QXD-005 - Arquitetura de Computadores

Visão de Alto Nível do Computador

Prof. Pedro Botelho

Nas aulas passadas…

- Introduzimos alguns conceitos da arquitetura de computadores
- Entendemos a evolução dos computadores
- Estudamos o sistema binário e suas aplicações
- **Pergunta**: Como funciona um computador internamente?

Nesta aula…

- Modelo de John Von Neumann
- Componentes do Computador
- O Computador IAS

Visão de Alto Nível do Computador

Programa Armazenado

Computadores com programa fixo

- Dos primeiros computadores até o ENIAC, o **programa** era feito em *hardware*
  - Programa ⟶ Sequência de etapas a se realizar
- Inconvenientes do ENIAC:
  - Religação de componentes eletrônicos através de **cabos**
  - Dificuldades para corrigir programas já existentes
  - O tempo de alteração girava em torno de 3 semanas
  - Utilização de **aritmética decimal**
  - Dificuldades em distinguir de maneira confiável os 10 níveis de tensão

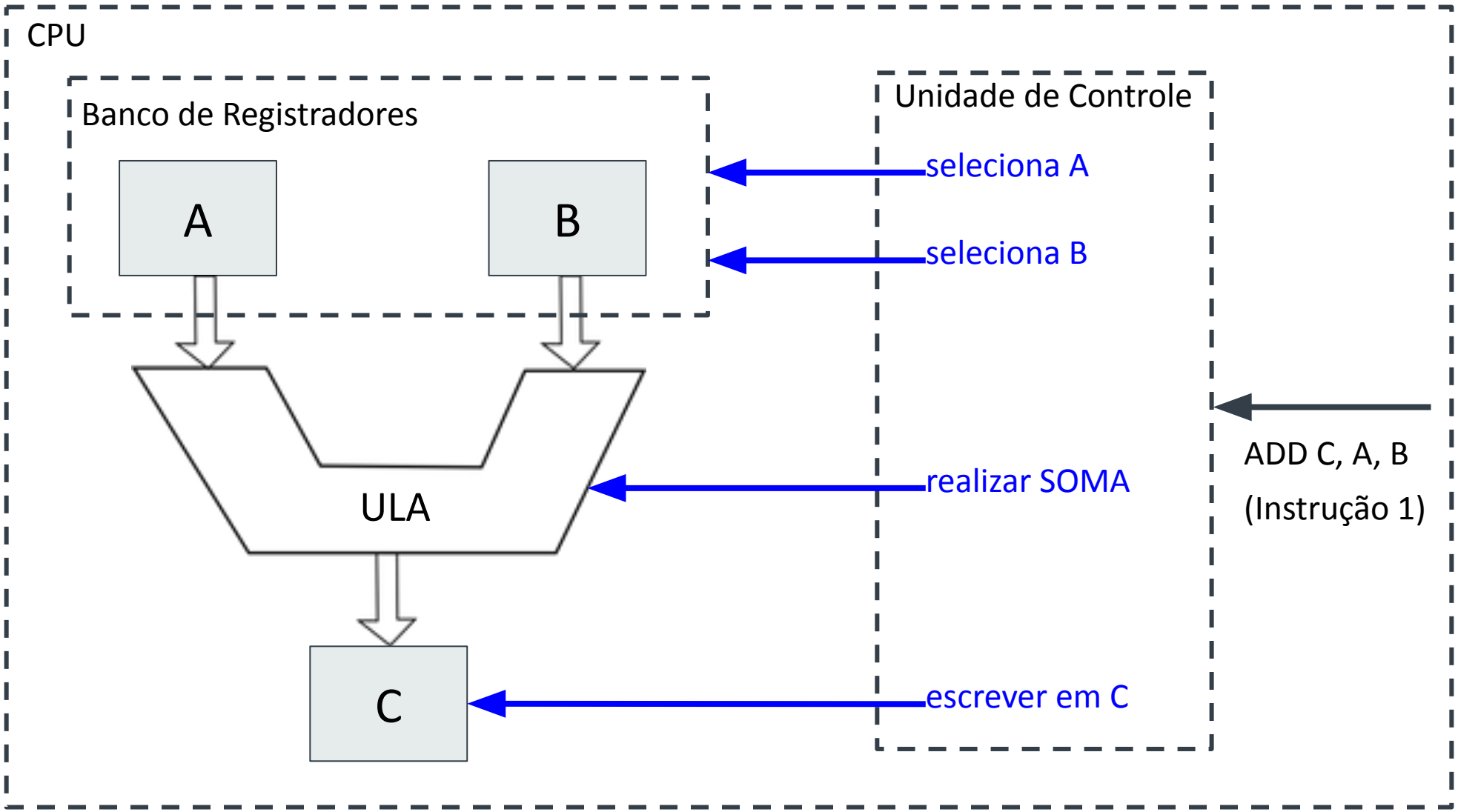
Programas fixos?

- Sistemas "*hardwired*" são inflexíveis
- *Hardware* de uso geral pode fazer diferentes tarefas, dado sinais de controle corretos
  - Ao invés de religar o *hardware*, forneça um conjunto de sinais de controle
  - Exemplo: Sinais de seleção de operação lógica e aritmética
- **Programa** ⟶ Uma sequência de etapas
  - Para cada etapa, é feita uma operação aritmética ou lógica ⟶ **Instrução**
- Para cada operação, é necessário um conjunto diferente de sinais de controle

Exemplo: Soma entre valores

Programa:

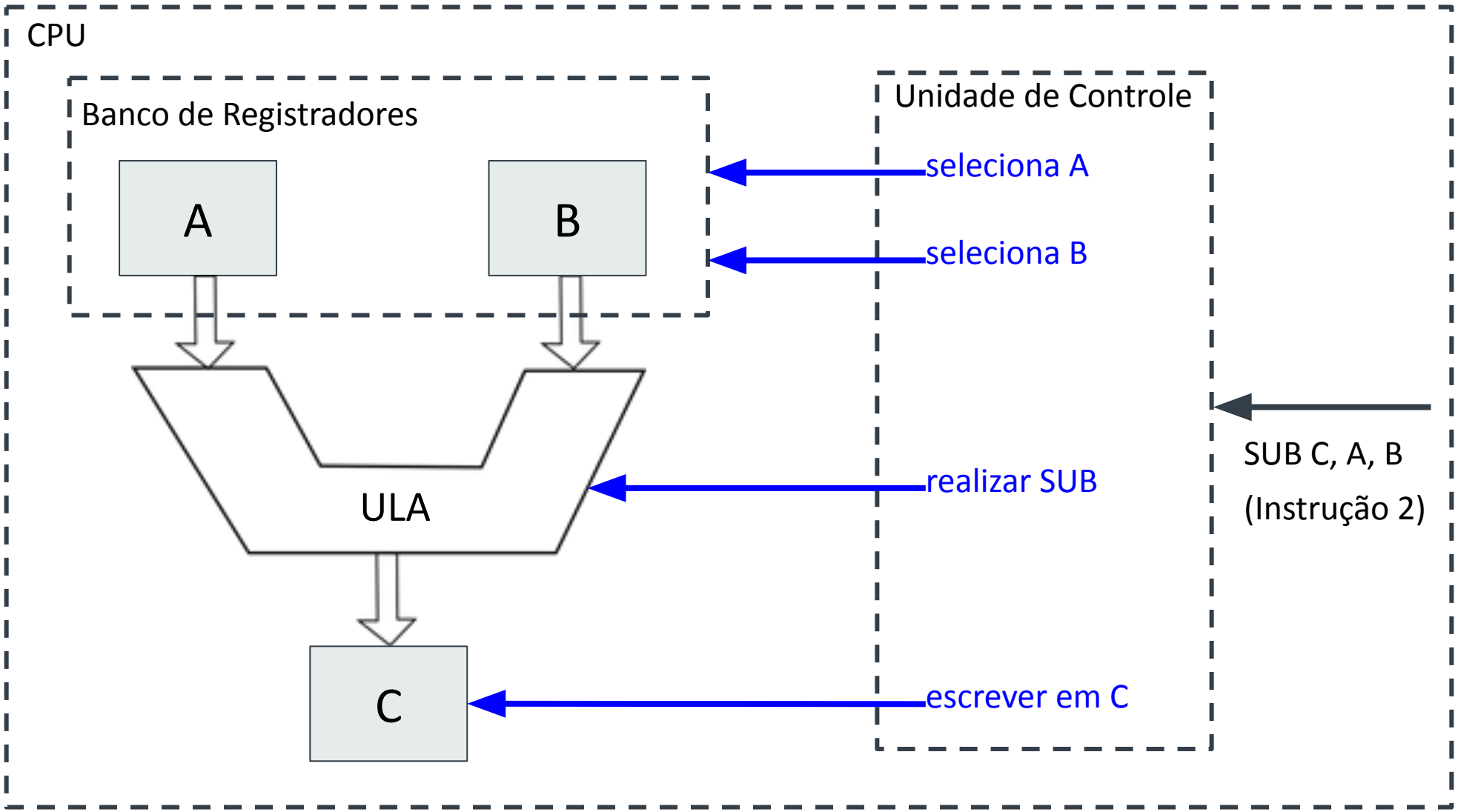
ADD C, A, B
SUB C, A, B



Exemplo: Soma entre valores

Programa:

ADD C, A, B
SUB C, A, B



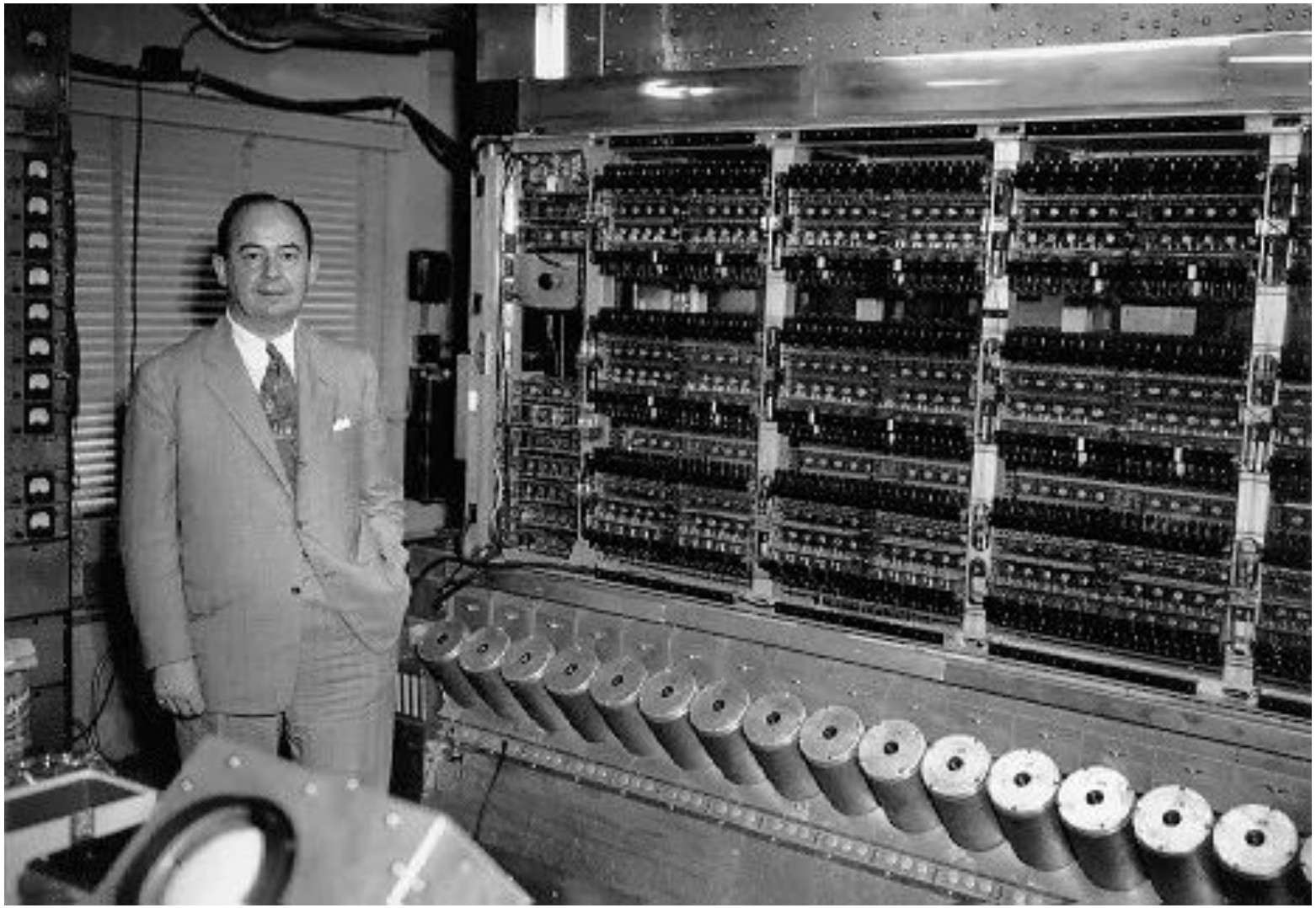
Visão de Alto Nível do Computador

Modelo de John Von Neumann

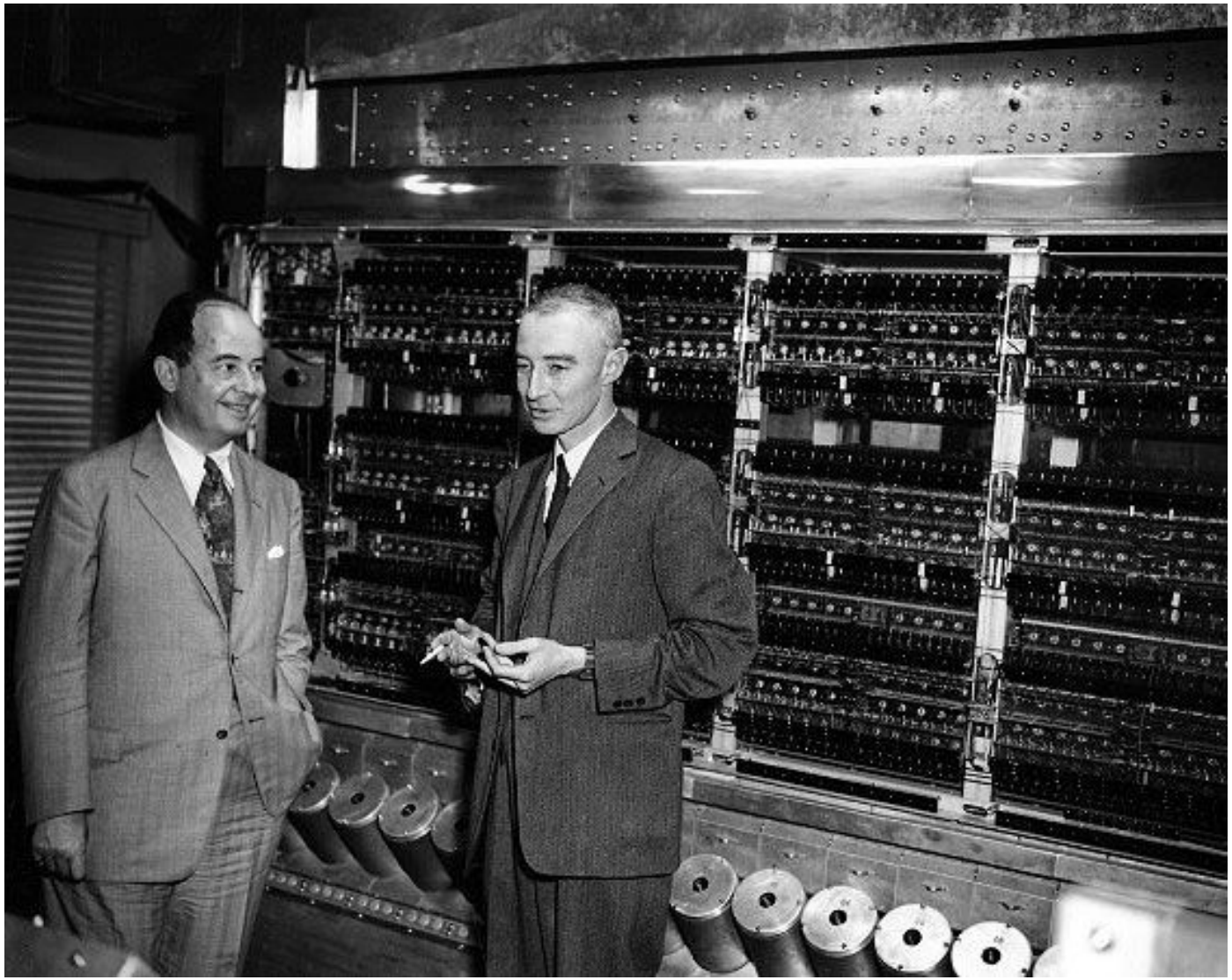
Arquitetura de von Neumann

- Programa e dados são armazenados juntos na memória principal
  - Dados e instruções armazenados na **mesma memória**
- O programa é executado sequencialmente
- A memória é **endereçável**
- Programas e dados representados de forma digital em memória
- Processamento baseado em **aritmética binária**, ao invés de decimal
- Forma a base de todos os computadores digitais
- Publicada em 1945 no *First Draft of a Report on the EDVAC* por John von Neumann (1903-1957)

John von Neumann e o computador IAS



John von Neumann, o computador IAS e quem mais?

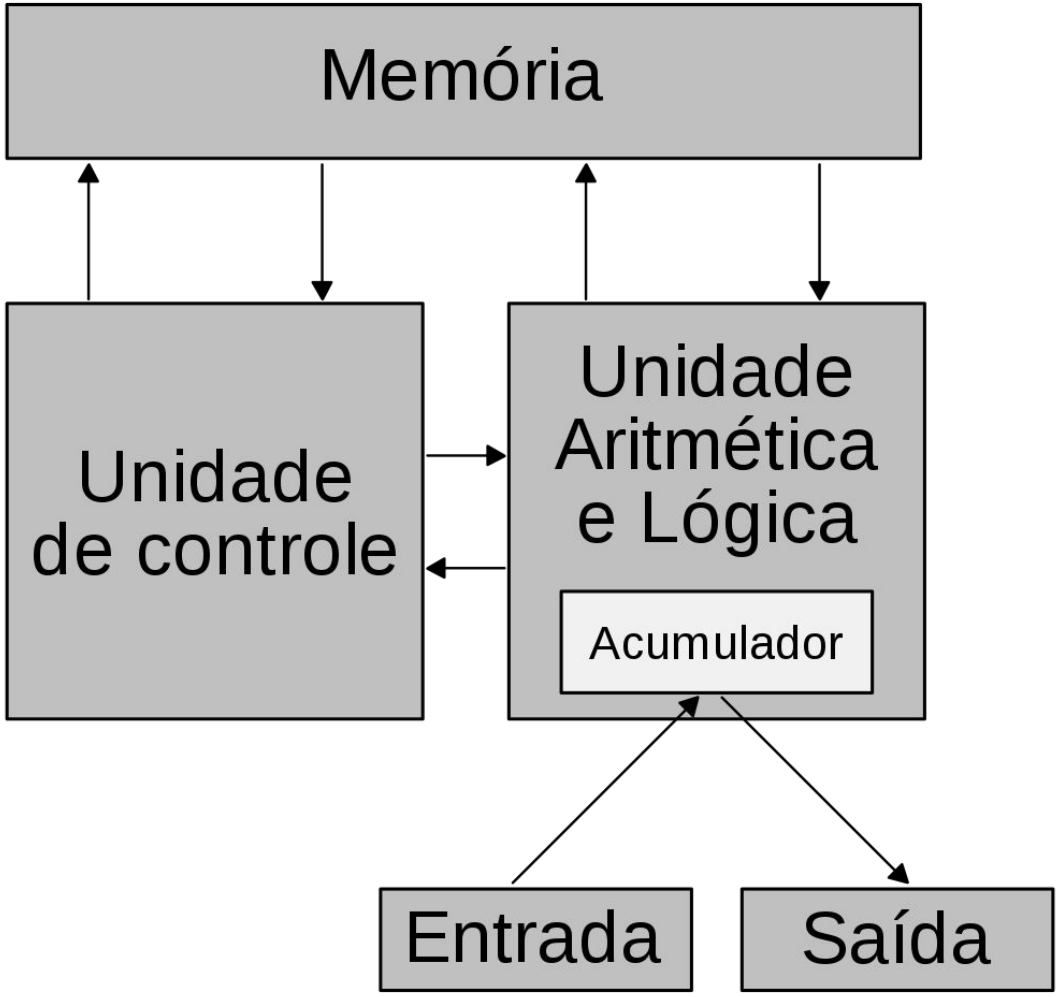


Arquitetura de von Neumann

- A arquitetura de Von Neumann consiste em três sistemas de *hardware*:
  - **Unidade Central de Processamento (CPU)**⟶ Realiza o processamento de dados e controle
    - Capaz de executar processamento sequencial de instruções por meio de vários componentes:
      - Unidade de Controle (UC)
      - Unidade Aritmética e Lógica (ULA)
      - Registradores (e.g. AC)
      - Contador de Programa (PC)
  - **Sistema de Memória Principal** ⟶ Armazena instruções (o programa) e dados
    - Contém um único caminho entre a memória principal e a CPU (barramento)
  - **Sistema de E/S** ⟶ Permite interação com o computador
    - Dados e instruções precisam entrar no sistema, e resultados precisam sair dele

Máquina de Von Neumann

- A base de quase todos os computadores digitais



Visão de Alto Nível do Computador

Componentes do Computador

Unidade Central de Processamento (CPU)

- Lê, decodifica e executa as instruções armazenadas na memória ⟶ Ciclo de execução
  - Unidade de Controle (UC) envia os sinais de controle necessários
- Realiza operações lógicas e aritmética (na ULA)
- Pode ler ou modificar o valor de alguma posição de memória
- Pode transferir valores da entrada para a memória e da memória para a saída
  - Entrada ⟶ Memória
  - Memória ⟶ Saída
- Executa as instruções sequencialmente até o término do programa

Memória

- É dividida em células (bytes)
- Cada célula armazena um valor e possui um endereço
- É controlada pela CPU, que informa…
  - …em qual endereço deseja executar uma operação
  - …qual a operação desejada → leitura ou escrita
- Operações de leitura não modificam os valores armazenados
- Operações de escrita apagam os valores anteriores, substituindo-os pelos novos valores

Exemplo de memória

- As *n* posições são numeradas de 0 a *n* - 1
  - 10 posições ⟶ Posições de 0 a 9
- Cada posição armazena *m* bits
  - 8 bits ⟶ 256 combinações de valores!
- Dizemos que é uma memória *n* x *m* ⟶ No caso, 10 x 8
- Valores permanecem até que um novo valor seja armazenado na mesma posição ou até que a memória seja desligada
  - E as memórias que mantém os dados mesmo desligadas?

