处理器设计

2. 数据通路的建立(2)

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

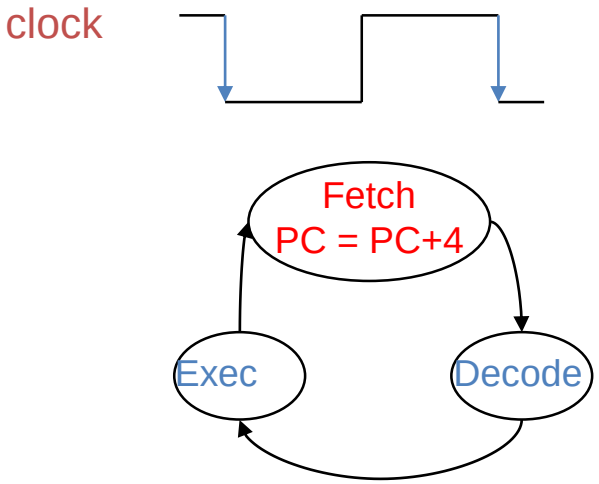
  执行指令 Execute Operation

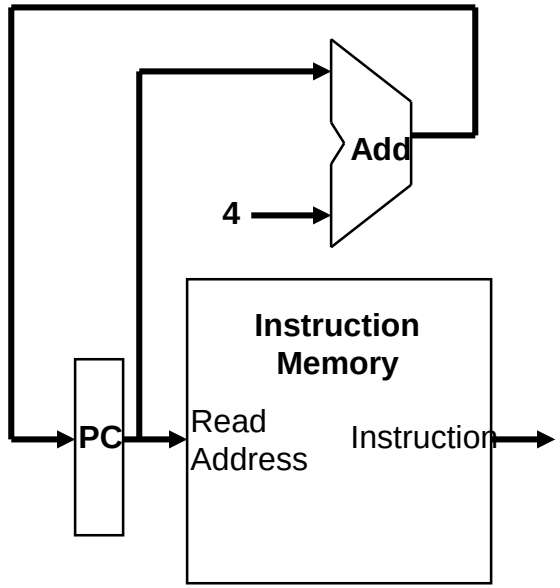
Fetching Instructions

- 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





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

指令译码 Decoding Instructions

