

Infraestrutura de Hardware

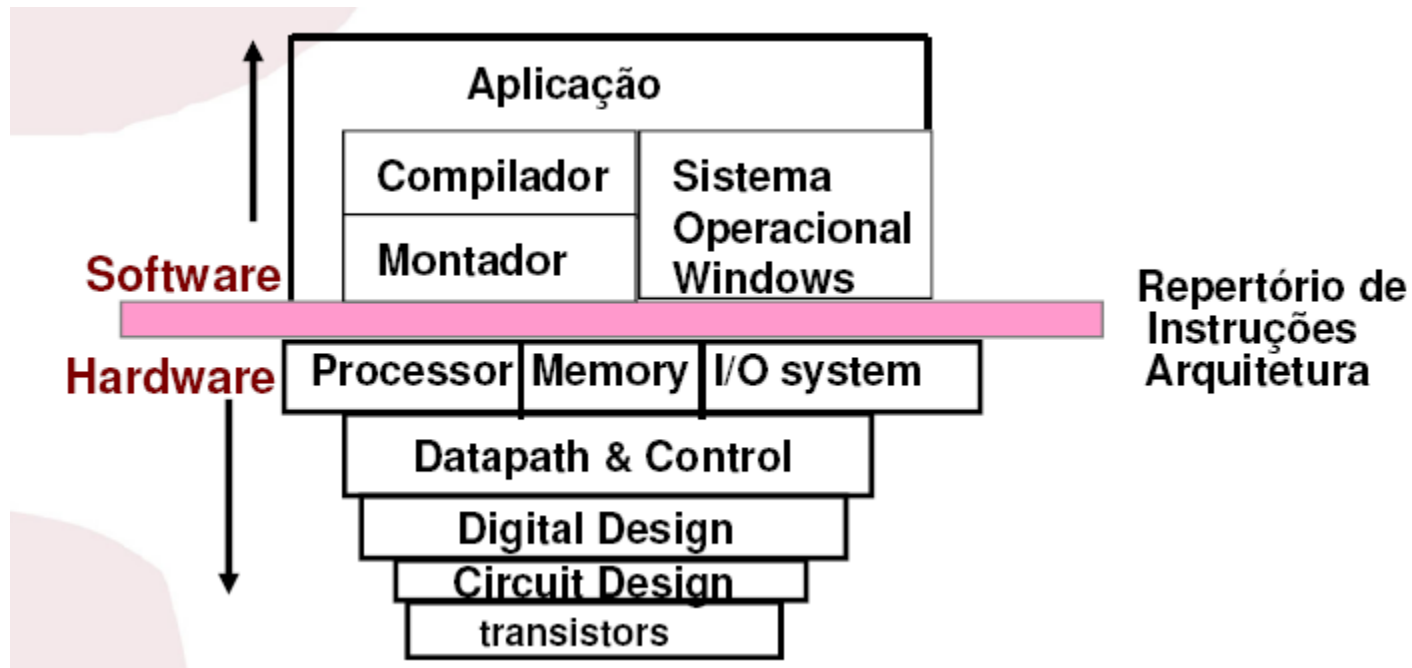
Funcionamento de um Computador



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

CIn.ufpe.br

Computador: Hardware + Software



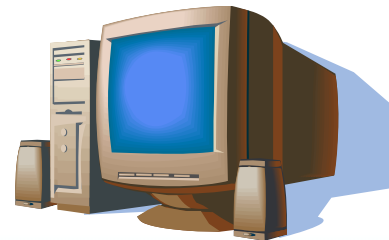
Perguntas que Devem ser Respondidas ao Final do Curso

- **Como um programa escrito em uma linguagem de alto nível é entendido e executado pelo HW?**
- Qual é a interface entre SW e HW e como o SW instrui o HW a executar o que foi planejado?
- O que determina o desempenho de um programa e como ele pode ser melhorado?
- Que técnicas um projetista de HW pode utilizar para melhorar o desempenho?

Que Linguagem o HW entende?

- HW entende sinais elétricos
- Alfabeto da linguagem entendida por HW possui dois valores:
 - Ligado (On), Desligado (Off)
 - Ou 0 e 1 (números binários)
- Instruções para o computador são sequências de números binários

10010010
10001110

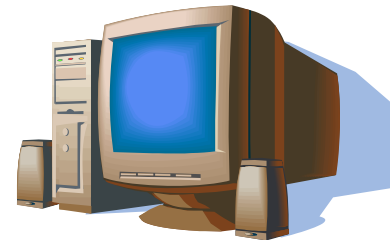


Abstraindo a Linguagem de Máquina

- Escrever um programa em linguagem de máquina é impraticável!



10010010
10001110

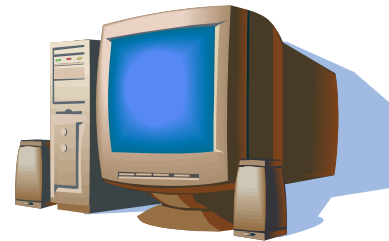


- Conceitos de HW foram abstraídos para que ser humano pudesse instruir o computador
- Criação de linguagens de programação

Linguagens de Programação

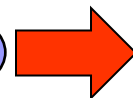
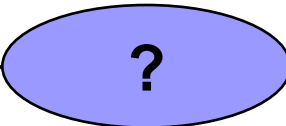
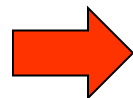
- Os programas têm que ser escritos em uma linguagem de programação:
que possa ser entendida pelo computador

```
10010010  
10001110
```

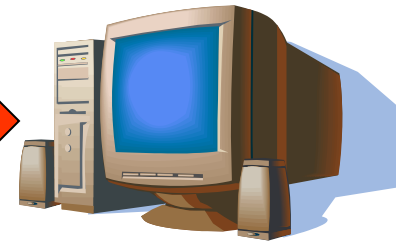
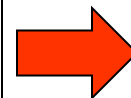


que possa ser **traduzida** para a linguagem entendida pelo computador

```
a = 10;  
a = a + 1;
```



```
10010010  
10001110
```



Níveis de Abstração de Linguagens

- Linguagens de programação variam de acordo com o seu nível de abstração

↑ conhecimento da máquina onde programa será executado
↓ nível de abstração

↓ conhecimento da máquina onde programa será executado
↑ nível de abstração

- Podem ser classificadas em 4 níveis:

Linguagem de máquina

Linguagem de montagem (assembly)

Linguagem de alto nível (Java, C, Pascal, C++, etc)

Linguagem de 4ª geração (PL/SQL, NATURAL, MATLAB, etc)

Níveis de Abstração de Linguagens

- Linguagem assembly é dependente da máquina, porém utiliza palavras reservadas para codificar instruções (mnemônicos)

Mnemônico

```
swap:
    muli $2, $5, 4
    add  $2, $4, $2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```

- Outros níveis são independentes de máquina e facilitam leitura e escrita dos programas por parte do ser humano

Complexidade atual de programas exigem cada vez mais o emprego destas linguagens

Como o Computador Entende um Programa?

- Deve-se traduzir um programa para a linguagem de máquina
- Um compilador é um programa que traduz um programa escrito (código fonte) em uma determinada linguagem de programação para outra linguagem (linguagem destino)
Se a linguagem destino for a de máquina, o programa pode, depois de compilado, ser executado
- Um interpretador é um programa que traduz instrução por instrução de um programa em linguagem de máquina e imediatamente executa a instrução

Compilação x Interpretação

Compilação

Código- fonte

```
a = 10;  
a = a + 1;
```

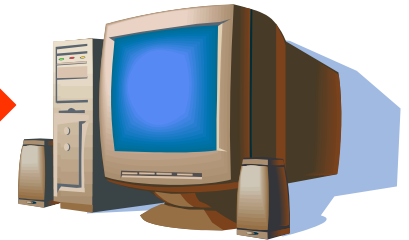


Compilador



Código de máquina

```
10010010  
10001110
```



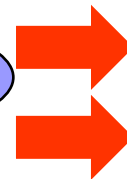
Interpretação

Código- fonte

```
a = 10;  
a = a + 1;
```

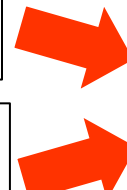


Interpretador



Código de máquina

