Arquitetura de um processador

2. Controle no caminho de dados

Prof. John L. Gardenghi
Adaptado dos slides do livro

Controle da ULA

- A ULA é utilizada para:
  - Load/Store: F = add
  - Desvio (beq): F = subtract
  - Instruções tipo R: F depende do campo `funct`

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

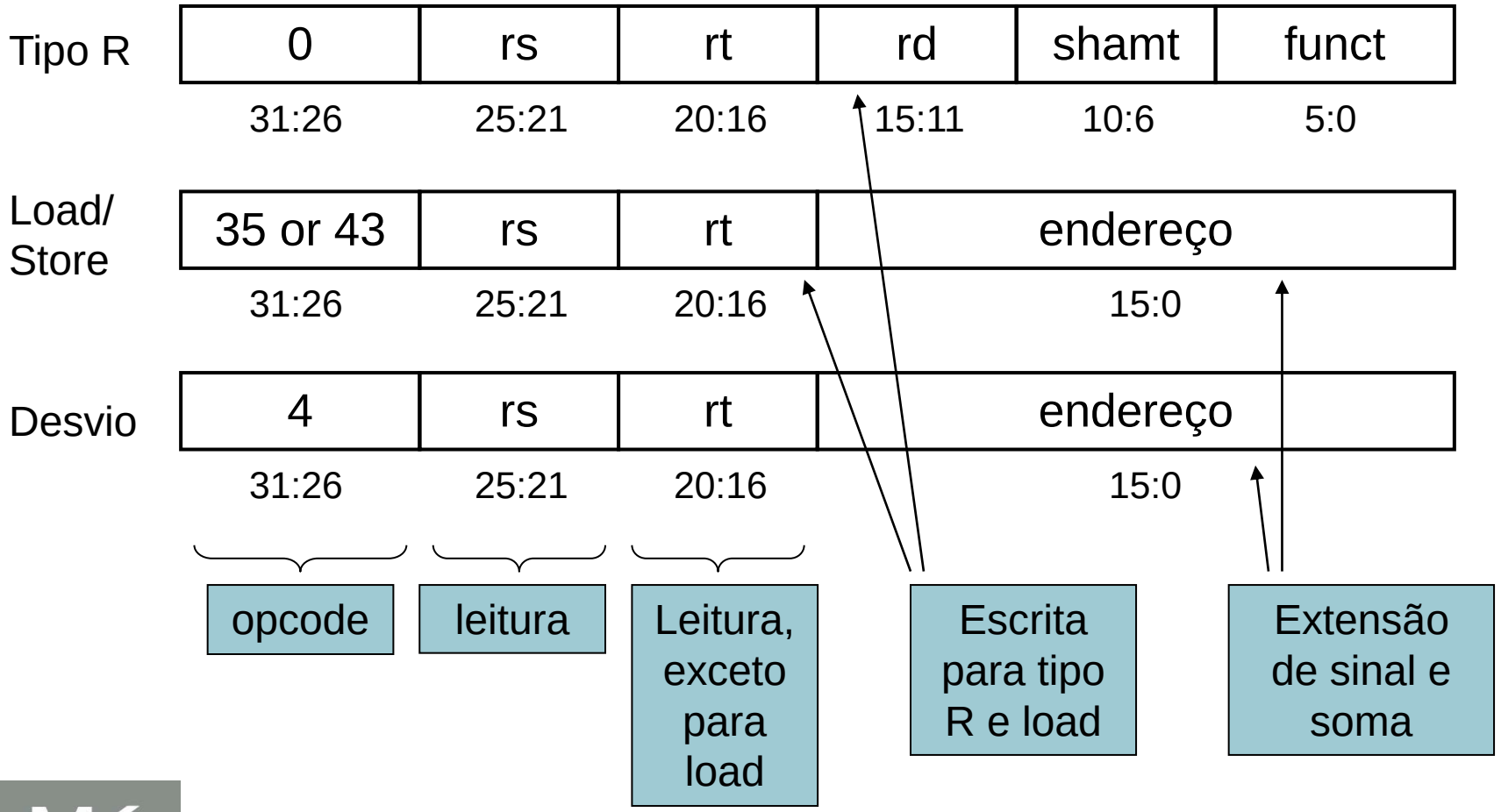
Controle da ULA

- ALUOp = opcode
  - Uma combinação deste com funct gera o código de controle da ULA

