ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I

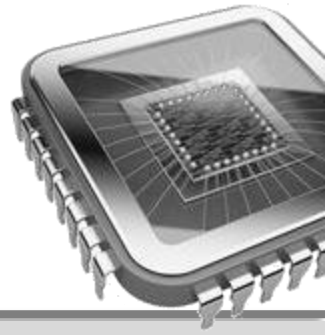
AULA 03: FUNCIONAMENTO DE UM COMPUTADOR

Prof. Max Santana Rolemberg Farias
max.santana@univasf.edu.br
Colegiado de Engenharia de Computação

GESE
Grupo de Engenharia de Sistemas Embarcados

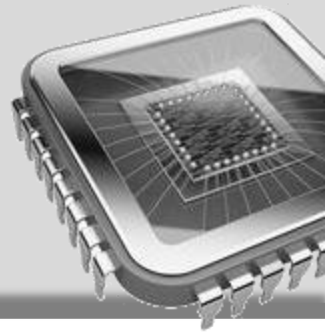
CEComp
Engenharia da Computação

UNIVASF



O QUE É UM COMPUTADOR?

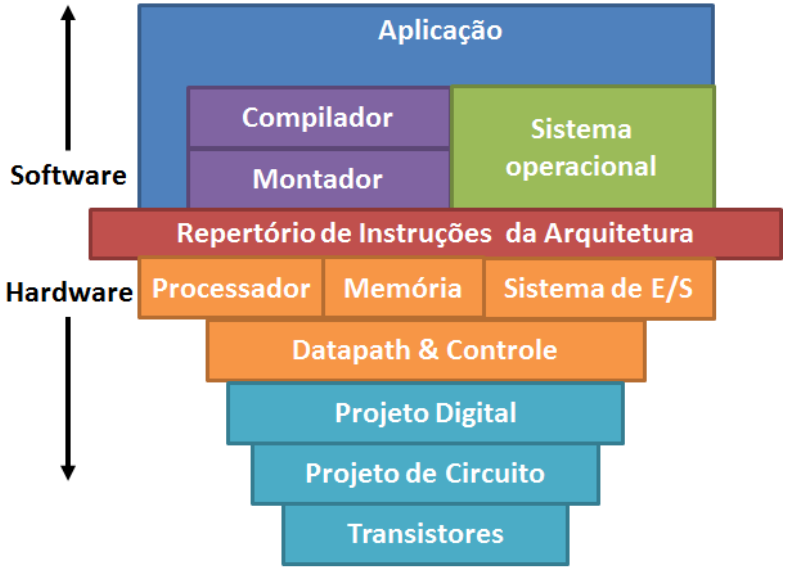
COMPUTADOR

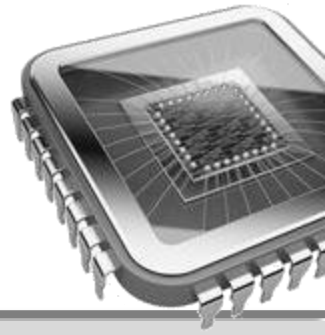


- Uma máquina digital que pode resolver problemas executando um programa (software).

Software é um conjunto de instruções que descrevem a maneira de realizar determinada tarefa.

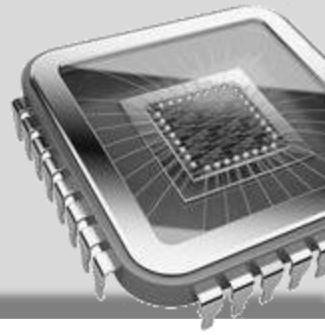
Hardware é responsável por reconhecer e executar o conjunto de instruções.



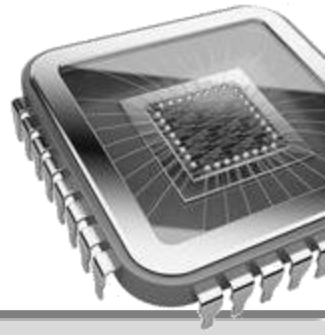


COMO UM PROGRAMA ESCRITO EM ALTO NÍVEL É ENTENDIDO E EXECUTADO PELO HARDWARE?

LINGUAGEM DO HARDWARE

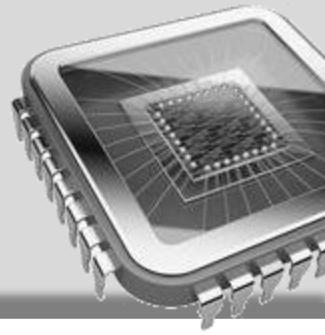


- O hardware só entende sinais elétricos (dois valores)
  - Ligado (*on*)/Desligado (*off*)
  - Zeros e uns (números binários)
- As instruções para o computador são sequências de números binários (linguagem de máquina)
- A linguagem de máquina está muito distante de uma linguagem natural (linguagem humana)
- Programar em linguagem de máquina é **impraticável**!