```
10010010  
10001110
```



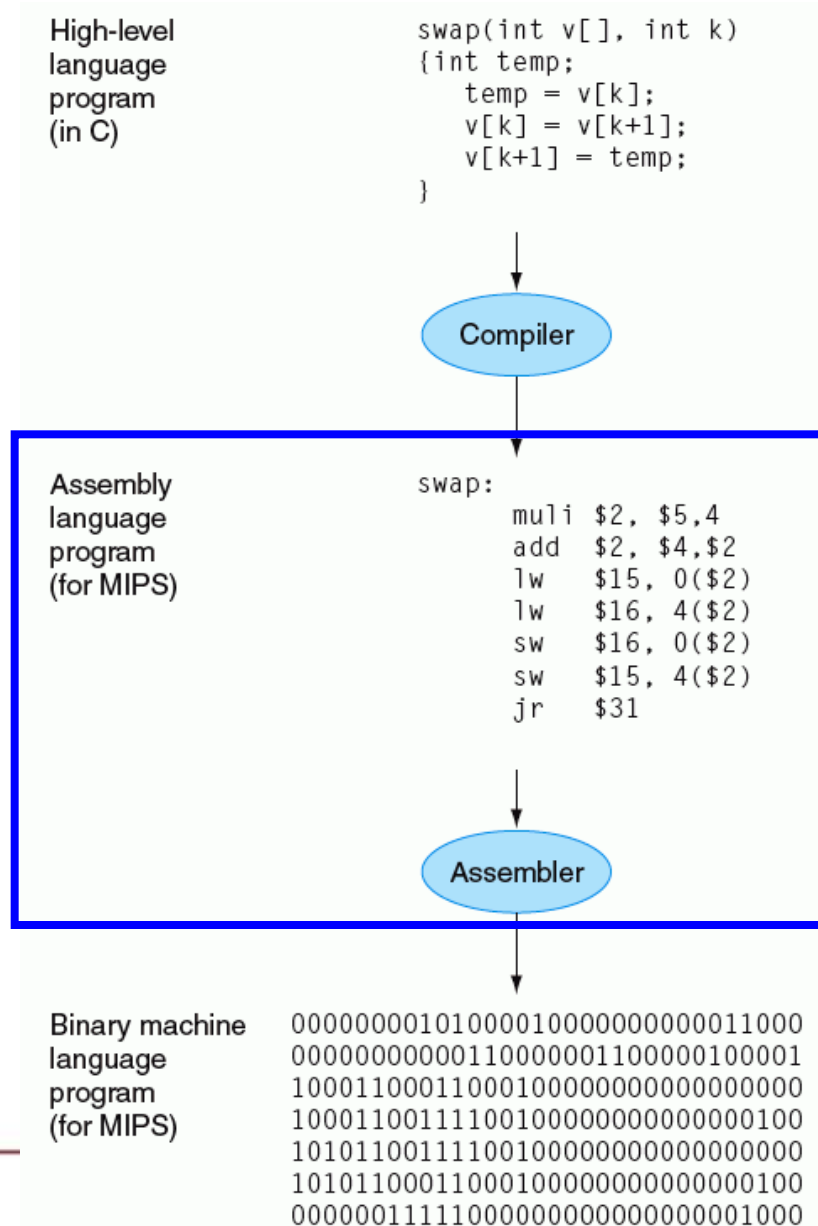
Compilação x Interpretação

- Existem vários exemplos tanto de linguagens interpretadas como de linguagens compiladas
- A linguagem C é um exemplo de linguagem compilada
- 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

Interpretador Java específico da máquina onde irá rodar o programa então traduz os bytecodes para linguagem de máquina e executa o código

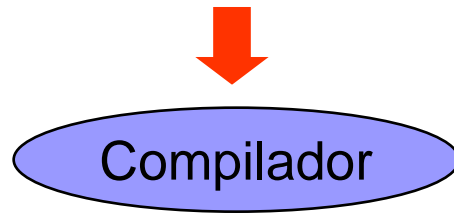
Exemplo de Compilação em 2 etapas



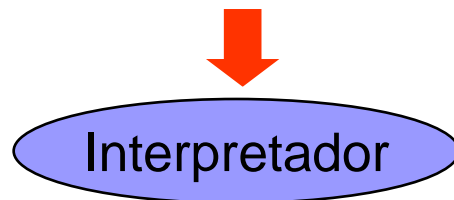
Maioria dos compiladores C omitem esta parte. Compilam diretamente para linguagem de máquina

Exemplo de Compilação e Interpretação

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```



```
0 getstatic #16 <java/lang/System/out  
  Ljava/io/PrintStream;>  
3 ldc #22 <Hello World>  
5 invokevirtual #24  
  <java/io/PrintStream/println(Ljava/lang/String;)V>  
8 return
```



00000000101000010000000000011000

Bytecodes

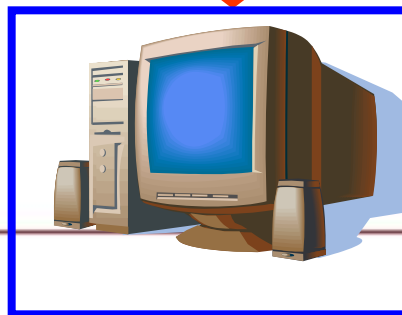
Compilação em SW e Interpretação em HW

