



## **Estrutura do tema ISC**

1. Representação de informação num computador
2. Organização e estrutura interna dum computador
- 3. Execução de programas num computador**
4. Análise das instruções de um processador
5. Evolução da tecnologia e da eficiência

## Representação de comandos/instruções num computador



```
int x = x+y;
```

```
addl 8(%ebp), %eax
```

Idêntico à expressão

```
x = x + y
```

```
0x401046: 03 45 08
```

- Código C
  - somar 2 inteiros (c/ sinal)
- Assembly (da GNU p/ IA-32)
  - somar 2 inteiros de 4 *bytes*
    - operandos “*long*” em GCC
    - a mesma instrução, c/ ou s/ sinal
  - operandos:
    - x: em registo %eax
    - y: na memória M[%ebp+8]
- Código *object* em IA-32
  - instrução com 3 *bytes*
  - na memória a partir do endereço 0x401046

# Níveis de abstração na representação de programas num computador



## Níveis de abstração:

Slide anterior:

- Código C

- somar 2 inteiros (c/ sinal)

```
int x = x+y;
```

- Assembly (da GNU p/ IA-32)

- somar 2 inteiros de 4 bytes

```
addl 8(%ebp), %eax
```

Idêntico à expressão

**x = x + y**

- Código *object* em IA-32

- instrução com 3 bytes

- na memória a partir do endereço

0x401046

```
0x401046: 03 45 08
```

- nível das linguagens HLL (*High Level Languages*):  
as linguagens convencionais de programação (puro texto)
  - » imperativas e OO (Basic, Fortran, C/C++, Java, ...)
  - » funcionais (Lisp, Haskell, ...)
  - » lógicas (Prolog, ...)
- nível da linguagem *assembly* (de “montagem”):  
uma linguagem intermédia (comandos da PU em formato texto)
- nível da linguagem máquina: a linguagem de comandos,  
específica p/ cada PU ou família de PU's (em binário puro)
  - » arquiteturas CISC (*Complex Instruction Set Computers*)
  - » arquiteturas RISC (*Reduced Instruction Set Computers*)

## **Execução de instruções** (em linguagem máquina) *numa PU*



### **Ciclo de execução de instruções:**

- 1. Leitura** de uma instrução da memória  
... e incremento do IP
- 2. Descodificação** da instrução
- 3. Execução** da operação
  - cálculo da localização do(s) operando(s),  
e ir buscá-lo(s), se necessário
  - execução da ação especificada
  - guardar resultado, se necessário

**Análise de um exemplo:** `movl Mem_Loc,%eax`

**Mecanismos de conversão entre níveis de abstração**

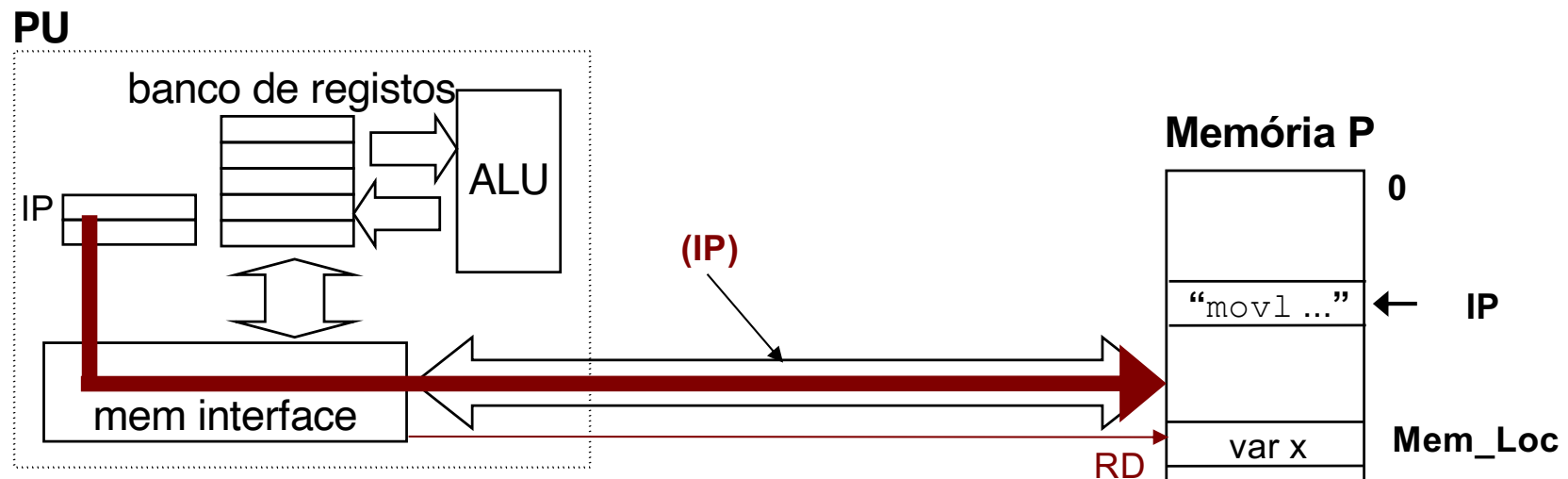
**Modelo de computação de von Neumann (1945)**

