1) sll → Shift Left Logic

TipoR → sll rd, rt, shamt

(rd) = (rt) << shamt

(rs = 0, ignorado)

TipoR: | 31:26 OPCODE 000000 | 25:21 RS 0 | 20:16 RT | 15:11 RD | 10:6 shamt | 5:0 Function 000000 |

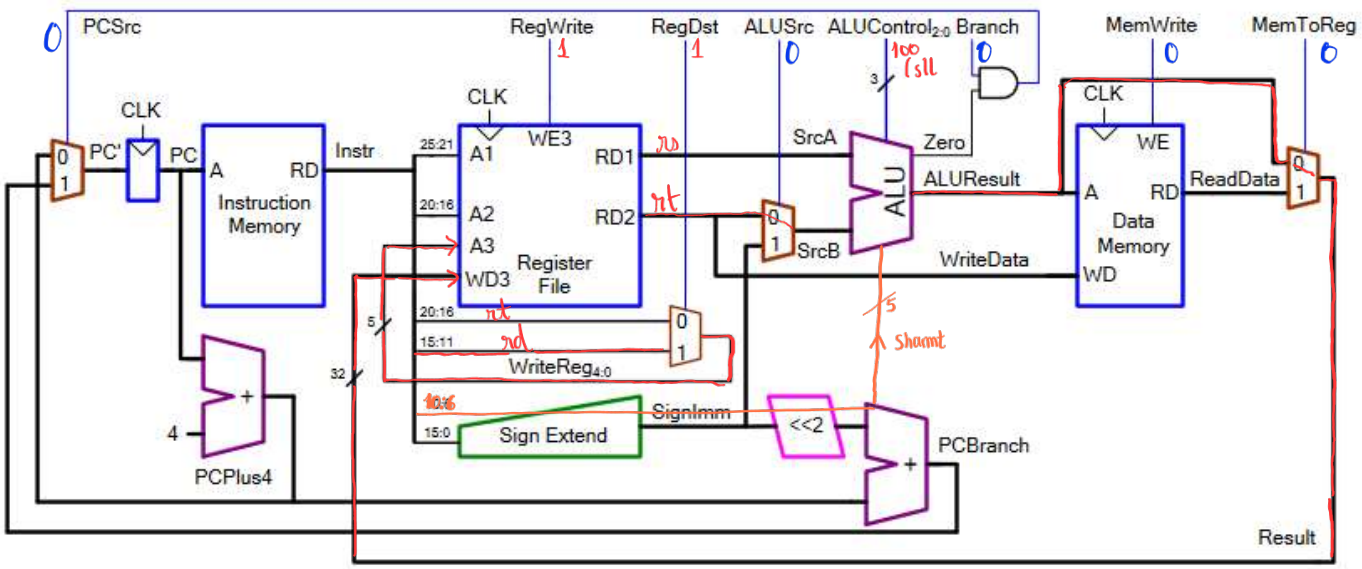


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)
10	000000 (sll)	100 (sll)

Tabela II - Descodificador da ALU

sll → Shift Left Logic

TipoR → sll rd, rt, shamt

(rd) = (rt) << shamt

(rs = 0, ignorado)

TipoR: | 31:26 OPCODE 000000 | 25:21 RS 0 | 20:16 RT | 15:11 RD | 10:6 shamt | 5:0 Function 000000 |

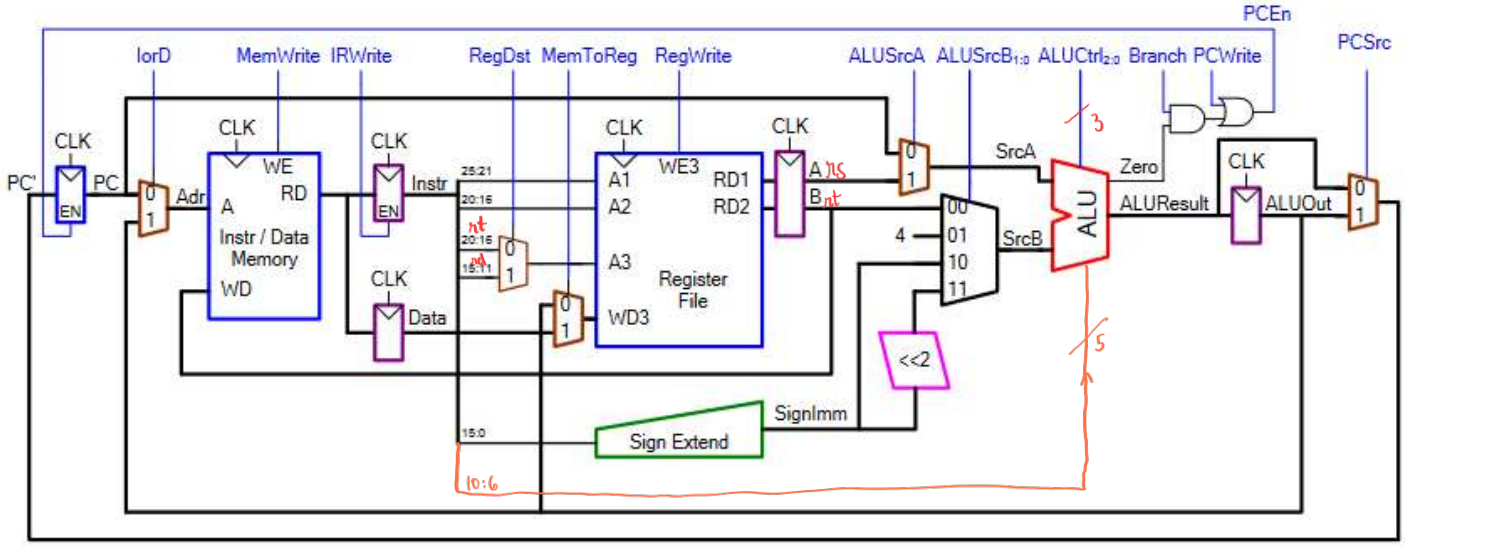
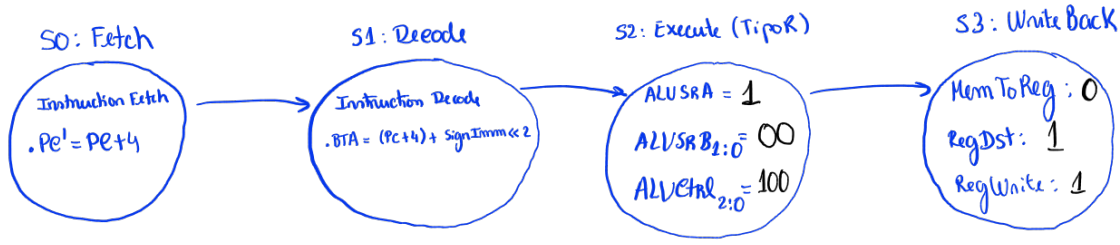


Figura 2 - *Datapath multicycle*



2) lui Tipo I → Load Upper Immediate ⇒ Carregar uma constante de 16 bits nos bits 16 a 31 de um registo de 32 bits.

