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

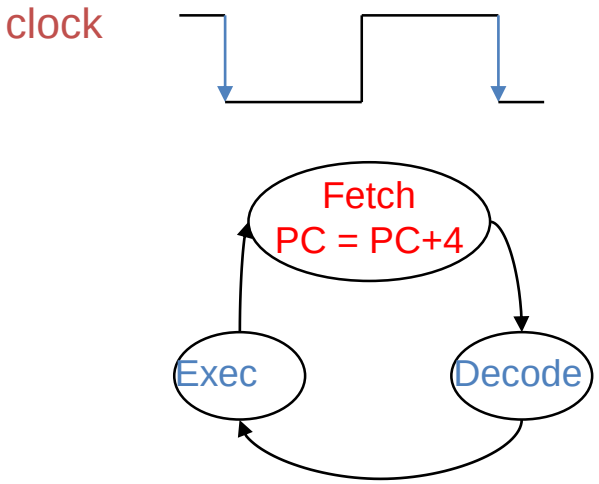
  执行指令 Execute Operation

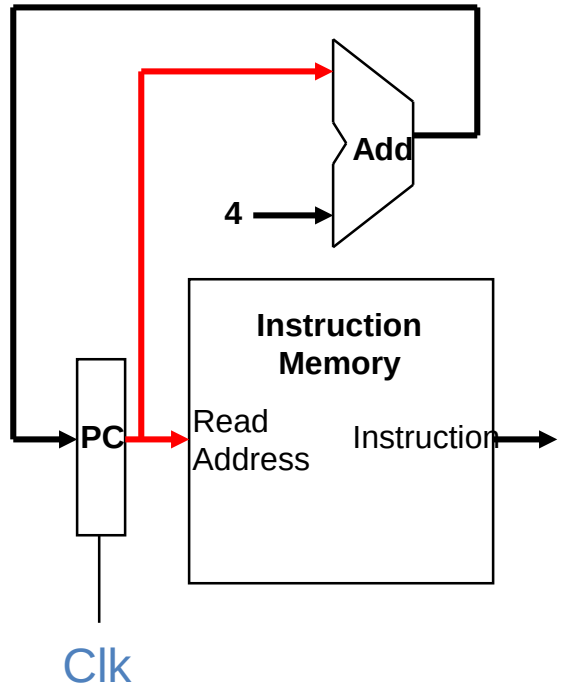
Fetching Instructions

- 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



