


Criando uma máquina virtual



3 de setembro de 2018

Aula por Alexandre Martins



Objetivo

Estudar conceitos de organização de computadores visando criar uma máquina virtual simples.



Puzzly at a computer
Wikimedia Commons / Guillom

Roteiro

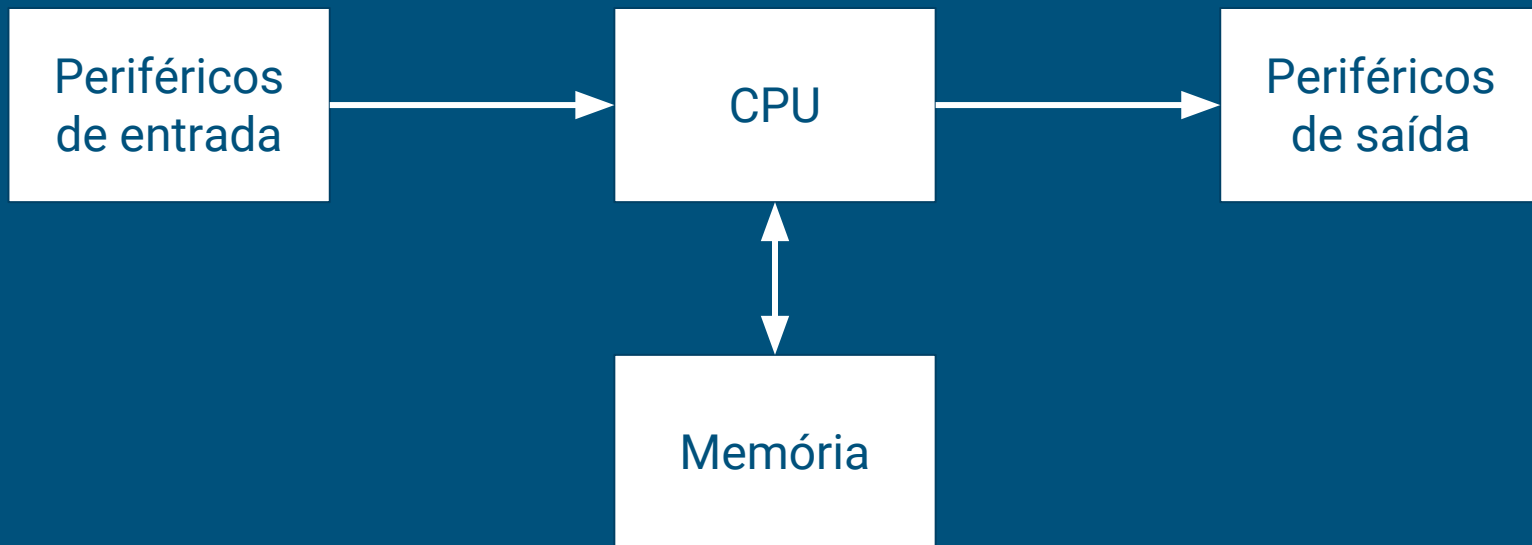
- Conceitos básicos
- Funcionamento da CPU
- Criação de uma máquina virtual simples
- Construção de programas



Conceitos

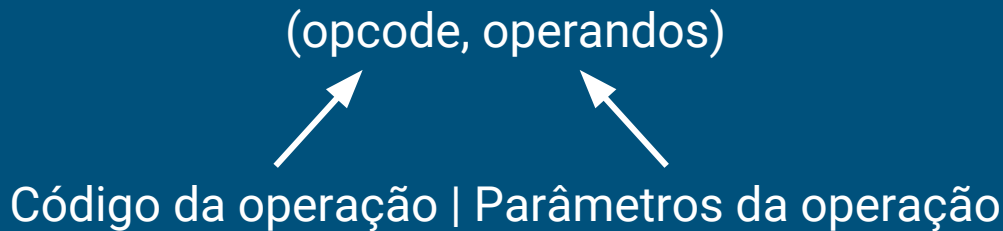
Funcionamento básico de um computador

São 4 funções básicas: entrada, processamento, storage/recovery e saída.