lui rt, imm

Tipo I → OPCODE | RS | RT | imm16: ctte ou endereço

0

rt = (imm16) << 16

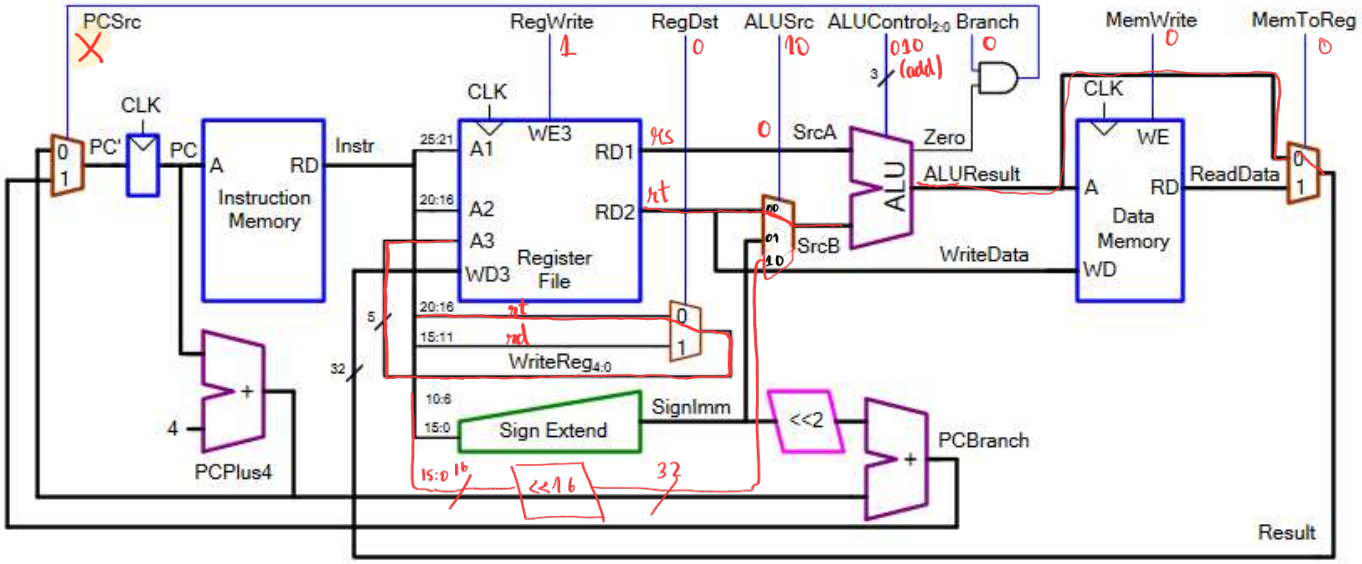


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)

Tabela II - Descodificador da ALU

lui rt, imm

rt = (imm16) << 16

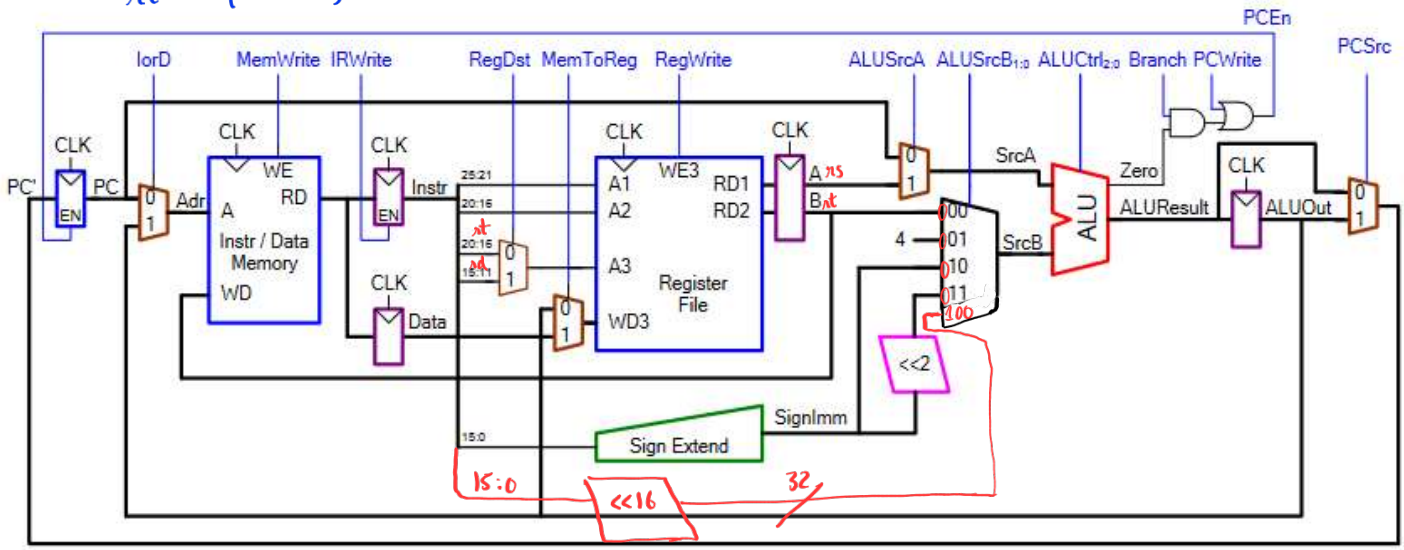
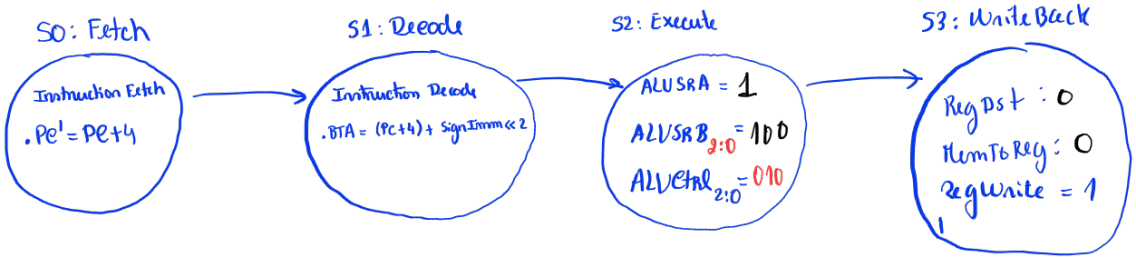


