INE 5416/5636 - Paradigmas de programação

Turmas 04208/08238

Prof. Dr. João Dovicchi – dovicchi@inf.ufsc.br

http://www.inf.ufsc.br/~dovicchi

Máquina de Antiquitera

Primeira calculadora mecânica conhecida, considerada o primeiro computador analógico ou o mais antigo computador conhecido.

Máquina Dedicada como várias outras.

Idéia do Computador...

Partiu dos conceitos de:

Idéia do Computador...

Partiu dos conceitos de:

- Máquina programável (Ada Lovelace)

Panorama histórico

Idéia do Computador...

Partiu dos conceitos de:

- Máquina programável (Ada Lovelace)
- Lógica combinatória (Schönfinckel – Haskell B. Curry)

Panorama histórico

Idéia do Computador...

Partiu dos conceitos de:

- Máquina programável (Ada Lovelace)
- Lógica combinatória (Schönfinckel – Haskell B. Curry)
- Computabilidade (Church – Turing)

Panorama histórico

Idéia do Computador...

Partiu dos conceitos de:

- Máquina programável (Ada Lovelace)
- Lógica combinatória (Schönfinckel – Haskell B. Curry)
- Computabilidade (Church – Turing)
- Arquitetura de máq. de estado com memória e endereço (Modelo von Neumann)

Panorama histórico

Ada Lovelace

Cartões da máquina de bordados de Jacquard.

Babbage e Ada: Inventaram a máquina analítica a partir da idéia do tear de Jacquard, onde os cartões de operação (de desenho do jacquard) foram substituídos por cartões de padrões algébricos.

Ada criou os cartões para programar a máquina, tornando-se a primeira programadora.

Panorama histórico

Lógica combinatória (Schönfinkel – Haskell B. Curry)

Lógica combinatória: notação introduzida por Moses Schönfinkel , revista e ampliada por Haskell Curry para eliminar a necessidade de variáveis na lógica matemática.

Panorama histórico

Lógica combinatória (Schönfinkel – Haskell B. Curry)

Lógica combinatória: notação introduzida por Moses Schönfinkel , revista e ampliada por Haskell Curry para eliminar a necessidade de variáveis na lógica matemática.

São transformações por meio de funções de alta ordem descritas previamente, chamados de combinadores SKI.

Panorama histórico

Lógica combinatória (Schönfinkel – Haskell B. Curry)

Lógica combinatória: notação introduzida por Moses Schönfinkel , revista e ampliada por Haskell Curry para eliminar a necessidade de variáveis na lógica matemática.

São transformações por meio de funções de alta ordem descritas previamente, chamados de combinadores SKI.

- **I** retorna o argumento (identidade): $Ix \rightarrow x$

Panorama histórico

Lógica combinatória (Schönfinkel – Haskell B. Curry)

Lógica combinatória: notação introduzida por Moses Schönfinkel , revista e ampliada por Haskell Curry para eliminar a necessidade de variáveis na lógica matemática.

São transformações por meio de funções de alta ordem descritas previamente, chamados de combinadores SKI.

I retorna o argumento (identidade): $Ix \rightarrow x$

K aplicado a um argumento retorna uma função constante de um argumento. Esta função aplicada a qualquer argumento, retorna o argumento da função constante: $Kx \rightarrow (Kx)$ e $(Kx)y \rightarrow x$

Panorama histórico

Lógica combinatória (Schönfinkel – Haskell B. Curry)

Lógica combinatória: notação introduzida por Moses Schönfinkel , revista e ampliada por Haskell Curry para eliminar a necessidade de variáveis na lógica matemática.

São transformações por meio de funções de alta ordem descritas previamente, chamados de combinadores SKI.

I retorna o argumento (identidade): $Ix \rightarrow x$

K aplicado a um argumento retorna uma função constante de um argumento. Esta função aplicada a qualquer argumento, retorna o argumento da função constante: $Kx \rightarrow (Kx)$ e $(Kx)y \rightarrow x$

S é um operador de substituição que recebe 3 argumentos e retorna a aplicação do primeiro aplicado ao terceiro no segundo aplicado ao terceiro: $Sxyz \rightarrow xz(yz)$

Panorama histórico

Lógica combinatória (Schönfinkel – Haskell B. Curry)

A lógica combinatória é a base do cálculo-lambda.

