Disciplina de Programação Funcional
Engenharia de Computação

Tradução x Interpretação
Paradigmas de Programação

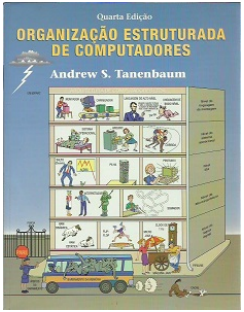
Mauro Hemerly Gazzani
mauro.hemerly@gmail.com

Universidade Estadual de Minas Gerais
Câmpus de Ituiutaba, 2º semestre de 2018
https://bit.ly/2A0eUl8
https://github.com/mauro-hemerly/UEMG-2018-2

Tradução x Interpretação

Execução de Programas de Computador

- **Linguagem de Máquina**
- Máquinas Multiníveis[1]
- Processos de execução: tradução e interpretação
- Máquina Virtual

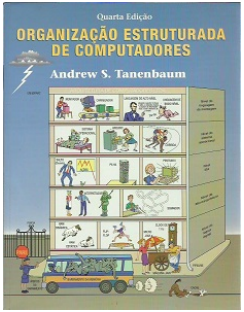


[1] Fonte: TANENBAUM,Andrew S. Organização estruturada de computadores. 4. ed. São Paulo: Pearson Prentice Hall.

Tradução x Interpretação

Execução de Programas de Computador

- **Linguagem de Máquina**
- **Máquinas Multiníveis**[1]
- **Processos de execução: tradução e interpretação**
- **Máquina Virtual**

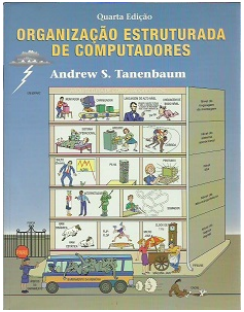


[1] Fonte: TANENBAUM,Andrew S. Organização estruturada de computadores. 4. ed. São Paulo: Pearson Prentice Hall.

Tradução x Interpretação

Execução de Programas de Computador

- **Linguagem de Máquina**
- **Máquinas Multiníveis**[1]
- **Processos de execução: tradução e interpretação**
- **Máquina Virtual**

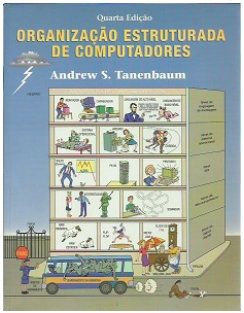


[1] Fonte: TANENBAUM,Andrew S. Organização estruturada de computadores. 4. ed. São Paulo: Pearson Prentice Hall.

Tradução x Interpretação

Execução de Programas de Computador

- **Linguagem de Máquina**
- **Máquinas Multiníveis**[1]
- **Processos de execução: tradução e interpretação**
- **Máquina Virtual**

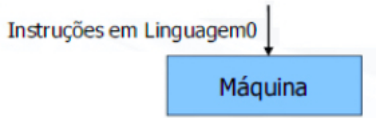


[1] Fonte: TANENBAUM,Andrew S. Organização estruturada de computadores. 4. ed. São Paulo: Pearson Prentice Hall.

Tradução x Interpretação

Linguagem de Máquina

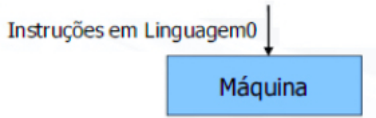
- **Computador Digital**: É uma máquina capaz de solucionar problemas através da execução de instruções que lhe são fornecidas.
- **Programa**: É uma sequência de instruções que descrevem como executar uma determinada tarefa.
- **Instruções de Máquina**: Instruções que o computador é capaz de reconhecer e executar, para o qual todos os programas devem ser convertidos antes que eles possam ser executados. (**Linguagem de Máquina: L0**)



Tradução x Interpretação

Linguagem de Máquina

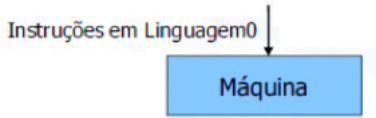
- **Computador Digital**: É uma máquina capaz de solucionar problemas através da execução de instruções que lhe são fornecidas.
- **Programa**: É uma sequência de instruções que descrevem como executar uma determinada tarefa.
- **Instruções de Máquina**: Instruções que o computador é capaz de reconhecer e executar, para o qual todos os programas devem ser convertidos antes que eles possam ser executados. (**Linguagem de Máquina: L0**)



