计算机组织与体系结构

第八讲

计算机科学与技术学院

舒燕君

Recap

- 指令系统的分类
  - ✓在CPU中操作数的存储方法；
- 指令系统的设计和优化
  - ✓ 指令系统设计的基本原则
  - ✓ 控制指令
  - ✓ 指令操作码的优化
- 指令系统的发展和改进
  - ✓CISC
  - ✓RISC

4.7 指令格式举例

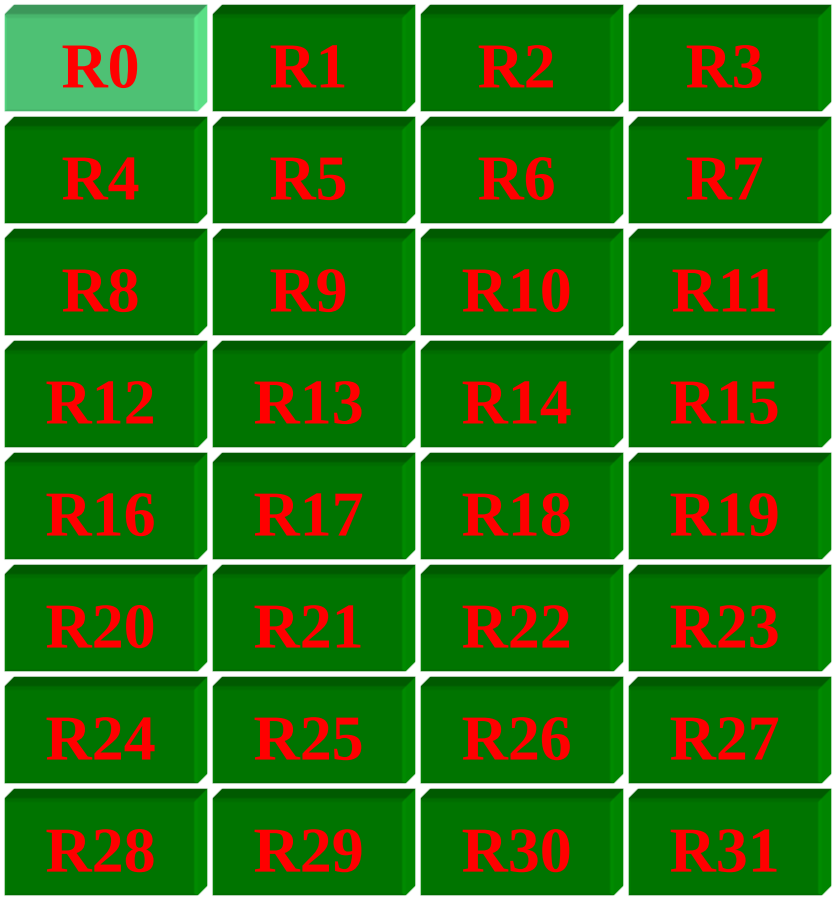
介绍MIPS32的一个子集，简称为MIPS。

- Load/Store型指令集结构
- MIPS是一种多元指令集结构
  - 体现了当今多种机器（AMD29K、DEC station 3100、HP850、IBM 801、MIPS M/120A、MIPS M/1000、Motorola 88k、RISC I、SGI4D/60、SPARC station 1、Sun 4/110、Sun 4/260等）的指令集结构的共同特点。
  - 还将会体现未来一些机器的指令集结构的特点。