Veja mais em:

`http://people.cs.uchicago.edu/~odonnell/Teacher/Lectures/Formal_Organization_of_Knowledge/Examples/combinator_calculus/`

Panorama histórico

O Problema da computabilidade (Church-Turing)

Alonso Church: usando a lógica combinatória $\rightarrow$ desenvolve o cálculo lambda

Usando as reduções do cálculo-λ $\rightarrow$ prova a computabilidade

Panorama histórico

O Problema da computabilidade (Church-Turing)

Alonso Church: usando a lógica combinatória $\rightarrow$ desenvolve o cálculo lambda

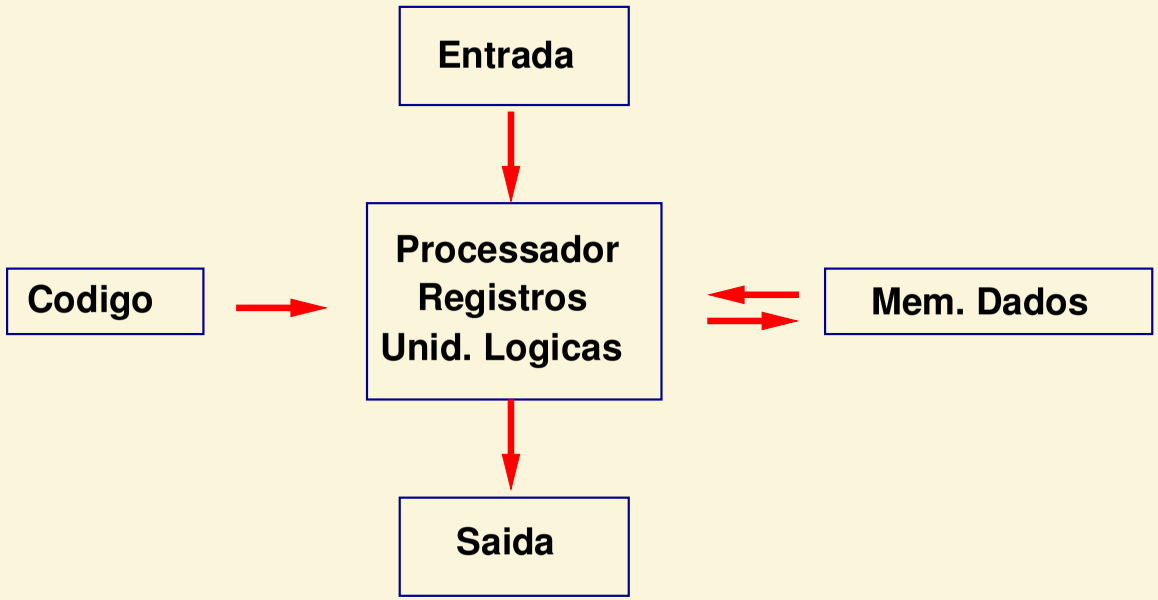
Usando as reduções do cálculo-λ $\rightarrow$ prova a computabilidade

Turing: usando o modelo de máquinas de estado para provar a mesma coisa.

Panorama histórico

Modelo de von Neuman

Proposto por von Neumann em 1940, é o modelo utilizado atualmente: O Processador segue as instruções armazenadas em uma memória de programas, para ler canais de entrada, enviar comandos sobre canais de saída e alterar as informações contidas em uma memória de dados.



Panorama histórico

Transformação histórica:

Primeiras linguagens: Funcionais / lógicas (circuitos)

Panorama histórico

Transformação histórica:

Primeiras linguagens: Funcionais / lógicas (circuitos)

Modelo de von Neuman: Procedural

Panorama histórico

Transformação histórica:

Primeiras linguagens: Funcionais / lógicas (circuitos)

Modelo de von Neuman: Procedural

Linguagem de máquina: procedural conforme conjunto de instruções

Panorama histórico

Transformação histórica:

Primeiras linguagens: Funcionais / lógicas (circuitos)

