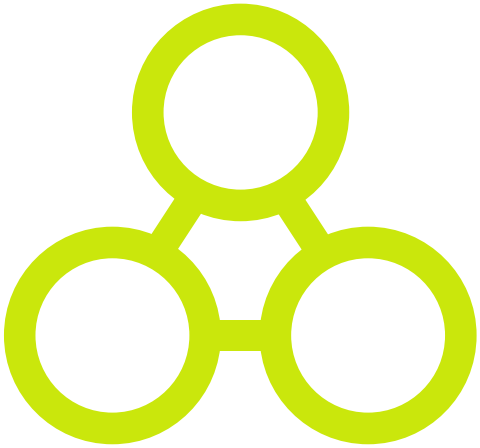


Introdução à Computação

Jordana Sarmenghi Salamon
jssalamon@inf.ufes.br
jordanasalamon@gmail.com
http://inf.ufes.br/~jssalamon

Departamento de Informática
Universidade Federal do Espírito Santo

Agenda

- Introdução a Arquitetura de Computadores
  - Conceitos Básicos
  - Tradução vs Interpretação
  - Máquinas Multi-Níveis
  - Arquitetura de Computadores
  - Hardware, Software e Firmware
  - Execução de um Programa

Conceitos

- **Computador**
  - Máquina capaz de **resolver problemas** de diferentes naturezas e variados graus de complexidade, **executando uma série de instruções simples** reconhecidas por seus circuitos internos.
  - Os circuitos [eletrônicos] internos de um computador podem reconhecer e executar diretamente apenas um **conjunto finito** de instruções primitivas.
  - As instruções de um computador são feitas simples e precisas, visando **reduzir a complexidade e o custo dos circuitos internos** da máquina.

Conceitos

- **Computador Digital**
  - É uma máquina que pode resolver problemas executando uma série de instruções que lhe são fornecidas, ou seja, uma **máquina programável**.
- **Programa**
  - Um conjunto finito de **instruções** que descrevem a maneira de se realizar uma determinada tarefa.

Conceitos

- **Arquitetura de Computadores**
  - A arquitetura é frequentemente definida como o conjunto de atributos da máquina que um programador deve compreender para que consiga programar o computador específico com sucesso, ou seja, para que consiga compreender o que o programa irá fazer quando da sua execução. Por exemplo, parte da arquitetura são as instruções e os operadores manipulados por elas.

Conceitos

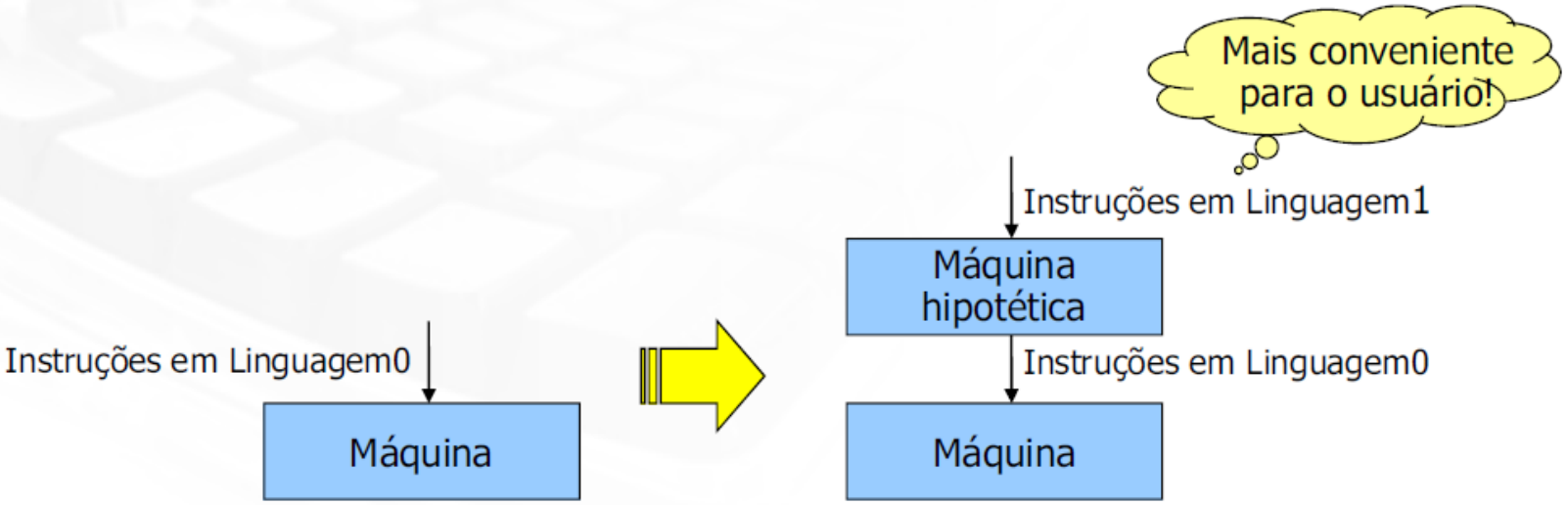
- **Linguagem de Máquina**
  - É o conjunto finito de instruções que os circuitos eletrônicos de um determinado computador pode reconhecer e executar diretamente (é a linguagem de programação que a máquina “realmente” entende).
- **Linguagem de Programação**
  - É a maneira pela qual as instruções dadas ao computador são expressas. Ex: Haskell, Phyton, C, Java, Fortran, Prolog, Assembly, Lisp, etc.

