



# 计算机原理

COMPUTER PRINCIPLE

第四章 第一节 (3) Load、Store、条件分支指令的数据通路

□以Load指令LW R1, R2, #4为例

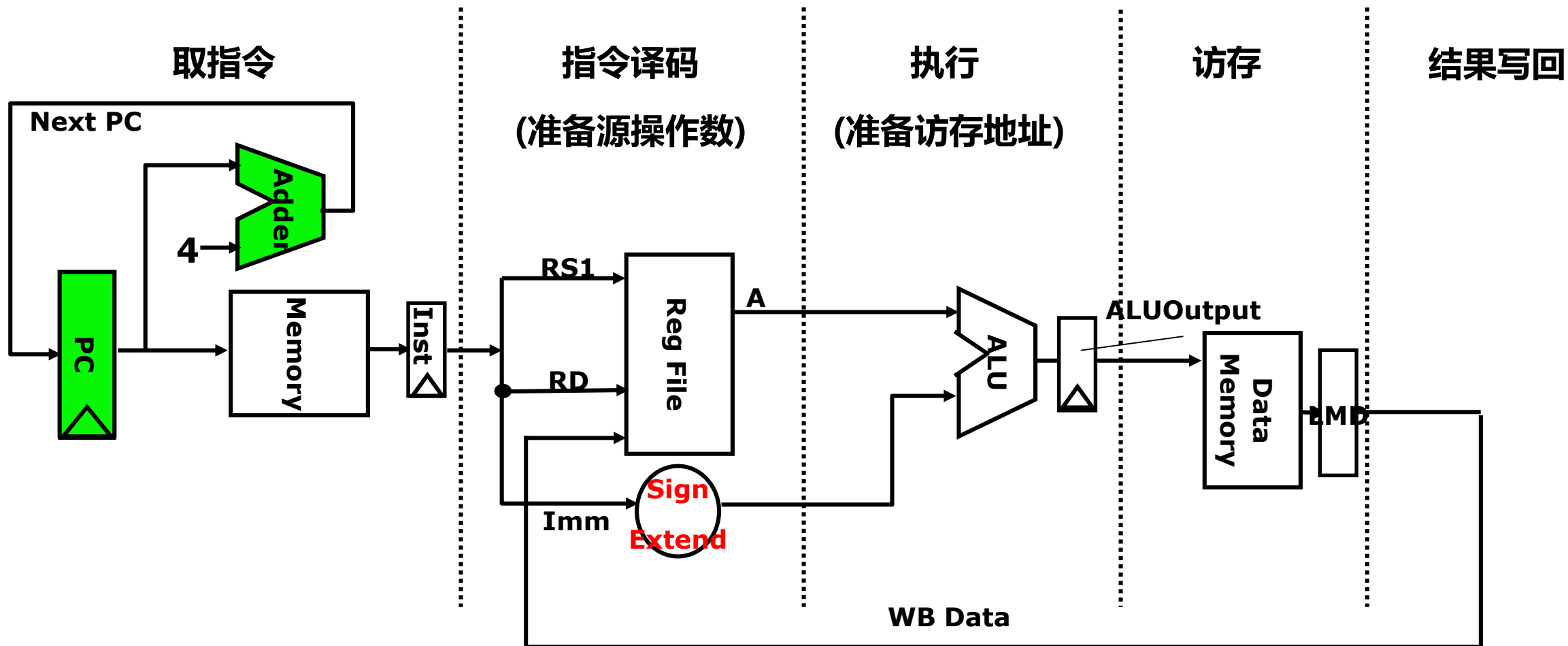
## Load指令LW R1, R2, #4

功能	$R1 \leftarrow \text{Mem}[R2+4]$
过程	<ul style="list-style-type: none"><li>① 取指令, <math>\text{Inst} \leftarrow [\text{PC}]</math>, <math>\text{PC} \leftarrow \text{PC}+4</math></li><li>② 指令译码, <math>A \leftarrow [R2]</math>, 立即数符号扩展为32位</li><li>③ 计算访存地址<math>A+4</math></li><li>④ 读地址为<math>A+4</math>的存储单元, <math>\text{LMD} \leftarrow [A+4]</math></li><li>⑤ 结果写回, <math>[R1] \leftarrow \text{LMD}</math></li></ul>