```
swap(int v[], int k)
{
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

Compilador

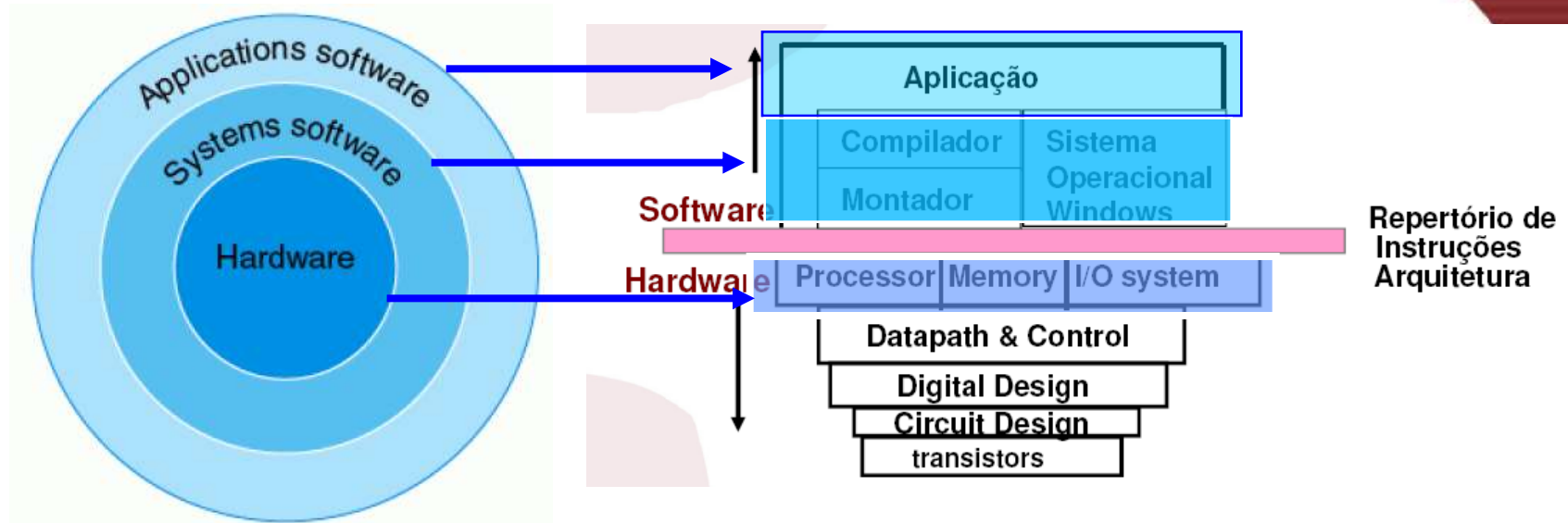
```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

HW interpreta
instrução a
instrução



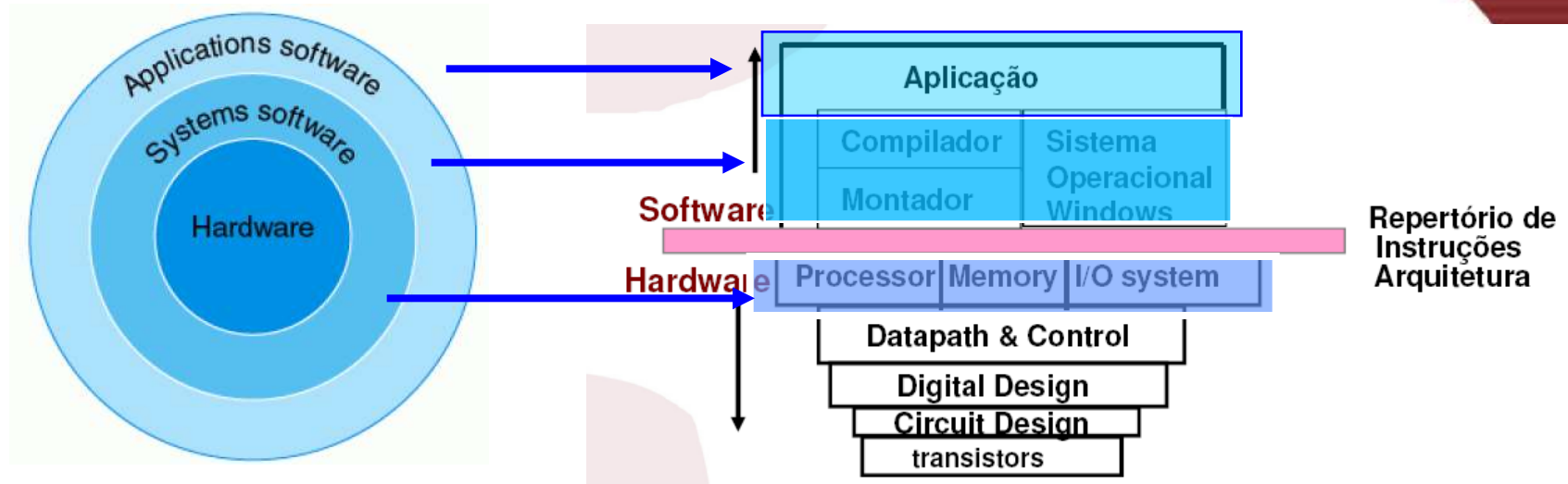
```
000000001010000100000000000011000
```

Abstrações de um Computador



- Faz-se necessário a criação de camadas de abstrações que escondam detalhes de implementação de um computador para desenvolver as aplicações atuais cada vez mais complexas

Abstrações de Software



- Aplicação: abstração de dados, armazenamento, procedural
- Softwares de sistema

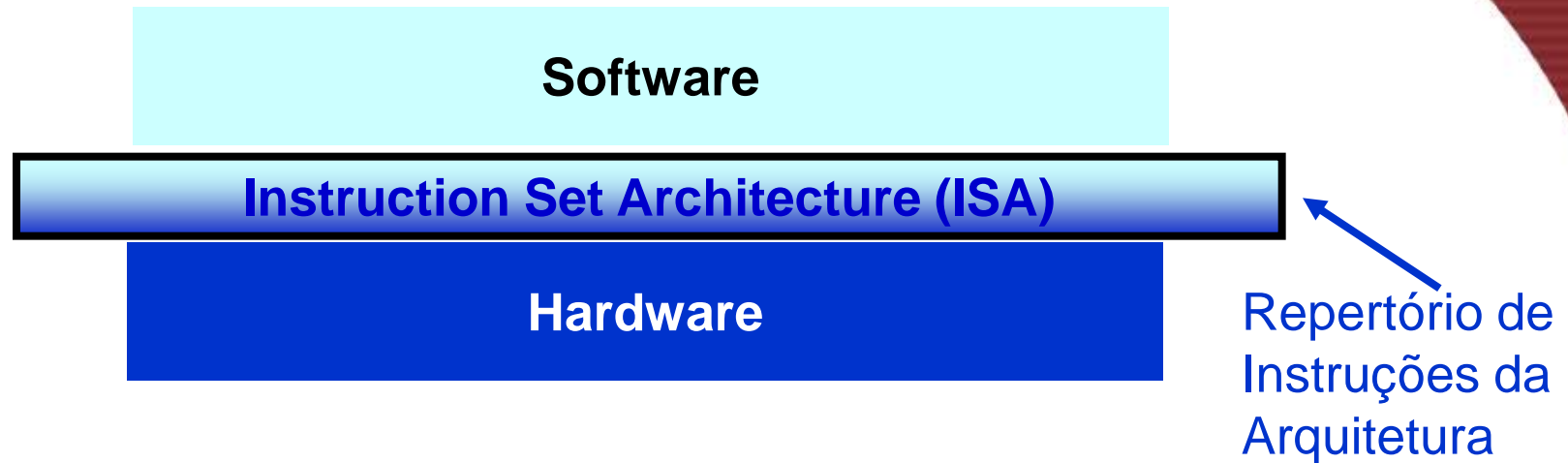
Compiladores: abstração do repertório de instruções da máquina

Sistema Operacional: abstração de concorrência, recursos de HW, hierarquia de memória

Perguntas que Devem ser Respondidas ao Final do Curso