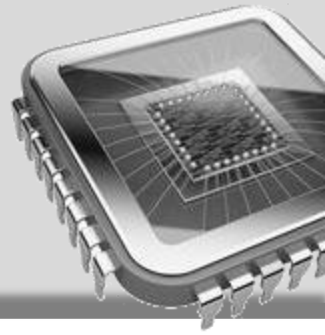
COMO INSTRUIR OS COMPUTADORES?

SISTEMA HIERÁRQUICO

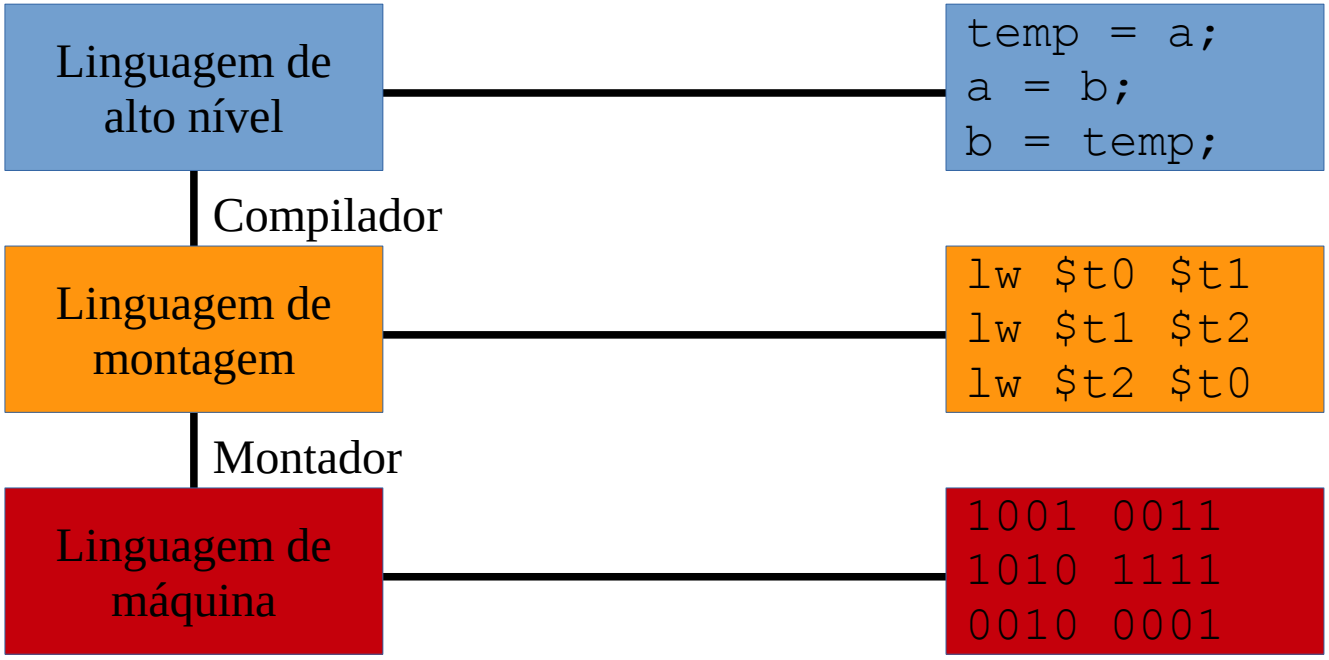


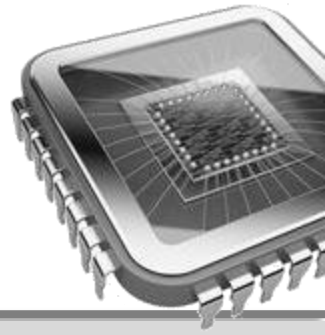
- Criar uma **organização estruturada de computadores** para facilitar a comunicação **homem-computador**.
  - Criando uma hierarquia de abstrações de níveis mais alto baseados nos níveis mais baixos.
    - Um conjunto de subsistemas inter-relacionados, onde, cada qual, possui uma estrutura hierárquica, que contém em seu nível mais baixo subsistemas elementares.
    - Cada nível depende apenas de uma caracterização abstrata e simplificada do sistema de nível imediatamente inferior.
    - Abstraindo os conceitos de hardware.
  - Criando linguagens de programação.
    - Linguagens entendida pelos seres humanos e que possam ser traduzidas para a linguagem de máquina (entendida pelos computadores)

SISTEMAS HIERÁRQUICO



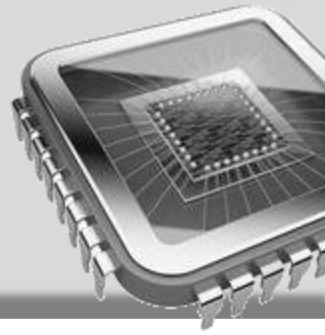
- As linguagens de programação variam de acordo com o seu nível de abstração.



