第五讲：指令系统与MIPS汇编

1.1 指令系统概述

- 指令系统的基本问题
  - 操作类型：应该提供哪些（多少）操作？
    - 用LD/ST/INC/BRN已经足够编写任何计算程序，但不实用，程序太长。
  - 操作对象：如何表示？可以表示多少？
    - 大多数是双值运算（如A←B+C)
    - 存在单值运算（如A←~B)
  - 指令格式：如何将这些内容编码成一致的格式？
    - 指令长度、字段、编码等问题

1.1 指令系统概述

❖ 机器指令的要素

- 操作码(Operation Code)：指明进行的何种操作
- 源操作数地址(Source Operand Reference)：参加操作的操作数的地址，可能有多个。
- 目的操作数地址(Destination Operand Reference)：保存操作结果的地址。
- 下条指令的地址(Next Instruction Reference)：指明下一条要运行的指令的位置，一般指令是按顺序依次执行的，所以绝大多数指令中并不显式的指明下一条指令的地址，也就是说，指令格式中并不包含这部分信息。只有少数指令需要显示指明下一条指令的地址。

1.1 指令系统概述

❖指令类型

- 数据传输指令：寄存器与存储器之间，寄存器之间传递数据；
- 算术/逻辑运算指令：寄存器（或存储器）中整型数或逻辑型数据的运算操作。
- 程序控制指令：控制程序执行顺序，条件转移或跳转，子程序调用和返回等；
- 浮点运算指令：处理浮点数的运算。

1.1 指令系统概述

- 操作数的类型
  - 数值（无符号、定点、浮点）
  - 逻辑型数、字符
  - 地址（操作数地址、指令地址）
- 操作数的位置
  - 存储器（存储器地址）
  - 寄存器（寄存器地址）
  - 输入输出端口（输入输出端口地址）
- 操作数的存储方式
  - 大端（big-endian）次序：最高有效字节存储在地址最小位置
  - 小端（little-endian）次序：最低有效字节存储在地址最小位置

例： Int a; //0x12345678

地址	值
a+0	12
a+1	34
a+2	56
a+3	78

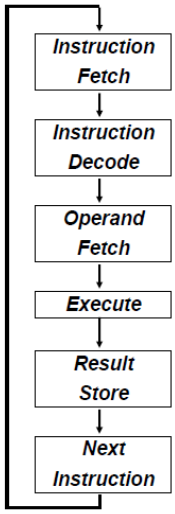
大端次序

地址	值
a+0	78
a+1	56
a+2	34
a+3	12

小端次序

1.1 指令系统概述

- 从指令执行周期看指令涉及的内容



Obtain instruction from program storage
指令地址、指令长度（定长/变长）

Determine required actions
指令格式、操作码编码、操作数类型

Locate and obtain operand data
地址码格式、寻址方式、操作数格式和存放方式

Compute result value or status
操作类型、标志或条件码

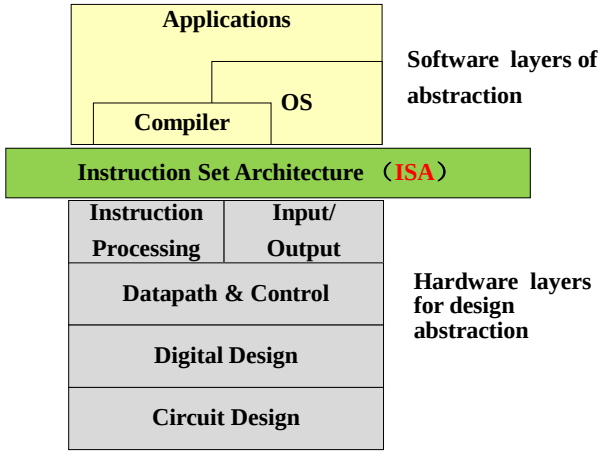
Deposit results in storage for later use
结果数据位置

Determine successor instruction
下条指令地址（顺序 / 转移）

from： 南大袁春风老师ppt

❖ 指令集系统结构(ISA)

