

## 本节主题



# MIPS指令简介

北京大学·慕课  
计算机组成  
制作人：陆俊林



# MIPS指令

**MIPS** Reference Data[illegible][illegible]

## BASIC INSTRUCTION FORMATS

<b>W</b>	opcode	rs	rt	rd	shamt	func
	31	26-30	21-25	16-20	11-15	6-10
<b>V</b>	opcode	rs	rt	immediates		
	31	26-30	21-25	0-10		
<b>J</b>	opcode	address				
	31	0-30				

Copyright 2009 by Elsevier Inc. All rights reserved. From Patterson and Hammer, *Compass Organization and Design*, 4th ed.

## ARITHMETIC CORE INSTRUCTION SET

[illegible]

### FLOATING-POINT INSTRUCTION FORM

P0	quadr	for	8	5	12	100
P1	quadr	for	8	median		

## PSEUDONSTRUCTION SET

STATE	TRANSITION	OPERATION
Branch Less Than	$b < a$	$if(b < a) goto PC = Label$
Branch Greater Than	$b > a$	$if(b > a) goto PC = Label$
Branch Less Than or Equal	$b \leq a$	$if(b \leq a) goto PC = Label$
Branch Greater Than or Equal	$b \geq a$	$if(b \geq a) goto PC = Label$
Load Immediate	$r \leftarrow i$	$goto r = immediate$
Store	$a[r] \leftarrow i$	$goto r = i$

REGISTER NAME	NUMBER	USE	CALL CONVENTION
---------------	--------	-----	-----------------

NAME	NUMBER	LINE	PREFERRED ACTION
Var0	0	The Constant Value 0	OK
Var1	1	Assembly Temporary	Yes
Int0-Var1	2-3	Value for Function Results and Expression Evaluation	Yes
Var2-Var3	4-7	Arguments	Yes
Var4-Var5	8-10	Arguments	Yes
Var6-Var7	11-13	Local Temporaries	Yes
Var8-Var9	14-16	Temporaries	Yes
Var10-Var11	17-17	Reserved for OS Kernel	Yes
Var12	18	Global Pointer	Yes
Var13	19	Stack Pointer	Yes
Var14	20	Frame Pointer	Yes
Var15	21	Return Address	Yes

## OPCODES, BASE CONVERSION, ASCII SYMBOLS

[illegible]

```

(1) speedup(1:2) := 2
(2) speedup(1:3) := vbest(Tbest) if best(T:2) = vbest(Tbest) / 2 + 1 (avg)
    if best(T:2) = vbest(Tbest) / 2 + 2 (better)

```

SEE 754 FLOATING-POINT STANDARD

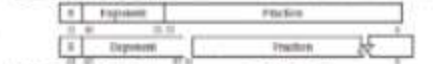
$$f: \mathbb{R}^n \rightarrow \mathbb{R} \text{ (function)} \rightarrow \mathbb{R}^{\text{dimension: } n}$$

where Single Precision Step = 128.

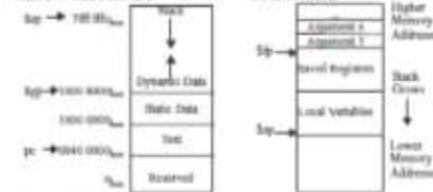
Double Precision Map - 1922

### GLL Single Premium and

### Double Precision Formats:



### MEMORY ALLOCATION



#### DATA ALIGNMENT

Double Word							
Word				Word			
Halfword		Halfword		Halfword		Halfword	
byte	byte	byte	byte	byte	byte	byte	byte

value of these four significant bits of byte address (byte index)

## EXCEPTION CONTROL REGISTERS: CAUSE AND STATUS

W		Interrupt		Exception	
D		Mask		Code	
W					
		Pending		U	S
		Interrupt		M	I

HD = Head Injury; LAR = Low Airway Resistance; SI = Inspiration Level; R = Rebreathing Valve

### EXCEPTION CODES

Number	Form	Amount of exception	Number	Form	Amount of exception
3	III	Maximum 1000000	9	III	Maximum 1000000
4	Ad III	Admission Travel Exception	10	III	Maximum 1000000
5	Ad III	Admission Travel Exception	11	III	Maximum 1000000
6	III	Maximum 1000000	12	III	Maximum 1000000
7	III	Maximum 1000000	13	III	Maximum 1000000
8	III	Maximum 1000000	14	III	Maximum 1000000

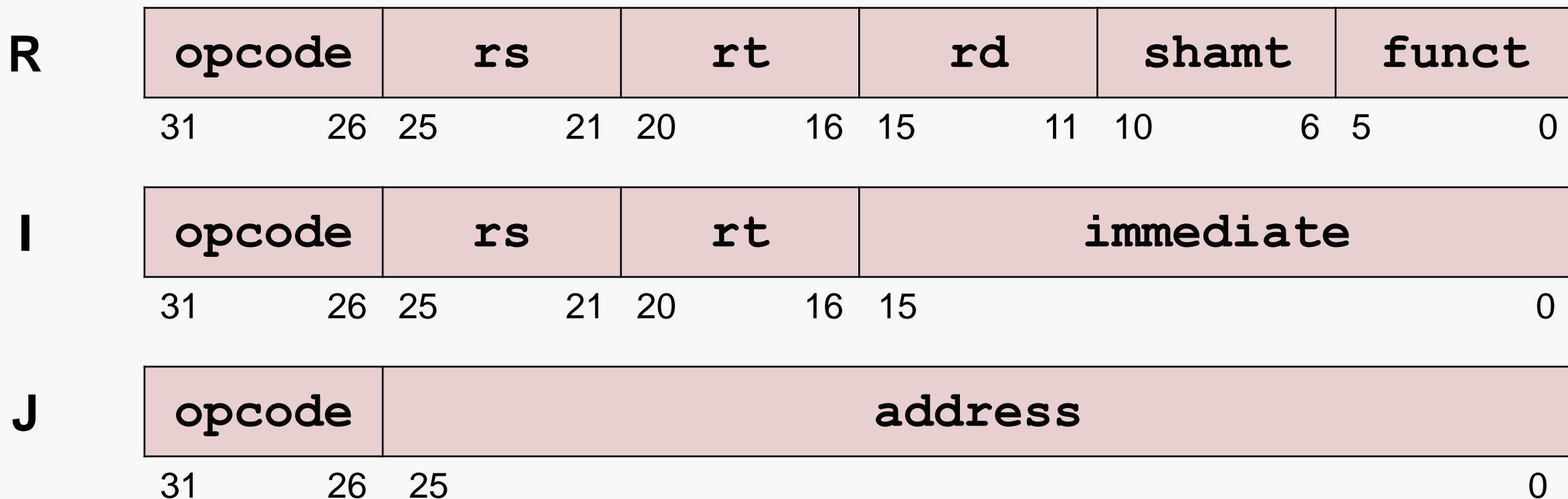
SIZE PREFIXES ( $10^3$  for Disk, Communication;  $2^3$  for Memory)

