Introdução à Arquitetura de Computadores

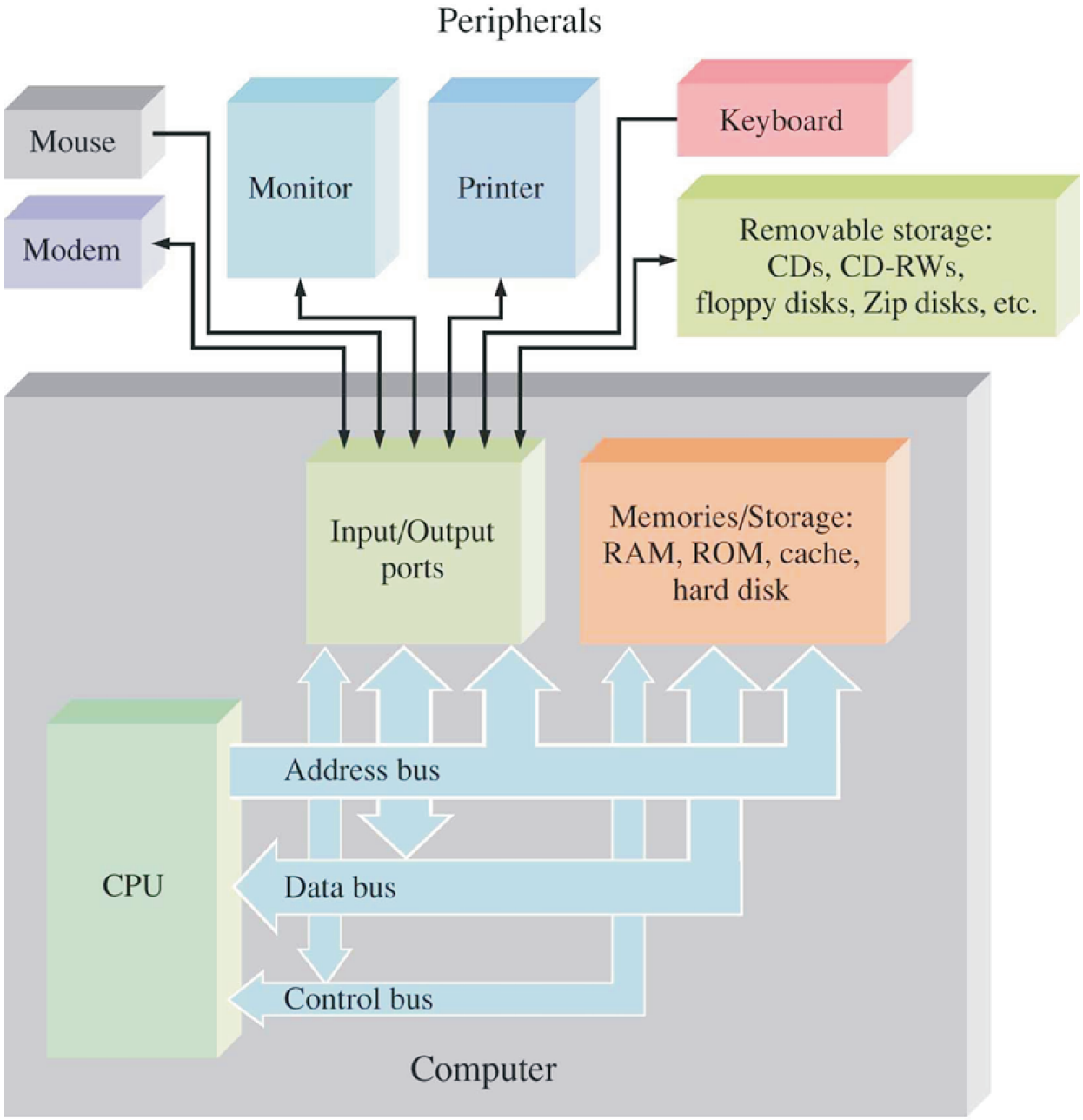
- Hardware e software
- Organização de um computador:
  - Processador: registradores, ALU, unidade de controle
  - Memórias
  - Dispositivos de E/S
  - Barramentos
- Linguagens de programação
- Geração de um programa executável
- Carga e execução de programa
- Instruções de máquina
- Ciclo de execução de instrução

Sistema Computacional

- Principais componentes de um sistema computacional:
  - Computador
  - Periféricos:
    - Dispositivos de entrada e saída (E/S)

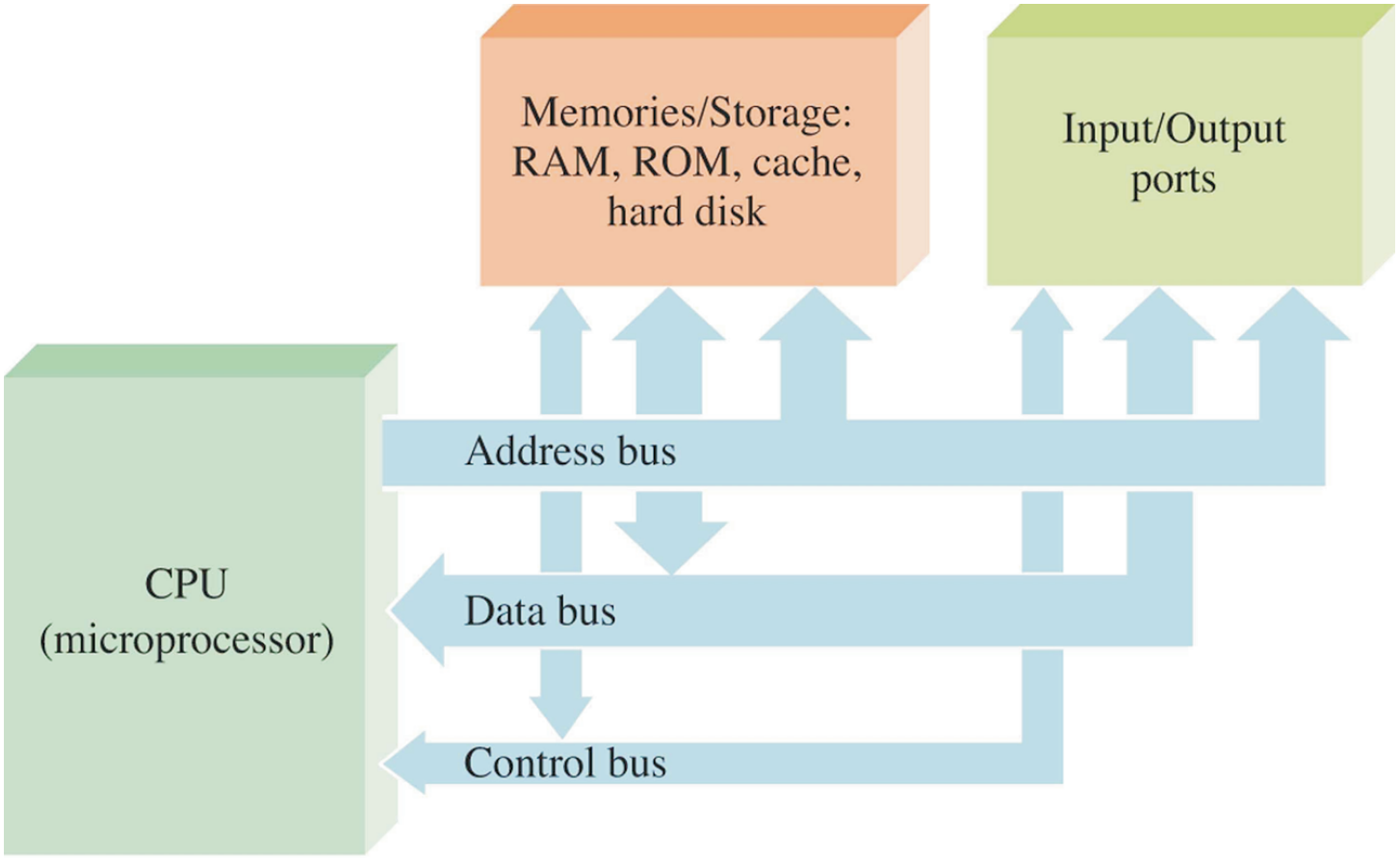
Sistema Computacional

- Principais componentes de um sistema computacional:
  - Computador
    - Processador
    - Memórias
    - Portas de E/S
    - Barramentos
  - Periféricos:
    - Dispositivos de E/S