# Load指令的数据通路

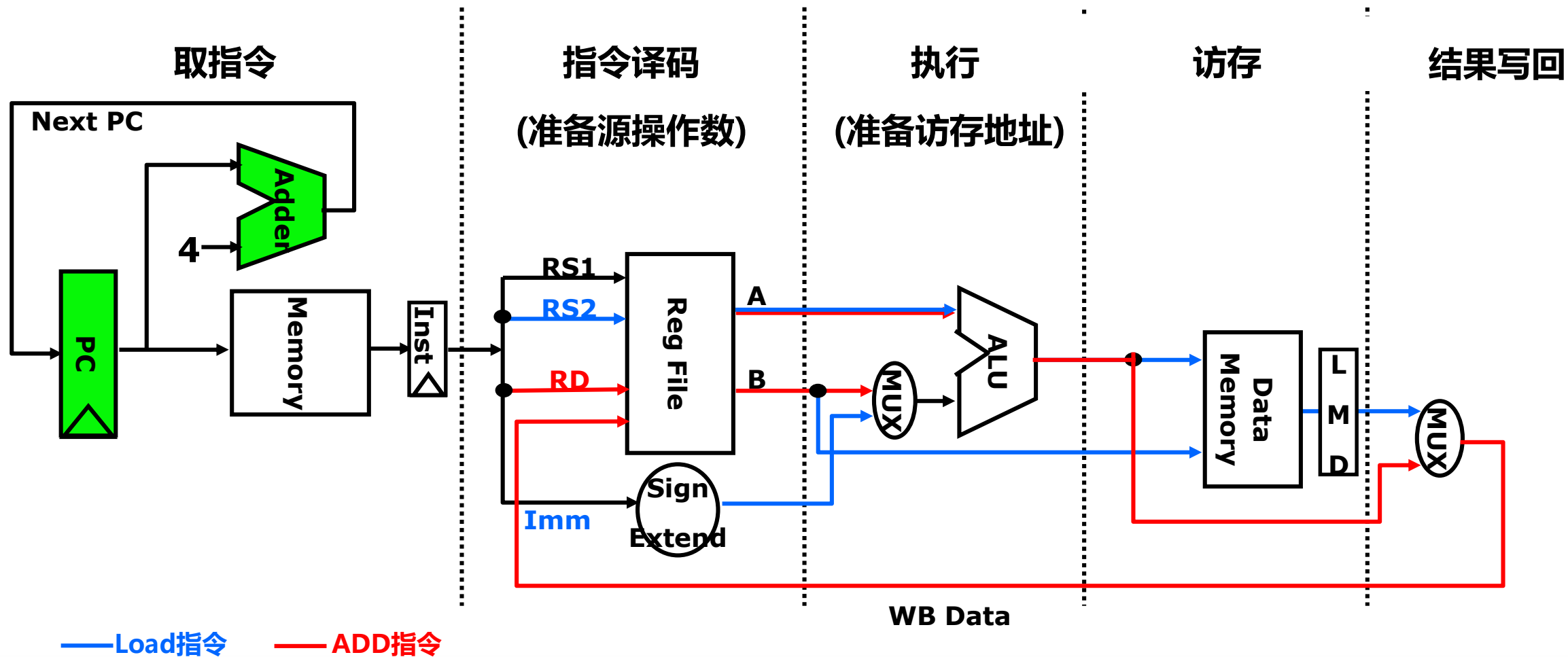
## □ Load指令的数据通路





# 合并ADD指令和Load指令的数据通路

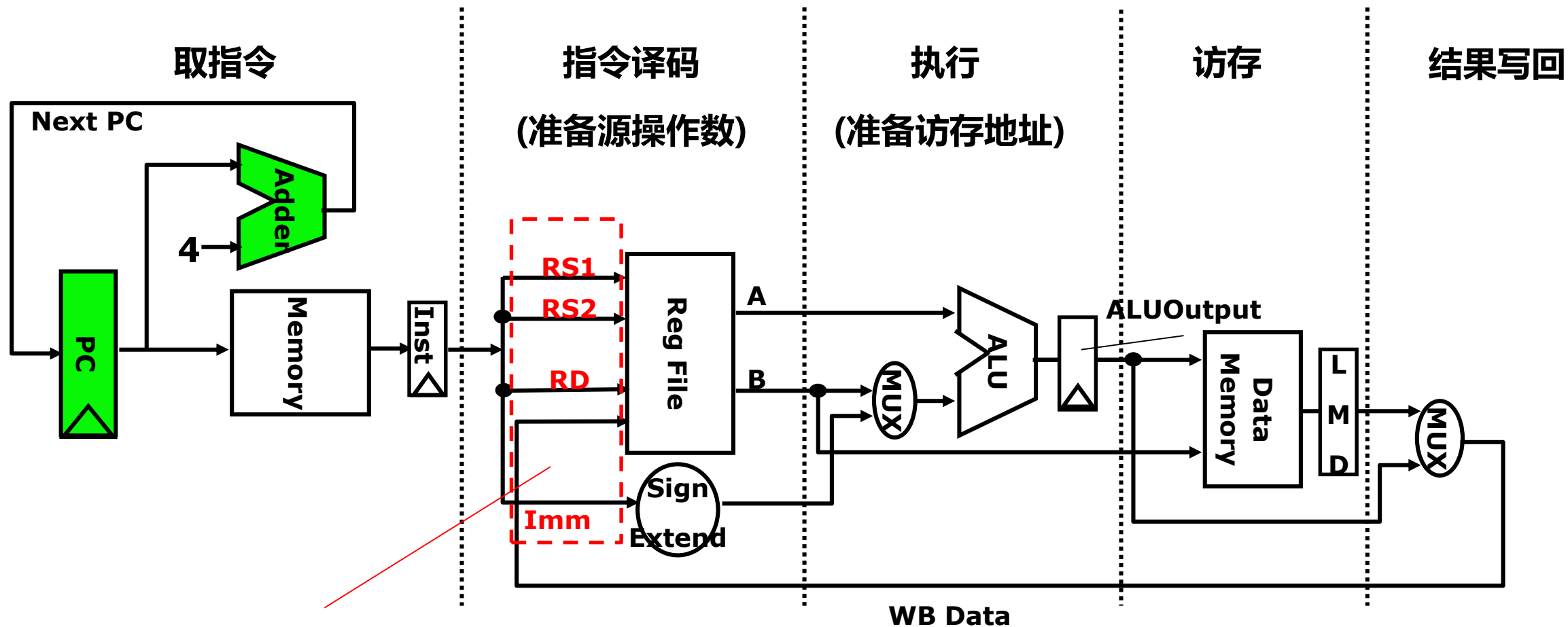
## □合并后的数据通路





# 合并ADD指令和Load指令的数据通路

## 合并后的数据通路



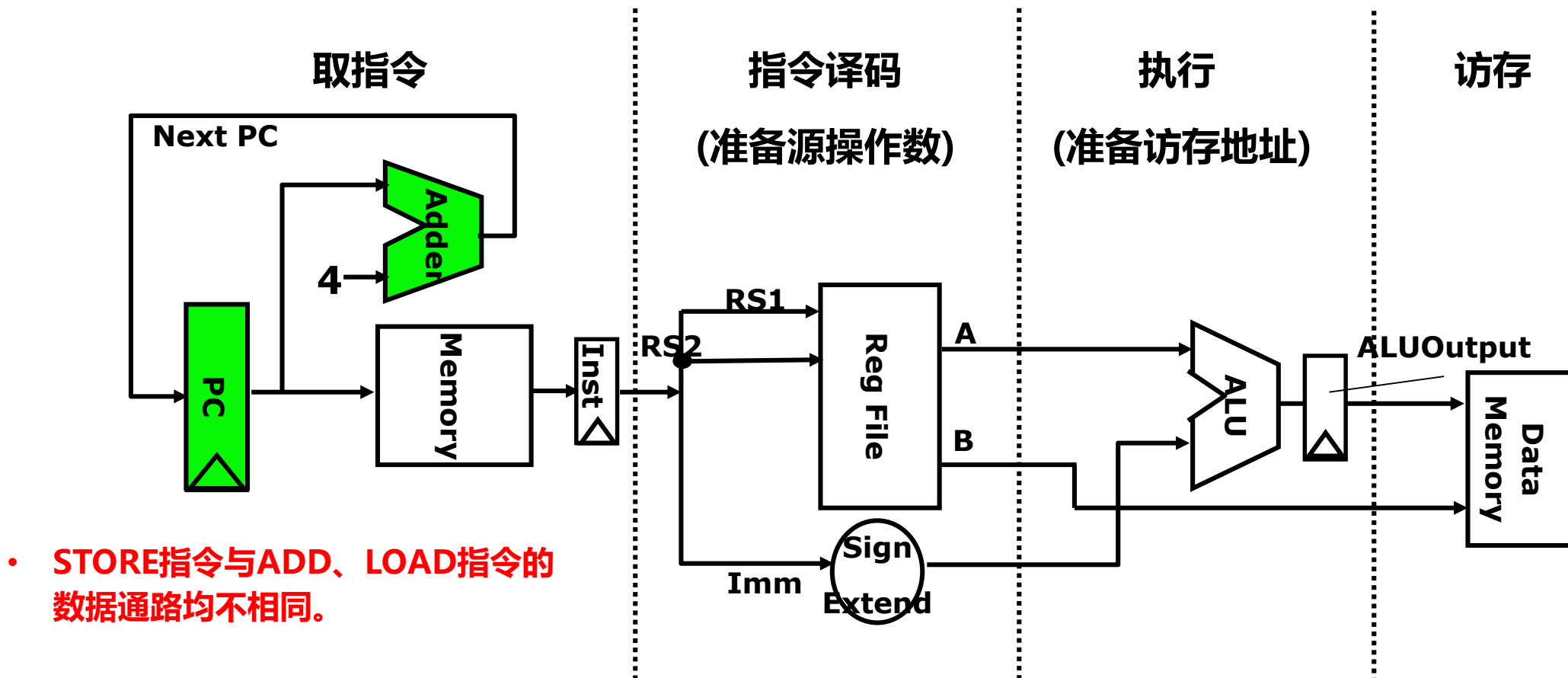
MIPS的“固定字段译码技术”

□以Store指令SW R1, R2, #4为例

## Store指令SW R1 , R2 , #4

功能	$\text{Mem}[\text{R2}+4] \leftarrow \text{R1}$
过程	<ul style="list-style-type: none"><li>① 取指令, <math>\text{Inst} \leftarrow [\text{PC}]</math>, <math>\text{PC} \leftarrow \text{PC}+4</math></li><li>② 指令译码, <math>\text{A} \leftarrow [\text{R2}]</math>, <math>\text{B} \leftarrow [\text{R1}]</math>, 立即数符号扩展为32位</li><li>③ 计算访存地址<math>\text{A}+4</math></li><li>④ 将B写入地址为<math>\text{A}+4</math>的存储单元, <math>\text{Mem}[\text{A}+4] \leftarrow \text{B}</math></li></ul>