Tradução x Interpretação

Linguagem de Máquina

- **Computador Digital**: É uma máquina capaz de solucionar problemas através da execução de instruções que lhe são fornecidas.
- **Programa**: É uma sequência de instruções que descrevem como executar uma determinada tarefa.
- **Instruções de Máquina**: Instruções que o computador é capaz de reconhecer e executar, para o qual todos os programas devem ser convertidos antes que eles possam ser executados. (**Linguagem de Máquina: L0**)



Processador ARM[2]: Linguagem de Máquina e Assembly

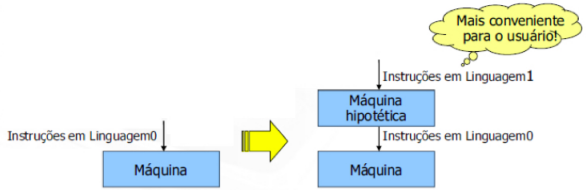
```
11100101100111110001000000010000   E59F1010   LDR   R1, num1
11100101100111110001000000001000   E59f0008   LDR   R0, num2
11100000100000010101000000000000   E0815000   ADD   R5, R1, R0
11100110100011110101000000001000   E68F5008   STR   R5, num3
```

[1] Fonte: https://www.arm.com/products/processors

Tradução x Interpretação

Máquinas Multiníveis

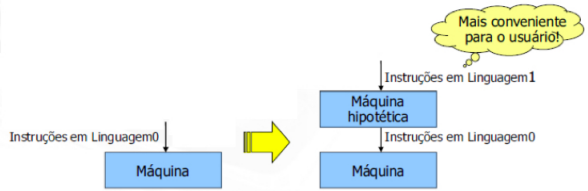
- Problema:
  - Existe uma grande lacuna entre uma **linguagem de programação** conveniente para uso humano e a **linguagem de máquina** entendida pelos circuitos eletrônicos dos computadores.
- Como resolvê-lo?
  - Deve-se projetar um novo conjunto de instruções [ ] que seja mais conveniente para as pessoas usarem do que o conjunto de instruções que já vem embutido na máquina [ ].



Tradução x Interpretação

Máquinas Multiníveis

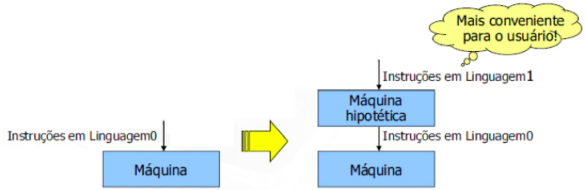
- Problema:
  - Existe uma grande lacuna entre uma **linguagem de programação** conveniente para uso humano e a **linguagem de máquina** entendida pelos circuitos eletrônicos dos computadores.
- Como resolvê-lo?



Tradução x Interpretação

Máquinas Multiníveis

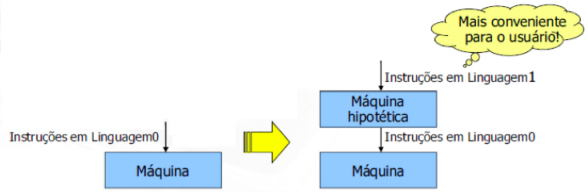
- Problema:
  - Existe uma grande lacuna entre uma **linguagem de programação** conveniente para uso humano e a **linguagem de máquina** entendida pelos circuitos eletrônicos dos computadores.
- Como resolvê-lo?
  - Deve-se projetar um novo conjunto de instruções (**Linguagem L1**) que seja mais conveniente para as pessoas usarem do que o conjunto de instruções que já vem embutido na máquina (**Linguagem L0**).



Tradução x Interpretação

Máquinas Multiníveis

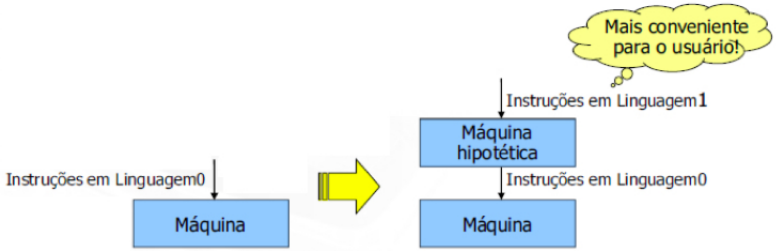
- Problema:
  - Existe uma grande lacuna entre uma **linguagem de programação** conveniente para uso humano e a **linguagem de máquina** entendida pelos circuitos eletrônicos dos computadores.
- Como resolvê-lo?
  - Deve-se projetar um novo conjunto de instruções (**Linguagem L1**) que seja mais conveniente para as pessoas usarem do que o conjunto de instruções que já vem embutido na máquina (**Linguagem L0**).



