Arquitetura de Computadores
Aulas Práticas 2021/2022

Licenciatura em Engenharia Informática e Computação
FCUP/FEUP – Universidade do Porto

Implementação AArch64 (II)

1. Tendo por base o CPU da implementação multi-ciclo da arquitetura MIPS apresentado na Figura 1 e a correspondente máquina de estados apresentada na Figura 3, siga o trajecto do fluxo da informação correspondente à execução das seguintes instruções MIPS conforme os passos descritos na Figura 2 (use como referência a resolução apresentada nos slides 58 e 59 das aulas teóricas):

   (a) `lw $t2, 100($t1)` (instrução de *load*: `$t2 = MEM[$t1 + 100]`)

   (b) `sw $t2, -12($t1)` (instrução de *store*: `MEM[$t1 -12] = $t2`)

   (c) `add $t3, $t1, $t2` (instrução de adição: `$t3 = $t1 + $t2`)

   (d) `beq $t1, $t2, _label` (instrução de salto condicional: `if ($t1==$t2) PC = _label`)

   (e) `j _label` (instrução de salto incondicional: `PC = _label`)

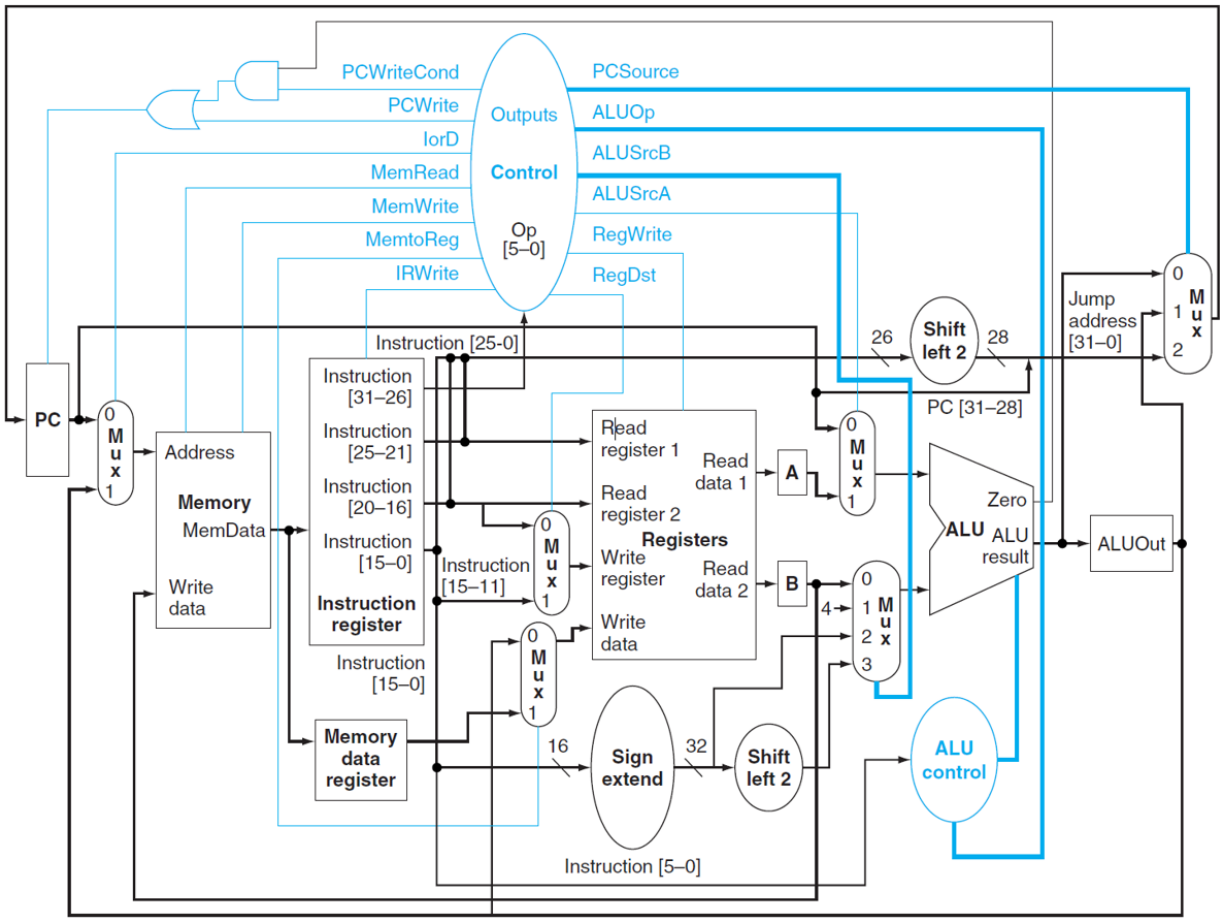

Figure 1: CPU da implementação multi-ciclo da arquitetura MIPS

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR <= Memory[PC] PC <= PC + 4			
Instruction decode/register fetch	A <= Reg [IR[25:21]] B <= Reg [IR[20:16]] ALUOut <= PC + (sign-extend (IR[15:0]) << 2)			
Execution, address computation, branch/jump completion	ALUOut <= A op B	ALUOut <= A + sign-extend (IR[15:0])	if (A == B) PC <= ALUOut	PC <= {PC [31:28], (IR[25:0]),2'b00)}
Memory access or R-type completion	Reg [IR[15:11]] <= ALUOut	Load: MDR <= Memory[ALUOut] or Store: Memory [ALUOut] <= B		
Memory read completion		Load: Reg[IR[20:16]] <= MDR		

Figure 2: Passos da implementação multi-ciclo da arquitetura MIPS

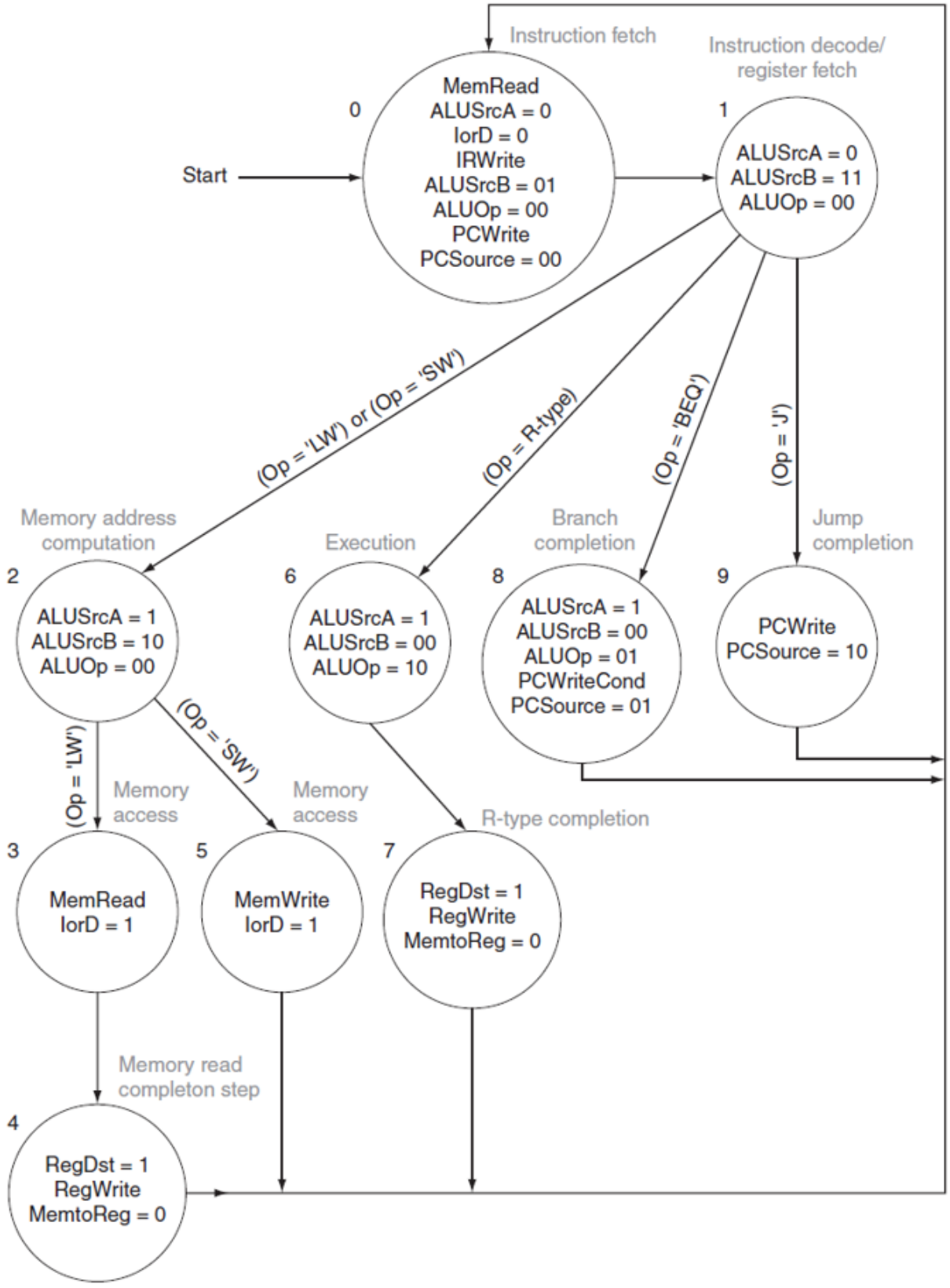

Figure 3: Máquina de estados da implementação multi-ciclo da arquitetura MIPS