## Exemplo de execução de uma instrução em linguagem máquina (1)



Ex.: `movl Mem_Loc,%eax`

### 1. Leitura da instrução na memória (1)



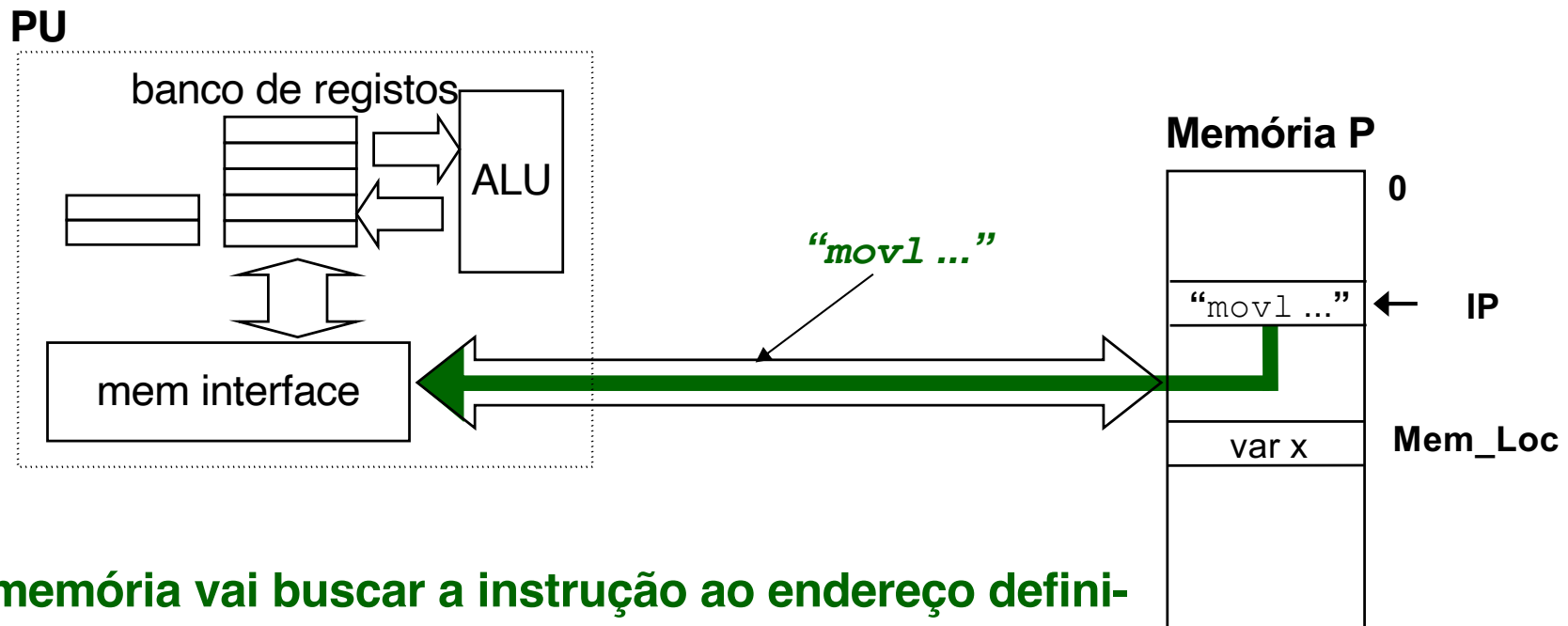
**A PU coloca no *address bus* o valor em IP (endereço p/ próxima instrução), e ativa o sinal de controlo RD**