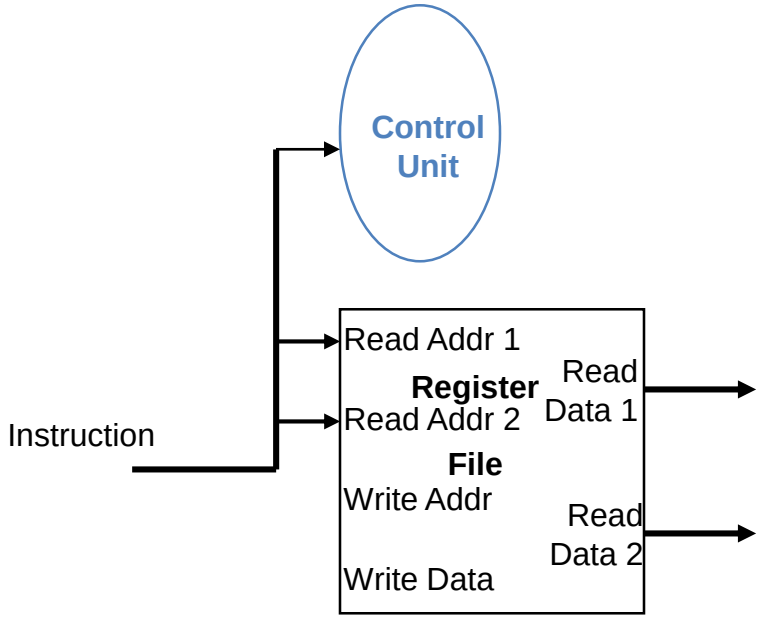


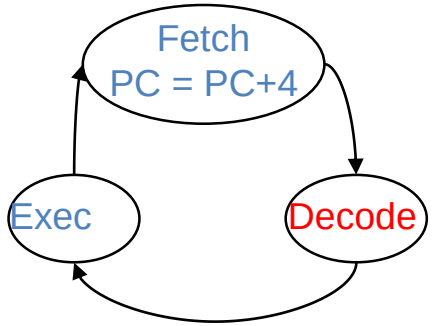
读Instruction Memory 是一个组合电路实现逻辑，不需要显式的read control signal

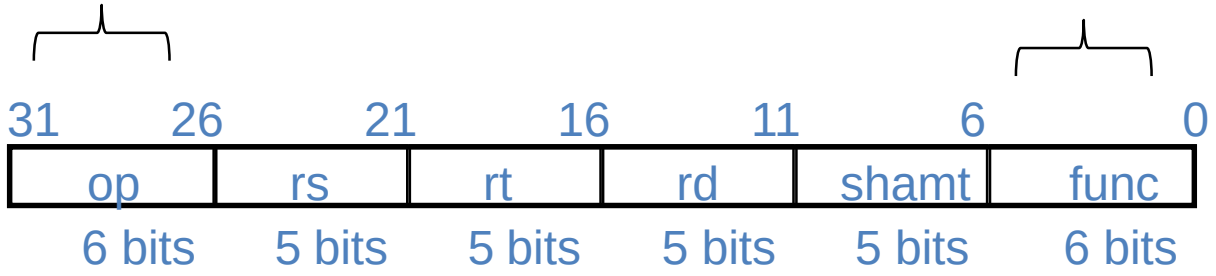
- PC 每个时钟周期更新一次, 时钟边沿触发状态的写入，不需要一个显式的 write enable 控制信号

