算机MIPS值更高？ A

$$\frac{1}{1} \times 4 > \frac{1}{1.1} \times 4$$

MIPS数 = 一周期完成的指令条数 × 时钟频率

哪台计算机更快？ B

$$\frac{1}{4} \times 100 \times 1 > \frac{1}{4} \times 80 \times 1.1$$

CPU执行时间 = 指令数目 × CPI × 一个时钟周期长度



小结



- 执行时间
- 吞吐量
- CPU时间
- MIPS

1.4 性能设计的基本原则





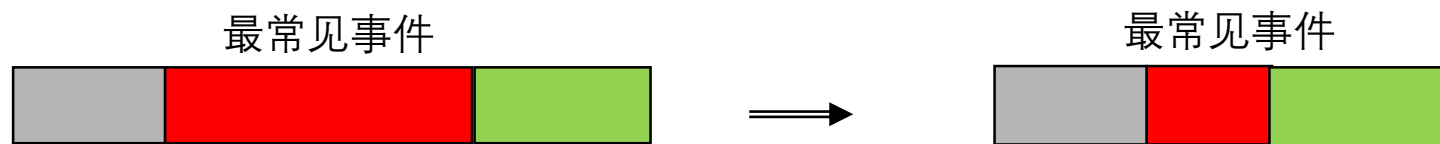
基本原则

1. 大概率事件优先
2. Amdahl定律

大概率事件 (common case) 优先 (first)

对于大概率事件（最常见的事件）赋予优先的处理权和资源使用权

$$\text{系统加速比} = \frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$



Amdahl 定律

加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中所占的重要性比例。

$$\text{系统总加速比} = \frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}} = \frac{(1 - \text{改进部分占比}) + \text{改进部分占比}}{(1 - \text{改进部分占比}) + \frac{\text{改进部分占比}}{\text{改进部分提升的加速比}}}$$

$$\Rightarrow \frac{(1 - \text{改进部分占比}) + \text{改进部分占比}}{(1 - \text{改进部分占比}) + \frac{\text{改进部分占比}}{\infty}} \Rightarrow \frac{1}{(1 - \text{改进部分占比})}$$

Amdahl 定律

如果只针对整个任务的一部分进行优化，那么所获得的加速比是有上限的

性能增加的递减规则：如果仅仅对计算机中的一部分做性能改进，改进越多，系统获得的效果越小

- 举例：
- 分析一个Web服务器的性能。假定该服务器，有50%的时间用于计算，另外50%的时间用于输入输出操作。采用新的处理器，Web服务器上的程序运行速度可以提升10倍，系统整体获得的加速比是多少？
- 系统获得的性能加速比： $1/(0.5+0.5/10) = 1/0.55 = 1.818$
- 采用更快的处理器，系统性能加速比不超过2
- 优化方向：减少输入输出操作所占的时间

性能设计基本原则：小结

- 大概率事件优先原则：首先优化在系统中最重要部分。
- Amdahl 定律：如果只针对整个系统的某一部分进行优化，所获得的加速比有上限
- 一个“好”的计算机系统：是一个吞吐量平衡的系统

Amdahl 定律分析并行计算的性能

举例：

- 假设执行一个程序的总时间为1；程序的不可并行化部分占40%，就是0.4；可并行化部分就是 $1 - 0.4 = 0.6$ ；
- 处理器个数：2，加速比： $1/(0.4 + 0.6/2) \approx 1.43$
- 处理器个数：5，加速比： $1/(0.4 + 0.6/5) \approx 1.92$
- 加速比上限： $1/0.4 = 2.5$
- 结论：并行计算性能的提升，受到了程序中必须串行执行部分的限制。

Amdahl 定律分析并行计算的性能

举例:

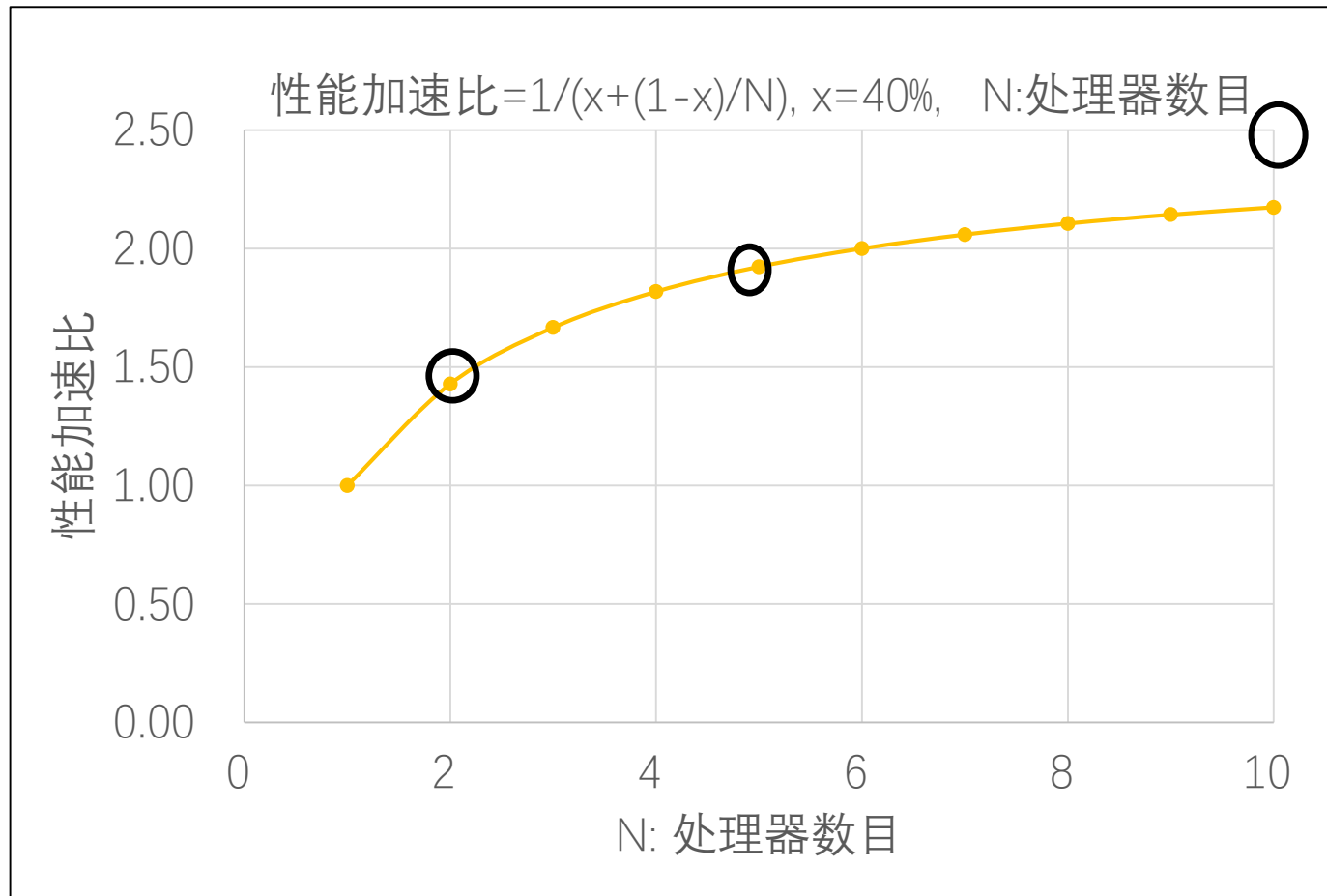
- 总时间为1;
- 串行部分: 40%;
- 可并行化部分: 0.6;