## Exemplo de execução de uma instrução em linguagem máquina (2)



Ex.: `movl Mem_Loc, %eax`

### 1. Leitura da instrução na memória (2)



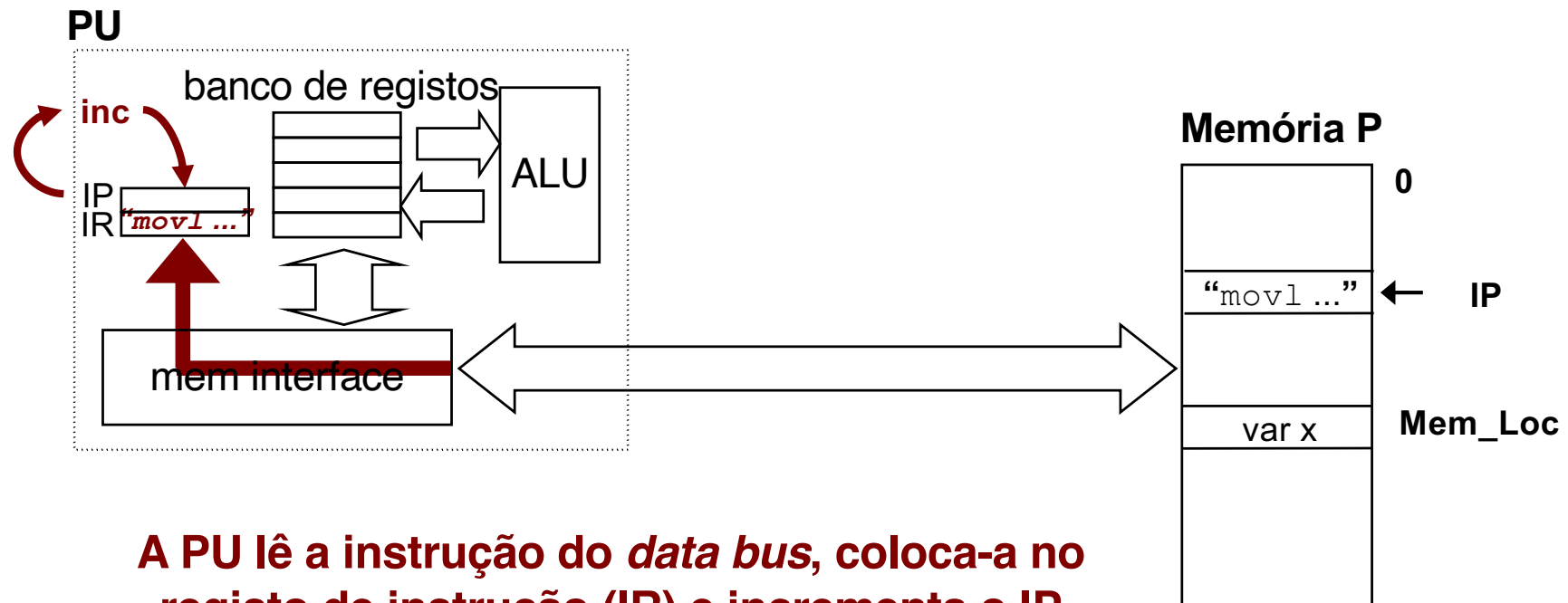
A memória vai buscar a instrução ao endereço definido por IP e coloca-a no *data bus* p/ ser lida pela PU

## Exemplo de execução de uma instrução em linguagem máquina (3)



Ex.: `movl Mem_Loc, %eax`

### 1. Leitura da instrução na memória (3) ... e incremento do IP



**A PU lê a instrução do *data bus*, coloca-a no registo de instrução (IR) e incrementa o IP**

## Exemplo de execução de uma instrução em linguagem máquina (4)



Ex.: `movl Mem_Loc,%eax`

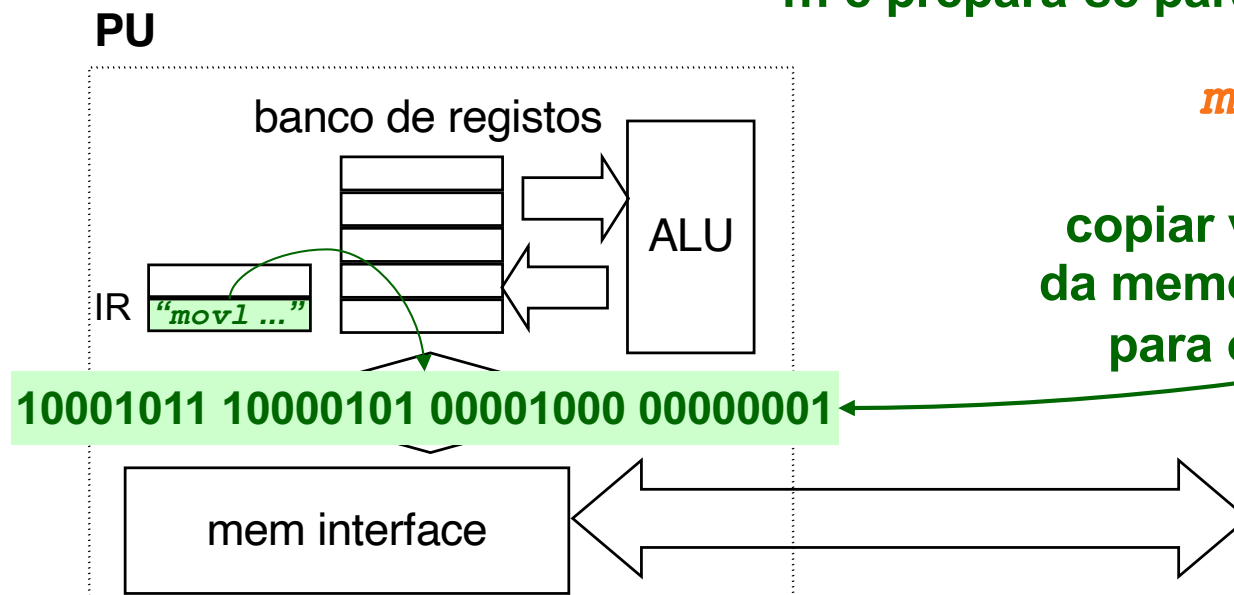
### 2. Descodificação da instrução

A unidade de controlo da PU descodifica a instrução...