Figura 2 - *Datapath multicycle*



Tipo I

Zero Extended

ori rt, rs, imm16

Opcode	rs	rt	imm16

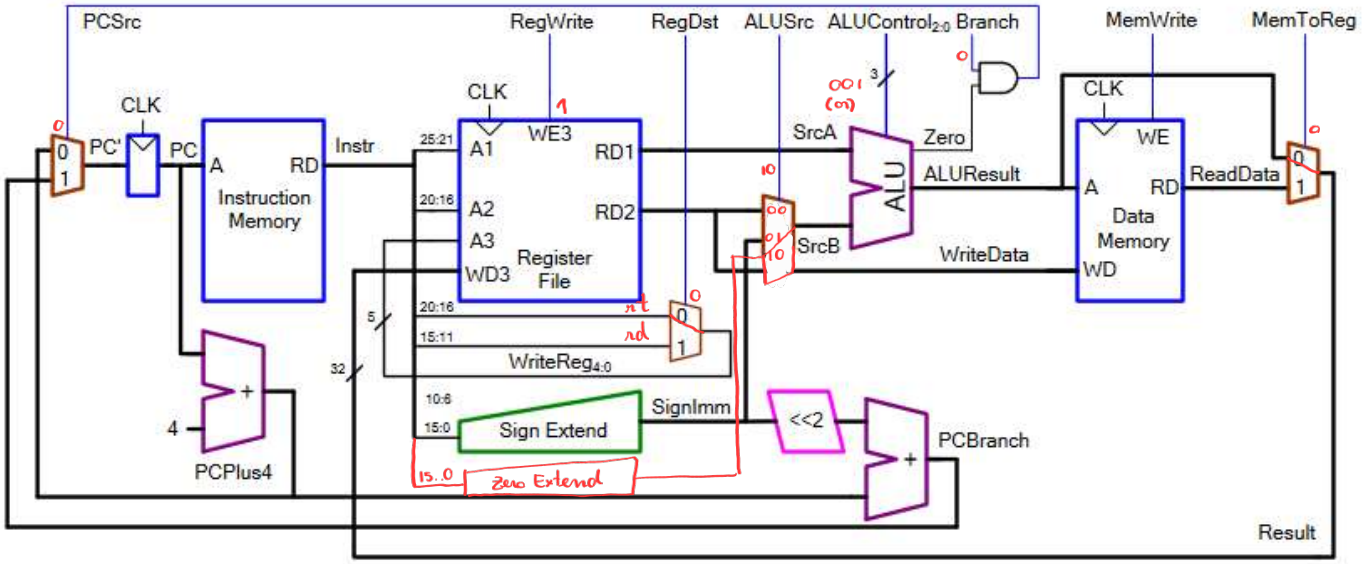


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)
11	XXXXXX	001

Tabela II - Descodificador da ALU

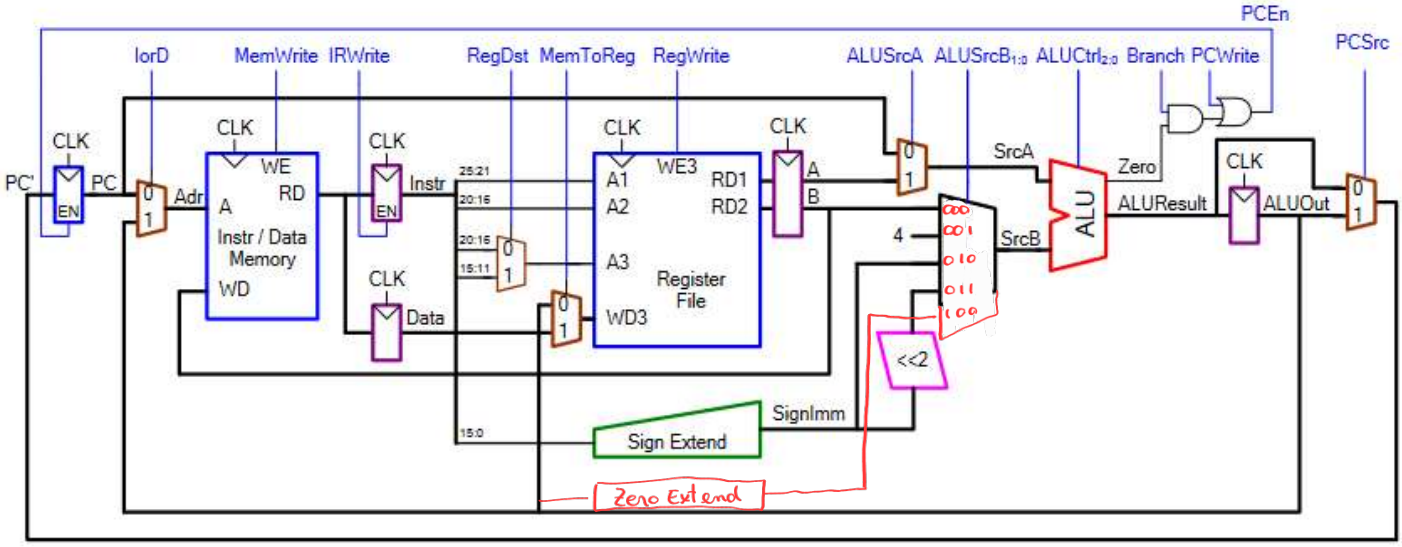


Figura 2 - *Datapath multicycle*

3) ORi → OR immediate

ori rt, rs, imm → zero-extended!

Tipo I: OPCODE RS RT imm16: cte ou endereço

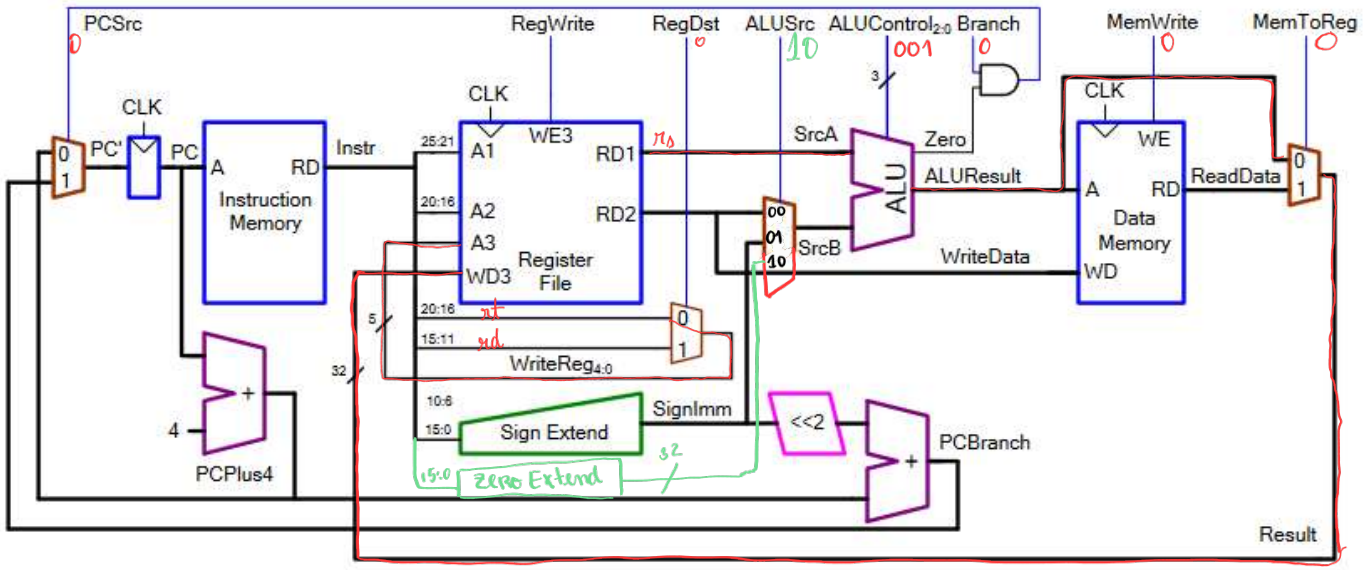


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)
11	XXXXXX	001 (OR)

Tipo I →

Tabela II - Descodificador da ALU

ORi → OR immediate

ori rt, rs, imm → zero-extended!