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







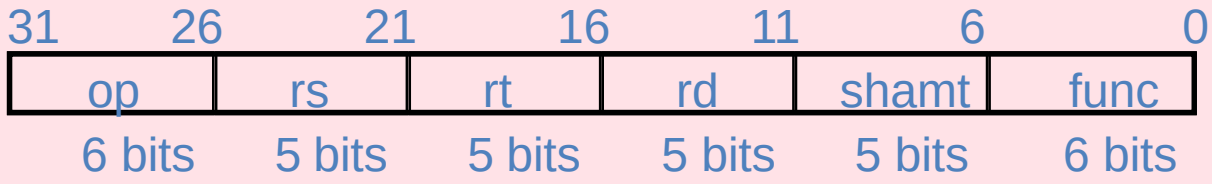
执行指令： R-type Instructions

Instructions：

- ADD and subtract

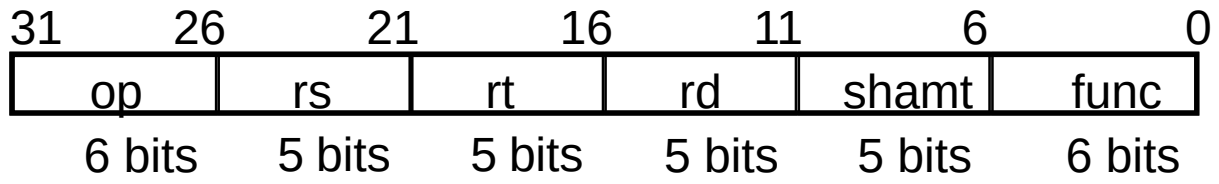
  add rd, rs, rt

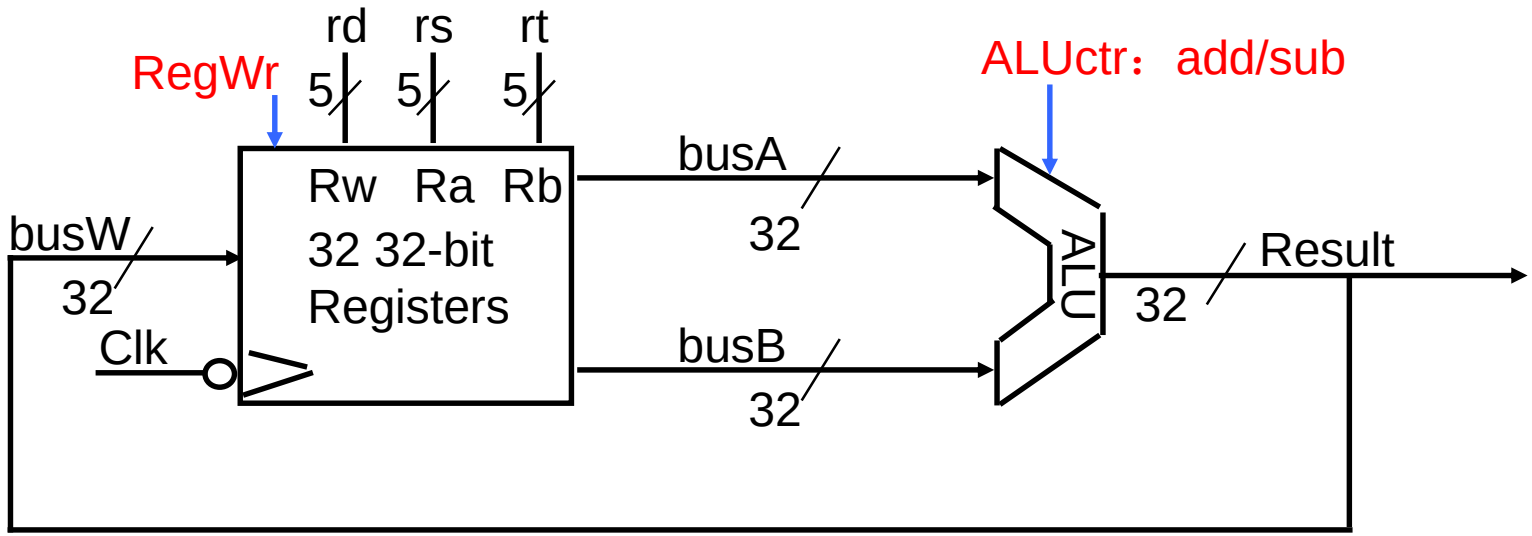
  sub rd, rs, rt



执行（R-type）指令的数据通路

- 例如: add　rd, rs, rt
- RTL：R[rd] ← 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？

