Bloco Operacional Multiciclo

Luciano L. Caimi

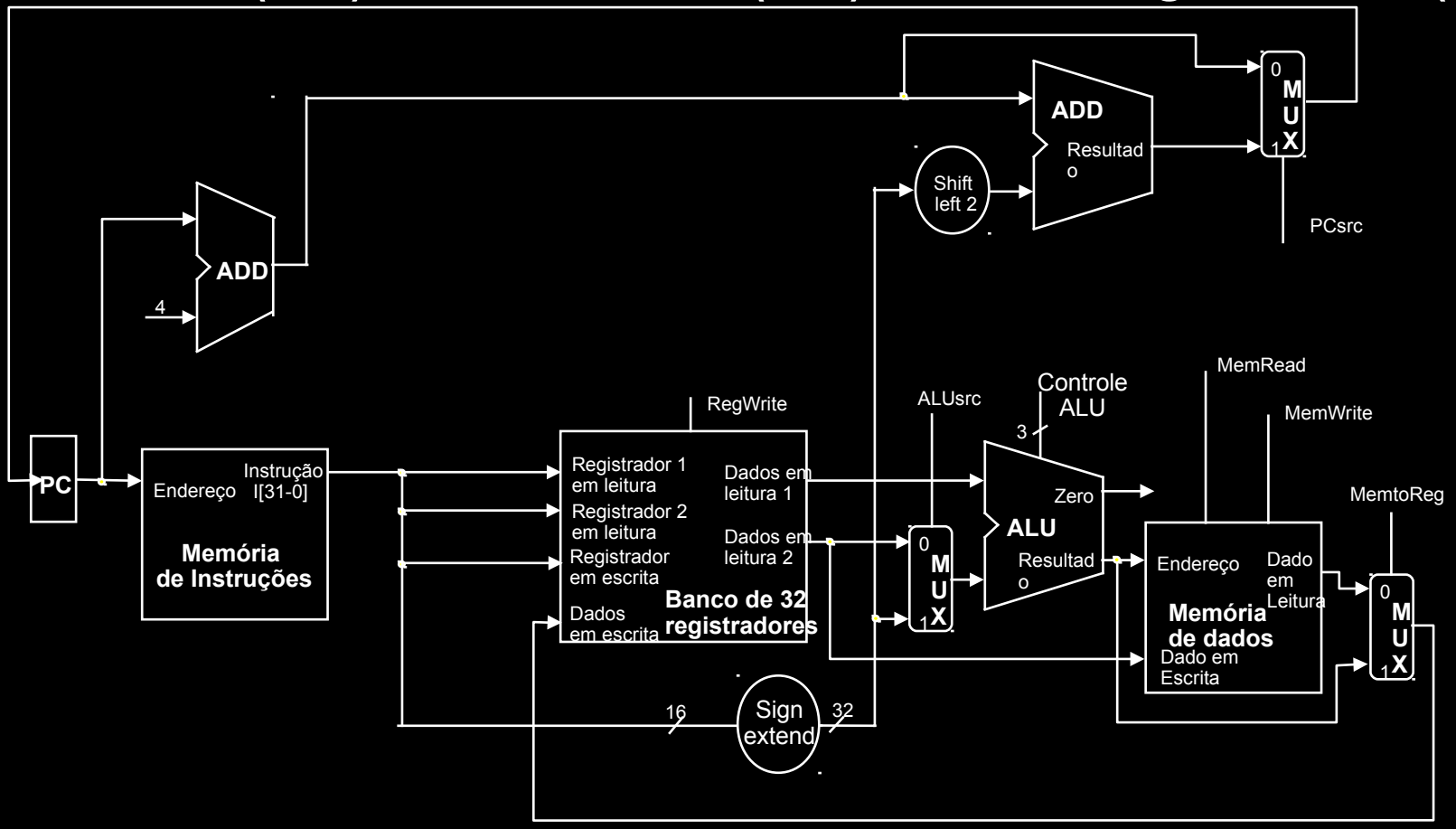
lcaimi@uffs.edu.br

Roteiro

1. Introdução
2. Ciclos das instruções
3. Bloco operacional completo
4. Execução das instruções
   - Busca de instrução
   - Decodificação de instrução
   - Instruções aritméticas
   - Instruções Load
   - Instruções Store
   - Instruções Branch
   - Instruções Jump
5. Cálculo de desempenho

Implementação único Ciclo

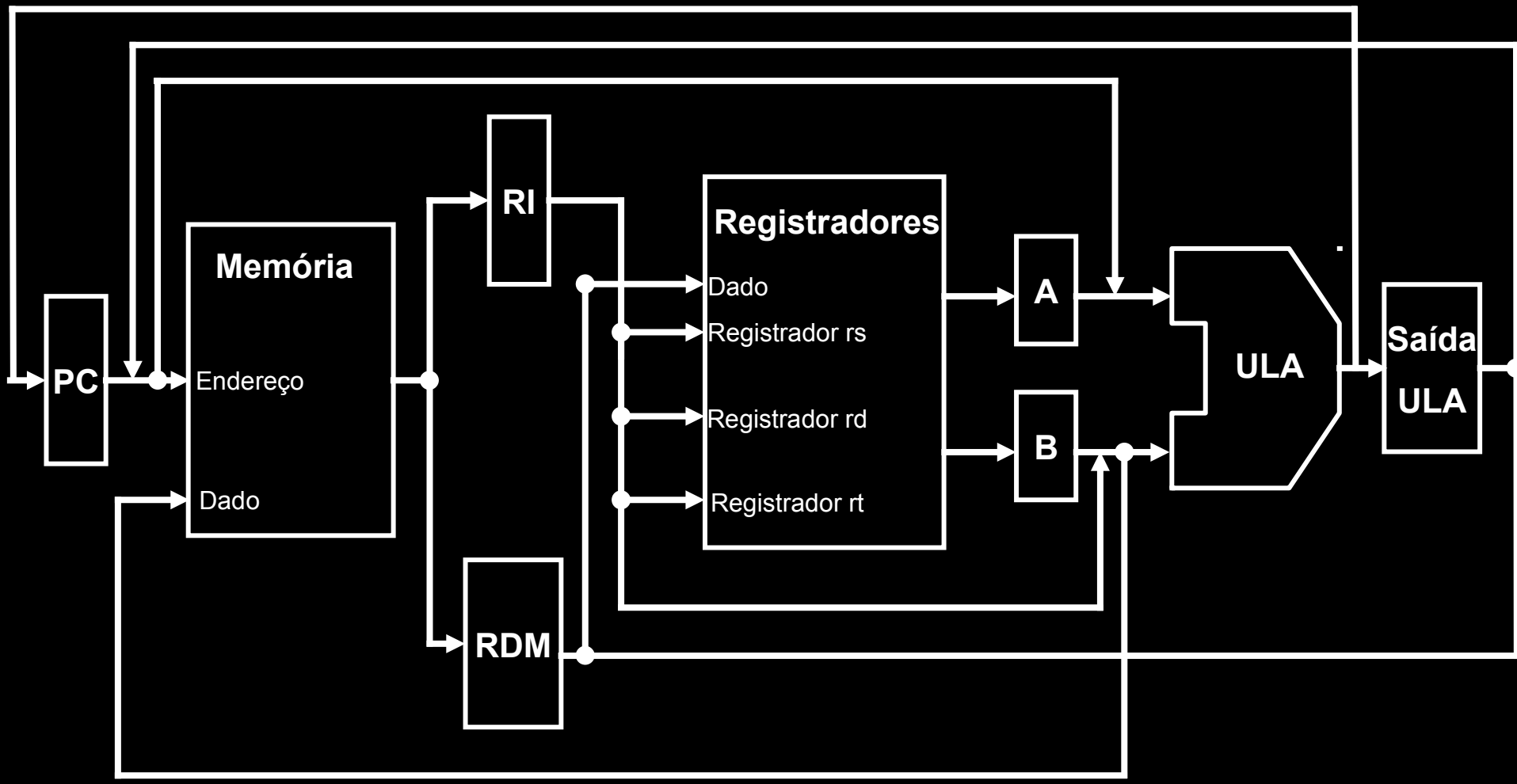
- Cálculo do *tempo de ciclo* com atrasos desprezíveis exceto
  - memória (2ns), ALU and ADDs (2ns), acesso a registradores (1ns)