- Como um programa escrito em uma linguagem de alto nível é entendido e executado pelo HW?
- **Qual é a interface entre SW e HW e como o SW instrui o HW a executar o que foi planejado?**
- O que determina o desempenho de um programa e como ele pode ser melhorado?
- Que técnicas um projetista de HW pode utilizar para melhorar o desempenho?

Interface HW/SW: Repertório de Instruções da Arquitetura



- Última abstração do HW vista pelo SW
- 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, etc

Visão Funcional de um Computador

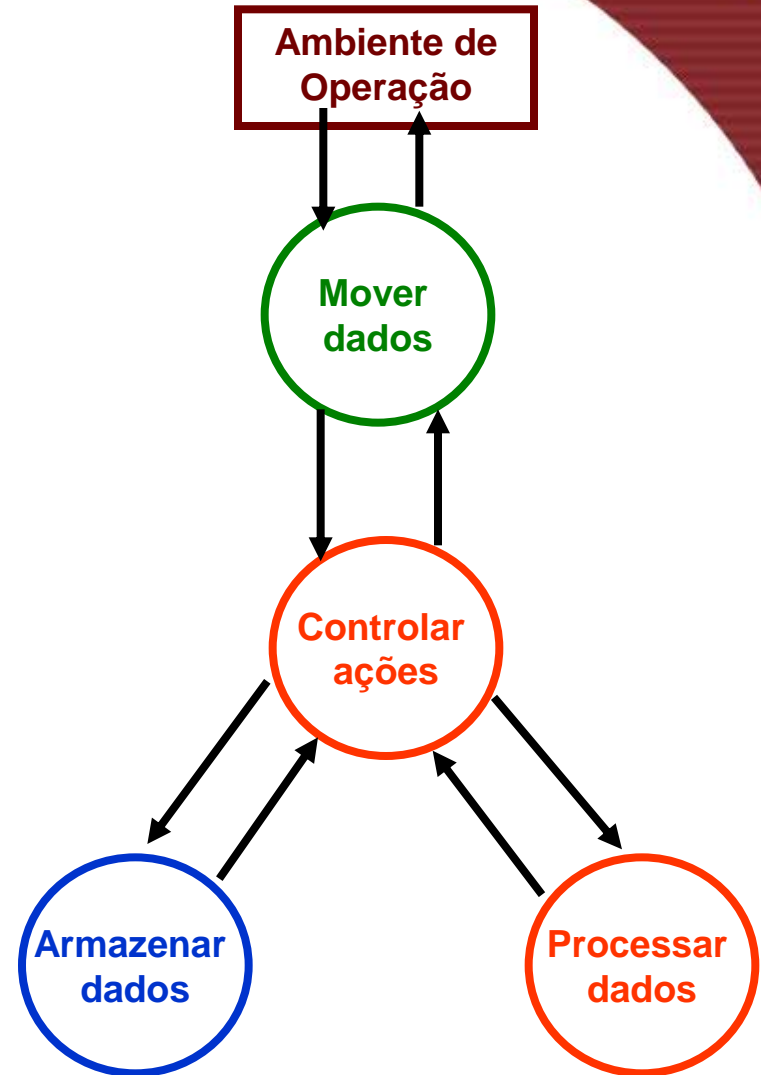
- O HW de um computador deve realizar 4 ações:

Mover dados

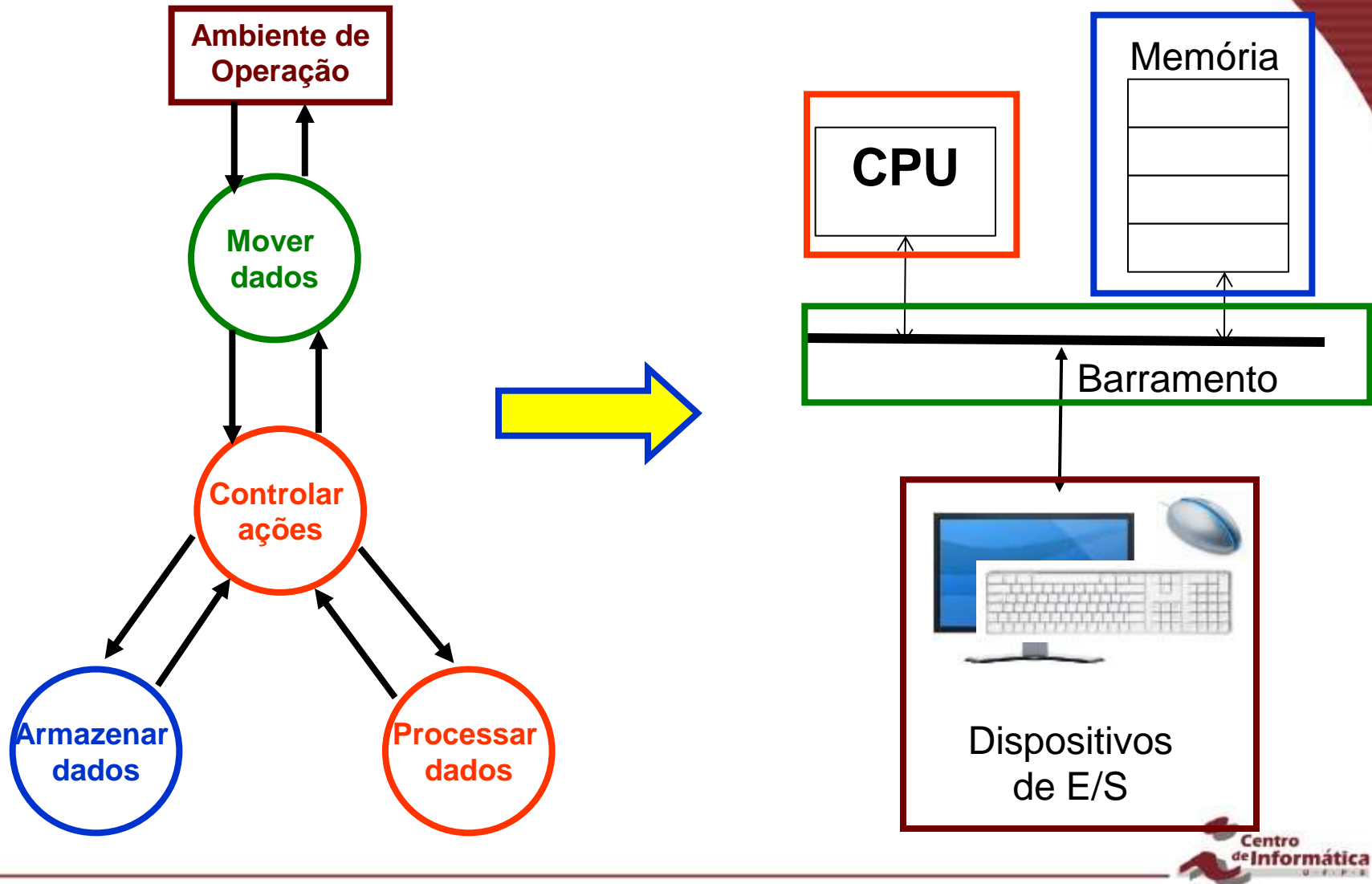
Armazenar dados

Processar dados

Controlar as ações mencionadas



Mapeando Funcionalidades em um Computador



Como Funciona um Computador?

- Conceitos básicos para funcionamento de um computador:

Dados e instruções são armazenados na memória

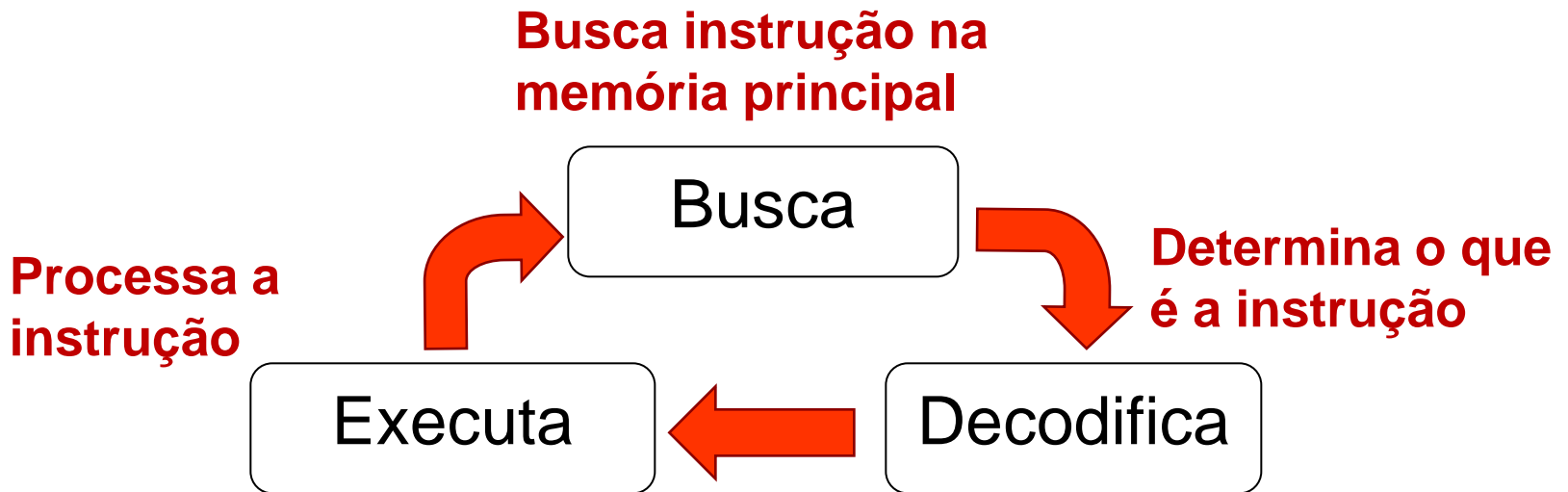
- Para simplificar, vamos considerar que é uma única memória para instruções e dados

Conteúdo da memória é acessado através de um endereço, não importando o tipo de dado armazenado

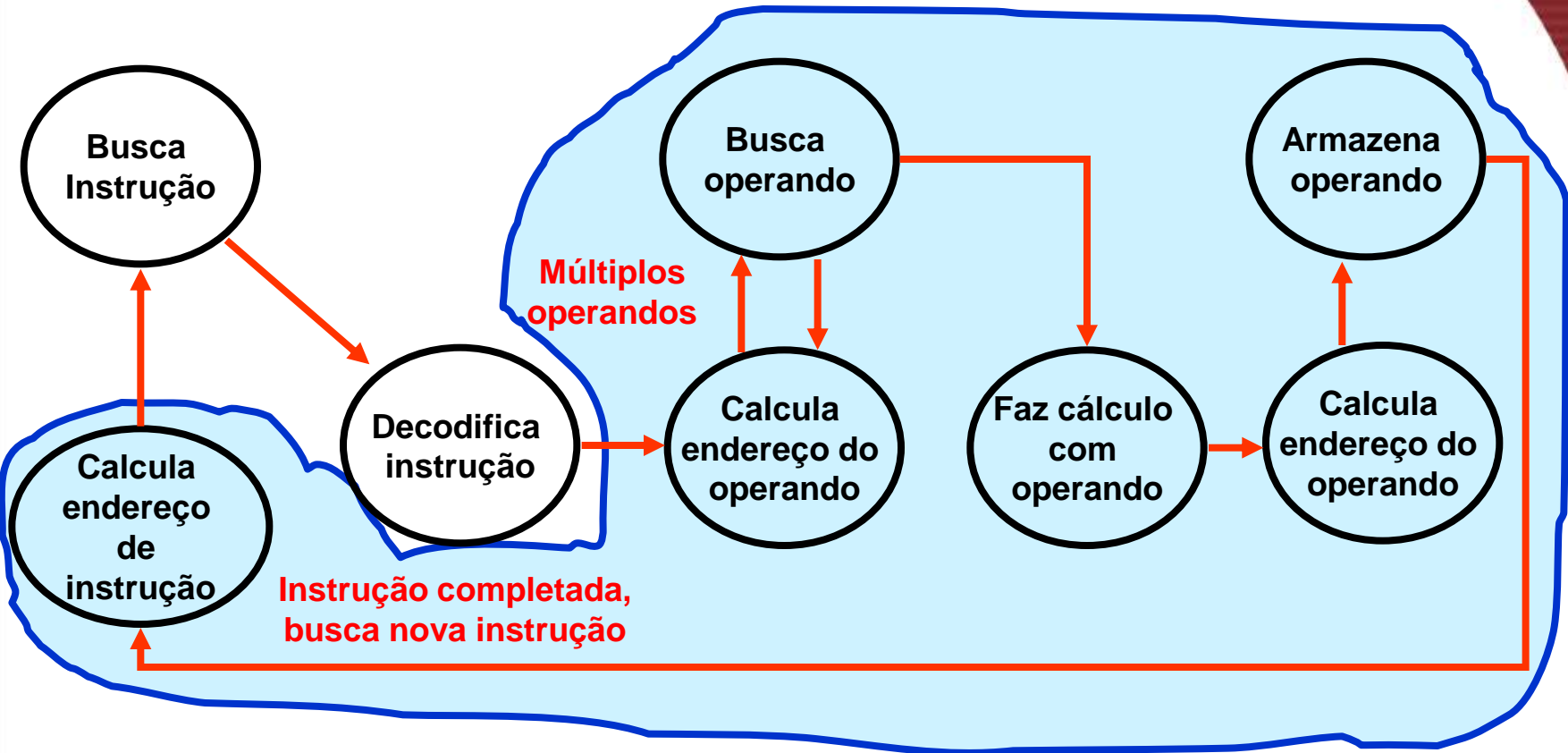
Execução ocorre de maneira sequencial (a não ser que seja explicitamente especificado), uma instrução após a outra

Visão Simplificada de Processamento de Instrução

- CPU faz continuamente 3 ações:

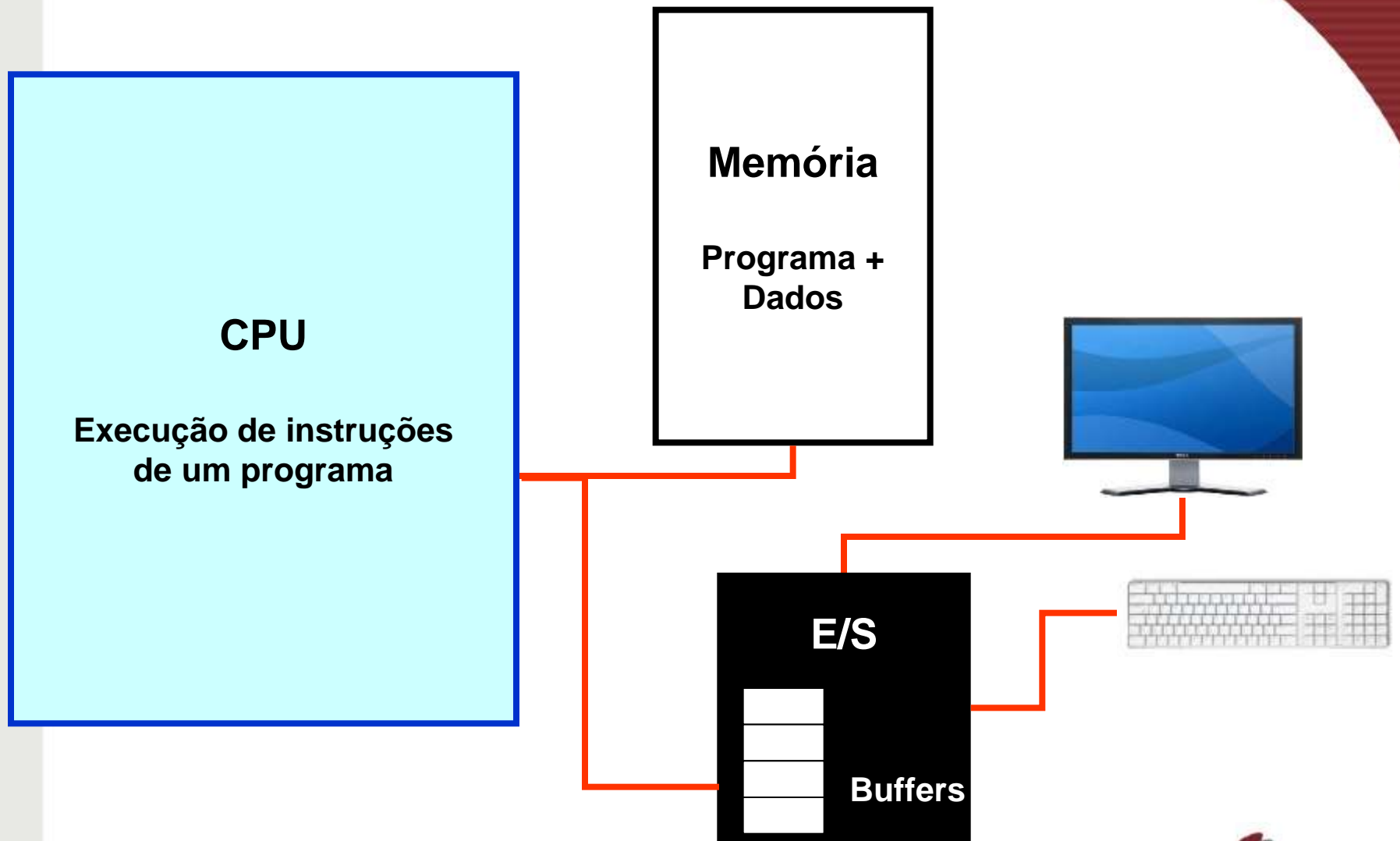


Visão Detalhada da Execução de uma Instrução

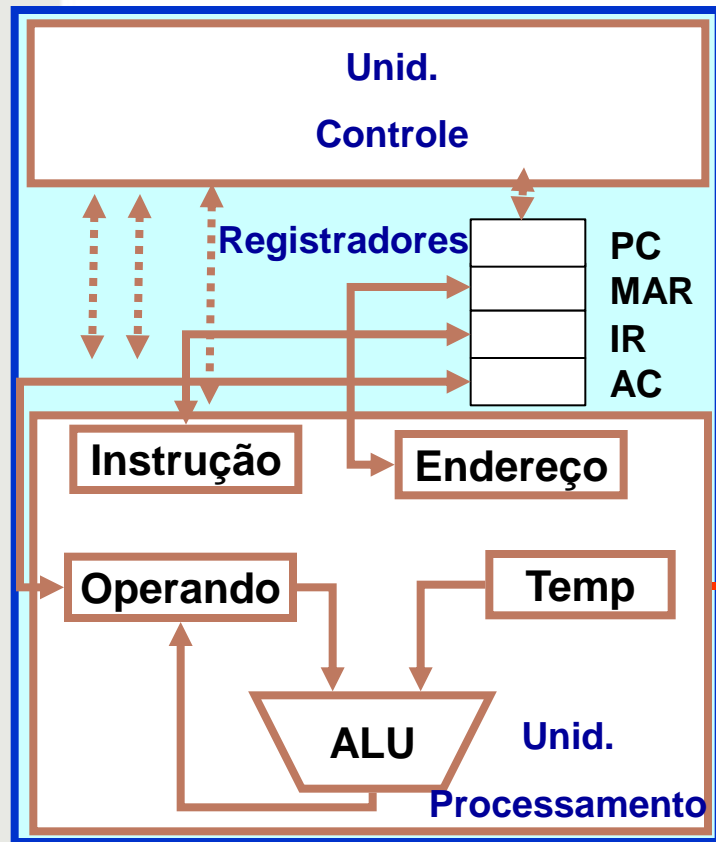


Etapa de execução de instrução

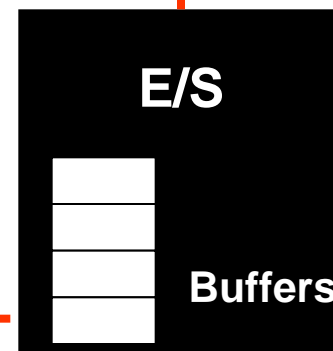
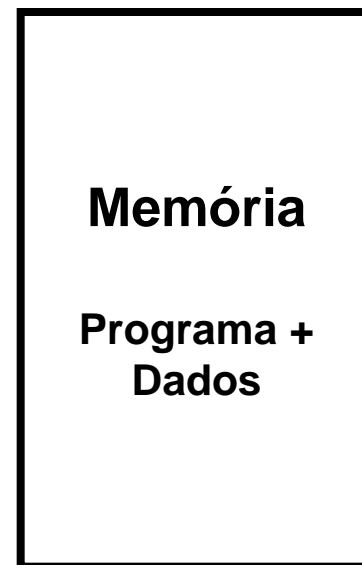
Componentes de um Computador



Mais Detalhes de uma CPU

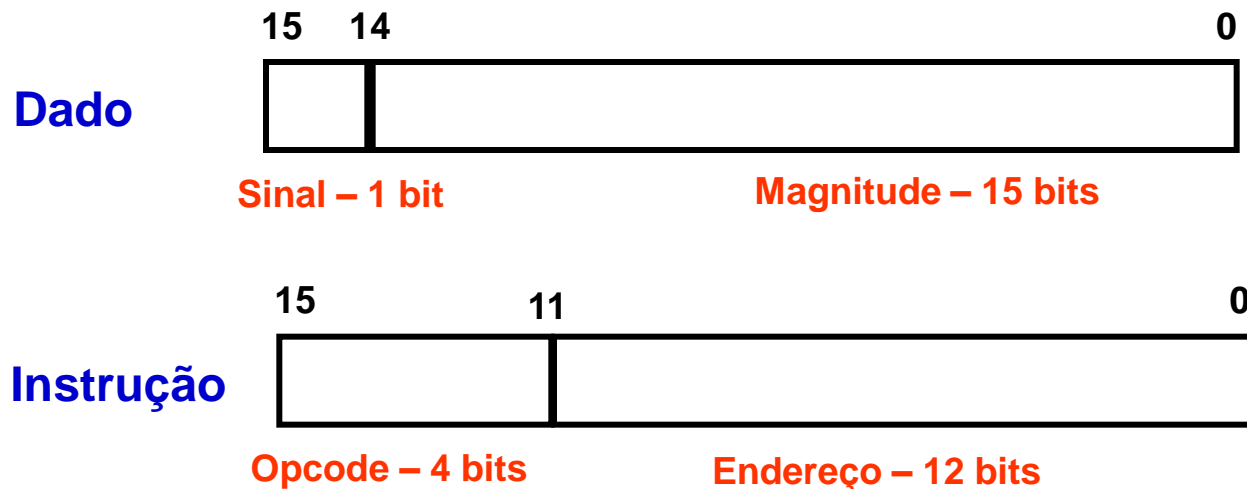


