# Como um computador funciona?
## Princípios de arquiteturas computacionais

**MSc. Joao Rossi**

IDP

joaorossiborba@gmail.com

github.com/joaorossi15

# Revolução digital

- Terceira revolução humana, fechando a tríade histórica
- Progresso extremamente rápido pautado pela **Lei de Moore**
- Aplicações infinitas, desde itens do dia a dia, como *smartphones*, chegando a aplicações mais sensíveis, como controles de usinas nucleares
- Mas como essas máquinas "mágicas" funcionam?

# O que iremos aprender?

- Como programas são traduzidos para linguagem de máquina
- Como um computador executa programas traduzidos
- Interface hardware/software
- Detalhes físicos de um computador

# As oito ideias

- Uso de abstrações para simplificação do design

# As oito ideias

- Uso de abstrações para simplificação do design
- Otimização para o caso comum

# As oito ideias

- Uso de abstrações para simplificação do design
- Otimização para o caso comum
- Performance via paralelismo

# As oito ideias

- Uso de abstrações para simplificação do design
- Otimização para o caso comum
- Performance via paralelismo
- Performance via *pipelining*

# As oito ideias

- Uso de abstrações para simplificação do design
- Otimização para o caso comum
- Performance via paralelismo
- Performance via *pipelining*
- Performance por predições

# As oito ideias

- Uso de abstrações para simplificação do design
- Otimização para o caso comum
- Performance via paralelismo
- Performance via *pipelining*
- Performance por predições
- Hierarquia de memória

# As oito ideias

- Uso de abstrações para simplificação do design
- Otimização para o caso comum
- Performance via paralelismo
- Performance via *pipelining*
- Performance por predições
- Hierarquia de memória
- Confiabilidade por redundância

# O que existe por trás de um programa?

# O que existe por trás de um programa?

- Softwares de Aplicações:
  - Programas escritos em linguagens de alto-nível

# O que existe por trás de um programa?

- Softwares de Aplicações:
  - Programas escritos em linguagens de alto-nível
- Softwares de Sistema:
  - Programas que funcionam como um meio termo entre o hardware e as aplicações

# O que existe por trás de um programa?

- Softwares de Aplicações:
  - Programas escritos em linguagens de alto-nível
- Softwares de Sistema:
  - Programas que funcionam como um meio termo entre o hardware e as aplicações
  - Exemplo: SOs e compiladores

# O que existe por trás de um programa?

- Softwares de Aplicações:
  - Programas escritos em linguagens de alto-nível
- Softwares de Sistema:
  - Programas que funcionam como um meio termo entre o hardware e as aplicações
  - Exemplo: SOs e compiladores
- Hardware
  - Processador, memória e I/O

# Bases Numéricas

Como representar informações em um computador?

# Bases Numéricas

Como representar informações em um computador?

- Informações são representadas através de dígitos binários, ou bits (*BInary digiTs*)
- Dígitos 0 e 1
- Pergunta: Quantos estados diferentes podemos representar com 3 dígitos da base binária?

# Bases Numéricas

Existem infinitas bases diferentes e a quantidade de símbolos (ou valores) distintos que que cada dígito pode ter define a base numérica.

- Base 2 (binária) $\rightarrow [0, 1]$

# Bases Numéricas

Existem infinitas bases diferentes e a quantidade de símbolos (ou valores) distintos que que cada dígito pode ter define a base numérica.

- Base 2 (binária) $\rightarrow [0, 1]$
- Base 8 (octal) $\rightarrow [0, 7]$

# Bases Numéricas

Existem infinitas bases diferentes e a quantidade de símbolos (ou valores) distintos que que cada dígito pode ter define a base numérica.

- Base 2 (binária) $\rightarrow [0, 1]$
- Base 8 (octal) $\rightarrow [0, 7]$
- Base 10 (decimal) $\rightarrow [0, 9]$

# Bases Numéricas

Existem infinitas bases diferentes e a quantidade de símbolos (ou valores) distintos que que cada dígito pode ter define a base numérica.

- Base 2 (binária) $\rightarrow$ [0, 1]
- Base 8 (octal) $\rightarrow$ [0, 7]
- Base 10 (decimal) $\rightarrow$ [0, 9]
- Base 16 (hexadecimal) $\rightarrow$ 0 - ?

# Bases Numéricas

Qual é a base dos números abaixo?

- AC10
- 727
- 1001

# Bases Numéricas

Qual é a base dos números abaixo?