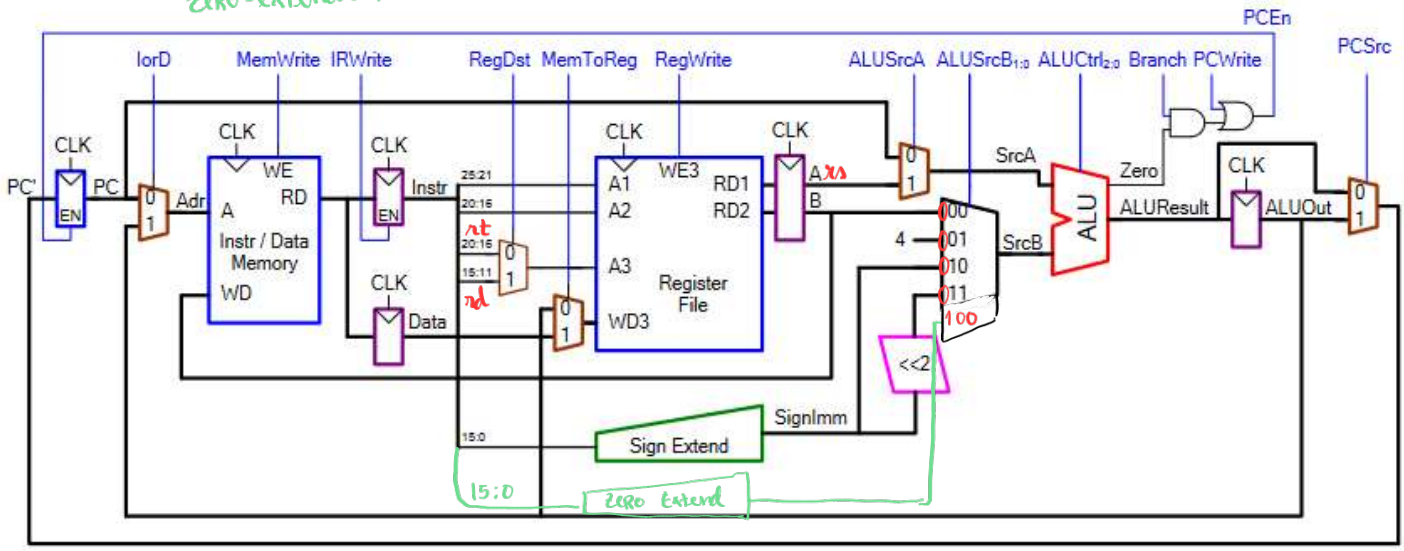
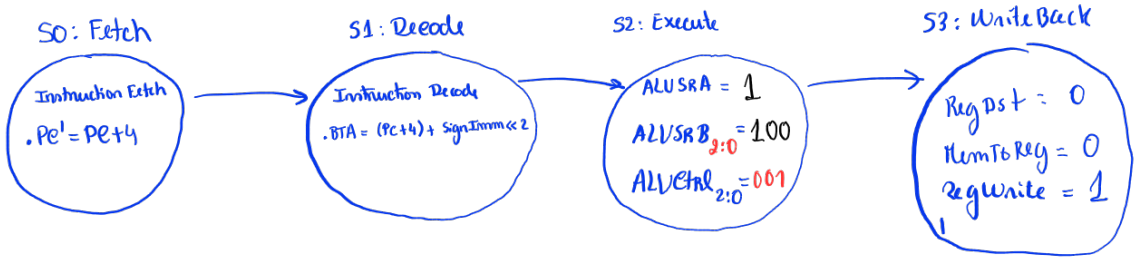


Figura 2 - *Datapath multicycle*



4) Tipo I) Sign Extended

slti rt, rs, imm → set less than immediate

- Compara rs com o valor do imediato
- se RS for menor → rt = 1

Tipo I: OPCODE RS RT imm16 : cte ou endereço

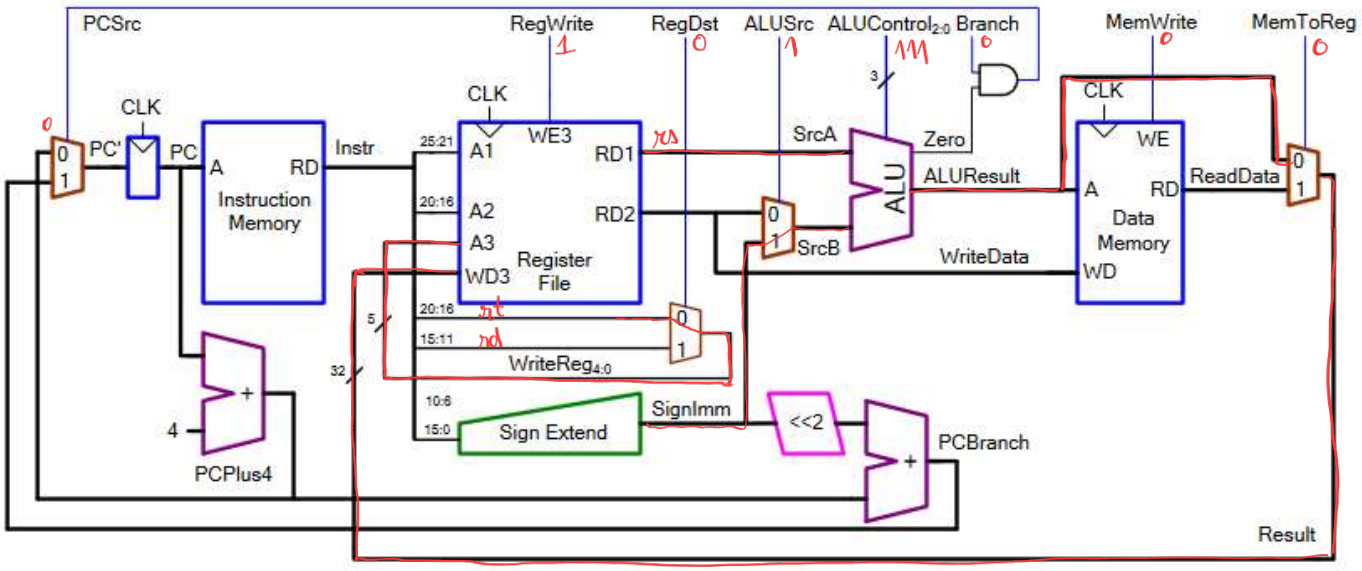


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)
11	xxx xxx	111 (slt)

