多周期操作设计

① 取值周期

取值周期从存储器中取出一条指令并计算下一条指令地址 IR<=memory[PC]; PC<=PC+4; 执行操作：为了实现该周期的操作，需要添加几个控制信号。MemRead，IRWrite 并指置 LorD 为 0，选择 PC 输出在地址，要实现 PC+4，需要通过运算其进行 PC+4 结果送回 PC 的运算，置 ALUsrcA=0，ALUsrcB=0， ALUop=00（加法），并将 PCSource=00，置 PCWrite 有效就可以完成操作。PC 的累加和指令的读取可以在同一个周期并行完成。PC 的值在下一个周期更新。

② 指令译码以及寄存器访问周期

之前的周期以及当前都不能明确这条指令具体进行什么操作。在此周期可以进行适合所有指令的操作或不会对指令执行产生损害的操作。因此本周期可以进行指令字段里面指明的两个寄存器 rs 和 rt 的读取，即使后面指令执行过程中可能没有到这两个周期的值。但对于寄存器的读取对后面的执行没有坏处。本周期将他们从寄存器中读出并暂存到寄存器 A 和 B 中。

本周期还完成跳转指令的地址计算，A<=Reg[IR[25:21]]; B<=Reg[IR[20:16]];ALUout<=PC+(sign-extend), (IR[15:0]) 执行操作，从寄存器中读取 rs 和 rt 的值存放在寄存器 A 和 B 中，同时置 ALUsrcA=0,ALUsrcB=11, ALUop=00,计算跳转指令。指令的内容决定了机器执行什么操作。

③ 执行存储器地址计算或跳转操作

这个周期是由指令码确定执行操作的第一个周期，ALU 依据之前周期准备的两个操作数按照指令码所指的计算类型进行计算。具体指令类型有：

1）R-addu，addi，sub，subu，or，nor

ALUout<=A OP B 执行操作：ALU 对先前周期读出的两个寄存器的内容进行计算，置 ALUsrcA=1，ALUsrcB=00，ALUop=10 R-sll，srl，sra，sllu，slt 。Slt：if（A<B）ALUout<=1 else ALUout<=0

其余指令都是 B 移位后得 B~，ALUout<B~

执行操作：ALU 操作前一个寄存器内容对另一个字段的移位，置 ALUsrcA=01，ALUsrcB=10，ALUop 根据 funct 字段确定 R-Jr：PC<=A

执行操作：PC 的值被指令中的跳转地址替换，置 ALUsrcA=01，ALUsrcB=00，ALUop=0011，PCSource=01，PCWrite 信号有效。

2）I-addi，ori，xiri，addiu

ALUout<=A OP (sign-extend) Imme

执行操作：ALUsrcA=1，ALUsrcB=10，ALUop 根据 funct 字段确定。

3）I-beq,bltz,bgtz

If（A OP B）PC<=PC+4+（sign-extend）Imme<<2

执行操作：PC 的值被指令中的跳转地址替换，PCSource 选择跳转地址，PCWrite 信号有效。

4）I-LW,SW

LW：从存储器中读取一个字的数据到寄存器中

SW：把一个字的数据从寄存器中存储到存储器中

执行操作：存储在地址计算 ALUsrcA=01，ALUsrcB=10，ALUop=0000

5）J PC<={(PC+4)[31:28], IR[25:0], Z beo}

执行操作：PC 的值被指令中的跳转地址替换，PCSource 选择跳转地址，PCWrite 信号有效。

④ 存储器访问或 R-type 指令执行周期

该周期是 load 和 store 指令的访存操作，以及算术逻辑指令写回结果的周期，当从存储器中取得一个数据，先把它暂存到 MDR 中，为下一个周期访问做准备。

存储器相关，MDR<=Memory[ALUout] 或者 Memory<=B

执行操作：如果是 Load 指令，一个数据字从存储器中获取并存入到 MDR，如果是 store 指令，该数据写入到存储器中，这种情况的存储器地址都是在前面周期计算完成并存入到 ALUout 中的，如果是 store 指令，源操作数存到 B 中，通过 MemRead 或者 MemWrite 完成相应指令的操作，此外还需要将 LorD 置 1 指定在存储器地址来自 ALU 而不是 PC 算术逻辑指令（R-type）

Reg[IR[15:11]]<=Aluout

执行操作：将 ALUout 的值写入结果寄存器中，RegDst=1，RegWrite 信号有效，将 MemtoReg 置 0

⑤ 存储器读结束周期

该周期从在存储器读取的值写回完成 Load 指令；

Load 指令：Reg[IR[20:16]]<=MDR;

执行操作：MemtoReg=1, RegWrite 有效，RegDst=0