- ➢ 机器语言编程者的视角，机器内部结构和行为能力的指令级抽象



❖指令集系统架构（ISA）种类

- ➢大部分ISA都可归类为通用寄存器系统结构
- ➢Register-Memory式ISA（如80X86）
  - ▪ 多种指令可以访问内存；
  - ▪ 存在寄存器操作数和内存操作数直接运行的指令；
- ➢Register-Register (Load-Store) 式ISA（如MIPS）
  - ▪ 只有装载（LOAD）和存储（STORE）指令可以访问内存
  - ▪ 运算指令操作数全部为寄存器操作数；

❖Load-Store是ISA的一种趋势

❖通用寄存器的优势

- 寄存器比存储器快
- 寄存器便于编译器使用
- 寄存器可以保存变量
- 减少存储器访问，提高速度
- 提高代码密度，寄存器地址比存储器地址短

第五讲：指令系统与MIPS汇编

1.2 指令格式

❖ **机器指令：** 计算机硬件可以执行的表示一种基本操作的二进制代码

- 指令格式：操作码 + 操作数（操作数地址）
- 操作码：指明指令的操作性质
- 操作数（地址）：指明操作数的位置（或操作数本身）

操作码	操作数地址
11010101	10000100 01010001 10100000

❖ **指令的表示**

- 机器表示：二进制代码
- 符号化表示：助记符，如： MOV AX, BX

1.2 指令格式

❖ **操作数地址的数目**

- 三地址：Des ← (Sur1) OP (Sur2)

OP	Des Add	Sur1 Add	Sur2 Add

- 双地址：Des ← (Sur) OP (Des)

OP	Des Add	Sur Add

- 单地址：累加器作为其中一个操作数的双操作数型，或单操作数型

OP	Add

- 零地址：隐含操作数型，或无操作数型

OP

1.2 指令格式

❖操作码结构

- ➢固定长度操作码：操作码长度（占二进制位数）固定不变。
  - 硬件设计简单
  - 指令译码时间开销较小
  - 指令空间效率较低
- ➢可变长度操作码：操作码长度随指令地址数目的不同而不同。
  - 硬件设计相对复杂
  - 指令译码时间开销较大
  - 指令空间利用率较高

❖指令长度

- ➢定长指令系统，如MIPS指令
- ➢变长指令系统：一般为字节的整数倍，如80X86指令

第十二讲

第五讲：指令系统与MIPS汇编

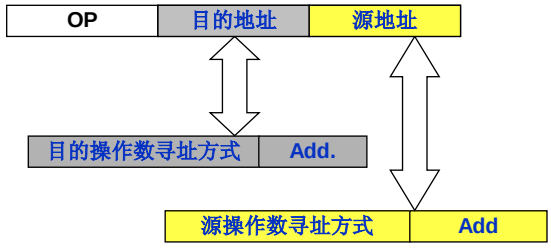
1.3 寻址方式

- 形式地址与有效地址
  - 形式地址：指令中直接给出的地址编码。
  - 有效地址：操作数在内存单元中的地址，可根据形式地址和寻址方式计算出来。
  - 寻址方式：根据形式地址计算出操作数有效地址的方式（算法）
- 常用寻址方式
  - 立即寻址
  - 寄存器直接寻址
  - 寄存器间接寻址
  - 基址寻址/变址寻址
  - 相对寻址：基址寻址的特例，程序计数器PC作为基址寄存器
  - 堆栈寻址

1.3 寻址方式

❖寻址方式的确定

- ➢在操作码中给定寻址方式：
  - ▪ 如MIPS指令，指令中仅有一个主(虚)存地址的，且指令中仅有一二种寻址方式。Load/store型机器指令属于这种情况。
- ➢专门的寻址方式位
  - ▪ 如X86指令，指令中有多个操作数，且寻址方式各不相同，需要各自说明寻址方式。



1.3 寻址方式

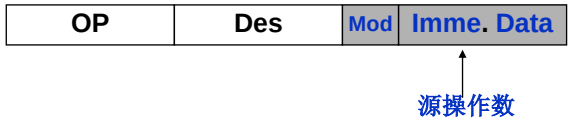
❖指令代码和寻址描述中有关缩写的约定

- ➢OP：操作码
- ➢Des：目的操作数地址
- ➢Sur：源操作数地址
- ➢A或Add： 形式地址（内存地址）
- ➢Mod：寻址方式
- ➢Rn： 通用寄存器
- ➢Rx：变址寄存器
- ➢Rb： 基址寄存器
- ➢SP：堆栈指针（寄存器）
- ➢EA：有效地址
- ➢Data：操作数
- ➢Operand：操作数
- ➢(Rn)：寄存器Rn的内容（值）
- ➢Mem[A]：内存地址为A的单元的内容
- ➢Imme. Data： 立即数
- ➢XXH：16进制数XX