Conceitos

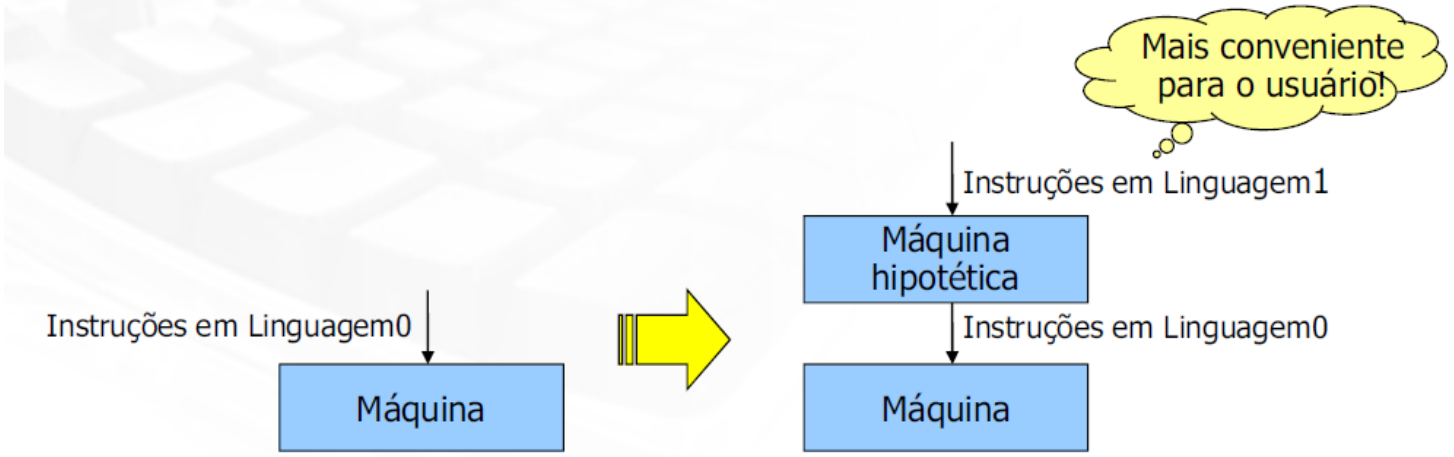
- **Problema:**
  - Existe uma grande lacuna entre uma linguagem de programação conveniente para uso humano e a linguagem de máquina entendida pelos circuitos eletrônicos dos computadores.
- **Como resolvê-lo?**
  - Deve-se projetar um novo conjunto de instruções (linguagem L1) que seja mais conveniente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

Conceitos

- **Solução: Adicionar um nível a mais na máquina!**



Conceitos



- **Questão fundamental:**
  - Como programas escritos em linguagem L1 são executados pelo computador que, afinal, só pode executar programas escritos em linguagem de máquina L0?