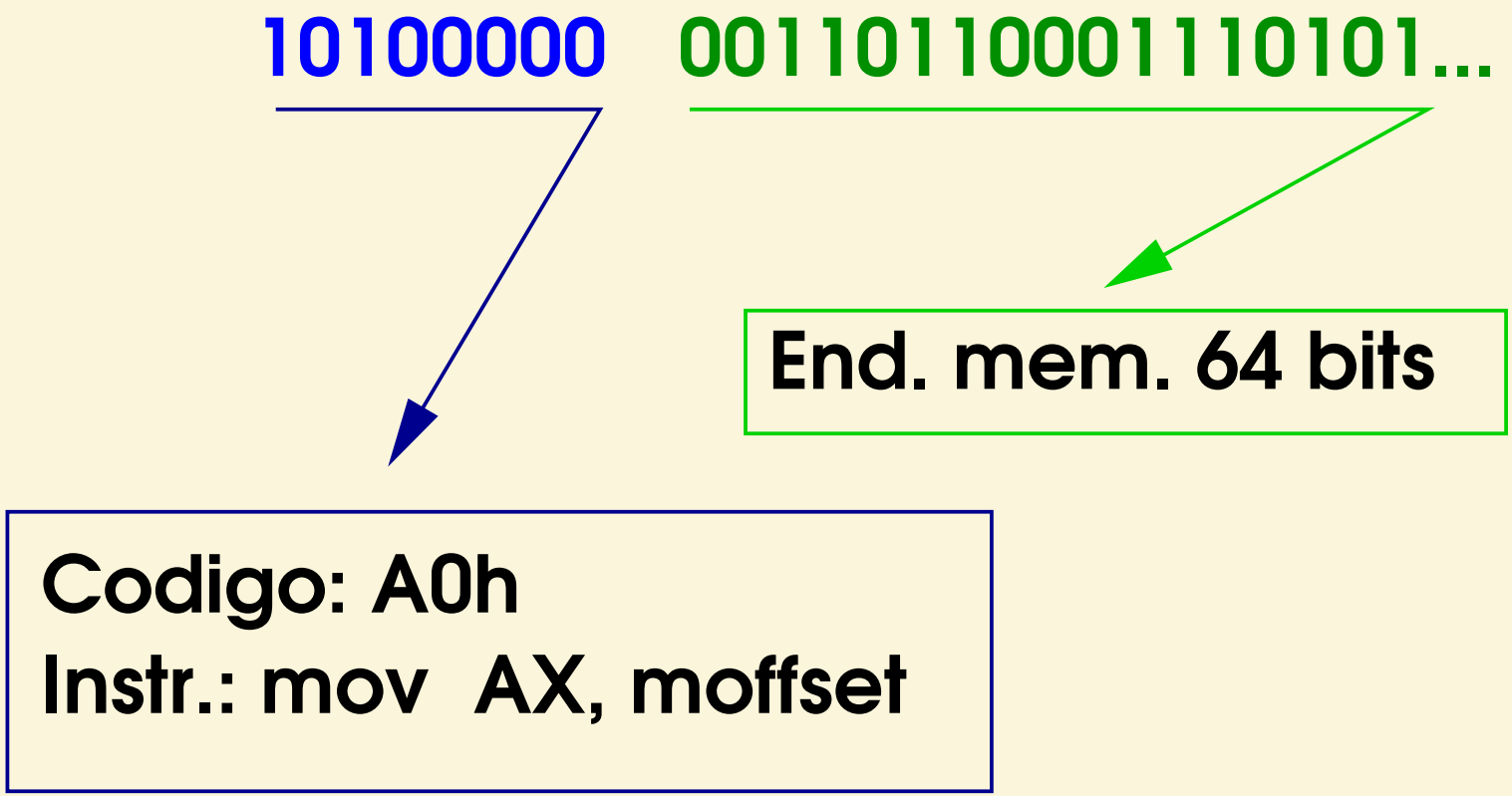
Modelo de von Neuman: Procedural

Linguagem de máquina: procedural conforme conjunto de instruções

Até quando???

