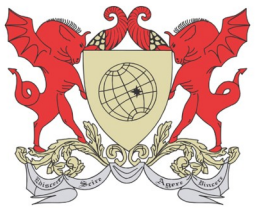
Aula 07 - Ciclo de busca e execução de instruções



Arquitetura de Von Neumann

- Arquitetura de Von Neumann
  - Dados e instruções armazenados em uma única memória;

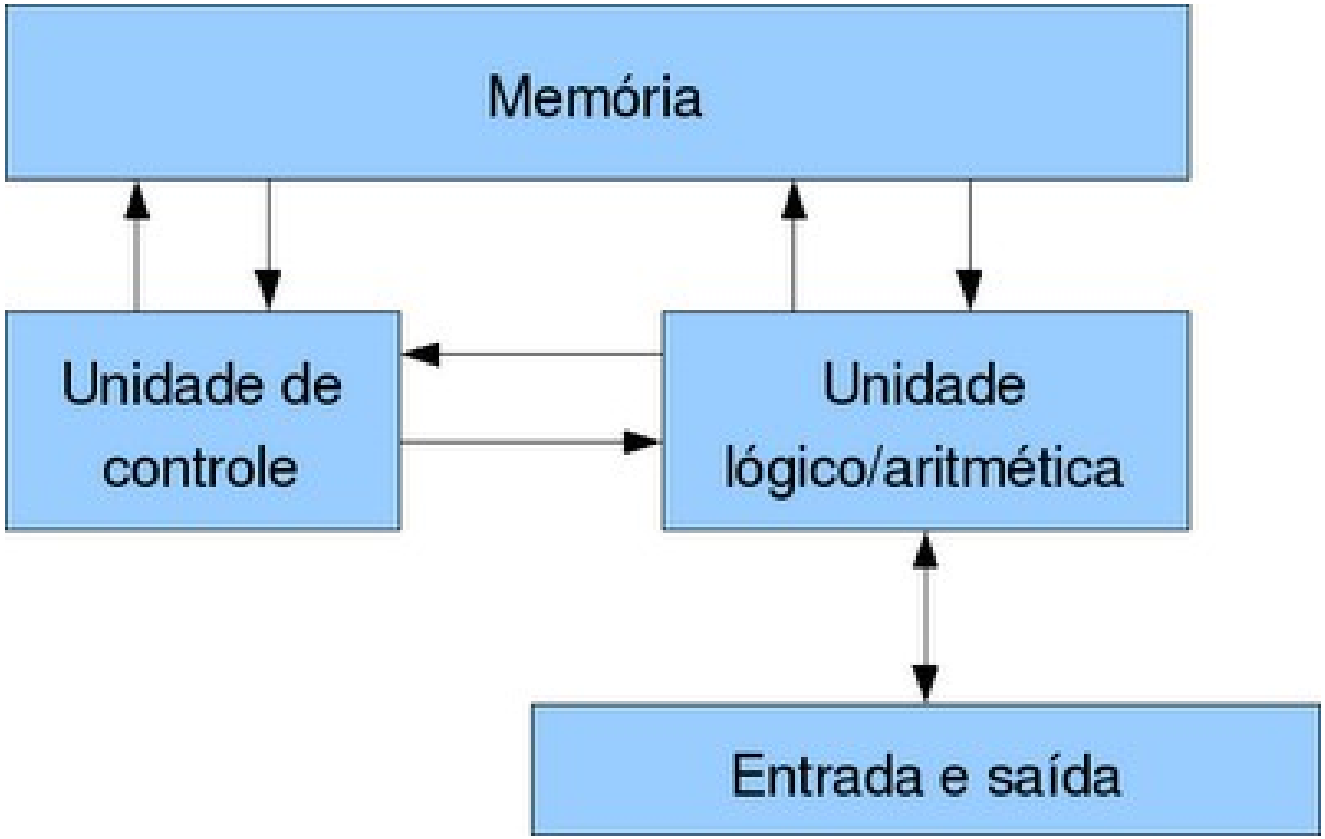
Arquitetura de Von Neumann

- Arquitetura de Von Neumann
  - Dados e instruções armazenados em uma única memória;
  - Conteúdo da memória é endereçado pela sua posição (dados ou instruções);

Arquitetura de Von Neumann

- Arquitetura de Von Neumann
  - Dados e instruções armazenados em uma única memória;
  - Conteúdo da memória é endereçado pela sua posição (dados ou instruções);
  - Instruções executadas de forma sequencial.