Onde Nós Estamos

- **Problemas do ciclo único:**
  - **Instruções complicadas como ponto flutuante, como fazer?**
  - **Gasto de área**
- **Uma solução:**
  - **usar o menor *tempo de ciclo***
  - **Ter diferentes números de ciclos para instruções diferentes**
  - **datapath "multiciclo":**

Abstração do Multiciclo



1. Introdução

- máquina monociclo
  - todas as operações devem ser feitas em um só ciclo
  - duração do ciclo calculada pelo pior caso
  - leitura da instrução e acesso à memória no mesmo ciclo
  - cálculos de endereço e operações aritméticas no mesmo ciclo
  - Duas memórias e três ULAs
- máquina multiciclo
  - vários ciclos por instrução
  - instruções podem ser executadas com nros diferentes de ciclos
  - unidades funcionais podem ser reutilizadas em ciclos distintos
    - pequeno acréscimo de multiplexadores e registradores
- compromisso no desempenho:

  CPI aumenta => desempenho cai;

  período do relógio diminui => desempenho sobe

1. Introdução

- quando é necessário inserir registradores?
  - quando valor é computado num ciclo e utilizado em outro ciclo
  - quando entradas de unidade funcional podem mudar antes que a saída seja salva em outro registrador ou memória
  - exemplo: Instruction Register (RI)
    - memória vai mudar saída devido à atualização do PC e campos da instrução precisam se manter estáveis nas entradas do banco de registradores durante todos os ciclos
- instrução deve ser dividida em passos de duração similar

2. Ciclo das instruções

a. Busca da instrução

Calculo de PC + 4

b. Decodificação da instrução

Leitura dos registradores – mesmo que não sejam utilizados

Cálculo do endereço do branch – mesmo que instrução não seja branch

c. Execução da operação – instruções tipo R

Cálculo do endereço efetivo do operando – instruções load e store

Determinar se branch deve ser executado – instruções branch

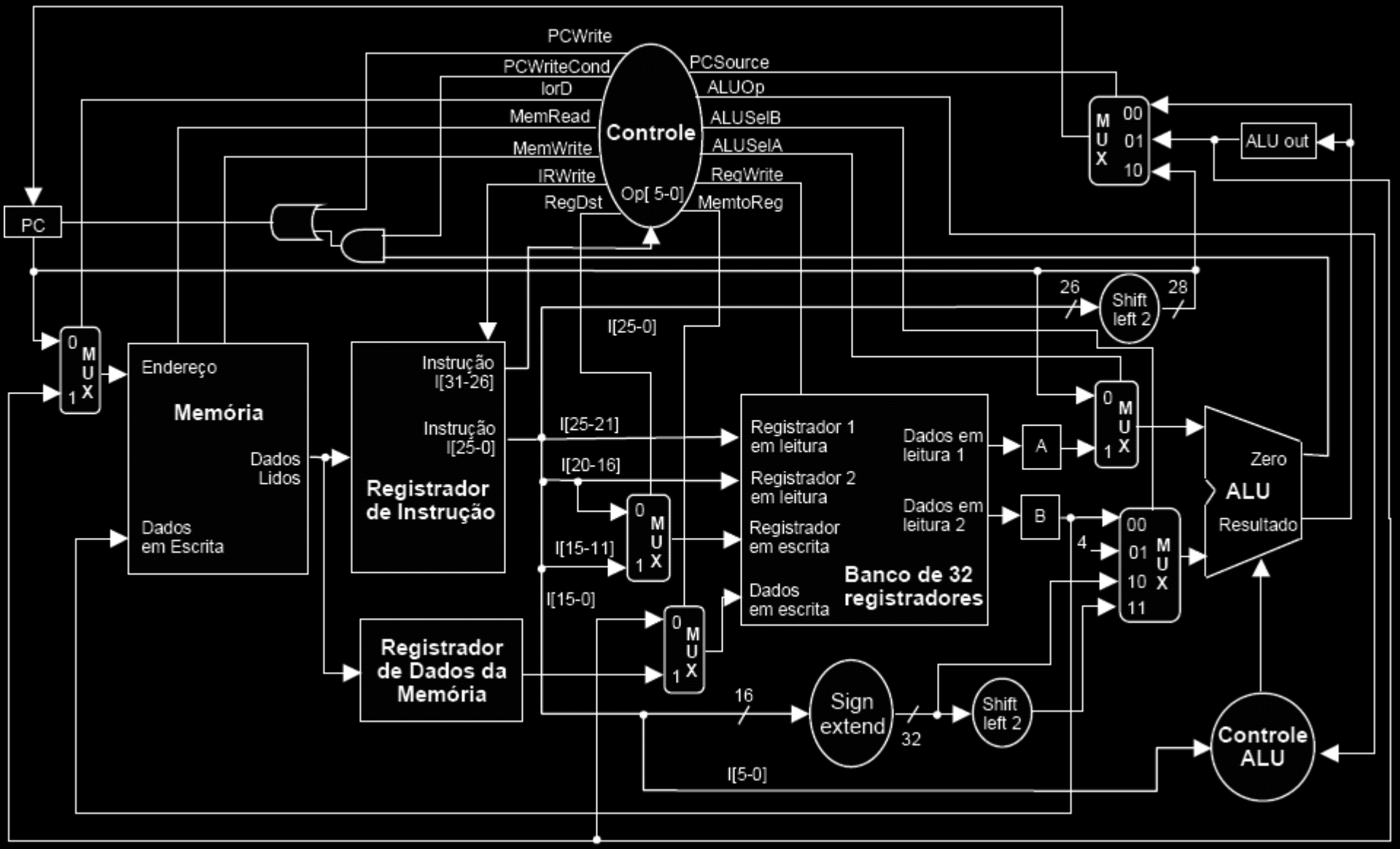
Deslocamento e concatenação no Jump

d. Acesso à memória – instruções load e store

Escrita de registrador – instruções tipo R

e. Escrita de registrador – instruções load

3. Bloco operacional completo



3. Bloco operacional completo

- uma única memória para dados e instruções
- uma única unidade lógica e aritmética para todas as operações
  - operações das instruções tipo R
  - cálculo de PC = PC + 4
  - cálculo de endereço efetivo de memória: base + deslocamento
  - cálculo de endereço de desvio: PC + deslocamento
- novos registradores
  - Registrador de Instruções
  - ALUop: para guardar endereço de desvio / resultado Aritméticas
- novos multiplexadores ou extensão dos já existentes