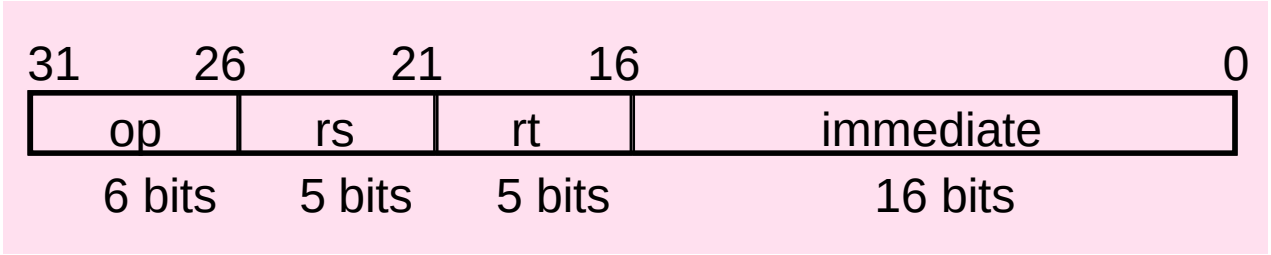
ALUctr=add，RegWr=1

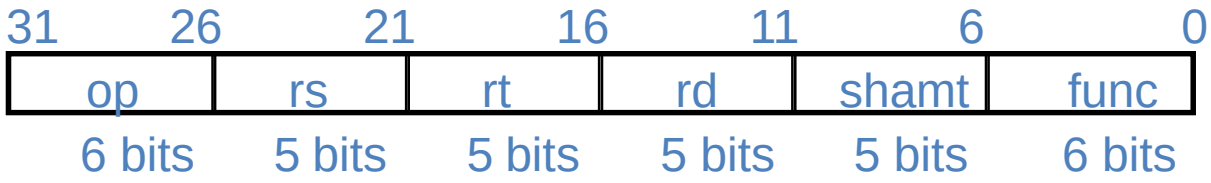
执行指令： I-type Instructions

Instructions：

- OR Immediate:

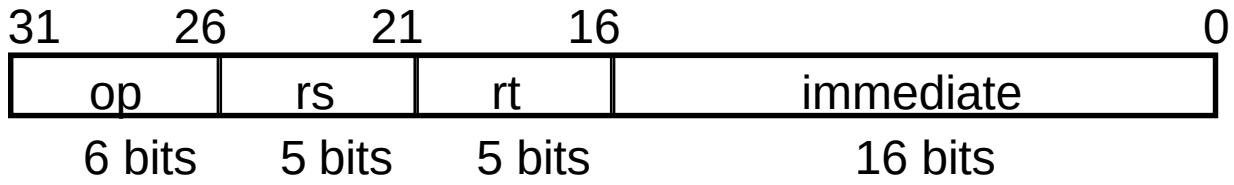
  ori rt, rs, imm16





RTL: The OR Immediate Instruction

- ori rt, rs, imm16



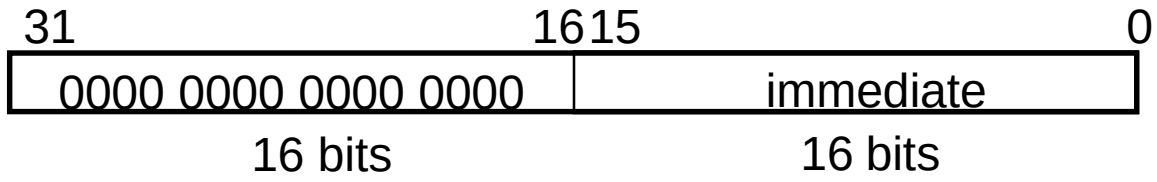
M[PC] Instruction Fetech

R[rt] ← R[rs] or ZeroExt(imm16)

zero extension of 16 bit constant or R[rs]