O processamento ocorre via instruções

Cada CPU tem um conjunto de **instruções** que visa processar os dados.



Ex: soma, subtração, salve, carregue, jump, etc.

Programas de computador

Aqui, entenderemos que um **programa** é uma sequência de instruções:
(possivelmente acompanhada de um conjunto de dados)

```
mov  eax, 4  
mov  ebx, 1
```

```
mov  ecx, str  
mov  edx, len  
int  80h
```

```
mov  eax, 1  
mov  ebx, 0  
int  80h
```

Programa em linguagem Assembly

Representação de dados

Para representar dados em computadores eletrônicos, utiliza-se notação binária:

- **Bit** (*binary digit*): 0 ou 1
 - Representa 2 valores
- **Byte**: sequência de 8 bits
 - Representa $2^8 = 256$ valores
 - Ex: $(10)_2 = 1 \cdot 2^1 + 0 \cdot 2^0 = 2$

“Só existem 10 tipos de pessoas no mundo: as que entendem binário e as que não.”
- autor desconhecido

CPUs trabalham com palavras

A **palavra** é a unidade de processamento de dados na CPU (medida em bits).

- Representa números inteiros
- Codifica endereços de memória
- Define arquitetura da CPU

Unidades: word, dword, qword...

Exemplos

- Mega-drive
 - Processador MC 68000
 - Console de 16 bits
- Nintendo 64
 - Processador NEC VR 4300
 - Trabalhava com até 64-bits
- Hoje
 - Processadores de 64 bits



Fonte: Wikipedia

Registradores

Parte importante das CPUs são os **registradores**.

- Armazenam valores temporários na CPU
- Têm o tamanho de uma palavra
- Utilizados para a realização de instruções, cálculos, acesso à memória, etc.
- Existem em número reduzido na CPU
 - Dados não utilizados no momento ficam na memória.

Memória

A **memória** é um conjunto de células:

- Cada célula tem o mesmo tamanho (ex: byte)
- Cada célula tem um endereço (um número)
- Armazena dados, programas...

addr	data
FF02h	01100110
FF03h	01100111
FF04h	01100011
FF05h	00000000

Organização da memória

Programas ficam na memória.

Principais modelos: von Neumann, Harvard.