- AC10
- 727
- 1001

Para distinguir, utilizamos a notação: $AC10_{16}$, $727_8$ e $1001_2$

# Bases Numéricas

Qual é a base dos números abaixo?

- AC10
- 727
- 1001

Para distinguir, utilizamos a notação: $AC10_{16}$, $727_8$ e $1001_2$

Portanto, temos como representação formal: $\forall n \in b, n_b$

# Bases Numéricas

Bases especificas utilizadas na computação possuem diferentes nomeclaturas:

- $AC10_{16} \rightarrow 0xAC10$
- $727_{8} \rightarrow 0727$
- $1001_{2} \rightarrow 0b1001$

# Bases Numéricas

Bases especificas utilizadas na computação possuem diferentes nomeclaturas:

- $AC10_{16} \rightarrow 0xAC10$
- $727_8 \rightarrow 0727$
- $1001_2 \rightarrow 0b1001$

Para distinguir, utilizamos a notação: $AC10_{16}$, $727_8$ e $1001_2$

Portanto, temos como representação formal: $\forall n \in b, n_b$

# Bases Numéricas

Qual é o valor de cada dígito nos números abaixo?

- $1011_2$
- $135_{10}$

# Bases Numéricas

Qual é o valor de cada dígito nos números abaixo?

- $1011_2$
- $135_{10}$

O valor é dado por: $digit \times base^{position}$.

# Bases Numéricas

Com isso, o valor decimal de um número na base **b** com **n** dígitos é o somatório dos valores dos dígitos:

$$v = \sum_{i=0}^{n-1} d_i \times b^i$$

onde $d_i$ é o dígito na posição i.

# Bases Numéricas

Com isso, o valor decimal de um número na base **b** com **n** dígitos é o somatório dos valores dos dígitos:

$v = \sum_{i=0}^{n-1} d_i \times b^i$

onde $d_i$ é o dígito na posição i.

- $1011_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 11$
- $135_{10} = 1 * 10^2 + 3 * 10^1 + 5 * 10^0 = 135$

# Bases Numéricas

Como fazemos para encontrar a representação de um valor na base binária?

Seja D um número decimal e $r_0, r_1, ..., r_n$ os módulos de divisões sucessivas por 2, então a representação binaria de D é $r_n r_{n-1} ... r_0$

# Bases Numéricas

Como fazemos para encontrar a representação de um valor na base binária?

Seja D um número decimal e $r_0, r_1, ..., r_n$ os módulos de divisões sucessivas por 2, então a representação binaria de D é $r_n r_{n-1} ... r_0$

Exemplo: $13_{10} = 1101$

| Divisão | Quociente | Resto |
|---|---|---|
| $13 \div 2$ | 6 | 1 |
| $6 \div 2$ | 3 | 0 |
| $3 \div 2$ | 1 | 1 |
| $1 \div 2$ | 0 | 1 |

# Bases Numéricas

Represente os números a seguir na base binária:

- $13_6$
- $67_{10}$

Represente os números a seguir na base hexadecimal:

- $111000_2$
- $16_{10}$
- $30_8$

# Bases Numéricas: Conversão

| **Conversão** | **Algoritmo** |
|---|---|
| Decimal $\rightarrow$ Binário | divisões sucessivas por 2 até se obter zero no quociente e leitura do resto de baixo para cima |
| Binário $\rightarrow$ Decimal | soma de potências de 2 |
| Hex $\rightarrow$ Binário | expandir dígitos hex em quatro dígitos binários segundo seu valor |
| Binário $\rightarrow$ Hex | compactar cada quatro dígitos binários em um único dígito hex |
| Decimal $\rightarrow$ Base $N$ | divisões sucessivas por $N$ até se obter zero no quociente e leitura do resto de baixo para cima |
| Base $N$ $\rightarrow$ Decimal | soma de potências de $N$ |

# Níveis de linguagens

Ok, computadores entendem bits.

Mas como podemos executar código escrito em linguagens de alto-nível?

# Níveis de linguagens

Ok, computadores entendem bits.