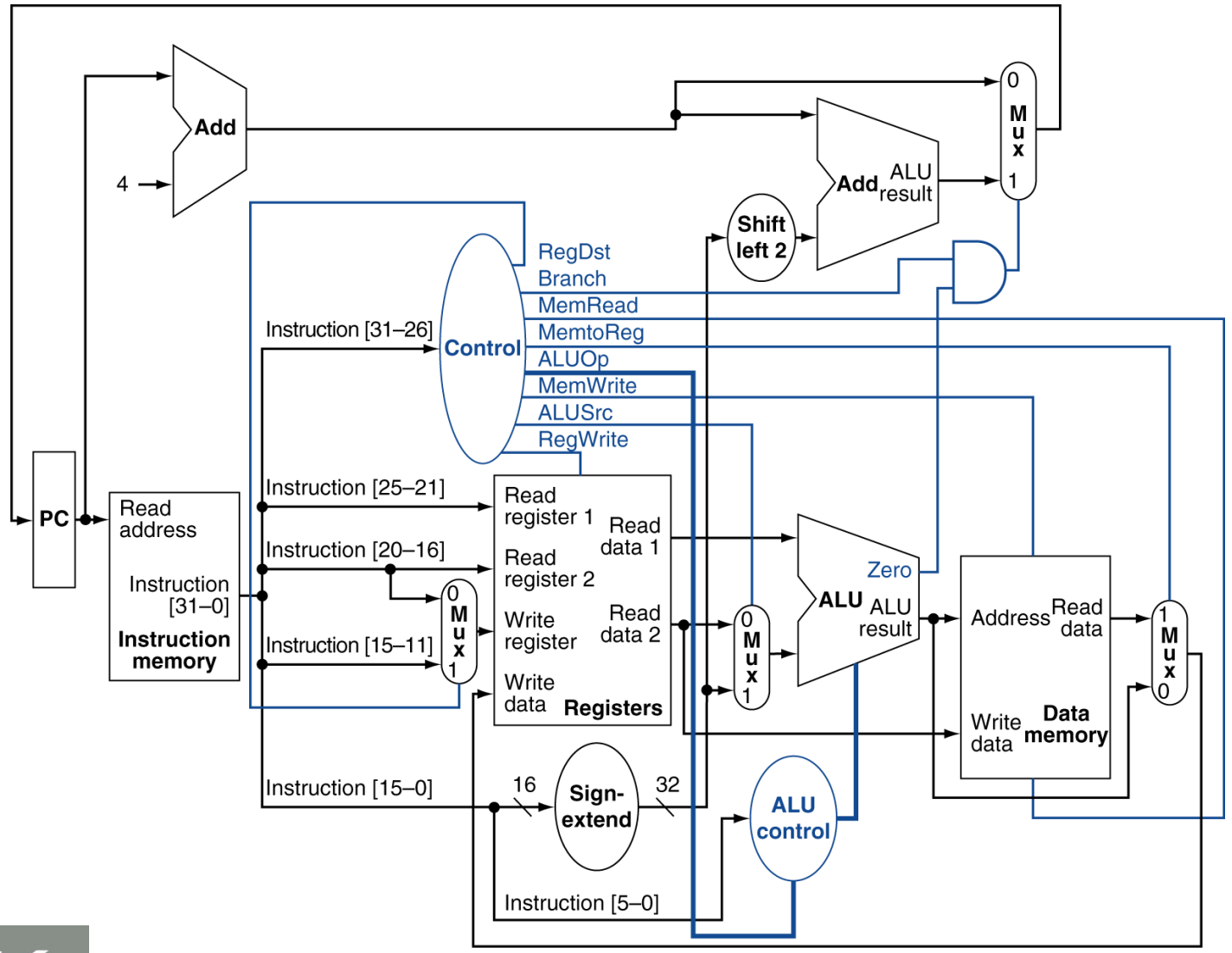
opcode	ALUOp	Operação	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