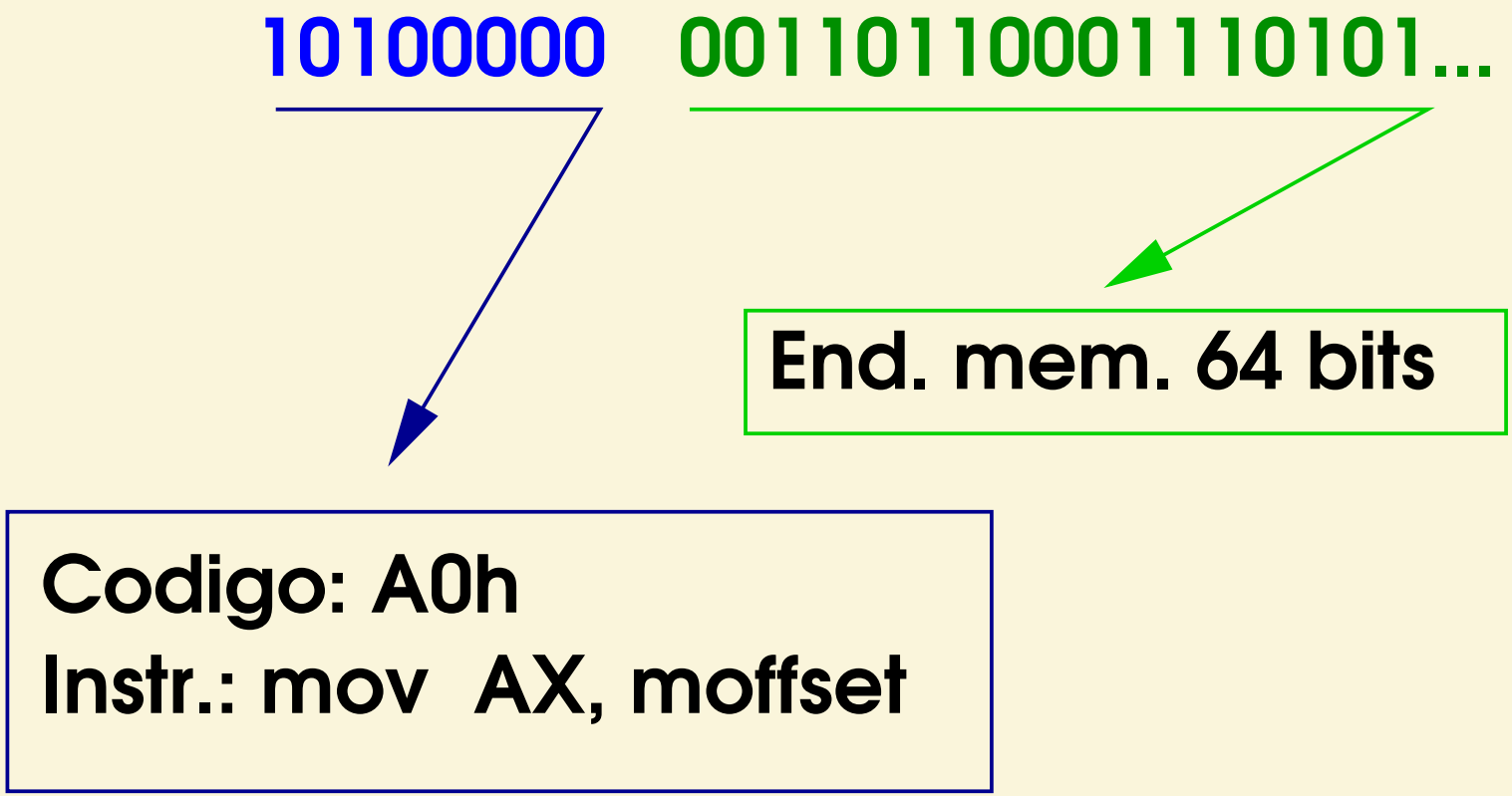
Panorama histórico

Linguagem de máquina:



Panorama histórico

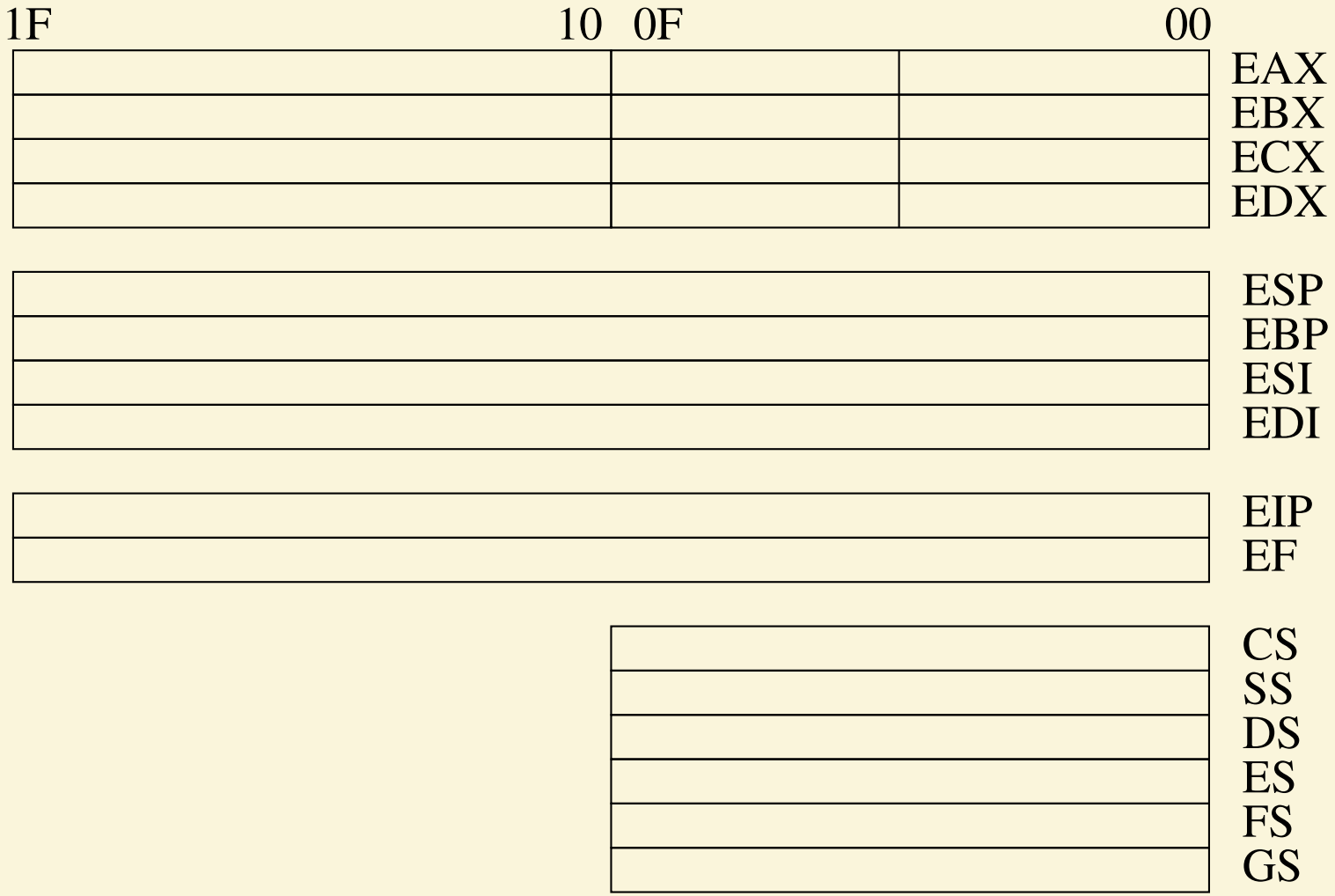
Linguagem de máquina:



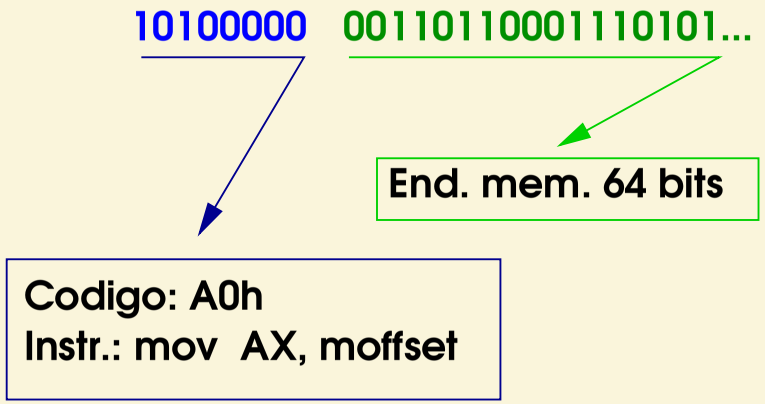
Carregue, no registrador `AH` o conteúdo do endereço `moffset`