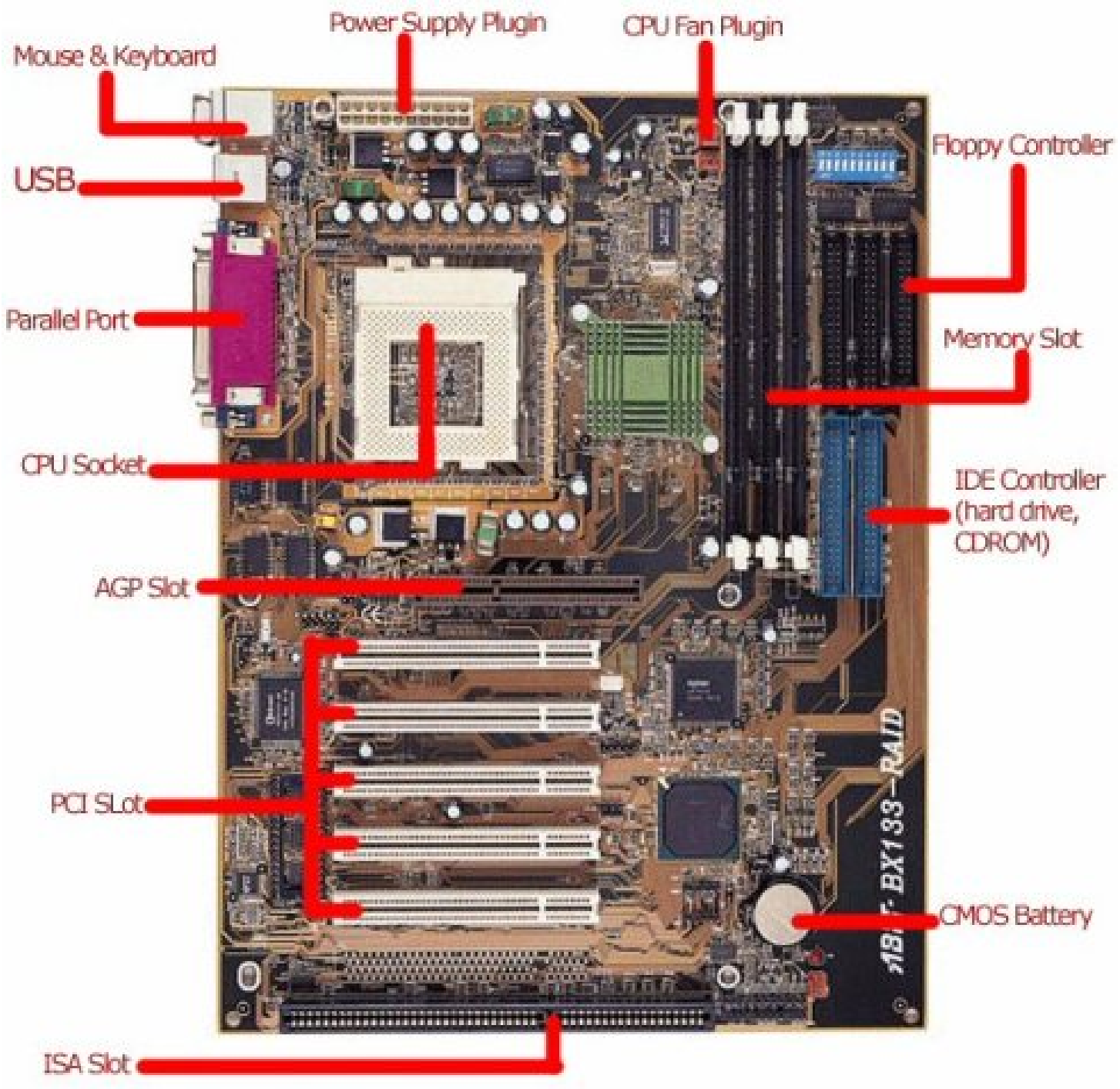


Organização de um Computador

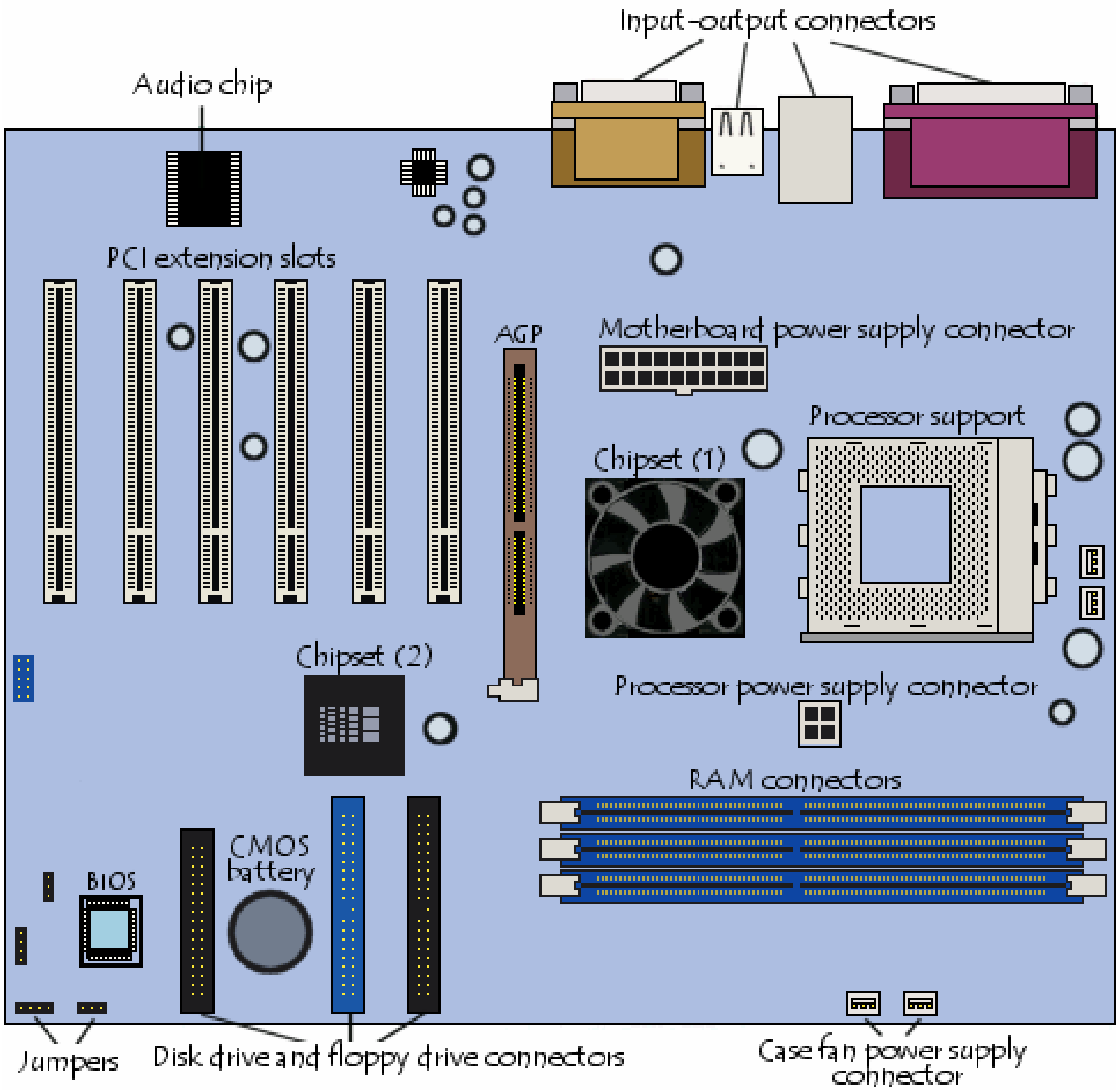
- Principais componentes de um computador:
  - Processador
  - Memórias
  - Portas de E/S: conexões de E/S para periféricos
  - Barramentos



Placa Mãe (Motherboard)

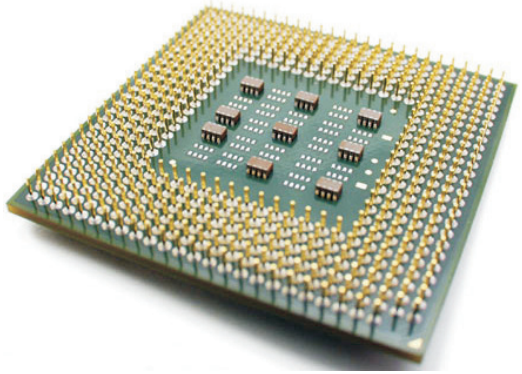


Placa Mãe (Motherboard)

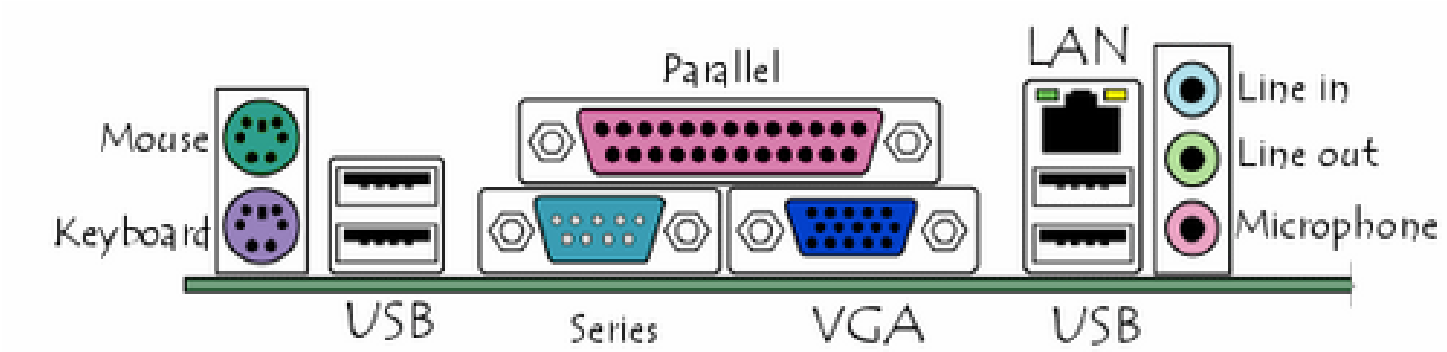


Componentes

- **Chip do processador**:



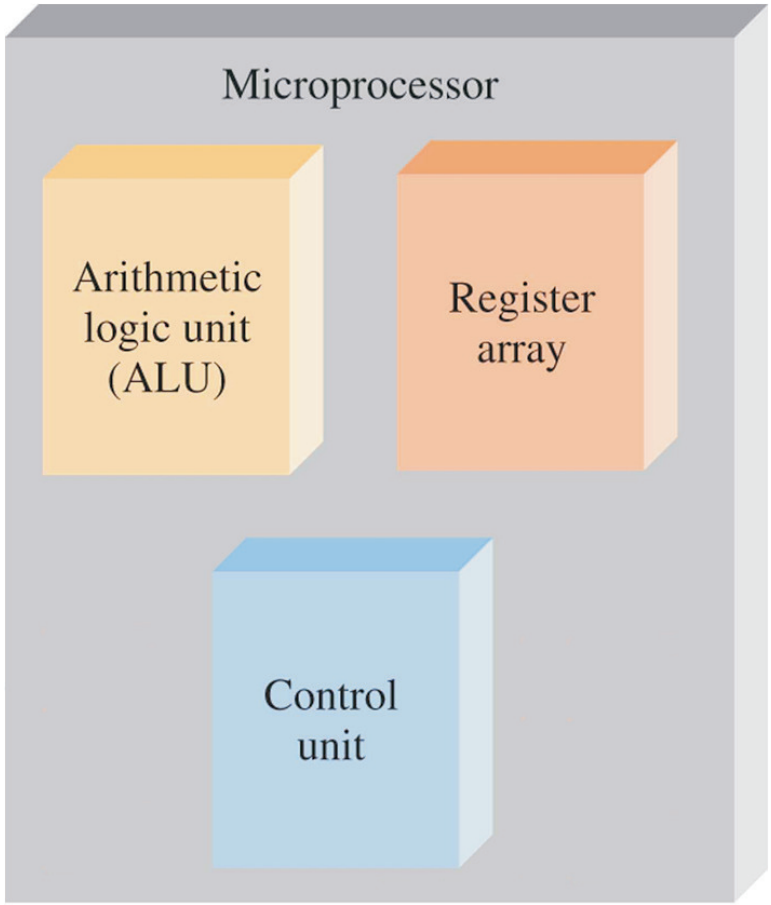
- **Interfaces**:



Processador

- **CPU**: Central Processing Unit

- Executa programas armazenados na memória:
  - Busca instruções na memória
  - Decodifica instruções
  - Executa instruções



- Composto por:
  - **Unidade de controle** (UC) (Control Unit)
  - **Datapath** (via de dados, caminho de dados):
    - **Unidade lógico-aritmética** (ULA)
    - **Conjunto de registradores**

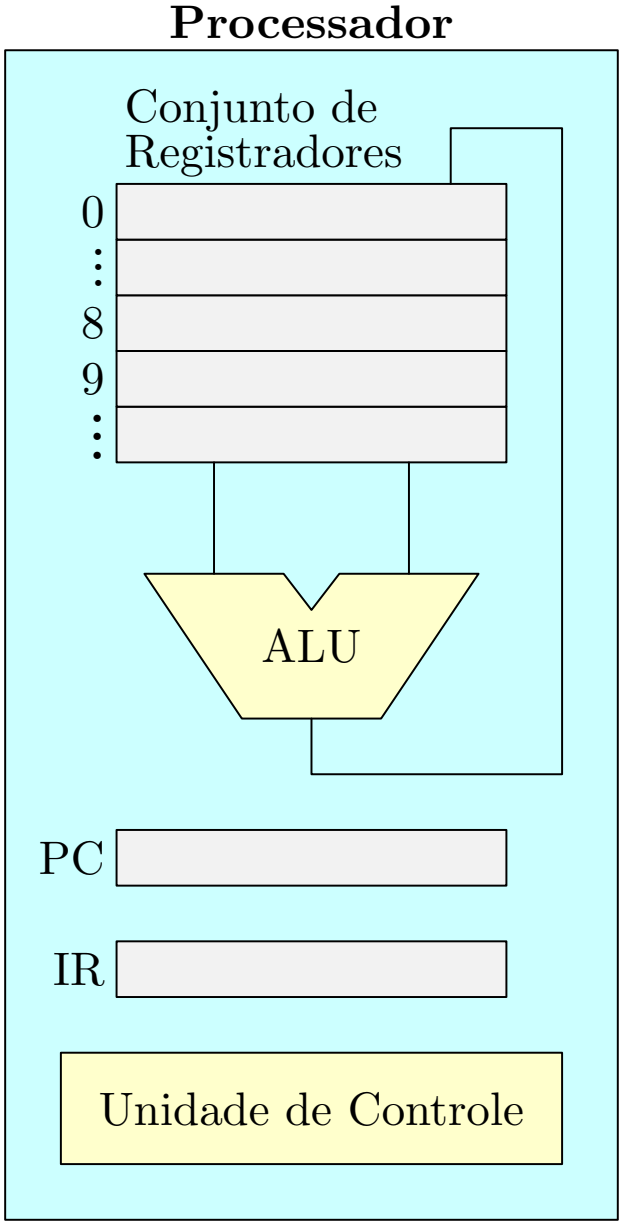
Unidade de Controle

- Comanda datapath, memória e dispositivos de E/S sobre o que fazer, de acordo com instruções do programa

- Controla execução das instruções no processador:
  - Comanda busca na memória da próxima instrução a ser executada, trazendo-a para o processador
  - Decodifica instrução (determina que operação ela representa)
  - Comanda operação a ser realizada pelo datapath

Datapath

- Realiza a execução das instruções
- Executa operações comandadas pela UC
- Composto por:
  - **Unidade lógico-aritmética**
  - **Conjunto de registradores**
- **Unidade lógico-aritmética** (ULA):
  - Realiza operações lógicas e aritméticas (comparação, soma, subtração, and, ...)



