计算机作曲理论第6期

-指令集架构(3)-

大连理工大学立命馆大学国际信息软件学部大森孝之

讲座内容

- 指令格式
  - 什么是R格式和I格式?
  - 指令与机器语言的对应关系
  - 使用数组的汇编代码_用于分支的汇编代码
  - 无条件分支和 J 格式
  - 大/小比较指令
  - 寻址方式

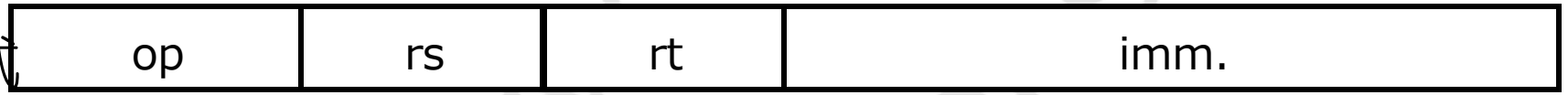
分支处理

- 用于确定 C 条件和实现循环 (if、while、for 等)

beq $s0, $s1, L1 · 当$ s0 和$ s1 的值相等时标记为L1 的指令跳转 (branch on equal)

bne $s0, $s1, L1 · 当$s0和$s1的值不同时标记为L1的指令跳转 (branch on not equal)

I 格式



beq (rs), (rt), (imm.)

bne (rs), (rt), (imm.)

带符号的16位整数
相对地址规范

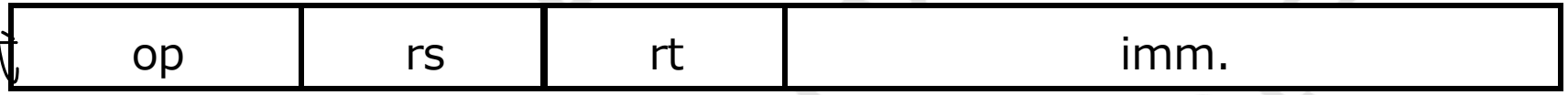
分支处理

- beq 指令，bne 指令是相对寻址

```
1000        beq $s0, $s1, L1
  ⋮           ...
1024      L1:
```

+24
（提前6个订单）

I 格式



- op：命令操作码
- rs：第一个源操作数 rt：第二个源操作数immediate：立即操作数
-

4
16_{10}
17_{10}
5

程序计数器
20 + 4

000100 10000 10001 0000000000000101

分支处理

- 程序计数器
  - PC: program counter
  - 保存当前执行指令所在地址的寄存器
    - 不在32种通用寄存器中
  - 由于MIPS指令是32位（4个字节），如果当前PC是2048，那么下一条指令的地址是……正常情况下，当执行一条指令时，PC=PC+4
  - 正在工作
  - 但是J指令来的时候，比PC低。
    用指定值替换 28 位

讲座内容

- 指令格式
  - 什么是R格式和I格式?
  - 指令与机器语言的对应关系
  - 使用数组的汇编代码 用于分支的汇编代码 无条件分支和 J 格式
  - 大/小比较指令
  - 寻址方式

无条件分支

- 无条件跳转

  j L1. 无条件跳转到标签为 L1 的指令

- J 格式

  - op: 指令操作码地址:
  - 绝对地址

    但是，实际地址的 1/4

    (表示指令的编号)