P202		P201		P202		P202	
SIZE	FOR	SIZE	FOR	SIZE	FOR	SIZE	FOR
$10^{1-2}$	Kilo	$10^{1-2}$	Mega	$10^{1-2}$	Mega	$10^{1-2}$	Mega
$10^{2-3}$	Mega	$10^{2-3}$	Giga	$10^{2-3}$	Giga	$10^{2-3}$	Giga
$10^{3-4}$	Giga	$10^{3-4}$	Tera	$10^{3-4}$	Tera	$10^{3-4}$	Tera
$10^{4-5}$	Tera	$10^{4-5}$	Peta	$10^{4-5}$	Peta	$10^{4-5}$	Peta

The expected rate each profile is paid the total effort, except  $\mu$  is used for the search.

# MIPS指令的基本格式

- ▶ R : Register , 寄存器
- ▶ I : Immediate , 立即数
- ▶ J : Jump , 无条件转移



# 不同维度的指令分类（示例）

运算 指令	<div>add rd,rs,rt sll rd,rt,shamt</div>	<div>addi rt,rs,imm slti rt,rs,imm</div>	/
访存 指令	/	<div>lw rt,imm(rs) sw rt,imm(rs)</div>	/
分支 指令	jr rs	beq rs,rt,imm	j addr
	R型指令	I型指令	J型指令



# R型指令的格式（1）

- 🔍 R型指令格式包含6个域
  - 2个6-bit域，可表示0~63的数
  - 4个5-bit域，可表示0~31的数

思考：  
为什么不将opcode域和  
funct域合并成一个12-bit  
的域？

用于指定指令的类型。对于所有R型指令，该域的值均为0

opcode

6-bit

5-bit

5-bit

与opcode域组合，精确地指定指令的类型

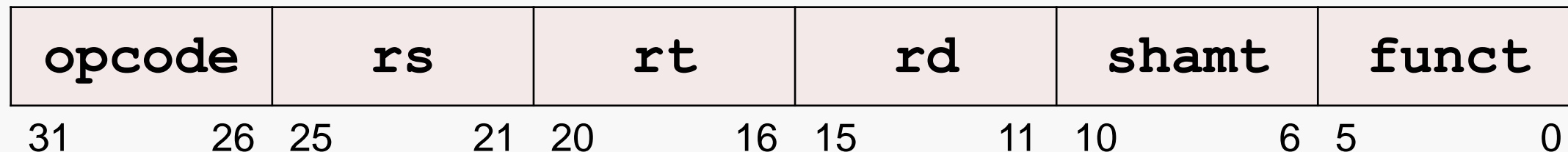
funct

5-bit

5-bit

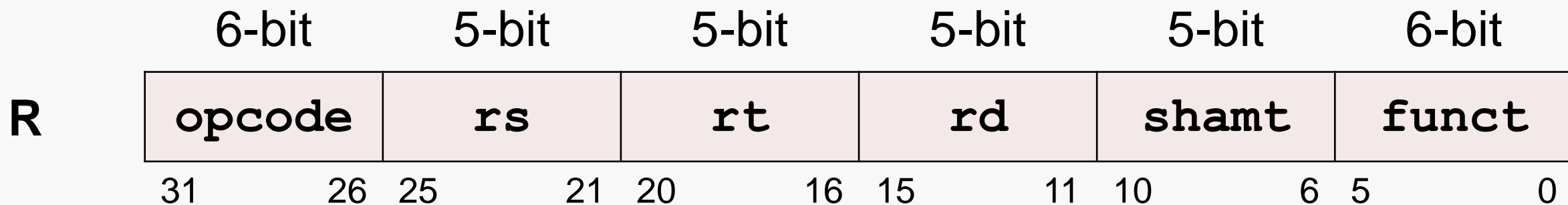
6-bit

R



# R型指令的格式（2）

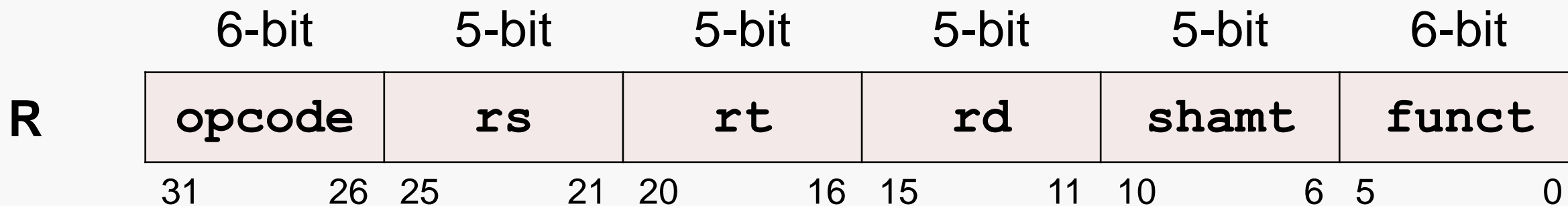
- rs Source Register
  - 通常用于指定第一个源操作数所在的寄存器编号
- rt Target Register
  - 通常用于指定第二个源操作数所在的寄存器编号
- rd Destination Register
  - 通常用于指定目的操作数（保存运算结果）的寄存器编号
- 5-bit的域可表示0~31，对应32个通用寄存器