Tabela II - Descodificador da ALU

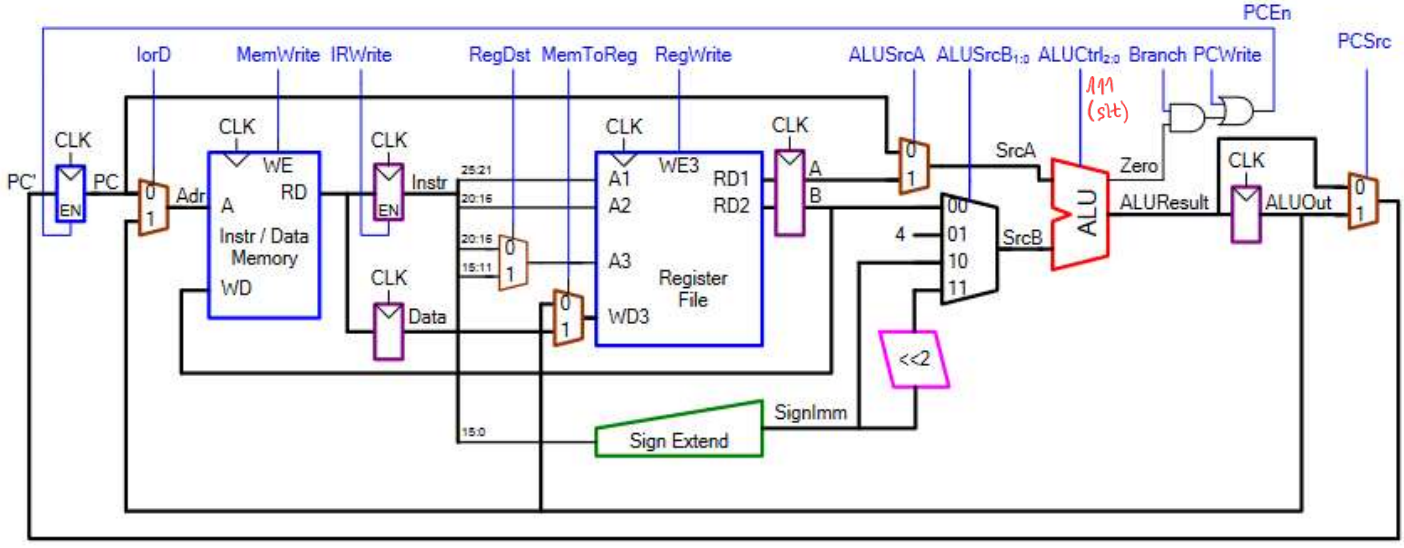
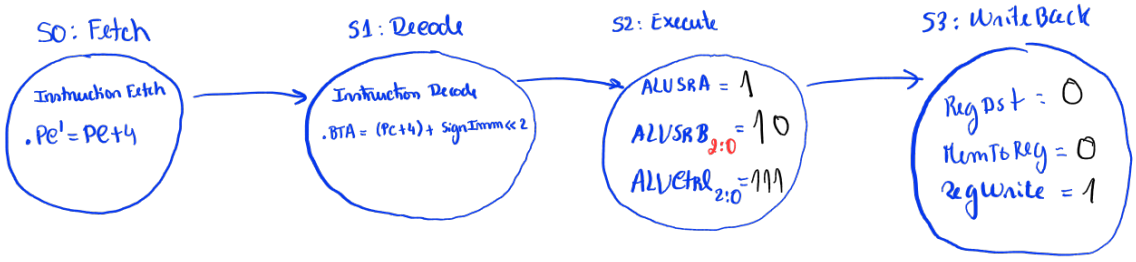


Figura 2 - *Datapath multicycle*



5) Tipo R jR → jR rs jump register

Desvio incondicional para o endereço contido no registo. (Pega nesse endereço (32 bits) e coloca-o no PC+4)!

Exemplo: jR $Ra

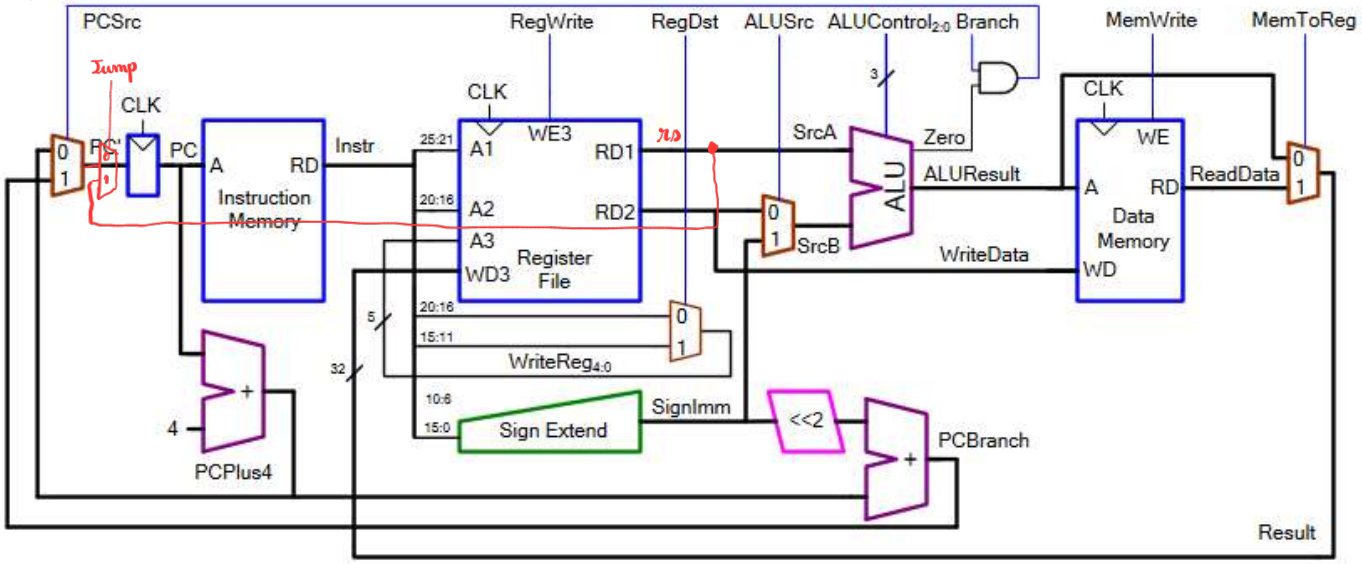


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)
10	001000 (jR)	010 (Add)

