计算机组成理论第5期

-指令集架构(2)-

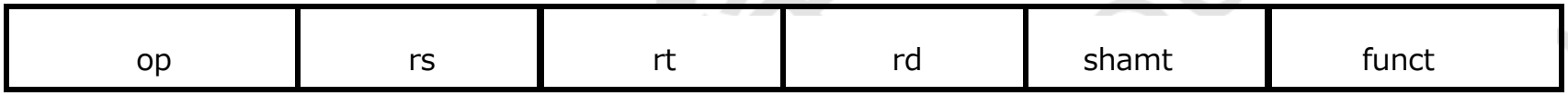
大连理工大学立命馆大学国际信息软件学部大森孝之

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

MIPS指令格式

R格式

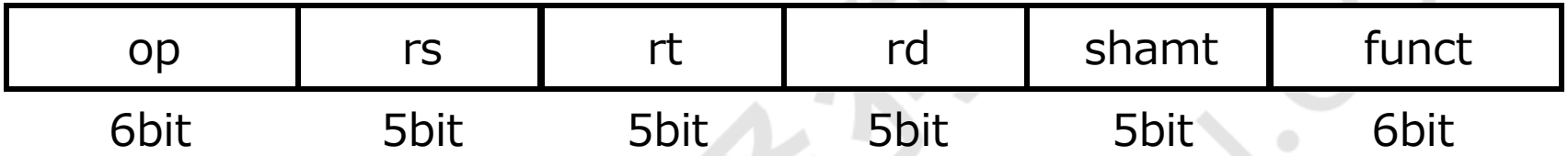


我格式化

J格式

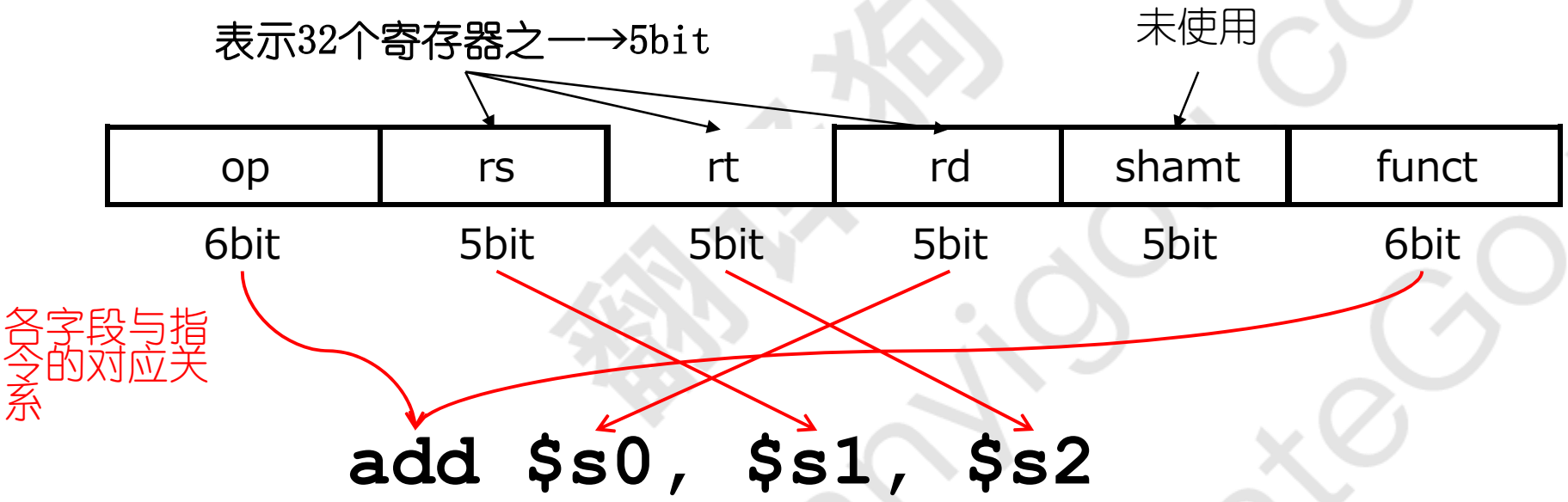
6bit 26bit

R格式



- op: 指令操作码（opcode）rs：第一个源操作数 rt：第二个源操作数
- rd: 目标操作数
- shamt：移位量函数： 功能码
-

R 格式（用于添加指令）



- op: 指令操作码（opcode）rs：第一个源操作数 rt： 第二个源操作数
- rd: 目标操作数 (destination operand)
- shamt：移位量
- funct: 功能码