# R型指令的格式（3）

shamt **shift amount**

- 用于指定移位指令进行移位操作的位数
- 5-bit的域可表示0~31，对于32-bit数，更多移位没有实际意义
- 对于非移位指令，该域设为0





# R型指令的编码示例

▶ **add      \$8 , \$9 , \$10**

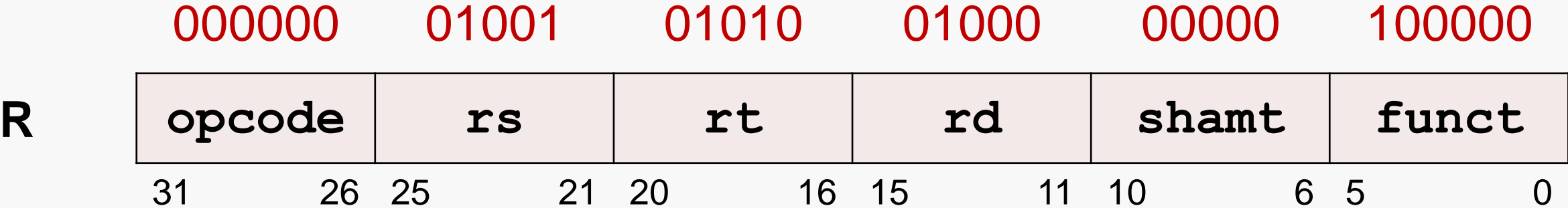
◦ 查指令编码表得到：

opcode = 0 , funct = 32 , shamt = 0 （非移位指令）

◦ 根据指令操作数得到：

rd = 8 （目的操作数） , rs = 9 （第一个源操作数）

rt = 10 （第二个源操作数）



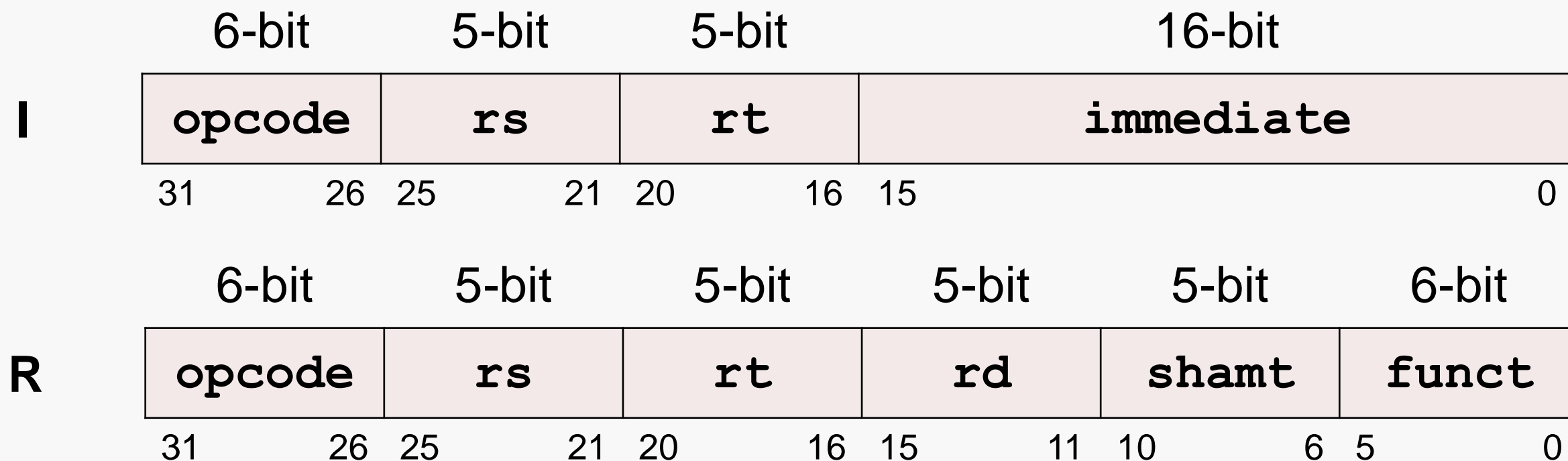


# 不同维度的指令分类（示例）

运算 指令	<code>add rd,rs,rt</code> <code>sll rd,rt,shamt</code>	<code>addi rt,rs,imm</code> <code>slti rt,rs,imm</code>	/
访存 指令	/	<code>lw rt,imm(rs)</code> <code>sw rt,imm(rs)</code>	/
分支 指令	<code>jr rs</code>	<code>beq rs,rt,imm</code>	<code>j addr</code>
	R型指令	I型指令	J型指令

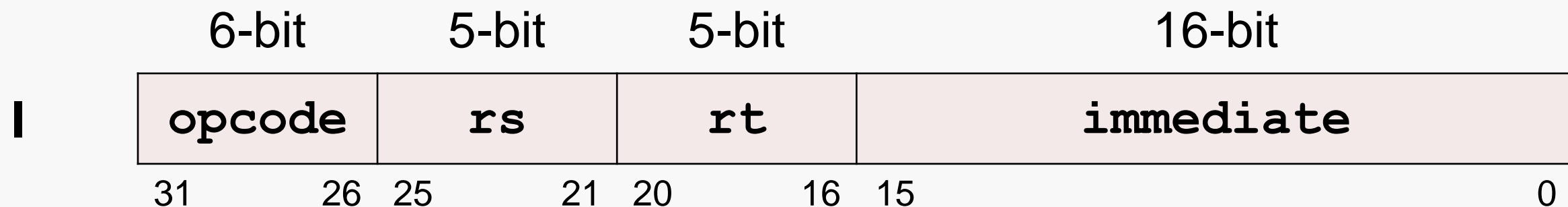
# I型指令的格式 (1)

- ▶ R型指令只有一个5-bit域表示立即数，范围为0~31
- ▶ 常用的立即数远大于这个范围，因此需要新的指令格式
- ▶ I型指令的大部分域与R型指令相同



# I型指令的格式 ( 2 )

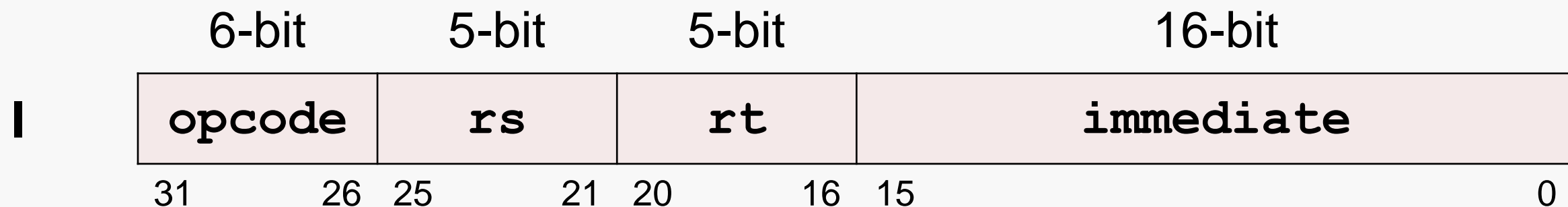
- ▶ opcode
  - 用于指定指令的操作类型 ( 但没有 `funct` 域 )
- ▶ `rs` Source Register
  - 指定第一个源操作数所在的寄存器编号
- ▶ `rt` Target Register
  - 指定用于目的操作数 ( 保存运算结果 ) 的寄存器编号
  - 对于某些指令, 指定第二个源操作数所在的寄存器编号