PC ← PC + 4 update PC

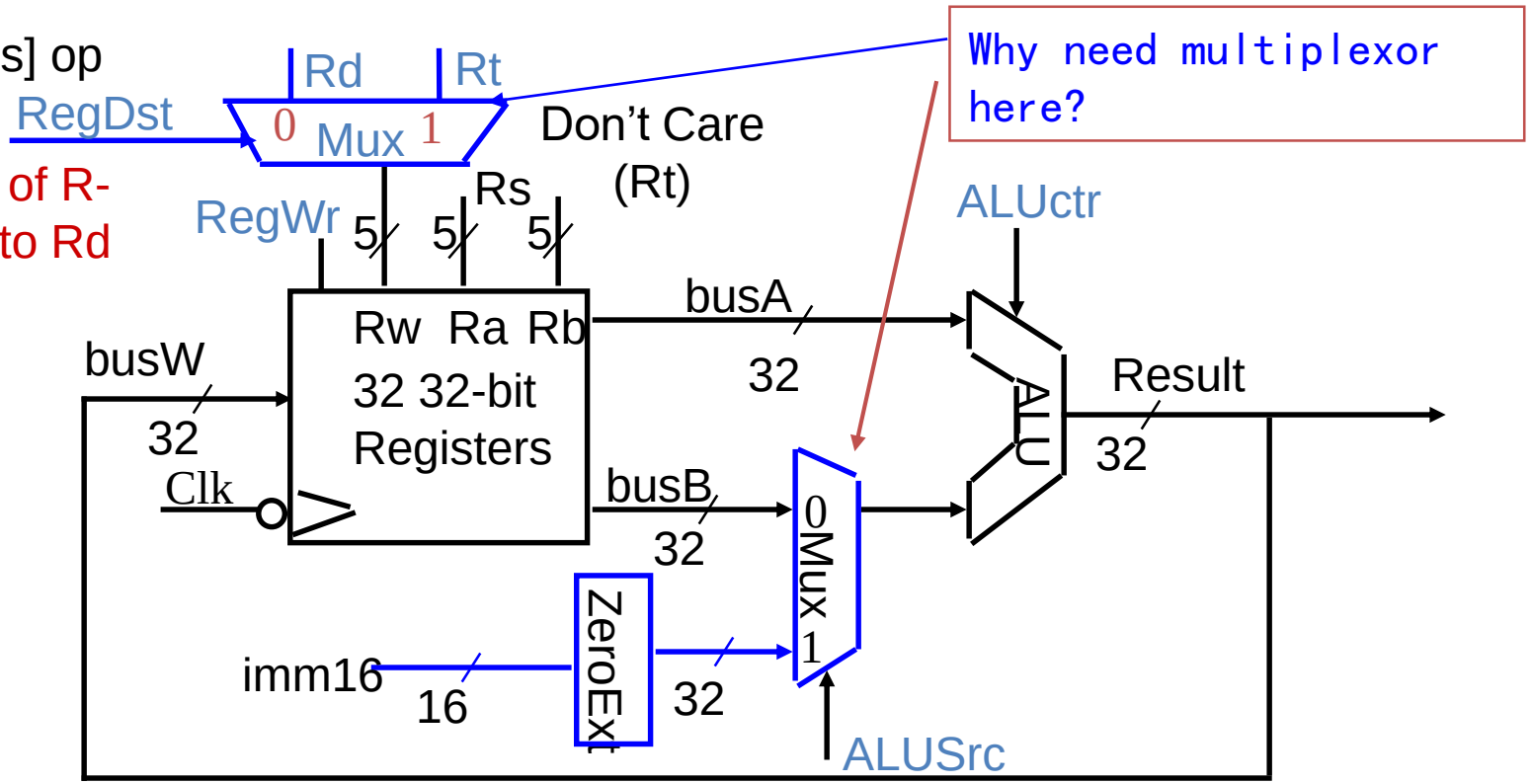
Zero extension ZeroExt(imm16)



执行 OR Immediate 指令的数据通路

- R[rt] ← R[rs] op ZeroExt[imm16]]　　Example: ori　rt, rs, imm16　　不同于：　add　rd, rs, rt
- RTL：R[rd] ← R[rs] op

Write the results of R-Type instruction to Rd



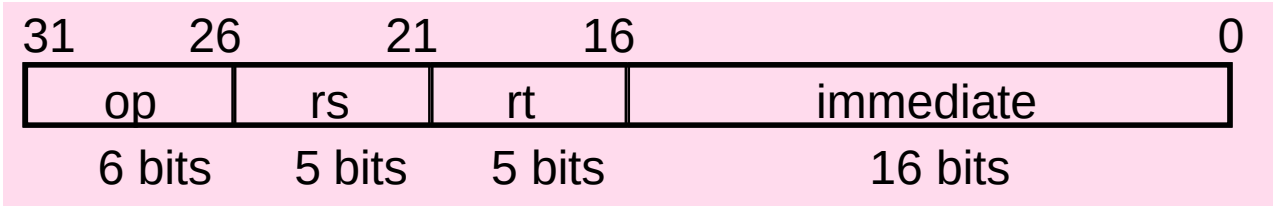
Ori control signals:　RegDst=?　;　RegWr=?　;　ALUSrc=1;　ALUctr=?