立即数

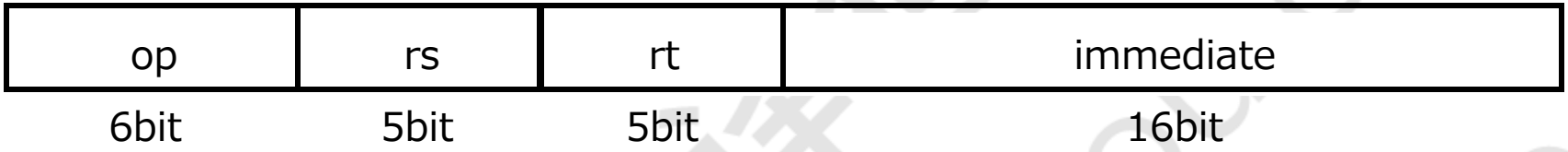
```
add a, b, c      # a = b+c
```

```
add a, b, 4      # a = b+4 ???
```

```
addi a, b, 4     # a = b+4
```

- 使用addi指令添加常量addi：addimmediate
- 即时：
  直接写入用于操作的操作数中的值

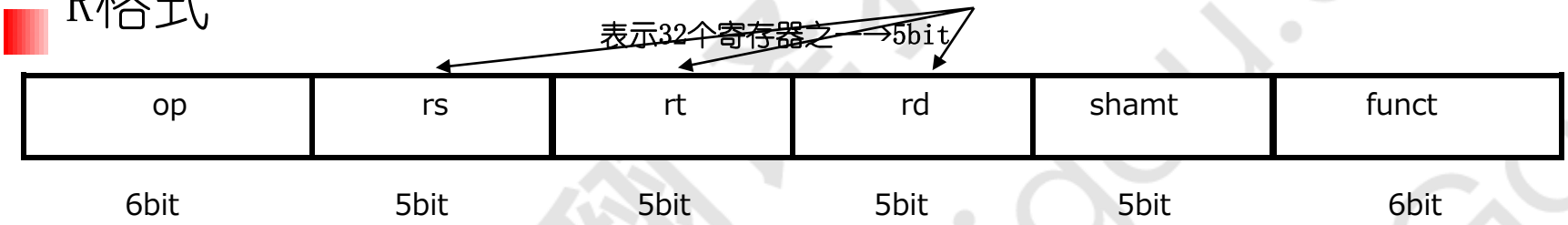
我格式化



- op: 指令操作码（opcode）rs：第一个源操作数
- rt: 第二个源操作数
- 立即数：立即数操作数

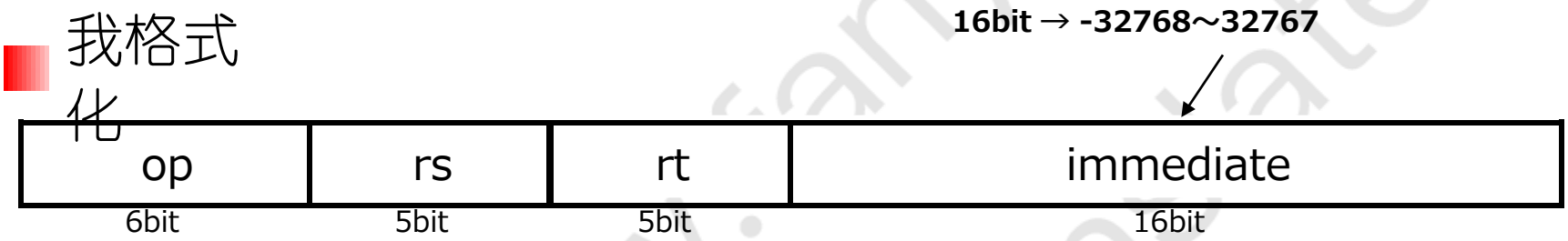
立即数

R格式




```
add a, b, c          # a = b+c
```

我格式化




```
addi a, b, 4         # a = b+4

lw $s1, 8($s2) # 8 is immediate
sw $s1, 8($s2) # 8 is immediate
```

unsigned immediate

```
addi    a, b, -4      # a = b-4
addiu   a, b, 4       # a = b+4
```