# I型指令的格式 ( 3 )

## ④ immediate

- 16-bit的立即数，可以表示 $2^{16}$ 个不同数值
- 对于访存指令，如 `lw rt, imm(rs)`  
通常可以满足访存地址偏移量的需求 ( -32768~+32767 )
- 对于运算指令，如 `addi rt, rs, imm`  
无法满足全部需求，但大多数时候可以满足需求



# I型指令的编码示例

🔍 **addi \$21, \$22, -50** #  $\$21 = \$22 + (-50)$

◦ 查指令编码表得到：

opcode = 8

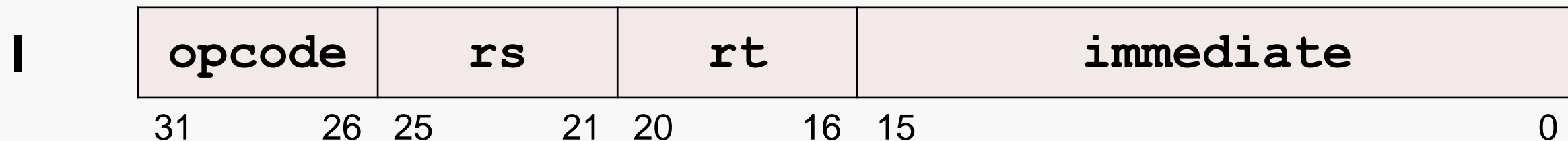
◦ 分析指令得到：

rs = 22 (源操作数寄存器编号)

rt = 21 (目的操作数寄存器编号)

immediate = -50 (立即数)

001000    10110    10101    1111 1111 1100 1110





# 不同维度的指令分类（示例）

运算指令	add rd,rs,rt sll rd,rt,shamt	addi rt,rs,imm slti rt,rs,imm	/
访存指令	/	lw rt,imm(rs) sw rt,imm(rs)	/
分支指令	jr rs	beq rs,rt,imm	j addr
	R型指令	I型指令	J型指令

# 分支指令的分类



## ▶ Branch

- 分支：改变控制流

## ▶ Conditional Branch

- 条件分支：根据比较的结果改变控制流
- 两条指令：branch *if* equal (beq) ; branch *if not* equal (bne)

## ▶ Unconditional Branch

- 非条件分支：无条件地改变控制流
- 一条指令：jump (j)

# 条件分支指令（I型）

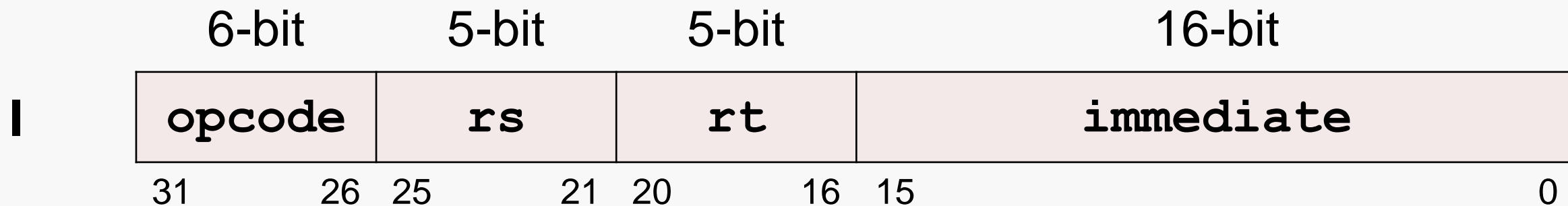


## 条件分支

- `beq rs,rt,imm` # opcode=4
- `bne rs,rt,imm` # opcode=5

## 格式：**beq reg1,reg2,L1**

`if (value in reg1) == (value in reg2)`  
`goto L1`





# 条件分支指令的示例



```
if (i==j)
    f=g+h;
else
    f=g-h;
```

C语言代码

```
beq $s3,$s4,True      # branch i==j
sub $s0,$s1,$s2        # f=g-h(false)
j  Fin                 # goto Fin

True: add $s0,$s1,$s2 # f=g+h (true)
Fin:  ...
```

MIPS汇编语言代码

# 条件分支指令的目标地址范围

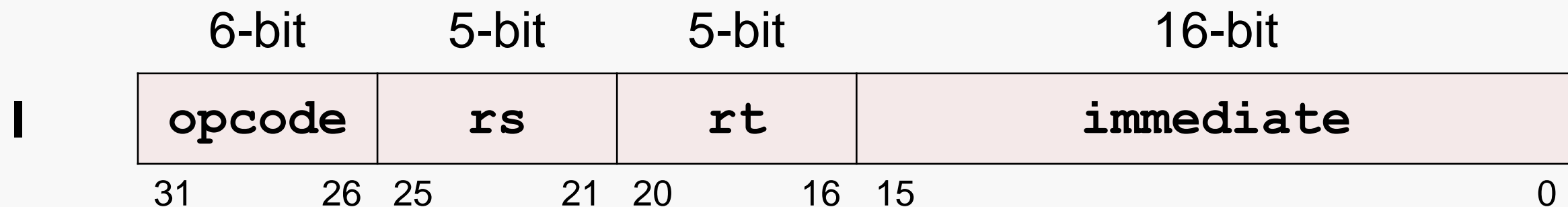


## 如何充分发挥16-bit的作用？

- 以当前PC为基准，16-bit位移量可以表示 $\pm 2^{15}$  bytes
- MIPS的指令长度固定为32-bit ( word )
- 16-bit位移量可以表示  $\pm 2^{15}$  words =  $\pm 2^{17}$  bytes (  $\pm 128\text{KB}$  )