lw 指令数据通路 (memory access instruction)

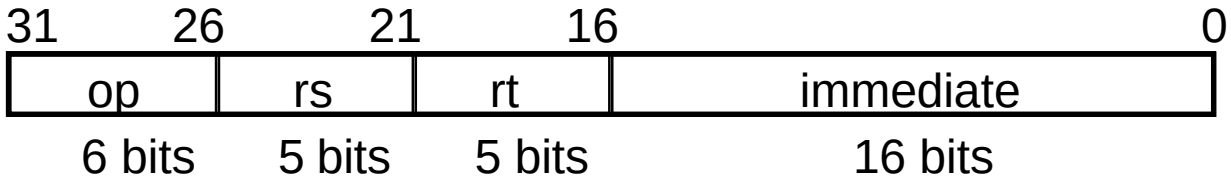
- LOAD and STORE

  lw rt, rs, imm16

  sw rt, rs, imm16



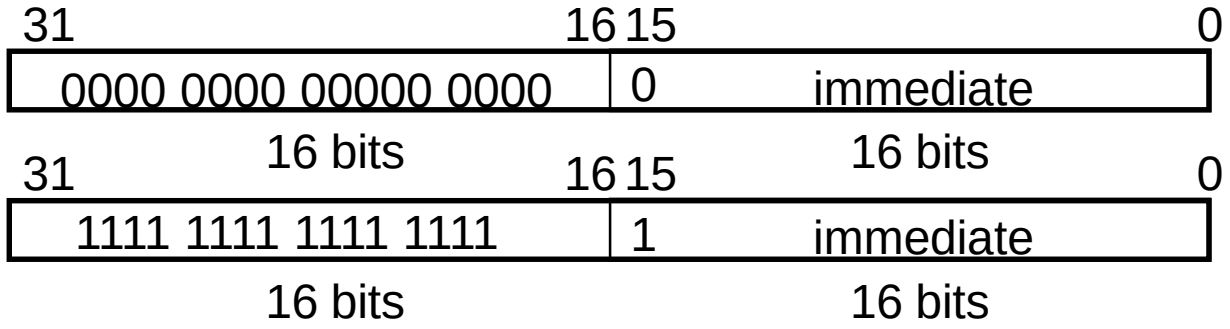
RTL: The Load Instruction



- lw rt, rs, imm16

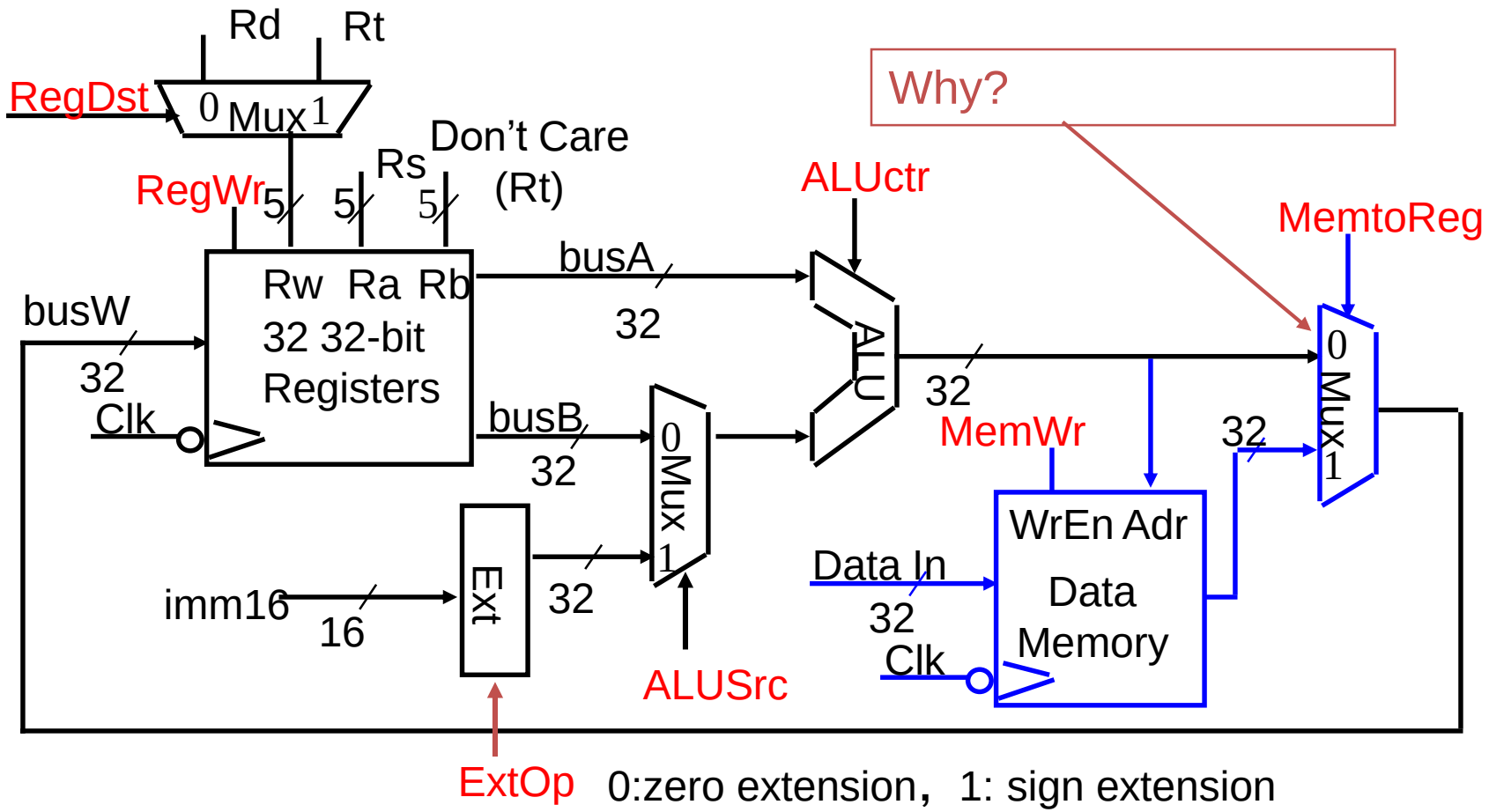
M[PC]	Instruction Fetch
Addr ← R[rs] + SignExt(imm16)	Compute the address
R[rt] ← M [Addr]	Load Data to rt
PC ← PC + 4	Update PC

Why using signed extension rather than zero extension?



Data path for Load Instruction

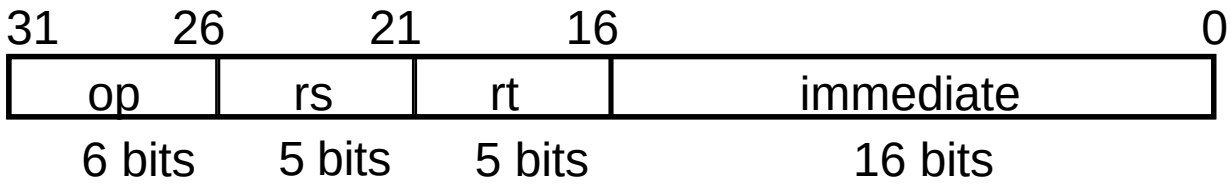
- R[rt] ← M[R[rs] + SignExt[imm16]] Example: lw rt, rs, imm16 对比 add rd, rs, rt



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

RTL: The Store Instruction

- sw rt, rs, imm16



M[PC]