Arquitetura de Von Neumann



Arquitetura de Von Neumann

- Utilizando a Arquitetura de Von Neumann:
  - Não é mais necessário configurar o *hardware* (programa *hardwired*).

Programa *hardwired*: A funcionalidade (comportamento) é definida APENAS pelo circuito lógico implementado.

Arquitetura de Von Neumann

- Hardware de **propósito geral**:

Arquitetura de Von Neumann

- Hardware de **propósito geral**:
  - Pode realizar várias funções diferentes;

Arquitetura de Von Neumann

- Hardware de **propósito geral**:
  - Pode realizar várias funções diferentes;
  - A funcionalidade (comportamento) é definida por um **programa**.

Arquitetura de Von Neumann

- Hardware de **propósito geral**:
  - Pode realizar várias funções diferentes;
  - A funcionalidade (comportamento) é definida por um **programa**.
  - O Controle do hardware é realizado por **sinais**.

- Programa:

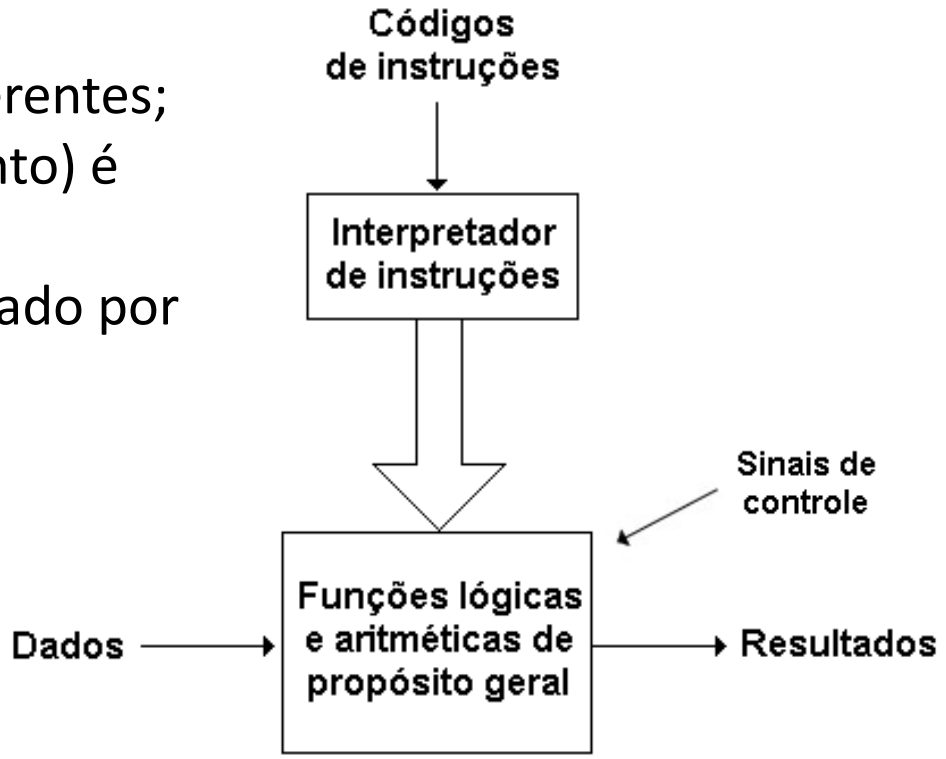
Arquitetura de Von Neumann

- Hardware de **propósito geral**:
  - Pode realizar várias funções diferentes;
  - A funcionalidade (comportamento) é definida por um **programa**.
  - O Controle do hardware é realizado por **sinais**.

- Programa:
  - Sequencia de instruções armazenadas na memória.

Arquitetura de Von Neumann

- Hardware de **propósito geral**:
  - Pode realizar várias funções diferentes;
  - A funcionalidade (comportamento) é definida por um **programa**.
  - O Controle do hardware é realizado por **sinais**.
- Programa:
  - Sequencia de instruções armazenadas na memória.