Tradução vs 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

Tradução vs Interpretação

- **Interpretação**
- Cada instrução individual do programa em L1 é traduzida para a linguagem L0 e é executada imediatamente.
- Ex: Python, Lisp, Haskell
- 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 vs Interpretação

- **Linguagens de alto nível**
- Facilitam a comunicação dos seres humanos com a máquina, sendo linguagens mais próximas dos humanos.
- Devem ser convertidas em linguagem de máquina para serem executadas.
- **Ex:** C, C++, Delphi, Pascal, Fortran, JAVA,...
- Java é um exemplo de linguagem que utiliza métodos híbridos de interpretação e tradução

Tradução vs Interpretação

- **Exemplo de tradução**

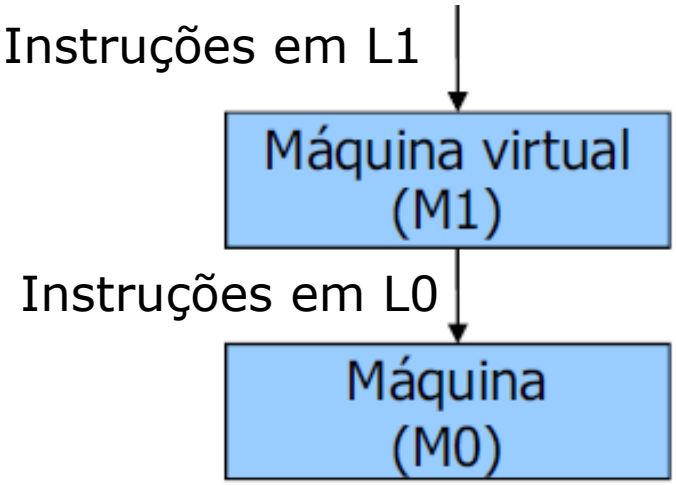
```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw      $t0, 0($2)
lw      $t1, 4($2)
sw      $t1, 0($2)
sw      $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Máquinas Multi-Níveis

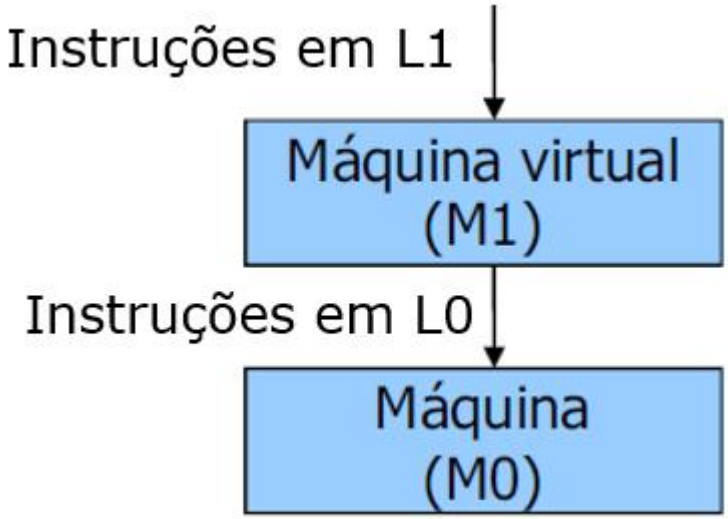
- **Máquina Virtual**
- Representa uma **abstração** capaz de reconhecer e executar diretamente as instruções de uma linguagem específica