无条件分支

1000 `j L1`

⋮ `...`

1024 `L1:`

跳转到 1024

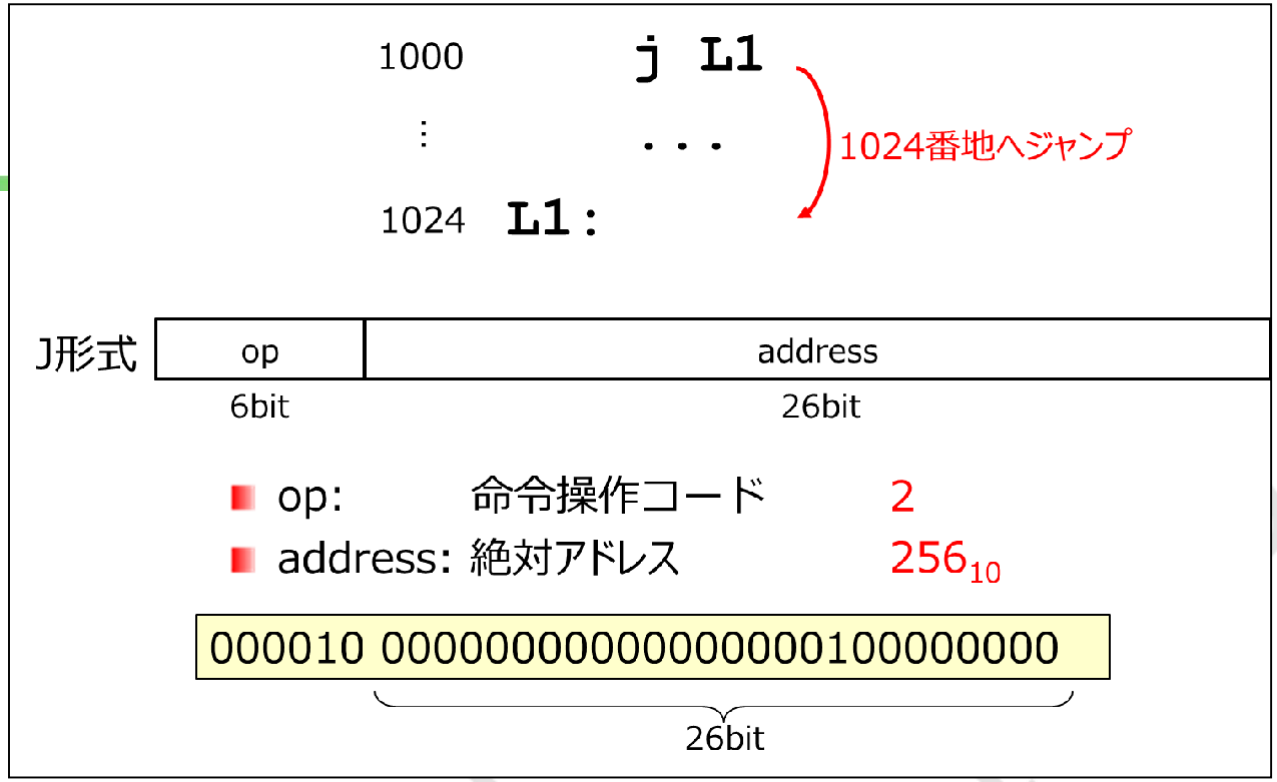
J 格式

- op: 命令操作码 2
- address: 绝对地址 256_{10}

000010 00000000000000100000000

26bit

无条件分支



PC 值的变化

0000 0000 0000 0000 0000 0011 1110 1000

0000 0000 0000 0000 0000 0100 0000 0000

26bit

高4位不变

低 2 位始终为 00（因为它乘以 4）

分支的汇编代码

(示例) if (i ==j) f=g=h; else f=g-h;

```
        bne $s3, $s4, Else
        add $s0, $s1, $s2
        j Exit
Else:   sub $s0, $s1, $s2
Exit:
```

f: $s0
g: $s1
h: $s2
i: $s3
j: $s4

确认问题

有一个while语句和对应的汇编代码如下所示。填写汇编代码中的空白。

(例) **while(i!=a[j]) { i=i+1; }**

```
Loop:  (1)   $t0, $s2, 2
       add   $t0, $t0, $s1
       lw    $t1, 0($t0)
       (2)   $s0, $t1, Exit
       (3)   $s0, $s0, (4)
       (5)   (6)
Exit:
```

i: $s0
a: $s1
j: $s2

确认问题

假设标签为 Loop 的语句的地址是 768_{10}。当这些指令用二进制表示时，红色显示的Exit和Loop对应的值是什么？分别以 16 位和 26 位二进制数回答。

```
Loop:   beq   $s0,   $s1,  Exit
        addi  $s0,   $s0,  1
        j     Loop
Exit:
```

确认问题

当以下指令用二进制表示时，它们是红色的。显示的 Loop 对应的值是多少？以 16 位二进制回答。

```
Loop:   add   $s0,  $s1,  $s2
        addi  $s0,  Ss0,  1
        beq   $s0,  $s1,  Loop
Next:
```

讲座内容

- 指令格式
  - 什么是R格式和I格式?
  - 指令与机器语言的对应关系
  - 使用数组的汇编代码 用于分支的汇编代码 无条件分支和 J 格式
  - 大/小比较指令
  - 寻址方式

大小对比

- < 对应的比较指令

```
slt $t0, $s0, $s1
```

. 当 $ s0 小于 $ s1 时
将 $t0 设置为 1。
否则设置为 0。

```
slti $t0, $s0, 5
```

. 当 $ s0 小于 5 时
将 $t0 设置为 1。
否则设置为 0。

```
slt (rd), (rs), (rt)
slti (rt), (rs), (imm.)
```

