

数据的机器级表示与运算

整数的基本运算：按位运算、逻辑运算

主讲人：邓倩妮

上海交通大学

部分内容来自：

1. 《深入理解计算机系统》第三版，机械工业出版社，作者：Bryant,R.E.等
2. Computer Organization and Design, 4th Edition, Patterson & Hennessy



上海交通大学

SHANGHAI JIAO TONG UNIVERSITY



本节内容



- 对整数的基本操作
 - 按位运算
 - 逻辑运算



布尔运算



- 逻辑代数
- Encode “True” as 1 and “False” as 0

And

- $A \& B = 1$ when both $A=1$ and $B=1$

$\&$	0	1
0	0	0
1	0	1

Or

- $A | B = 1$ when either $A=1$ or $B=1$

$ $	0	1
0	0	1
1	1	1

Not

- $\sim A = 1$ when $A=0$

\sim	
0	1
1	0

Exclusive-Or (Xor)

- $A \wedge B = 1$ when either $A=1$ or $B=1$, but not both

\wedge	0	1
0	0	1
1	1	0



二进制向量按位运算



- 对位向量的操作：

01101001	01101001	01101001	
& 01010101	01010101	^ 01010101	~ 01010101
<u> </u>	<u> </u>	<u> </u>	<u> </u>
01000001	01111101	00111100	10101010

- 运用布尔代数的特性, 对整个向量按位进行计算
- All of the Properties of Boolean Algebra Apply



C语言中的按位运算



- C语言中的运算符 $\&$, $|$, \wedge , \sim
- 运用于任何 “integral” 数据类型：long, int, short, char, unsigned
- 按位运算：将以上数据类型的参数，看做数据原有长度不变的位向量
- Examples (Char 数据类型)
 - $0x41 \& 0xBE$
 - $01000001_2 \& 10111110_2$
 - $0x00 | 0xFF$
 - $00000000_2 | 11111111_2$
 - $0x69 \wedge 0x41$
 - $01101001_2 \wedge 01000001_2$



MIPS Arithmetic Logic Unit (ALU)

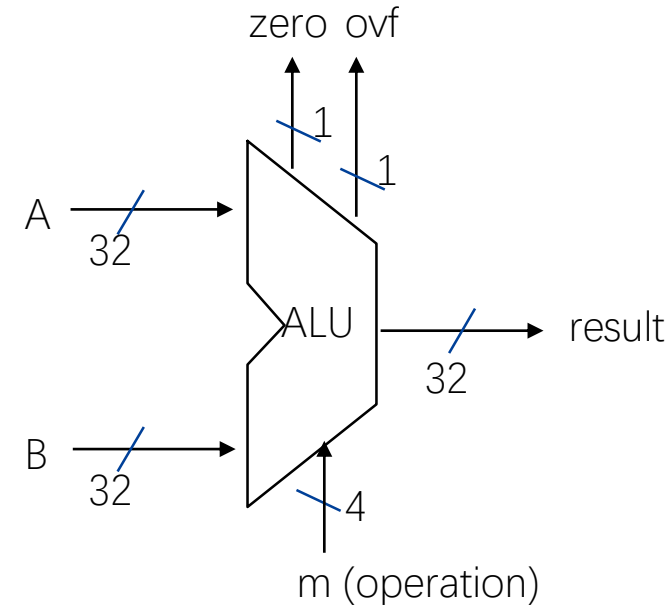


- MIPS指令

- add, addi, addiu, addu
sub, subu
mult, multu, div, divu
sqrt

and, andi, nor, or, ori, xor, xori

beq, bne, slt, slti, sltiu, sltu



按位运算的应用



- 掩码 (masking) 操作
 - 与给定的一个位模式按位“与”，提取需要的位
 - 例如： $0x0F \& 0x8C = 0X0C$
 - 通过掩码提取了一个字节的低四位
- 对位向量进行：
 - “置1”： $0xBC \mid 0xFF$
 - “清0”： $0xBC \& 0x00$
 - “测试是否为1”： $0xBC \& 0xFF$
 - “测试是否为0”： $0xBC \mid 0x00$



按位运算的应用（续）

对集合（Sets）进行表达和操作

例如：Width w bit vector represents subsets of $\{0, \dots, w-1\}$

- $a_j = 1$ if $j \in A$

- 01101001 $\{0, 3, 5, 6\}$

- 76543210

- 01010101 $\{0, 2, 4, 6\}$

- 76543210

Operations

- $\&$ Intersection 交集 01000001 $\{0, 6\}$
- $|$ Union 并集 01111101 $\{0, 2, 3, 4, 5, 6\}$
- \wedge Symmetric difference 对称差集 00111100 $\{2, 3, 4, 5\}$
- \sim Complement 补集 10101010 $\{1, 3, 5, 7\}$





按位运算的特点

▪ 运算符优先级

- \sim
- $>$ 算术运算 $>$ 移位运算 $>$ 关系运算
- $>$ $\&$
- $>$ \wedge
- $>$ $|$

例如表达式：

$\sim 0x41 \ \& \ 0x69 \ | \ 0x55 \ \wedge \ 0x01$

运算顺序 先 \sim ，再 $\&$ 、 \wedge ，最后 $|$



按位运算的特点（续）



若：两运算数位数不同，
较短数的高位：带符号扩展 or 无符号扩展？
这取决于编译器(国际标准化组织 ISO C90 , ISO C99).