- Na prática, M1 representa uma máquina virtual (programa) desenvolvida na linguagem L0, sendo executada pela máquina M0

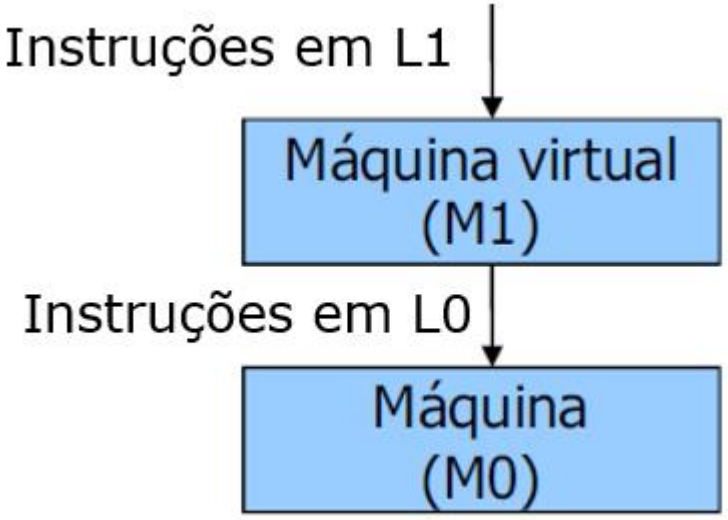
Máquinas Multi-Níveis

- **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.



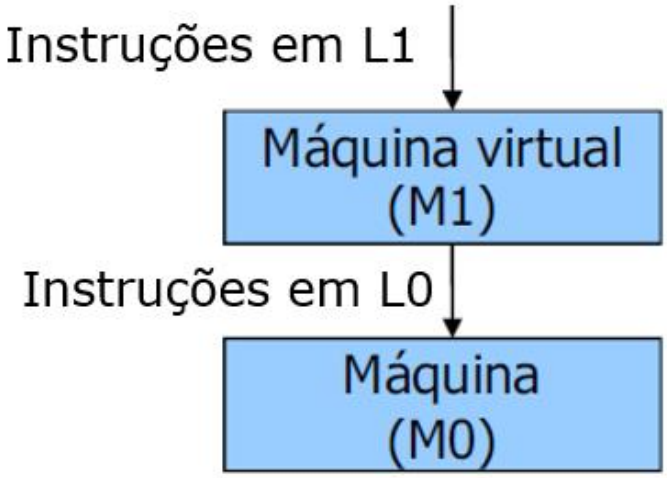
Máquinas Multi-Níveis

- **Máquina Virtual**
- 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.



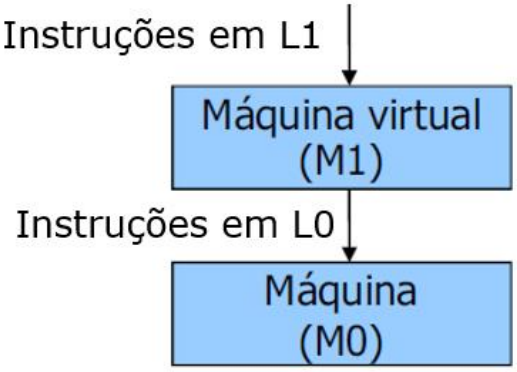
Máquinas Multi-Níveis

- **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”.



Máquinas Multi-Níveis

- **Máquina Virtual**
- Esse fato é desanimador à luz do propósito original de L1 – livrar o programador da carga de ter de expressar algoritmos em uma linguagem mais adequada às máquinas do que às pessoas.
- Assim, L1, embora mais amigável do que a linguagem de máquina L0, ainda está longe de ser ideal para a maioria das aplicações.
- Solução: máquinas multiníveis.

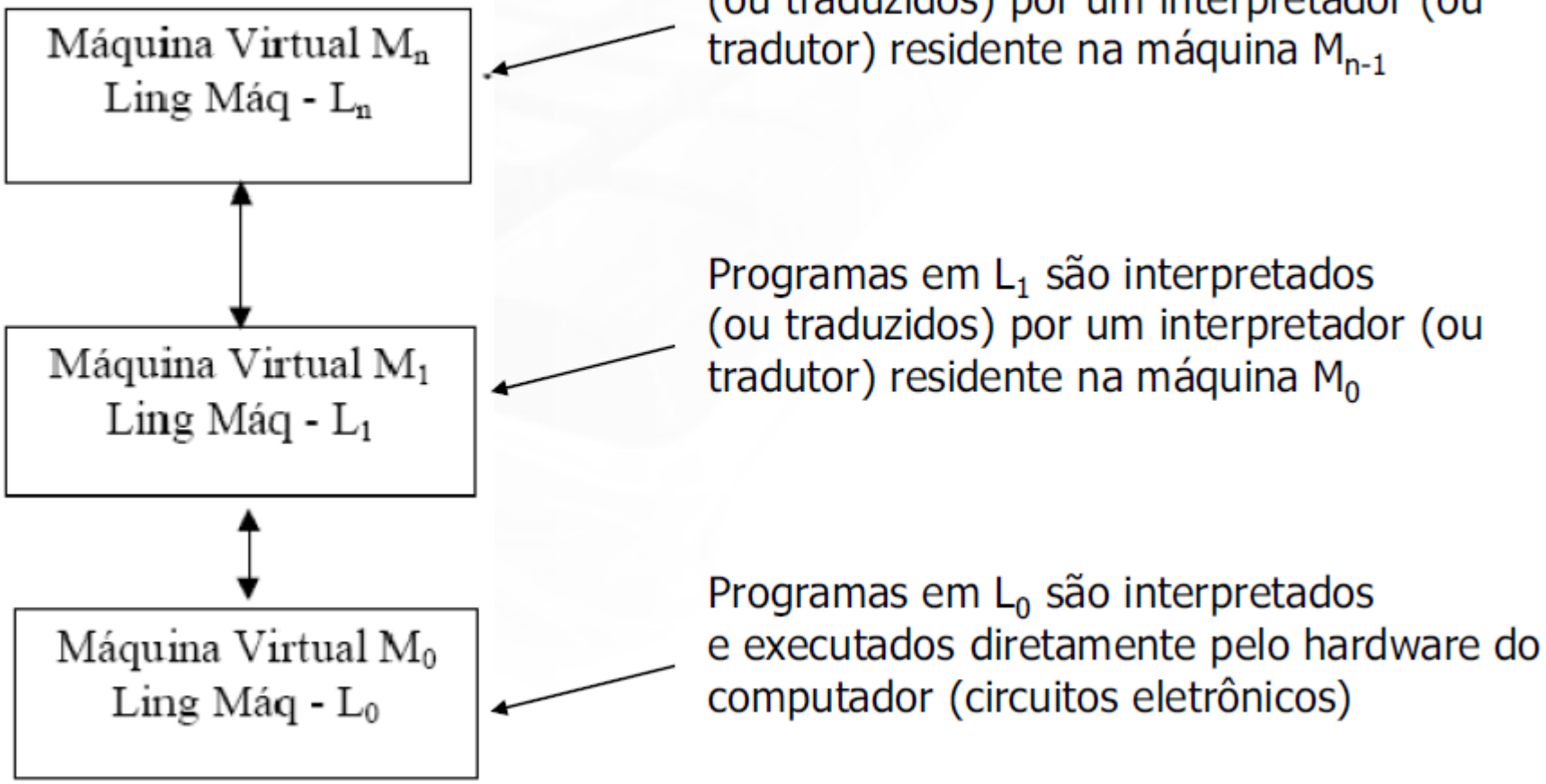


Máquinas Multi-Níveis

- Possuem múltiplas camadas ou níveis de abstração.
- Nível ou Camada: uma máquina (computador real ou virtual) e a correspondente linguagem, sobre a qual uma nova camada pode ser acrescentada.