MIPS指令集结构

- 具有一个简单的Load/Store指令集；
- 注重指令流水效率；
- 简化指令的译码；
- 高效支持编译器。

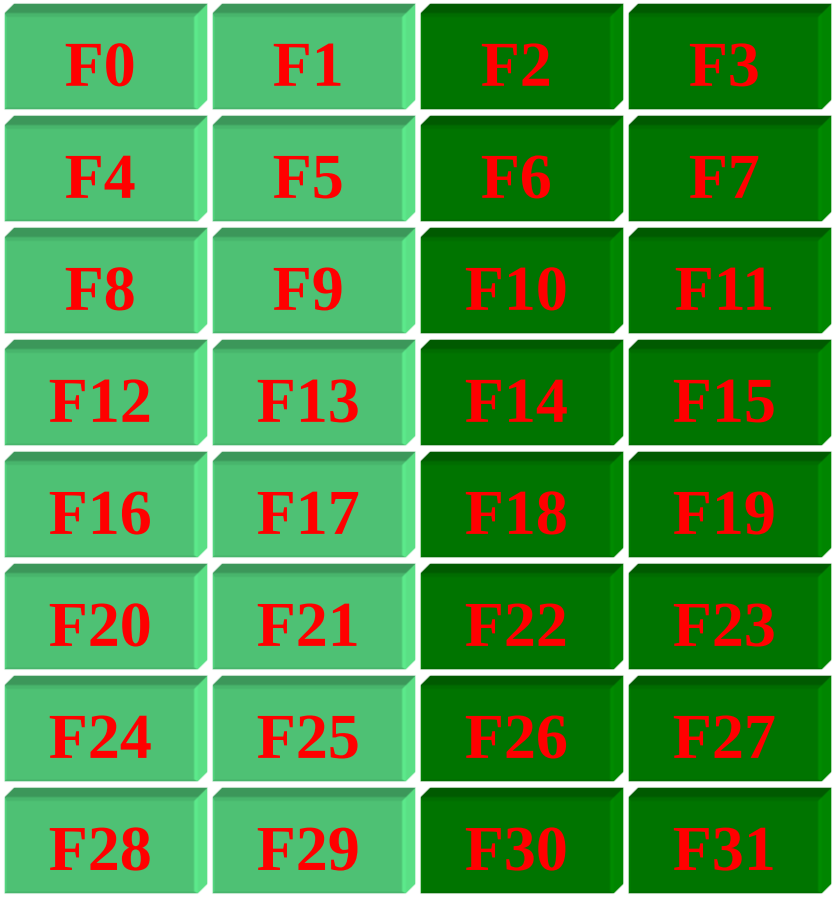
MIPS指令集结构：寄存器



32个32位的通用寄存器（GPRs）。

寄存器R0的内容恒为全0。

MIPS指令集结构：寄存器



32个32位浮点寄存器（FPRs）。

单精度浮点数表示和双精度浮点数表示。

MIPS指令集结构：数据类型

- 整型数据：
  - 8位、16位、32位。
- 浮点数据：
  - 32位单精度浮点；
  - 64位双精度浮点；
  - IEEE 754标准。

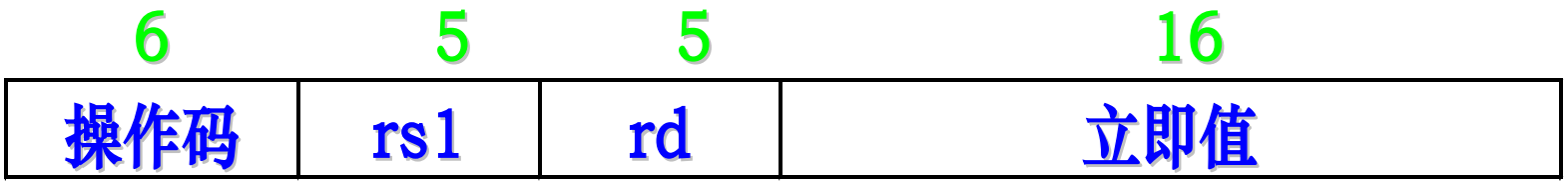
00000000	00000000	00000000	01100100
11111111	11111111	10011000	01100100

MIPS指令集结构：寻址方式

- 寄存器寻址；如 **ADD R1, R2, R3**
- 立即值寻址；如 **ADD R1, R2, #42**
- 偏移寻址； 如 **LW R2, 40(R3)**
  - 寄存器间接寻址； 如 **LW R2, 0(R3)**
  - 直接寻址；如 **LW R2, 40(R0)**

MIPS指令集结构：指令格式

I 类型指令

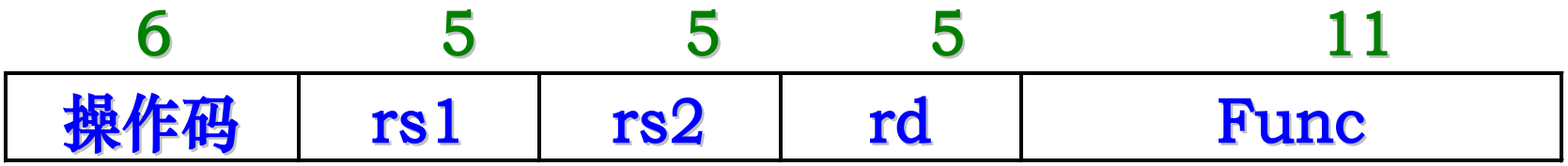


字节、半字、字的载入和存储；

rd ←rs1　op　立即值。

MIPS指令集结构：指令格式

R 类型指令



寄存器－寄存器 ALU 操作：rd←rs1 func rs2;
函数对数据的操作进行编码：加、减、…;
对特殊寄存器的读/写和移动。

MIPS指令集结构：指令格式

J 类型指令

跳转，跳转并链接，从异常（exception）处自陷和返回。

MPIS指令集结构：操作类型

- **Load和Store操作；**
- **ALU操作；**
- **分支和跳转操作；**
- **浮点操作。**

MIPS指令集结构：操作类型

- 符号“←”表示数据传送操作，其后附带一个下标n，也即“←n” 表示传送一个n位数据。
- 符号“##”用来表示两个域的串联操作，它可以出现在数据传送操作的任何一边。