PC: Program Counter
MAR: Memory Address Register
IR: Instruction Register
AC: Accumulator



Executando um Programa em um Computador Hipotético

- Instruções e Dados ocupam 16 bits na memória
- Memória composta por **palavras** de 16 bits
- Formato de Dados e Instruções:



- $2^4 = 16$ instruções possíveis nesta arquitetura

Executando um Programa em um Computador Hipotético

- Por simplicidade, examinaremos 3 registradores

PC – Contém o endereço da instrução a ser executada

AC – Contém um operando

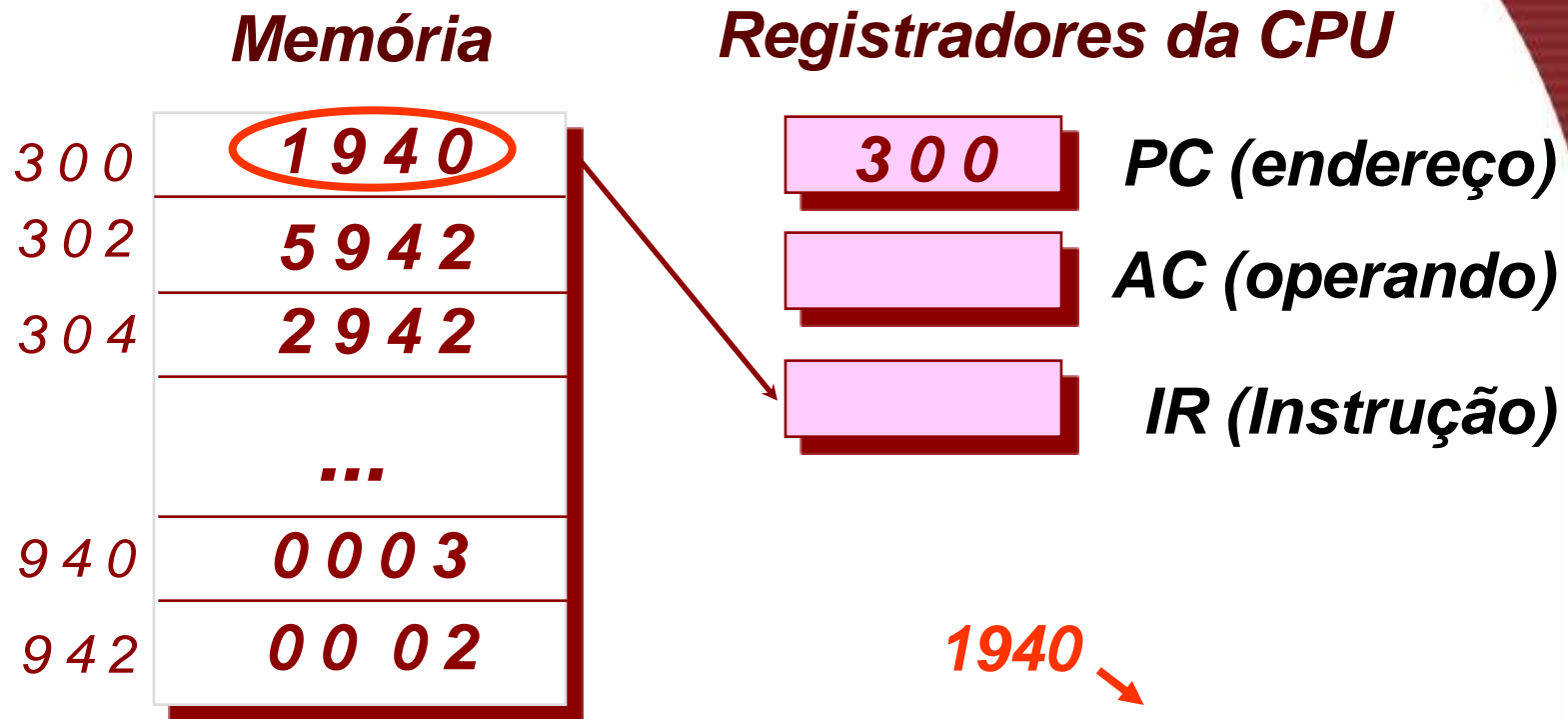
IR – Contém a instrução executada

- Repertório de Instruções

Opcode	Significado	Descrição
0001	$AC \leftarrow Mem$	Carrega em AC conteúdo de memória
0010	$Mem \leftarrow AC$	Salva na memória conteúdo de AC
0101	$AC \leftarrow AC + Mem$	Soma a AC conteúdo de memória

Passo a Passo da Execução de um Programa