(a) Busca de instrução

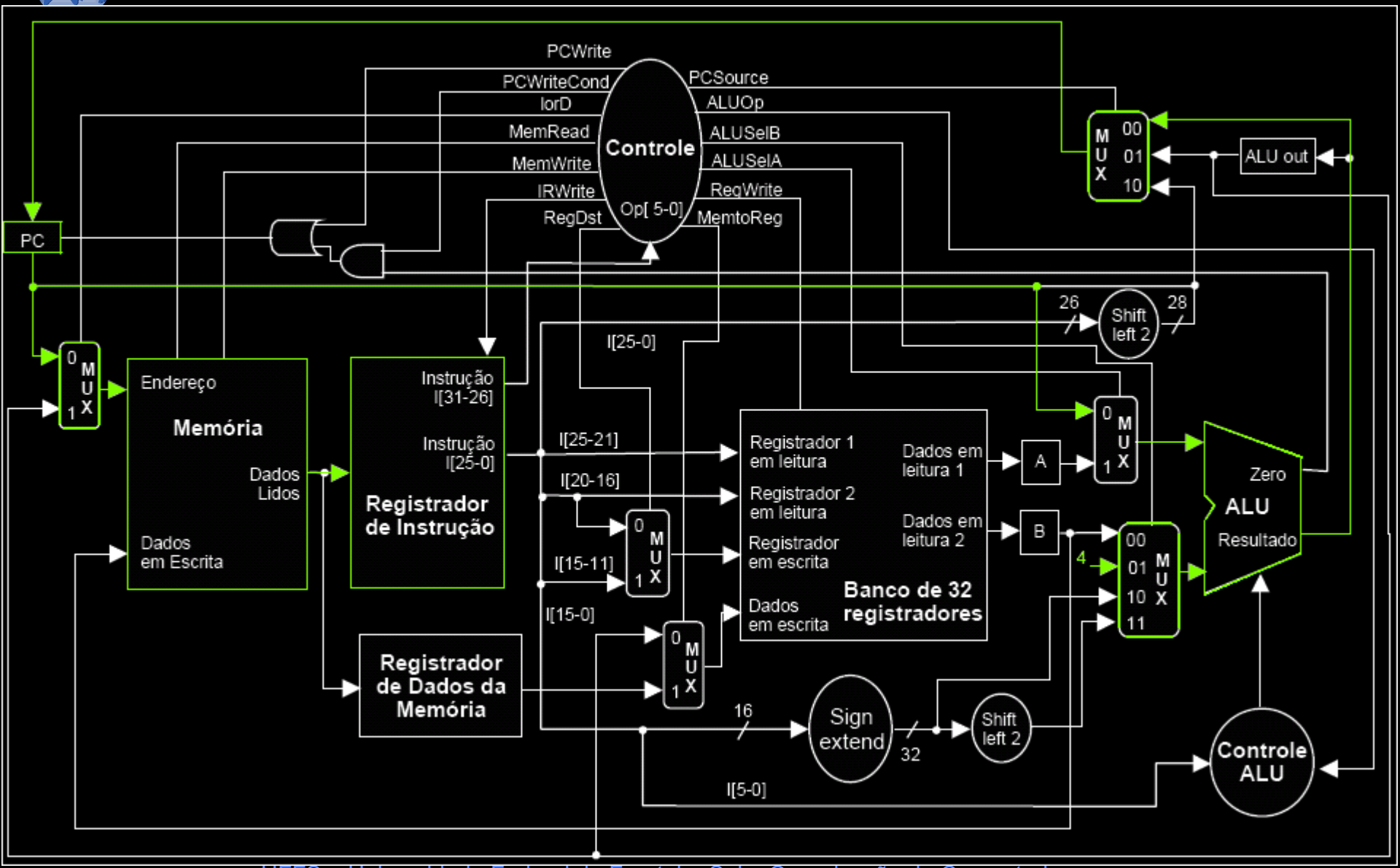
Operações:

IR = Memória[PC]
PC = PC + 4

Caminho de dados:

PC
Memória
IR
ULA

(a) Busca de instrução



(b) Decodificação de instrução (leitura de registradores, cálculo de end. branch)

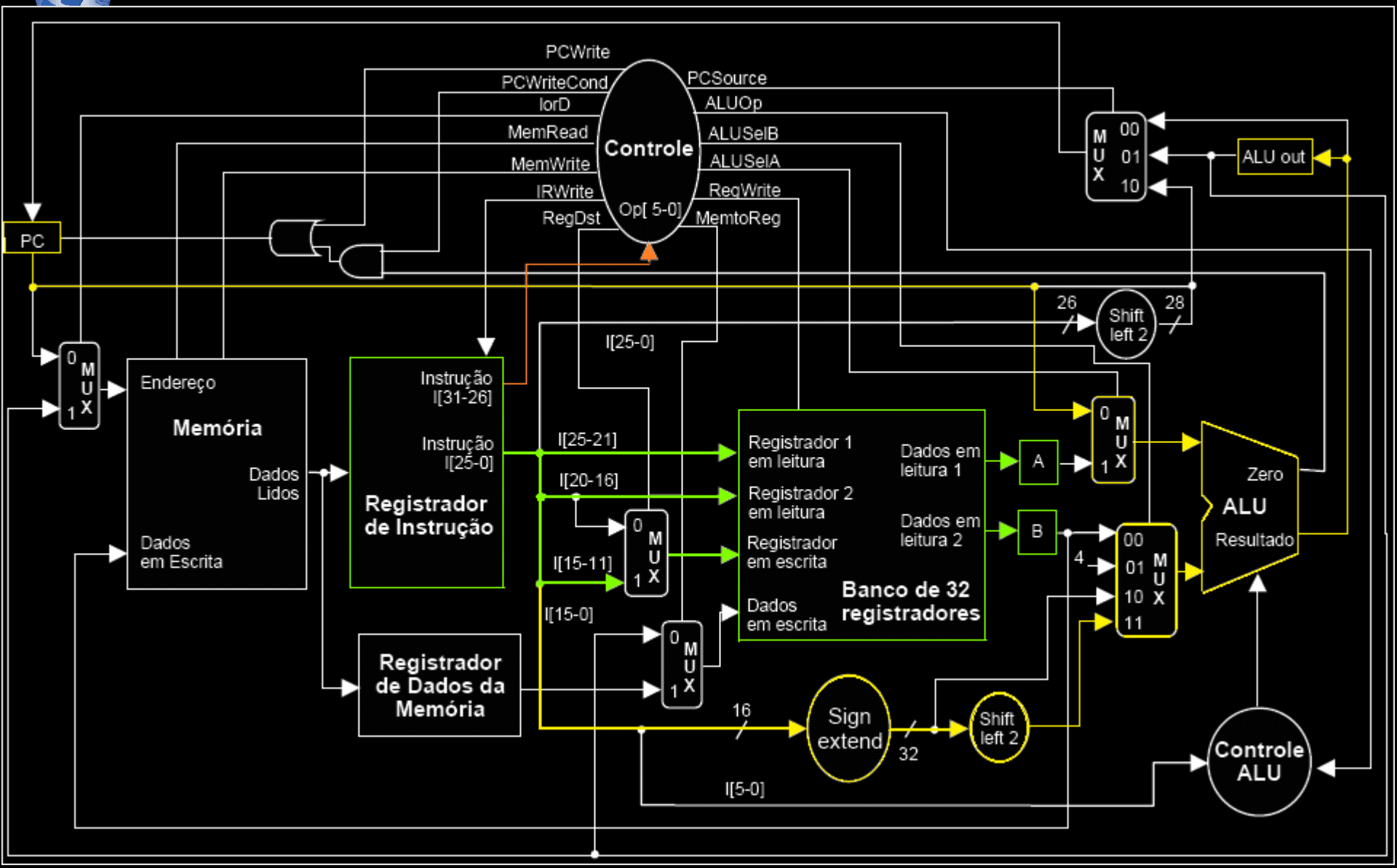
Operações:

A = Reg[25-21]
B = Reg[20-16]
UALSaida = PC + ext. de sinal(IR[15-0] << 2)

Caminho de dados:

PC
Banco de Registradores
IR
A
B
UAL
UALSaida

(b) Decodificação de instrução (leitura de registradores, cálculo de end. branch)



(c) Execução

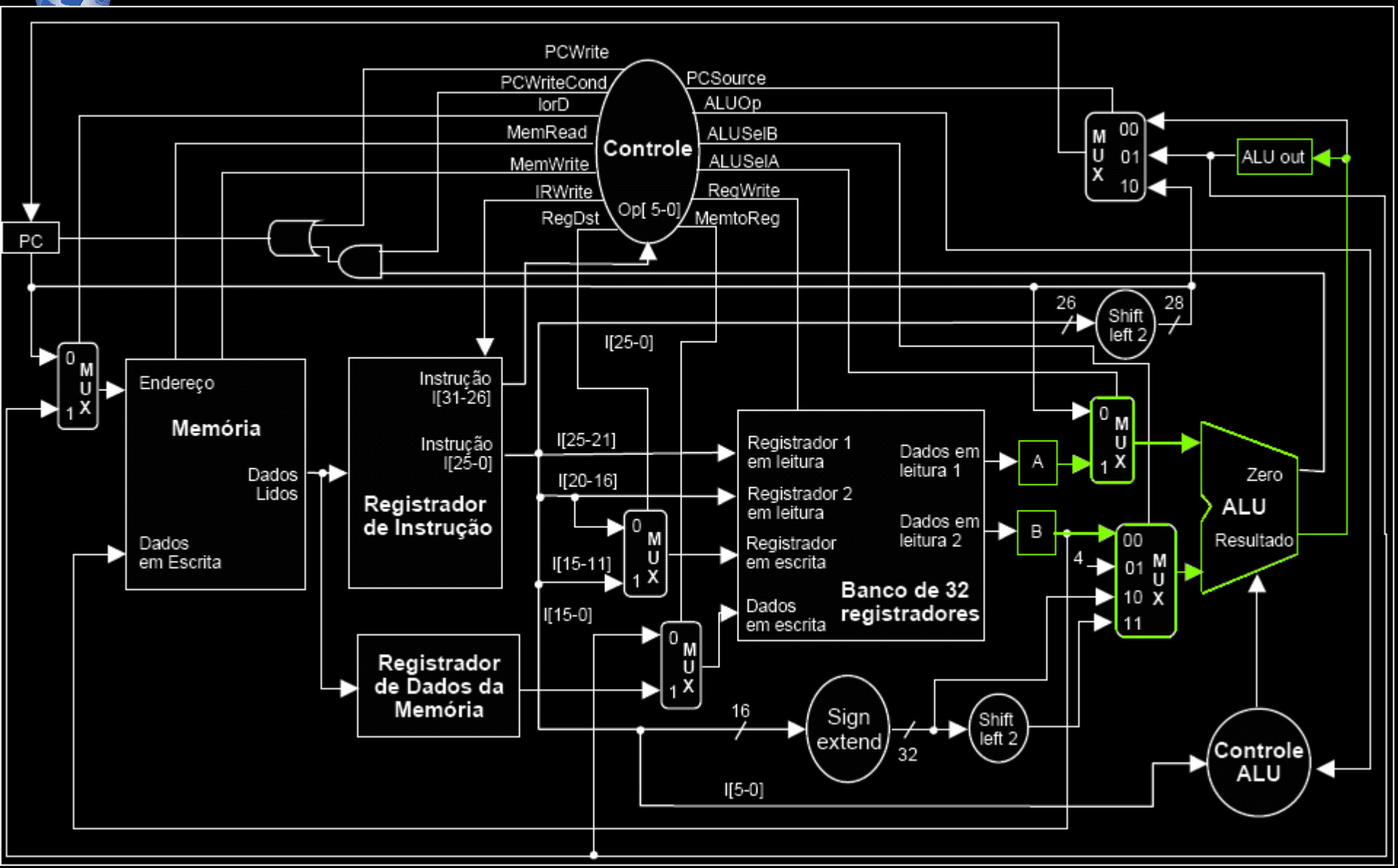
Operações:

- **Ref. Memória:** UALSaida = A + ext. de sinal(IR[15-0])
- **Aritmética:** UALSaida = A + B
- **Desvio Cond:** se (A==B) PC = UALSaida
- **Desvio Incond:** PC = PC[31-28] : (IR[25-0] << 2)

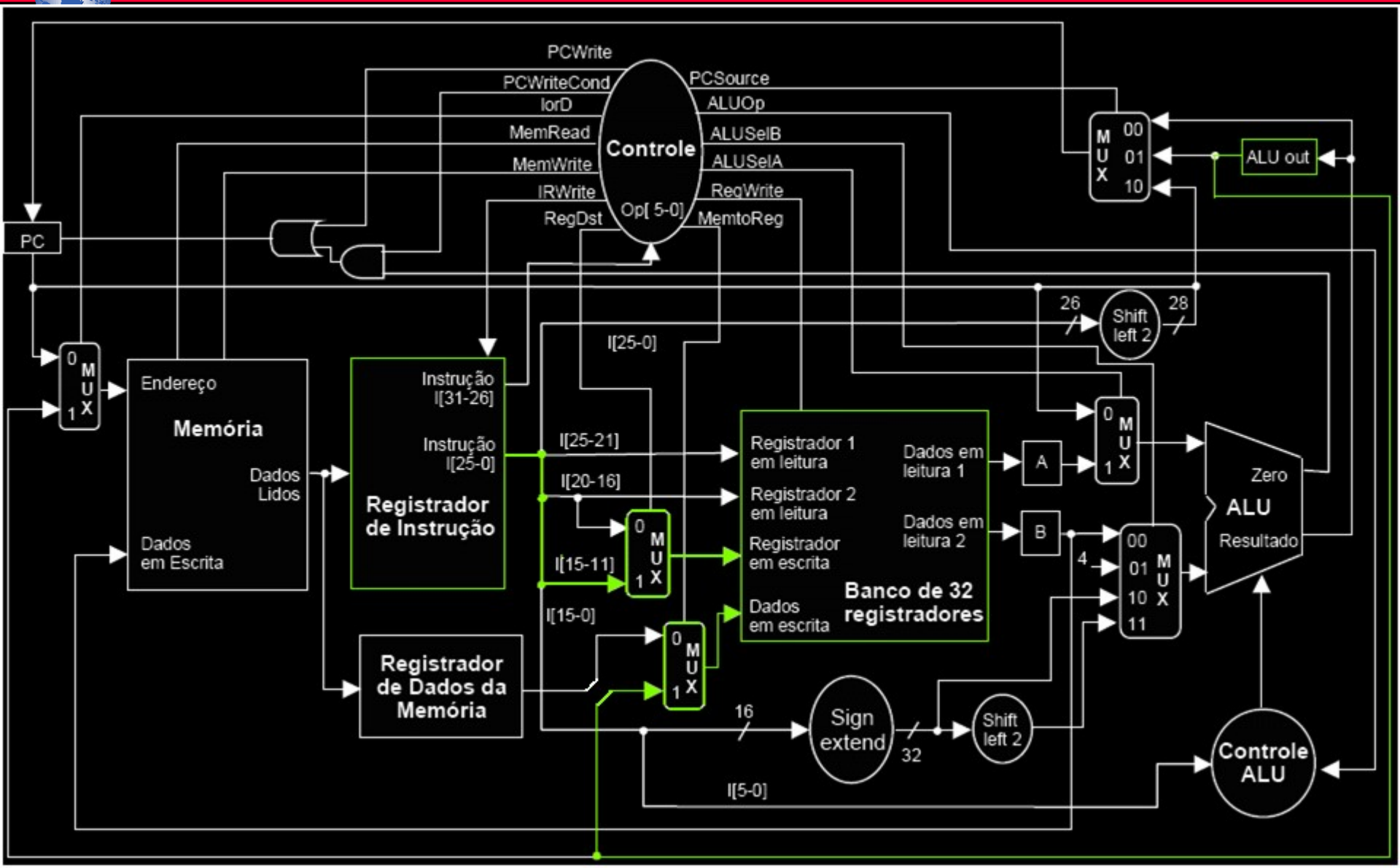
Caminho de dados:

PC
IR
Banco de Registradores
A
B
UAL
UALSaida

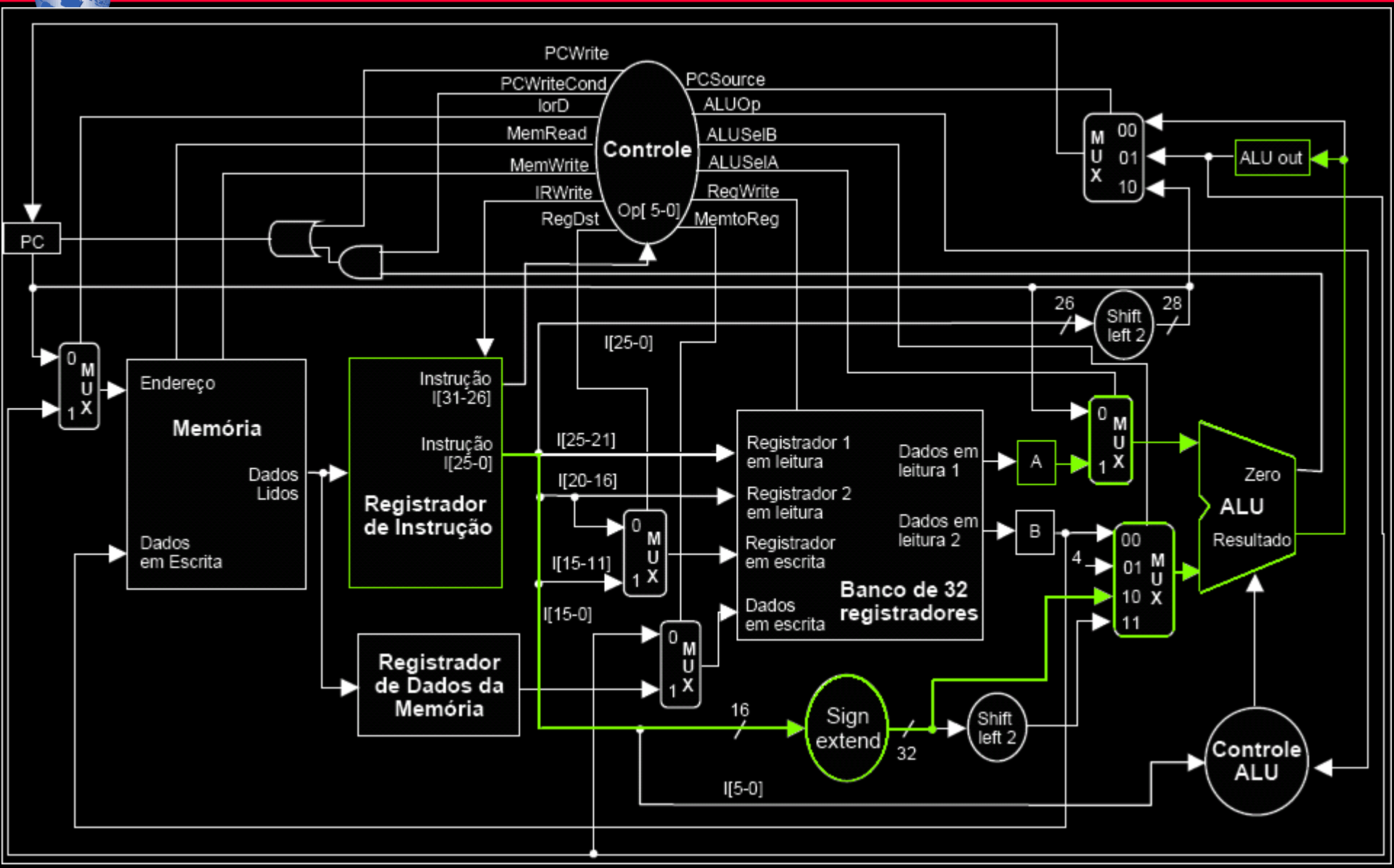
(c) Instruções aritméticas