MIPS指令集结构：操作类型

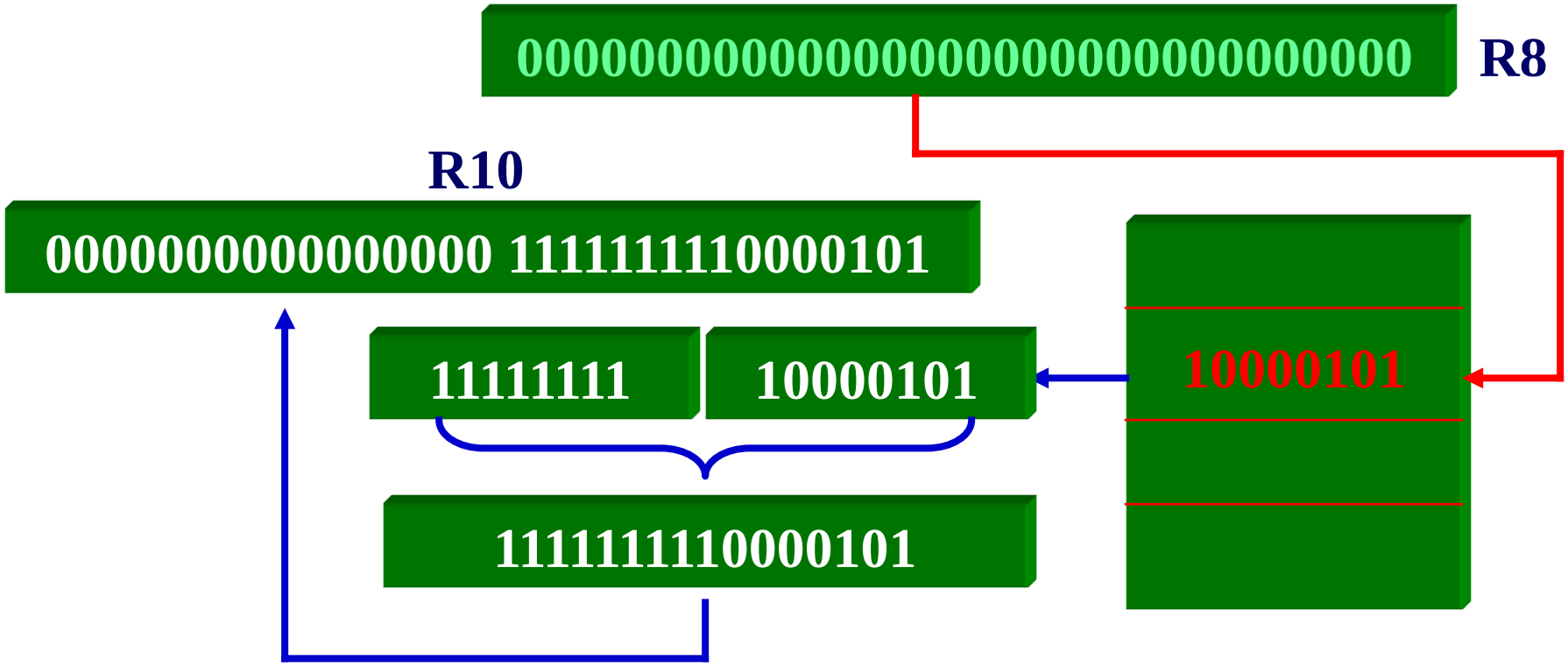
- 域的下标用来表明从该域中选择某一位。域中位的标记是从最高位开始标记，并且起始标记为0。下标可以是一个单独的数字，如$Regs[R4]_0$表示选择寄存器R4中内容的符号位；下标也可以是一个范围，如$Regs[R3]_{24..31}$表示选择寄存器R3中内容的最低一个字节。

MIPS指令集结构：操作类型

- 上标表示复制一个域，如0^{24}可以得到一个24位全为0的一个域。
- 变量Mem用来表示存储器中的一个数组，存储器按照字节寻址，它可以传送任何数目的字节。

MIPS指令集结构：操作类型

$$Regs[R10]_{16..31} \leftarrow_{16} (Mem[Regs[R8]]_0)^8 \,\#\#\, Mem[Regs[R8]]$$



MIPS指令集结构：Load和Store

- **Load和Store操作**：可以对MIPS的所有通用寄存器和浮点寄存器进行Load（载入）和Store（储存）操作，但是对通用寄存器R0的Load操作没有任何效果。

指令举例	指令名称	含 义
LW R2，40(R3)	装入字	Regs[R2]$\leftarrow_{32}$Mem[40+Regs[R3]]
LB R2，30(R3)	装入字节	Regs[R2]$\leftarrow_{32}$ (Mem[30+Regs[R3]]$_0$)^{24} ## Mem[30+Regs[R3]]
LBU R2，40(R3)	装入无符号字节	Regs[R2]$\leftarrow_{32}$ 0^{24} ## Mem[40+Regs[R3]]
LH R2，30(R3)	装入半字	Regs[R2]$\leftarrow_{32}$ (Mem[30+Regs[R3]]$_0$)^{16} ## Mem[30+Regs[R3]]## Mem[31+Regs[R3]]
LS F2，60(R4)	装入单精度浮点	Regs[F2]$\leftarrow_{32}$ Mem[60+Regs[R4]]
SW 300(R5)，R4	保存字	Mem[300+Regs[R5]]$\leftarrow_{32}$ Regs[R4]
SH 502(R4)，R5	保存半字	Mem[502+Regs[R4]]$\leftarrow_{16}$ Regs[R5]$_{16..31}$
SB 41(R3)，R2	保存字节	Mem[41+Regs[R3]]$\leftarrow_{8}$ Regs[R2]$_{24..31}$
SS 40(R2)，F2	保存单精度浮点数	Mem[40+Regs[R2]]$\leftarrow_{32}$ Regs[F2]

MIPS指令集结构：ALU操作

- **ALU操作**：在MIPS中，所有的ALU指令都是寄存器—寄存器型指令，其运算包含了简单的算术和逻辑运算，如加、减、AND、OR、XOR和移位。
- “设置相等”、“设置不等”、“设置小于”：寄存器比较指令（=，≠，<，>，≤，≥），如果比较结果为真，这些指令就在目标寄存器中填入1（表示真），否则填入0（表示假）。

MIPS指令集结构：ALU操作

指令实例	指令名称	含 义
ADD R1,R2,R3	加	Regs[R1] ← Regs[R2] + Regs[R3]
ADDI R1,R2,#3	和立即值相加	Regs[R1] ← Regs[R2] + 3
LHI R1,#42	载入高位立即值	Regs[R1] ← 42 ## 0^{16}
SLLI R1,R2,#5	逻辑左移立即值形式	Regs[R1] ← Regs[R2] <<5
SLT R1,R2,R3	设置小于	if (Regs[R2] <Regs[R3]) Regs[R1] ← 1 else Regs[R1] ← 0