Arquitetura de Von Neumann

- Vantagens dessa abordagem:
  - Não é necessário projetar novos hardwares para cada aplicação;
  - O programador pode simplesmente fornecer uma nova sequência de sinais de controle (*programa*).

Arquitetura de Von Neumann

- PROGRAMA
  - Uma sequência de instruções (ou diretivas).

Arquitetura de Von Neumann

- PROGRAMA
  - Uma sequência de instruções (ou diretivas).
  - Para cada instrução, é realizada uma operação aritmética ou lógica.

Arquitetura de Von Neumann

- PROGRAMA
  - Uma sequência de instruções (ou diretivas).
  - Para cada instrução, é realizada uma operação aritmética ou lógica.
  - Para cada operação, é necessário um conjunto diferente de sinais de controle.

Arquitetura de Von Neumann

- PROGRAMA
  - Uma sequência de instruções (ou diretivas).
  - Para cada instrução, é realizada uma operação aritmética ou lógica.
  - Para cada operação, é necessário um conjunto diferente de sinais de controle.

- UNIDADE DE CONTROLE
  - Para cada operação, um código exclusivo é fornecido.

Arquitetura de Von Neumann

- PROGRAMA
  - Uma sequência de instruções (ou diretivas).
  - Para cada instrução, é realizada uma operação aritmética ou lógica.
  - Para cada operação, é necessário um conjunto diferente de sinais de controle.

- UNIDADE DE CONTROLE
  - Para cada operação, um código exclusivo é fornecido.
  - Exemplo: ADD, MOVE.

Arquitetura de Von Neumann

- PROGRAMA
  - Uma sequência de instruções (ou diretivas).
  - Para cada instrução, é realizada uma operação aritmética ou lógica.
  - Para cada operação, é necessário um conjunto diferente de sinais de controle.

- UNIDADE DE CONTROLE
  - Para cada operação, um código exclusivo é fornecido.
  - Exemplo: ADD, MOVE.
  - Um segmento de hardware aceita o código e emite os sinais de controle.