... e prepara-se para executar a operação:

*move long*

copiar valor com 32 bits  
da memória, em **Mem\_Loc**  
para o registo **%eax**



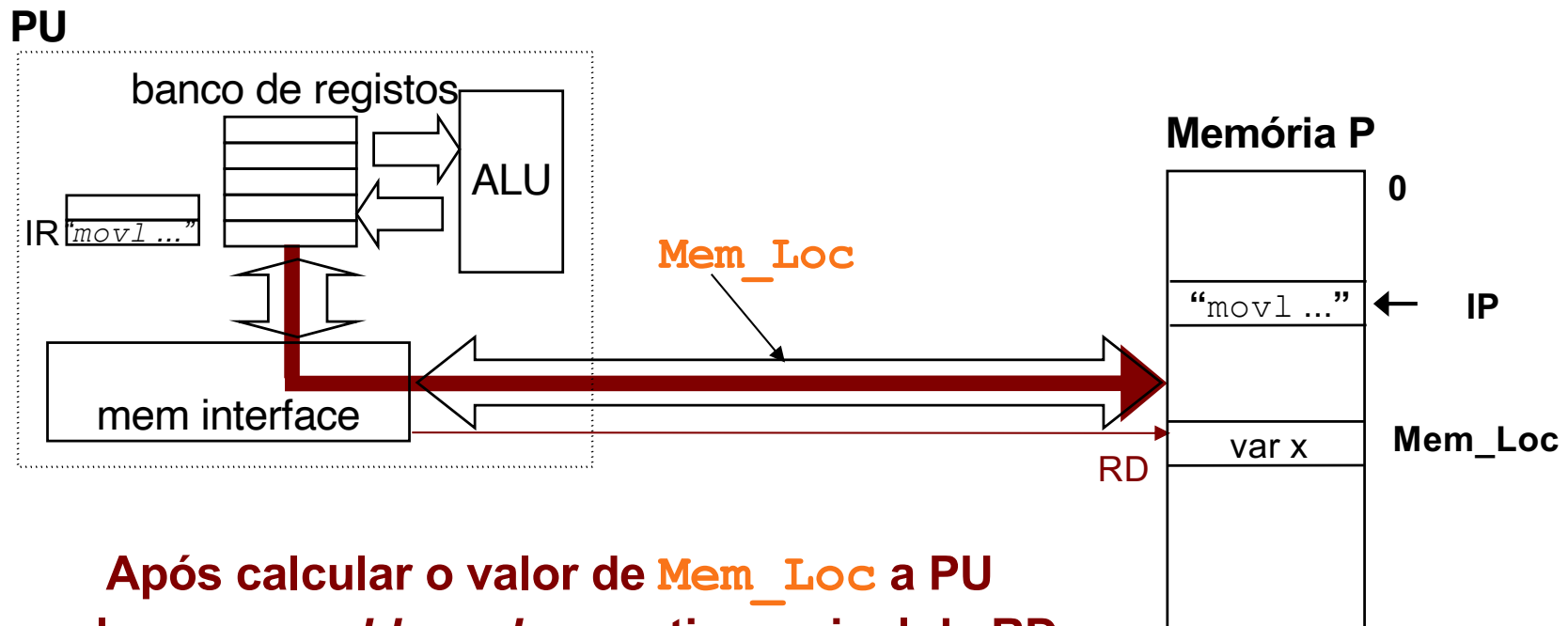


## Exemplo de execução de uma instrução em linguagem máquina (5)



Ex.: `movl Mem_Loc, %eax`

### 3. Execução da operação (1)



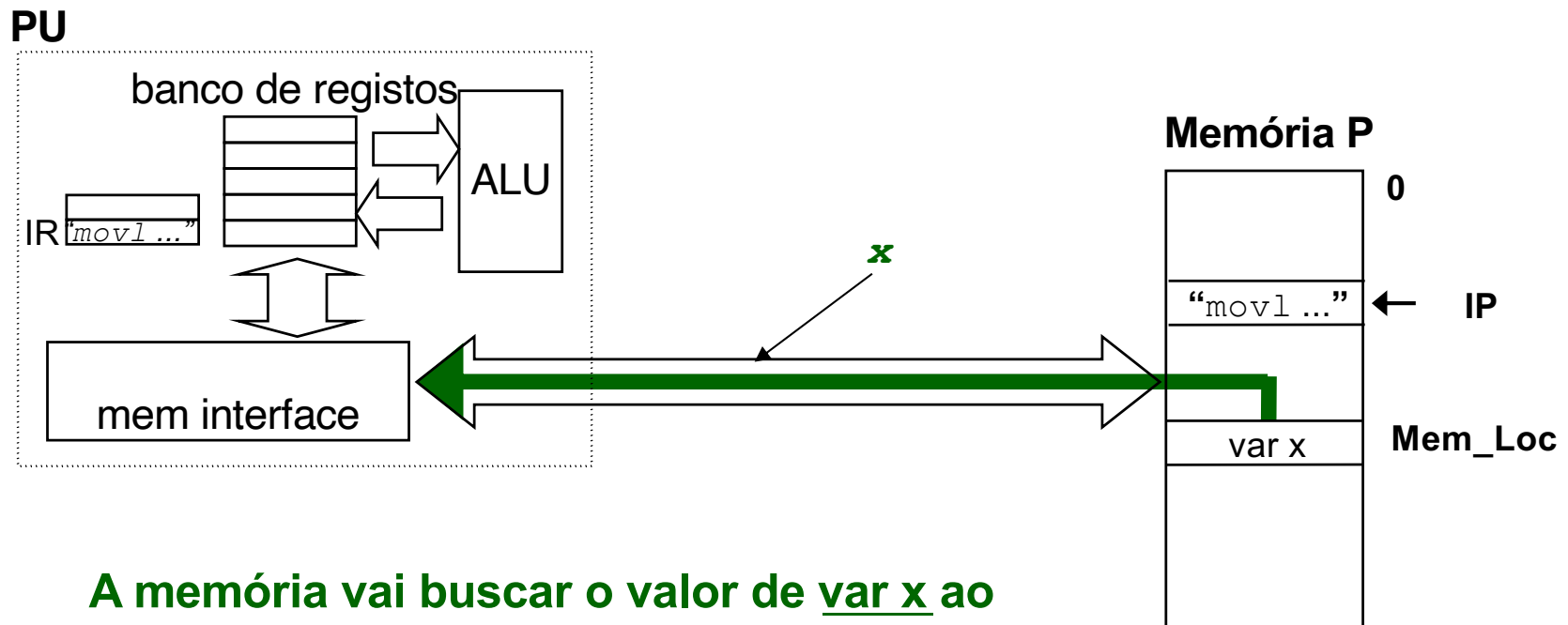