von Neumann
mesma memória



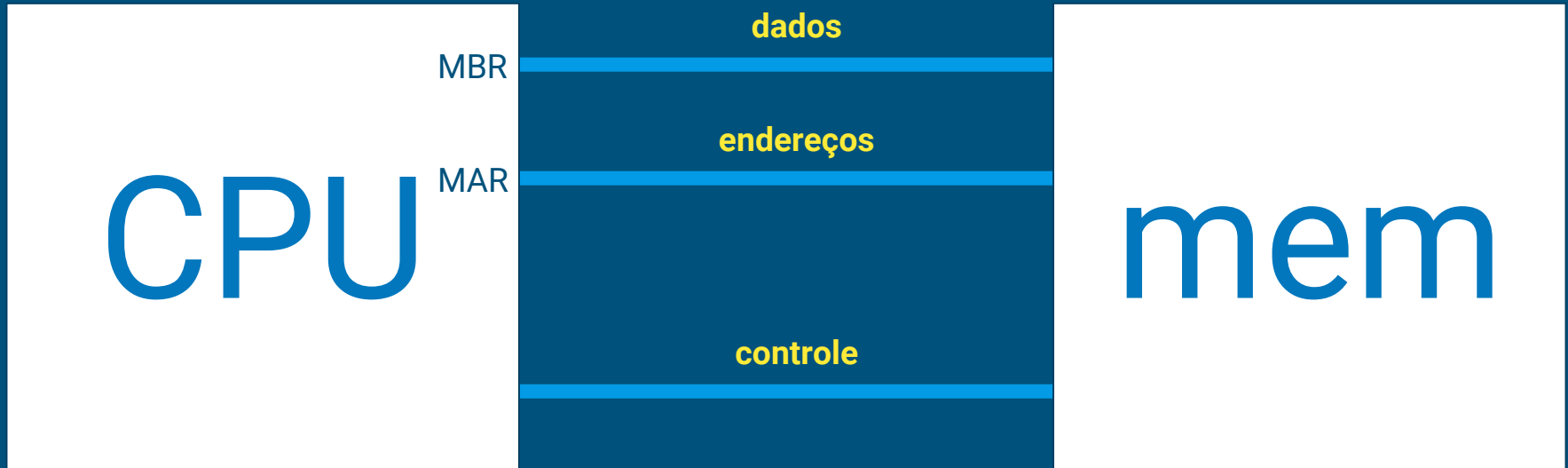
Harvard
memórias separadas

Interface CPU-memória

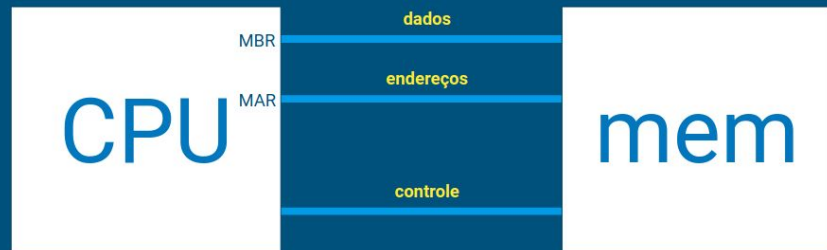
A comunicação entre CPU e memória se dá pelo **barramento** local.

- Um barramento (*bus*) é um canal por onde trafegam os bits
- A CPU não opera diretamente com os dados da memória
 - As operações são feitas nos registradores
 - A CPU lê e armazena dados na memória
 - Os programas ficam na memória
- Barramento local: dados | endereços | controle

Interface CPU-memória



Interface CPU-memória



- MAR (*memory address register*): guarda o endereço da memória a ser lido/escrito.
- MBR (*memory buffer register*): guarda os dados que foram lidos/serão escritos na memória (aka *MDR*).
- Barramento de dados: liga o MBR à memória.
- Barramento de endereços: liga o MAR à memória.
- Barramento de controle: sinais READ, WRITE, sincronização de operações.

CPU



O que é a CPU?

A CPU (Unidade Central de Processamento) é peça chave no computador:

- Realiza processamento dos dados
- Faz acessos à memória (leitura/escrita)
- Desempenha atividades de controle com os demais componentes do computador
- Roda os programas

Roda os programas?

Executar sequência de instruções (*opcode*, *operandos*) na memória.
Há diferentes modelos de máquinas. Ex:

- 0 operandos
 - Máquinas de pilha
- 1 operando
 - Máquinas de acumuladores (1 registrador é implícito)
- 2 operandos...

O **instruction set** determina a qtd. de operandos p/ cada opcode.

Roda os programas?

Para executar as instruções, são usados registradores:

- Registradores de propósito geral
- PC (*program counter*): armazena o endereço da próxima instrução, dispostas de forma sequencial na memória (aka *IP*)
- IR (*instruction register*): registra a instrução corrente
- MAR, MBR: registradores de acesso à memória

Roda os programas?

O **clock** se refere à frequência com que um dispositivo (como a CPU) funciona.

- Um dispositivo gerador de pulsos gera sinais elétricos em intervalos regulares de tempo (milhões, bilhões de vezes por segundo)
- Esses sinais são usados para sincronizar circuitos (sinal de “JÁ!”)
- É uma medida de desempenho da CPU (MHz, GHz...)
- Outros componentes podem trabalhar com frequências diferentes
 - deve-se sincronizar