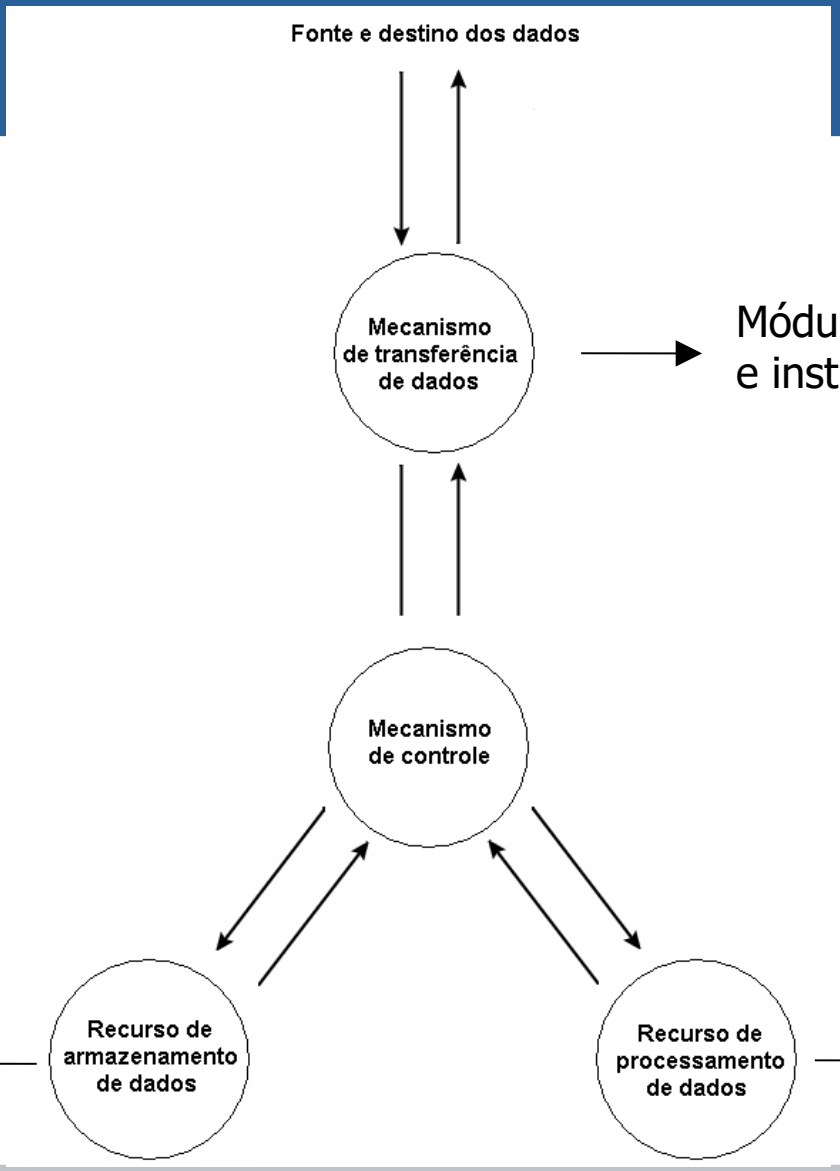
Arquitetura de Von Neumann

- PROGRAMA
  - Uma sequência de instruções (ou diretivas).
  - Para cada instrução, é realizada uma operação aritmética ou lógica.
  - Para cada operação, é necessário um conjunto diferente de sinais de controle.

- UNIDADE DE CONTROLE
  - Para cada operação, um código exclusivo é fornecido.
  - Exemplo: ADD, MOVE.
  - Um segmento de hardware aceita o código e emite os sinais de controle.