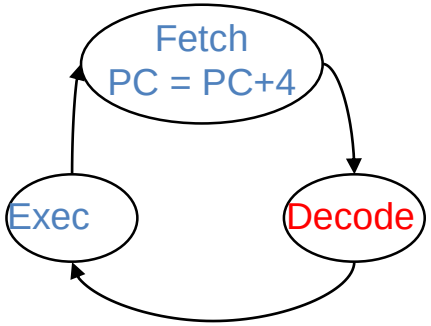
- 指令译码包括：

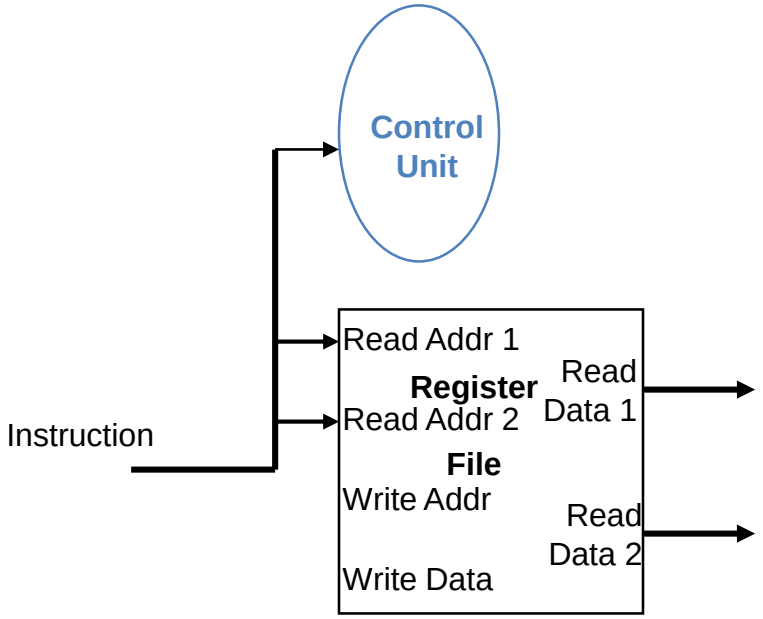
  将取到的指令的 opcode 和 function 域对应的位数送给控制器

  并：

  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

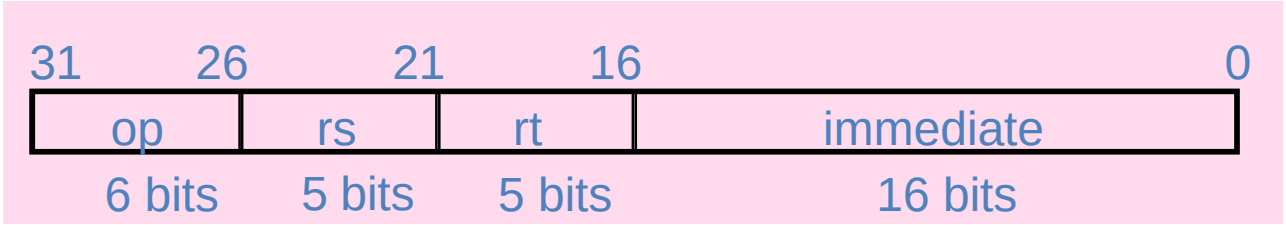




Beq

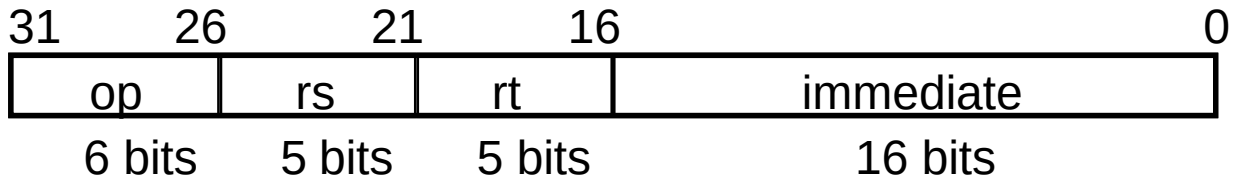
- BRANCH:

  beq rs, rt, imm16



RTL: The Branch Instruction

- beq rs, rt, imm16



M[PC]