Tabela II - Descodificador da ALU

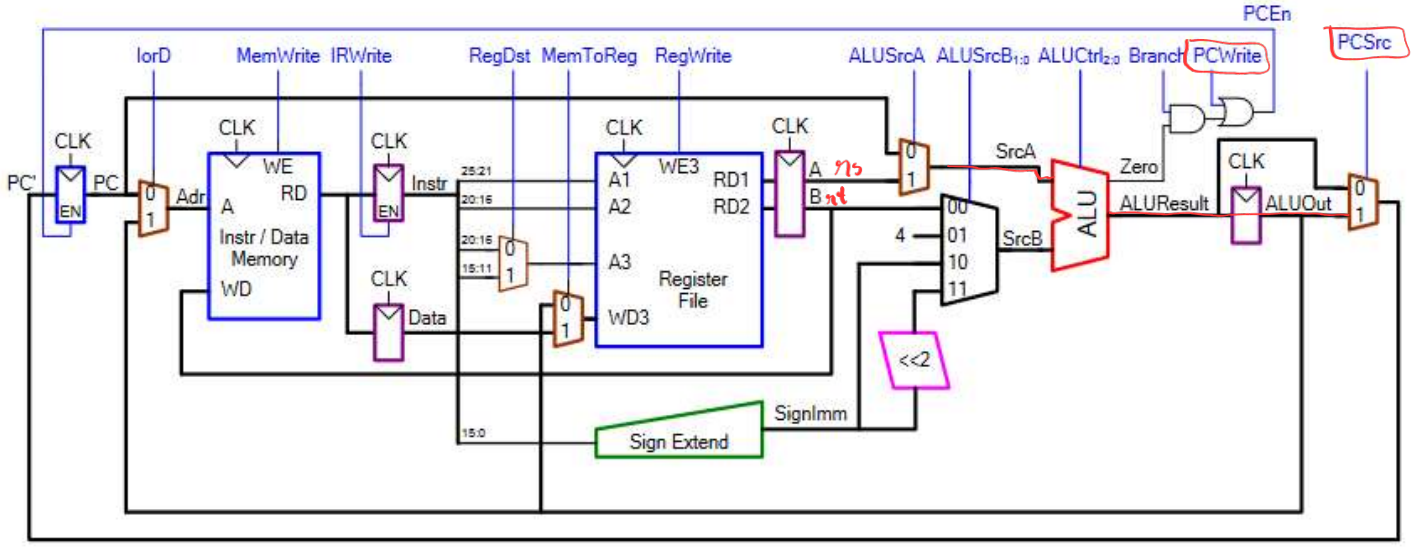


Figura 2 - *Datapath multicycle*

6) Tipo J → `j label` → jump

Salto incondicional para um endereço específico //

! $JTA = (PC+4)_{31:28} : (Imm_{26} \ll 2)$ Tenho q colocar no PC'

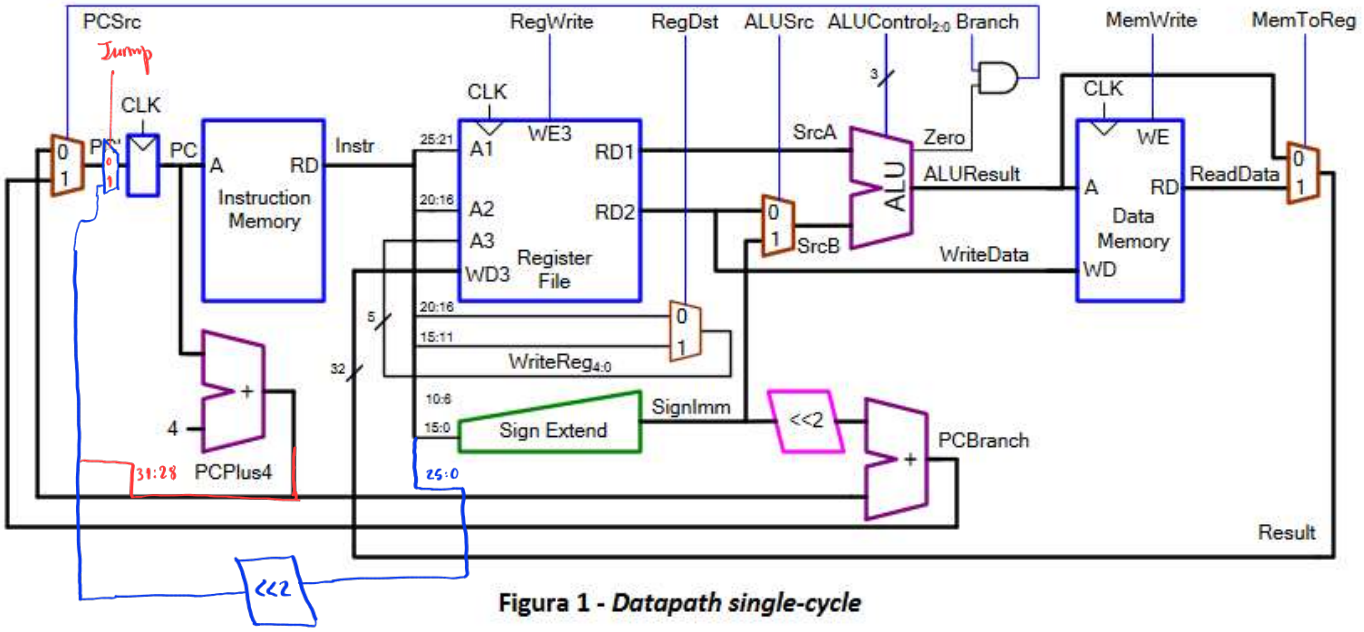


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)

Tabela II - Descodificador da ALU

! $JTA = (PC+4)_{31:28} : (Imm_{26} \ll 2)$ → PC'

Cálculo feito pela ALU

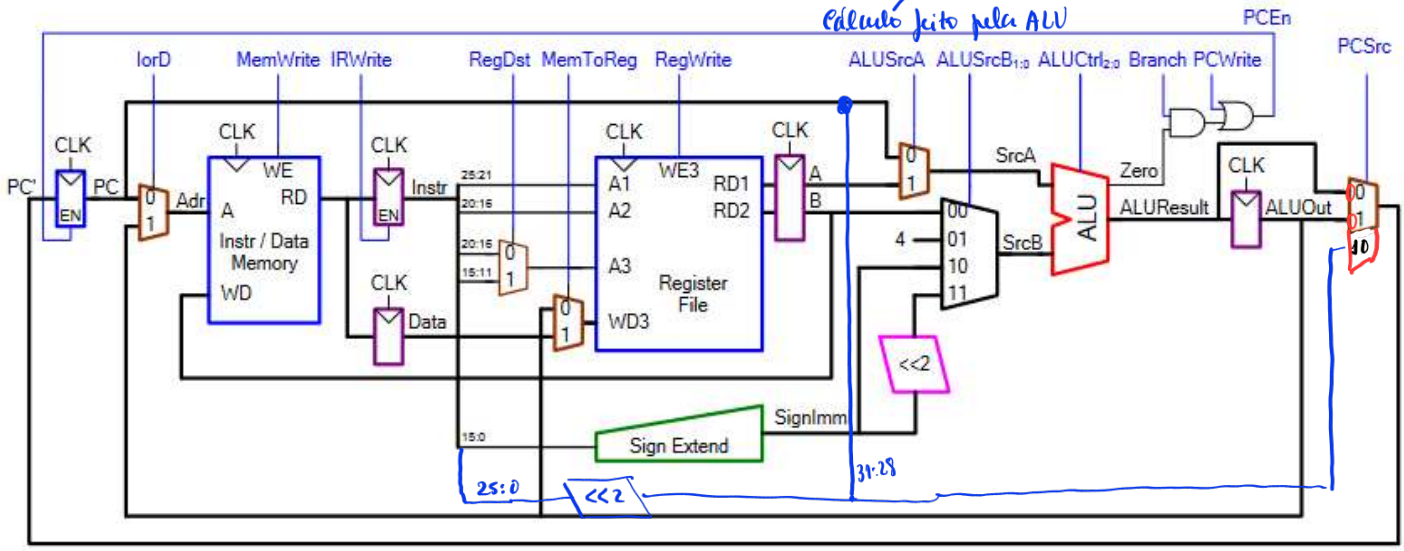
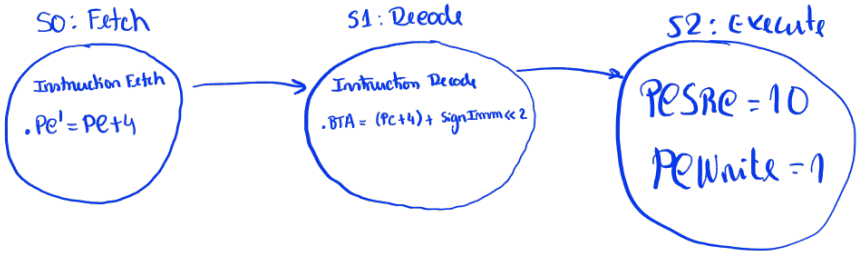