Conteúdo de memória e registradores em hexadecimal

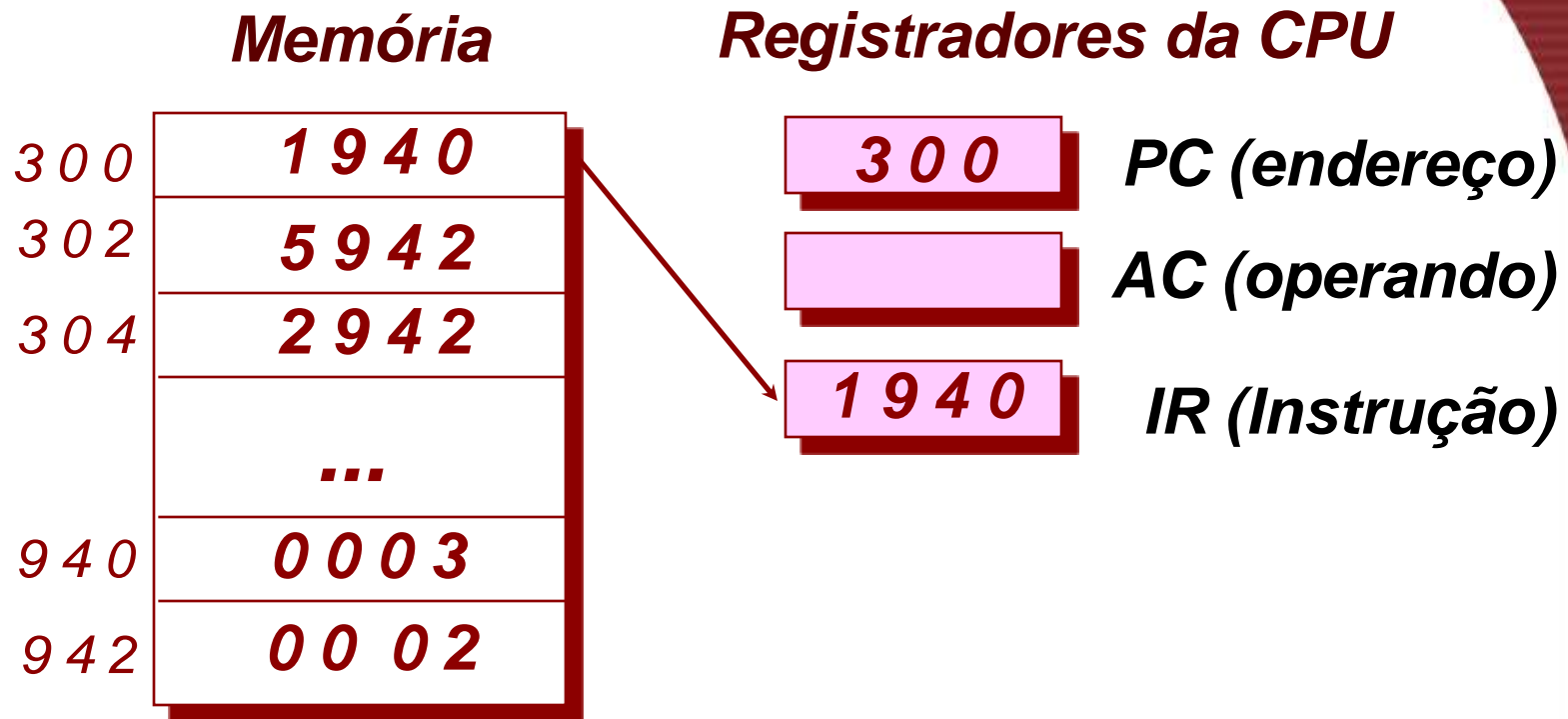


Opcode = 1 (0001)

Endereço = 940

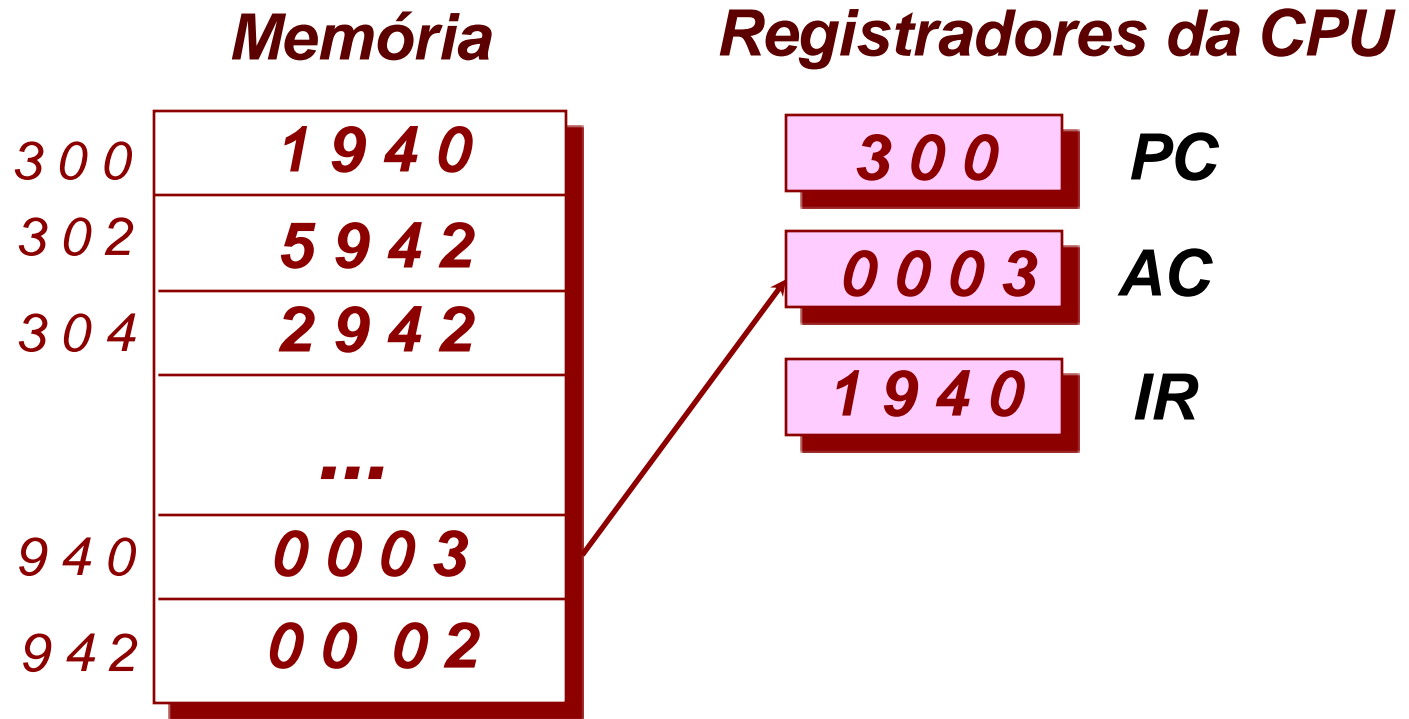
0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

Passo a Passo da Execução de um Programa



0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

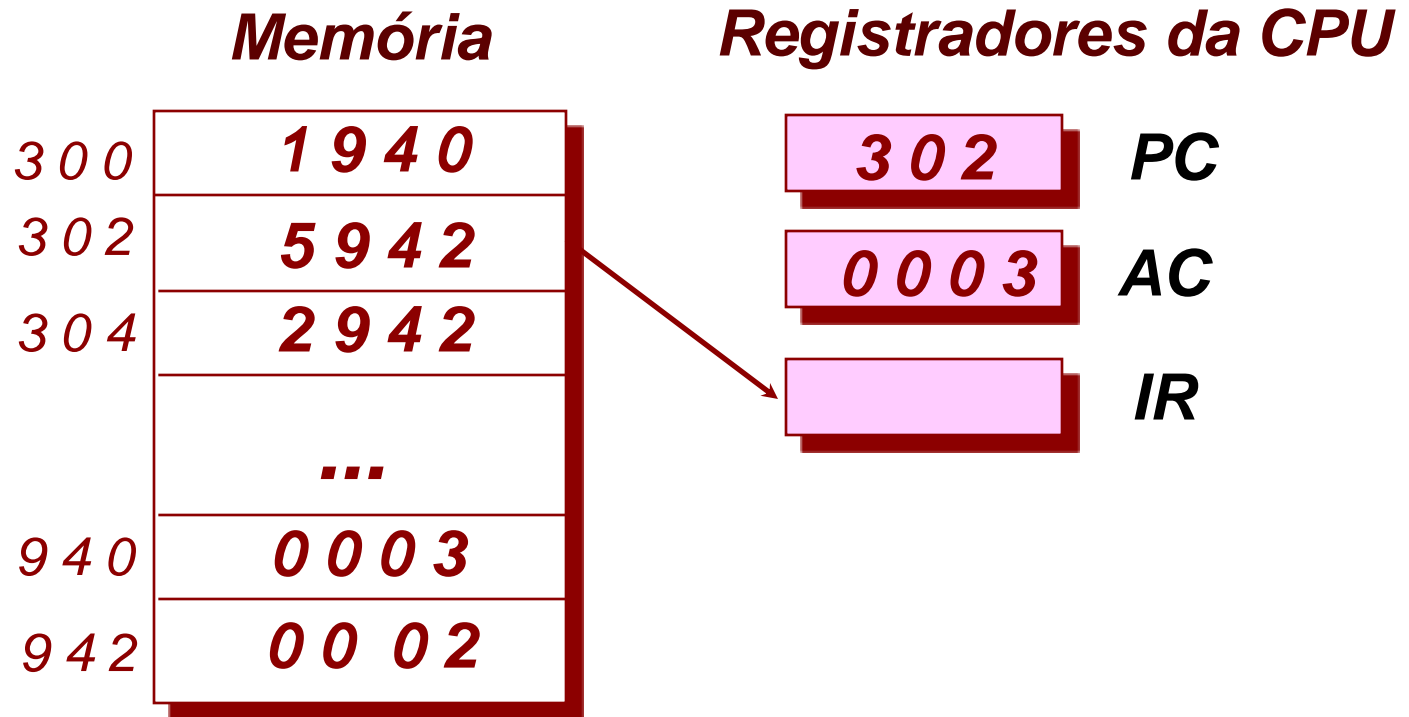
Passo a Passo da Execução de um Programa



0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

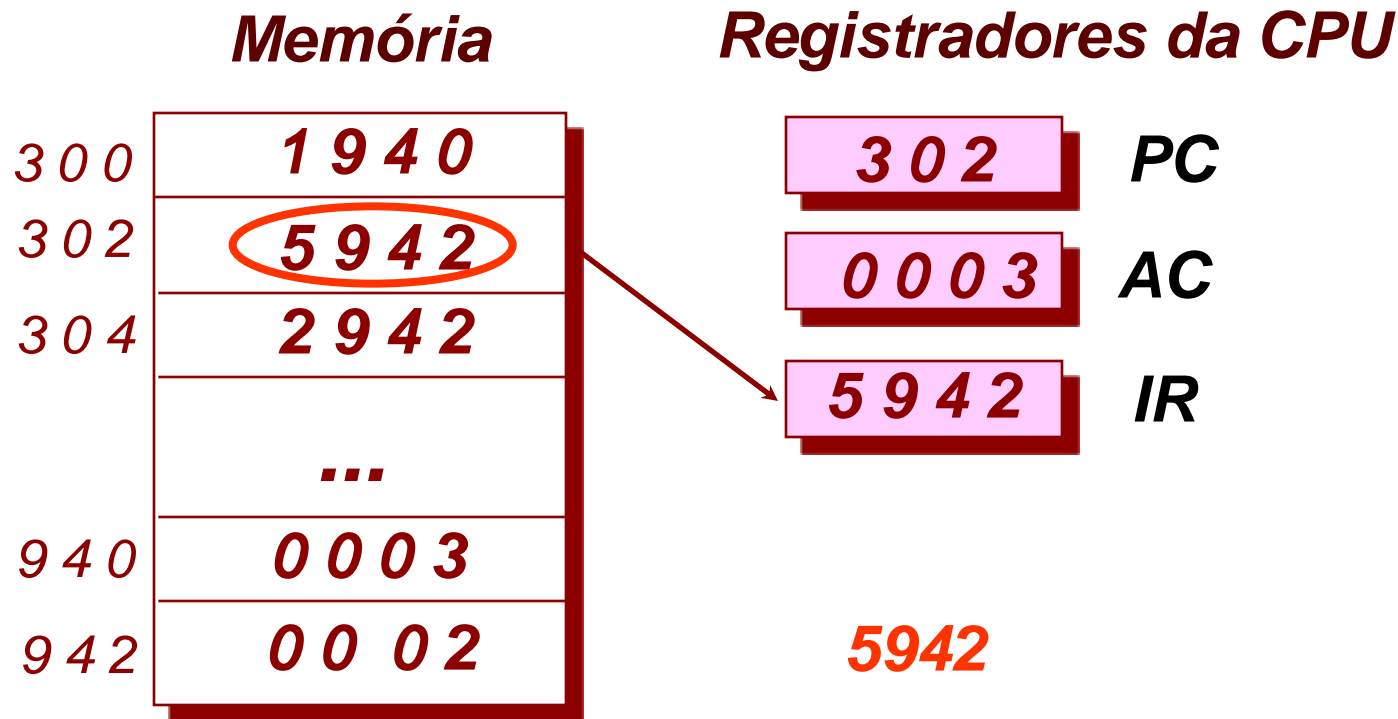


Passo a Passo da Execução de um Programa



0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

Passo a Passo da Execução de um Programa



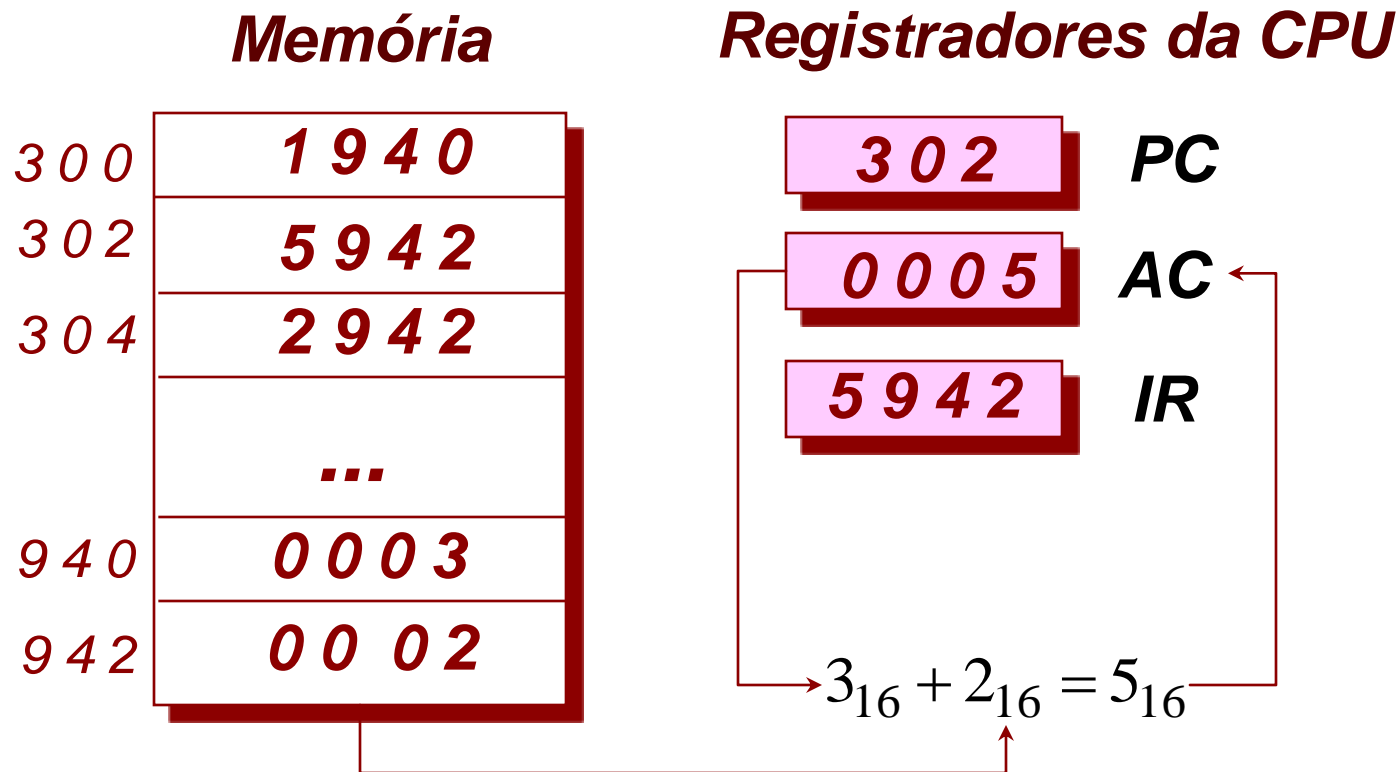
0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

5942

Opcode = 5 (0101)

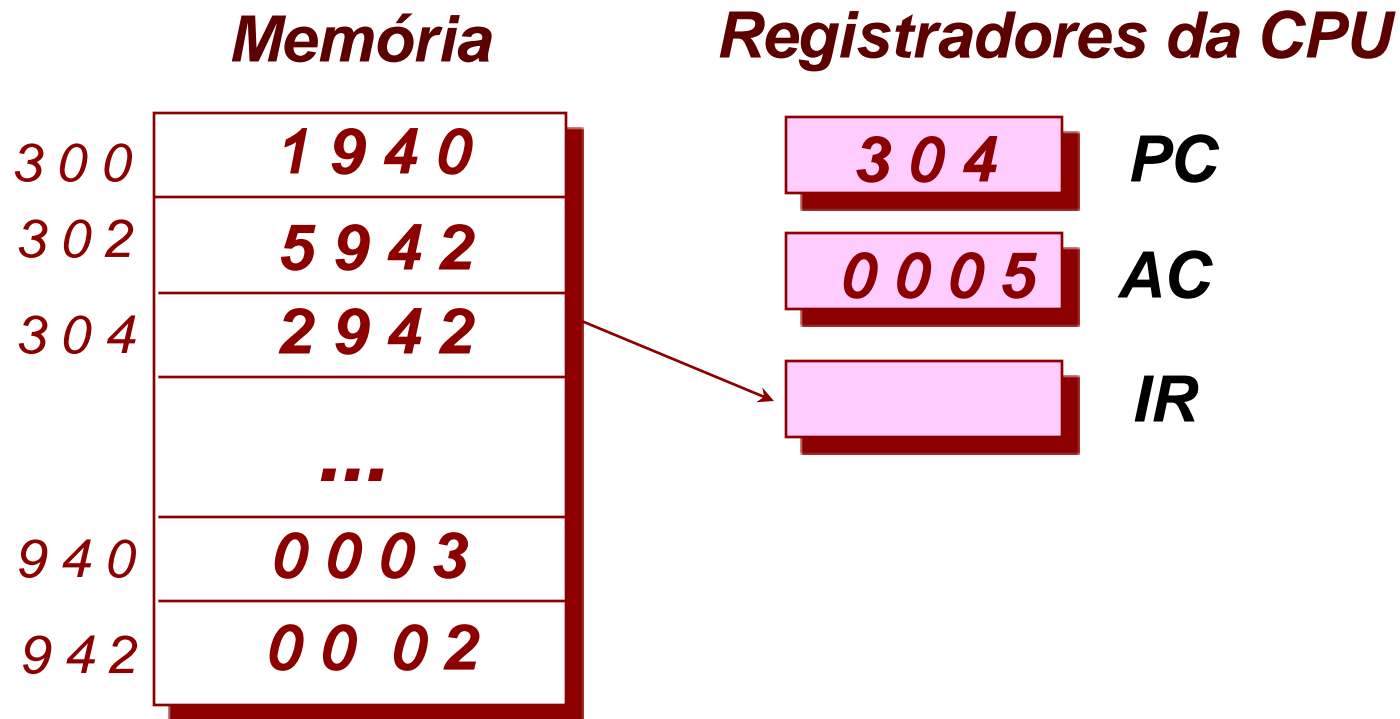
Endereço = 942

Passo a Passo da Execução de um Programa