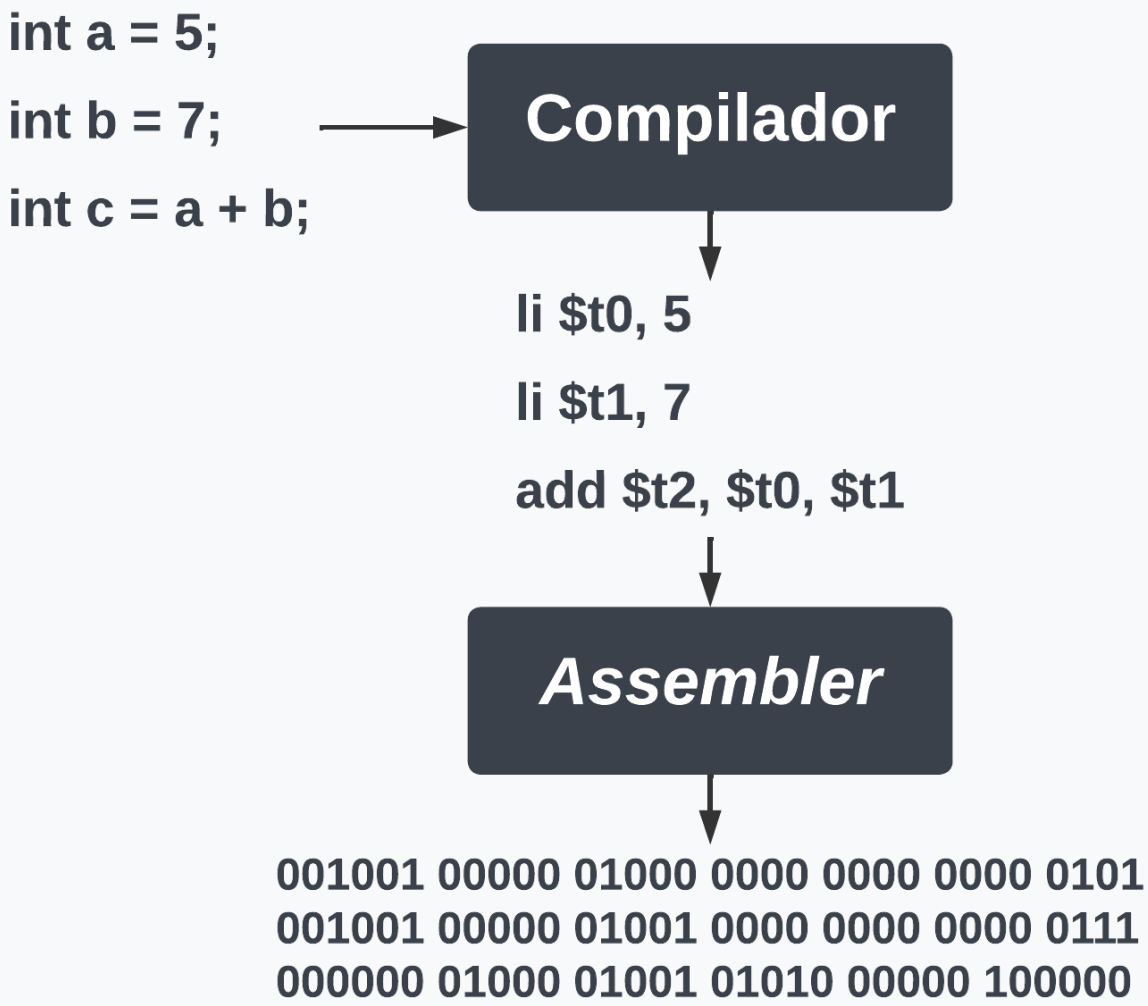
Mas como podemos executar código escrito em linguagens de alto-nível?

Pela maior descoberta da computação: como transformar linguagens complexas em números binários.

# Níveis de linguagens

- Instruções: conjuntos de bits que definem os comandos mais básicos para um computador
  - Exemplo: 1000110010100000 -> some dois números
  - **Ideia importante:** uso de números para *encoding* de instruções e dados
- *Assembler*: programa que traduz instruções escritas em linguagem textual para binário
- Compilador: programa que traduz linguagens de alto-nível para *assembly*

# Níveis de linguagens



# Componentes de um computador

Mas como computadores entendem e executam as todos esses diferentes conjuntos de bits?

# Componentes de um computador

Mas como computadores entendem e executam as todos esses diferentes conjuntos de bits?

A resposta é simples: O conjunto que compõe o hardware: CPU, memória e I/O.