处理器个数: 2,
加速比: 1.43

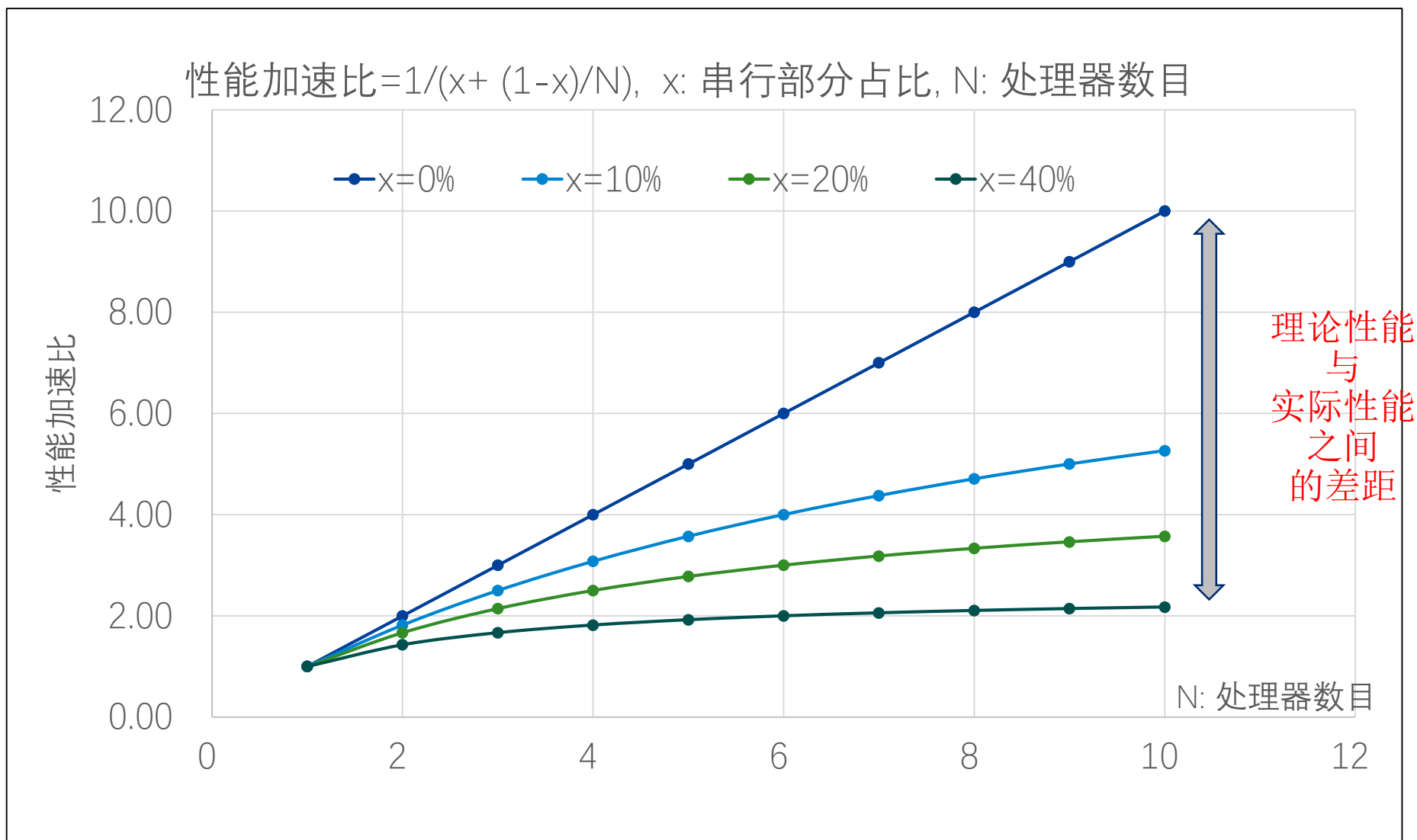
处理器个数: 5,
加速比: 1.92

加速比上限: 2.5

并行计算性能, 受到程
序中串行执行部分的限
制。



- 矛盾：某些应用的性能确实大幅提升了，为什么？



某些应用的性能确实大幅提升，为什么？



- 原因：经典的Amdahl 定律并不适用于规模可扩展的并行应用程序的性能分析。



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



谢谢观看

上海交通大学