A unidade de controle principal

- Os sinais de controle derivam da instrução

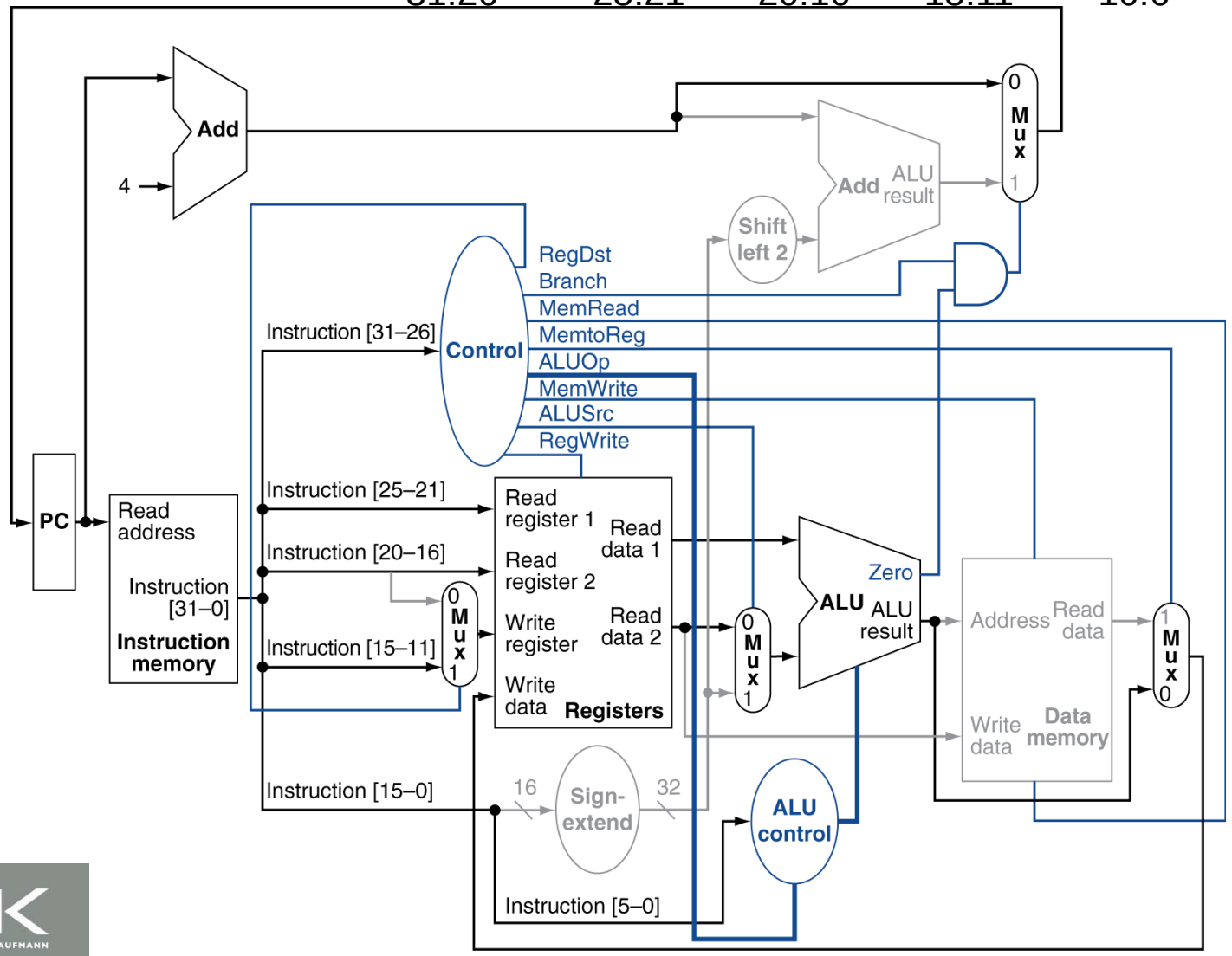


Caminho de dados com controle

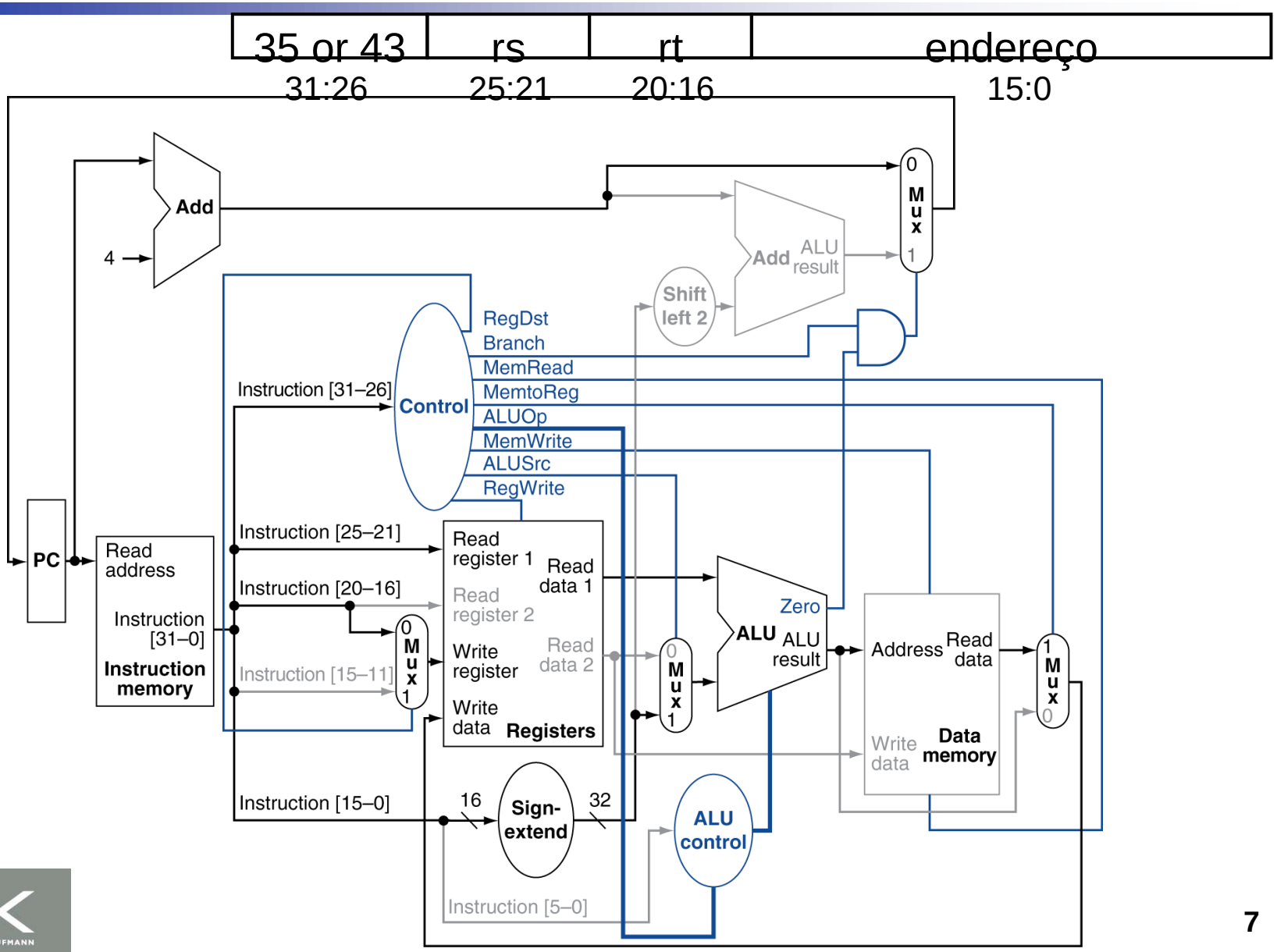


Instruções do tipo R

0	rs	rt	rd	shamt	funct
31:26	25:21	20:16	15:11	10:6	5:0