**Após calcular o valor de `Mem_Loc` a PU coloca-o no *address bus* e ativa o sinal de RD**

## Exemplo de execução de uma instrução em linguagem máquina (6)



Ex.: `movl Mem_Loc, %eax`

### 3. Execução da operação (2)



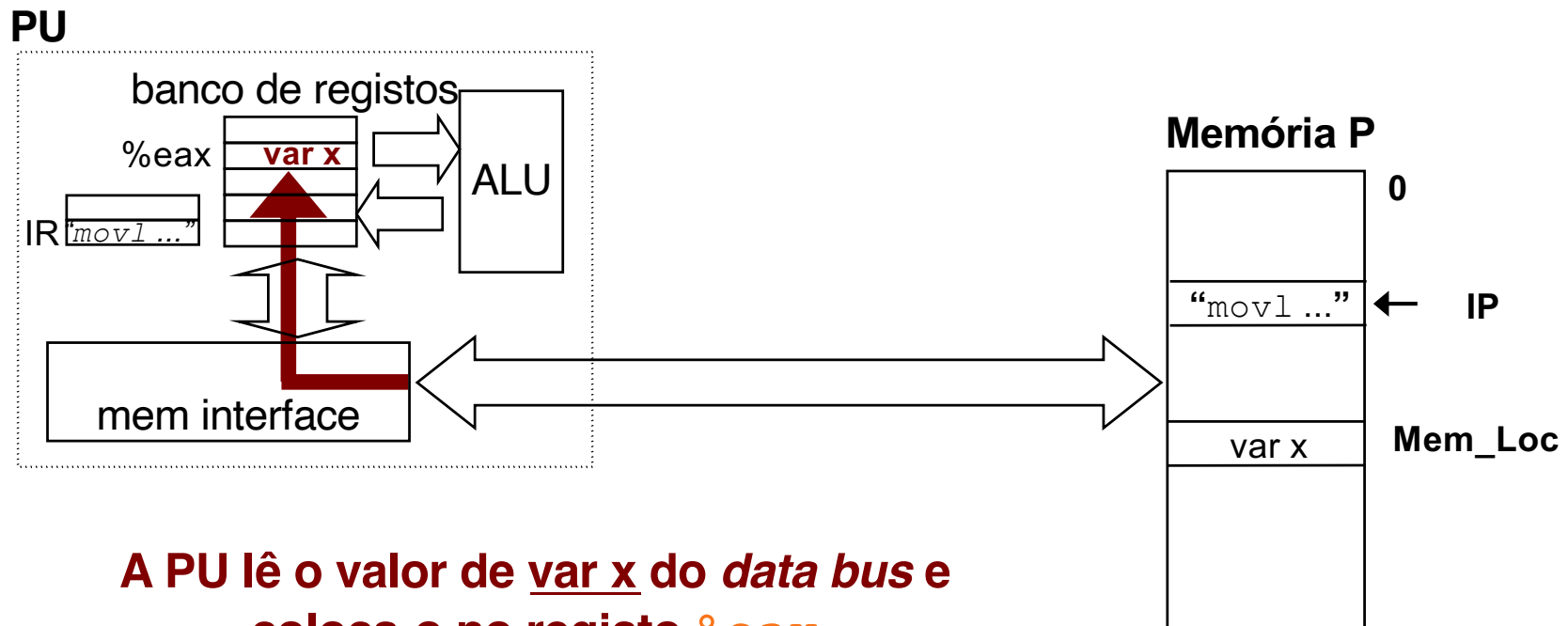
A memória vai buscar o valor de var x ao endereço **Mem\_Loc** e coloca-o no *data bus*