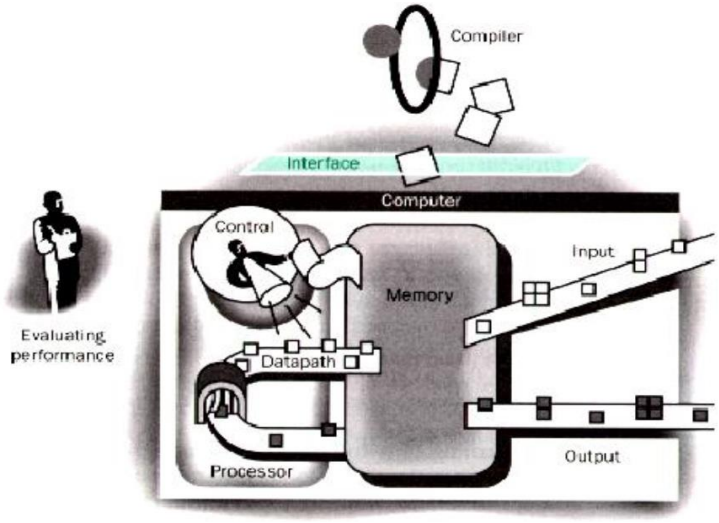


COMO UM COMPUTADOR ENTENDE UMA LINGUAGEM DE PROGRAMAÇÃO?

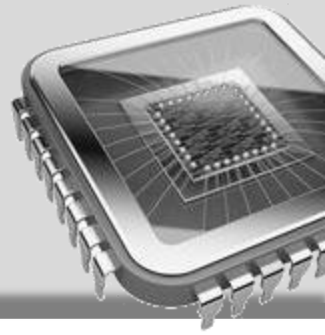
COMO O COMPUTADOR ENTENDE UMA LINGUAGEM DE PROGRAMAÇÃO



- A linguagem de programação deve ser traduzida para linguagem de máquina.
  - Usando um compilador (tradutor) ou um interpretador.
  - Pode ser utilizada uma metodologia híbrida que utilizada um compilador e um interpretador.

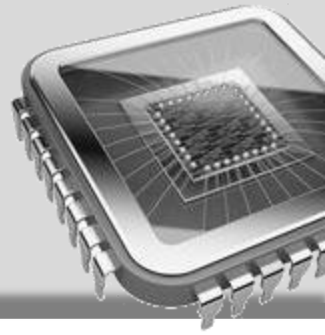


COMPILADOR



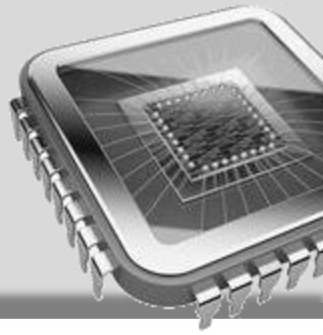
- O compilador é um programa (software) que traduz um programa escrito em uma linguagem de programação de alto nível (código fonte) em uma linguagem de montagem (*assembly*).
  - Cada instrução da linguagem é substituída por um conjunto de instruções equivalentes da linguagem de montagem.
  - Todo o programa em linguagem de montagem é carregado em memória e executado
  - O programa pode ser traduzido uma única vez e executado várias vezes.

INTERPRETADOR



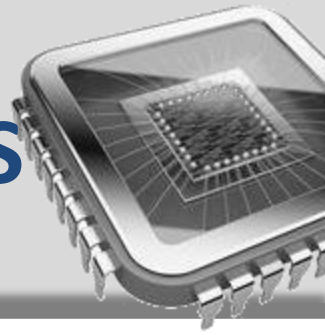
- O interpretador traduz instrução por instrução de um programa escrito em linguagem de programação em linguagem de máquina e imediatamente executa a instrução.
  - Cada instrução da linguagem de programação é substituída por um conjunto de instruções equivalentes da linguagem de máquina.
  - Cada instrução da linguagem de programação transformada para linguagem de máquina é carregada na memória e executada.
  - Não é criado um programa em linguagem de máquina.
  - O programa deve ser novamente interpretado para ser executado.

COMPILAÇÃO X INTERPRETAÇÃO



- Existem vários exemplos tanto de linguagens compiladas como de linguagens compiladas.
  - A linguagem C é um exemplo de linguagem compilada.
  - A linguagem Java é uma linguagem de programação que utiliza um processo híbrido de tradução.
    - O compilador Java traduz o código-fonte em um formato intermediário independente de máquina chamado ***bytecode***.
    - O interpretador Java específico da máquina (onde irá rodar o programa) traduz o ***bytecode*** para linguagem de máquina.