# Processador

- Parte ativa de um computador
  - Ex: Sinaliza dispositivos de I/O, realiza operações, etc

# Processador

- Parte ativa de um computador
  - Ex: Sinaliza dispositivos de I/O, realiza operações, etc
- Composto de três partes principais
  - *Datapath*: realiza as operações aritméticas
  - Controle: controla memória, I/O e o *datapath* conforme as instruções de um programa
  - Cache: SRAM pequena, rápida e cara para acesso imediato

# Processador: Abstrações

- Ajudam a lidar com complexidades

# Processador: Abstrações

- Ajudam a lidar com complexidades
- Principal abstração: interface entre hardware e software de baixo-nível
  - Arquitetura: conjunto de instruções disponíveis para uma CPU especifica

# Processador: Abstrações

- Ajudam a lidar com complexidades
- Principal abstração: interface entre hardware e software de baixo-nível
  - Arquitetura: conjunto de instruções disponíveis para uma CPU especifica
  - *Application binary interface* (ABI): combinação do conjunto de instruções da CPU e a interface do sistema operacional para o desenvolvimento de aplicações

# Processador: Abstrações

- Ajudam a lidar com complexidades
- Principal abstração: interface entre hardware e software de baixo-nível
  - Arquitetura: conjunto de instruções disponíveis para uma CPU especifica
  - *Application binary interface* (ABI): combinação do conjunto de instruções da CPU e a interface do sistema operacional para o desenvolvimento de aplicações
- Implementação de uma arquitetura: hardware que obedece a abstração de arquitetura

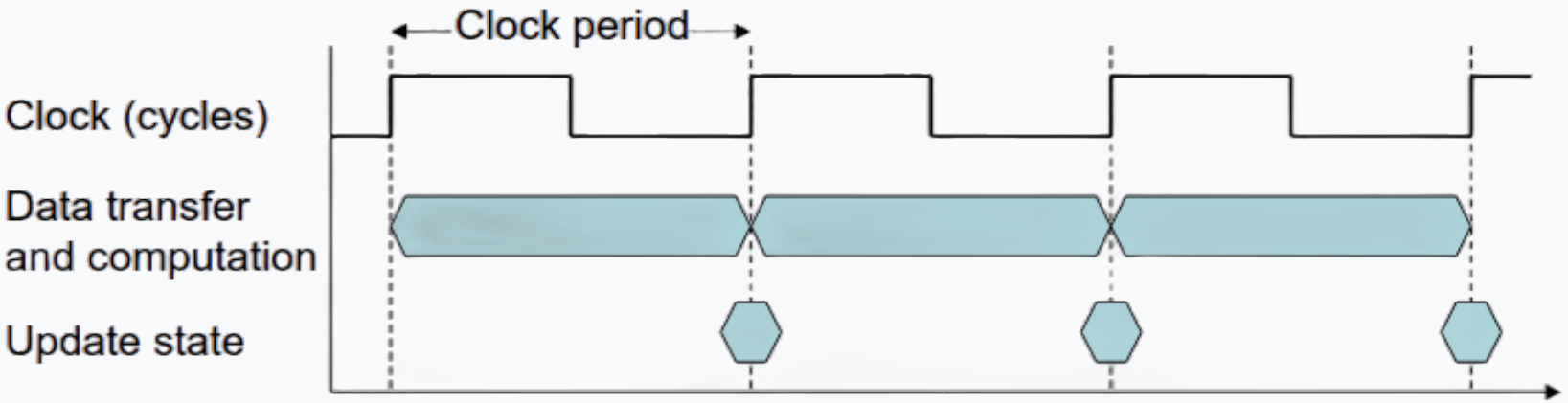
*Tanto o hardware como o software consistem em camadas hierárquicas que utilizam a abstração, sendo que cada camada inferior oculta detalhes do nível superior. Uma interface fundamental entre os níveis de abstração é a arquitetura do conjunto de instruções, a interface entre o hardware e o software de baixo nível.* ***Esta interface abstrata permite que muitas implementações de custo e desempenho variáveis executem software idêntico.***

[1]

# Processador: Tempo

- Tempo total
  - Inclui todos os aspectos: processamento, I/O, tempo de espera e *overhead* de OS
  - Determina a performance total do sistema
- Tempo de CPU
  - Tempo gasto dentro da CPU
  - Considera tempo de CPU do usuário e do sistema

# Processador: CPU *Clocking*




- Operação controlada por um relógio de taxa constante (oscilador feito de cristais de quartzo) que determina quando os eventos ocorrem no hardware

# Processador: Tempo de CPU

- Período de *Clock*: duração de um ciclo de *clock*
- Frequência de *Clock* (*rate*): ciclos por segundo (Hz)
- Tempo de CPU = Ciclos de *Clock* CPU x Tempo Ciclo de *Clock*
- Tempo de CPU = Ciclos de *Clock* CPU / *Clock Rate*
- Como um designer de hardware melhora a performance de um sistema?