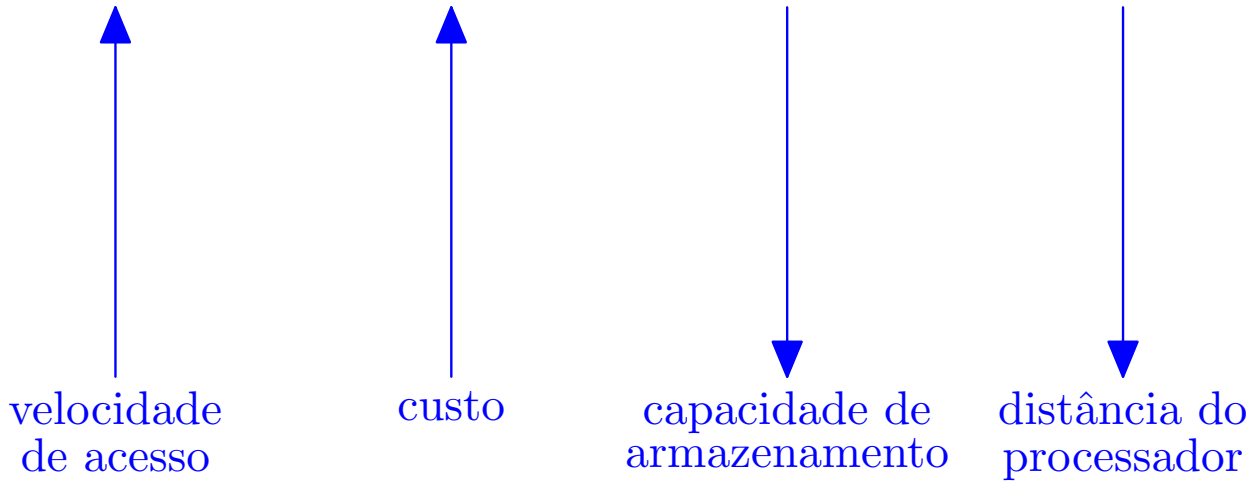
- **Exemplo**:
  - UC busca na memória instrução a executar (por exemplo: `add A, B`)
  - ALU realiza soma dos 2 operandos, produzindo um resultado

Datapath

- **Registradores**:
  - Pequenas unidades capazes de guardar informações (dados, endereços, instruções)
  - Dispositivos de armazenamento de rápido acesso $\Rightarrow$ Custo elevado $\Rightarrow$ Processador tem poucos registradores
  - Meio de armazenamento volátil
  - Registradores de propósito específico e de uso geral
  - Variam de um tipo de processador para outro
- Quase todos os processadores têm registradores para:
  - Endereço da próxima instrução a executar (PC – **Program Counter**)
  - Próxima instrução a executar (IR – **Instruction Register**)
  - Endereço do dado que está no topo da pilha de execução (SP – **Stack Pointer**)
  - Uso geral: realização de operações lógico-aritméticas

Sistema de Memórias

- Sistema de memória organizado em níveis $\Rightarrow$ **Hierarquia de Memórias**
  - Registradores
  - Memória cache
  - Memória principal (MP)
  - Disco (memória virtual)



Memória Principal

- Memória Principal ou memória primária:
  - **Armazena instruções e dados dos programas sendo executados**

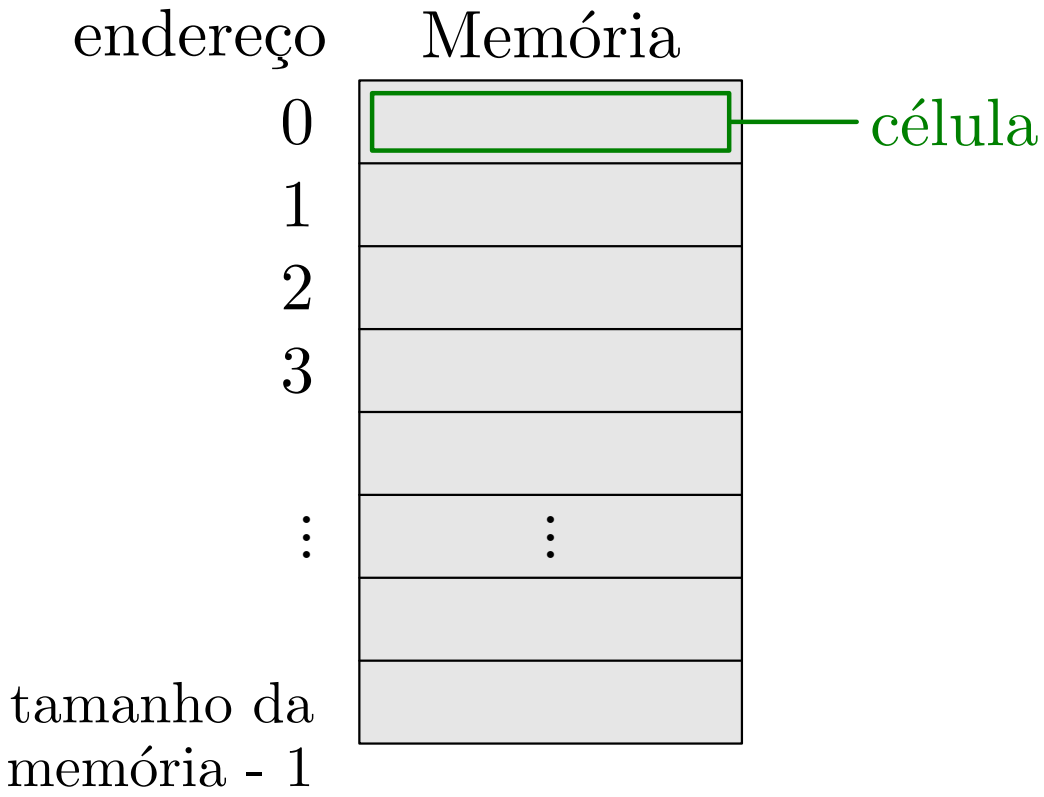
- Memória **RAM** (Random-Access Memory):
  - Qualquer posição pode ser acessada com mesmo tempo de acesso

- Acesso mais lento que registradores $\Rightarrow$
  Menor custo $\Rightarrow$
  Maior capacidade de armazenamento

- Meio de armazenamento volátil

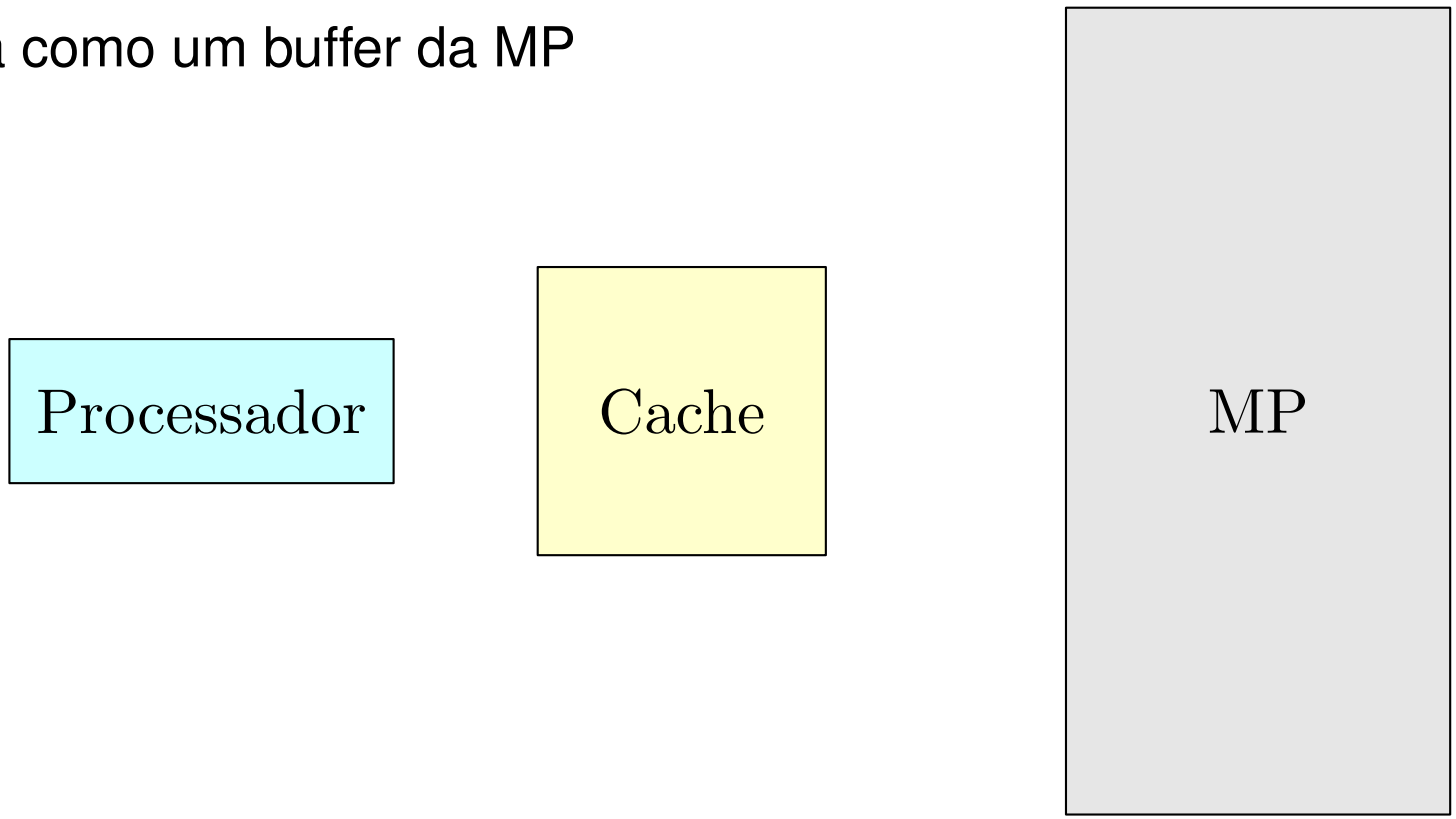
Memória Principal

- Pode ser vista como estrutura unidimensional:
  - Cada posição é uma **célula** onde pode-se armazenar uma informação
  - Todas as células têm mesmo nº de bits
  - Cada célula possui um endereço pelo qual é acessada



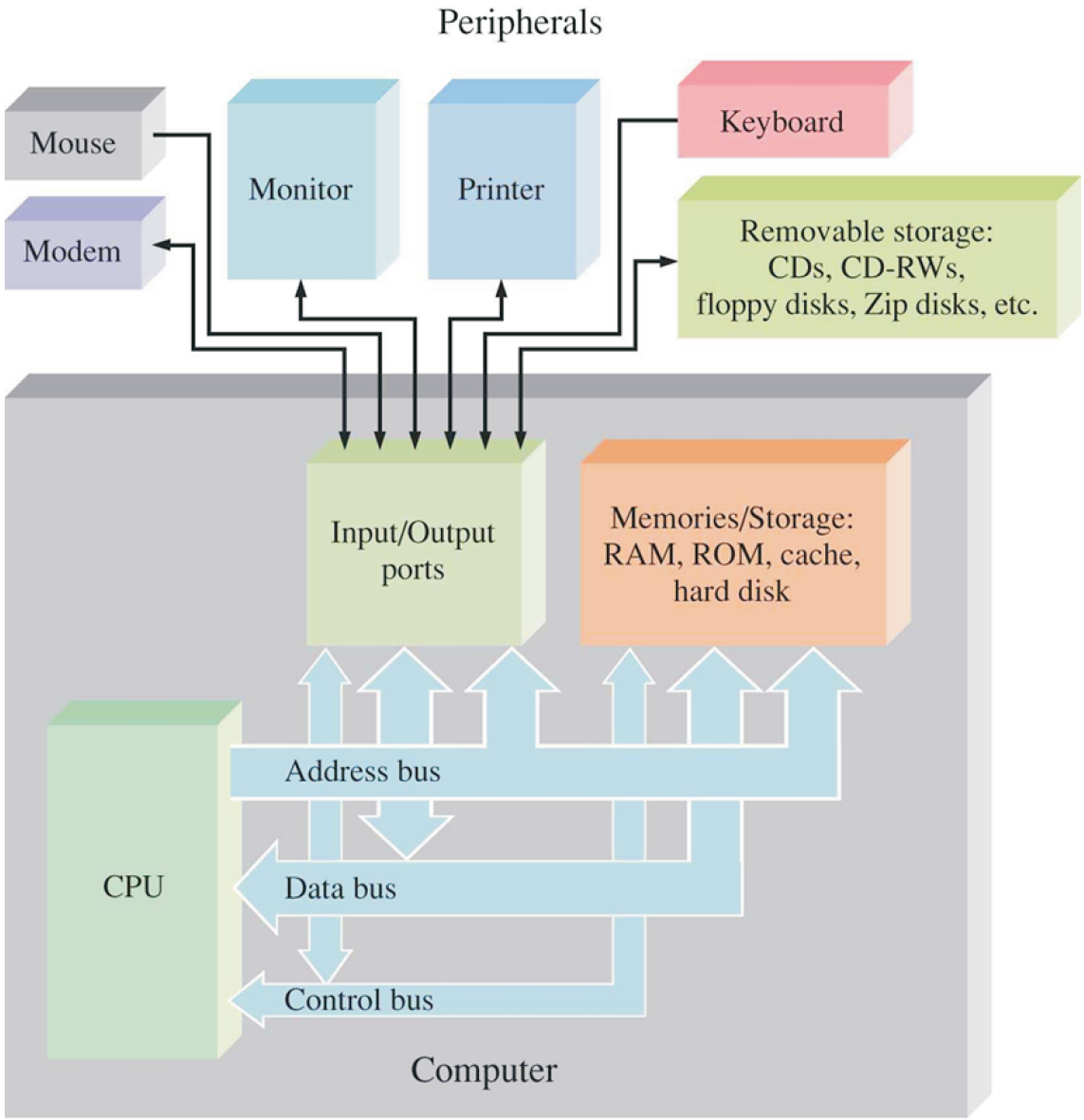
Memória Cache

- Nível da hierarquia de memórias entre memória principal (MP) e processador
- Mais rápida que MP $\Rightarrow$
  Maior custo $\Rightarrow$
  Menor capacidade de armazenamento
- Meio de armazenamento volátil
- Situada mais próxima ao processador que MP
- Funciona como um buffer da MP