Cond ← R[rs] - R[rt] Compare rs and rt

if (COND eq 0) Calculate the next instruction's address

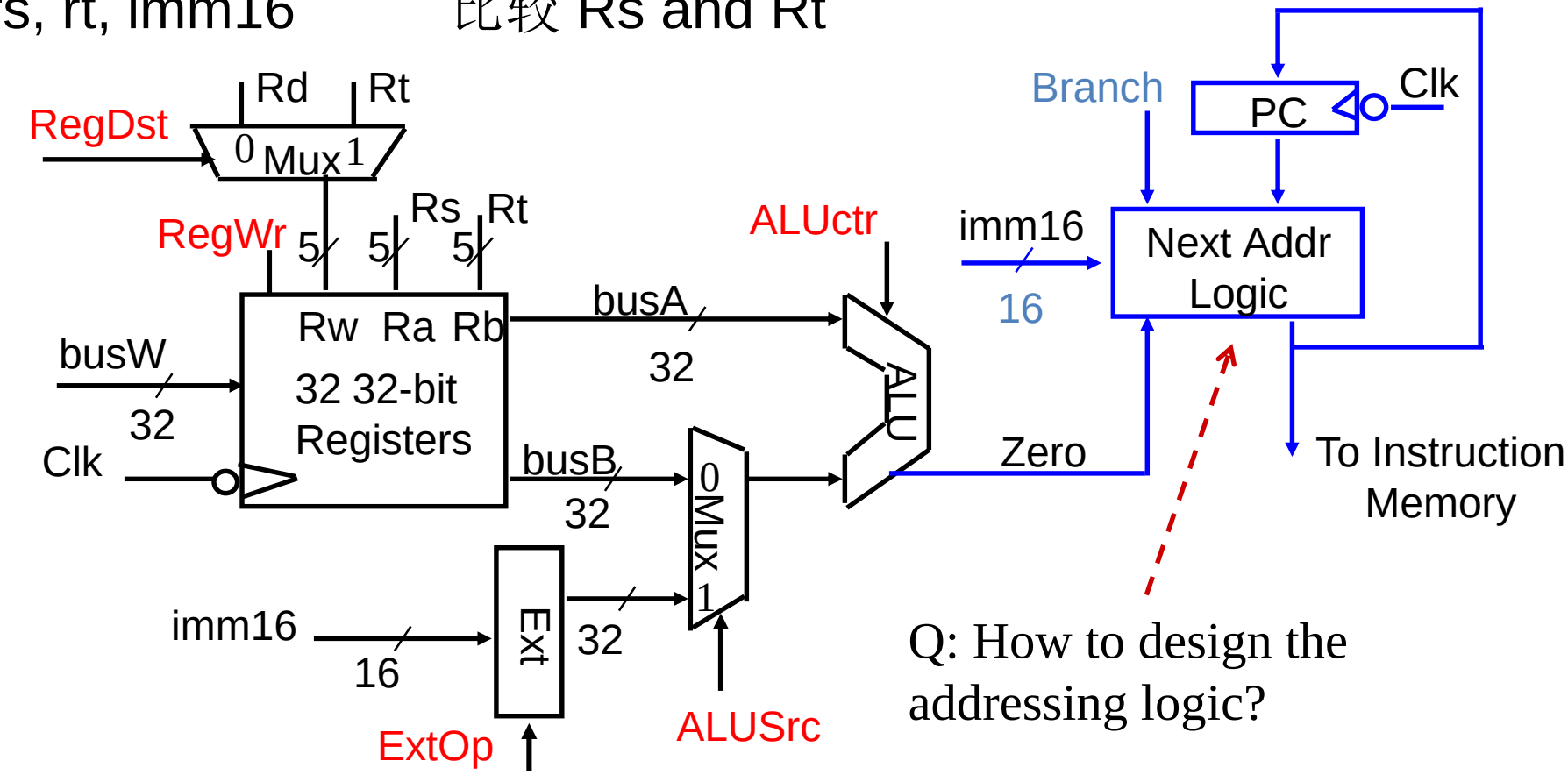
  PC ← PC + 4 + (SignExt(imm16) x 4)