R 格式
I 格式

大小对比

(例) **while(i>=0) i=i-1;**

i: $s1

```
Loop:  slt   $t0,  $s1,  $zero
       bne   $t0,  $zero,  Exit
       addi  $s1,  $s1,  -1
       j     Loop

Exit:
```

大小对比

- 为什么 slt 和 slti 不直接分支
  - * 因为说明书太复杂
  - * 需要延长时钟周期时间等。
- >、>=等对应的指令是什么？
  - 准备为伪指令
    - 汇编器转换为具有相同功能的指令序列
  - blt (branch less than), bgt (- greater than), ble (branch less or equal), bge (- greater -)

讲座内容

- 指令格式
  - 什么是R格式和I格式?
  - 指令与机器语言的对应关系
  - 使用数组的汇编代码 用于分支的汇编代码 无条件分支和 J 格式
  - 大/小比较指令
  - 寻址方式

寻址方式

- 操作数解释因指令而异
  → 指令的操作目标，（在添加指令的情况下要添加的值）指令所需的地址
  （例如，j指令跳转目的地址）
  取决于指令
- 其中解释方法的寻址方式的类型称为寻址方式。
- - 寄存器寻址 (register addressing) 基本相对寻址 (base addressing)
  - 即值寻址 (immediate addressing) PC相对寻址 (PC-relative addressing)
  - 伪直接寻址 (pseudo direct addressing)

寄存器寻址

- 使用指定寄存器的内容作为操作数

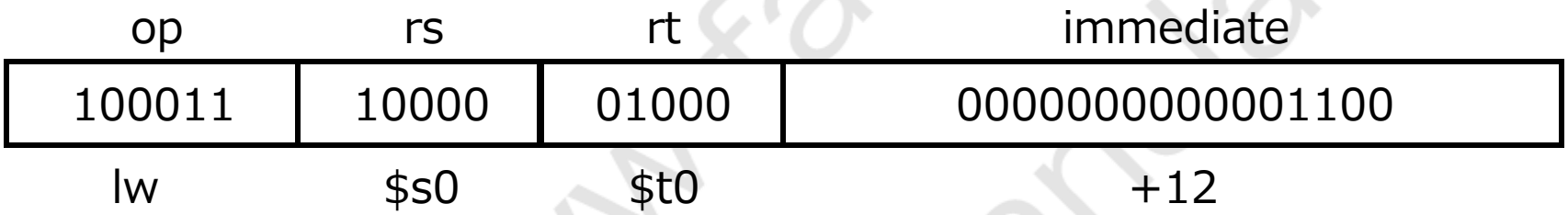
```
add $t0, $s0, $s1
```

op	rs	rt	rd	shamt	funct
000000	10000	10001	01000	00000	100000
add	$s0	$s1	$t0		add