## 目标地址计算方法：

- 分支条件不成立， $PC = PC + 4 = \text{next instruction}$
- 分支条件成立， $PC = (PC+4) + (\text{immediate} * 4)$



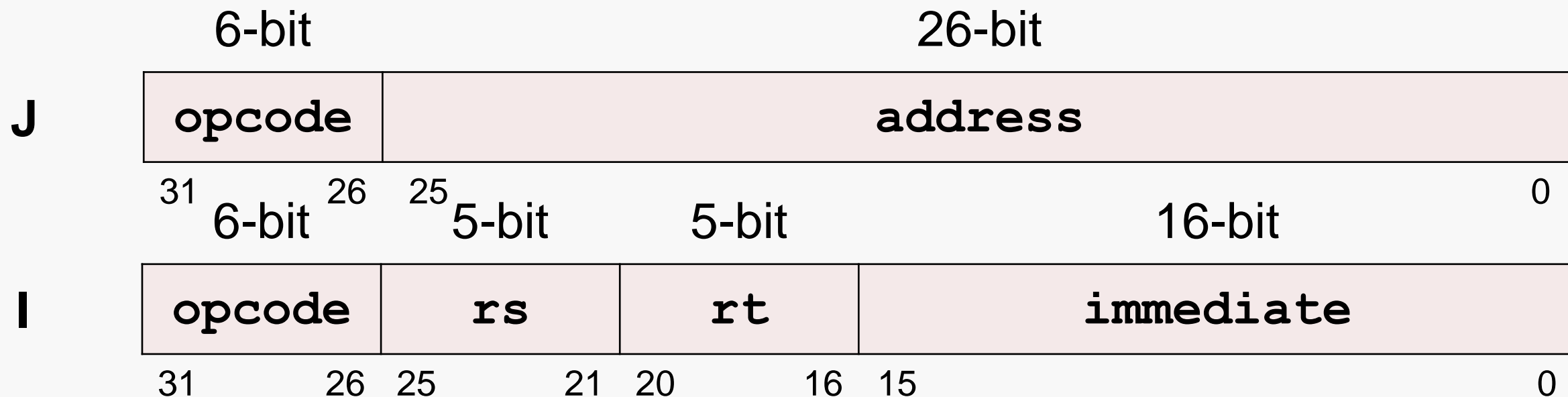
# 非条件分支指令（J型）

## 在不需要条件判断的情况下，如何扩大目标地址范围

- 理想情况，直接使用32-bit地址
- 冲突：MIPS的指令长度固定为32-bit，opcode占用了6-bit

## 目标地址计算方法：

- $\text{New PC} = \{ (\text{PC}+4) [31..28], \text{address}, 00 \}$



# 两种分支指令示例

🔍 假设变量和寄存器的对应关系如下

$f \rightarrow \$s0$

$g \rightarrow \$s1$

$h \rightarrow \$s2$

$i \rightarrow \$s3$

$j \rightarrow \$s4$

if (i == j)

    f = g + h;

else

    f = g - h;

bne \$s3, \$s4, Else

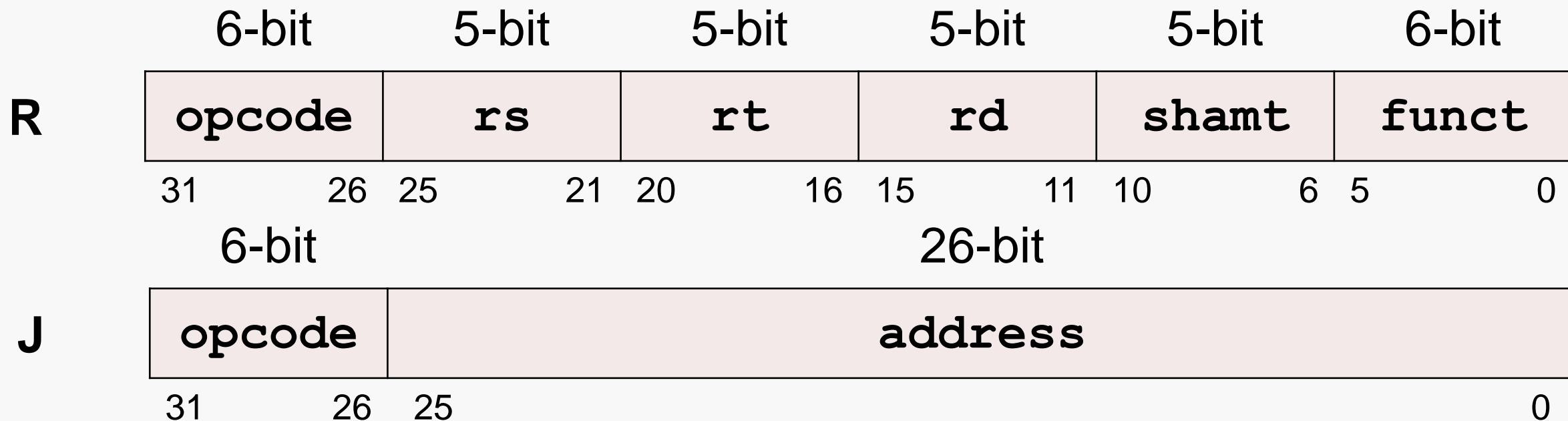
add \$s0, \$s1, \$s2

j Exit

Else: sub \$s0, \$s1, \$s2

Exit:

- IB )