Figura 2 - *Datapath multicycle*



Tipo I

jal imm 26

Opcode	imm 26
31..26	25..0

- JTA = (PC+4)$_{31..28}$: (imm26 << 2)
- $ra = (PC+4)

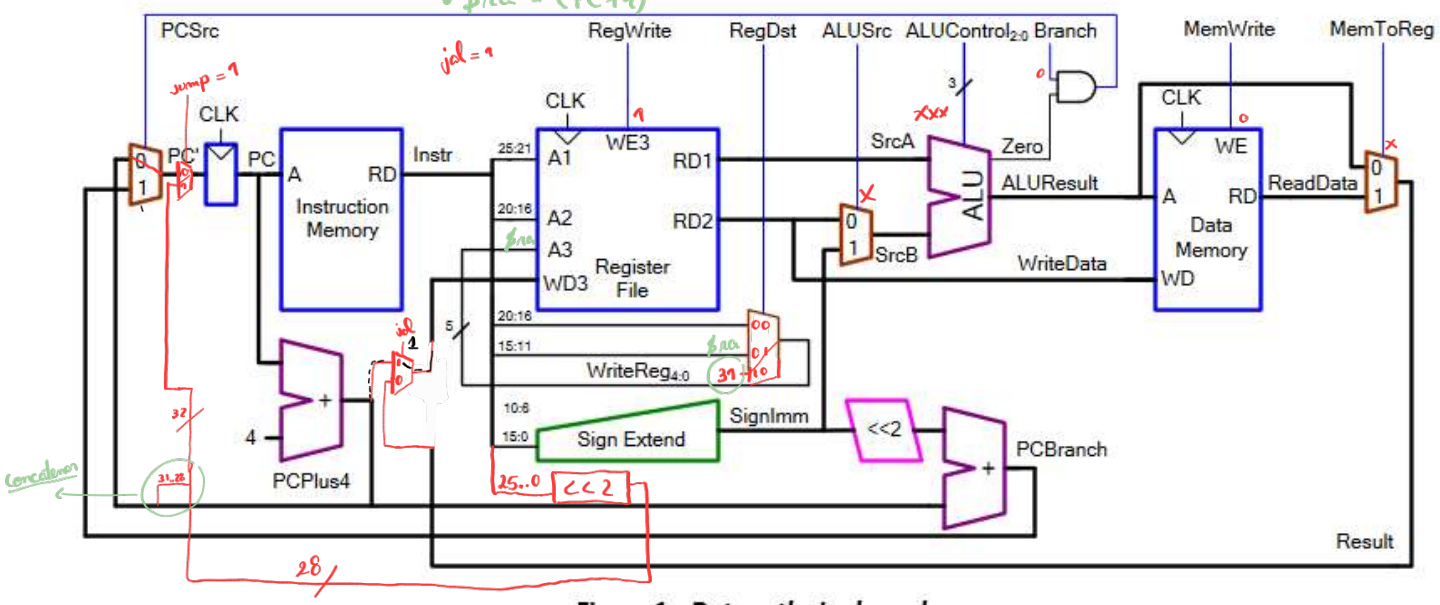


Figura 1 - *Datapath single-cycle*

$ALUOp_{1:0}$	$Funct_{5:0}$	$ALUControl_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)

Tabela II - Descodificador da ALU

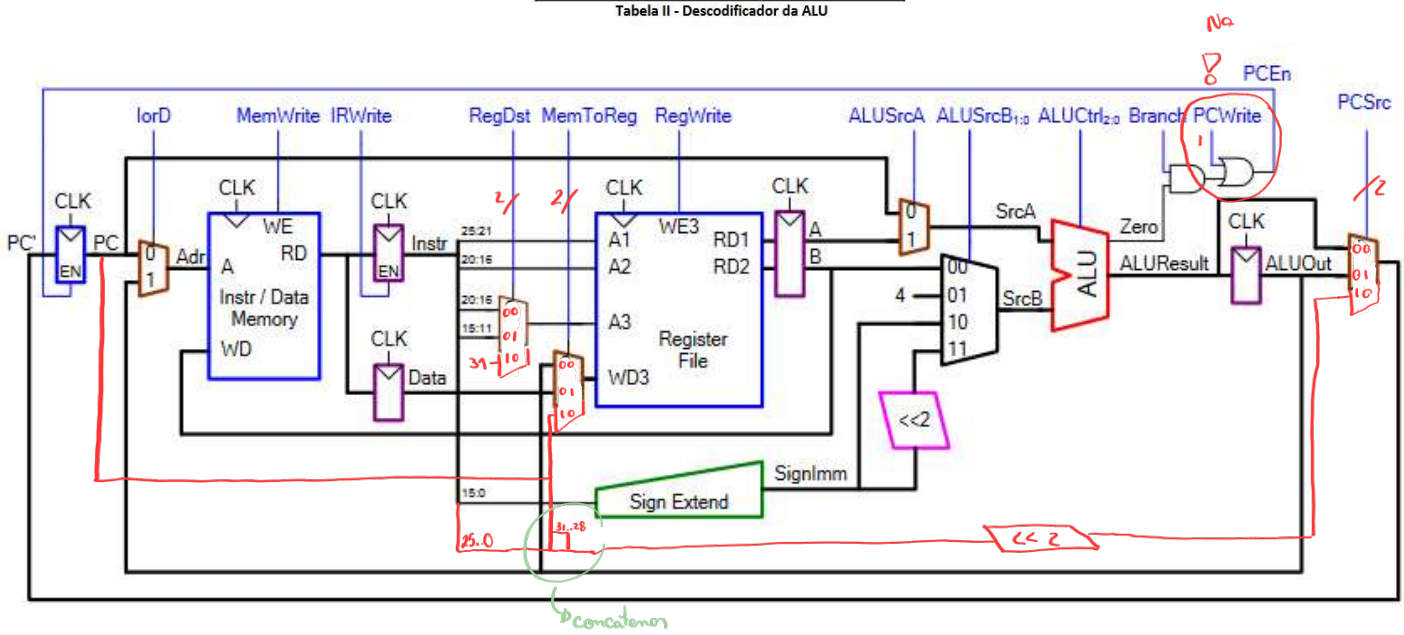
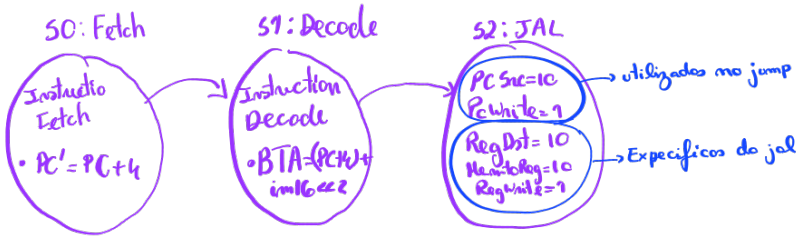