## Exemplo de execução de uma instrução em linguagem máquina (7)



Ex.: `movl Mem_Loc, %eax`

### 3. Execução da operação (3)



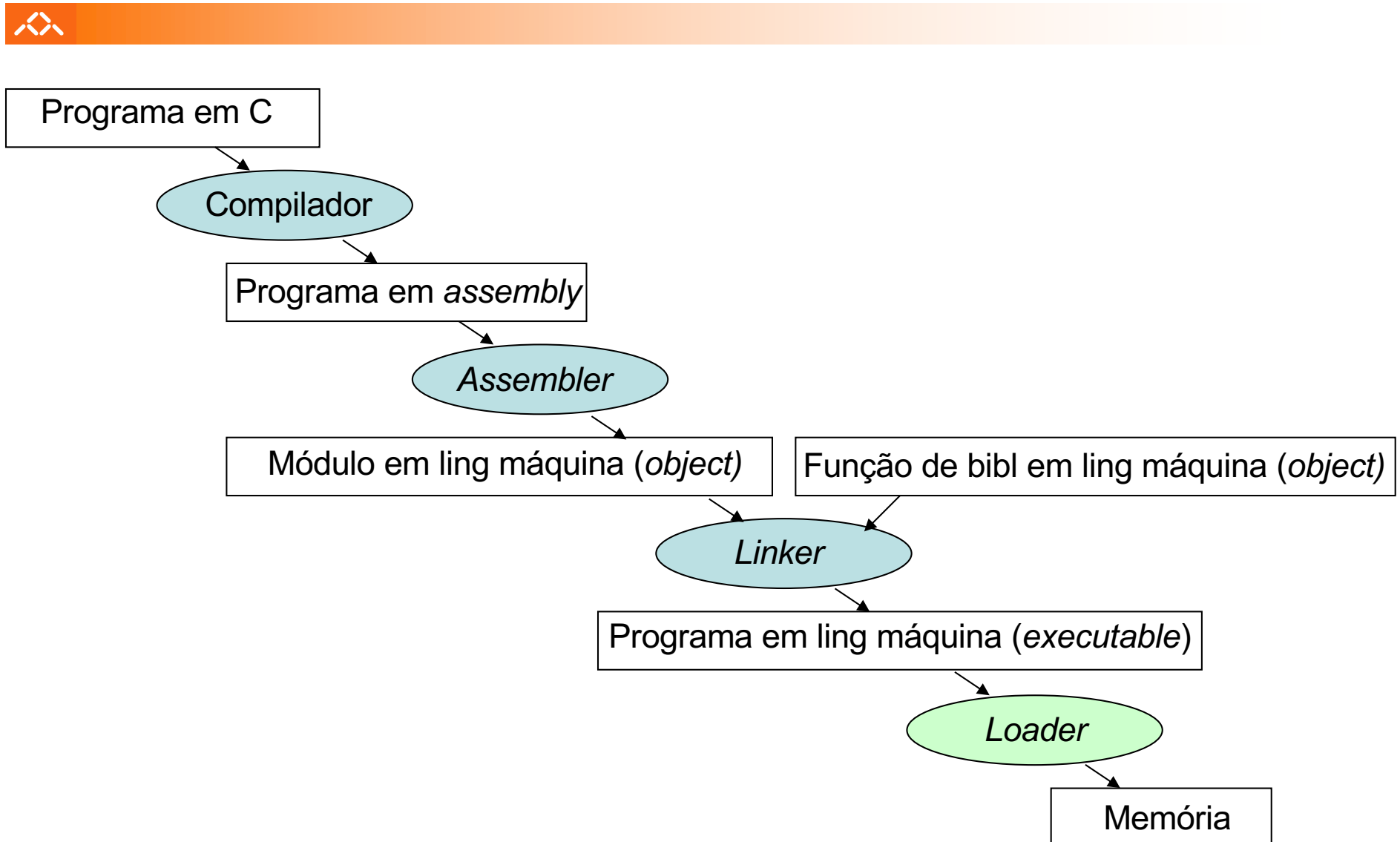
## Execução de programas num computador: de HLL para linguagem máquina