Sistema Computacional

- Principais componentes:
  - Computador
  - Dispositivos de E/S



Dispositivos de Entrada e Saída

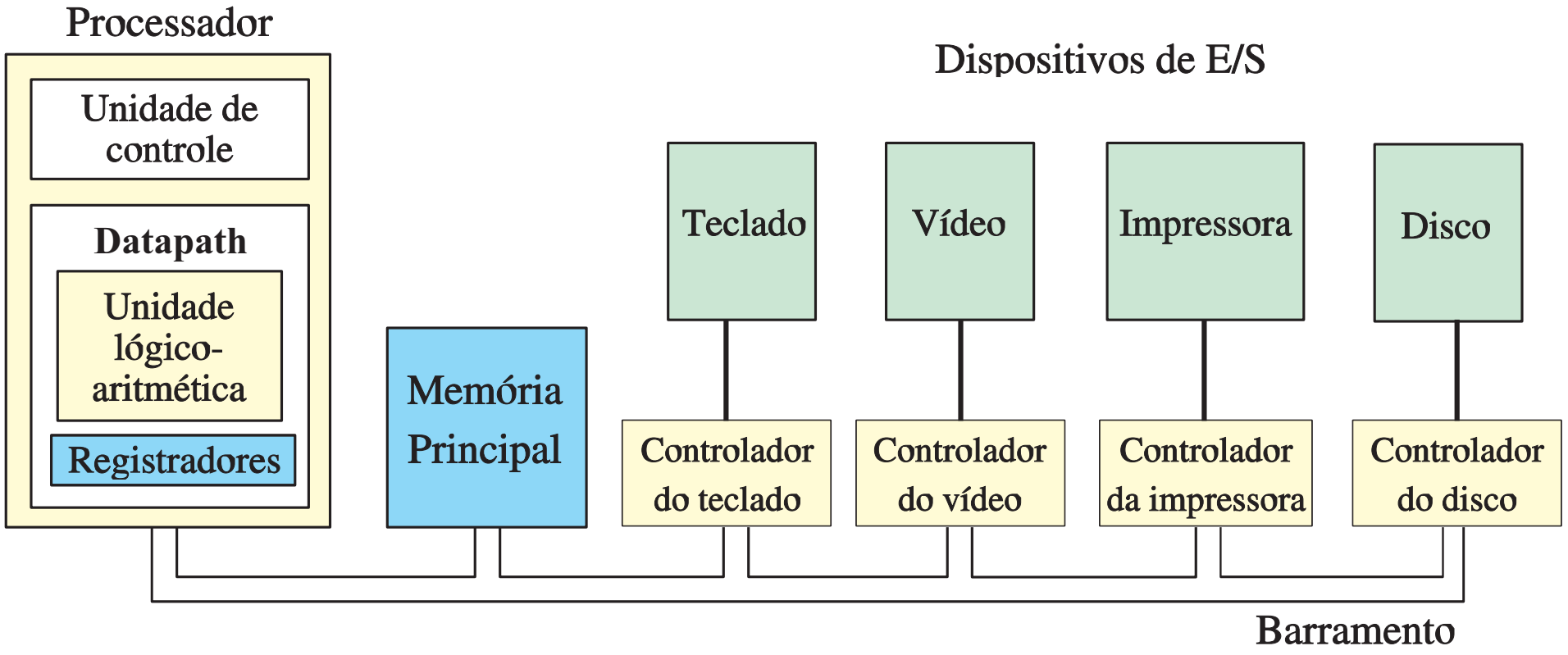
- **Dispositivos para comunicação entre usuário e computador**:
  - Permitem que usuário forneça e receba dados ao/do computador
  - **Exemplos**:
    - Teclado, vídeo, mouse, impressora
- **Dispositivos de armazenamento secundário**:
  - Permitem que informações sejam armazenadas em um meio **não** volátil
  - **Exemplos**:
    - Disco magnético, fita magnética, disco ótico
  - Denominados **memória secundária** em contraposição à MP
  - Mais lentos que MP $\Rightarrow$

    Menor custo $\Rightarrow$

    Maior capacidade de armazenamento

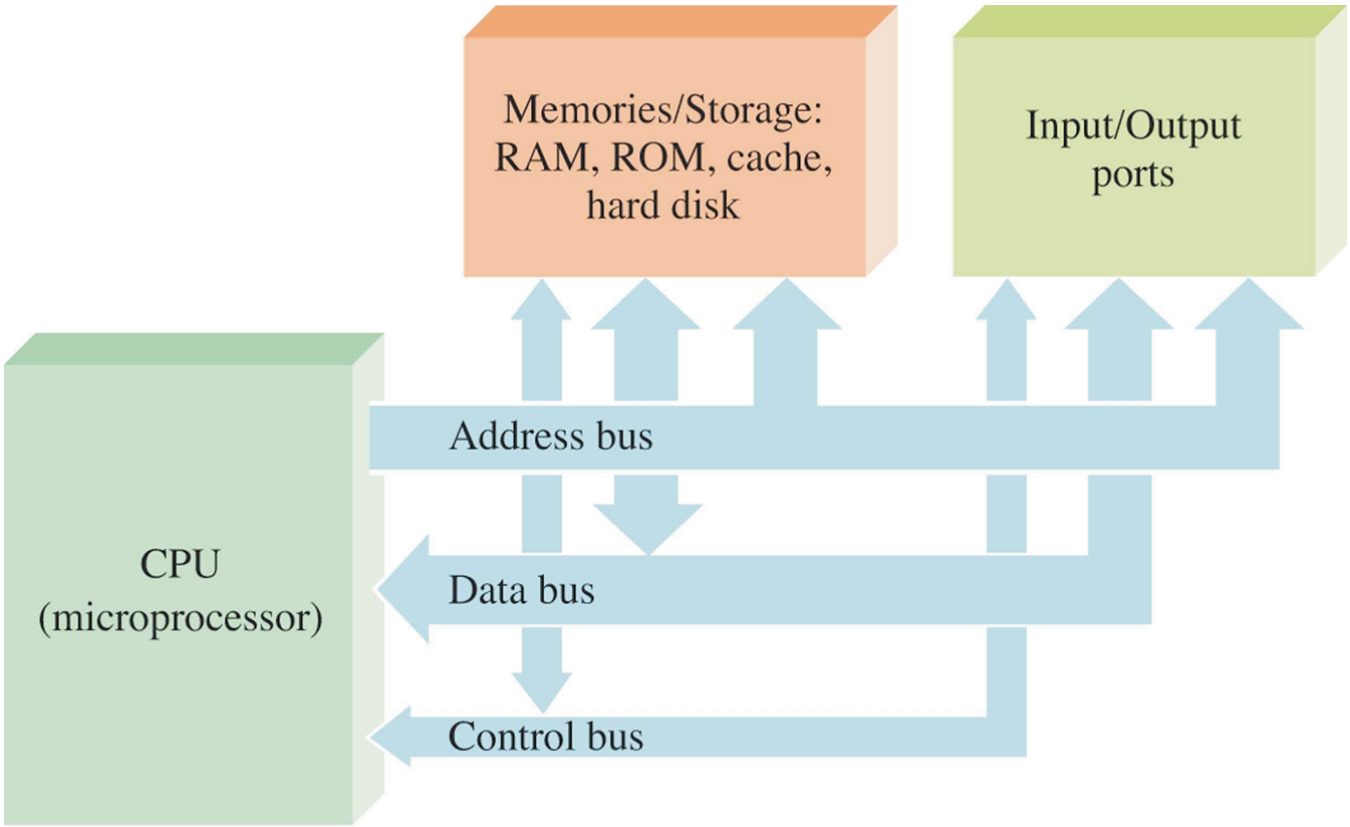
Dispositivos de Entrada e Saída

- Compostos de 2 partes:
  - Parte mecânica: **Dispositivo de E/S** realmente
  - Parte eletrônica: **Controlador de E/S**
    - Circuito que controla operação do dispositivo de E/S
    - Controlador fica conectado ao barramento
    - Dispositivo é ligado ao controlador por um cabo

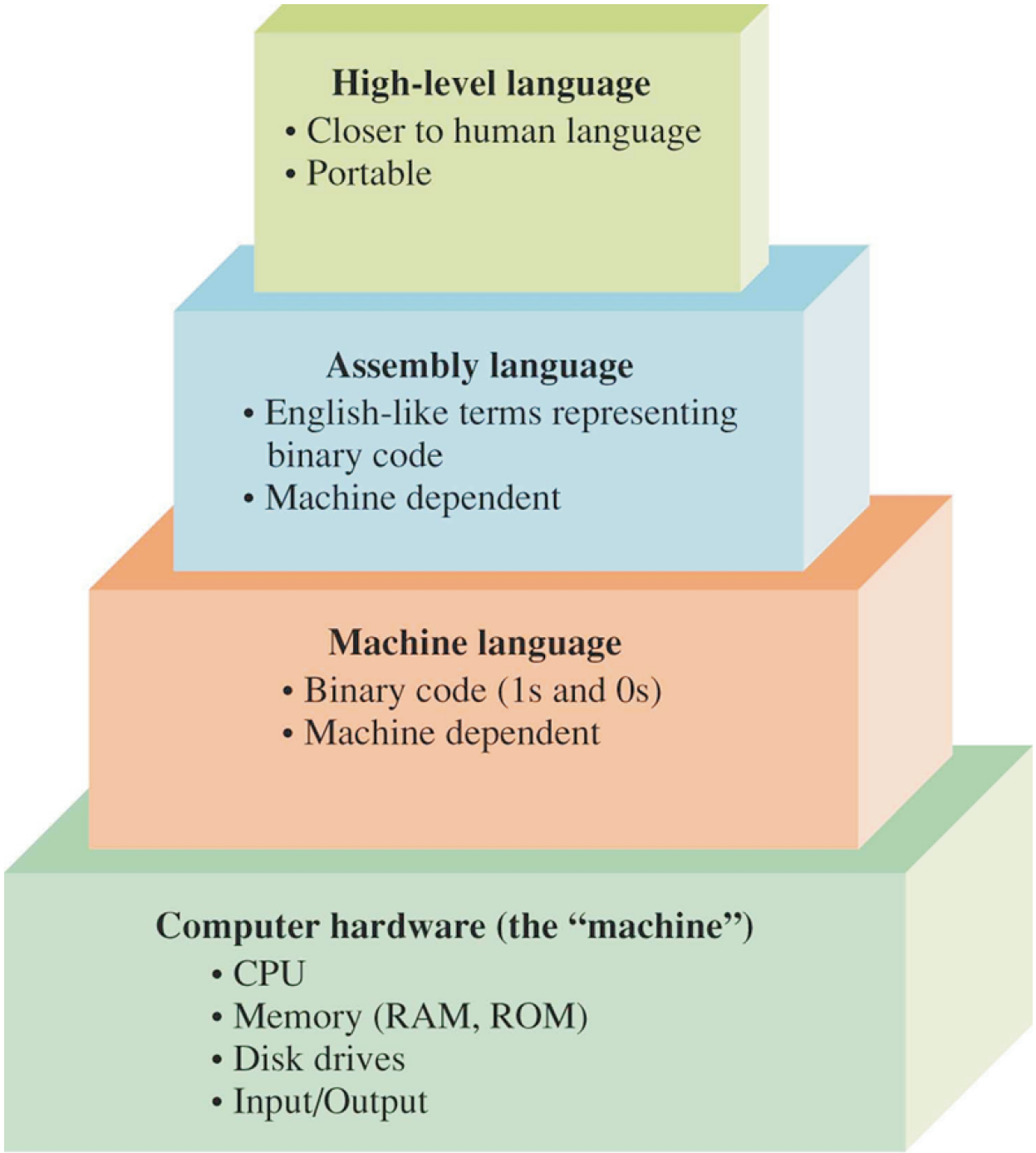


Barramentos

- **Conjunto de fios** por onde trafegam informações do processador para memória e para dispositivos de E/S e vice-versa
- Informações podem ser:
  - Dados
  - Endereços de memória
  - Sinais de controle



Interface Hardware/Software

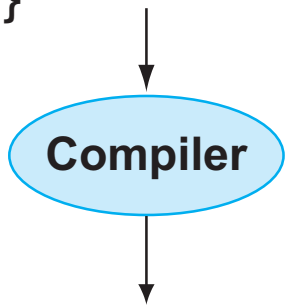


Interface Hw/Sw

- **Linguagem alto nível**
  - Tradutor: **Compilador**
- **Linguagem de montagem**
  - Tradutor: **Montador**
- **Linguagem de máquina**

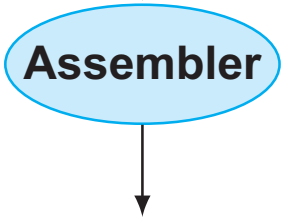
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```



Assembly language program (for MIPS)