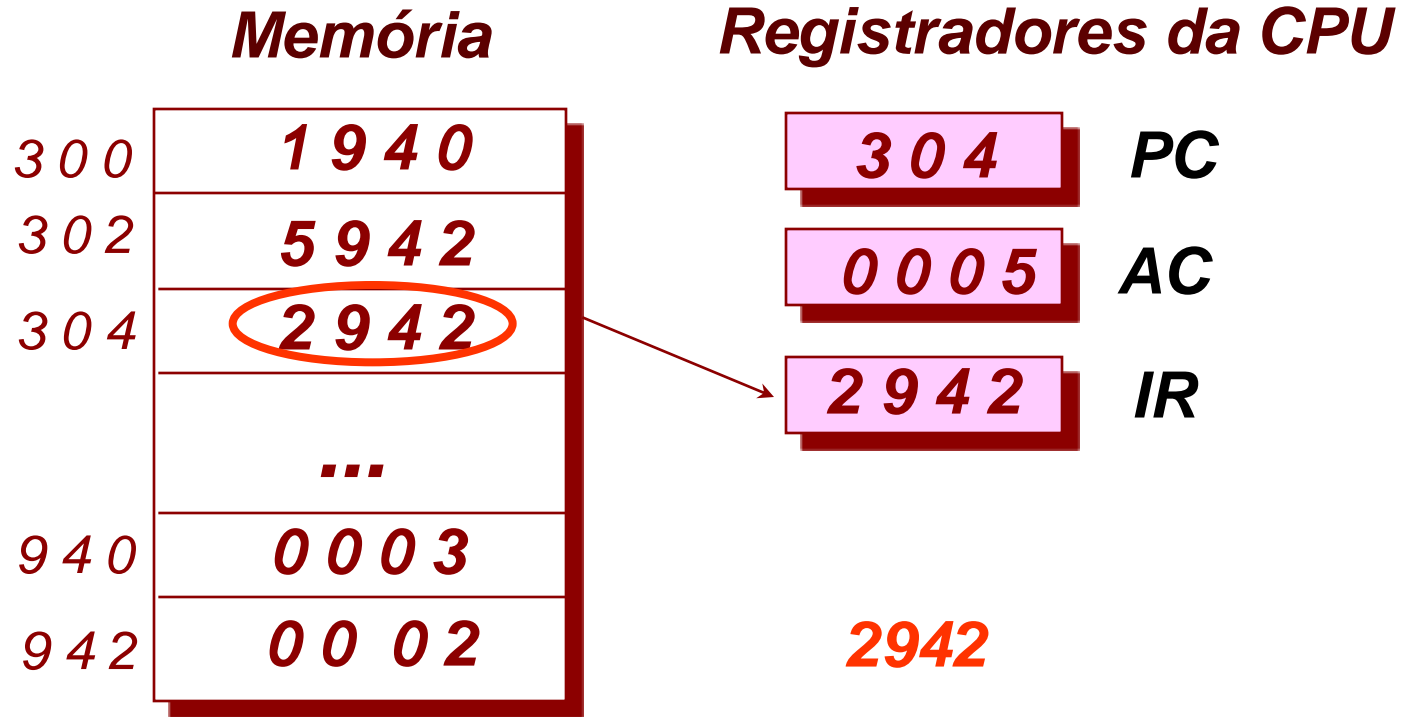
0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

Passo a Passo da Execução de um Programa



0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

Passo a Passo da Execução de um Programa



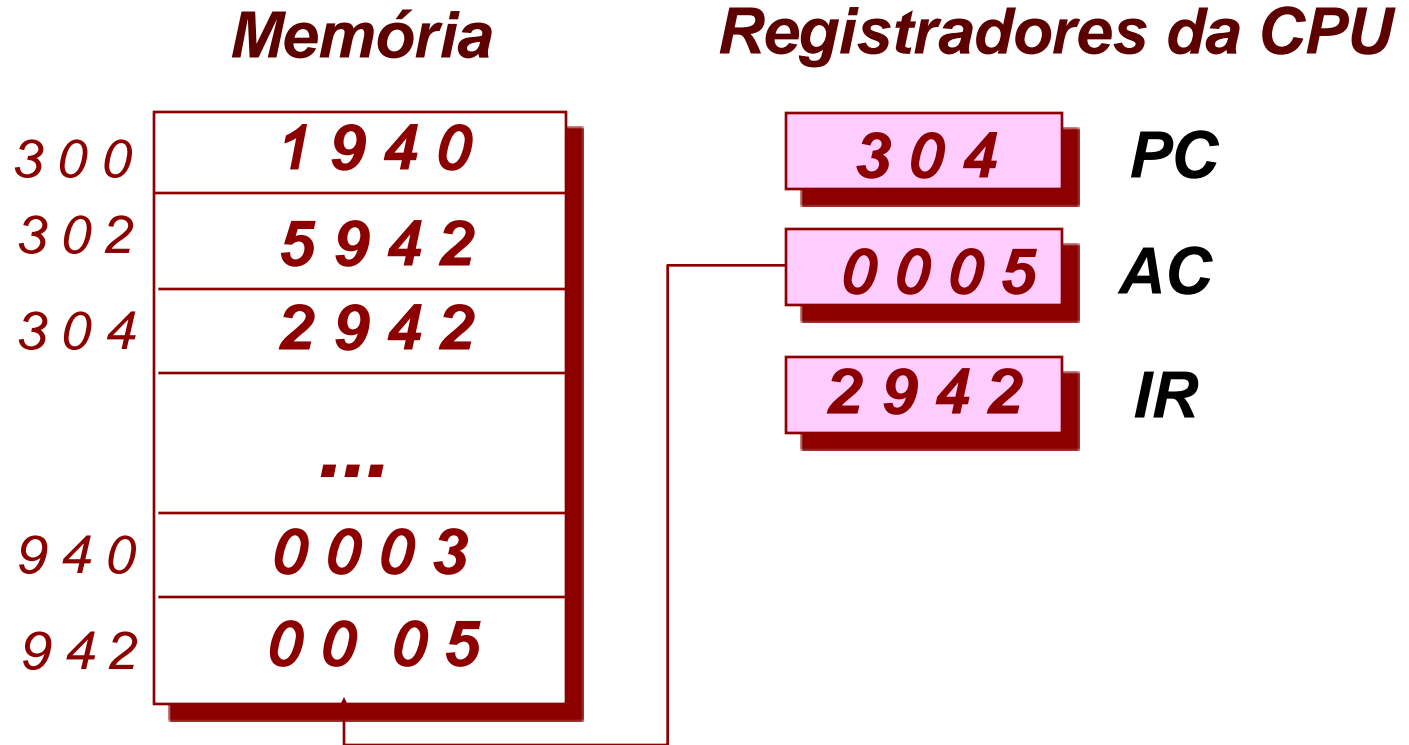
2942

Opcode = 2 (0010)

Endereço = 942

0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.

Passo a Passo da Execução de um Programa



0001	AC <- Mem.
0010	Mem. <- AC
0101	AC <- AC + Mem.



Arquitetura x Organização de Computadores

- **Arquitetura** refere-se aos atributos do sistema visíveis ao programador

- Conjunto de registradores
- Tipos de Dados
- Modo de endereçamento
- Formato e Repertório de instruções

- **Organização** refere-se às unidades operacionais que implementam a arquitetura

- Tecnologia de memória
- Interfaces
- Implementação das instruções
- Interconexões