### Mecanismos de conversão (para comandos da PU):

- compilador
  - traduz um programa de um nível de abstração para outro inferior (converte um ficheiro de texto noutra de texto);  
por ex., de C para *assembly*
  - normalmente inclui mais que um passo de conversão, até chegar à linguagem máquina
- *assembler* (“montador”)
  - “monta” os comandos / instruções, em texto, para binário (*object*), de acordo com as regras do fabricante da PU
- interpretador
  - analisa, uma a uma, as instruções de um programa em HLL, e:
    - » gera código em linguagem máquina para essa instrução, e
    - » executa esse código (nota: não guarda o código gerado)

## Execução de programas num computador: de HLL até à sua execução



## Modelo de computação de von Neumann, 1945/46 (1)



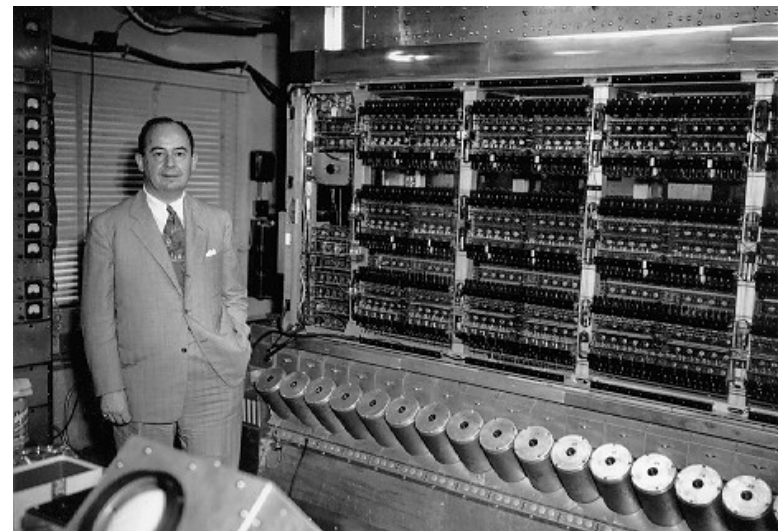
### ENIAC (1ª geração, 1945)

- objetivo: cálculo tabelas de artilharia (mas 1º teste foi p/ bomba H)
- máquina **decimal** (base 10)
- 17.468 válvulas, 27 toneladas
- programação: manual, alterando as conexões (cablagem)



### Von Neumann introduz o conceito de **stored-program**

- dados e instruções em **binário**, e armazenados numa memória
- memória acedida pelo endereço da informação
- execução de instruções de modo sequencial (daí o *Program Counter*, PC), interpretadas pela unidade de controlo
- constrói novo computador, o IAS