0	0111_0000
1	0011_0000
2	0011_0000
3	1101_0000
4	1110_1000
5	1100_0110
6	0000_0001
7	0111_1100
8	0011_0110
9	1001_0001

Entrada e Saída (E/S)

- Sob comando da CPU, é possível ler valores
  - …de um dispositivo de entrada, armazenando-os na memória (**operação de entrada**)
  - …da memória, enviando-os para um dispositivo de saída (**operação de saída**)
- Existe uma grande variedade de dispositivos de entrada e saída
- Obedecem ao comando da CPU
  - Exemplo: Ler porta de entrada do teclado ou escrever na porta de saída do monitor

Registradores

- Registrador ⟶ Células que armazenam uma quantidade fixa de bits (e.g. 32 bits)
- Podem ser de:
  - **Propósito específico**: Só pode ser usado para uma função (e.g. manter um endereço)
  - **Propósito geral**: Podem ser usados para várias operações (e.g. acumulador, ou AC)
- Alguns processadores possuem poucos registradores de propósito geral ⟶ **IAS** tem 2
- Outros possuem vários registradores de propósito geral ⟶ **MIPS** tem 32!
  - Possui então um **banco de registradores** ⟶ Conjunto de registradores selecionáveis

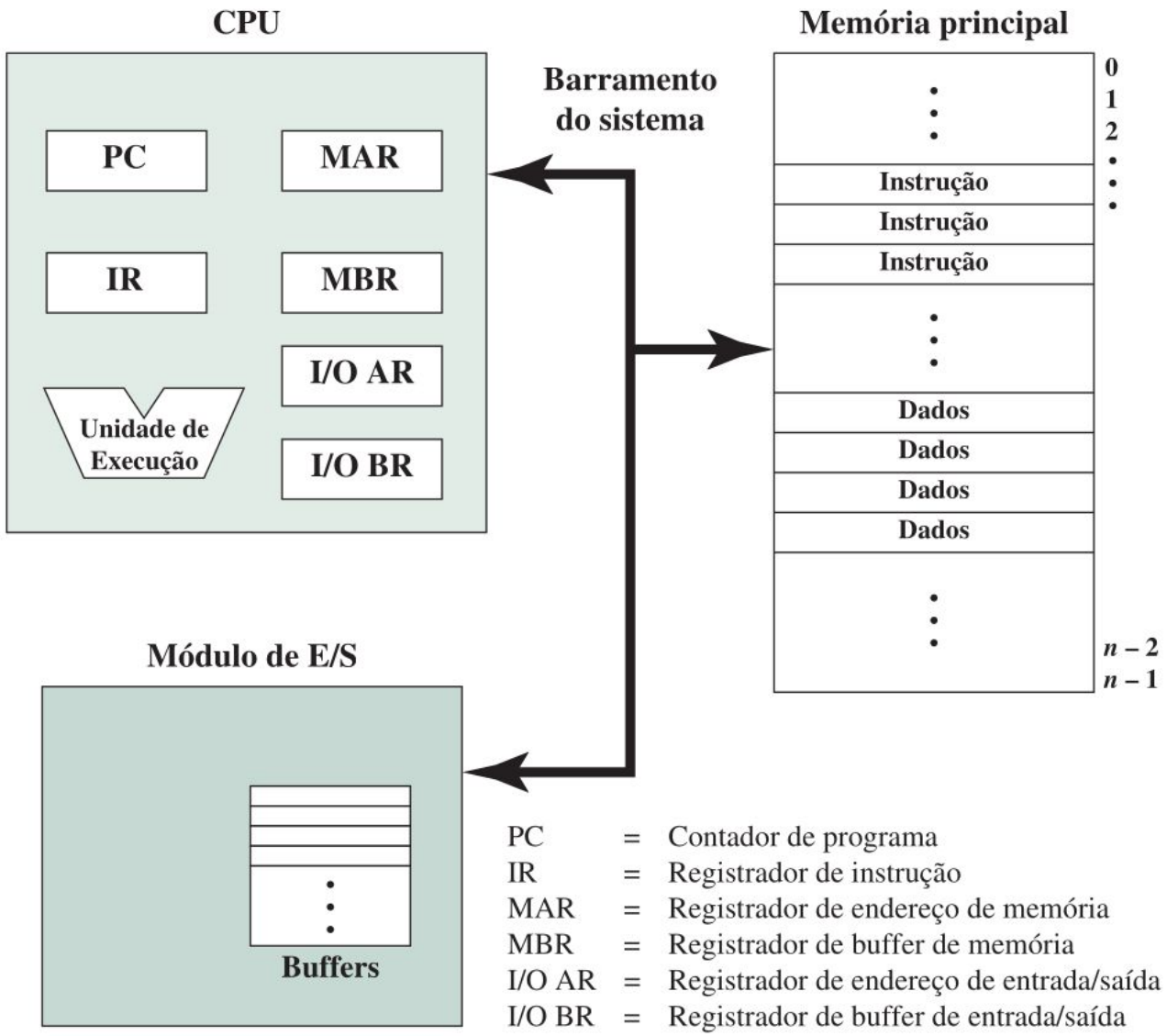
Principais registradores da CPU

- A Unidade Lógica Aritmética (ULA) contém…
  - **Reg. Temporário de Dados (MBR)**: contém o dado a ser lido ou escrito na memória
  - **Acumulador (AC)**: contém o dado que está sendo “acumulado” no momento
- A Unidade de Controle (UC) também tem registradores especiais…
  - **Reg. de Endereço de Memória (MAR)**: contém o endereço da posição da memória a ser acessada
  - **Reg. de Instruções (IR)**: contém a instrução que será executada
  - **Program Counter (PC)**: contém o endereço na memória com a próxima instrução a ser executada.
- Os dois formam a Unidade Central de Processamento (CPU) dos computadores modernos