MIPS指令集结构：转移操作

- 描述目标地址的方法：
  - 用带符号位的26位偏移量加上程序计数器的值来确定跳转的目标地址；
  - 指定一个寄存器，由寄存器中的内容决定跳转的目标地址。

MIPS指令集结构：转移操作

- 两种跳转类型：
  - 一种是简单跳转；
  - 另一种是跳转并链接（用于过程调用），它将下一条顺序指令地址（返回地址）保存在寄存器R31中。

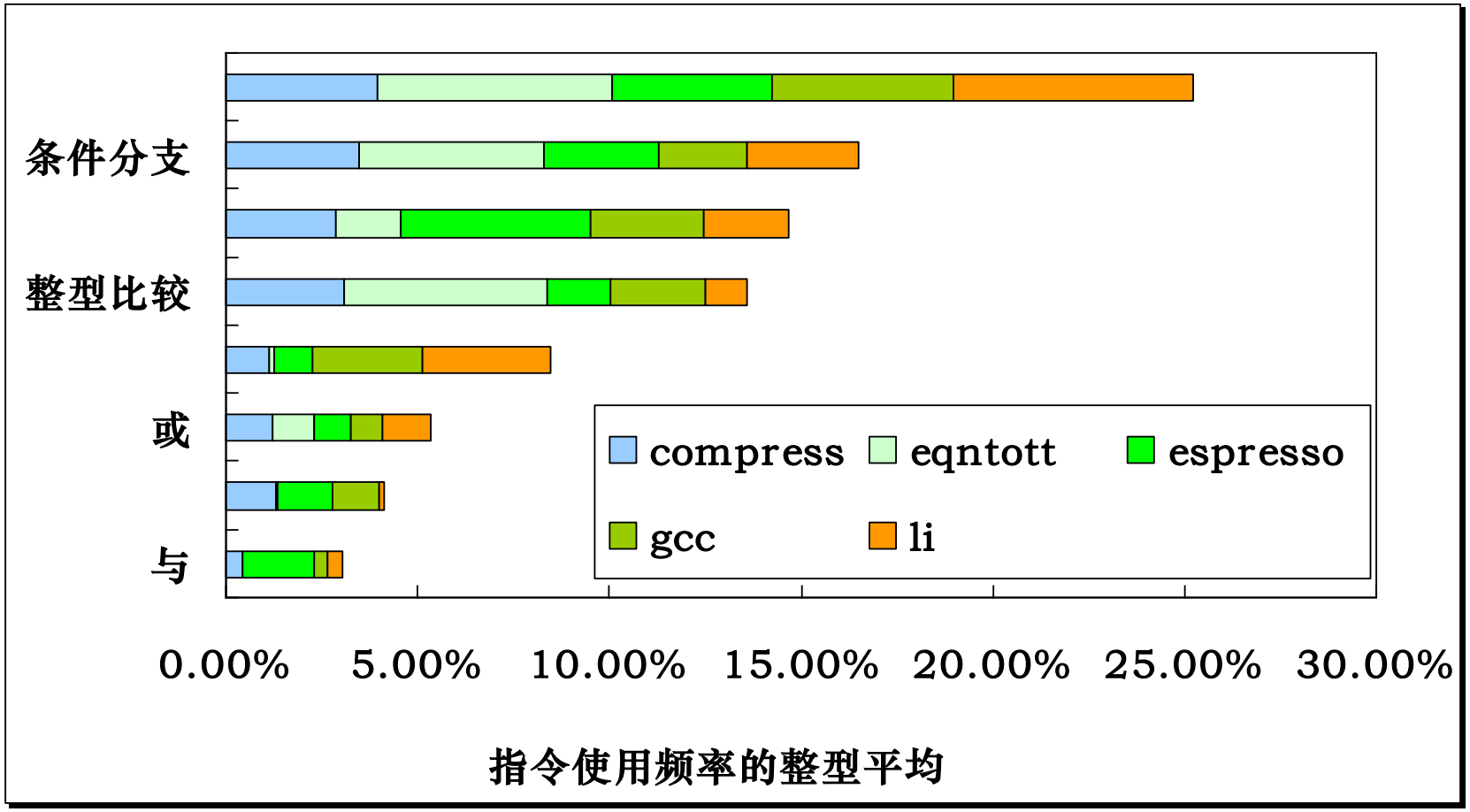
MIPS指令集结构：转移操作

指令实例	指令名称	含义
J name	跳转	PC ←name+PC+4; $-2^{25} \le$ name $\le 2^{25}$
JAL name	跳转并链接	Regs[R31] ← PC+4; PC ←name +PC+4; $-2^{25} \le$ name $\le 2^{25}$
JR R3	寄存器型跳转	PC ← Regs[R3];
JALR R2	寄存器型跳转并链接	Regs[R31]←PC+4; PC← Regs[R2];
BEQZ R4 , name	“等于0”分支	if (Regs[R4]==0) PC ← name+ PC+4; $-2^{15} \le$ name $\le 2^{15}$
BNQZ R4 , name	“不等于0”分支	if (Regs[R4]!=0) PC ← name+ PC+4; $-2^{15} \le$ name $\le 2^{15}$