1.3 寻址方式

❖立即寻址

➢操作数直接在指令代码中给出。



❖说明

➢立即寻址只能作为双操作数指令的源操作数。

➢Operand = Imme. Data

➢例：MOV AX,1000H （80X86指令， AX←1000H）

addi $s1, $s2, 100 （MIPS指令, $s1 ← $s2+100）

❖思考

➢立即寻址的操作数在什么地方，存储器 or 寄存器？

➢立即数的地址？

1.3 寻址方式

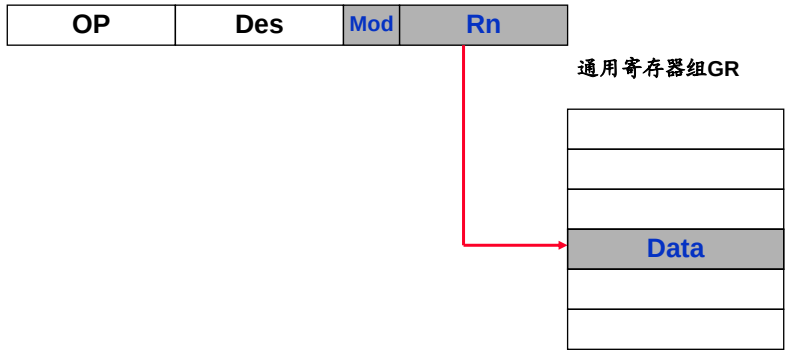
❖寄存器直接寻址

➢操作数在寄存器中，指令地址字段给出寄存器的地址（编码）

➢EA = Rn, Operand = (Rn)

➢例：MOV [BX], AX （80X86指令）

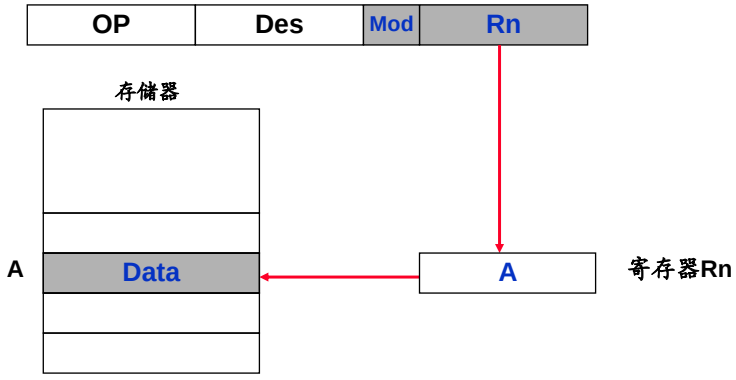
add $s1,$s2,$s3 （MIPS指令， $s1←$s2+$s3）



1.3 寻址方式

❖寄存器间接寻址

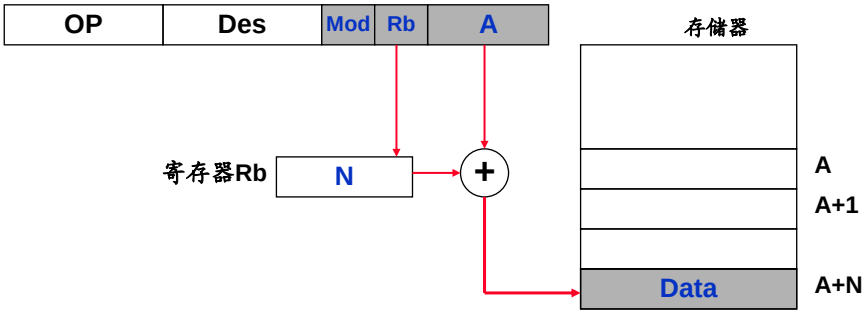
- 操作数在存储器中，指令地址字段中给出的寄存器的内容是操作数在存储器中的地址。
- EA = (Rn), Operand = Mem[(Rn)]
- 例：MOV AX, [BX] （80X86指令, AX ←Mem[(BX)]）