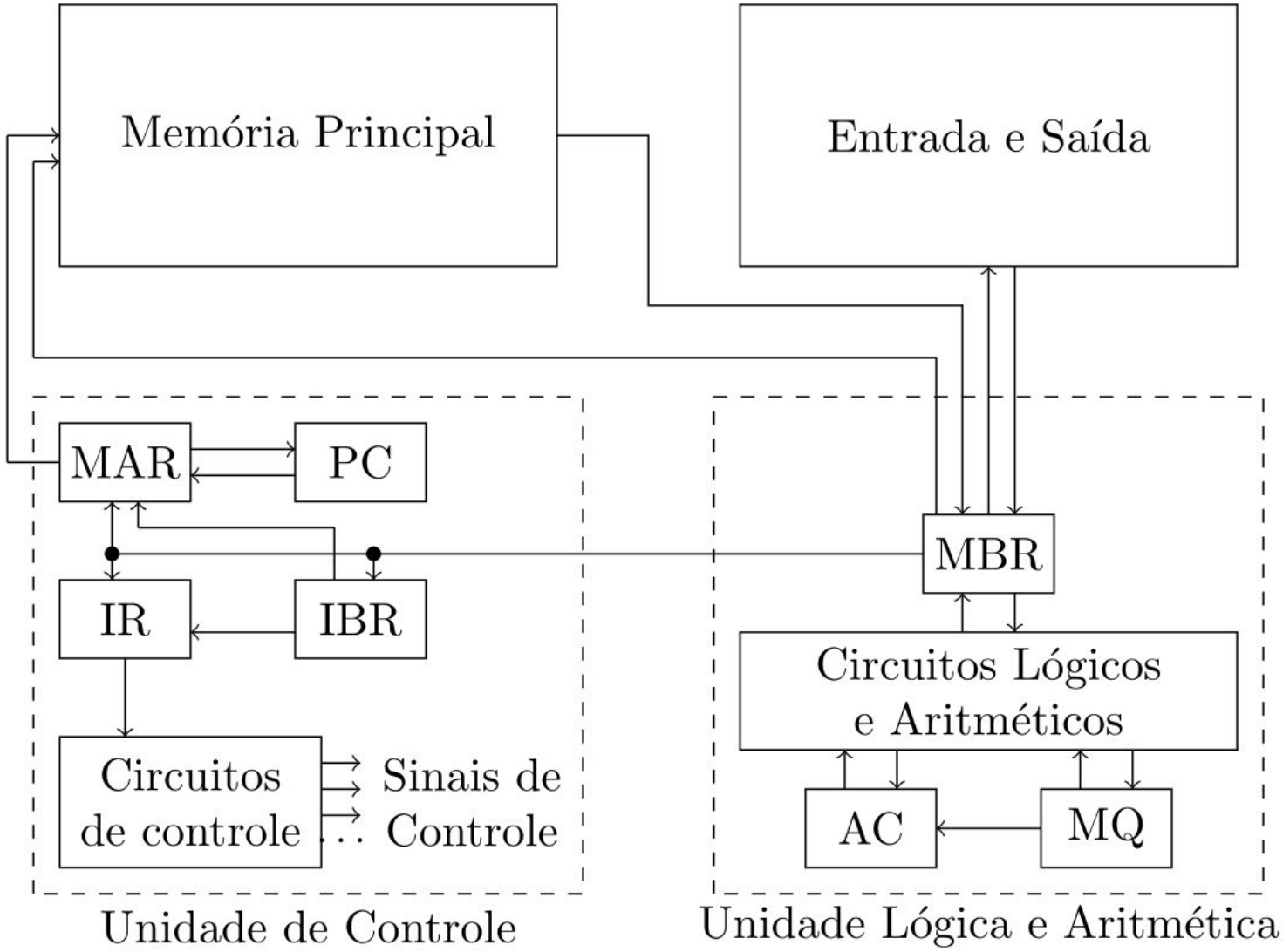
Barramentos

- Caminhos físicos que interligam os componentes ⟶
  - **Paralelo** ⟶ Dados enviados de uma vez e.g. 32 bits acarreta 32 fios
  - **Serial** ⟶ Dados enviados bit-a-bit por um único fio
- Barramentos **internos** ⟶ Interligam componentes internos à outro componente e.g. CPU
  - Barramento interno da CPU interliga registradores, ULA e UC
- Barramentos **externos** ⟶ Interligam os componentes do computador
  - Barramento do sistema ⟶ Barramentos simples que interligam todos os componentes
- Podem haver vários…
  - Barramento principal ⟶ CPU à memória (paralelo)
  - Barramento PCIe ⟶ Dispositivos de E/S ao barramento principal, logo, memória e CPU (serial)