Panorama histórico

CPU

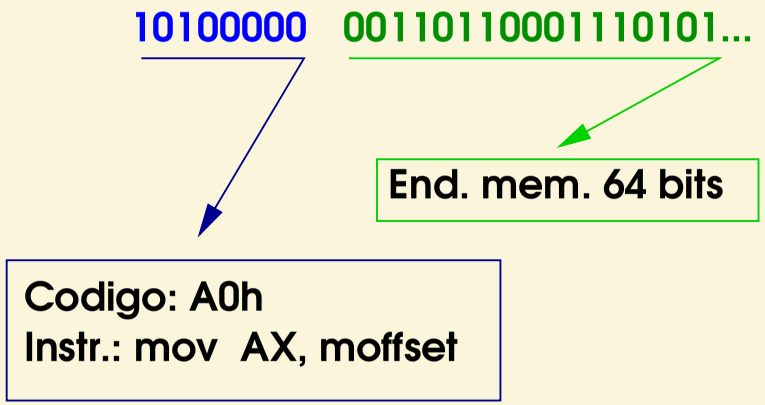


Panorama histórico



O processador:

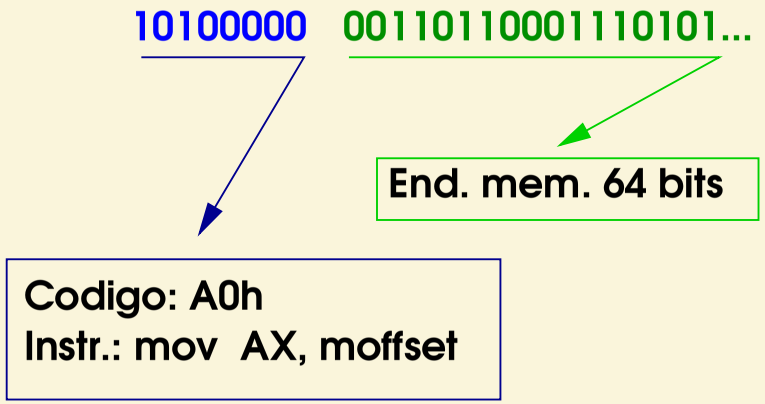
Panorama histórico



O processador:

1. Le a instrução apontada pelo registro IP (Instruction Pointer).

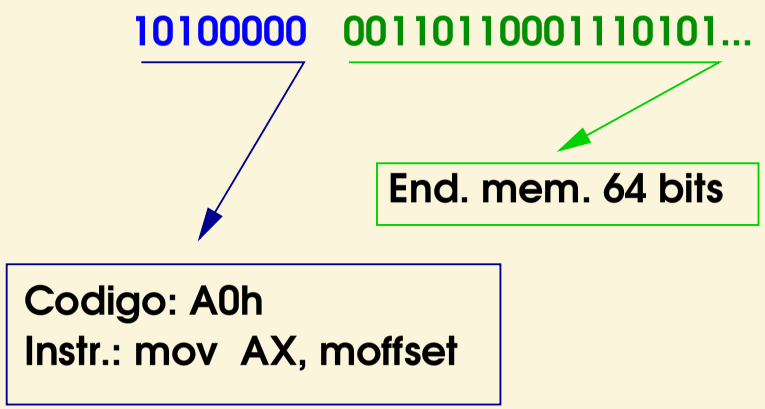
Panorama histórico



O processador:

1. Le a instrução apontada pelo registro IP (Instruction Pointer).
2. Carrega a instrução A0h no reg. CS (Code segment).

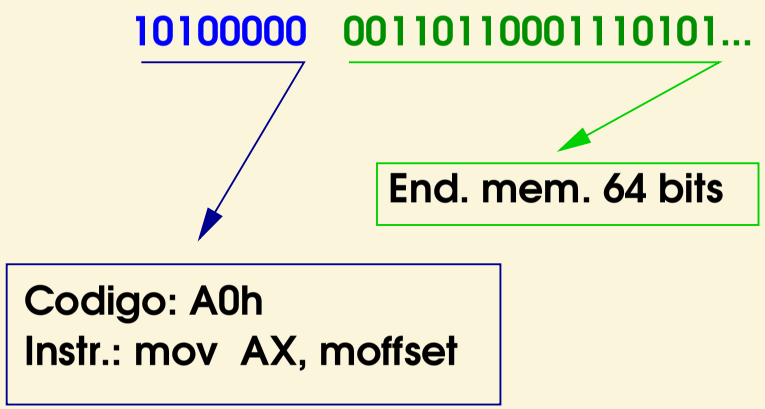
Panorama histórico



O processador:

1. Le a instrução apontada pelo registro IP (Instruction Pointer).
2. Carrega a instrução A0h no reg. CS (Code segment).
3. Carrega os 64 bits de moffset no reg. DS (Data Segment) ou 32+32 nos registradores DS:ES.

Panorama histórico



O processador:

1. Le a instrução apontada pelo registro IP (Instruction Pointer).
2. Carrega a instrução A0h no reg. CS (Code segment).
3. Carrega os 64 bits de moffset no reg. DS (Data Segment) ou 32+32 nos registradores DS:ES.
4. Transfere 64/32 bits no endereço de memória apontado pelo reg. para o reg. AX e incrementa o registro IP.

Panorama histórico

Linguagens:

Dificuldade em montar programa diretamente no set de instruções do processador $\rightarrow$ Assembly, autocode, IPL (códigos simbólicos).

Panorama histórico

Linguagens:

Dificuldade em montar programa diretamente no set de instruções do processador $\rightarrow$ Assembly, autocode, IPL (códigos simbólicos).

Assembly foi criado para facilitar a montagem do programa (assembler = montador).

Assembly não é propriamente uma linguagem

Panorama histórico

Linguagens:

Dificuldade em montar programa diretamente no set de instruções do processador $\rightarrow$ Assembly, autocode, IPL (códigos simbólicos).

Assembly foi criado para facilitar a montagem do programa (assembler = montador).

Assembly não é propriamente uma linguagem

Linguagem de alto nível $\rightarrow$ código objeto $\rightarrow$ montador

Solução: Compiladores !!!

Panorama histórico

Assembly

rótulo: mnemônico argumento1, argumento2, argumento3 ...

- O rótulo é um identificador seguido de dois pontos.
- O mnemônico é uma palavra reservada para o código da instrução do processador.
- Os operadores “argumento” são opcionais e podem ser em número de 0 a 3, dependendo do código do processador.

Panorama histórico

Assembly

rótulo: mnemônico argumento1, argumento2, argumento3 ...

- O rótulo é um identificador seguido de dois pontos.
- O mnemônico é uma palavra reservada para o código da instrução do processador.
- Os operadores “argumento” são opcionais e podem ser em número de 0 a 3, dependendo do código do processador.

Exemplo: `carr_reg:  mov  EAX, BFH`

Panorama histórico

Linguagens:

FORTRAN (FORmula TRANslator) - 1954 – 1958

Procedural e imperativa

Criada pela IBM (John Backus)

Dedicada a resolução de equações e fórmulas matemáticas

FORTRAN II: loops, funções, sub-rotinas e a primitiva do comando FOR.

Panorama histórico

Linguagens:

FORTRAN 77 (Exemplo)

```
Program fatorial
    implicit none

    integer N
    parameter (N=8)
    integer i
    integer fact

    fact = 1

    do i = 1,N,1
           fact = fact * i
    enddo

    write(*,*) 'fatorial de ', N, ' = ', fact

    end
```

Panorama histórico

Linguagens:

LISP (LISt Processor) - 1958 – 1960

Funcional

Criada por MacCarthy

Desenvolvida para processamento de listas

Puramente recursiva e não iterativa

Não diferencia código e dados

Panorama histórico

Linguagens:

LISP (Exemplo)

```
;; Programa fatorial em Lisp

 (defun factorial (n)
   (if (<= n 1)
       1 (* n (factorial (- n 1))))
   )
 )
```

Panorama histórico

Linguagens:

ALGOL (ALGOrithmic Language) - 1958 – 1968

Procedural, criada em 58 como IAL (International Algorithmic Language)

Criada por comite de especialistas em computação

Primeira linguagem autônoma, independente de arquitetura (portável)

Introduziu a declaração em blocos e variáveis locais, arrays dinâmicos, := para atribuição, loops IF...THEN...ELSE, FOR, SWITCH, WHILE ALGOL 68 define cast de tipos e UNION.