1.3 寻址方式

❖基址寻址

- 操作数在存储器中，指令地址字段给出一基址寄存器和一形式地址，基址寄存器的内容与形式地址之和是操作数的内存地址。
- EA = (Rb)+A, Operand = Mem[(Rb)+A]
- 例：MOV AX, 1000H[BX] （80X86指令，AX←Mem[(Bx)+1000]）
  lw $s1,100($s2) （MIPS指令，$s1 ←Mem[$s2+100]）

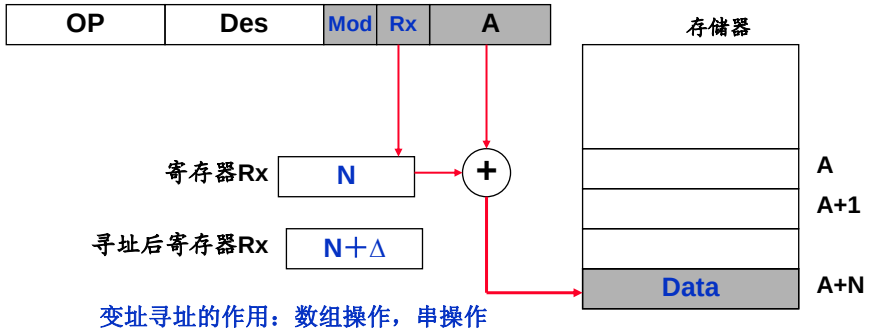


基址寻址的作用：较短的形式地址长度可以实现较大的存储空间的寻址。

1.3 寻址方式

❖变址寻址

- 操作数在存储器中，指令地址字段给出一变址寄存器和一形式地址，变址寄存器的内容与形式地址之和是操作数的内存地址。
- EA = (Rx)+A, Operand = Mem[(Rx)+A]
- 有的系统中，变址寻址完成后，变址寄存器的内容将自动进行调整。
  Rx ← (Rx) + Δ（操作数Data的字节数）
- 例：MOV AX, 1000H[DI]（80X86指令）

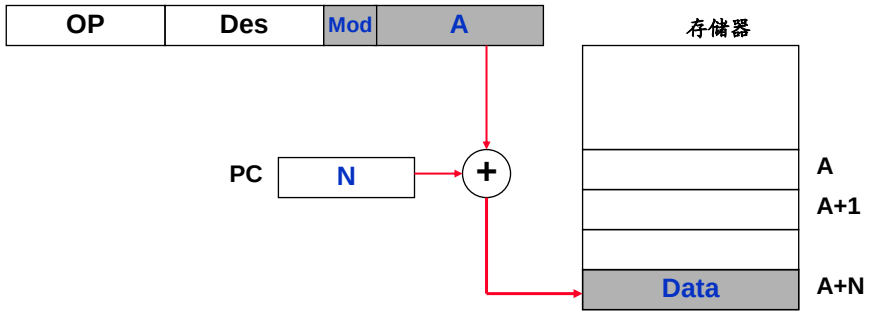


变址寻址的作用：数组操作，串操作

1.3 寻址方式

❖相对寻址

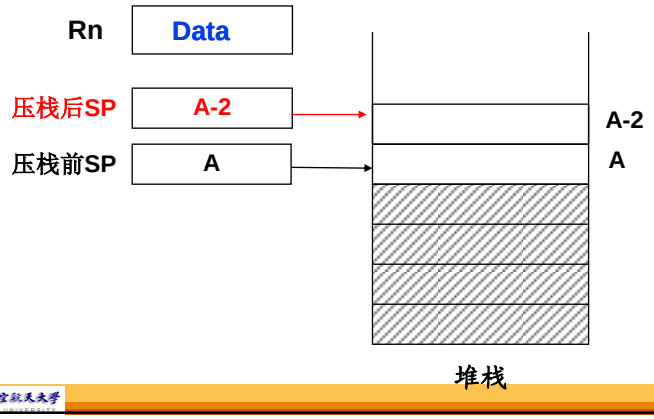
- 基址寻址的特例，由程序计数器PC作为基址寄存器，指令中给出的形式地址作为位移量，二者之和是操作数的内存地址。
- EA = (PC)+A, Operand = Mem[(PC)+A]
- 例：JNE A（80X86指令）
  beq $s1, $s2, 100 (MIPS指令)