Diagrama de alto nível do computador



Exemplo: Organização simplificada do IAS



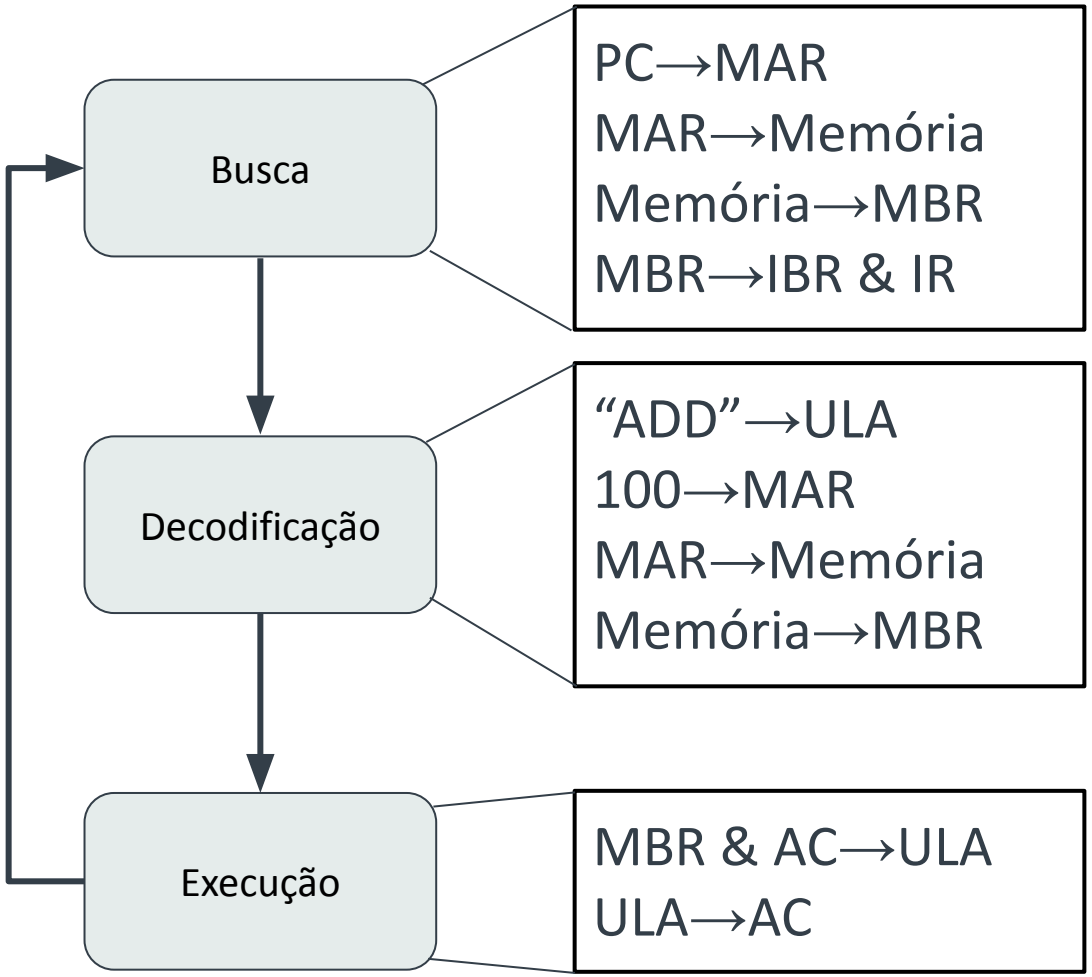
Visão de Alto Nível do Computador

Características da Execução

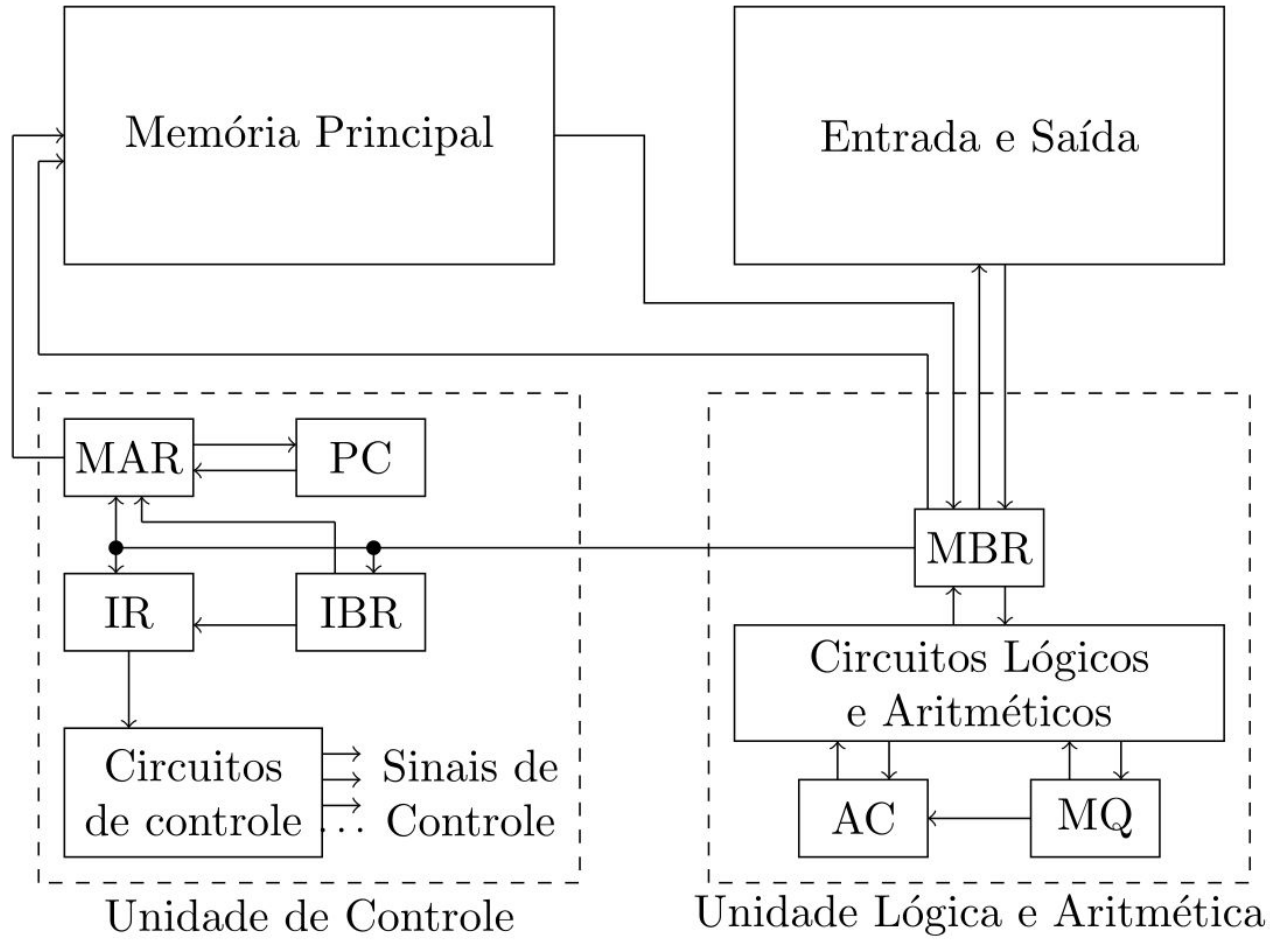
Ciclo de execução de von Neumann

- Duas coisas a se lembrar…
  - Todas trocas de dados passam pela ULA (**MBR**)
  - Todos os endereços passam pela UC (**MAR** se for dados, e **PC** se for instruções)
- Ciclo de execução de instrução de Von Neumann:
  - **Busca** – **Decodificação** – **Execução**
- A UC **busca** a próxima instrução do programa na memória usando o PC
  - Determina onde a instrução está localizada
- A instrução é **decodificada** para uma linguagem que a ULA possa entender (sinais de controle)
  - Qualquer operando de dados requerido para executar a instrução é carregado da memória e colocados em registradores dentro da CPU
- A ULA **executa** a instrução e coloca os resultados em registradores ou na memória.

Exemplo: Ciclo de execução no IAS



Instrução: ADD M(100)



Função da unidade de controle

- Para cada operação, um código exclusivo é fornecido
  - Exemplo: ADD, MOVE, SUB.
- Um segmento de *hardware* aceita o código e emite os sinais de controle
  - Essa é a unidade de controle!

Gargalo de von Neumann

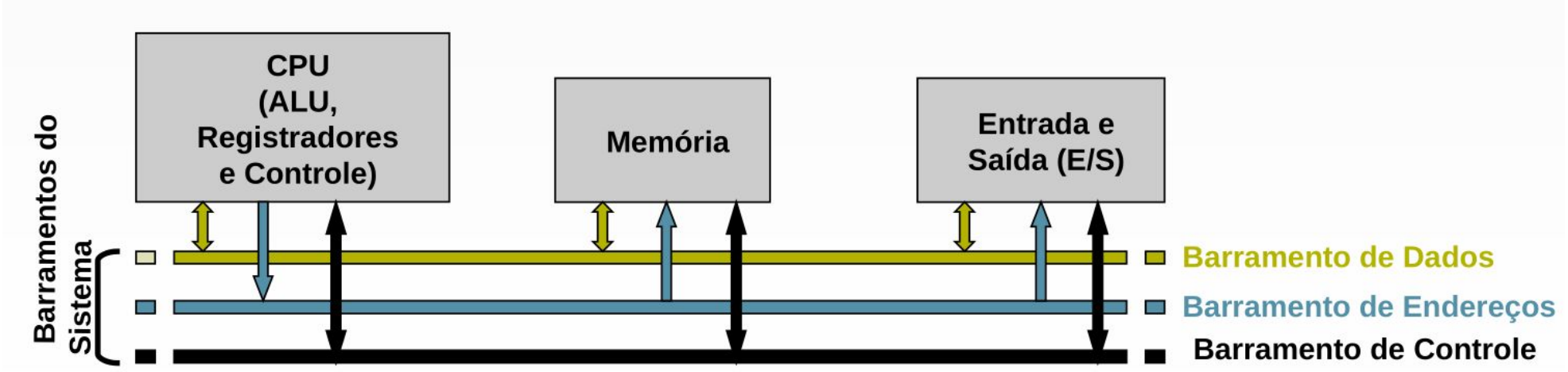
- A separação entre a CPU e a memória leva ao **gargalo de von Neumann**
  - Problema: Somente um caminho entre o sistema de memória principal e CPU
  - Força-se a alternância entre ciclos de **instrução** e **execução:**
    - …vai endereço da instrução volta instrução
    - …vão endereços dos operandos e voltam operandos
  - Transferência limitada entre a CPU e memória em comparação a quantidade de memória
- O termo "gargalo de von Neumann" foi dado por John Backus em sua palestra Award 1977 ACM Turing
- Este gargalo permanece até hoje!
- Exemplo: Aplicações de acesso de dados intensivo não preenchem 100% de processamento devido à diferença de desempenho entre processadores modernos e a memória principal

Gargalo atualmente

- A evolução das memórias é mais lenta que a evolução dos processadores:
  - **SDRAM** (100 - 133 MHz)
  - **DDR**: Double Data Rate, ou dupla taxa de transferência (266 MHz).
  - **DDR2**: 2 * DDR (até 1.300 MHz)
  - **DDR3**: 2 * DDR2 (até 2.400 MHz)
  - **DDR4**: 2 * DDR3, **DDR5**: 2 * DDR4, e por aí vai…
  - Latência da memória (de 50 a 100 ns) bem maior que o ciclo da CPU (0.5 ns)
- Para contornar o gargalo…
  - …diminuir tráfego de informações
  - …manter informações na CPU e inclusão de registradores
  - …diminuir tamanho em bits das informações transferidas
  - …separar as memórias, cada um com caminhos independentes (dados e instruções) → **Arquitetura de Harvard**

Extensões do modelo de von Neumann

- A primeira extensão do modelo de Von Neumann:
  - Permitir que programas e dados armazenados em meios de armazenamento mais lentos possa ser copiados para as memória voláteis de acesso rápido como a RAM, antes da execução
  - Exemplo: Disco Rígido para Memória RAM
- Outra adaptação do modelo de Von Neumann é o modelo com **barramentos de sistema:**
  - O **barramento de dados** move dados da memória principal para os registradores da CPU e vice-versa
  - O **barramento de endereços** contém o endereço dos dados que o barramento de dados está acessando
  - O **barramento de controle** carrega os sinais que especificam como a transferência de informações deve ser feita