- A linguagem ou o nível mais baixo é o mais simples, enquanto a linguagem ou o nível mais alto é o mais sofisticado.

Máquinas Multi-Níveis

- Máquinas Multi-Níveis gerais



Máquinas Multi-Níveis

- Cada máquina virtual tem associada a si uma linguagem, composta de todas as instruções que essa máquina pode executar.

> Uma máquina define uma linguagem.
> Uma linguagem define uma máquina.
> (a saber, a máquina que pode executar todos os programas escritos na linguagem.

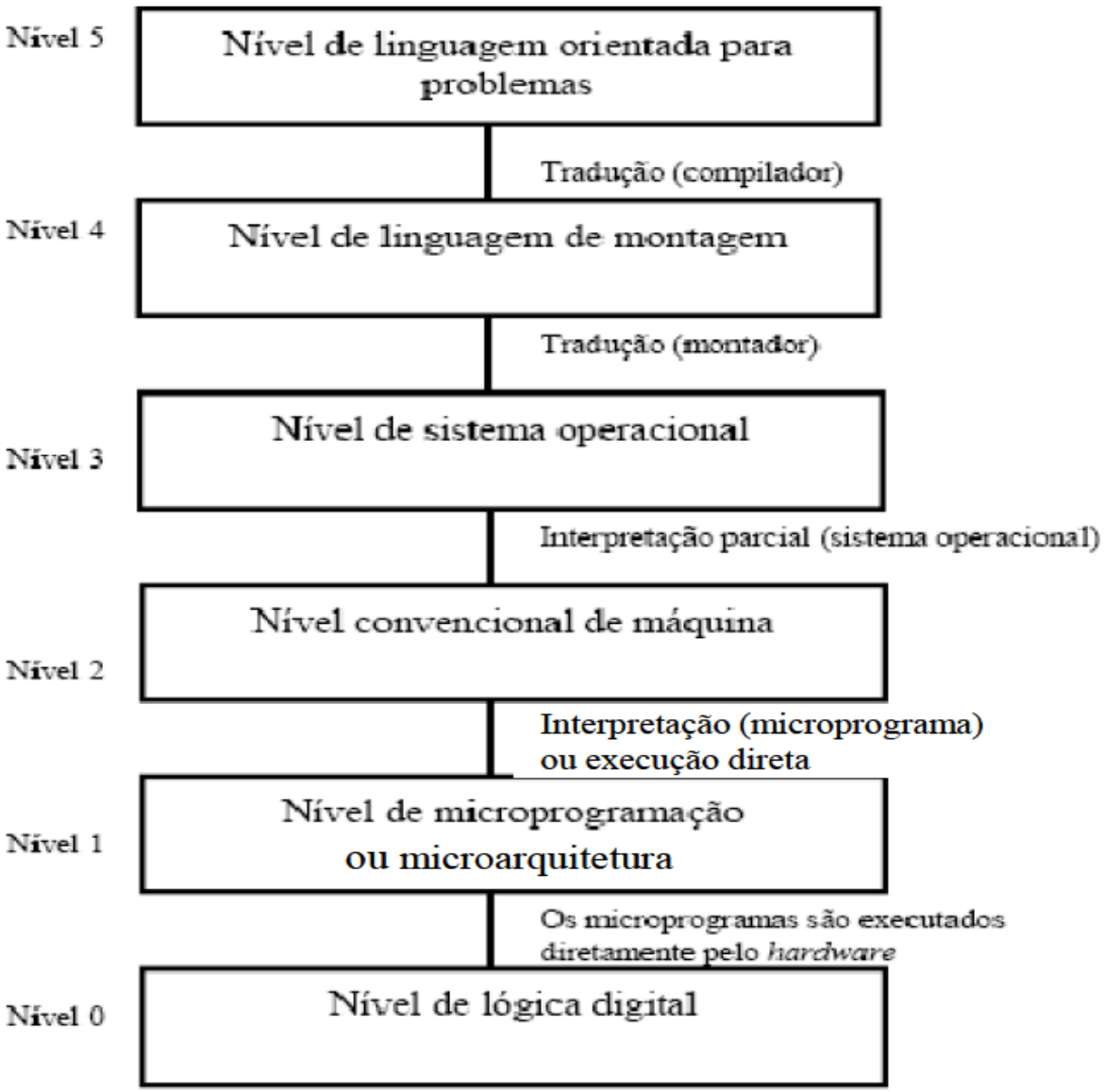
- Um computador com *n* níveis pode ser visto como *n* máquinas virtuais distintas.

Máquinas Multi-Níveis

- A estrutura de níveis permite ver o computador como um conjunto hierárquico de facilidades que possibilitam flexibilidade e independência ao usuário.
- Uma pessoa cujo trabalho seja gerar programas para a máquina virtual de nível "k" ou outro qualquer, não precisa conhecer ou se preocupar com as particularidades dos níveis inferiores.

Máquinas Multi-Níveis

- As máquinas modernas possuem os níveis semelhantes aos mostrados na figura:



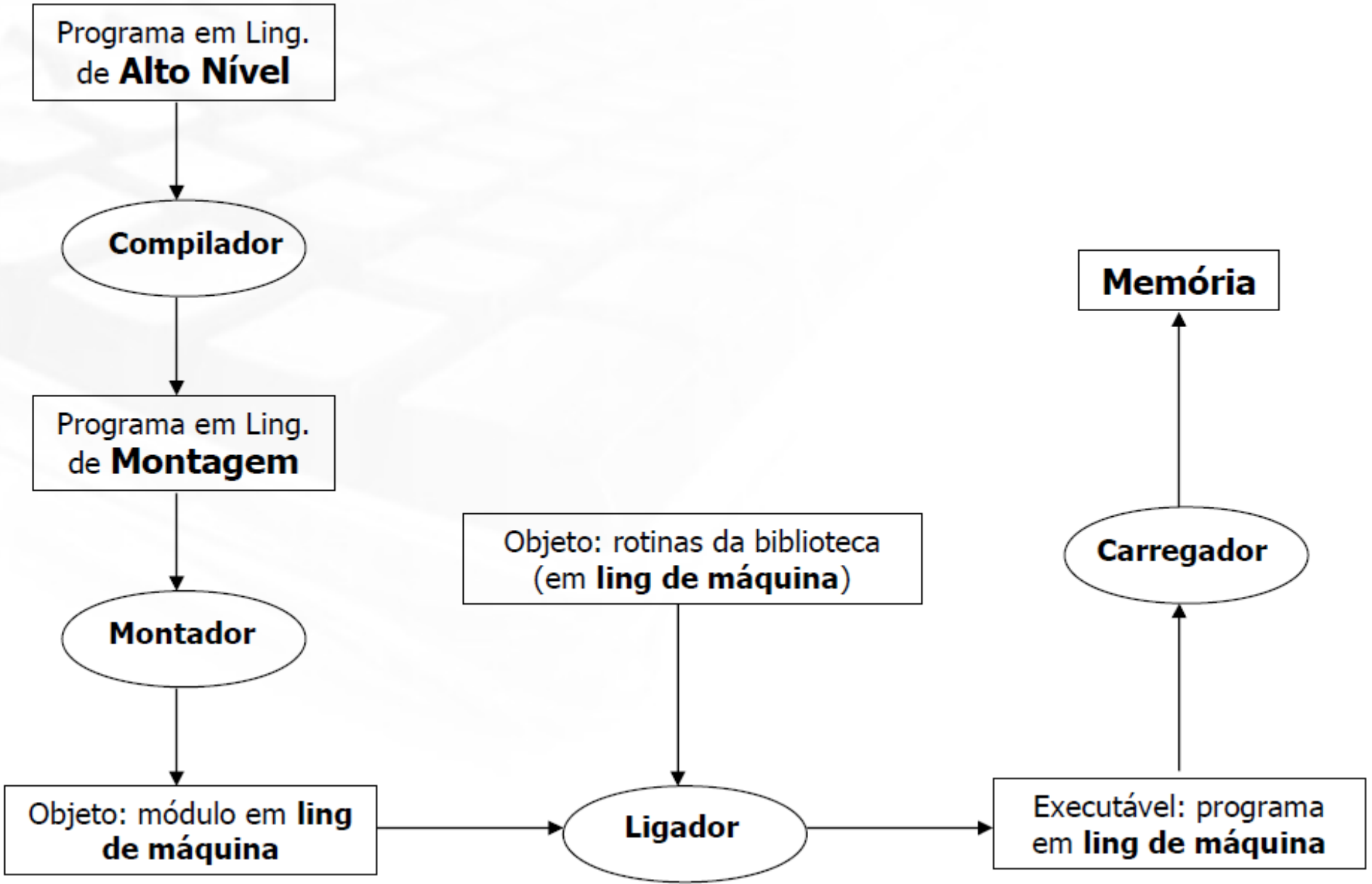
Arquitetura de Computadores