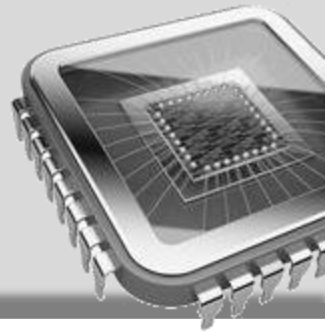
NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS



- As linguagens de programação podem ser classificadas em **6 níveis**.
  - Nível lógico digital (**Nível 0**)
  - Nível de microarquitetura (**Nível 1**)
  - Nível de conjunto de instruções (**Nível 2**)
  - Nível de sistema operacional (**Nível 3**)
  - Nível de montagem (**Nível 4**)
  - Nível de Aplicação (**Nível 5**)

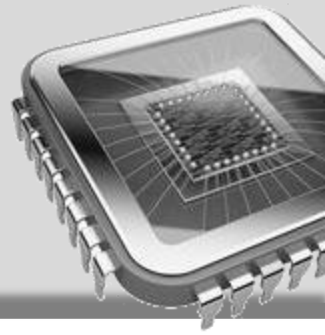
- Cada linguagem usa a sua linguagem antecessora como base, de modo que um computador que use essa técnica pode ser visto como um conjunto de camadas ou níveis.

NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS
NÍVEL 0: NÍVEL LÓGICO DIGITAL



- Nível mais baixo da estrutura.
- Os objetos de interesse são conhecidos como **portas lógicas**.
- Cada porta lógica possui 1 ou mais entradas digitais (aceitam 0 ou 1) e calculam funções lógicas simples sobre essas entradas:
  - AND, OR, XOR, ...

NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS
NÍVEL 1: NÍVEL DE MICROARQUITETURA

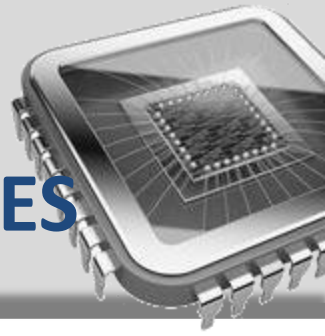


- Uma memória local (8 a 32 registradores) e uma ULA (Unidade Lógica e Aritmética) que realiza operações aritméticas muito simples.

- O caminho dos dados é formado pela conexão dos registradores com a ULA.

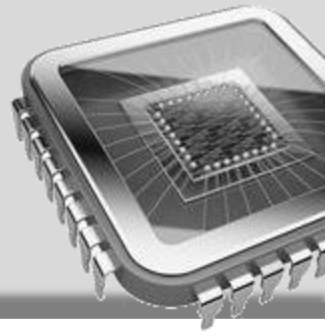
- As operações são controladas por um micro programa ou diretamente por hardware.
  - O micro programa é um interpretador para as instruções no **Nível 2**.
  - Busca, decodifica e executa as instruções, uma a uma, usando o caminho de dados para a realização desta tarefa.

NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS
NÍVEL 2: NÍVEL DE CONJUNTO DE INSTRUÇÕES



- Nível ISA (*Instruction Set Architecture*).
- Definida pelo fabricante e depende da arquitetura da máquina.
- O fabricante disponibiliza o **manual de referência da linguagem de máquina** ou o **princípios de operação do computador modelo XYZW**
  - Os manuais descrevem como as instruções são executadas interpretativamente pelo micro programa ou como são executadas diretamente pelo hardware.
  - Essas informações são necessárias para o desenvolvedores de sistemas operacionais.

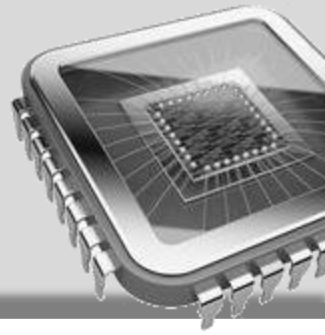
NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS
NÍVEL 3: NÍVEL DE SISTEMA OPERACIONAL



- Instruções da linguagem deste nível também podem conter instruções do nível ISA.
  - Suporta uma organização diferente de memória.
  - Suporta capacidade de rodar 2 ou mais programas simultaneamente.
  - Suporta sistemas de comandos ou de janelas (*windows*).

- Os programadores deste nível, e também dos níveis mais baixos, são conhecidos como programadores de sistema.
  - Os programadores dos níveis mais altos que este são chamados de programadores de aplicações.