1.3 寻址方式

❖堆栈寻址

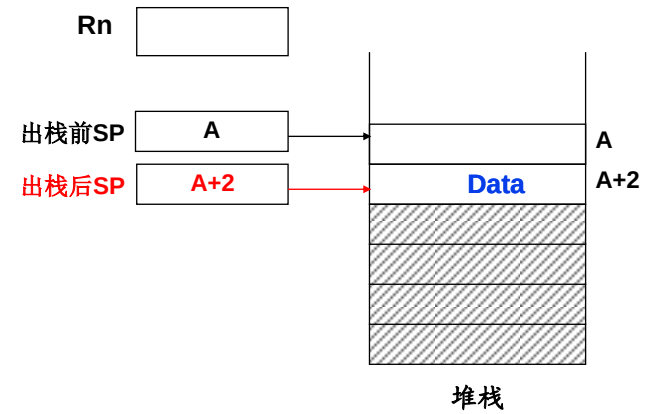
- 堆栈的结构：一段内存区域。
- 堆栈指针(SP)：是一个特殊寄存器部件, 指向栈顶
- 压栈操作：PUSH Rn，假定寄存器Rn为16位寄存器
  (SP) ← (Rn)，SP ← (SP)-2



1.3 寻址方式

❖堆栈寻址

- 出栈操作：POP Rn，假定寄存器Rn为16位寄存器
  SP ← (SP) + 2， Rn ← ((SP))



1.3 寻址方式 - 小结

- **指令操作数的寻址方式**
  - 形式地址、有效地址、寻址

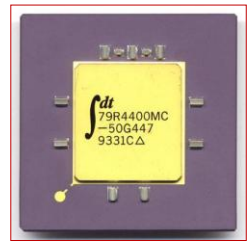
寻址方式	操作数存放地	操作数有效地址EA	操作数的值 Operand	访问操作数所需访存次数
立即寻址	指令中	指令中的立即数字段	Imme. Data	0
寄存器直接寻址	寄存器	Rn	(Rn)	0
寄存器间接寻址	存储器	(Rn)	((Rn))	1
基址寻址	存储器	(Rb)+A	((Rb)+A)	1
变址寻址	存储器	(Rx)+A	((Rx)+A)	1
相对寻址	存储器	(PC)+A	((PC)+A)	1
堆栈寻址	存储器	(SP)	((SP))	1

第五讲：指令系统与MIPS汇编

2. 2 MIPS 指令系统

❖ MIPS R系列CPU简介

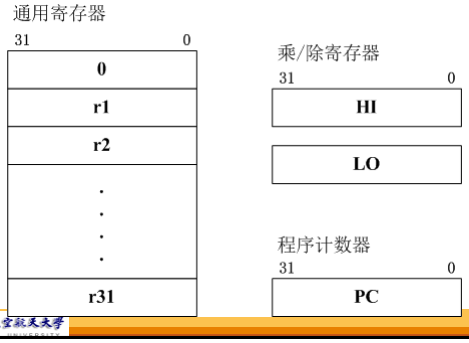
- RISC（Reduced Instruction set Computer，精简指令集计算机, RISC）微处理器
- MIPS（Microprocessor without interlocked piped stages，无内部互锁流水级的微处理器），
- 最早在80年代初由Stanford大学Patterson教授领导的研究小组研制出来，MIPS公司的R系列就是在此基础上开发的RISC微处理器。
- 1986年，推出R2000（32位）
- 1988年，推出R3000（32位）
- 1991年，推出R4000（64位）
- 1994年，推出R8000 （64位）
- 1996年，推出R10000
- 1997年，推出R20000
- 通用指令体系MIPS I、MIPS II、MIPS III、MIPS IV到MIPS V，嵌入式指令体系MIPS16、MIPS32到MIPS64，发展已经十分成熟。在设计理念上MIPS强调软硬件协同提高性能，同时简化硬件设计。

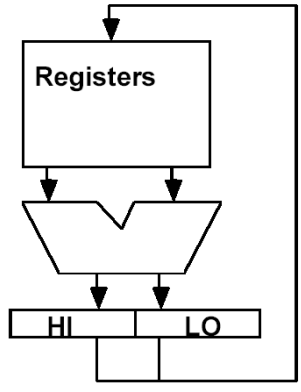


2.2 MIPS 指令系统

❖ MIPS R2000/R3000 寄存器结构

- 32位虚拟地址空间
- 32×32 bit GPRs, $0~$31
- 32×32 bit FPRs, $f0~$f31
- HI, LO, PC（相应的指令mfhi/mthi,mflo/mtlo来实现HI和LO寄存器与通用寄存器之间的数据交换）