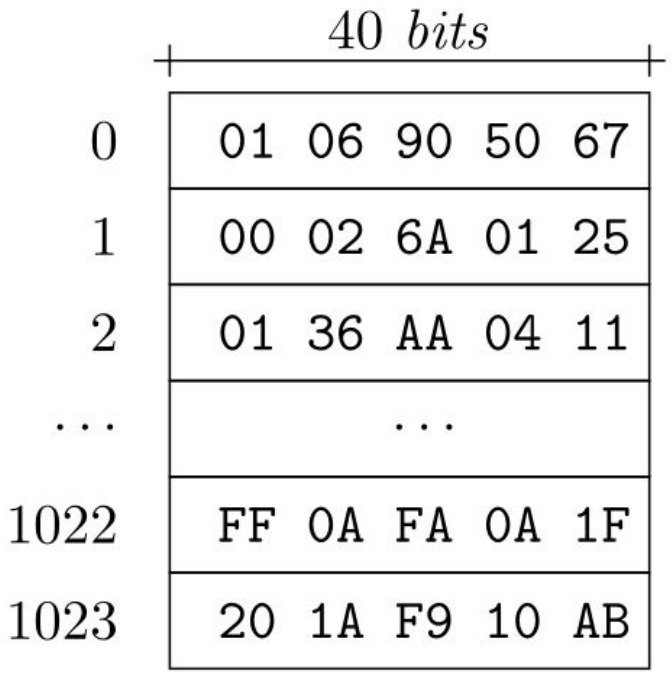
Visão de Alto Nível do Computador

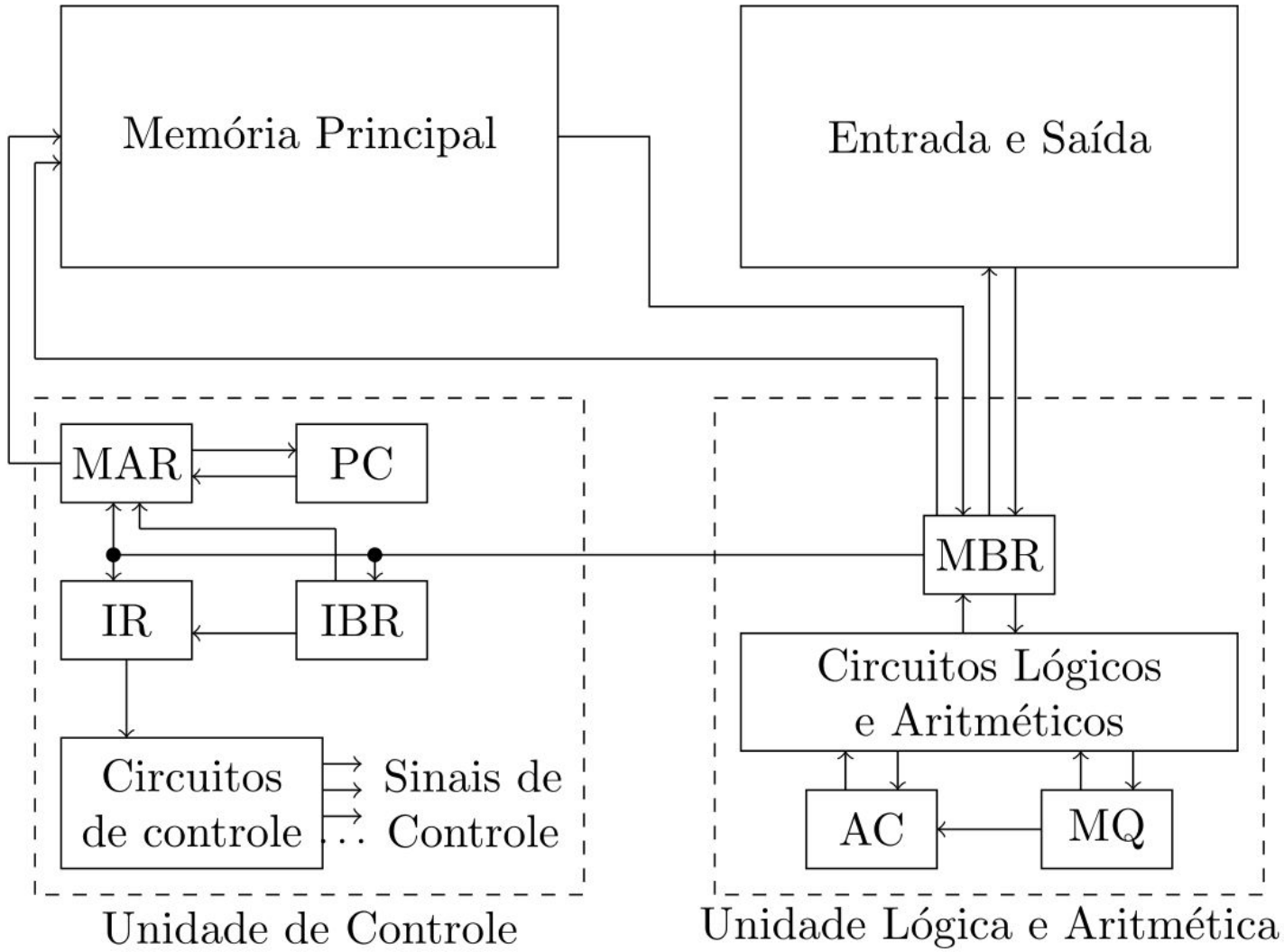
O Computador IAS

Introdução ao Computador IAS

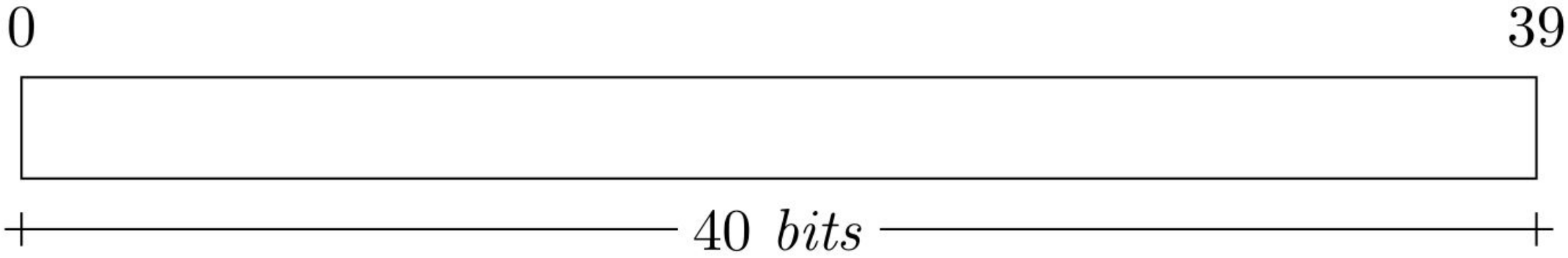
- Computador IAS (1946, 1ª geração) → "Instituto de Estudos Avançados de Princeton"
- Constituído de quatro unidades principais: Memória, ULA, UC e E/S
- Possuía memória de 1000 posições, chamadas **palavras**, de 40 bits
- Tanto os dados quanto instruções eram representados da mesma **forma binária**
  - E armazenados na **mesma memória** → Arquitetura de von Neumann!
- Possuía 21 instruções de 20 bits cada uma
- Cada instrução era composta de dois campos:
  - Um de **8 bits** denominado **Código de Operação** (ou *opcode*)
  - Outro de **12 bits** que informa o **endereço** para localizar cada uma das 1000 palavras (de 0 à 999)
    - Curiosidade: 12 bits implica 2^{12} = 4096 posições, porém somente 1000 eram usadas!