Tradução x Interpretação

Máquinas Multiníveis

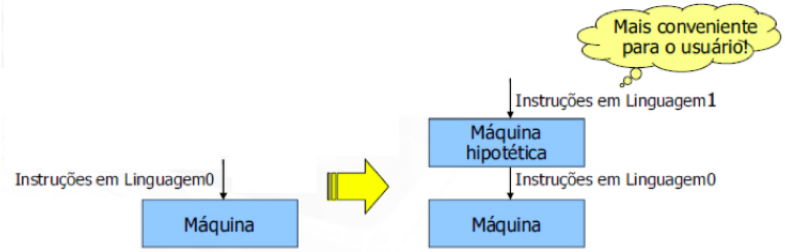
- **Questão fundamental**: Como programas escritos em **Linguagem L1** são executados pelo computador que, afinal, só pode executar programas escritos em **Linguagem de Máquina L0** ?
- **Resposta**: **Tradução** ou **interpretação**.



Tradução x Interpretação

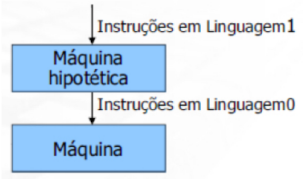
Máquinas Multiníveis

- **Questão fundamental**: Como programas escritos em **Linguagem L1** são executados pelo computador que, afinal, só pode executar programas escritos em **Linguagem de Máquina L0** ?
- **Resposta**: **Tradução** ou **interpretação**.



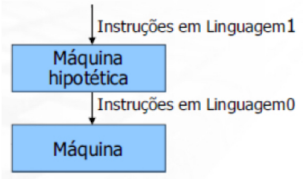
Tradução

- Cada instrução do **programa escrito em L1** é substituída por uma sequência equivalente de instruções em **L0**;
- Ao final, o programa escrito na **Linguagem L1** estará convertido por completo para a **Linguagem de Máquina L0**;
- O computador executa o novo programa em **L0** em vez do antigo programa escrito em **L1**, que é **descartado**;
- Exemplos:
  - C, Pascal, Haskell, etc.



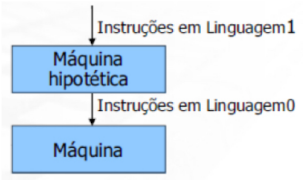
Tradução

- Cada instrução do **programa escrito em L1** é substituída por uma sequência equivalente de instruções em **L0**;
- Ao final, o programa escrito na **Linguagem L1** estará convertido por completo para a **Linguagem de Máquina L0**;
- O computador executa o novo programa em **L0** em vez do antigo programa escrito em **L1**, que é **descartado**;
- Exemplos:
  - C, Pascal, Haskell, etc.



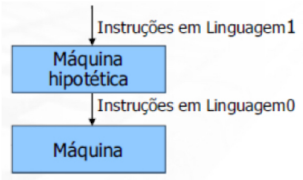
Tradução

- Cada instrução do **programa escrito em L1** é substituída por uma sequência equivalente de instruções em **L0**;
- Ao final, o programa escrito na **Linguagem L1** estará convertido por completo para a **Linguagem de Máquina L0**;
- O computador executa o novo programa em **L0** em vez do antigo programa escrito em **L1**, que é **descartado**;
- Exemplos:
  - C, Pascal, Haskell, etc.



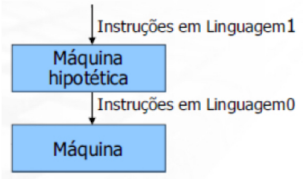
Tradução

- Cada instrução do **programa escrito em L1** é substituída por uma sequência equivalente de instruções em **L0**;
- Ao final, o programa escrito na **Linguagem L1** estará convertido por completo para a **Linguagem de Máquina L0**;
- O computador executa o novo programa em **L0** em vez do antigo programa escrito em **L1**, que é **descartado**;
- Exemplos:
  - C, Pascal, Haskell, etc.



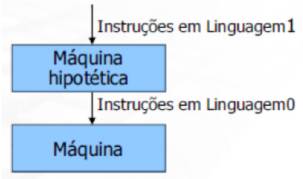
Tradução

- Cada instrução do **programa escrito em L1** é substituída por uma sequência equivalente de instruções em **L0**;
- Ao final, o programa escrito na **Linguagem L1** estará convertido por completo para a **Linguagem de Máquina L0**;
- O computador executa o novo programa em **L0** em vez do antigo programa escrito em **L1**, que é **descartado**;
- Exemplos:
  - C, Pascal, Haskell, etc.