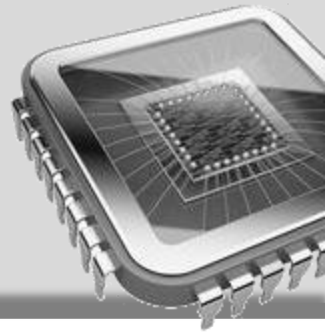
NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS
NÍVEL 4: NÍVEL DE MONTAGEM



- A linguagem de montagem é uma forma simbólica de representação das linguagens do nível mais baixo.

- Programas em linguagem de montagem são, inicialmente, traduzidos para as linguagens dos níveis 1, 2 e 3 e depois interpretados pela máquina.
  - O programa que realiza essa tradução é chamado de montador.

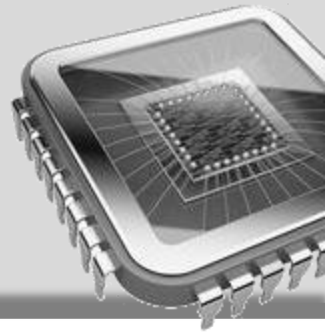
NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS
NÍVEL 4: NÍVEL DE APLICAÇÃO



- Nesse nível as linguagens são conhecidas como linguagens de alto nível.
  - C
  - Python
  - Java

- Os programas são geralmente traduzidos para o nível 3 e 4 por **compiladores**.
  - Alguns desses programas podem ser interpretados, como os programas em Java.

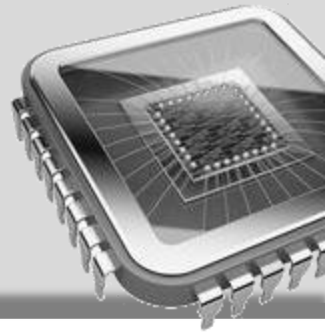
NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS OBSERVAÇÕES IMPORTANTES



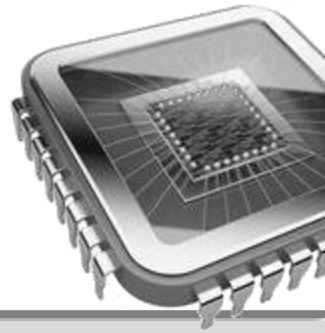
- Os computadores são projetados como uma série de níveis, onde cada nível é construído em cima de seus precursores.
  - O programador de um nível, em geral, não deve se preocupar com implementações de níveis inferiores.
  - Do ponto de vista do programador, não tem muita importância a maneira como uma instrução é realmente implementada.

- Cada nível representa uma abstração distinta, com diferentes objetivos e operações.
  - A abstração permite ignorar, abstrair, temporariamente detalhes irrelevantes, de níveis mais baixos, **reduzindo uma questão complexa a algo muito mais fácil de ser entendido**.

NÍVEIS DE ABSTRAÇÃO DE LINGUAGENS OBSERVAÇÕES IMPORTANTES

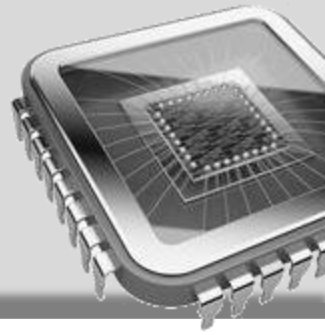


- Atualmente é muito difícil separar o **hardware** do **software**.
  - Nos primeiros computadores essa fronteira era muito bem definida.
  - Hoje, o hardware e o software são equivalentes logicamente
    - Qualquer operação realizada por software pode ser realizada diretamente por hardware.
    - Qualquer instrução executada por hardware pode ser simulada em software.
    - O que está implementado em software poderá em breve estar implementado em hardware (**o software hoje pode ser o hardware de amanhã e vice-versa**).



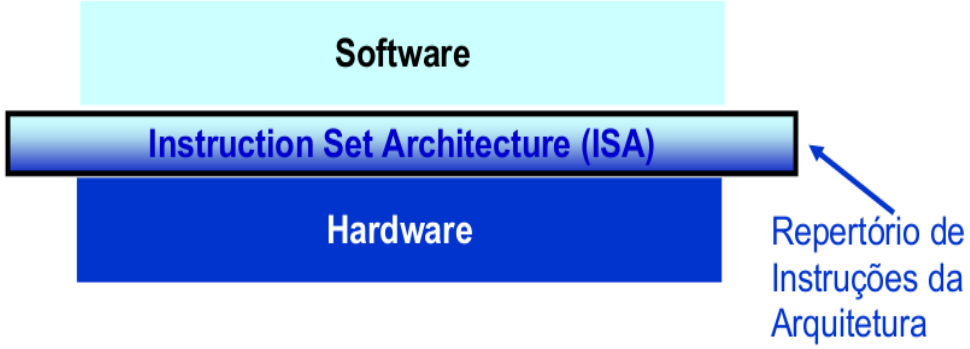
QUAL É A INTERFACE ENTRE O SOFTWARE E O HARDWARE?