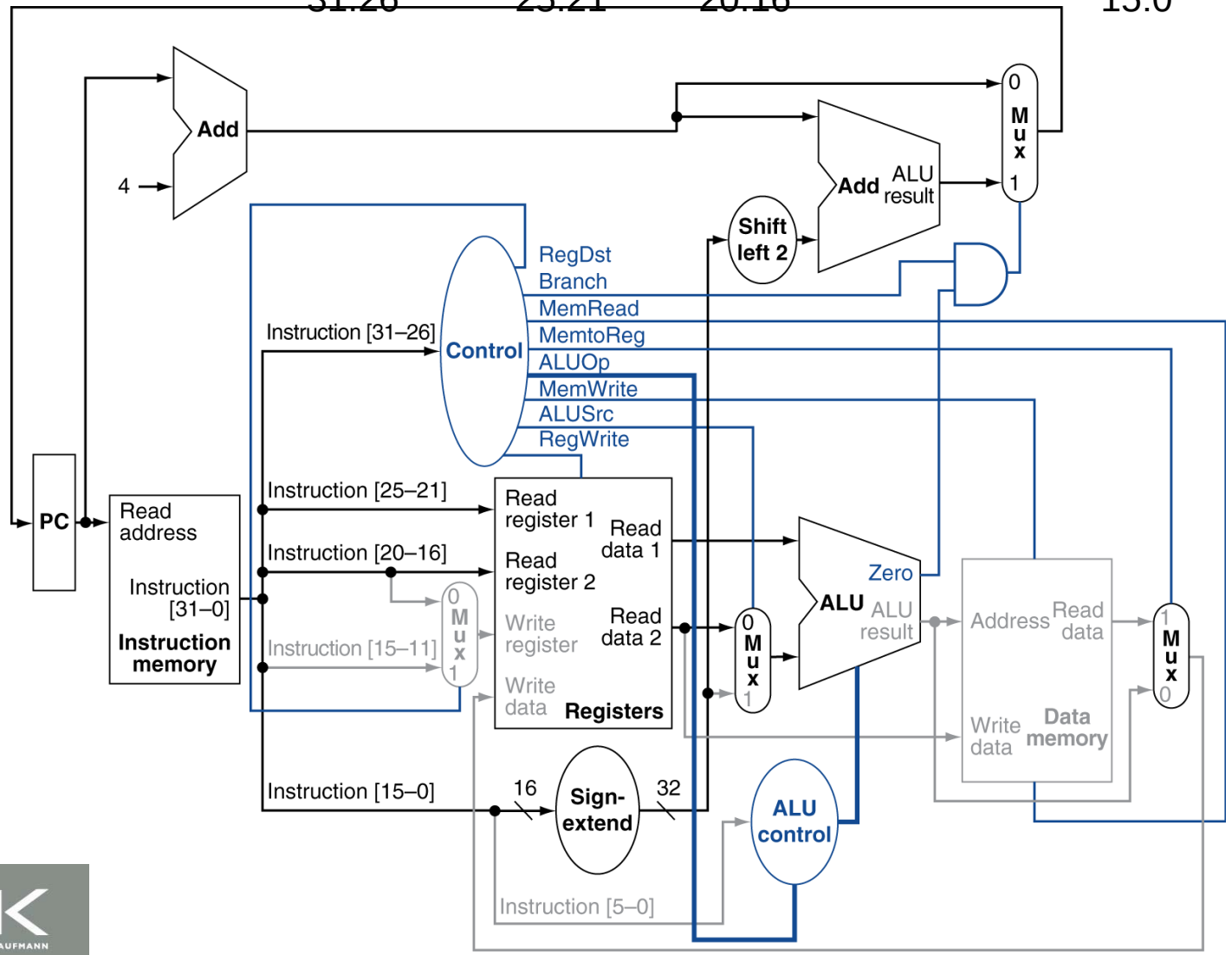


Instruções load

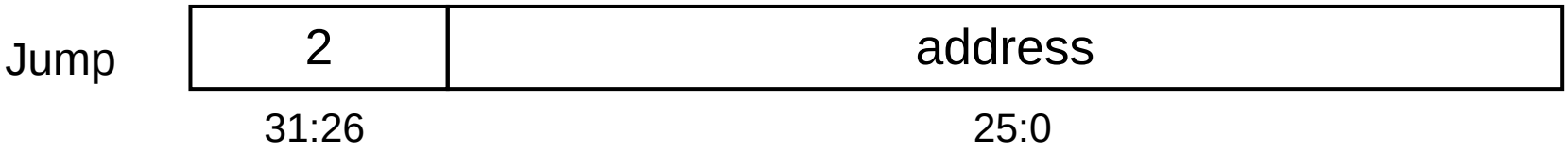


Instruções beq

4	rs	rt	endereço
31:26	25:21	20:16	15:0

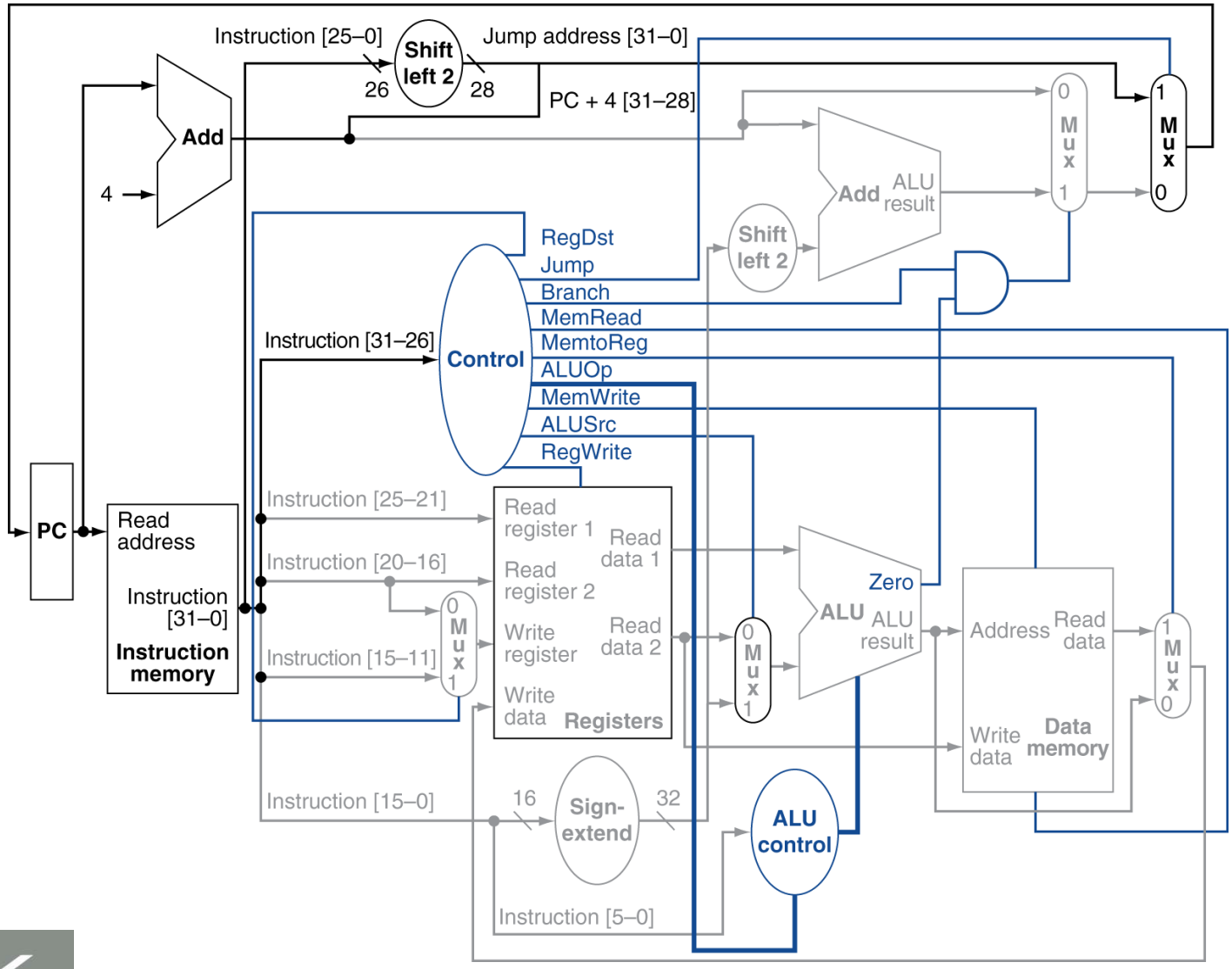


Implementando Jumps



- Jump usa palavras como endereçamento
- Atualiza o PC com a concatenação de
  - Os 4 bits mais significativos do PC
  - O endereço de 26-bits da instrução
  - 00 (por causa do deslocamento por palavra)
- É necessário um sinal de controle adicional, extraído do opcode

Caminho de dados com jump



Problemas de desempenho

- A instrução mais demorada determina o período de clock
  - Caminho crítico: instrução load
  - Memória de instrução → banco de registradores → ULA → memória de dados → banco de registradores
- Entretanto, outras instruções são mais rápidas
- Isso será melhorado com pipeline