# Exercício: Tempo de CPU

- Computador A: 2GHz *clock*, 10s tempo de CPU
- Design do computador B: 6s de tempo de CPU e pode aumentar velocidade de *clock*, mas isso multiplica ciclo de *clock* * 1.2
- Qual deve ser a velocidade do *clock* do computador B?

# Exercício: Tempo de CPU

$$\textit{Clock Rate}_B = \frac{\text{Ciclo de } \textit{Clock}_B}{\text{Tempo de CPU}_B} = \frac{1.2 \times \text{Ciclo de } \textit{Clock}_A}{6\text{s}}$$

$$\text{Ciclos de } \textit{Clock}_A = \text{Tempo de CPU}_A \times \textit{Clock Rate}_A$$

$$\text{Ciclos de } \textit{Clock}_A = 10s \times 2\text{GHz} = 20 \times 10^9$$

$$\textit{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

# Processador: *Instruction Count* e CPI

$$\text{Ciclos de } \textit{Clock} = \textit{Instruction Count} \times \text{Ciclos por Instrução}$$

$$\text{Tempo de CPU} = \textit{Instruction Count} \times \text{CPI} \times \text{Tempo Ciclo de } \textit{Clock}$$

$$\text{Tempo de CPU} = \frac{\textit{Instruction Count} \times \textit{CPI}}{\textit{Clock Rate}}$$

- Contagem de instruções para um programa
  - Determinado por programa, ISA e compilador
- Media de ciclos por instrucao
  - Determinado por hardware

# Exercício: CPI

- Computador A: Tempo de Ciclo = 250ps, CPI = 2
- Computador B: Tempo de Ciclo = 500ps, CPI = 1.2
- Mesma ISA
- Qual maquina é mais rápida? E qual a razão entre elas?

# Exercício: CPI

$$\text{Tempo de CPU}_A = \textit{Instruction Count} \times \text{CPI}_A \times \text{Tempo Ciclo}_A$$
$$\text{Tempo de CPU}_A = IC \times 2 \times 250\text{ps} = IC \times 500\text{ps}$$

$$\text{Tempo de CPU}_B = \textit{Instruction Count} \times \text{CPI}_B \times \text{Tempo Ciclo}_B$$
$$\text{Tempo de CPU}_B = IC \times 1.2 \times 500\text{ps} = IC \times 600\text{ps}$$

$$\frac{\text{Tempo de CPU}_B}{\text{Tempo de CPU}_A} = \frac{IC \times 600\text{ps}}{IC \times 500\text{ps}} = 1.2$$

# Processador: CPI variável

- CPI varia para diferentes tipos de instruções:
  Ciclos de *Clock* $= \sum_{i=1}^{n} \textit{Instruction Count}_i \times \text{CPI}_i$
- Média ponderada CPI:
  $\text{CPI} = \frac{\text{Ciclos de } \textit{Clock}}{\textit{Instruction Count}} = \sum_{i=1}^{n} \left( \textit{CPI}_i \times \frac{\textit{Instruction Count}_i}{\textit{Instruction Count}} \right)$

# Exercício: CPI variável

Abaixo temos duas sequências alternativas de código compilado utilizando instruções das classes A, B e C

Calcule o CPI médio de cada uma e defina qual, nesse caso, é mais rápida

| **Classe** | A | B | C |
|---|---|---|---|
| **CPI por classe** | 1 | 2 | 3 |
| **IC na sequência 1** | 2 | 1 | 2 |
| **IC na sequência 2** | 4 | 1 | 1 |

**Sequência 1: IC = 5**

- Ciclo de *Clock*: $2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
- CPI médio = $10/5 = 2$

**Sequência 2: IC = 6**

- Ciclo de *Clock*: $4 \times 1 + 1 \times 2 + 1 \times 3 = 9$
- CPI médio = $9/6 = 1.5$

# Processador: Performance Total

$$\text{Tempo de CPU} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Ciclos de } \textit{Clock}}{\text{Instruções}} \times \frac{\text{Segundos}}{\textit{Ciclo de Clock}}$$

Performance depende de:

- Algoritmos, Linguagens de programação e Compiladores: afetam IC e CPI
- Arquitetura (conjunto de instruções): afeta IC, CPI e Tempo de *Clock*

# Conclusao

No fim das contas, enganamos uma pedra pra pensar... e agora tem gente jogando LoL com ela. Vai entender.

e como disse Arthur C. Clarke:

> *Any sufficiently advanced technology is indistinguishable from magic.*

# Referências

[1] David A. Patterson and John L. Hennessy. *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. 5th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013. ISBN: 0124077269.