INTERFACE HARDWARE/SOFTWARE
REPERTÓRIO DE INSTRUÇÕES

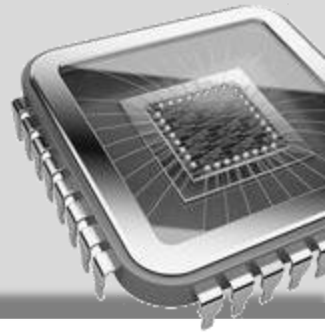


- Última abstração do Hardware vista pelo software.
  - É a interface entre o software e o hardware
  - É a arquitetura da máquina

- Provê a informação necessária para que se escreva um código em linguagem de máquina (ou montagem) que execute corretamente na arquitetura.
  - Instruções
  - Registradores
  - Acesso a memória
  - Entrada/Saída

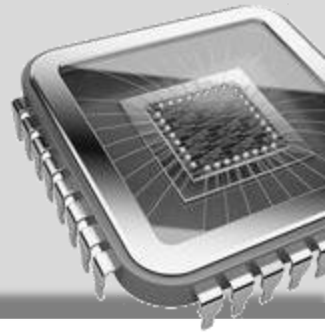


EVOLUÇÃO DA INTERFACE HARDWARE/SOFTWARE



- Até metade da década de 1960, os computadores tinham ISAs com quantidades reduzida de instruções e instruções simples.
  - Para simplificar as implementações
- No final da década de 1960 surge ISAs com grnade número de instruções complexas.
  - ***Complex Instruction Set Computer*** (**CISC**)
  - Difícil implementação e existência de muitas instruções pouco usadas
- Começo da década de 1980 surge ISAs com instruções simples.
  - Voltam a ser comuns
  - ***Reduced Instruction Set Computer*** (**RISC**)

PROCESSADORES RISC E CISC



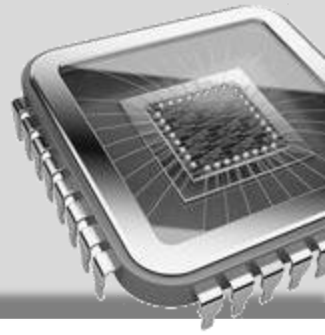
Arquitetura RISC

- Poucas instruções.
- Instruções executadas pelo hardware.
- Instruções com formatos fixo.
- Instruções utilizam poucos ciclos de máquina.
- Instruções com pouco modos de endereçamento.
- Arquitetura com muitos registradores.
- Arquitetura pipeling.

Arquitetura CISC

- Muitas instruções.
- Instruções executadas por microcódigos.
- Instruções com diversos formatos.
- Instruções utilizam múltiplos ciclos.
- Instruções com diversos modos de endereçamento.
- Arquitetura com poucos registradores.
- Pouco uso da técnica de pipeling.

EXEMPLO DE PROCESSADORES RISC E CISC



Arquitetura RISC

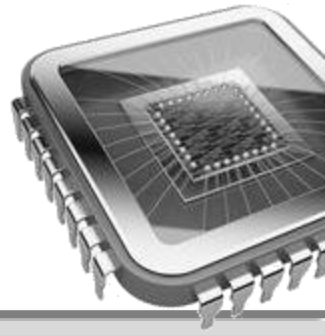
- MIPS;
- SPARC;
- ARM;
- PowerPC.

Muito utilizado em sistemas embarcados.

Arquitetura CISC

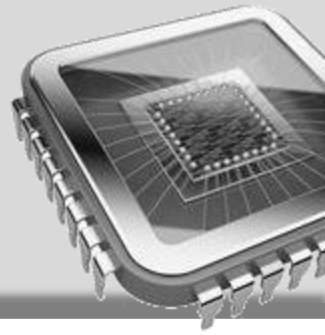
- Intel x86;
- Pentium;
- AMD x86;
- AMD Athlon.

Muito utilizados em PCs.



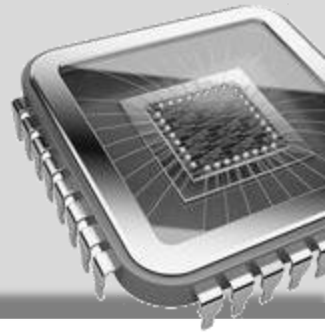
QUAL A FUNÇÃO BÁSICA DO COMPUTADOR?