# MIPS指令

**MIPS** Reference Data

CORE INSTRUCTION SET		FOR	OPERATION (in Verilog)	OPCODE / FUNCTION
NAME, MINIMUM	CODE	FOR	OPERATION (in Verilog)	OPCODE / FUNCTION
Add	0000	0	$R[dst] \leftarrow R[src] + R[imm]$	(1) 8/20
Add Immediate	0001	0	$R[dst] \leftarrow R[src] + \text{sign}(R[imm])$	(1,2) 8/20
Add Imm. Unsigned	0010	0	$R[dst] \leftarrow R[src] + \text{sign}(R[imm])$	(2) 8/20
Add Unsigned	0011	0	$R[dst] \leftarrow R[src] + R[dst]$	6/22
And	0100	0	$R[dst] \leftarrow R[src] \& R[dst]$	6/22
And Immediate	0101	1	$R[dst] \leftarrow R[src] \& \text{sign}(R[imm])$	(3) 8/20
Branch On Equal	0110	1	$PC \leftarrow PC + 4 - \text{branch\_addr}$	(4) 8/20
Branch On Not Equal	0111	1	$PC \leftarrow PC + 4 - \text{branch\_addr}$	(4) 8/20
Jump	1	0	$PC \leftarrow \text{jump\_addr}$	(5) 8/20
Jump And Link	1	1	$R[31] \leftarrow PC; PC \leftarrow \text{jump\_addr}$	(5) 8/20
Jump Register	1	0	$PC \leftarrow R[imm]$	6/22
Load Byte Unsigned	1000	1	$R[dst] \leftarrow \{24'MEM[R[src]]\}$ $\leftarrow \text{sign}(R[dst], \text{imm})$	(6) 24/20
Load Halfword Unsigned	1001	1	$R[dst] \leftarrow \{16'MEM[R[src]]\}$ $\leftarrow \text{sign}(R[dst], \text{imm})$	(6) 22/20
Load Link	1	1	$R[dst] \leftarrow R[src] \oplus \text{sign}(R[dst], \text{imm})$	(2,7) 20/20
Load Upper Imm.	1010	1	$R[dst] \leftarrow \{imm, 16'0\}$	8/22
Load Word	1011	1	$R[dst] \leftarrow R[src] \oplus \text{sign}(R[dst], \text{imm})$	(2) 20/20
Nor	1100	0	$R[dst] \leftarrow \sim (R[src] \& R[dst])$	6/22
Or	1101	0	$R[dst] \leftarrow R[src]   R[dst]$	6/22
Or Immediate	1110	1	$R[dst] \leftarrow R[src]   \text{sign}(R[imm])$	(6) 8/20
Set Less Than	1111	0	$R[dst] \leftarrow (R[src] > R[dst]) ? 1 : 0$	6/22
Set Less Than Imm.	1111	1	$R[dst] \leftarrow (R[src] > \text{sign}(R[imm]) ? 1 : 0)$	8/20
Set Less Than Unsigned	1111	1	$R[dst] \leftarrow (R[src] > R[imm]) ? 1 : 0$	(3,4) 8/20
Set Less Than Using	1111	0	$R[dst] \leftarrow (R[src] > R[dst]) ? 1 : 0$	(4) 8/20
Shift Left Logical	1111	0	$R[dst] \leftarrow R[src] \ll \text{shamt}$	6/20
Shift Right Logical	1111	0	$R[dst] \leftarrow R[src] \gg \text{shamt}$	6/20
Store Byte	1000	1	$MEM[R[src] + \text{sign}(R[dst], \text{imm})] \leftarrow R[imm]$	(2) 20/20
Store Conditional	1001	1	$MEM[R[src] + \text{sign}(R[dst], \text{imm})] \leftarrow R[imm]$ $\text{if } (R[dst] \> \text{zero}) ? 1 : 0$	(3,7) 20/20
Store Halfword	1010	1	$MEM[R[src] + \text{sign}(R[dst], \text{imm})] \leftarrow R[imm]$ $\text{if } (R[dst] \> 0)$	(3) 20/20
Store Word	1011	1	$MEM[R[src] + \text{sign}(R[dst], \text{imm})] \leftarrow R[imm]$	(3) 20/20
Subsume	1100	0	$R[dst] \leftarrow R[src] \& R[dst]$	(3) 6/22
Subsume Unsigned	1101	0	$R[dst] \leftarrow R[src] \& R[dst]$	6/22

- (1) `key` cannot overflow integer
- (2) `next` cannot = 16 (underflow) or 17 (overflow)
- (3) `next` cannot = 18 (0) or immediate
- (4) `branch` can't = 16 (immediate) or 17 (immediate, 3 bits)
- (5) `jump` can't = 16 (0) or 17 (immediate, 3 bits)
- (6) operands combined unequal numbers (0, 2) or comp.
- (7) `operands` combined pair, 4 bits = 1 if pair, none, 0 if not any

## BASIC INSTRUCTION FORMATS

<b>W</b>	opcode	rs	rt	rd	shamt	func
	31	26-30	21-25	16-20	11-15	6-10
<b>V</b>	opcode	rs	rt	immediate		
	31	26-30	21-25	0-31		
<b>J</b>	opcode	address				
	31	0-31				

Copyright 2009 by Elsevier Inc. All rights reserved. From Patterson and Hammer, *Compass Organization and Design*, 4th ed.

## ARITHMETIC CORE INSTRUCTION SET

[illegible]

### FLOATING-POINT INSTRUCTION FORM

P0	quadr	for	8	5	12	100
P1	quadr	for	8	median		

## PSEUDONSTRUCTION SET

STATE	TRANSITION	OPERATION
Branch Less Than	$b < a$	$if(b < a) goto PC = Label$
Branch Greater Than	$b > a$	$if(b > a) goto PC = Label$
Branch Less Than or Equal	$b \leq a$	$if(b \leq a) goto PC = Label$
Branch Greater Than or Equal	$b \geq a$	$if(b \geq a) goto PC = Label$
Load Immediate	$r \leftarrow i$	$reg[r] \leftarrow immediate$
Store	$a[r] \leftarrow r$	$store(r, r)$

REGISTER NAME	NUMBER	USE	CALL CONVENTION
---------------	--------	-----	-----------------

NAME	NUMBER	LINE	PREFERRED ACTION
Var0	0	The Constant Value 0	OK
Var1	1	Assembly Temporary	OK
Int0-Var1	2-3	Value for Function Results and Expression Evaluation	OK
Var2-Var3	4-7	Arguments	OK
Var4-Var5	8-10	Arguments	OK
Var6-Var7	11-13	Local Temporaries	OK
Var8-Var9	14-16	Temporaries	OK
Var10-Var11	17-19	Reserved for OS Kernel	OK
Var12	20	Global Pointer	OK
Var13	21	Stack Pointer	OK
Var14	22	Frame Pointer	OK
Var15	23	Return Address	OK