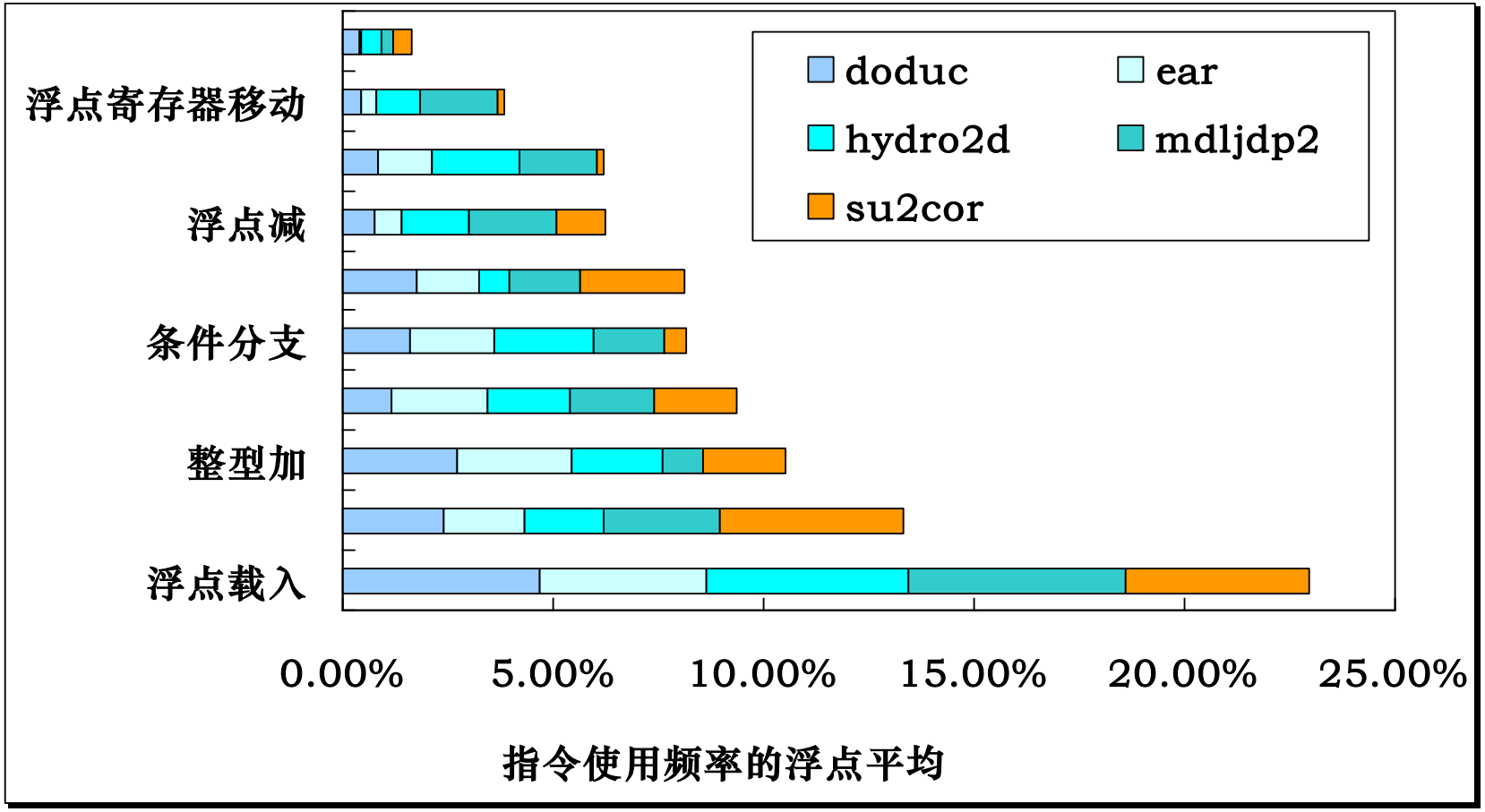
MIPS指令集结构：浮点操作

- 浮点操作：浮点指令的操作数来源于浮点寄存器，同时它还指明了相应的操作是单精度浮点操作还是双精度浮点操作。
  - 后缀D代表双精度浮点操作
  - 而后缀S代表单精度浮点操作
  - 如：ADDD、ADDS、SUBD、SUBS、MULTD、MULTS、DIVD、DIVS。
- MIPS的浮点操作有：加、减、乘、除。

MIPS中的常用指令


条件分支
整型比较
或
与
compress
eqntott
espresso
gcc
li
0.00%
5.00%
10.00%
15.00%
20.00%
25.00%
30.00%
指令使用频率的整型平均

