## 处理器设计

2. 数据通路的建立(1)

主讲人: 邓倩妮

上海交通大学

大部分内容来自于:

Computer Organization and Design, 4th Edition, Patterson & Hennessy,





### 设计处理器的五个步骤

- 1. 分析指令系统, 得出对数据通路的需求
  - > The meaning of each instruction is given by the register transfers
  - ▶ 例如:ADDU指令的数据通路需求:R[rd] <- R[rs] + R[rt];
- 2. 选择数据通路上合适的组件
  - ▶ 例如:加法器? 算术逻辑运算单元?寄存器堆?
- 3. 连接组件构成数据通路
- 4. 分析每一条指令的实现, 以确定控制信号,
  - > 不同的控制信号影响寄存器之间的数据传送
- 5. 集成控制信号, 完成控制逻辑



### Step 3:连接组件构成数据通路

- □ 装配数据通路 Datapath Assembly
- □指令执行的数据通路:

取指令 Instruction Fetch

读操作数 Read Operands

执行指令 Execute Operation

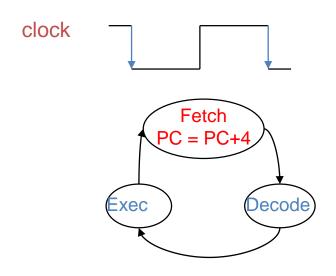


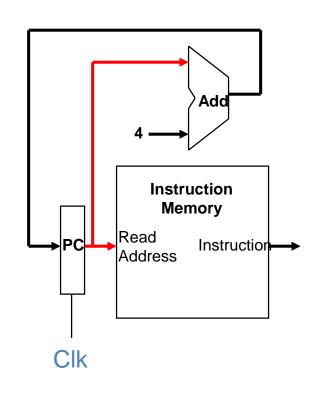
### **Fetching Instructions**

#### 取指令步骤:

从Instruction Memory读指令 M[PC]

将PC值更新为顺序执行的下一条指令的地址PC ← PC + 4





读Instruction Memory 是一个组合电路实现逻辑,不需要显式的read control signal

PC 每个时钟周期更新一次, 时钟边沿触发状态的写入,不需要一个显式的 write enable 控制信号



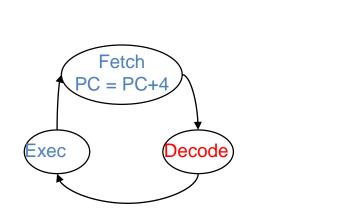
### 指令译码 Decoding Instructions

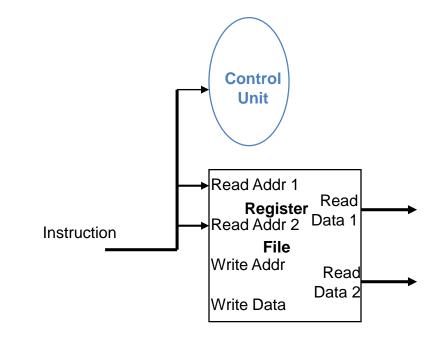
#### □ 指令译码包括:

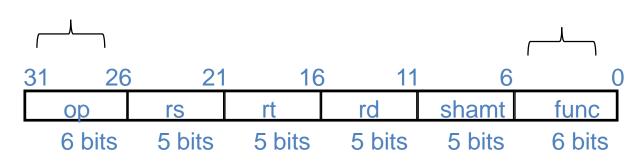
将取到的指令的 opcode 和 function 域对应的位数送给控制器

#### 读寄存器

- 根据指令中的寄存器地址
- 从寄存器文件读两个值





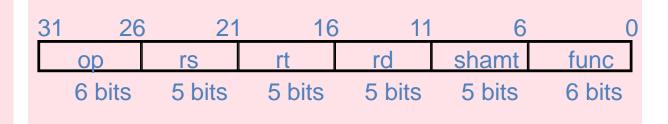




### 执行指令: R-type Instructions

#### Instructions:

ADD and subtractadd rd, rs, rtsub rd, rs, rt

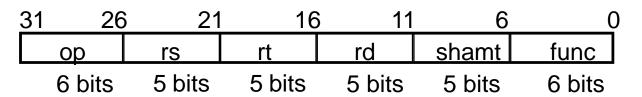


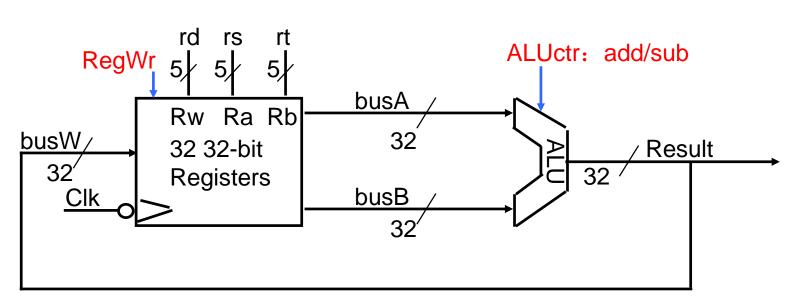


### 执行 (R-type) 指令的数据通路

□ 例如: add rd, rs, rt

□ RTL:  $R[rd] \leftarrow R[rs]$  op R[rt]





Ra, Rb, Rw 对应 rs, rt, rd

ALUctr,RegWr: 控制信号

"add rd, rs, rt" 控制信号?

ALUctr=add, RegWr=1

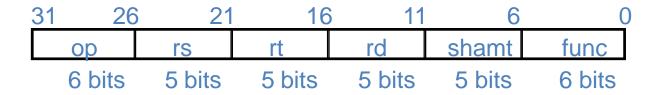


### 执行指令: I-type Instructions

#### Instructions:

OR Immediate:ori rt, rs, imm16







#### **RTL: The OR Immediate Instruction**

ori rt, rs, imm16

| 31 26  | <u>3</u> 21 | 16     | S C       |
|--------|-------------|--------|-----------|
| op     | rs          | rt     | immediate |
| 6 bits | 5 bits      | 5 bits | 16 bits   |

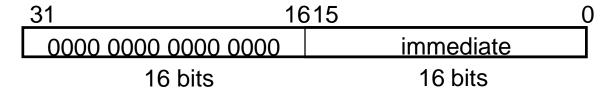
M[PC] Instruction Fetech

 $R[rt] \leftarrow R[rs]$  or ZeroExt(imm16)

zero extension of 16 bit constant or R[rs]

PC ← PC + 4 update PC

Zero extension ZeroExt(imm16)





### 执行 OR Immediate 指令的数据通路

rt, rs, imm16 不同于: add  $R[rt] \leftarrow R[rs]$  op ZeroExt[imm16]] Example: ori rd, rs, rt Why need multiplexor RTL:  $R[rd] \leftarrow R[rs]$  op Rd Rt here? RegDst Don't Care (Rt) Write the results of R-RegWr 5 Rs 5 **ALUctr** Type instruction to Rd busA Rw Ra Rb busW 32 Result 32 32-bit 32 Registers 32 busB, ZeroEx imm16 32 **ALUSrc** 

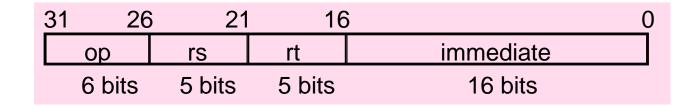
Onicontrolsignals:: RegDst=?; ; RegDwr=1?; ; AAUGct=1?; ;



### Iw 指令数据通路 (memory access instruction)

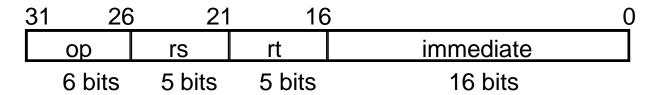
#### LOAD and STORE

lw rt, rs, imm16 sw rt, rs, imm16





#### **RTL: The Load Instruction**



□ lw rt, rs, imm16

M[PC]

Addr  $\leftarrow$  R[rs] + SignExt(imm16)