## OPCODES, BASE CONVERSION, ASCII SYMBOLS

APPS specification (1st 34)	APPS function (35-36)	APPS function (37-38)	January 1960 (39-40)	February 1960 (41-42)	March 1960 (43-44)	April 1960 (45-46)	May 1960 (47-48)	June 1960 (49-50)	July 1960 (51-52)	August 1960 (53-54)	September 1960 (55-56)	October 1960 (57-58)	November 1960 (59-60)	December 1960 (61-62)
111	4-111	4-111	100-0000	1	0	0000	24	40	40	40	40	40	40	40
112	4-112	4-112	100-0000	1	0	0000	24	40	40	40	40	40	40	40
113	4-113	4-113	100-0000	1	0	0000	24	40	40	40	40	40	40	40
114	4-114	4-114	100-0000	1	0	0000	24	40	40	40	40	40	40	40
115	4-115	4-115	100-0000	1	0	0000	24	40	40	40	40	40	40	40
116	4-116	4-116	100-0000	1	0	0000	24	40	40	40	40	40	40	40
117	4-117	4-117	100-0000	1	0	0000	24	40	40	40	40	40	40	40
118	4-118	4-118	100-0000	1	0	0000	24	40	40	40	40	40	40	40
119	4-119	4-119	100-0000	1	0	0000	24	40	40	40	40	40	40	40
120	4-120	4-120	100-0000	1	0	0000	24	40	40	40	40	40	40	40
121	4-121	4-121	100-0000	1	0	0000	24	40	40	40	40	40	40	40
122	4-122	4-122	100-0000	1	0	0000	24	40	40	40	40	40	40	40
123	4-123	4-123	100-0000	1	0	0000	24	40	40	40	40	40	40	40
124	4-124	4-124	100-0000	1	0	0000	24	40	40	40	40	40	40	40
125	4-125	4-125	100-0000	1	0	0000	24	40	40	40	40	40	40	40
126	4-126	4-126	100-0000	1	0	0000	24	40	40	40	40	40	40	40
127	4-127	4-127	100-0000	1	0	0000	24	40	40	40	40	40	40	40
128	4-128	4-128	100-0000	1	0	0000	24	40	40	40	40	40	40	40
129	4-129	4-129	100-0000	1	0	0000	24	40	40	40	40	40	40	40
130	4-130	4-130	100-0000	1	0	0000	24	40	40	40	40	40	40	40
131	4-131	4-131	100-0000	1	0	0000	24	40	40	40	40	40	40	40
132	4-132	4-132	100-0000	1	0	0000	24	40	40	40	40	40	40	40
133	4-133	4-133	100-0000	1	0	0000	24	40	40	40	40	40	40	40
134	4-134	4-134	100-0000	1	0	0000	24	40	40	40	40	40	40	40
135	4-135	4-135	100-0000	1	0	0000	24	40	40	40	40	40	40	40
136	4-136	4-136	100-0000	1	0	0000	24	40	40	40	40	40	40	40
137	4-137	4-137	100-0000	1	0	0000	24	40	40	40	40	40	40	40
138	4-138	4-138	100-0000	1	0	0000	24	40	40	40	40	40	40	40
139	4-139	4-139	100-0000	1	0	0000	24	40	40	40	40	40	40	40
140	4-140	4-140	100-0000	1	0	0000	24	40	40	40	40	40	40	40
141	4-141	4-141	100-0000	1	0	0000	24	40	40	40	40	40	40	40
142	4-142	4-142	100-0000	1	0	0000	24	40	40	40	40	40	40	40
143	4-143	4-143	100-0000	1	0	0000	24	40	40	40	40	40	40	40
144	4-144	4-144	100-0000	1	0	0000	24	40	40	40	40	40	40	40
145	4-145	4-145	100-0000	1	0	0000	24	40	40	40	40	40	40	40
146	4-146	4-146	100-0000	1	0	0000	24	40	40	40	40	40	40	40
147	4-147	4-147	100-0000	1	0	0000	24	40	40	40	40	40	40	40
148	4-148	4-148	100-0000	1	0	0000	24	40	40	40	40	40	40	40
149	4-149	4-149	100-0000	1	0	0000	24	40	40	40	40	40	40	40
150	4-150	4-150	100-0000	1	0	0000	24	40	40	40	40	40	40	40
151	4-151	4-151	100-0000	1	0	0000	24	40	40	40	40	40	40	40
152	4-152	4-152	100-0000	1	0	0000	24	40	40	40	40	40	40	40
153	4-153	4-153	100-0000	1	0	0000	24	40	40	40	40	40	40	40
154	4-154	4-154	100-0000	1	0	0000	24	40	40	40	40	40	40	40
155	4-155	4-155	100-0000	1	0	0000	24	40	40	40	40	40	40	40
156	4-156	4-156	100-0000	1	0	0000	24	40	40	40	40	40	40	40
157	4-157	4-157	100-0000	1	0	0000	24	40	40	40	40	40	40	40
158	4-158	4-158	100-0000	1	0	0000	24	40	40	40	40	40	40	40
159	4-159	4-159	100-0000	1	0	0000	24	40	40	40	40	40	40	40
160	4-160	4-160	100-0000	1	0	0000	24	40	40	40	40	40	40	40
161	4-161	4-161	100-0000	1	0	0000	24	40	40	40	40	40	40	40
162	4-162	4-162	100-0000	1	0	0000	24	40	40	40	40	40	40	40
163	4-163	4-163	100-0000	1	0	0000	24	40	40	40	40	40	40	40
164	4-164	4-164	100-0000	1	0	0000	24	40	40	40	40	40	40	40
165	4-165	4-165	100-0000	1	0	0000	24	40	40	40	40	40	40	40
166	4-166	4-166	100-0000	1	0	0000	24	40	40	40	40	40	40	40
167	4-167	4-167	100-0000	1	0	0000	24	40	40	40	40	40	40	40
168	4-168	4-168	100-0000	1	0	0000	24	40	40	40	40	40	40	40
169	4-169	4-169	100-0000	1	0	0000	24	40	40	40	40	40	40	40
170	4-170	4-170	100-0000	1	0	0000	24	40	40	40	40	40	40	40
171	4-171	4-171	100-0000	1	0	0000	24	40	40	40	40	40	40	40
172	4-172	4-172	100-0000	1	0	0000	24	40	40	40	40	40	40	40
173	4-173	4-173	100-0000	1	0	0000	24	40	40	40	40	40	40	40
174	4-174	4-174	100-0000	1	0	0000	24	40	40	40	40	40	40	40
175	4-175	4-175	100-0000	1	0	0000	24	40	40	40	40	40	40	40
176	4-176	4-176	100-0000	1	0	0000	24	40	40	40	40	40	40	40
177	4-177	4-177	100-0000	1	0	0000	24	40	40	40	40	40	40	40
178	4-178	4-178	100-0000	1	0	0000	24	40	40	40	40	40	40	40
179	4-179	4-179	100-0000	1	0	0000	24	40	40	40	40	40	40	40
180	4-180	4-180	100-0000	1	0	0000	24	40	40	40	40	40	40	40
181	4-181	4-181	100-0000	1	0	0000	24	40	40	40	40	40	40	40
182	4-182	4-182	100-0000	1	0	0000	24	40	40	40	40	40	40	40
183	4-183	4-183	100-0000	1	0	0000	24	40	40	40	40	40	40	40
184	4-184	4-184	100-0000	1	0	0000	24	40	40	40	40	40	40	40
185	4-185	4-185	100-0000	1	0	0000	24	40	40	40	40	40	40	40
186	4-186	4-186	100-0000	1	0	0000	24	40	40	40	40	40	40	40
187	4-187	4-187	100-0000	1	0	0000	24	40	40	40	40	40	40	40
188	4-188	4-188	100-0000	1	0	0000	24	40	40	40	40	40	40	40
189	4-189	4-189	100-0000	1	0	0000	24	40	40	40	40	40	40	40
190	4-190	4-190	100-0000	1	0	0000	24	40	40	40	40	40	40	40
191	4-191	4-191	100-0000	1	0	0000	24	40	40	40	40	40	40	40
192	4-192	4-192	100-0000	1	0	0000	24	40	40	40	40	40	40	40
193	4-193	4-193	100-0000	1	0	0000	24	40	40	40	40	40	40	40
194	4-194	4-194	100-0000	1	0	0000	24	40	40	40	40	40	40	40
195	4-195	4-195	100-0000	1	0	0000	24	40	40	40	40	40	40	40
196	4-196	4-196	100-0000	1	0	0000	24	40	40	40	40	40	40	40
197	4-197	4-197	100-0000	1	0	0000	24	40	40	40	40	40	40	40
198	4-198	4-198	100-0000	1	0	0000	24	40	40	40	40	40	40	40
199	4-199	4-199	100-0000	1	0	0000	24	40	40	40	40	40	40	40
200	4-200	4-200	100-0000	1	0	0000	24	40	40	40	40	40	40	40
201	4-201	4-201	100-0000	1	0	0000	24	40	40	40	40	40	40	40
202	4-202	4-202	100-0000	1	0	0000	24	40	40	40	40	40	40	40
203	4-203	4-203	100-0000	1	0	0000	24	40	40	40	40	40	40	40
204	4-204	4-204	100-0000	1	0	0000	24	40	40	40	40	40	40	40
205	4-205	4-205	100-0000	1	0	0000	24	40	40	40	40	40	40	40
206	4-206	4-206	100-0000	1	0	0000	24	40	40	40	40	40	40	40
207	4-207	4-207	100-0000	1	0	0000	24	40	40	40	40	40	40	40
208	4-208	4-208	100-0000	1	0	0000	24	40	40	40	40	40	40	40
209	4-209	4-209	100-0000	1	0	0000	24	40	40	40	40	40	40	40
2														

```

(1) speedup(1:2) := 2
(2) speedup(1:3) := vbest(Tbest) if best(T:2) = vbest(Tbest) / 2 + 1 (avg)
    if best(T:2) = vbest(Tbest) / 2 + 2 (better)