Panorama histórico

Linguagens:

ALGOL68 (Exemplo)

```
co
"n" eh uma variavel fixa (global) e "produto" eh uma
variavel local que recebe o valor de n.
Equivale a expressao:
ref produto int local := n
co

begin int n = 4;
    begin int produto := n;
        for i to n do produto *:= i od;
        produto
    end
end
```

Panorama histórico

Linguagens:

BASIC (Beginers All-purposes Symbolic Instruction Code) - 1963 – 1964

Procedural

Criada por John Kemeny e Thomas Kurtz (não pelo Bill Gates como citam algumas fontes).

Originalmente, código de linhas numeradas e subrotinas chamadas por linha (GOTO e GOSUB)

Panorama histórico

Linguagens:

BASIC (Exemplo)

```
10 REM Fatorial em BASIC.
100 INPUT "X"; X
110 GOSUB 200
120 PRINT "Fatorial de X = "; F1
130 GOTO 100
200 IF (X < 0) THEN 999
210 IF (X > 0) THEN 300
220 F1 = 1
230 RETURN
300 F1 = 1
310 FOR I% = 1 TO X
320 F1 = F1 * X
330 X = X - 1
340 NEXT I%
350 RETURN
999 END
```

Panorama histórico

Linguagens:

C (nome dado depois da linguagem B) - 1965 – 1973

Procedural (da linhagem da ALGOL)

Criada por Brian Kernighan e Denis Ritchie.

Linguagem destinada para programar sistemas Unix a partir do BCPL (1965) e B (1967) desenvolvidas pela AT&T foi padronizada em 1973 (ANSI C).

Conceito de blocos, bibliotecas (headers) de funções, array, pointers e casting de tipos.

É a linguagem mais utilizada até hoje.

Panorama histórico

Linguagens:

C (Exemplo)

```
/* Fatorial em C */
#include <stdlib.h>
#include <stdio.h>

int main (){
    int fatorial=1;
    int n=8;
    int i;
    for(i=n;i>0;i--){
     fatorial = fatorial * i;
    }
    printf("fatorial de %d eh %d\n",n,fatorial);
exit (0);
}
```

Panorama histórico

Linguagens:

HASKELL (nome em homenagem a Haskell Curry) - 1990 – 1998

Funcional (diretamente derivada da ML)

Criada pela Universidade de Glasgow.

Linguagem puramente funcional e baseada em calculo lambda tipado.

Panorama histórico

Linguagens:

HASKELL (Exemplo)

```
-- Fatorial em Haskell
module Main () where

main = print (fact 20)

fact 0 = 1
fact n = n * fact (n-1)
```

Panorama histórico

Linguagens: Décadas de 1970 – 2000

Várias linguagens derivadas das anteriores:

Imperativas: Forth - Modula 2 - Perl - PHP - Javascript - C# ...

OOP: Smaltalk - ADA - C++ - Eiffel - Python - Ruby - Java ...

Declarativas: Prolog - SQL - HTML - UML - XML - Scheme - ML - Miranda - Haskell - O'Haskell (OOP) - Clean - CAML - OCAML (OOP) - UML ..

Panorama histórico

Objetivo → Gerar o mesmo código:

```
%include        'system.inc'

section .text
global  _start
_start:
  mov   cx, 8           ; CX (contador)recebe o valor 8. A
  mov   ax, 1           ; instrucao mul usa o par DX:AX de
  mov   dx, 0           ; registros, entao usamos estes registros
                        ; para armazenar o fatorial (1 para evitar
                        ; multiplicar por 0).
meuloop:
  mul   cx              ; DX:AX = AX * CX
  dec   cx              ; decrementa o contador
  cmp   cx, 0           ; compara o contador com 0
  jne   meuloop         ; novo loop se o contador nao eh zero

  push ax               ; coloca ax na pilha
  push dx               ; coloca dx na pilha
  push dword stdout     ; coloca 1 (dword stdout) na pilha
  sys.write             ; escreve o conteudo da pilha
  push dword 0          ; sai com sucesso (0)
  sys.exit
```