- addiu 指令添加一个无符号整数
  - addi 可以处理的常数是 -32768 到 32767。
    addiu 可以处理的常数是 0 到 65535。

确认问题

- 以下每个指令都是 R 格式或 I 格式。
  - add
  - addi
  - addiu
  - lw sw
- I in I form 是什么意思？
- R 风格的指令中存在多少个操作数
- 回答 I-format 指令可以容纳的立即位宽。

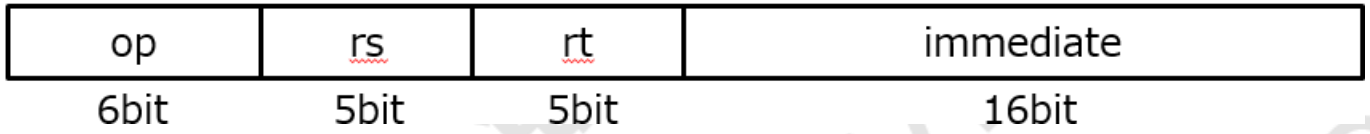
讲座内容

- 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

命令与机器语言的对

- R形式

- I形式

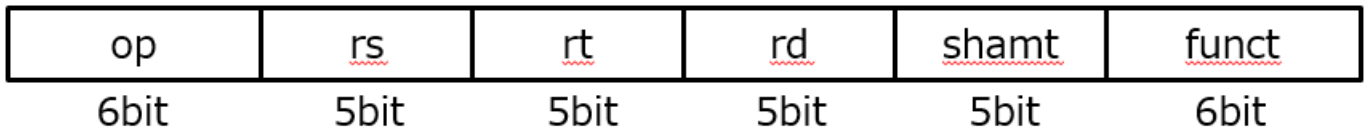


指令	opcode/funct	指令	opcode/funct
add	$0/20_{16}$	sub	$0/22_{16}$
addi	8_{16}	lw	23_{16}
addiu	9_{16}	sw	$2B_{16}$

参见“MIPS 参考数据”(无需记住考试编号)

命令与机器语言的对

- R形式



- I形式

レジスタの種類

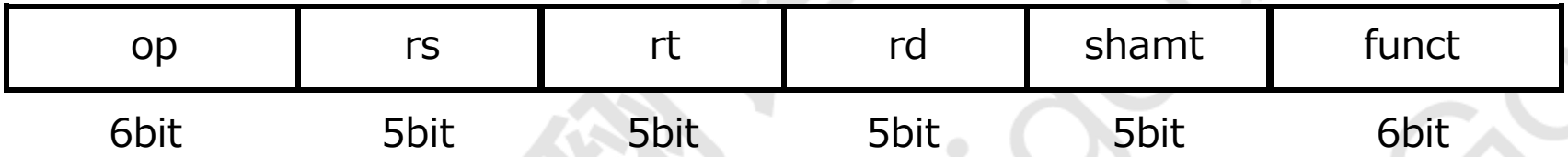
$zero	0	常にゼロ
$at	1	アセンブラが一時的に使用
$v0-v1	2-3	戻り値用
$a0-a3	4-7	引数用
$t0-t9	8-15, 24-25	一時レジスタ (一時変数用)
$s0-s7	16-23	退避レジスタ (変数用)