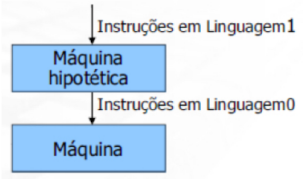
Interpretação

- Cada instrução individual do programa na **Linguagem L1** é traduzida para a **Linguagem L0** e é executada imediatamente;
- Exemplos:
  - Python, Lisp, Haskell, etc.
- O interpretador (programa escrito em **Linguagem L0**), considera os programas escritos em **Linguagem L1** como os dados de entrada;
- O interpretador examina cada instrução por vez, traduzindo e executando diretamente a sequência de instruções correspondentes na **Linguagem L0**.



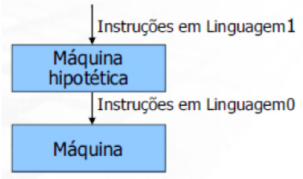
Interpretação

- Cada instrução individual do programa na **Linguagem L1** é traduzida para a **Linguagem L0** e é executada imediatamente;
- Exemplos:
  - Python, Lisp, Haskell, etc.
- O interpretador (programa escrito em **Linguagem L0**), considera os programas escritos em **Linguagem L1** como os dados de entrada;
- O interpretador examina cada instrução por vez, traduzindo e executando diretamente a sequência de instruções correspondentes na **Linguagem L0**.



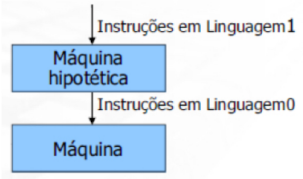
Interpretação

- Cada instrução individual do programa na **Linguagem L1** é traduzida para a **Linguagem L0** e é executada imediatamente;
- Exemplos:
  - Python, Lisp, Haskell, etc.
- O interpretador (programa escrito em **Linguagem L0**), considera os programas escritos em **Linguagem L1** como os dados de entrada;
- O interpretador examina cada instrução por vez, traduzindo e executando diretamente a sequência de instruções correspondentes na **Linguagem L0**.



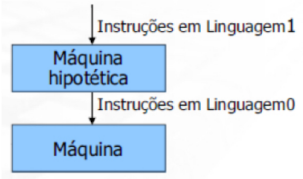
Interpretação

- Cada instrução individual do programa na **Linguagem L1** é traduzida para a **Linguagem L0** e é executada imediatamente;
- Exemplos:
  - Python, Lisp, Haskell, etc.
- O interpretador (programa escrito em **Linguagem L0**), considera os programas escritos em **Linguagem L1** como os dados de entrada;
- O interpretador examina cada instrução por vez, traduzindo e executando diretamente a sequência de instruções correspondentes na **Linguagem L0**.



Interpretação

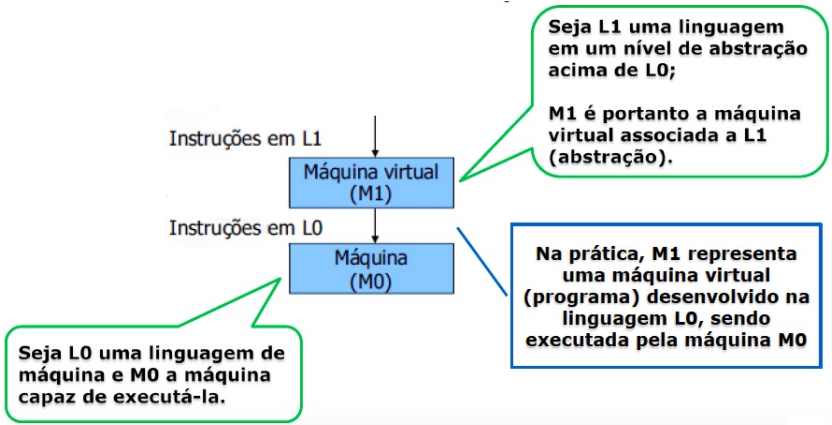
- Cada instrução individual do programa na **Linguagem L1** é traduzida para a **Linguagem L0** e é executada imediatamente;
- Exemplos:
  - Python, Lisp, Haskell, etc.
- O interpretador (programa escrito em **Linguagem L0**), considera os programas escritos em **Linguagem L1** como os dados de entrada;
- O interpretador examina cada instrução por vez, traduzindo e executando diretamente a sequência de instruções correspondentes na **Linguagem L0**.