2.2 MIPS 指令系统

❖MIPS 寄存器使用的约定

Name	Reg. Num	Usage
zero	0	**constant value =0(恒为0)**
at	1	**reserved for assembler(为汇编程序保留)**
v0 – v1	2 – 3	**values for results(过程调用返回值)**
a0 – a3	4 – 7	**Arguments(过程调用参数)**
t0 – t7	8 – 15	**Temporaries(临时变量)**
s0 – s7	16 – 23	**Saved(保存)**
t8 – t9	24 – 25	**more temporaries(其他临时变量)**
k0 – k1	26 – 27	**reserved for kernel(为OS保留)**
gp	28	**global pointer(全局指针)**
sp	29	**stack pointer (栈指针)**
fp	30	**frame pointer (帧指针)**
ra	31	**return address (过程调用返回地址)**

Registers are referenced either by number—$0…$31, or by name —$t0,$s1…$ra.

2.2 MIPS 指令系统

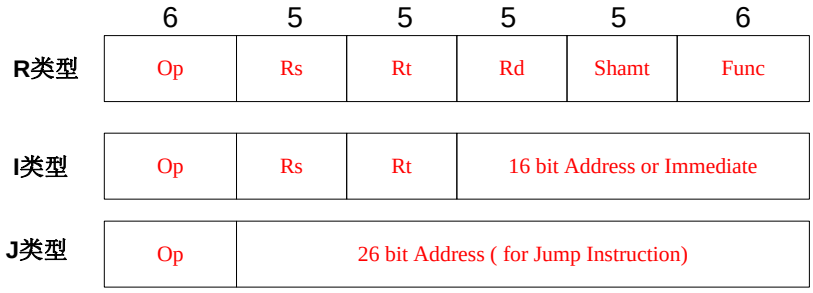
❖ MIPS指令格式

- MIPS只有3种指令格式，32位固定长度指令格式
  - R（Register）类型指令：两个寄存器操作数计算，结果送寄存器
  - I（Immediate）类型指令：使用1个16位立即数作操作数；
  - J（Jump）类型指令：跳转指令，26位跳转地址
- 最多3地址指令：add $t0, $s1, $s2 ($t0←$s1+$s2)
- 对于Load/Store指令，单一寻址模式：Base＋Displacement
- 没有间接寻址
- 16位立即数
- 简单转移条件（与0比较，或者比较两个寄存器是否相等）
- 无条件码

2.2 MIPS 指令系统

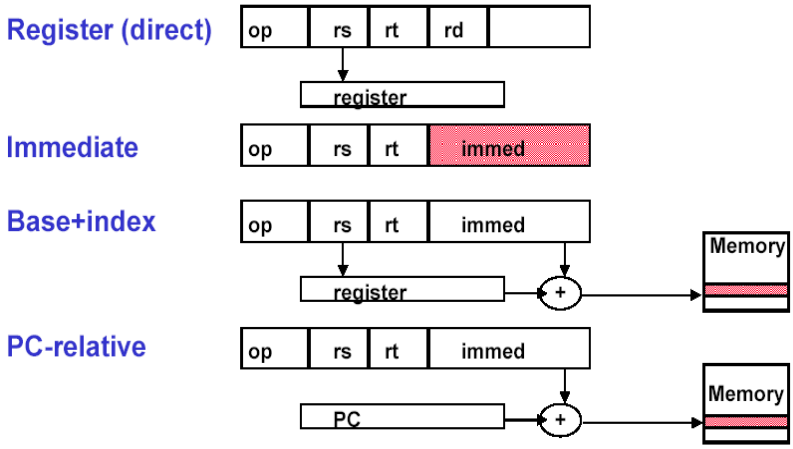
❖ MIPS 指令格式

- Op： 6 bits, Opcdoe
- Rs: 5 bits, The first register source operand
- Rt: 5 bits, The second register source operand
- Rd: 5 bits, The register destination operand
- Shamt: 5 bits, Shift amount (shift instruction)
- Func: 6 bits, function code (another Opcode)