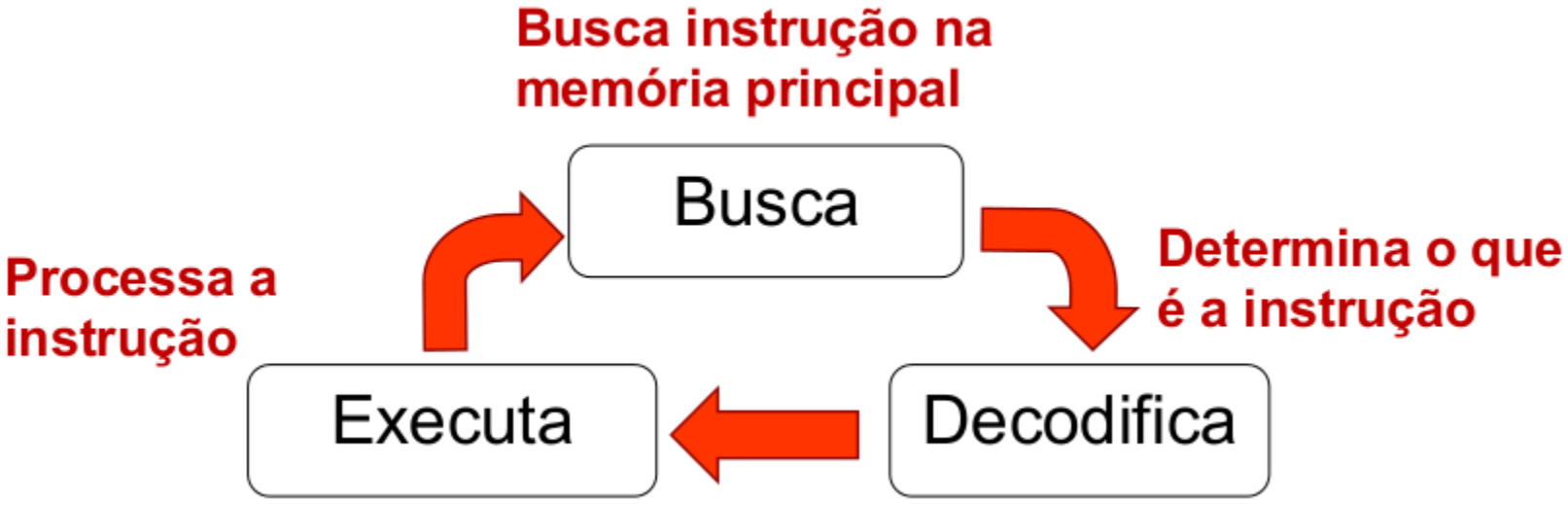
FUNÇÃO BÁSICA DO COMPUTADOR



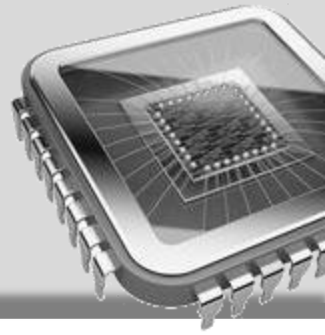
- A função básica do computador é executar um programa que é constituído por um conjunto de instruções armazenadas na memória.
  - O conteúdo da memória é acessado através de um endereço, não importando o tipo de dado armazenado.
  - A execução ocorre de maneira sequencial (a não ser que seja explicitamente especificado), uma instrução após a outra.
  - Para execução de um programa, o processador faz **3 ações** continuamente: **Busca**, **Decodificação** e **Execução**.

FUNÇÃO BÁSICA DO COMPUTADOR





FUNÇÃO BÁSICA DO COMPUTADOR



- O processador busca uma instrução na memória.
  - Um registrador contador de instruções ou contador de programa (***program counter*** – **PC**) é utilizado para armazenar o endereço da próxima instrução a ser buscada na memória.
  - A instrução buscada é carregada no registrador de instruções (***instruction register*** – **IR**) do processador.

- O processador decodifica a instrução e executa a ação requisitada.
  - As ações requisitadas são classificadas em **4 categorias**:
    - Processador - Memória;
    - Processador - E/S;
    - Processamento de dados;
    - Controle.