Tradução x Interpretação

Máquina Virtual

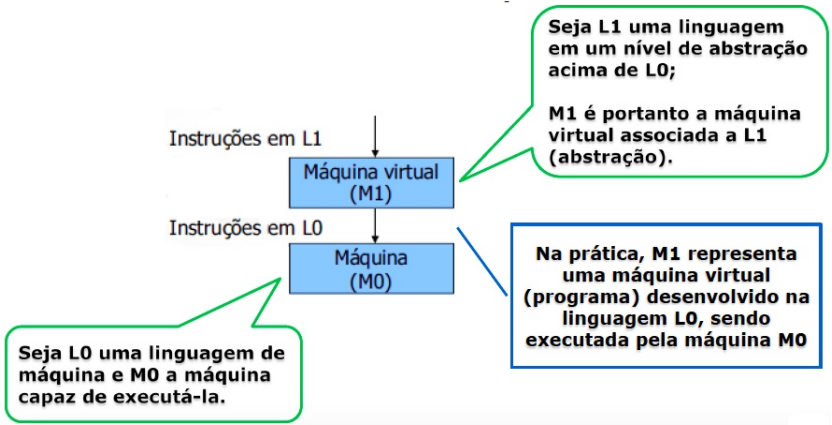
- Representa uma abstração capaz de reconhecer e executar diretamente as instruções de uma linguagem específica.



Tradução x Interpretação

Máquina Virtual

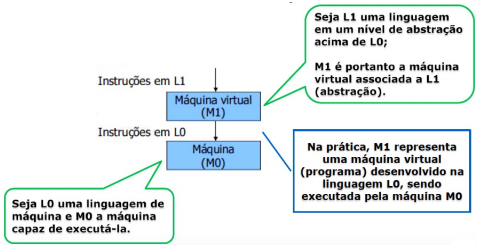
- Representa uma abstração capaz de reconhecer e executar diretamente as instruções de uma linguagem específica.



Tradução x Interpretação

Máquina Virtual

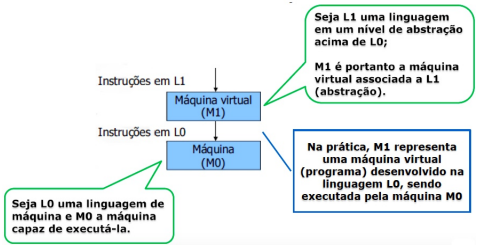
- Se fosse barato construir uma **Máquina M1** com **Linguagem de Máquina L1** não haveria a necessidade de se ter a **Linguagem L0** ou uma máquina que executasse programas em **L0**;
- As pessoas poderiam simplesmente escrever programas em **L1** e fazer com que o computador os executasse diretamente;
- Seria possível escrever programas para as máquinas **virtuais** como se elas existissem na realidade.



Tradução x Interpretação

Máquina Virtual

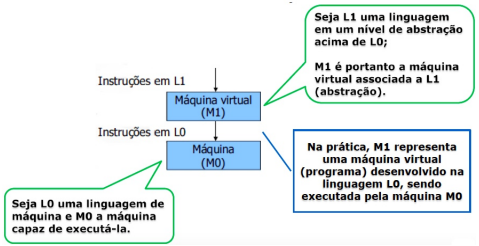
- Se fosse barato construir uma **Máquina M1** com **Linguagem de Máquina L1** não haveria a necessidade de se ter a **Linguagem L0** ou uma máquina que executasse programas em **L0**;
- As pessoas poderiam simplesmente escrever programas em **L1** e fazer com que o computador os executasse diretamente;
- Seria possível escrever programas para as máquinas **virtuais** como se elas existissem na realidade.



Tradução x Interpretação

Máquina Virtual

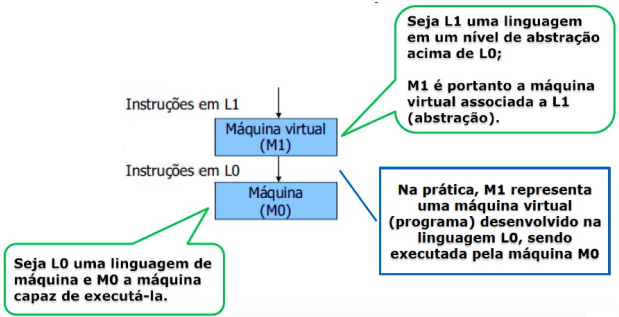
- Se fosse barato construir uma **Máquina M1** com **Linguagem de Máquina L1** não haveria a necessidade de se ter a **Linguagem L0** ou uma máquina que executasse programas em **L0**;
- As pessoas poderiam simplesmente escrever programas em **L1** e fazer com que o computador os executasse diretamente;
- Seria possível escrever programas para as máquinas **virtuais** como se elas existissem na realidade.