MIPS中的常用指令

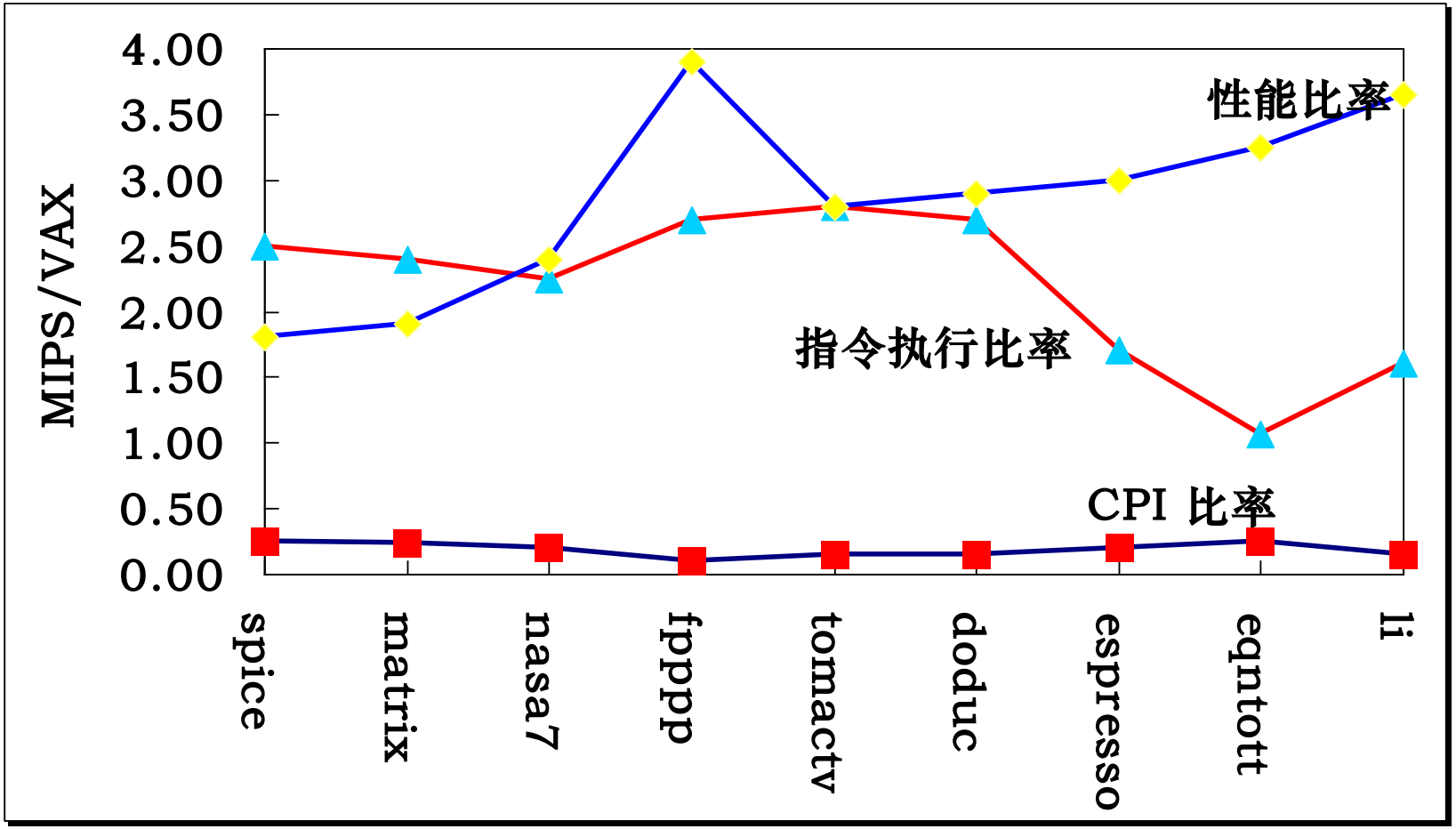

浮点寄存器移动
浮点减
条件分支
整型加
浮点载入
doduc
ear
hydro2d
mdljdp2
su2cor
0.00%
5.00%
10.00%
15.00%
20.00%
25.00%
指令使用频率的浮点平均

MIPS的效能分析

- 问题的提出：
  - MIPS指令集结构的指令格式、寻址方式和操作都非常简单。也许有人会担心，这些特性会使得目标代码中指令条数增多，导致程序运行时间加长，从而使这种指令集结构的机器性能并不会太高。

$$T_{CPU} = IC \times CPI \times T_{CLK}$$

MIPS的效能分析


MIPS/VAX
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
性能比率
指令执行比率
CPI 比率
spice
matrix
nasa7
fpppp
tomactv
doduc
espresso
eqntott
li

本章小结

- 讨论指令系统的基本概念
  - 指令的操作码与地址码
  - 操作数的类型
  - 操作的类型
- 寻址方式
  - 指令的寻址与操作数的寻址
  - 偏移寻址与立即寻址的范围

本章小结

- 指令系统结构的分类
  - 堆栈型、累加器型、通用寄存器型
  - 根据ALU操作数来源，通用寄存器型包括：寄存器-存储器型和寄存器-寄存器型
- 指令系统设计的基本原则
  - 完整性、规整性、正交性、高效率、兼容性
- **RISC和CISC**
- **MIPS指令集**

第四章作业

唐朔飞教材，P335，T1，T2，T6，T14

王志英教材，P62，T2，T12，T15，T18

注：T18，“每个地址字段的长度均为6位”