- ***Temos um computador!***

Arquitetura de Von Neumann

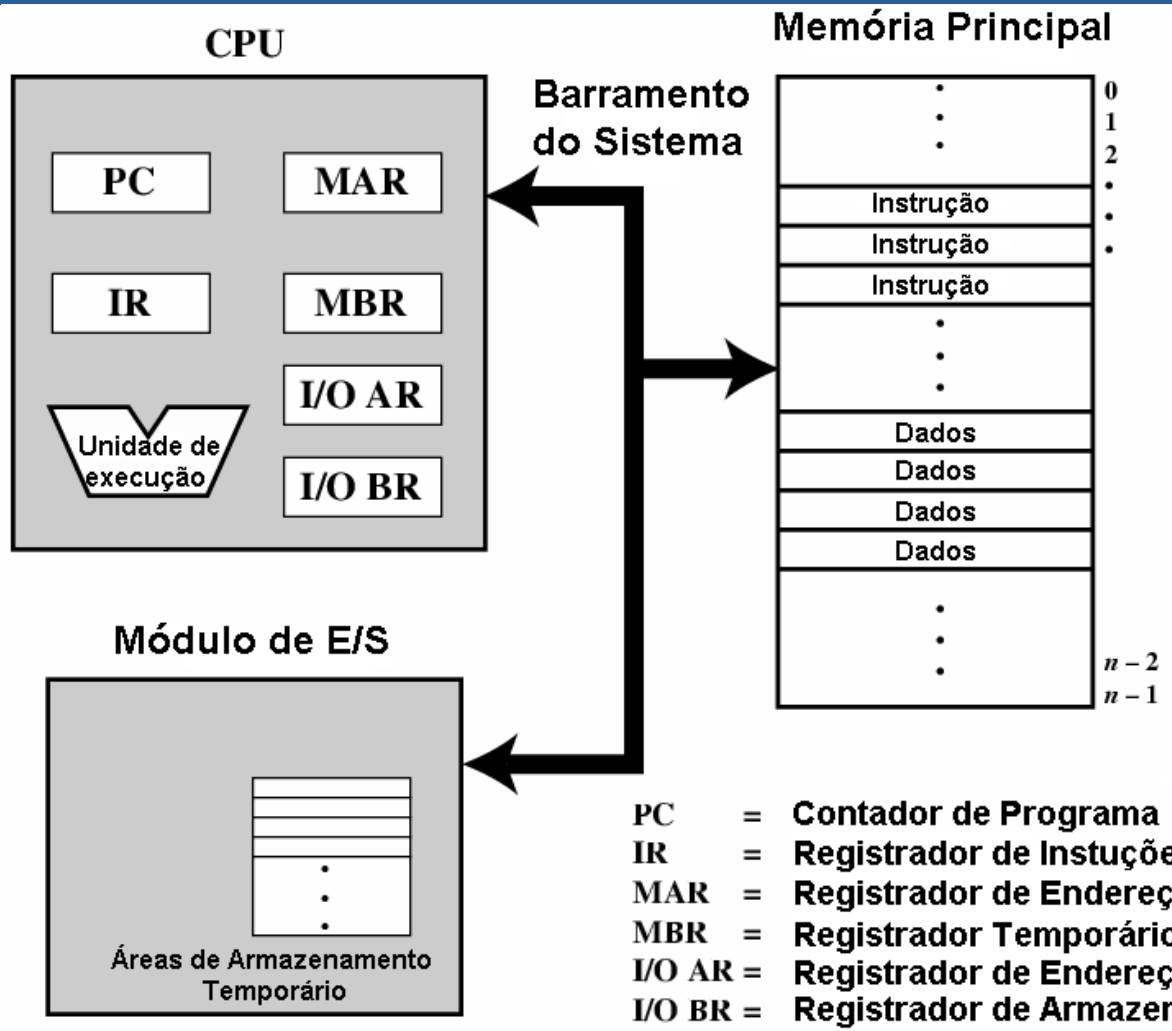


Módulo de E/S (entrada de dados e instruções)

Memória (armazenamento de dados e instruções)

Módulo de execução
Módulo de interpretação

Arquitetura de Von Neumann



PC = Contador de Programa
IR = Registrador de Instuções
MAR = Registrador de Endereçamento à Memória
MBR = Registrador Temporário de Dados
I/O AR = Registrador de Endereçamento de E/S
I/O BR = Registrador de Armazenamento Temporário de Dados de E/S

Função dos computadores

Funções dos Computadores

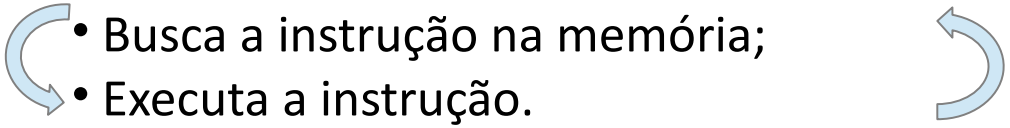
- Função básica de um computador:
  - Executar programas (conjunto de instruções):

Funções dos Computadores

- Função básica de um computador:
  - Executar programas (conjunto de instruções):
    - Busca a instrução na memória;
    - Executa a instrução.

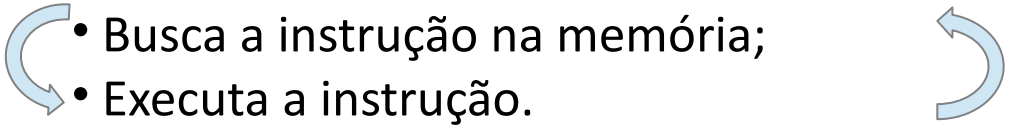
Funções dos Computadores

- Função básica de um computador:
  - Executar programas (conjunto de instruções):
    - Busca a instrução na memória;
    - Executa a instrução.