Tradução x Interpretação

Máquina Virtual

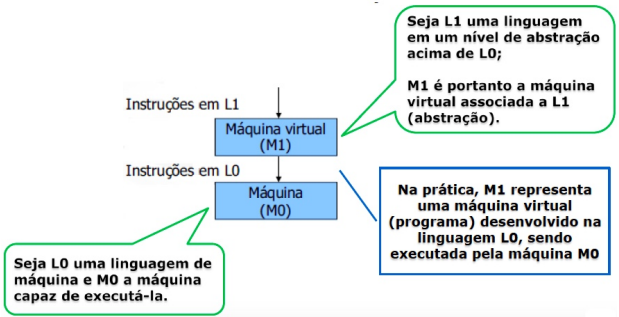
- Naturalmente, quanto mais alto o **nível de abstração de L1** mais próxima ela é da compreensão humana;
- Porém, para que a tradução ou a interpretação sejam tarefas práticas e o custo da **Máquina M1** razoável, **L0** e **L1** não devem ser linguagens com níveis de abstração muito **diferentes**;



Tradução x Interpretação

Máquina Virtual

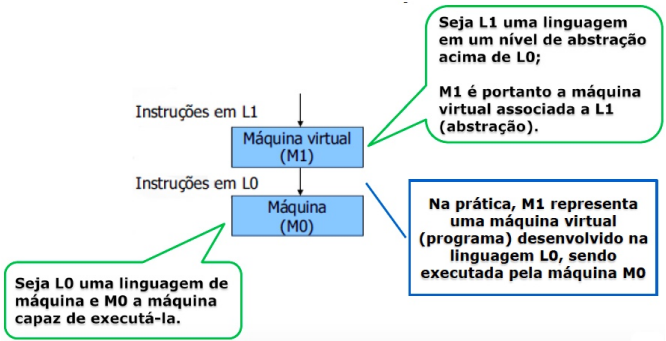
- Naturalmente, quanto mais alto o **nível de abstração de L1** mais próxima ela é da compreensão humana;
- Porém, para que a tradução ou a interpretação sejam tarefas práticas e o custo da **Máquina M1** razoável, **L0** e **L1** não devem ser linguagens com níveis de abstração muito **diferentes**;



Tradução x Interpretação

Máquina Virtual

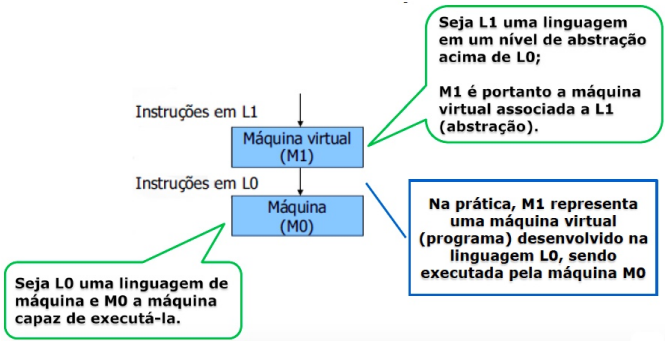
- Então, **L1**, embora mais amigável do que a **Linguagem de Máquina L0**, ainda está longe de ser ideal para a maioria das aplicações.
- Como então solucionar este problema?