```

SEE 754 FLOATING-POINT STANDARD

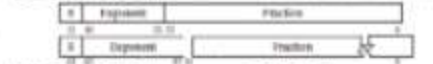
$$f: \mathbb{R}^n \rightarrow \mathbb{R} \text{ (function)} \rightarrow \mathbb{R}^{\text{dimension: } n}$$

where Single Precision Step = 128.

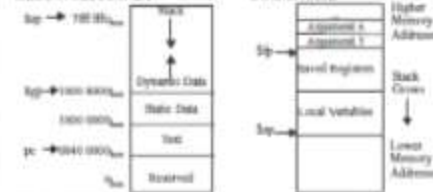
Double Precision Map - 1922

### GLL Single Premium and

### Double Precision Formats:



### MEMORY ALLOCATION



#### DATA ALIGNMENT

Double Word							
Word				Word			
Halfword		Halfword		Halfword		Halfword	
byte	byte	byte	byte	byte	byte	byte	byte

value of these four significant bits of byte address (byte index)

## EXCEPTION CONTROL REGISTERS: CAUSE AND STATUS

W		Interrupt		Exception	
D		Mask		Code	
W					
		Pending		U	S
		Interrupt		A	I
				C	E

HD = French (e.g., 1.00 = user mode, 0.01 = description level, 0.00 = message level)

### EXCEPTION CODES

Number	Form	Amount of exception	Number	Form	Amount of exception
3	III	Admission (Admission Exception)	9	III	Responsible Exception
4	Admission	Admission (Admission Exception)	10	III	Responsible Exception
5	Admission	Admission (Admission Exception)	11	III	Responsible Exception
6	III	Admission (Admission Exception)	12	III	Responsible Exception
7	III	Admission (Admission Exception)	13	III	Responsible Exception
8	III	Admission (Admission Exception)	14	III	Responsible Exception

SIZE PREFIXES ( $10^3$  for Disk, Communication;  $2^3$  for Memory)

P202		P201		P202		P202	
SIZE	FOR	SIZE	FOR	SIZE	FOR	SIZE	FOR
$10^{1-2}$	Kilo	$10^{1-2}$	Mega	$10^{1-2}$	Mega	$10^{1-2}$	Mega
$10^{2-3}$	Mega	$10^{2-3}$	Giga	$10^{2-3}$	Giga	$10^{2-3}$	Giga
$10^{3-4}$	Giga	$10^{3-4}$	Tera	$10^{3-4}$	Tera	$10^{3-4}$	Tera
$10^{4-5}$	Tera	$10^{4-5}$	Peta	$10^{4-5}$	Peta	$10^{4-5}$	Peta

The expected rate each profile is paid the total effort, except  $\mu$  is used for the search.

## 本节小结



# MIPS指令简介

北京大学·慕课  
计算机组成  
制作人：陆俊林