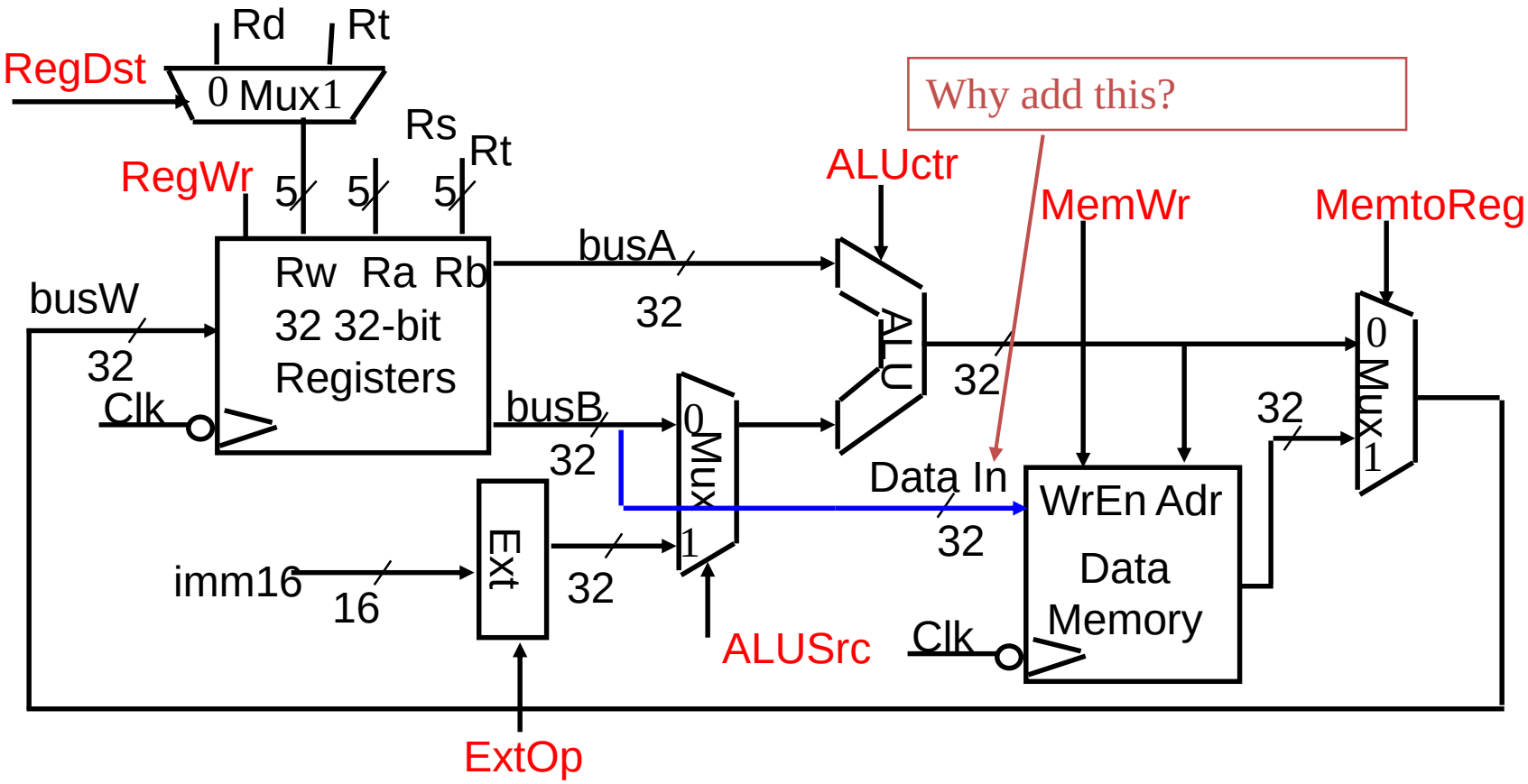
Addr ← R[rs] + SignExt(imm16)

Mem[Addr] ← R[rt]

PC ← PC + 4

Data path for SW

- M[R[rs] + SignExt[imm16]] ← R[rt] Example: sw rt, rs, imm16



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];
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！