Tradução x Interpretação

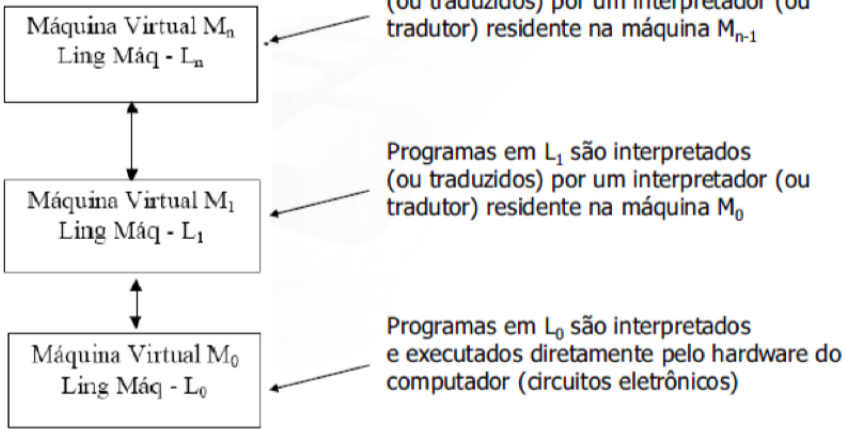
Máquina Virtual

- Então, **L1**, embora mais amigável do que a **Linguagem de Máquina L0**, ainda está longe de ser ideal para a maioria das aplicações.
- Como então solucionar este problema?



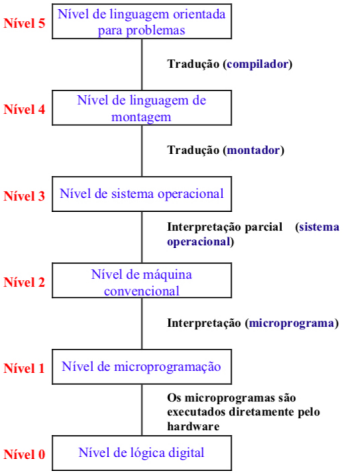
Tradução x Interpretação

Máquinas Multiníveis – Vários Níveis



Tradução x Interpretação

Máquinas Multinível Contemporâneas



Paradigmas e Linguagens de Programação

- Um paradigma de programação **fornece e determina a visão** que o programador possui sobre a **estruturação e a execução do programa**.
- Por exemplo:
  - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma coleção de objetos que interagem entre si.
  - Em **programação lógica** os programadores abstraem o programa como um conjunto de predicados que estabelecem relações entre objetos (axiomas), e uma meta (teorema) a ser provada usando os predicados.

Paradigmas e Linguagens de Programação

- Um paradigma de programação **fornece e determina a visão** que o programador possui sobre a **estruturação e a execução do programa**.
- Por exemplo:
  - Em **programação orientada a objetos**, programadores podem abstrair um programa como uma coleção de objetos que interagem entre si.
  - Em **programação lógica** os programadores abstraem o programa como um conjunto de predicados que estabelecem relações entre objetos (axiomas), e uma meta (teorema) a ser provada usando os predicados.

Paradigmas e Linguagens de Programação

- Diferentes **linguagens de programação** propõem diferentes **paradigmas de programação**.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
  - Smalltalk, Eiffel e Java suportam o paradigma orientado a objetos.
  - Haskell e Clean suportam o paradigma funcional.
  - OCaml, LISP, Scala, Perl, Python e C++ suportam múltiplos paradigmas.

Paradigmas e Linguagens de Programação

- Diferentes **linguagens de programação** propõem diferentes **paradigmas de programação**.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
  - Smalltalk, Eiffel e Java suportam o paradigma orientado a objetos.
  - Haskell e Clean suportam o paradigma funcional.
  - OCaml, LISP, Scala, Perl, Python e C++ suportam múltiplos paradigmas.

Paradigmas e Linguagens de Programação

- Diferentes **linguagens de programação** propõem diferentes **paradigmas de programação**.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
  - Smalltalk, Eiffel e Java suportam o paradigma orientado a objetos.
  - Haskell e Clean suportam o paradigma funcional.
  - OCaml, LISP, Scala, Perl, Python e C++ suportam múltiplos paradigmas.

Paradigmas e Linguagens de Programação

- Diferentes **linguagens de programação** propõem diferentes **paradigmas de programação**.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
  - Smalltalk, Eiffel e Java suportam o paradigma orientado a objetos.
  - Haskell e Clean suportam o paradigma funcional.
  - OCaml, LISP, Scala, Perl, Python e C++ suportam múltiplos paradigmas.

Paradigmas e Linguagens de Programação

- Diferentes **linguagens de programação** propõem diferentes **paradigmas de programação**.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
  - Smalltalk, Eiffel e Java suportam o paradigma orientado a objetos.
  - Haskell e Clean suportam o paradigma funcional.
  - OCaml, LISP, Scala, Perl, Python e C++ suportam múltiplos paradigmas.

Paradigmas e Linguagens de Programação

- Diferentes **linguagens de programação** propõem diferentes **paradigmas de programação**.
- Algumas linguagens foram desenvolvidas para suportar um paradigma específico.
- Por exemplo:
  - Smalltalk, Eiffel e Java suportam o paradigma orientado a objetos.
  - Haskell e Clean suportam o paradigma funcional.
  - OCaml, LISP, Scala, Perl, Python e C++ suportam múltiplos paradigmas.