Componentes da CPU

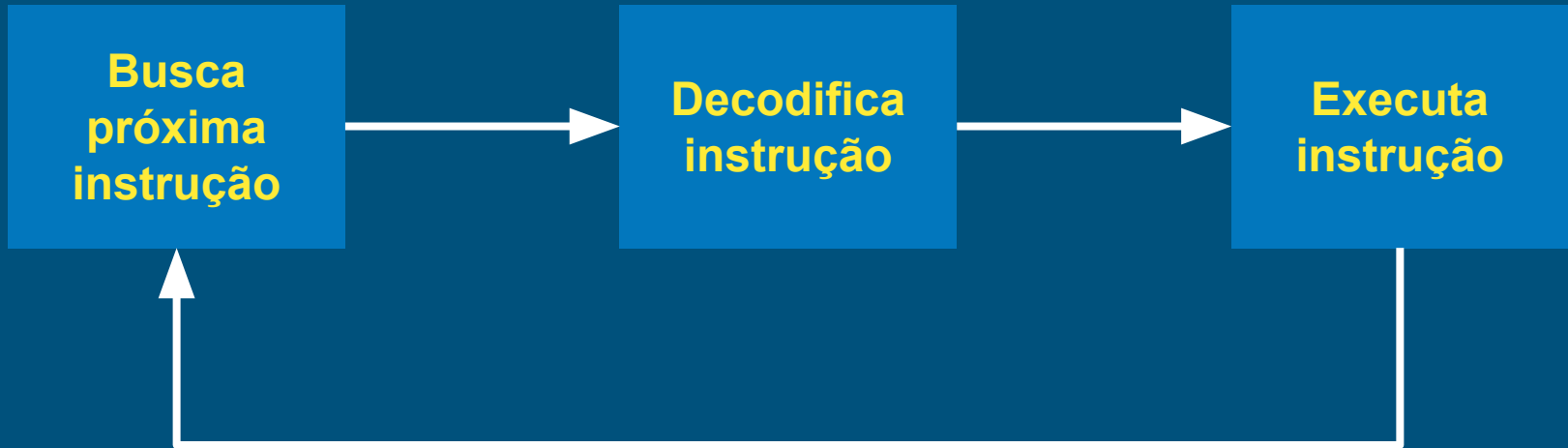
São dois os principais componentes:

- **ULA** (Unidade Lógico-Aritmética): realiza operações matemáticas
 - Exemplos: + - * / & | ^ ~ >> << ! == != <= >= < > ...
 - As operações são realizadas sob os registradores
- **UC** (Unidade de Controle): controla ULA, acesso à memória e E/S
 - Possui circuitos para controlar/sincronizar os elementos
 - Responsável por executar o *ciclo de instrução*

Ciclo de instrução



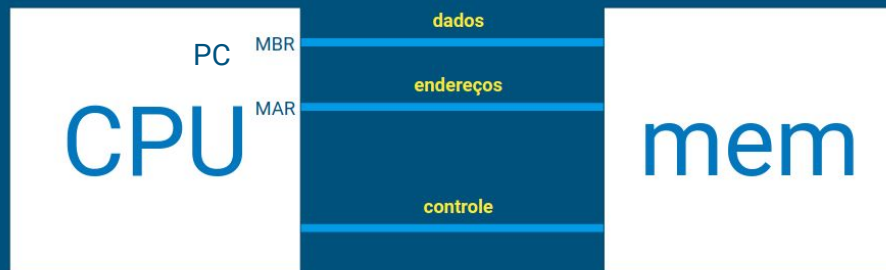
É como o processador executa as instruções dos programas:



Ciclo de instrução

Busca próxima instrução:

1. MAR recebe o endereço guardado no PC
2. CPU pede para ler dado da memória
3. Memória devolve dado no barramento local
4. Instrução é lida no MBR
5. IR recebe o valor do MBR
6. PC é incrementado para o endereço da próxima instrução



Barramento local

Ciclo de instrução

Decodifica instrução:

1. Determina qual é a operação (opcode)

Executa instrução:

1. Lê os operandos da instrução (se houver)
2. Busca dados da memória (se for o caso)
3. Executa a instrução (já com todos os dados em mãos)
4. Armazena resultado na memória (se for o caso)

Parte prática