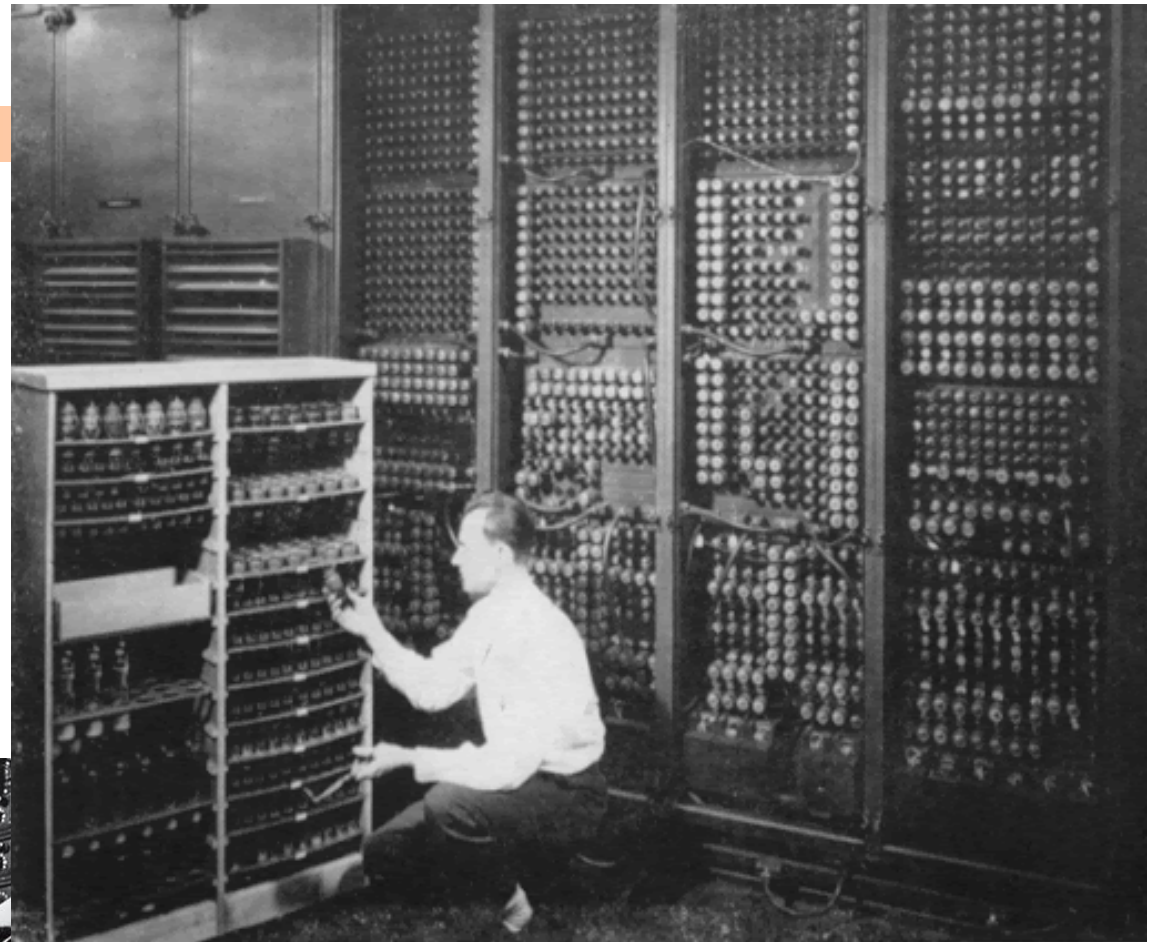


# A eletrônica dos primeiros computadores

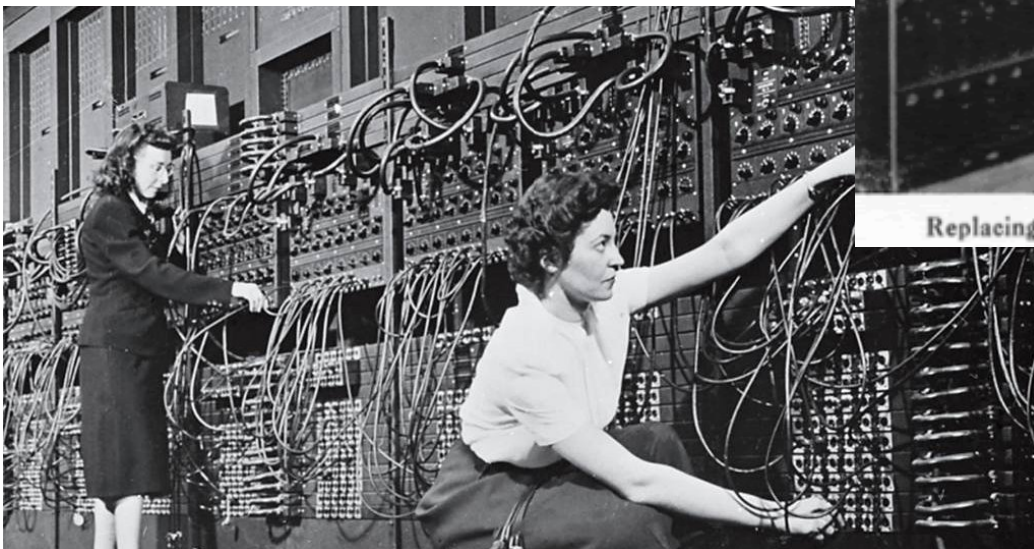
ENIAC



Válvulas num computador

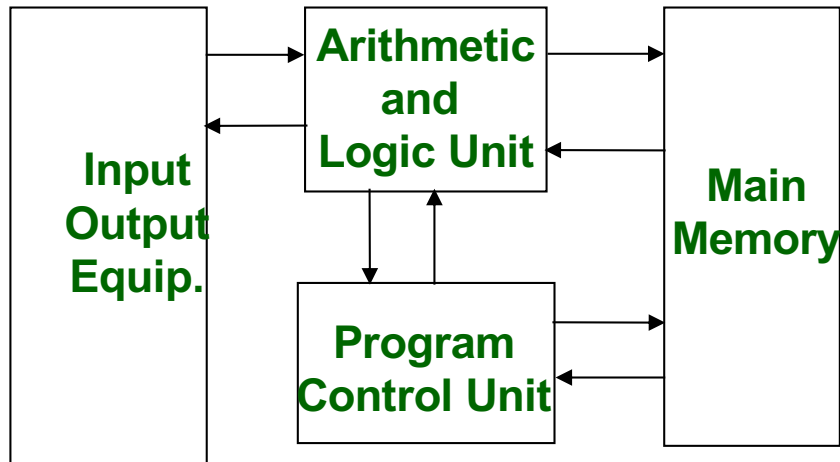


Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.



Interligando os diversos componentes por “hardware”...

# Modelo de computação de von Neumann, 1945/46 (2)



**Estrutura básica do IAS**  
(Princeton Institute for Advanced Studies)

**Estrutura expandida do IAS**