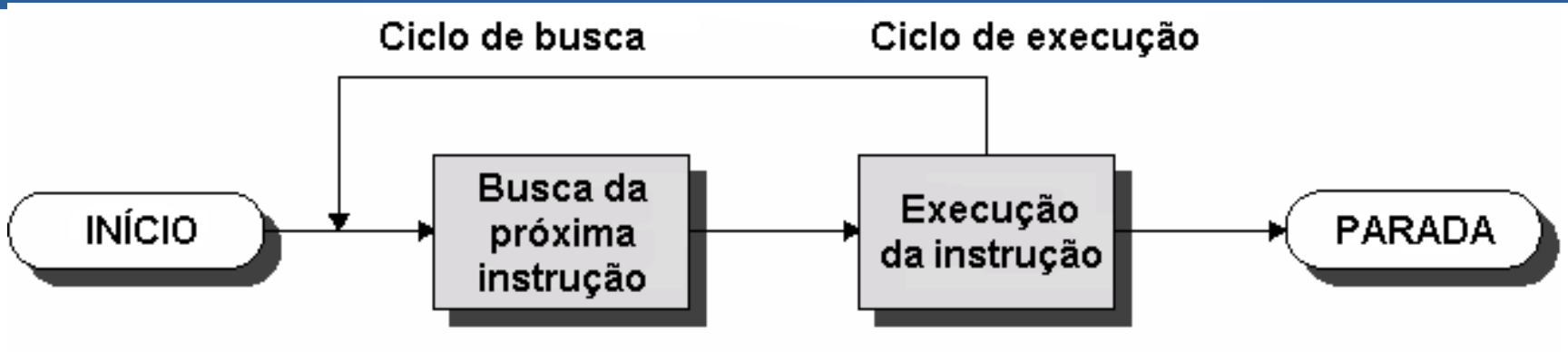
Funções dos Computadores

- Função básica de um computador:
  - Executar programas (conjunto de instruções):
    - Busca a instrução na memória;
    - Executa a instrução.



  - Uma instrução pode envolver várias operações;
  - Ciclo de instrução: processamento necessário para executar uma instrução.

Funções dos Computadores



- A instrução é buscada no endereço de memória contido no PC e carregada no Registrador de Instrução (IR):
  - Processador <-> memória;
  - Processador <-> E/S;
  - Processamento de dados;
  - Controle

Funções dos Computadores

- Contador de Programa (PC) mantém endereço da próxima instrução a buscar.

Funções dos Computadores

- Contador de Programa (PC) mantém endereço da próxima instrução a buscar.

- Processador busca instrução do local de memória apontado pelo PC.

Funções dos Computadores

- Contador de Programa (PC) mantém endereço da próxima instrução a buscar.

- Processador busca instrução do local de memória apontado pelo PC.

- Incrementar PC:
  - A menos que seja informado de outra forma.

Funções dos Computadores

- Contador de Programa (PC) mantém endereço da próxima instrução a buscar.

- Processador busca instrução do local de memória apontado pelo PC.

- Incrementar PC:
  - A menos que seja informado de outra forma.

- Instrução carregada no Registrador de Instrução (IR).

Funções dos Computadores

- Contador de Programa (PC) mantém endereço da próxima instrução a buscar.

- Processador busca instrução do local de memória apontado pelo PC.

- Incrementar PC:
  - A menos que seja informado de outra forma.

- Instrução carregada no Registrador de Instrução (IR).

- Processador interpreta instrução e realiza ações exigidas.

Funções dos Computadores

- Considerando uma máquina:
  - Com dados e instruções de 16 bits;
  - Um único registrador de armazenamento (AC).