第 1 章　计算机系统概论

第 2 章　计算机系统量化分析基础

第 3 章　总线

第 4 章　指令系统

第 5 章　CPU设计与实现

第 6 章　基本流水线技术

第 7 章　指令级并行

第 8 章　存储系统的结构与优化

第 9 章　IO系统

第5章　CPU设计与实现

5.1 CPU 的结构

5.2 运算方法与ALU

5.3 多级时序系统（X86）

5.4 MIPS CPU的简单实现

第5章　CPU设计与实现

5.1 CPU 的结构

5.2 运算方法与ALU

5.3 多级时序系统（X86）

5.4 MIPS CPU的简单实现

5.1 CPU 的结构框图

一、 CPU 的功能

1. 控制器的功能

取指令 — 指令控制

分析指令

执行指令，发出各种操作命令 — 操作控制

控制程序输入及结果的输出 — 时间控制

总线管理

处理异常情况和特殊请求 — 处理中断

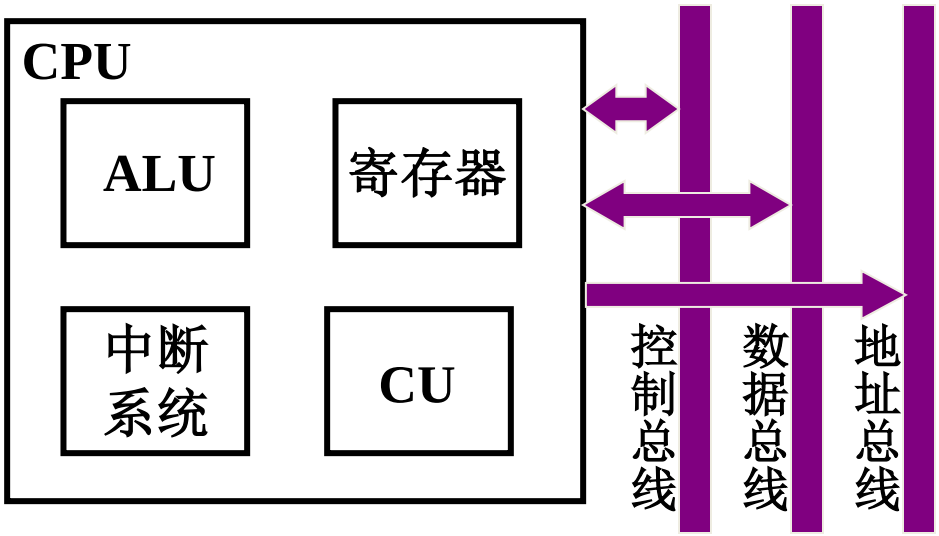
2. 运算器的功能

实现算术运算和逻辑运算 — 数据加工

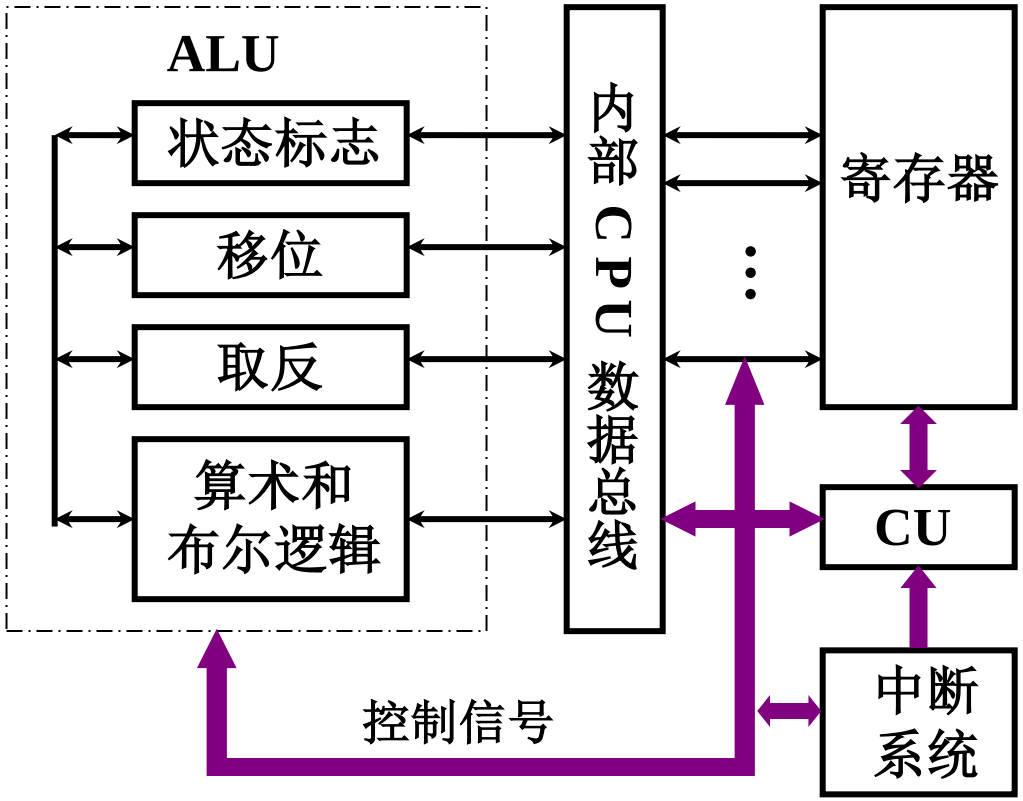
二、CPU 结构框图

1. CPU 与系统总线

指令控制		PC　IR
操作控制	}	CU　时序电路
时间控制	}	CU　时序电路
数据加工		ALU　寄存器
处理中断		中断系统



2. CPU 的内部结构



三、 CPU 的寄存器

1. 用户可见寄存器

(1) 通用寄存器　　存放操作数

可作 某种寻址方式所需的 专用寄存器

(2) 数据寄存器　　存放操作数（满足各种数据类型）

两个寄存器拼接存放双倍字长数据

(3) 地址寄存器　　存放地址，其位数应满足最大的地址范围

用于特殊的寻址方式　段基值　栈指针

(4) 条件码寄存器　存放条件码，可作程序分支的依据

如 正、负、零、溢出、进位等

2. 控制和状态寄存器

(1) 控制寄存器

PC → MAR → M → MDR → IR

控制 **CPU** 操作

其中 **MAR**、**MDR**、**IR**　　用户不可见

PC　　用户可见

(2) 状态寄存器

PSW 寄存器　　存放程序状态字

3. 举例　　**Z8000**　　**8086**　　**MC 68000**

四、 控制单元 CU 和中断系统

1. CU　产生全部指令的微操作命令序列

组合逻辑设计　　硬连线逻辑

微程序设计　　存储逻辑

2. 中断系统

五、ALU

第5章　CPU设计与实现

5.1 CPU 的结构

5.2 运算方法与ALU