## □Store指令的数据通路







# 合并ADD指令和Load指令的数据通路

## 合并后的数据通路

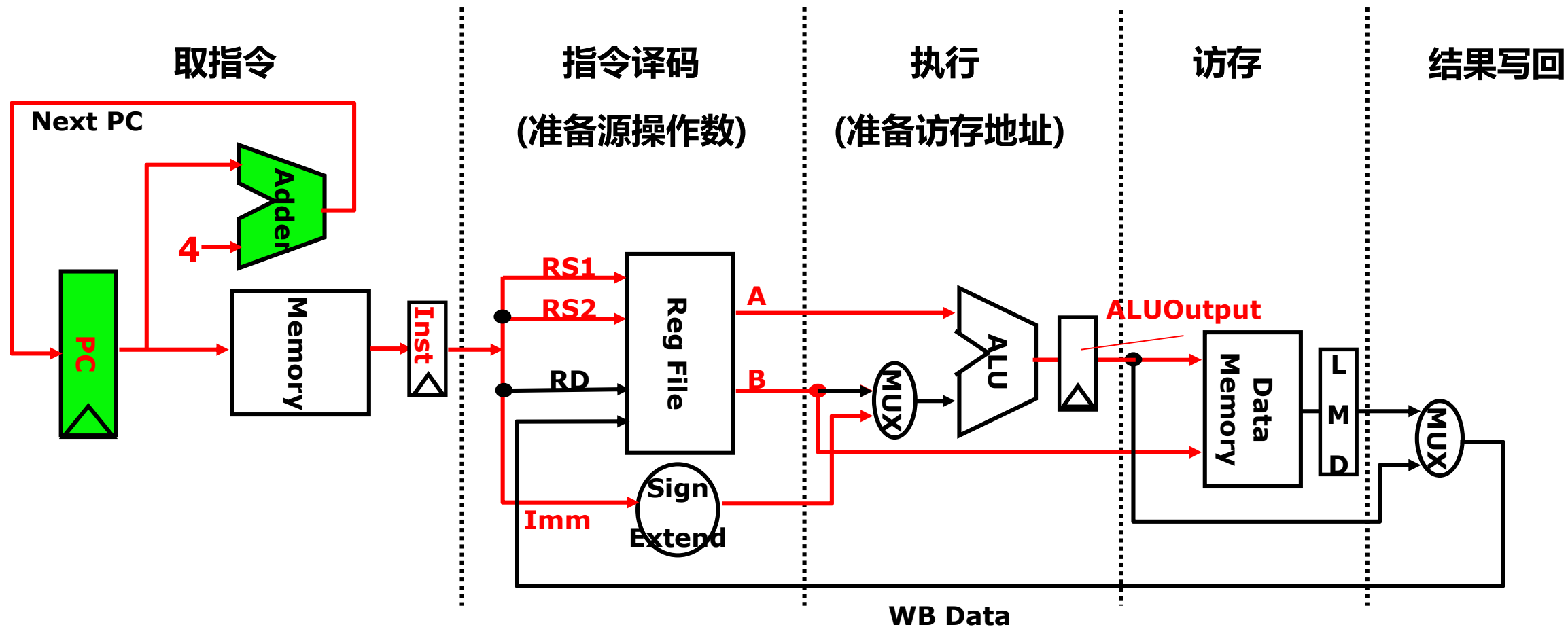






Diagram illustrating the MIPS processor data path, divided into five stages by vertical dashed lines:

- 取指令 (Instruction Fetch):** The PC (Program Counter) provides the address to Memory. Memory outputs the instruction (Inst). The PC is updated to Next PC.
- 指令译码 (Instruction Decode):** The instruction (Inst) is decoded. The Register File (Reg File) provides source registers (RS1, RS2) and the destination register (RD). The instruction also provides the immediate value (Imm). The ALU performs operations on RS1 and RS2, and the Zero flag (Zero?) is set.
- 执行 (Execution):** The ALU performs operations on RS1 and RS2, and the Zero flag (Zero?) is set. The ALU output (ALUOutput) is used to calculate the next PC for branch instructions.
- 访存 (Memory Access):** The ALU output is used to calculate the next PC for branch instructions. The ALU output is also used to calculate the address for Data Memory access.
- 结果写回 (Write Back):** The ALU output is used to calculate the next PC for branch instructions. The ALU output is also used to calculate the address for Data Memory access. The Data Memory outputs the result, which is then written back to the Register File.

Key components and signals shown:

- PC (Program Counter):** Holds the current instruction address.
- Memory:** Provides the instruction (Inst) and data from Data Memory.
- Reg File (Register File):** Provides source registers (RS1, RS2) and the destination register (RD).
- ALU (Arithmetic Logic Unit):** Performs operations on RS1 and RS2, and the Zero flag (Zero?) is set.
- MUX (Multiplexer):** Selects the next PC based on the Zero flag and the branch target address.
- Sign Extend:** Extends the immediate value (Imm) to 32 bits.
- WB Data (Write Back Data):** The result from the ALU is written back to the Register File.

Red lines indicate the data path for branch instructions, showing the flow from the ALU output to the MUX and the next PC.