else

  PC ← PC + 4

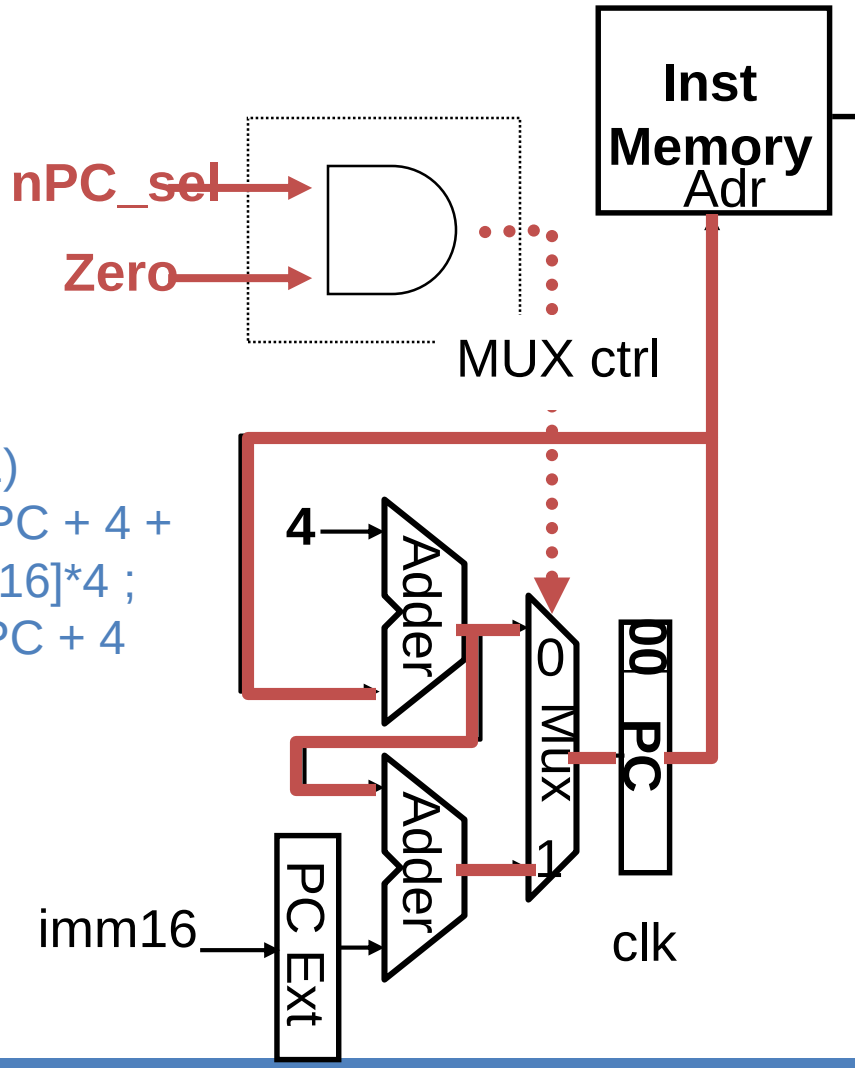
数据通路 for beq

- beq rs, rt, imm16 比较 Rs and Rt



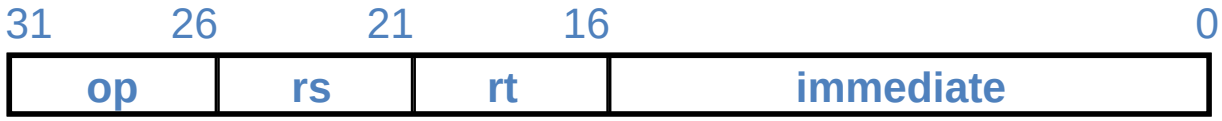
RegDst=x, RegWr=0, ALUctr=sub, ExtOp=x, ALUSrc=0, MemWr=0, MemtoReg=x, Branch=1

Instruction Fetch Unit at the End of Branch



if (Zero == 1)
then PC = PC + 4 + SignExt[imm16]*4 ;
else PC = PC + 4

- What is encoding of nPC_sel?
  - Direct MUX select?
  - Branch inst. / not branch
- Let's pick 2nd option