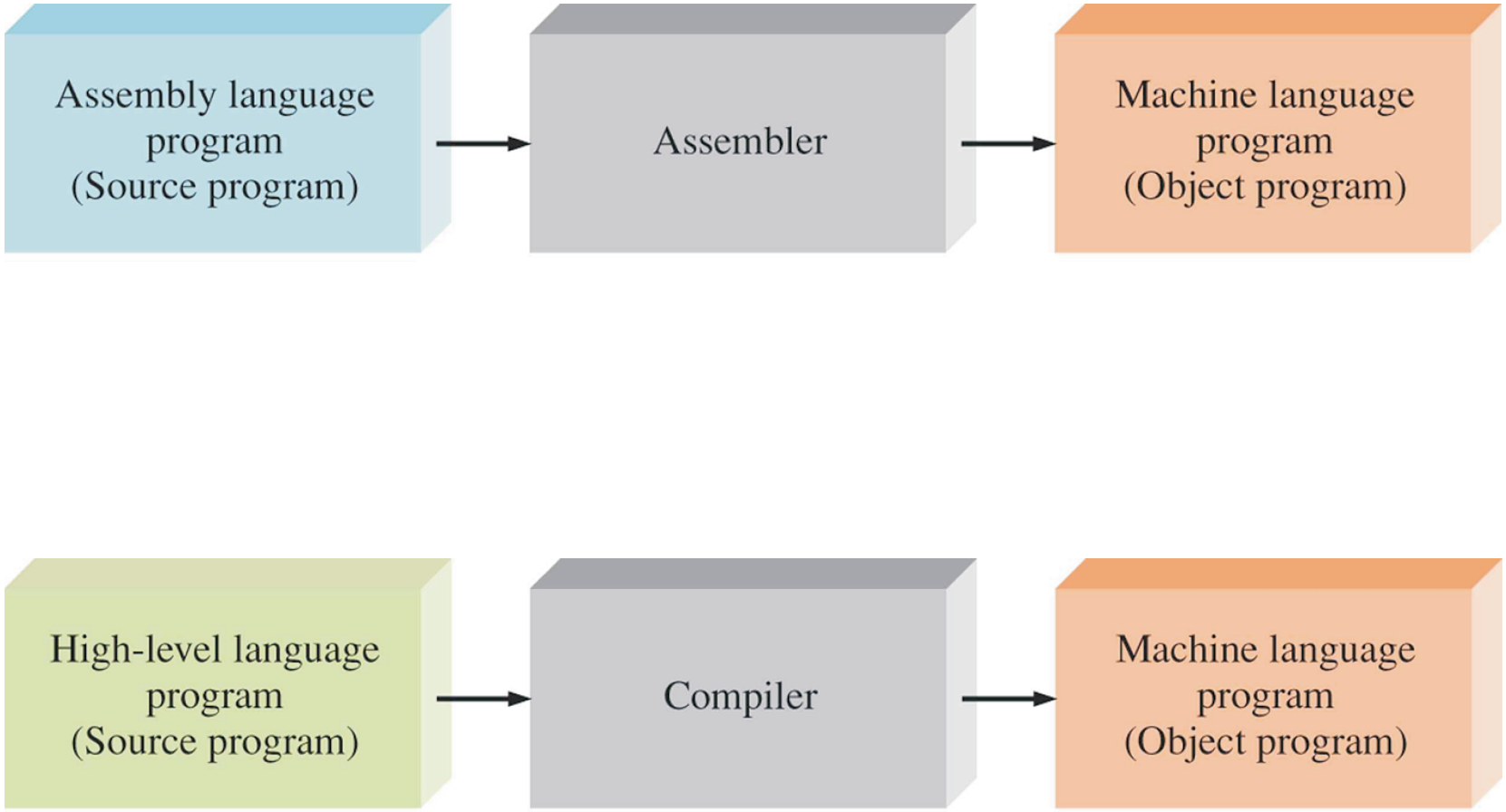
```
swap:   muli $2, $5,4
        add  $2, $4,$2
        lw   $15, 0($2)
        lw   $16, 4($2)
        sw   $16, 0($2)
        sw   $15, 4($2)
        jr   $31
```



Binary machine language program (for MIPS)

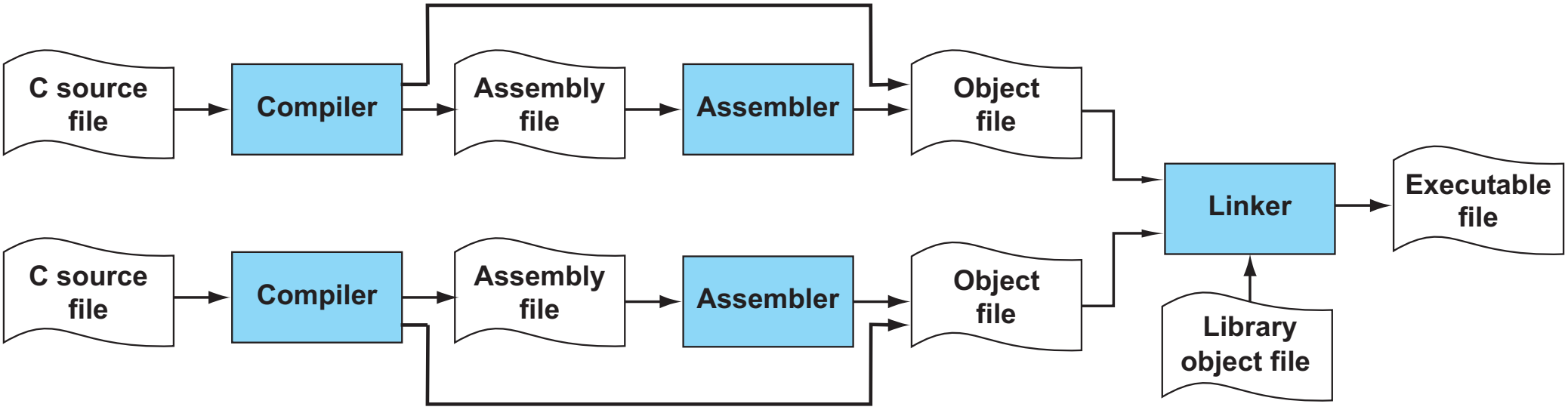
```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Geração do Programa Executável



Geração do Programa Executável

- A partir do programa fonte em uma linguagem alto nível (por exemplo C):



- **Passos**:
  - Compilação
  - Montagem
  - Link-edição

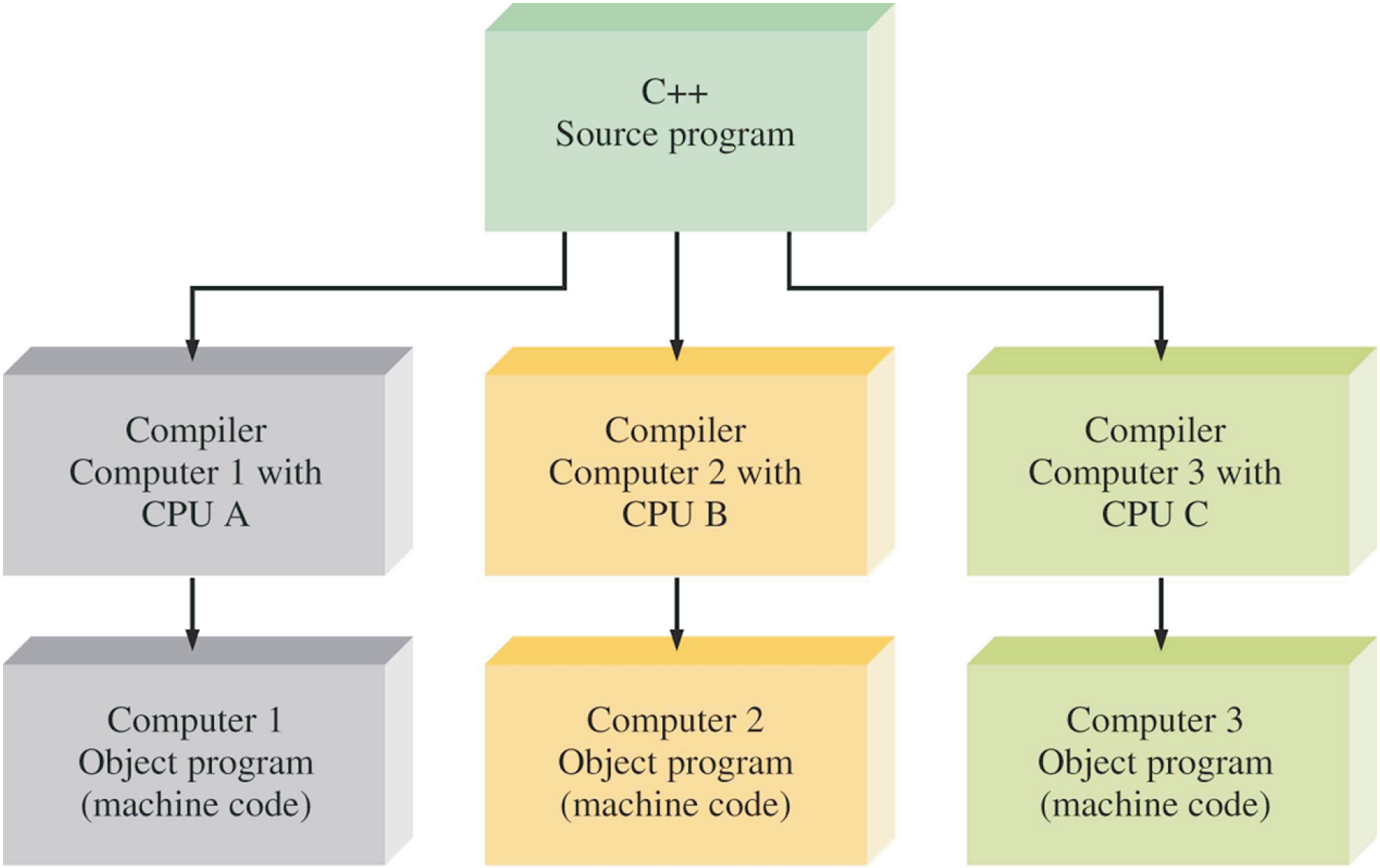
Passos para Geração do Programa Executável

- **Compilação**:
  - Realizada pelo programa **compilador**
  - Traduz programa fonte da linguagem alto nível
    para programa em linguagem de montagem
  - Pode gerar diretamente programa em linguagem de máquina:
    - **Módulo objeto** (ainda não é o programa executável)

- **Montagem**:
  - Realizada pelo programa **montador** (assembler)
  - Traduz programa fonte da linguagem de montagem (assembly language)
    para programa em linguagem de máquina, gerando **módulo objeto**

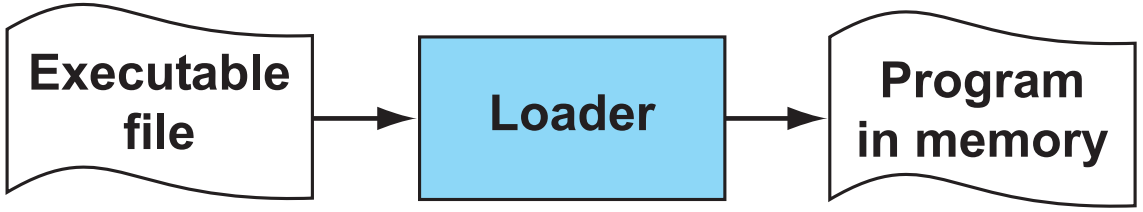
- **Link-edição**:
  - Realizada pelo programa **linker** (link-editor ou ligador)
  - Une vários módulos objeto e bibliotecas, gerando **programa executável**

Independência de Máquinado Programa em Linguagem Alto Nível



Execução do Programa

- **Programa armazenado** em memória:
  - Programa precisa ser carregado na memória para ser executado
  - **Instruções** de máquina e **dados** do programa são armazenados na memória, da onde são lidos para registradores do processador quando necessário
- **Carga** do programa executável na memória realizada pelo programa **loader**



- **Execução** do programa:
  - Programa é executado através da execução das suas instruções de máquina, uma a uma
  - Para cada instrução do programa a ser executada, processador realiza **ciclo de execução de uma instrução**

Ciclo de Execução de uma Instrução

- Processador realiza os passos:
  - **Busca instrução**: UC lê da memória próxima instrução a executar (cujo endereço está em PC) e copia-a para IR
  - **Decodifica instrução**: UC determina qual é a instrução, investigando conteúdo de IR
  - **Busca operandos**: UC lê operandos da instrução (de registradores ou da memória), se necessário
  - **Executa instrução**: ALU executa operação indicada pela instrução, utilizando operandos e gerando resultado
  - **Escreve resultado**: UC escreve resultado em registrador ou na memória (se necessário)
  - **Atualiza PC**: UC atualiza PC, para apontar para próxima instrução a executar

Instruções de Máquina

- **Instrução de máquina**:
  - Codificada como sequencia de bits
  - Representa comando que é interpretado e executado pelo processador
- **Exemplo 1**:
  - **Em C**:
    - Comando: `a = b + c ;`
    - `a`, `b`, `c` são variáveis do programa
  - **Em linguagem de montagem**:
    - Instrução: `add R8, R9, R10`
    - `R8`, `R9`, `R10` são registradores do processador
  - **Em linguagem de máquina**:
    - Instrução: `00000001001010100100000000100000`
    - Instrução de máquina dividida em **campos** que indicam operação e operandos

Exemplo 2: Execução de um Programa

- Em C:

```
b = a + a ;
```

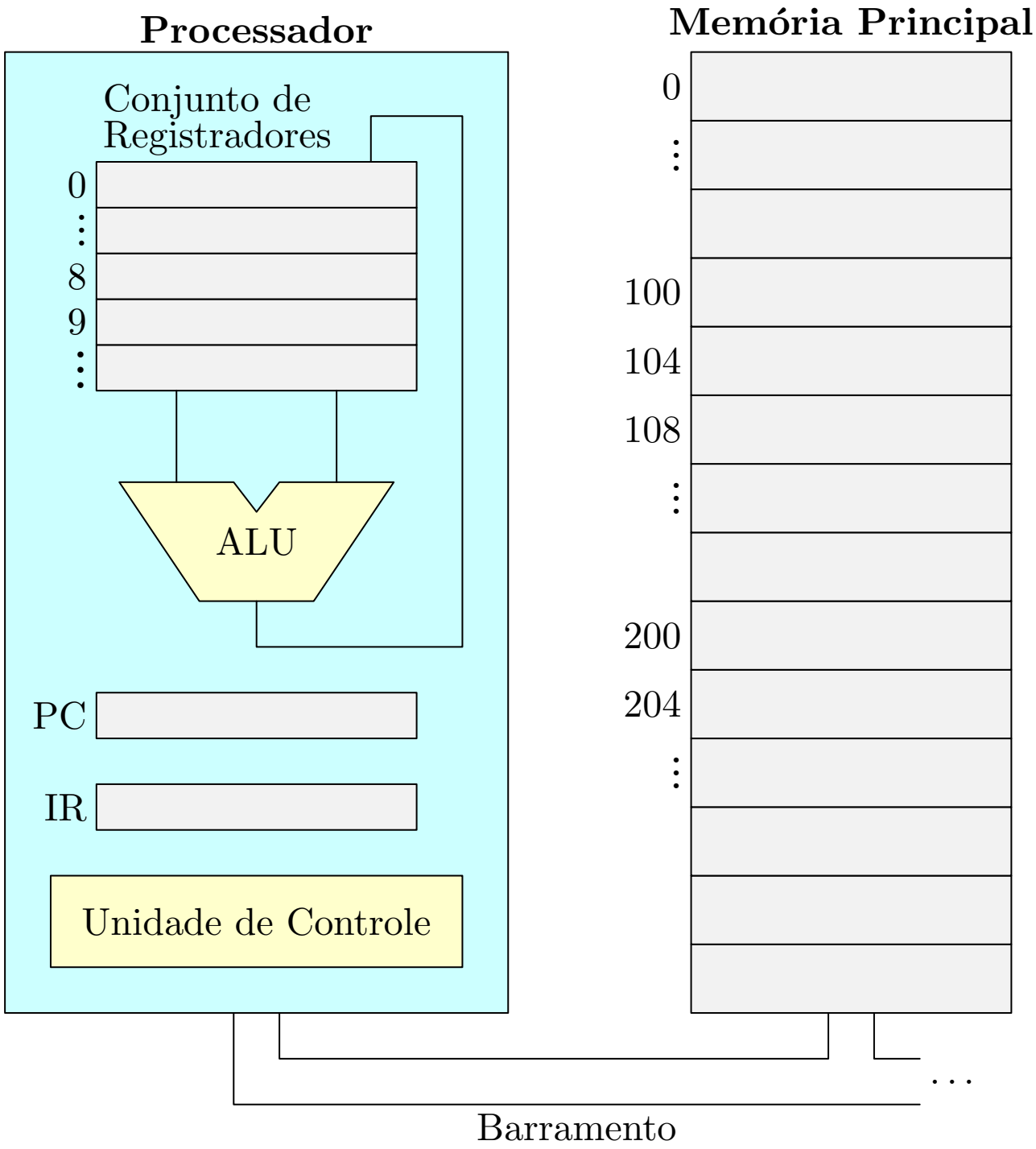
- Supondo que compilador associou variáveis `a` e `b` aos endereços 200 e 204
- Em linguagem de montagem:

```
load  R8, 200       % R8 := Mem[200]
add   R9, R8, R8    % R9 := R8 + R8
store R9, 204       % Mem[204] := R9
```

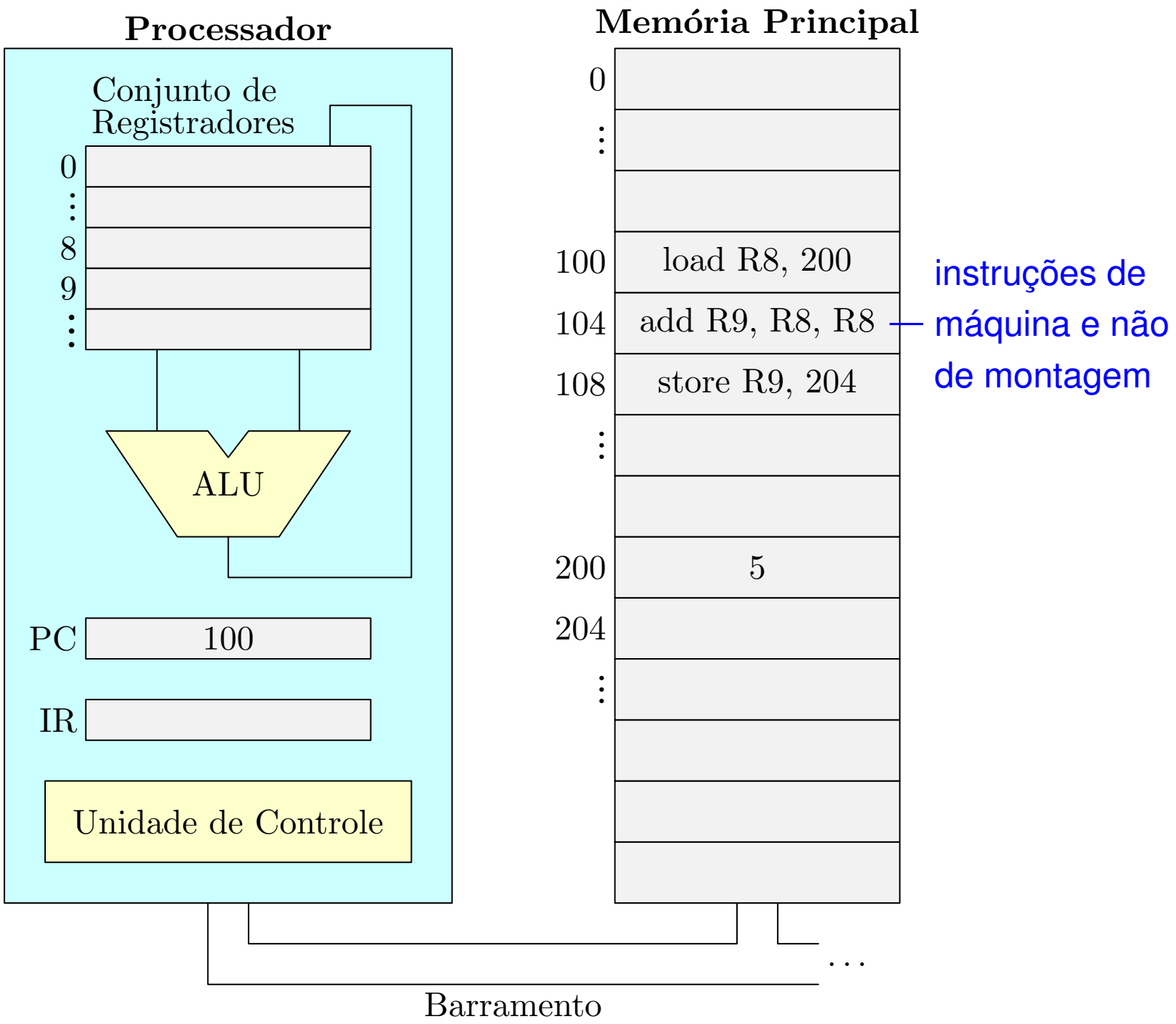
- Em linguagem de máquina (de um processador hipotético):

```
0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0
   load      R8                          200

0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
   add       R8      R8      R9