5.3 多级时序系统（X86）

5.4 MIPS CPU的简单实现

5.2 运算方法与ALU

5.2.1 定点运算

5.2.2 浮点四则运算

5.2.3 算术逻辑单元

5.2.1 定点运算

一、移位运算

1. 移位的意义

15. m = 1500. cm

小数点右移 2 位

机器用语　　15 相对于小数点 左移 2 位

（ 小数点不动 ）

左移　　绝对值扩大

右移　　绝对值缩小

在计算机中，移位与加减配合，能够实现乘除运算

2. 算术移位规则

符号位不变

真值	码　制	添补代码
正数	原码、补码、反码	0
负数	原　码	0
	补　码	左移 添 0
		右移 添 1
	反　码	1

设机器数字长为 8 位（含 1 位符号位），写出 $A = -26$时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解：　　$A = -26 = -11010$

原码

移位操作	机　器　数	对应的真值
移位前	1,0011010	– 26
左移一位	1,0110100	– 52
左移两位	1,1101000	– 104
右移一位	1,0001101	– 13
右移两位	1,0000110	– 6

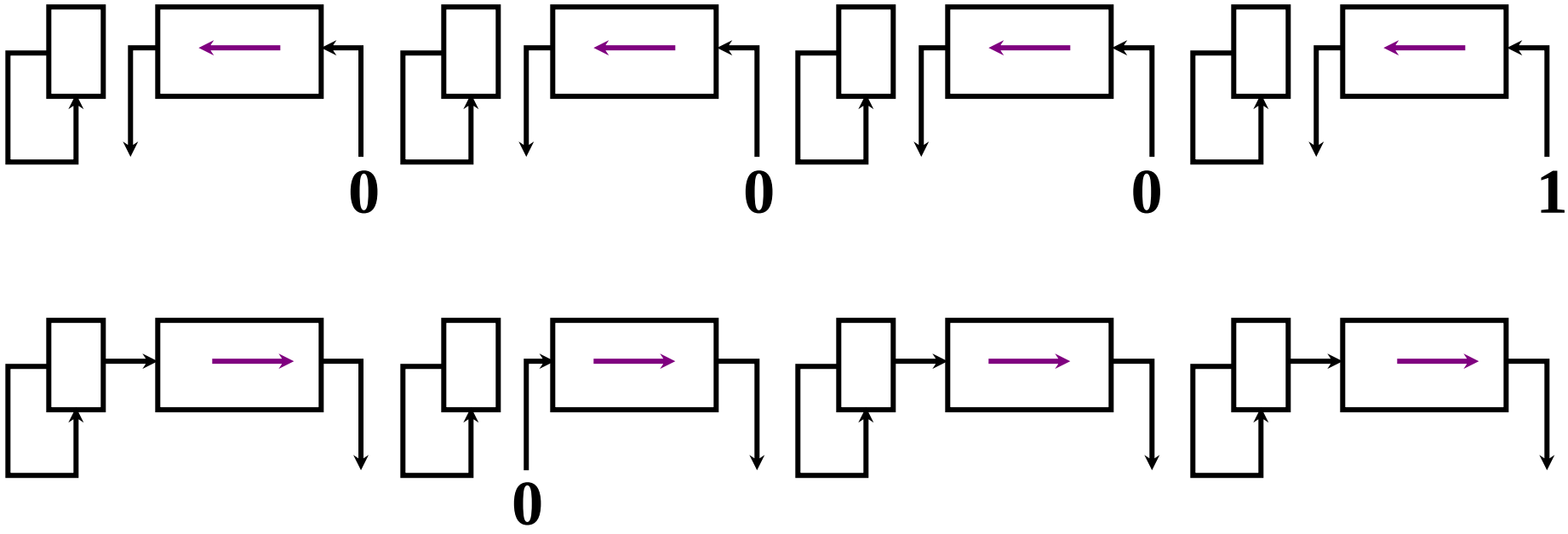
补码

移位操作	机　器　数	对应的真值
移位前	1,1100110	– 26
左移一位	1,1001100	– 52
左移两位	1,0011000	– 104
右移一位	1,1110011	– 13
右移两位	1,1111001	– 7

反码

移位操作	机　器　数	对应的真值
移位前	1,1100101	– 26
左移一位	1,1001011	– 52
左移两位	1,0010111	– 104
右移一位	1,1110010	– 13
右移两位	1,1111001	– 6

3. 算术移位的硬件实现



	（a）真值为正	（b）负数的原码	（c）负数的补码	（d）负数的反码
←丢 1	出错	出错	正确	正确
→丢 1	影响精度	影响精度	影响精度	正确

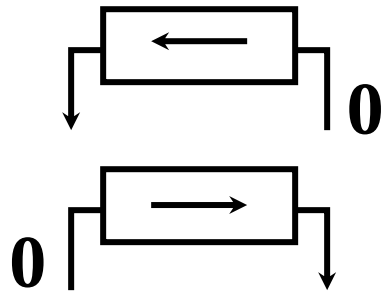
4. 算术移位和逻辑移位的区别

算术移位　　有符号数的移位

逻辑移位　　无符号数的移位

逻辑左移　　低位添 0，高位移丢

逻辑右移　　高位添 0，低位移丢



例如　　01010011　　10110010

逻辑左移　　10100110　　逻辑右移　　01011001

算术左移　　00100110　　算术右移　　11011001（补码）