基本相对寻址

- 指定寄存器的内容+常量
  使用内存地址的内容

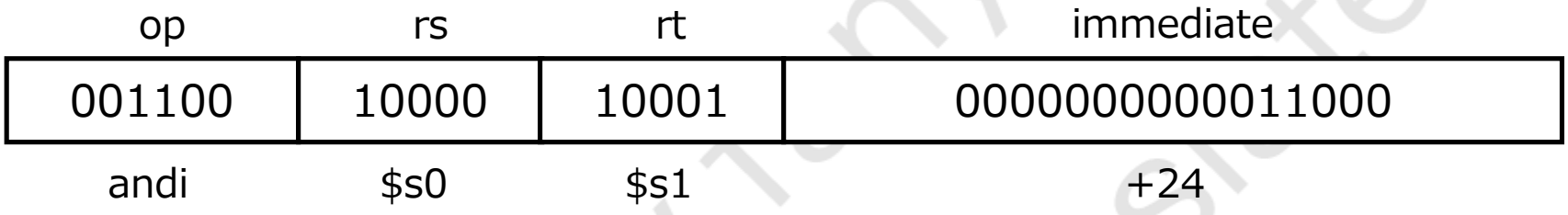
```
lw $t0, 12($s0)
```



立即寻址

- 使用指定的常量作为操作数

```
andi $s1, $s0, 24
```



PC 相对寻址

- 表示(PC+4)+(指定常数×4)的地址

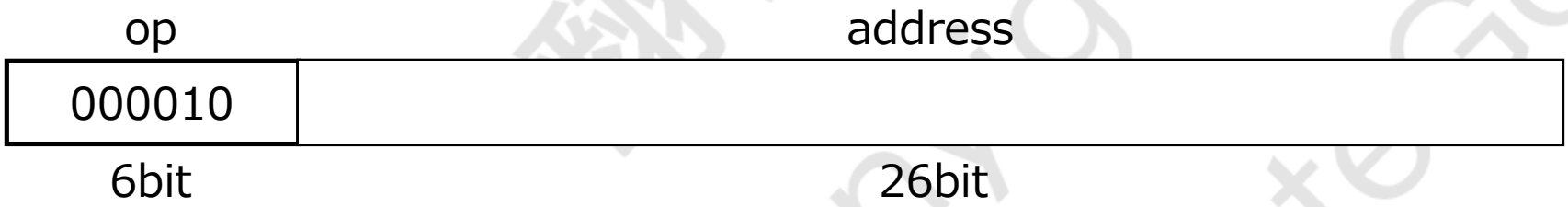
```
bne $t0, $s0, Label
```

op	rs	rt	address
000101	010000	10000	0000000000001100
bne	$t0	$s0	+12

伪直接寻址

- 表示连接PC高4位和指定常数x 4的地址。

j Label



PC = 00101100000000000100111100010000

当，上述指令的跳转目标为

00100000000000010011100010000000

32 位立即数操作数

- 指令长度为 32 位
  - 无法处理 32 位操作数，但：

0000 0000 0011 1101 0000 1001 0000 0000

高 16 位 = 6110　　低 16 位 = 230410

- 分上下两部分，用2条指令阅读

lui $s0, 61

. 到 $ s0 的高 16 位
读取常量

0000 0000 0011 1101 0000 0000 0000 0000

(load upper immediate)

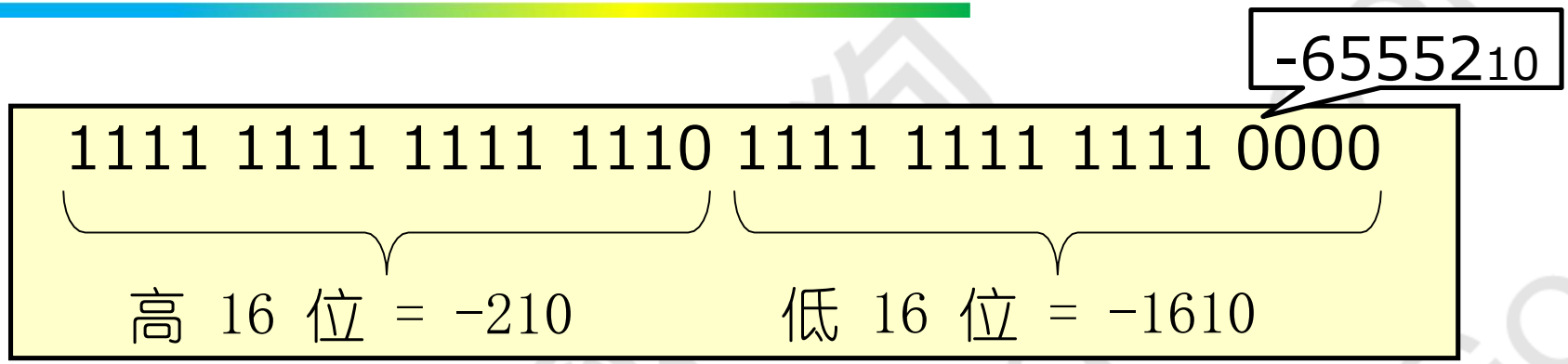
ori $s0, $s0, 2304

. . . 或操作

(or immediate)

0000 0000 0011 1101 0000 1001 0000 0000

32 位立即数操作数



lui $s0, -2

… 将常量读入 $ s0 的高 16 位

(load upper immediate)

1111 1111 1111 1110 0000 0000 0000 0000

ori $s0, $s0, -16

… 或操作

(or immediate)

1111 1111 1111 1110 1111 1111 1111 0000

确认问题

- 回答每个句子描述的寻址模式。
  - (1) 指定寄存器的内容+常量
    使用内存地址的内容
  - (2) 表示(PC+4)+(指定常数×4)的地址
  - (3) 使用指定寄存器的内容作为操作数
  - (4) 表示连接PC高4位和指定常数x 4的地址。
  - (5) 使用指定的常量作为操作数

参考

- 计算机配置和设计由大卫●帕特森，约翰●亨内塞，成田三崎，日经BP
- Shigeru Yamashita“计算机组成理论1”讲义