0 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0
   store     R9                          204
```

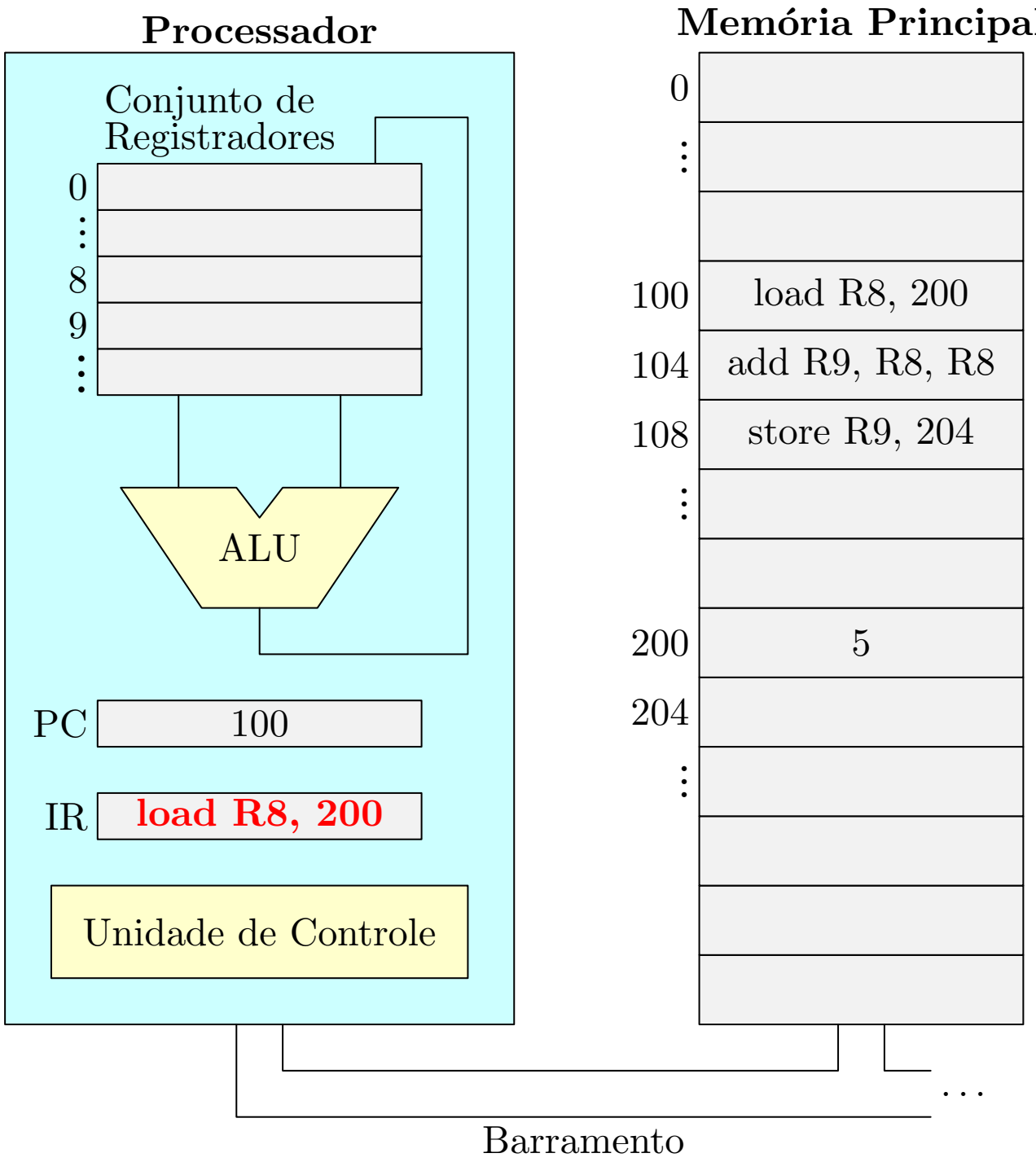
Exemplo 2: Computador



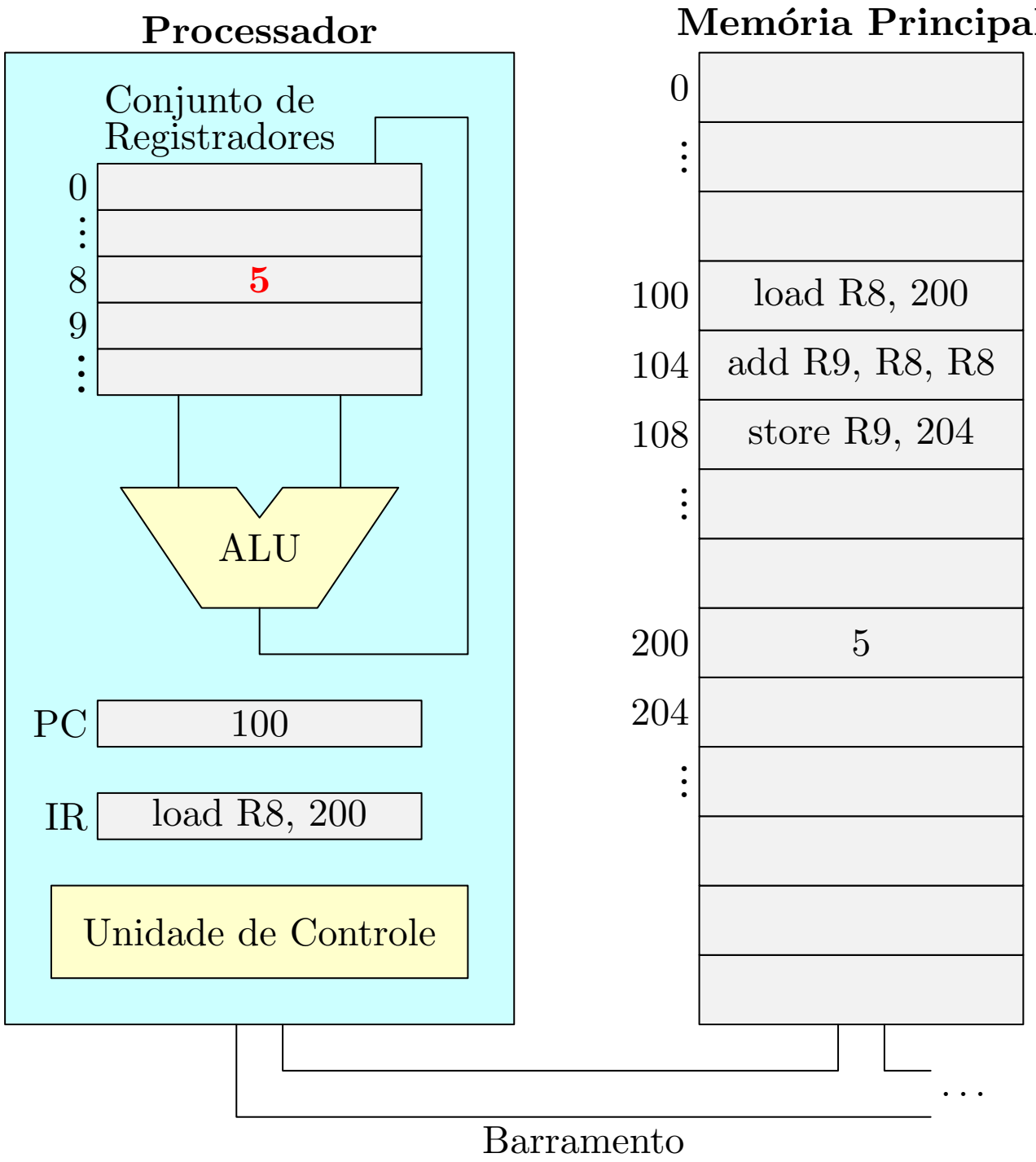
Exemplo 2: Programa carregado na memória



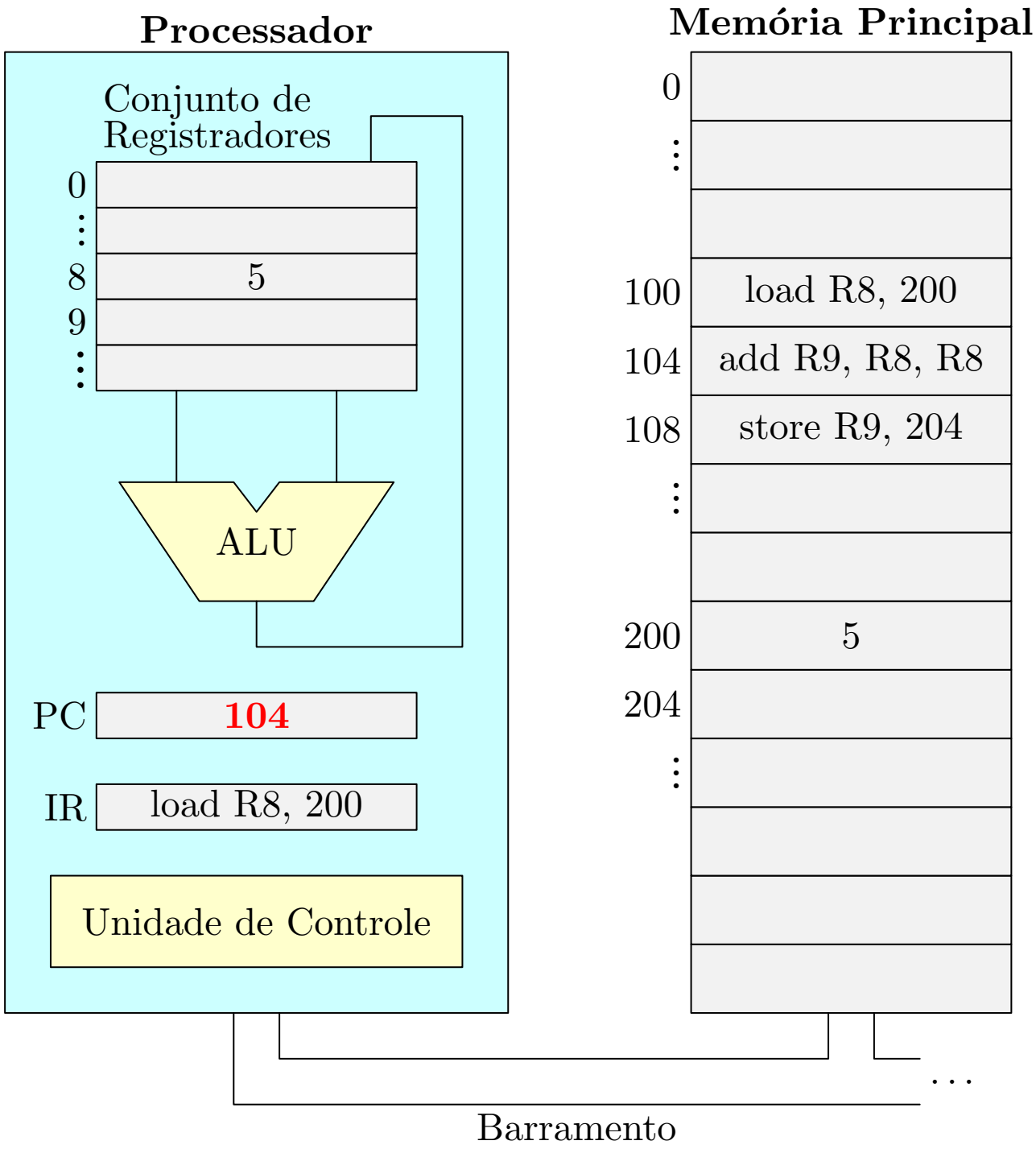
Exemplo 2: Instrução `load` – Busca instrução



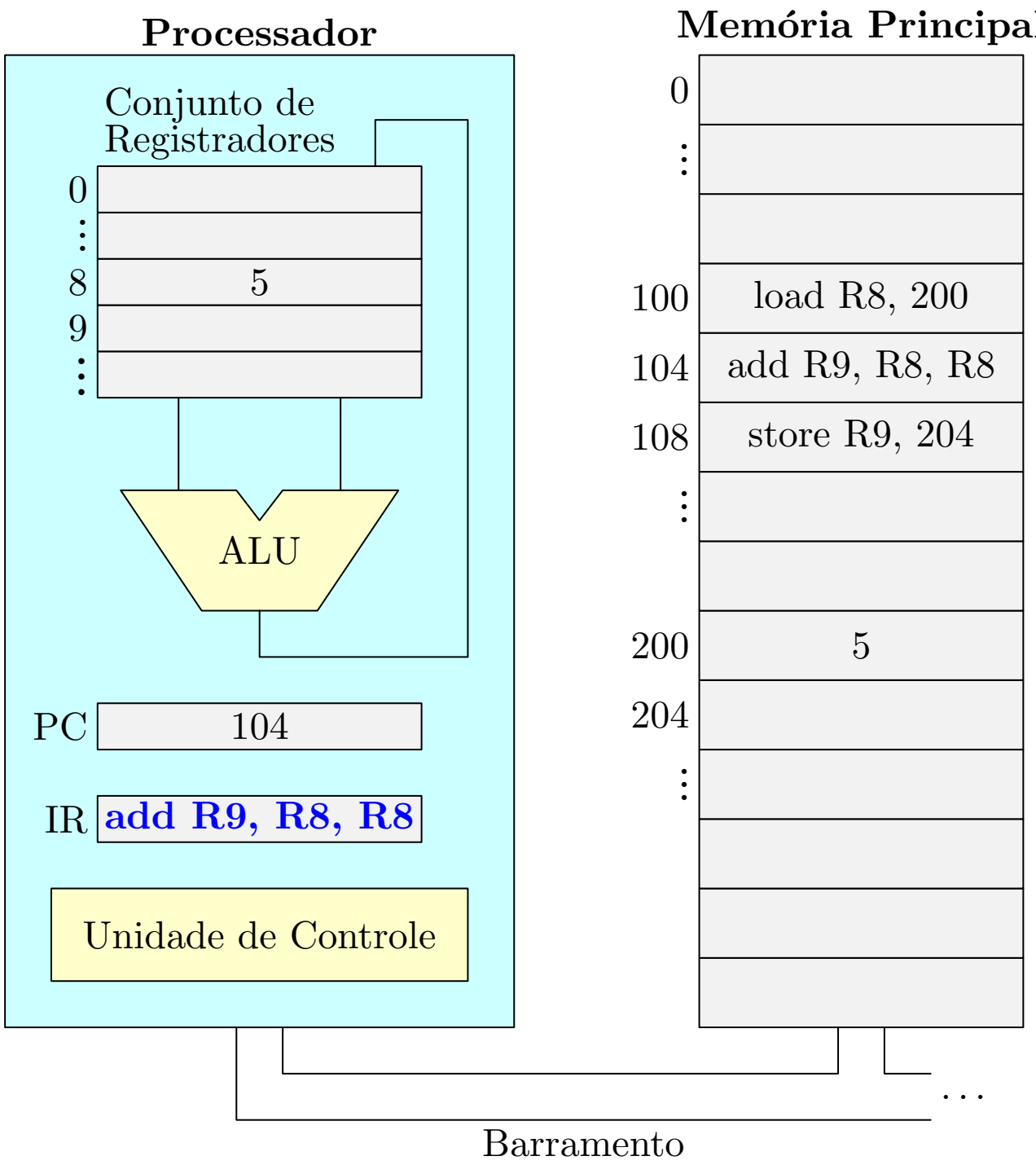
Exemplo 2: Instrução `load` – Executa instrução



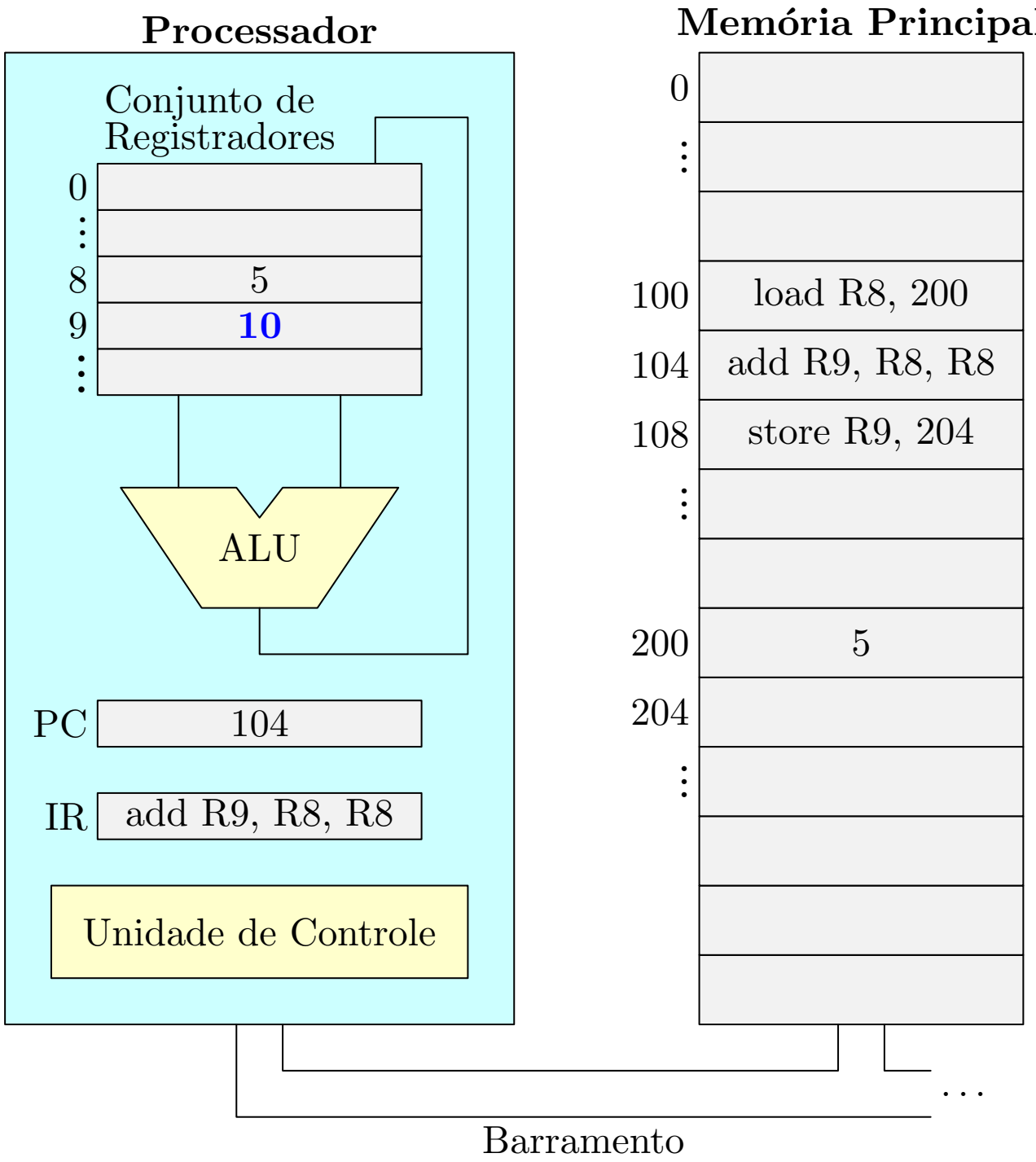