 $R[rt] \leftarrow M [Addr]$ 

 $PC \leftarrow PC + 4$ 

Instruction Fetch

Compute the address

Load Data to rt

Update PC

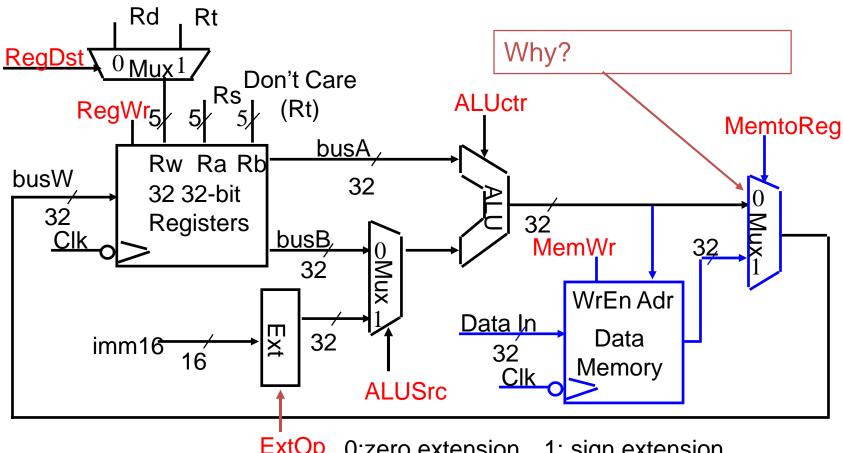
Why using signed extension rather than zero extension?

| 31  |                    | 1615 |           |   |  |  |
|-----|--------------------|------|-----------|---|--|--|
| 000 | 00 0000 00000 0000 | ) 0  | immediate |   |  |  |
| 31  | 16 bits            | 1615 | 16 bits   | 0 |  |  |
| 11  | 11 1111 1111 1111  | 1    | immediate |   |  |  |
|     | 16 bits            |      | 16 bits   |   |  |  |



### Data path for Load Instruction

对比 add rd, rs, rt □  $R[rt] \leftarrow M[R[rs] + SignExt[imm16]]$ Example: lw rt, rs, imm16



ExtOp 0:zero extension, 1: sign extension

RegDst=1, RegWr=1, ALUctr=add, ExtOp=1, ALUSrc=1, MemWr=0, MemtoReg=1



#### **RTL: The Store Instruction**

sw rt, rs, imm16

| 31 | 26     | 3 21   | 16     |           | 0 |
|----|--------|--------|--------|-----------|---|
|    | ор     | rs     | rt     | immediate |   |
|    | 6 bits | 5 bits | 5 bits | 16 bits   |   |

```
M[PC]
Addr \leftarrow R[rs] + SignExt(imm16)
Mem[Addr] \leftarrow R[rt]
PC \leftarrow PC + 4
```



### Data path for SW

 $M[R[rs] + SignExt[imm16]] \leftarrow R[rt]$ Example: sw rt, rs, imm16 Rd , Rt RegDst  $0 \, \text{Mux} 1$ Why add this? Rs<sub>,Rt</sub> **ALUctr** RegWr /MemWr MemtoReg busA Rw Ra Rb busW 32 32 32-bit 32 32 Registers busB<sub>2</sub> Data In WrEn Adr Ext Data imm16 32 Memory ALUSrc ExtOp

RegDst=x, RegWr=0, ALUctr=add, ExtOp=1, ALUSrc=1, MemWr=1, MemtoReg=x



### 小结:设计处理器的五个步骤

- 1. 分析指令系统, 得出对数据通路的需求
  - > The meaning of each instruction is given by the register transfers
  - ▶ 例如:ADDU指令的数据通路需求:R[rd] <- R[rs] + R[rt];</p>
- 2. 选择数据通路上合适的组件
  - ▶ 例如:加法器? 算术逻辑运算单元?寄存器堆?
- 3. 连接组件构成数据通路
- 4. 分析每一条指令的实现,以确定控制信号,
  - > 不同的控制信号影响寄存器之间的数据传送
- 5. 集成控制信号, 完成控制逻辑

# 谢谢!