FUNÇÃO BÁSICA DO COMPUTADOR

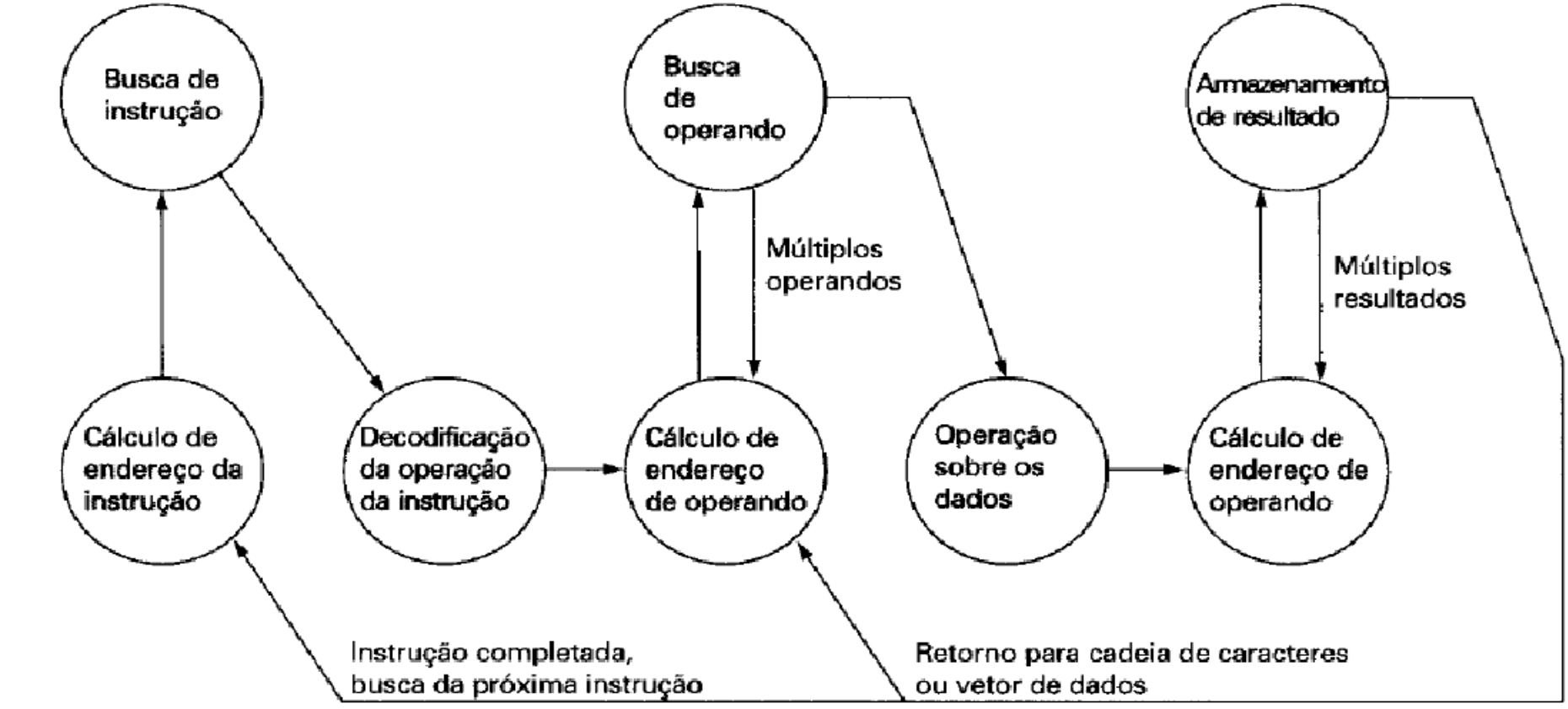
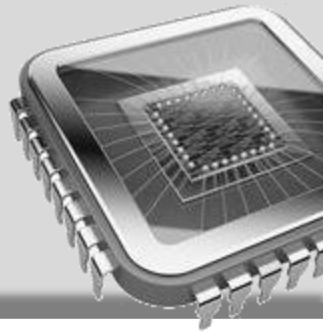


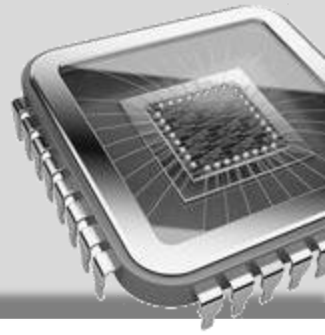
Diagrama de estados do ciclo de instruções

ESTADOS DO CICLO DE INSTRUÇÃO



- **Cálculo de endereço de instrução**
  - Calcula o endereço da próxima instrução a ser executada.
- **Busca de instrução**
  - Ler uma instrução da memória e armazena no registrador.
- **Decodificação da operação da instrução**
  - Decodifica e analisa o código da instrução para determinar qual é a operação a ser realizada e os operandos a serem utilizados.

ESTADOS DO CICLO DE INSTRUÇÃO



- **Cálculo de endereço de operando**
  - Se a operação envolver a referência a um operando na memória ou estiver disponível via E/S, o endereço do operando será determinado.
- **Busca do operando**
  - O operando é localizado na memória ou é lido no dispositivo de E/S.
- **Operação sobre os dados**
  - A operação indicada na instrução é executada.
- **Armazenamento de resultado**
  - O resultado é escrito na memória ou no dispositivo de E/S.

VISÃO DETALHADA DA EXECUÇÃO