Formato das Instruções

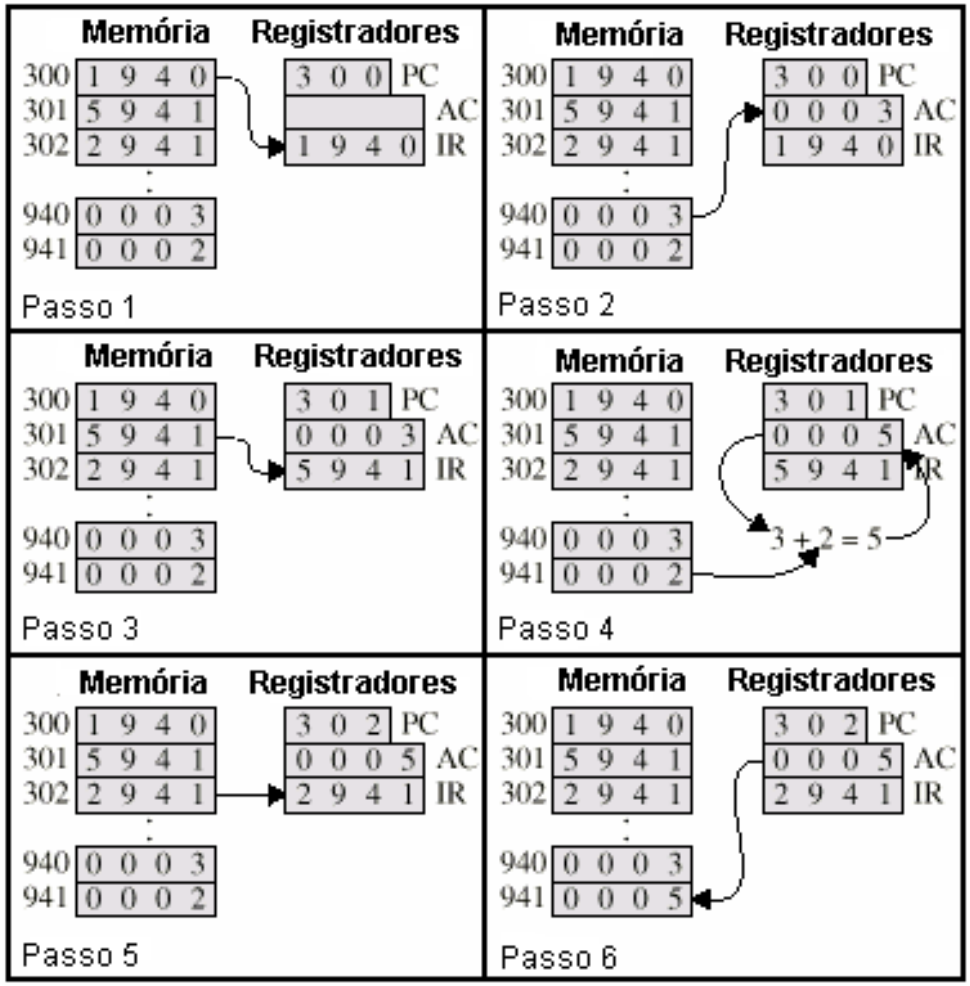
$2^4 = 16$ $2^{12} = 4096$ (4K)

Formato dos Dados

$2^{15} = 32768$

Funções dos Computadores

- **Instruções**:



Funções dos Computadores

- Instruções mais **complexas**,
  - necessitam de uma **menor** quantidade de ciclos para realizar uma operação;
- Portanto, em um ciclo de instrução:
  - Podemos ter **mais** de uma referência à memória;
  - Uma instrução pode especificar uma operação de E/S;