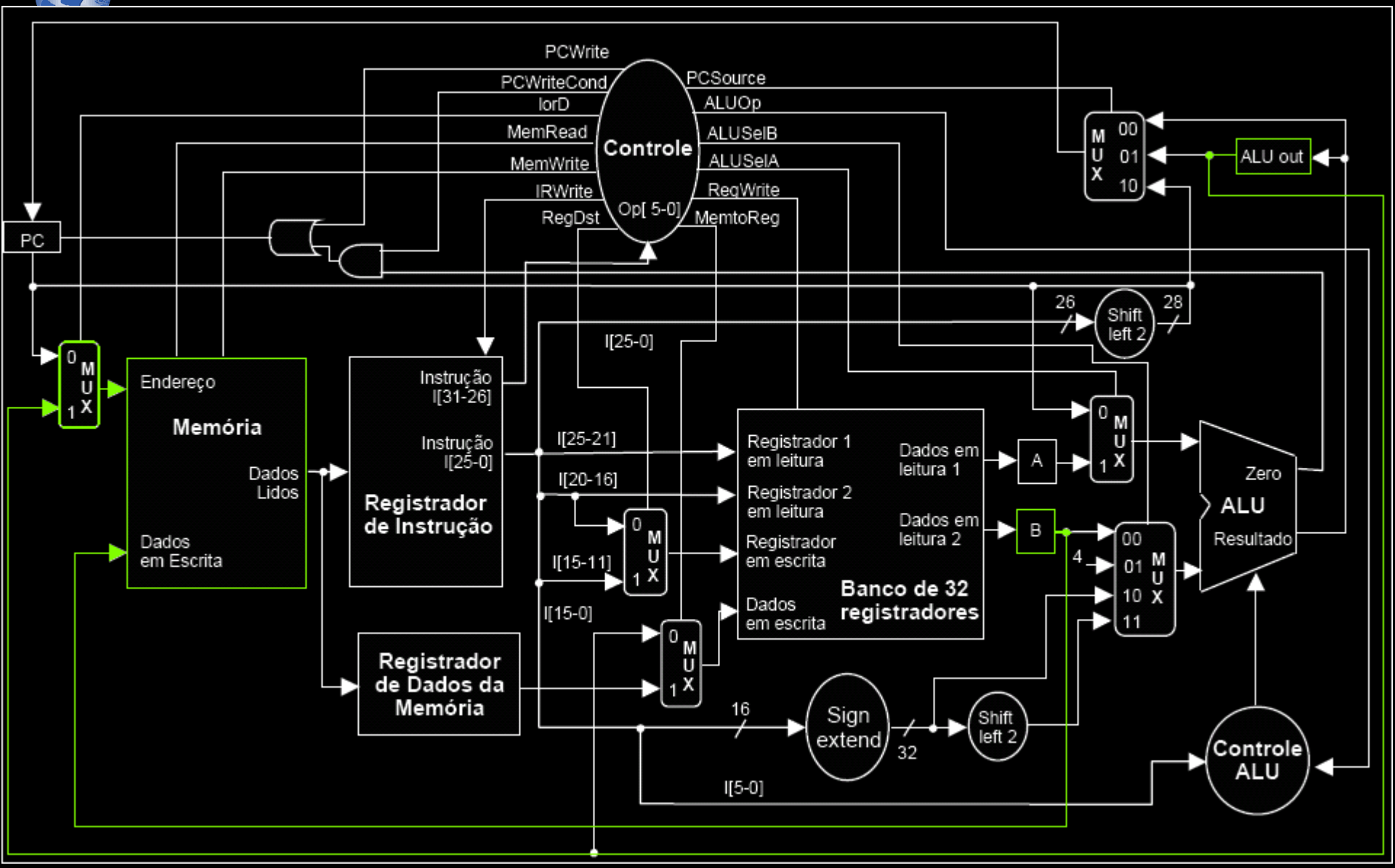
(d) Instruções aritméticas



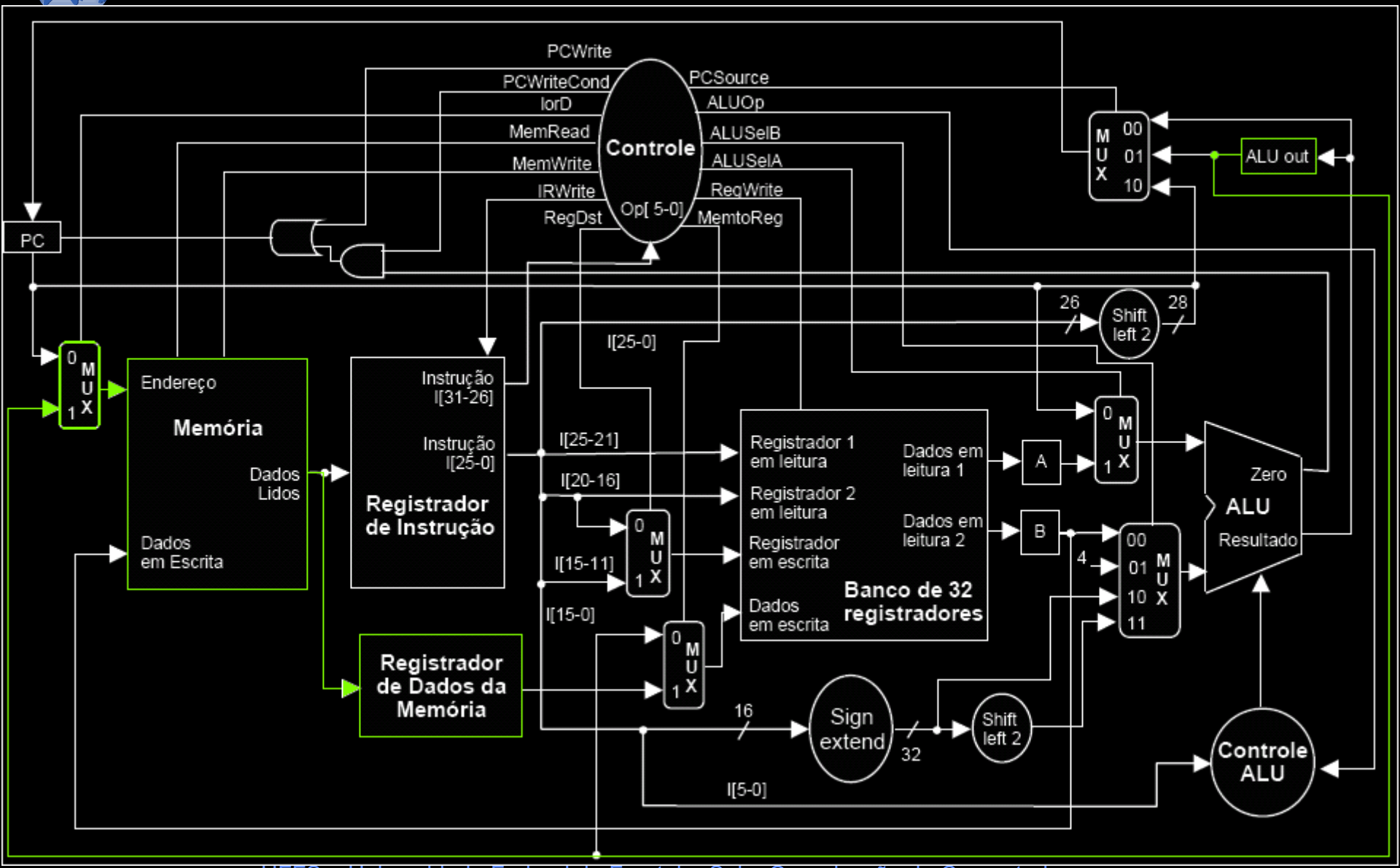
(c) Instruções Load/Store calcula endereço