2.2 MIPS 指令系统

❖ MIPS寻址方式



2.2 MIPS 指令系统 —— 指令类型

❖ Load/Store（取数/存储）指令

- I类型指令，存储器与通用寄存器之间传送数据
- 支持唯一的存储器寻址方式：Base+Index
- 取数指令：LB（取字节）、LBU（取不带符号字节）、LH（取半字）、LHU（取不带符号的半字）、LW（取字）、LWL、LWR
- 存储指令：SB（存字节）、SH（存半字）、SW（存字）、SWL、SWR

❖ 运算指令

- R类型指令 和 I类型指令
- 算术运算：add, addu, addi, addiu, sub, subu, mul, mulu, div, divu, mfhi, mflo等
- 逻辑运算：and, andi, or, ori, xor, xori, nor等
- 移位指令：sll, srl, sra, sllv, srlv, srav等

2.2 MIPS 指令系统 —— 指令类型

❖ 跳转和转移指令：控制程序执行顺序

- 跳转指令：J类型指令（26位绝对转向地址）或R类型指令（32位的寄存器地址）
- 转移指令：I 类型指令，PC-relative寻址方式，相对程序计数器的16位位移量（立即数）。
- 跳转：J 、JAL、JR、JALR
- 转移：BEQ（相等转移）、BNE（不等转移）、BLEZ（小于或等于0转移）、BGTZ（大于0 转移）、BLTZ（小于0转移）、BLTZAL、BGEZAL

❖ 特殊指令

- R类型指令
- 系统调用SYSCALL
- 断点BREAK

2.2 MIPS 指令系统

❖R-Type指令编码示例

➢指令：**add $t0, $s1, $s2 ; t0 ← (S1) + (s2)**

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
指令格式	Op	Rs	Rt	Rd	Shamt	Func
指令编码	00000	10001	10010	01000	00000	10000

Rd ← (Rs) + (Rt)

- Op = 000000（表示R-Type)
- Func = 100000（表示add）
- Rs = 10001（表示s1）
- Rt = 10010（表示s2）
- Rd = 01000（表示t0）
- Shamt=00000（表示没有移位）

2.2 MIPS 指令系统 ——— 指令示例

Instruction	*Example*	*Meaning*	*Comments*
add	add $1,$2,$3	$1← $2 + $3	3 operation
subtract	sub $1,$2,$3	$1← $2 – $3	3 operation
add immediate	addi $1,$2,100	$1← $2 + 100	+ constant
multiply	mult $2,$3	Hi,Lo← $2× $3	64-bit signed product
divide	div $2,$3	Lo ← $2 ÷ $3 Hi ← $2 mod $3	Lo = quotient Hi = remainder
move from Hi	mfhi $1	$1 ← Hi	Get a copy of Hi
move from Lo	mflo $1	$1 ← Lo	Get a copy of Lo
and	and $1,$2,$3	$1← $2 & $3	Logical AND
or	or $1,$2,$3	$1← $2 | $3	Logical OR
store	sw $3,500($4)	Mem($4+500)←$3	Store Word
load	lw $1,-30($2)	$1←Mem($2-30)	Load word
jump and link	jal 1000	$31=PC+4 Go to 1000	Procedure call
jump register	jr $31	Go to $31	procedure return
set on less than	slt $1,$2,$3	if ($2 < $3) than $1=1 else $1=0	

2.2 MIPS 指令系统 —— SWAP: MIPS过程示例

```
swap:
  addi   $sp,$sp, –12      ; Make room on stack for 3 registers
  sw     $31, 8($sp)       ; Save return address
  sw     $s2, 4($sp)       ; Save registers on stack
  sw     $s3, 0($sp)

  ....
```

```
sll     $s2, $a1, 2        ; mulitply k by 4
addu    $s2, $s2, $a0      ; address of v[k]
lw      $t0, 0($s2)        ; load v[k]
lw      $s3, 4($s2)        ; load v[k+1]
sw      $s3, 0($s2)        ; store v[k+1] into v[k]
sw      $t0, 4($s2)        ; store old v[k] into v[k+1]
```

```
  lw     $s3, 0($sp)       ; Restored registers from stack
  lw     $s2, 4($sp)
  lw     $31, 8($sp)       ; Restore return address
 addi    $sp,$sp, 12       ; restore top of stack
  jr     $31               ; return to place that called swap
```