命令与机器语言的对

（例子）

op rd rs rt

add $s0, $s1, $s2



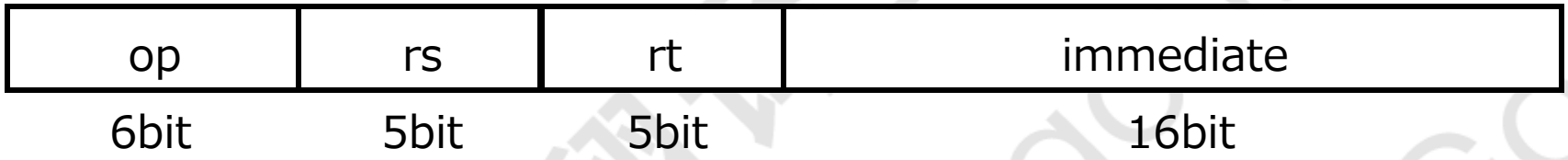
- op: 命令操作码 — 0
- rs: 第一个源操作数 — 17_{10}
- rt: 第二个源操作数 — 18_{10}
- rd: 目标操作数 — 16_{10}
- shamt: 班次金额 — 0
- funct: 功能码 — 32_{10}

000000 10001 10010 10000 00000 100000

命令与机器语言的对

op rt rs immediate

(例子) **addi $s0, $s1, 5**



- 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
-

8

17_{10}

16_{10}

5

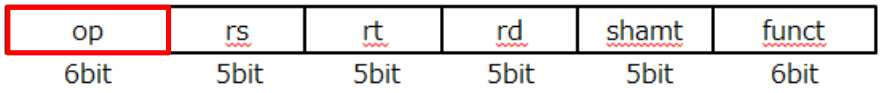
001000 10001 10000 0000000000000101

指令格式的区分

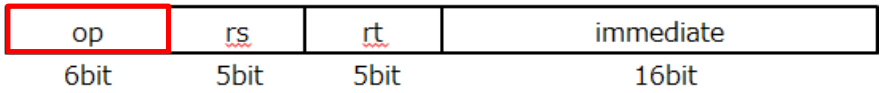
- 计算机如何区分指令格式？
  - 每条指令的前 6 位（操作码）

看了就可以看到格式（比如0是R格式，35_{10}（lw）是I格式）

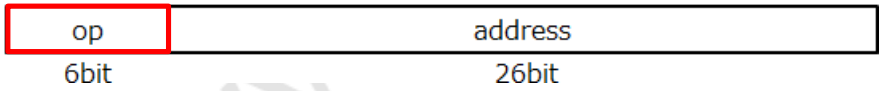
- R形式



- I形式



- J形式



换档操作

sll $s0, $s1, 4 . 将 $s1 的值向左移动 4 位逻辑移位并存储在 $ s0 中 (逻辑左移)

srl $s0, $s1, 4 . 将 $s1 的值向右移动 4 位逻辑移位并存储在 $ s0 中 (逻辑右移)

sll (rd), (rt), (shamt) op:0 funct:0

srl (rd), (rt), (shamt) op:0 funct:2

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

使用数组的汇编代码

(例子) `a[300] = b + a[200]`

a: $t1
b: $s1

```
lw    $t0,  800($t1)
add   $t0,  $s1, $t0
sw    $t0,  1200($t1)
```

- 内存地址偏移是数组元素个数的4倍

使用数组的汇编代码

(例子)

```
a[i] = b + a[i]
```

a: $t1
b: $s1
i: $s2

```
sll   $t2,  $s2,  2
add   $t2,  $t2,  $t1
lw    $t0,  0($t2)
add   $t0,  $s1, $t0
sw    $t0,  0($t2)
```

使用数组的汇编代码

使用数组的汇编代码

使用移位操作计算4x

确认问题

a[50] = b - a[i]

a: $t1
b: $s1
i: $s2

```
(1)  $t2, $s2, 2
(2)  $t2, $t2, $t1
(3)  $t0, 0($t2)
(4)  $t0, $s1, $t0
sw   $t0, (5)((6))
```

… 计算 4 次 i

… 将 a［i］的地址存储在 $ t2 中

… 将 a［i］的值加载到 $ t0 中

… $t0 = $s1 - $t0

. 将 $ t0 的值存储在［50］中

参考

- Computer Configuration and Design 5th Edition by David A. Patterson, John L. Hennessy, 成田光明翻译, Nikkei BP
- Shigeru Yamashita“计算机组成理论1”讲义