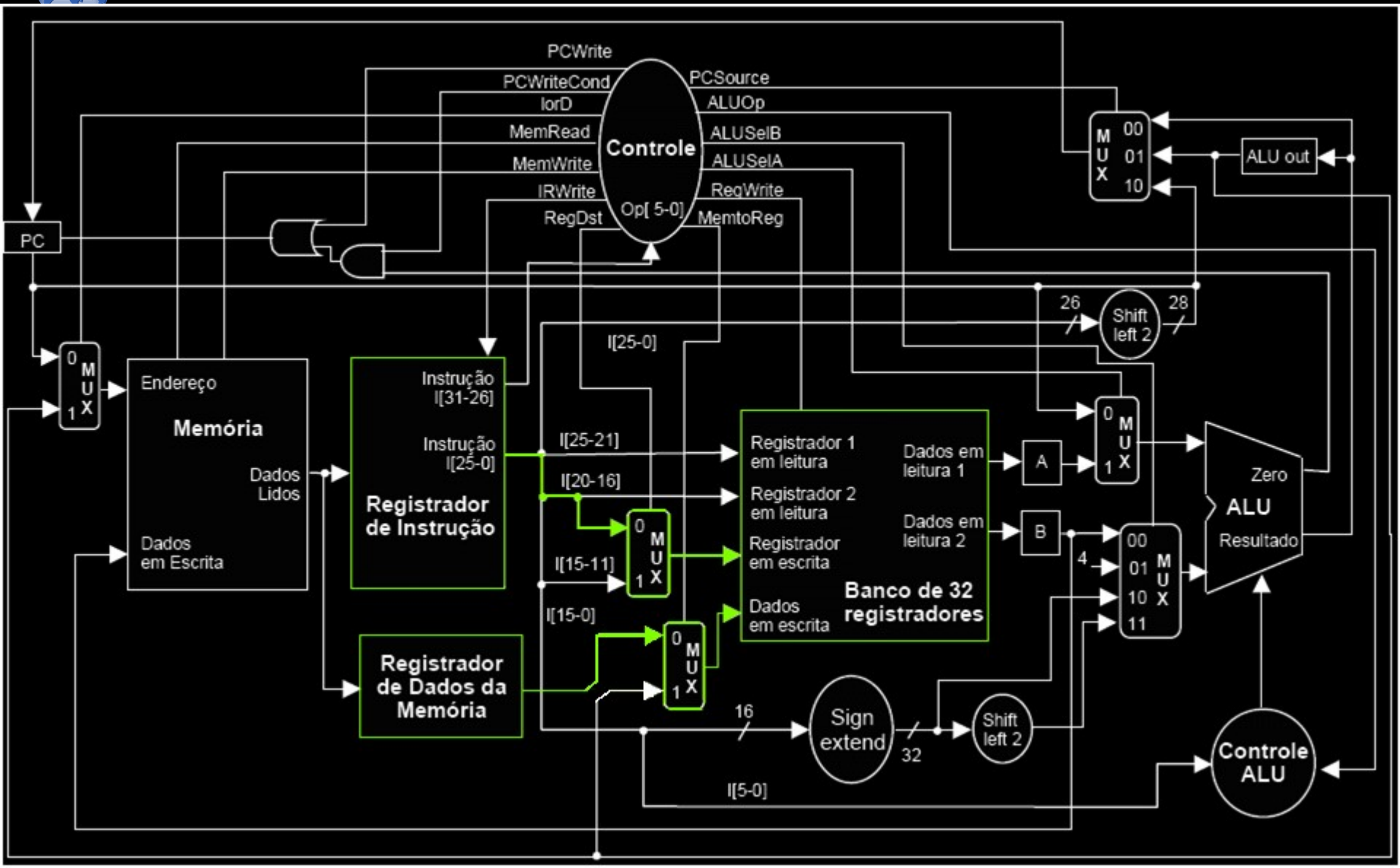
(d) Store – escreve na memória



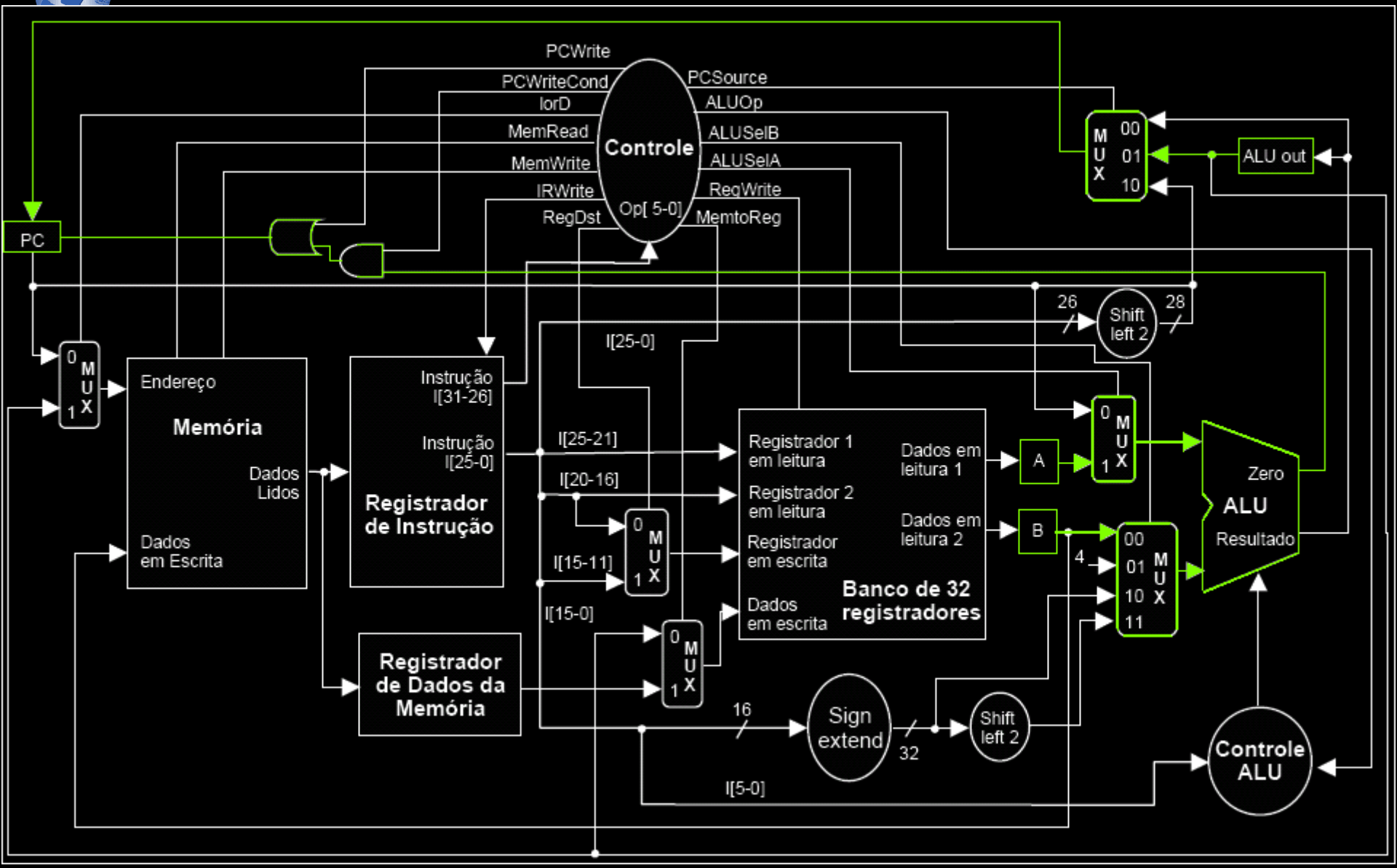
(d) Load – lê memória



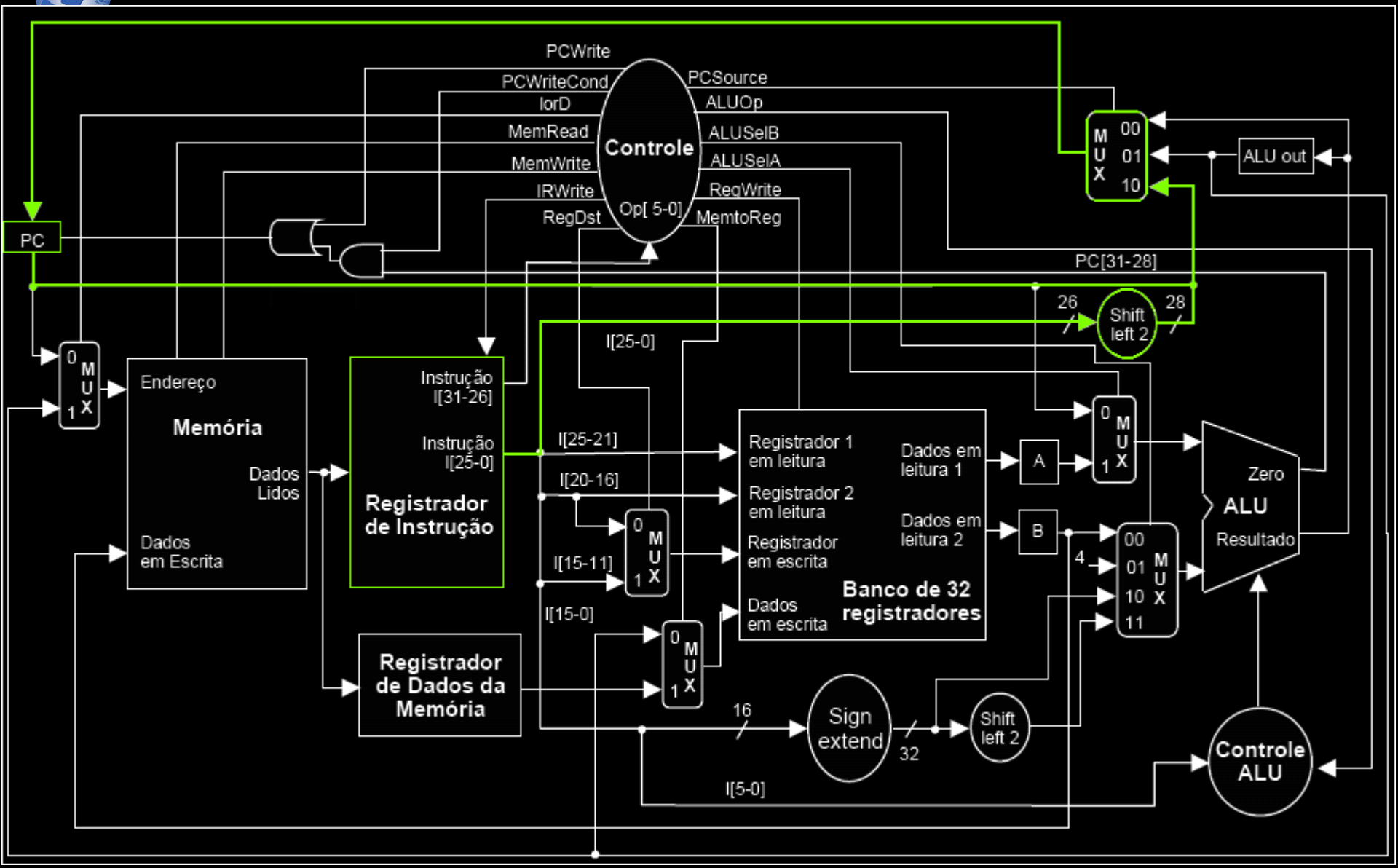
(e) Load – escreve registrador destino



(c) Instruções Branch



(c) Instruções Jump

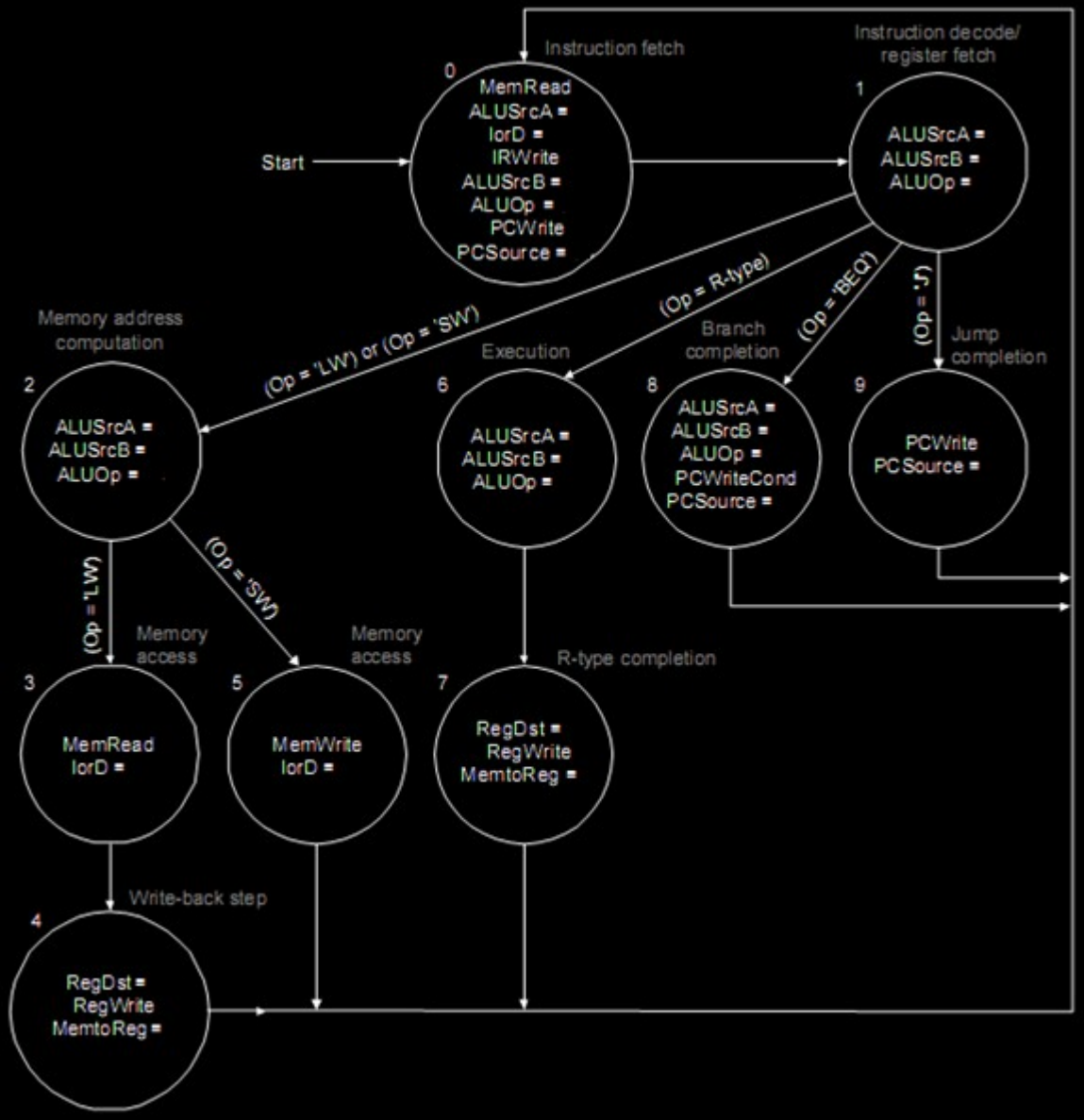


Unidade de Controle: sinais

- Sinais de controle do processador

PcEscCond	FontePC
PcEsc	UALOp
IouD	UALFonteB
LerMem	UALFonteA
EscMem	EscReg
MemParaReg	RegDst
IREsc	

Unidade de Controle: estados



5. Cálculo de desempenho

- clock pode ter período de 5 ns, considerando ...
  - 5 ns para acessos à memória
  - 3 ns para acessos ao banco de registradores
  - 4 ns para operações na ALU
  - 0 ns para demais blocos (muxs, portas, …)
- com estes valores, período da versão monociclo seria de 20 ns
- CPI para cada tipo de instrução
  - load: 5 - store: 4
  - tipo R: 4 - branch: 3
  - jump: 3

Cálculo de desempenho

- mix de instruções do compilador gcc
  - 22% loads, 11% stores, 49% tipo R, 16% branches, 2% jumps

- CPI médio = .22 x 5 + .11 x 4 + .49 x 3 + .16 x 3 + .02 x 3
  = 3.55

- tempo total de execução do programa gcc no DLX multiciclo
  = N x CPI médio x período do clock
  = N x 3.55 x 5 ns = 17.75 N

- tempo total de execução do programa gcc no DLX monociclo
  = N x CPI x período do clock
  = N x 1 x 20 ns = 20 N