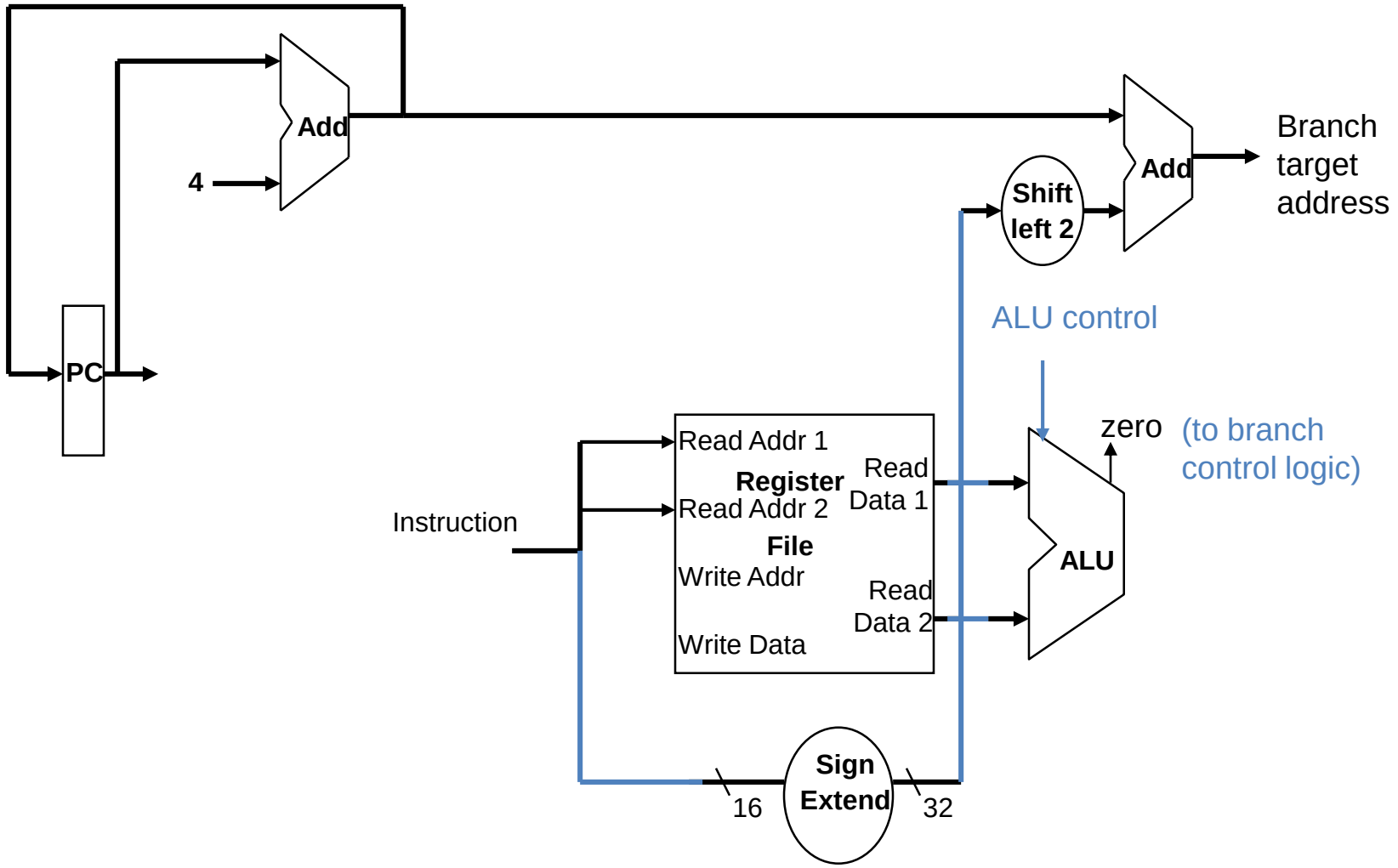
nPC_sel	zero?	MUX
0	x	0
1	0	0
1	1	1

Q: What logic gate?

执行 Branch 指令

Branch 操作包括:

- 比较从寄存器文件中取出的两个操作数是否相等(`zero` ALU output)
- 计算branch目标地址 updated PC +16-bit signed-extended offset field