Organização do computador IAS

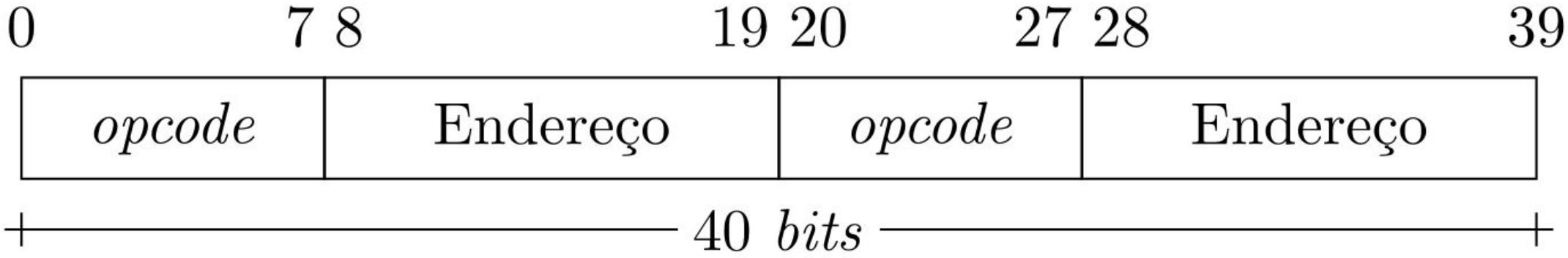




Palavras de memória do computador IAS

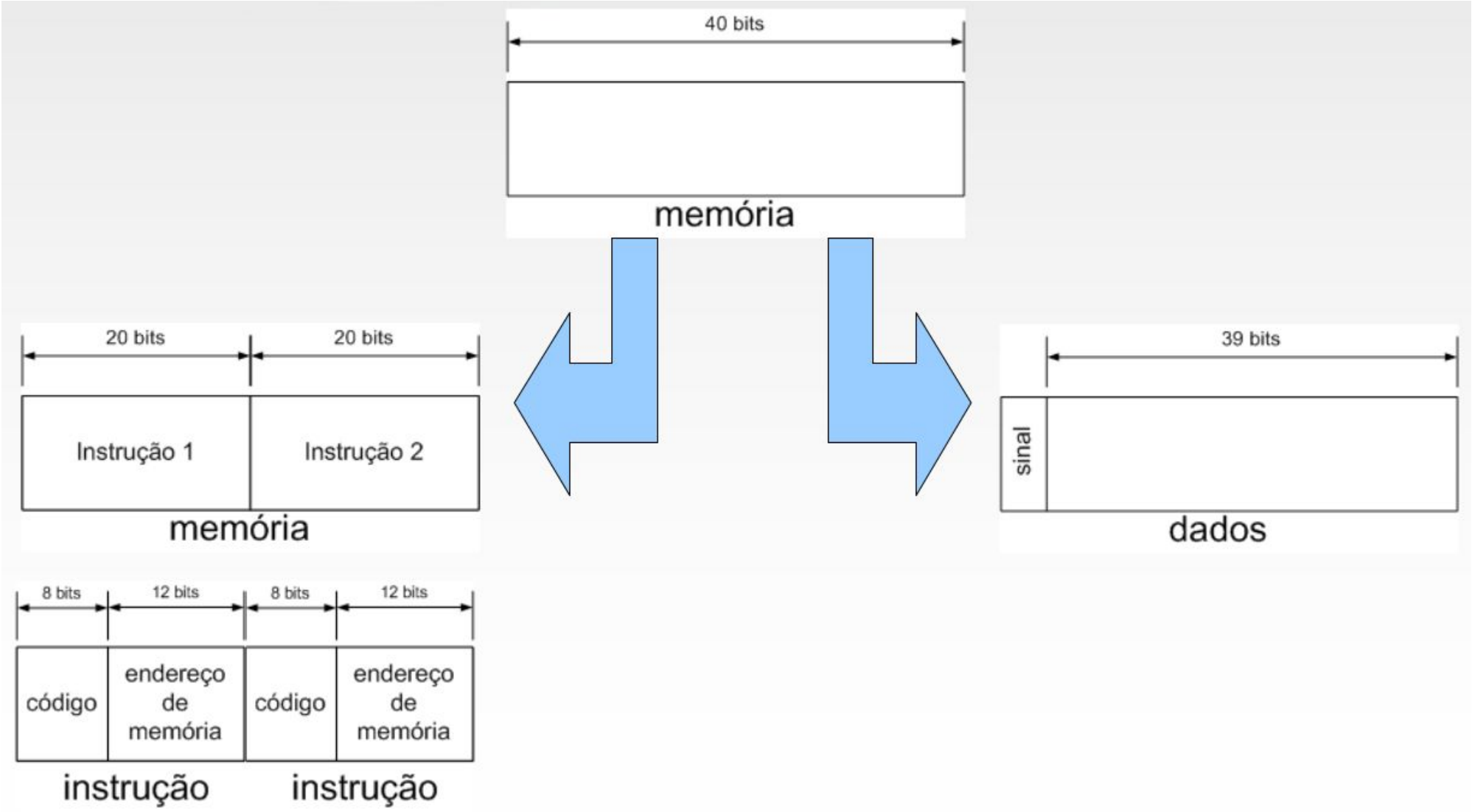


(a) Um número de 40 *bits* em uma palavra da memória



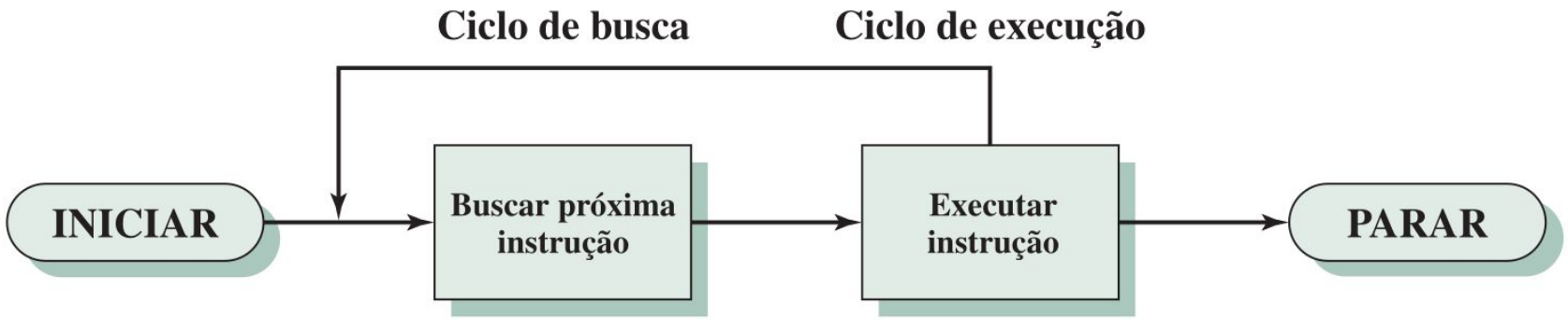
(b) Duas instruções de 20 *bits* em uma palavra da memória

Representação de dados da memória no IAS



Ciclo de execução do IAS

- Operava em modo repetitivo, um ciclo de execução e em seguida outro
- Cada ciclo de instrução consistia de 2 “subciclos”:
  - Ciclo de **busca** (*fetch*): O *opcode* da próxima instrução era trazido da memória para o IR…
    - Em detalhes: o endereço da próxima instrução (PC) era colocado em MAR para buscá-la
    - A instrução completa buscada, agora em MBR, era movida para IBR
  - …e a parte de endereço da instrução era armazenada no MAR
  - Ciclo de **execução**: O circuito de controle interpretava o código de operação e gerava os sinais apropriados para realizar o movimento de dados ou uma operação na ULA



Conjunto de instruções do IAS

- As 21 instruções do IAS eram agrupadas como segue:
- **Transferência de dados**: movem dados entre memória e registradores da ULA, ou entre dois registradores
- **Desvio incondicional**: normalmente, a unidade de controle executa instruções em sequência a partir da memória
  - Essa sequência pode ser alterada por uma instrução de desvio, que facilita operações repetitivas.
- **Desvio condicional**: o desvio pode se tornar dependente de uma condição, permitindo assim pontos de decisão
- **Aritméticas**: operações realizadas pela ULA
- **Modificação de endereço**: permite que os endereços sejam calculados na ULA e depois inseridos em instruções armazenadas na memória

Transferência de dados

- Movem dados entre memória e registradores da ULA, ou entre dois registradores

Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator

Desvios

- Normalmente, a unidade de controle executa instruções em sequência a partir da memória
- Essa sequência pode ser alterada por uma instrução de **desvio incondicional**, que facilita operações repetitivas

Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)

- O desvio pode se tornar dependente de uma condição (**desvio condicional**), permitindo assim pontos de decisão

Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP+ M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)

Operações aritméticas

- Operações elementares realizadas pela ULA

Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2, i.e., shift left one bit position
	00010101	RSH	Divide accumulator by 2, i.e., shift right one position

Modificação de endereço

- Permite que os endereços sejam calculados na ULA e depois inseridos em instruções armazenadas na memória
- Útil para modificar instruções que acessam a memória para acessar novos endereços

Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC

Atividade de Programação

- Escreva um programa para o IAS (use o simulador) que consiga somar todos os elementos de um vetor, para isso:
  - Defina o vetor na memória e seu tamanho
  - Dica: Use o conceito de repetição e de verificação de condição

Visão de Alto Nível do Computador

Conclusão

Resumo da Aula

- **Arquitetura de von Neumann**: Computador executa programa armazenado na memória
  - Instruções e dados ocupam a mesma memória
- **Gargalo**: Como acessar instruções e dados ao mesmo tempo?
- Possível solução → **Arquitetura de Harvard**
  - Instruções e dados ocupam memórias diferentes
- **Computador IAS** → Computador de 40 bits
  - Instruções (20 bits) divididas em opcode (8 bits) e endereços (12 bits)
  - Duas instruções cabem em uma palavra de 40 bits (esquerda e direita)

Conclusão

- Nessa Aula:
  - Visão de Alto Nível do Computador
- Bibliografia Principal:
  - Arquitetura e Organização de Computadores; Stallings, W.; 10ª Edição (Capítulo 3)
- Próxima Aula:
  - Estruturas de Interconexão e Barramentos