- **Ponto fundamental**: os computadores são projetados como uma série de níveis, cada um deles construído em cima de seus precursores.
- Cada nível representa uma abstração distinta, com diferentes objetos e operações presentes (**Arquitetura do Nível**).
- Abstrai-se o que é irrelevante, reduzindo a complexidade e focando no que interessa.
- **Arquitetura de Computadores**: é o estudo de como projetar as partes de um sistema de computador visíveis aos programadores.

Hardware, Software e Firmware

- **Hardware**
  - É composto por objetos tangíveis (parte física) - circuitos integrados, placas de circuito impresso, cabos, fontes de alimentação, memórias, impressoras, etc.
- **Software**
  - É composto de instruções, algoritmos e por suas representações computacionais - os programas.
- **Firmware**
  - Software embarcado, trata-se de um software que controla o hardware diretamente. Ex.: BIOS (Basic Input/Output System).

Execução de um Programa



Execução de um Programa

- **Compiladores**
  - São programas que recebem como entrada arquivos texto contendo módulos escritos em linguagem de alto nível e geram como saída arquivos objeto correspondentes a cada módulo.
  - Se todas as bibliotecas ou módulos são apresentados como entrada, geram um programa executável diretamente.

Execução de um Programa

- **Interpretadores**
  - Recebem como entrada arquivos texto contendo programas em linguagem assembly ou linguagem de alto nível, ou arquivos binários com instruções de máquina, e os executam diretamente.
  - Interpretadores percorrem os programas, a partir de seu ponto de entrada, executando cada comando.
- Processadores são interpretadores implementados em hardware!

Execução de um Programa

- **Montadores (Assemblers)**
  - Montam um programa em linguagem de máquina a partir de sua versão em linguagem de montagem. Geram um arquivo objeto. Em geral, não pode ser executado diretamente pela máquina, por conter referências a sub-rotinas e dados especificados em outros arquivos.

Execução de um Programa

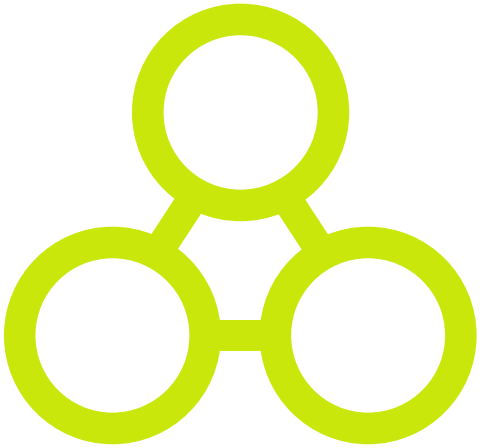
- **Ligadores (Linkers)**
  - São programas especiais que recebem como entrada arquivos objetos e geram como saída o programa final em linguagem de máquina. Gera um programa executável a partir de um ou mais arquivos objeto.
- **Carregadores (Loaders)**
  - Para executar um programa, um loader deve ser utilizado. O carregador é, em geral, parte do sistema operacional.

Execução de um Programa

- **Observações Finais**
- Pontos fundamentais:
  - Computadores são projetados como uma série de níveis
  - Cada nível é construído em cima de seus precursores.
  - Cada nível representa uma abstração distinta, com diferentes objetos e operações presentes
  - A abstração permite ignorar, "abstrair", temporariamente detalhes irrelevantes, de níveis mais baixos, reduzindo uma questão complexa a algo muito mais fácil de ser entendido.
  - Arquitetura do Nível: conjunto de tipos de dados, instruções e características de um nível de abstração da máquina

Referências

- Andrew S. Tanenbaum, **Organização Estruturada de Computadores**, 4ª edição, Prentice-Hall do Brasil, 2001.
- Lúcia Helena M. Pacheco, **Visão Geral de Organização Estruturada de Computadores e Linguagem de Montagem.** Universidade Federal de Santa Catarina. Centro Tecnológico, Departamento de Informática e de Estatística.



Introdução à Computação

Jordana Sarmenghi Salamon
jssalamon@inf.ufes.br
jordanasalamon@gmail.com
http://inf.ufes.br/~jssalamon

Departamento de Informática
Universidade Federal do Espírito Santo