- 例如：
- `char a= 0X80;`
- `short j= 0X8000;`
- `printf("常量计算： %d\n", 0X80 | 0X8000); // 32896`
- `printf("变量计算： %d\n", a | j); // -128`

`0X80 | 0X8000=?`

`0000 0000 1000 0000 | 1000 0000 0000 0000`

- `a | j=?` `1111 1111 1000 0000 | 1000 0000 0000 0000`



MIPS Arithmetic Logic Unit (ALU)

■ MIPS指令

■ add, addi, addiu, addu

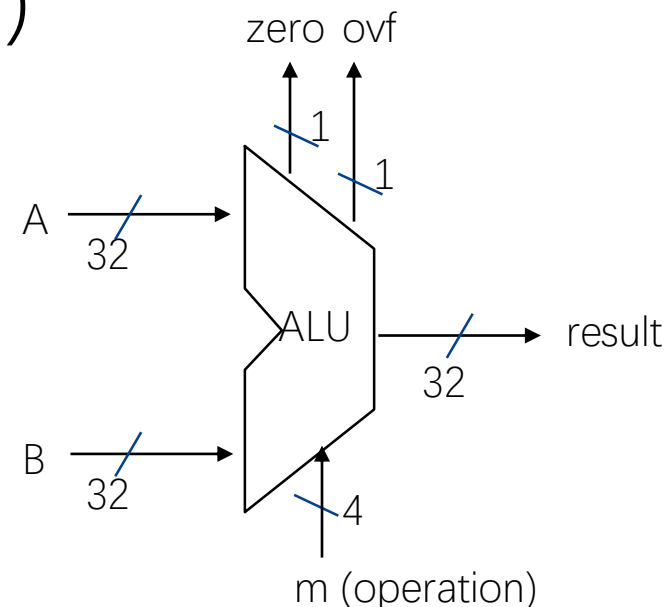
sub, subu

mult, multu, div, divu

sqrt

and, andi, nor, or, ori, xor, xori

beq, bne, slt, slti, sltiu, sltu



□ 对某些指令的特殊处理 With special handling for

sign extend – addi, addiu, slti, sltiu

zero extend – **andi, ori, xori**

overflow detection – add, addi, sub



本节内容



- 对整数的基本操作
 - 按位运算
 - 逻辑运算



C语言中的逻辑运算



- 逻辑运算符：
 - `&&` (and) `||` (or) `!` (not)
 - `&&` (对比 `&`) , `||` (对比 `|`) , `!` (对比 `~`)
 - 优先级：`!` > `~` > 关系运算 > `&` > `^` > `|` > `&&` > `||`
- 逻辑表达式：由逻辑运算符连接起来的表达式,其结果为“真(true)”或“假(false)”
 - 例子 (char 类型的数据)
 - `!0x41 && 0x00`
 - `!0x00 || 0x01`
 - `!0x41 || 0x010x69 && 0x55`
 - `0x69 || 0x55 && 0x01`



逻辑表达式



- 参加逻辑运算的对象可为任意类型的数据，
 - 0为false，非0为true。

$5 \% 2 \ \&\& \ p$

结果： p

- 短路求值： $x \ op \ y$

- 先处理左边x。如左边已能决定此逻辑表达式的结果，则右边y不执行。

$5 > 3 \ \&\& \ 2 \ || \ 8 < 4 - !0$ 结果： 1



按位运算与逻辑运算的对比



- 逻辑运算：非数值运算
 - 只有“true”“false”两个值
- 按位运算：一种数值运算
- 例如： $x = \text{FAH}$, $y = \text{7BH}$
 - $x \wedge y = \text{81H}$;
 - $\sim(x \wedge y) = \text{7EH}$
 - $!(x \wedge y) = \text{00H}$ 等价于“ $x = y$ ”,
 - 而不是 $\sim(x \wedge y)$



小结



- C语言中：按位运算、逻辑运算
 - 功能完全不同
 - 容易混淆
- 高级语言中的各种运算
 - 编译成底层的算术运算指令和逻辑运算指令实现
 - 底层指令在机器硬件上直接被执行
 - 作为程序员，应该理解其原理
 - 后续：移位、加法、乘法等等



谢谢！