Jump operation

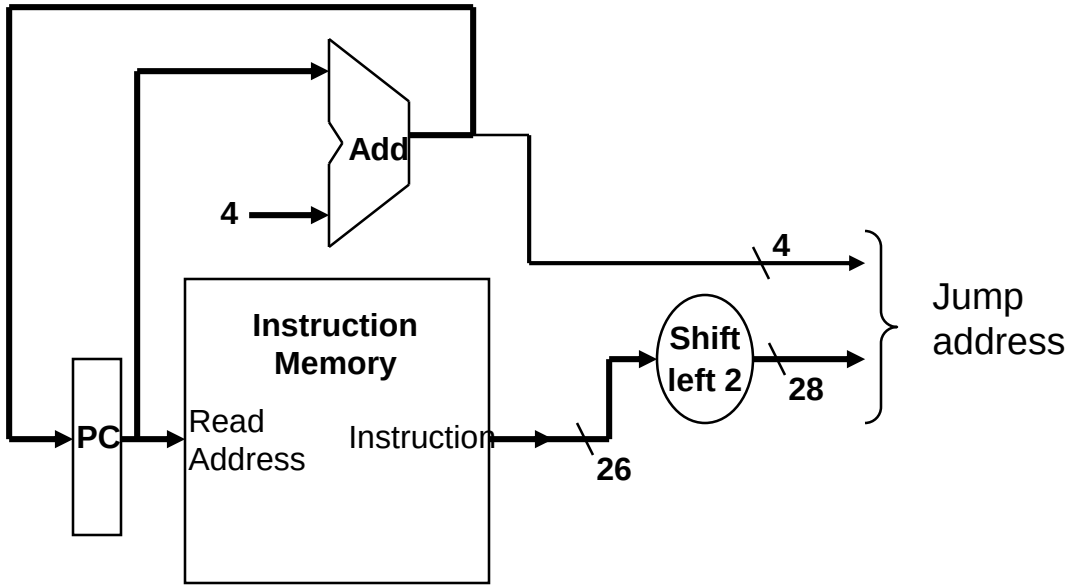
- JUMP:

```
j  target
```

执行Jump 指令

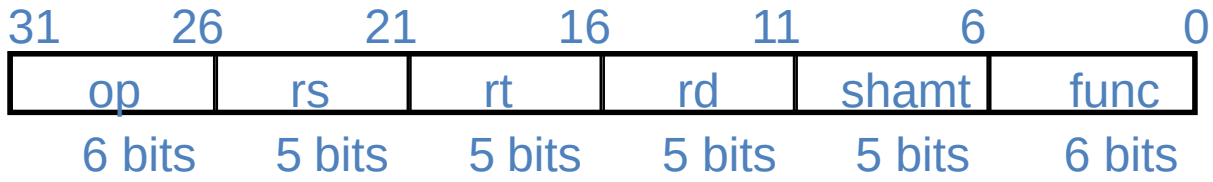
- Jump 操作包括：

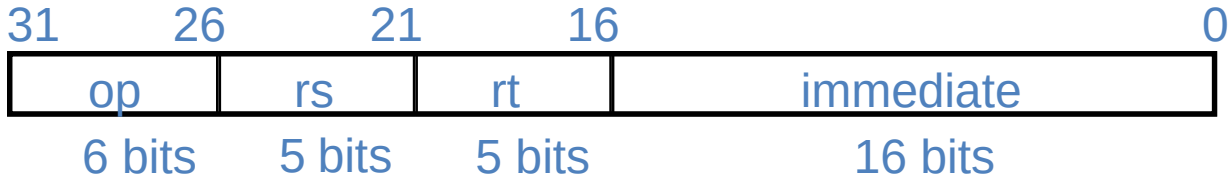
  将PC的低28位替换为： 指令中的最后 26位 左移 2 bits 的值



The MIPS Subset

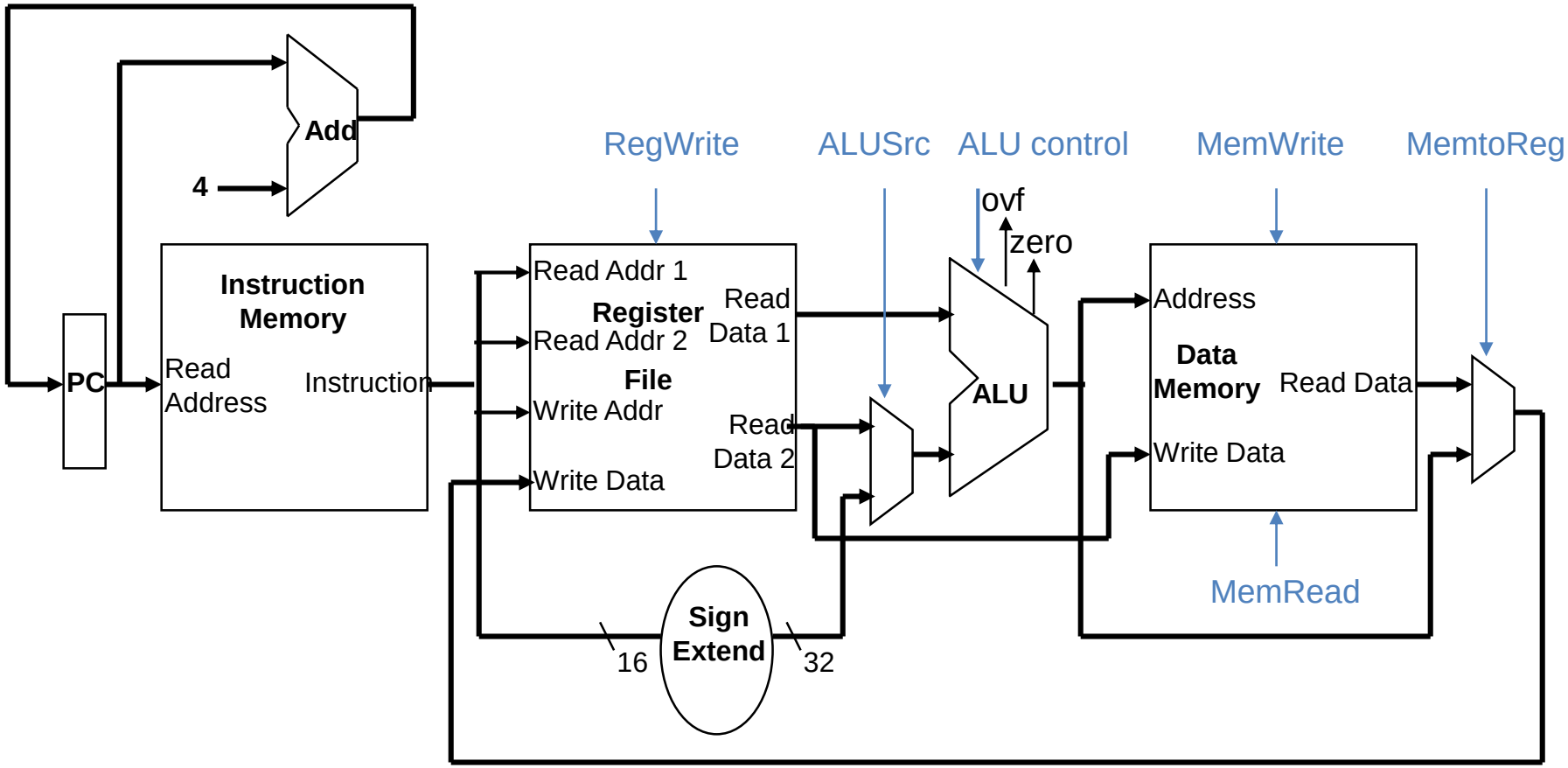
- ADD and subtract
  - add rd, rs, rt
  - sub rd, rs, rt
- OR Immediate:
  - ori rt, rs, imm16
- LOAD and STORE
  - lw rt, rs, imm16
  - sw rt, rs, imm16
- BRANCH:
  - beq rs, rt, imm16
- JUMP:
  - j target





Now, let's put them together!

Fetch（取址）,R（寄存器），Memory（存储器） 各部分



构建数据通路

单周期（Single cycle）处理器的设计

- 一个周期内完成指令的取址、译码、执行（fetch, decode and execute）等操作
- 资源在一个周期内不能重复利用，所以某些资源需要多份，例如存储器（分离的指令存储器和数据存储器）
- 构建数据通路，需要时增加控制信号线路和多路选择器（multiplexors）

  多路选择器（multiplexors）：多个输入共享一个输入组件时，利用控制信号，从多个输入中选择一个

  Write enable 信号，用来控制Register File 和 Data Memory 的写入
- 周期时间的长度：由最长数据通路的延迟决定

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！