Exemplo 2: Instrução `load` – Atualiza PC



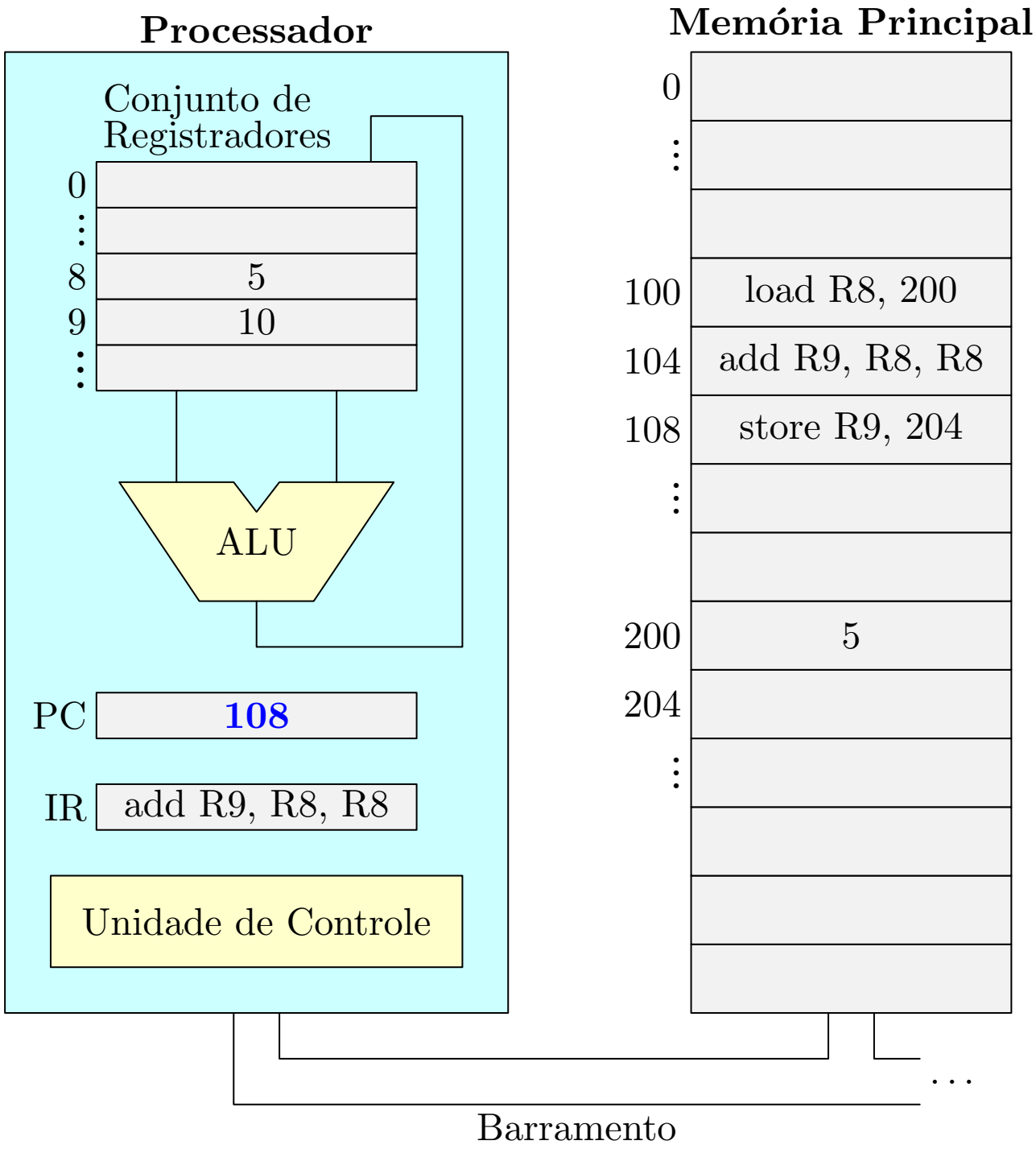
Exemplo 2: Instrução add – Busca instrução



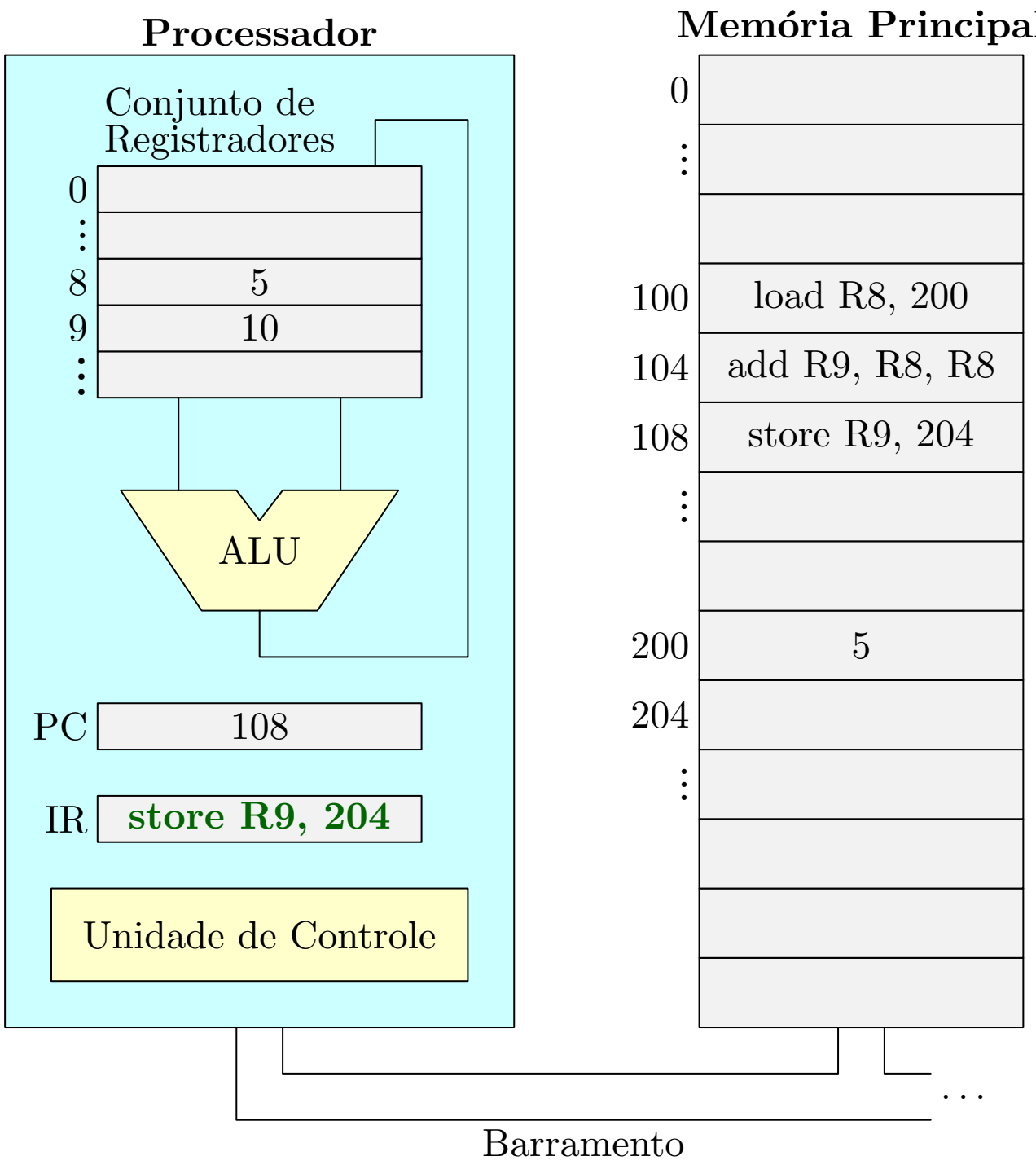
Exemplo 2: Instrução add – Executa instrução



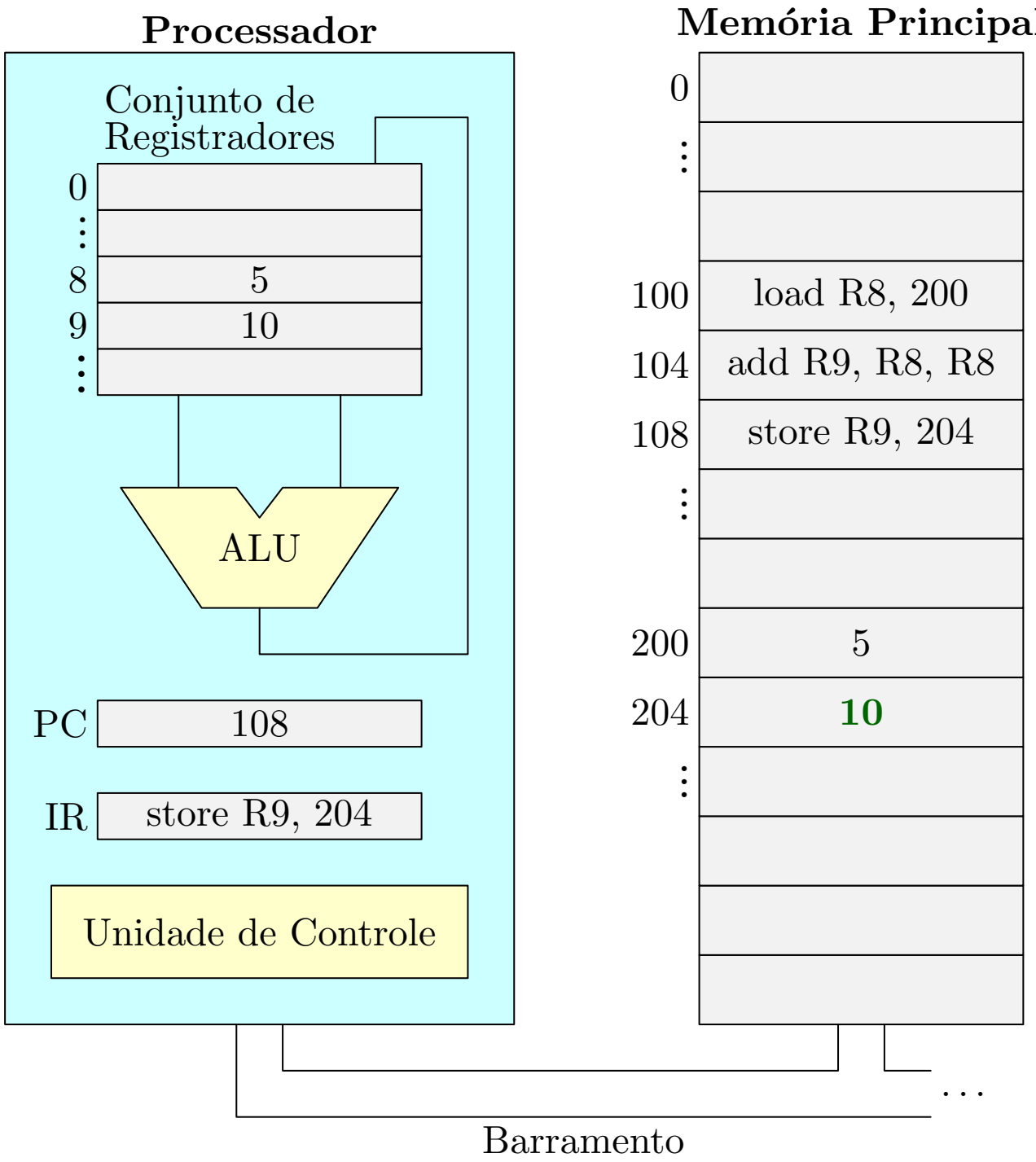
Exemplo 2: Instrução add – Atualiza PC



Exemplo 2: Instrução `store` – Busca instrução



Exemplo 2: Instrução `store` – Executa instrução



Exemplo 2: Instrução `store` – Atualiza PC