Figura 2 - *Datapath multicycle*



7) Jal → Tipo J : jal label

jump and link

jump → JTA = (PC+4)$_{31:28}$: (Imm26 << 2)

and link → $RA = (PC+4)

Onde vou escrever — O que vou escrever!

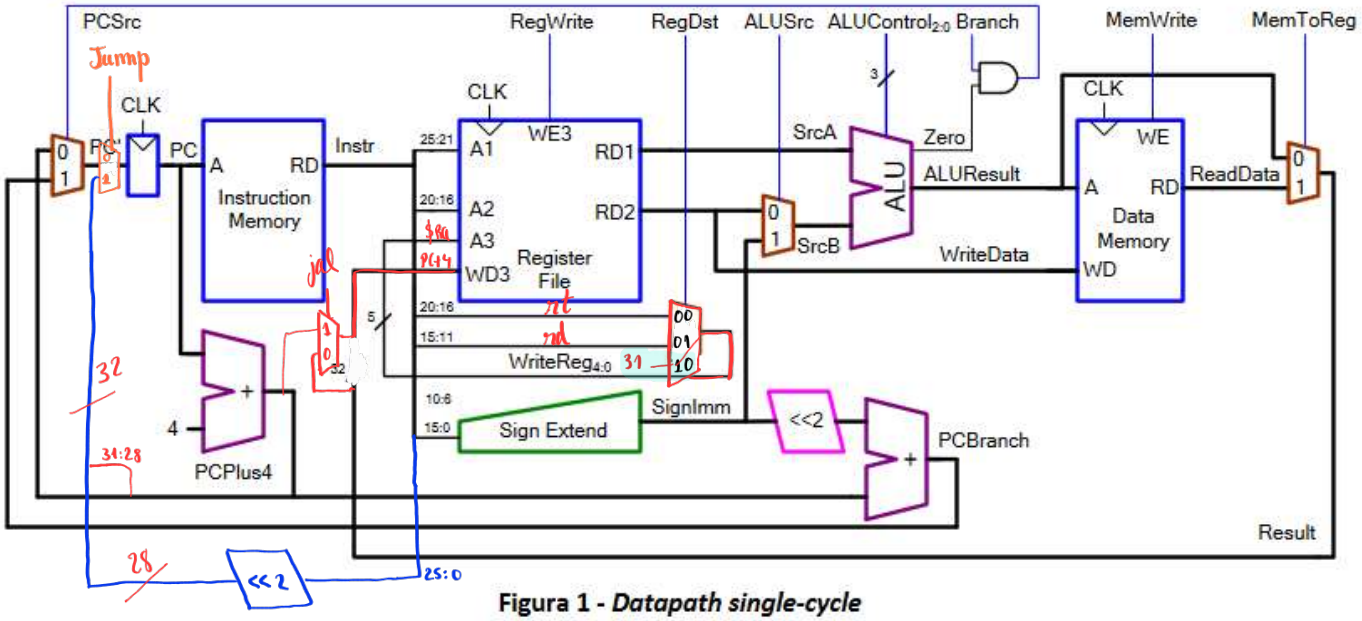


Figura 1 - *Datapath single-cycle*

ALUOp$_{1:0}$	Funct$_{5:0}$	ALUControl$_{2:0}$
00	XXXXXX	010 (Add)
01	XXXXXX	110 (Subtract)
10	100000 (add)	010 (Add)
10	100010 (sub)	110 (Subtract)
10	100100 (and)	000 (And)
10	100101 (or)	001 (Or)
10	101010 (slt)	111 (Slt)

Tabela II - Descodificador da ALU

jump → JTA = (PC+4)$_{31:28}$: (Imm26 << 2)

and link → $RA = (PC+4)

Onde vou escrever — O que vou escrever!

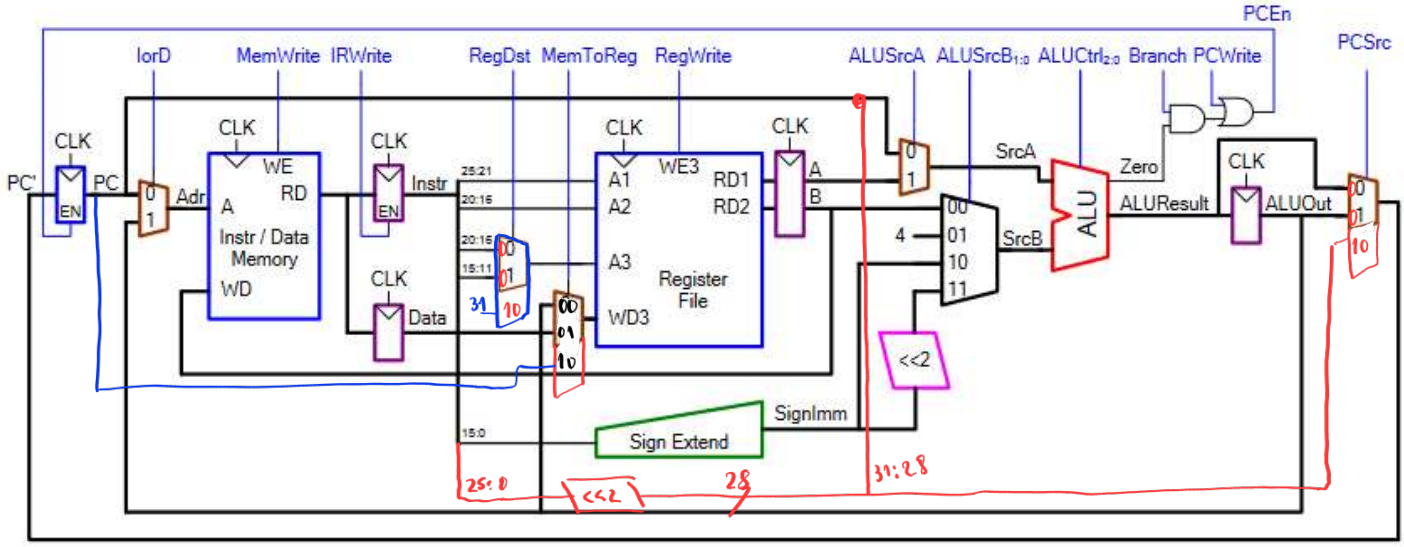


Figura 2 - *Datapath multicycle*