Programação Imperativa

- Descreve a computação como ações, enunciados ou comandos que mudam o **estado** (**variáveis**) de um programa, enfatizando como resolver um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar.
- O nome do paradigma, imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
  - Imperativo (procedimental): C, Pascal, etc, e
  - orientado a objetos: Smalltalk, Java, etc.

Programação Imperativa

- Descreve a computação como ações, enunciados ou comandos que mudam o **estado** (**variáveis**) de um programa, enfatizando como resolver um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar.
- O nome do paradigma, imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
  - Imperativo (procedimental): C, Pascal, etc, e
  - orientado a objetos: Smalltalk, Java, etc.

Programação Imperativa

- Descreve a computação como ações, enunciados ou comandos que mudam o **estado** (**variáveis**) de um programa, enfatizando como resolver um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar.
- O nome do paradigma, imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
  - Imperativo (procedimental): C, Pascal, etc, e
  - orientado a objetos: Smalltalk, Java, etc.

Programação Imperativa

- Descreve a computação como ações, enunciados ou comandos que mudam o **estado** (**variáveis**) de um programa, enfatizando como resolver um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar.
- O nome do paradigma, imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
  - **imperativo** (procedimental): C, Pascal, etc, e
  - **orientado a objetos**: Smalltalk, Java, etc

Programação Imperativa

- Descreve a computação como ações, enunciados ou comandos que mudam o **estado** (**variáveis**) de um programa, enfatizando como resolver um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar.
- O nome do paradigma, imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
  - **imperativo** (procedimental): C, Pascal, etc, e
  - **orientado a objetos**: Smalltalk, Java, etc

Programação Imperativa

- Descreve a computação como ações, enunciados ou comandos que mudam o **estado** (**variáveis**) de um programa, enfatizando como resolver um problema.
- Muito parecidos com o comportamento imperativo das linguagens naturais que expressam ordens, programas imperativos são uma sequência de comandos para o computador executar.
- O nome do paradigma, imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...
- Exemplos: paradigmas
  - **imperativo** (procedimental): C, Pascal, etc, e
  - **orientado a objetos**: Smalltalk, Java, etc

Programação Declarativa

- Descreve **o que o programa faz** e **não como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
  - funcional: Haskell, OCaml, LISP, etc...
  - lógico: Prolog, etc...

Programação Declarativa

- Descreve **o que o programa faz** e **não como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
  - funcional: Haskell, OCaml, LISP, etc...
  - lógico: Prolog, etc...

Programação Declarativa

- Descreve **o que o programa faz** e **não como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
  - **funcional**: Haskell, OCaml, LISP, etc, e
  - **lógico**: Prolog, etc.

Programação Declarativa

- Descreve **o que o programa faz** e **não como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
  - **funcional**: Haskell, OCaml, LISP, etc, e
  - **lógico**: Prolog, etc.

Programação Declarativa

- Descreve **o que o programa faz** e **não como** seus procedimentos funcionam.
- Ênfase nos **resultados**, no que se deseja obter.
- Exemplos: paradigmas
  - **funcional**: Haskell, OCaml, LISP, etc, e
  - **lógico**: Prolog, etc.

Programação Funcional

- Programação funcional é um paradigma de programação que descreve uma computação como uma **expressão** a ser avaliada.
- A principal forma de estruturar o programa é pela definição e aplicação de **funções**.

Programação Funcional

- Programação funcional é um paradigma de programação que descreve uma computação como uma **expressão** a ser avaliada.
- A principal forma de estruturar o programa é pela definição e aplicação de **funções**.

Exemplo: quick sort - C vs Haskell

```c
// To sort array a[] of size n: qsort(a,0,n-1)

void qsort(int a[], int lo, int hi) {
  int h, l, p, t;

  if (lo < hi) {
    l = lo;
    h = hi;
    p = a[hi];

    do {
      while ((l < h) && (a[l] <= p))
          l = l+1;
      while ((h > l) && (a[h] >= p))
          h = h-1;
      if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
      }
    } while (l < h);

    a[hi] = a[l];
    a[l] = p;

    qsort(a, lo, l-1);
    qsort(a, l+1, hi);
  }
}
```

```haskell
qs []     = []
qs (x:xs) = qs (filter (<x) xs) ++
            [x] ++
            qs (filter (>x) xs)
```