Funções dos Computadores

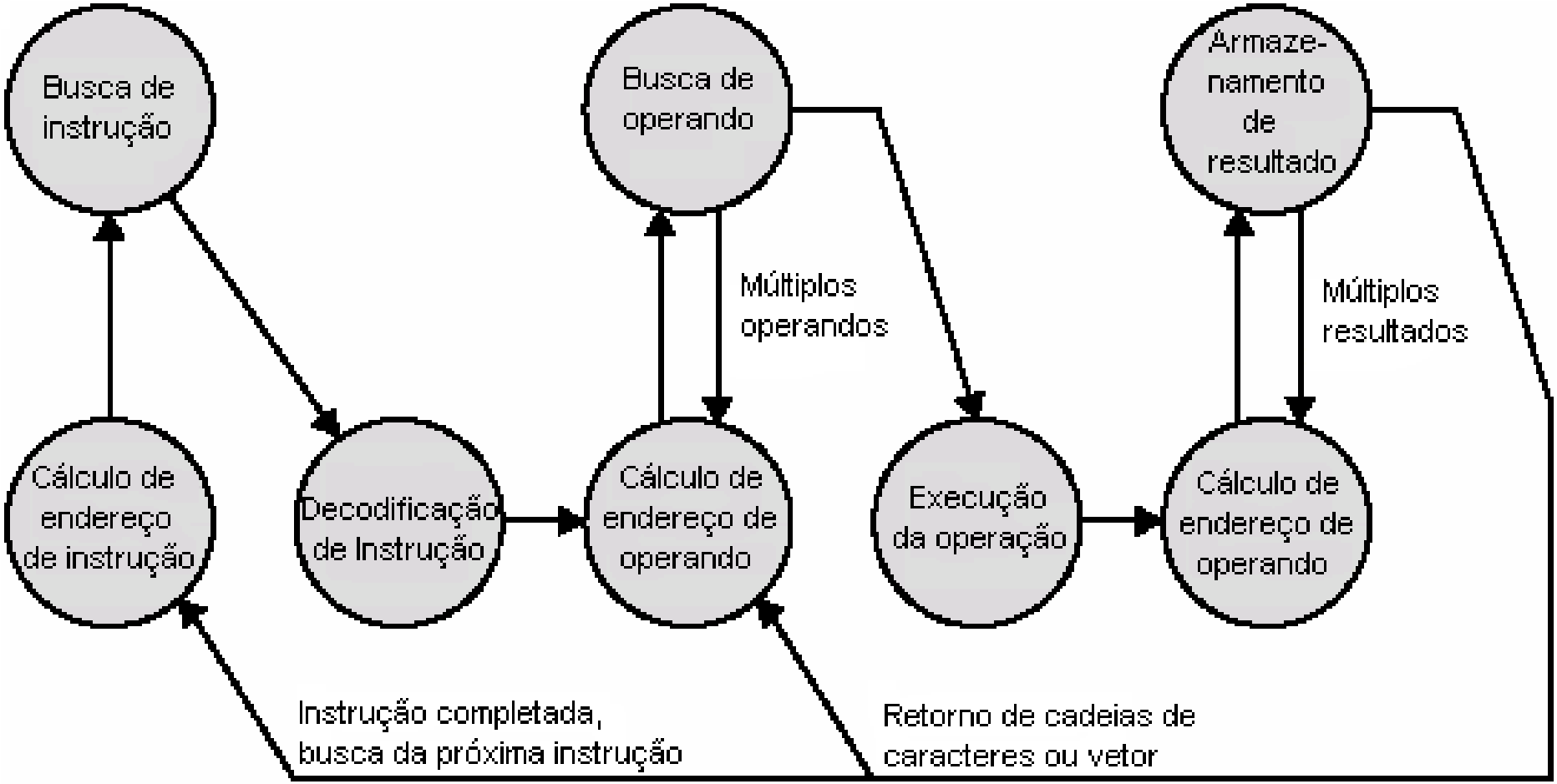


Diagrama de transição de estados

Conjunto de Instruções

Conjunto de instruções

- As instruções de máquina (do computador) determinam a operação que a CPU deve executar;

Conjunto de instruções

- As instruções de máquina (do computador) determinam a operação que a CPU deve executar;
- A coleção dessas instruções é conhecida como **conjunto de instruções** da CPU;

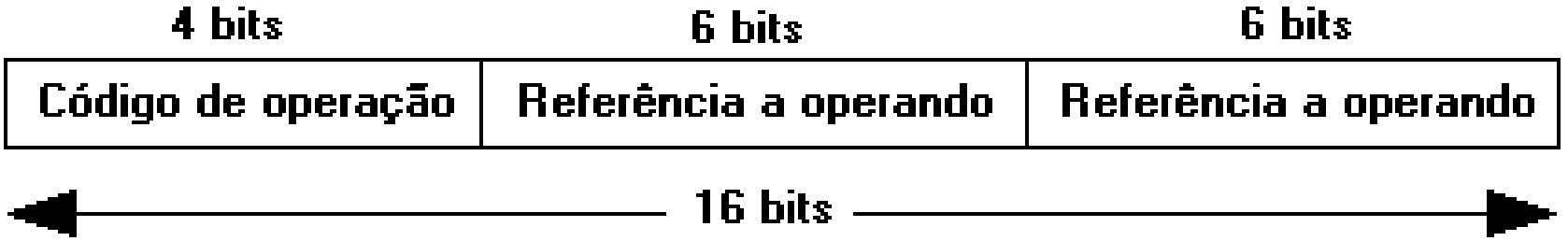
Conjunto de instruções

- As instruções de máquina (do computador) determinam a operação que a CPU deve executar;
- A coleção dessas instruções é conhecida como **conjunto de instruções** da CPU;
- Cada instrução deve conter todos os dados necessários para que a CPU possa executá-la.

Representação de instruções

- Cada instrução contém:
  - Código de operação;
  - Referência a operando fonte;
  - Referência a operando destino;
  - Endereço da próxima instrução.
    - Implícito pelo Contador de Programas, exceto para instruções de desvio.

- Cada arquitetura possui uma organização especifica para as suas instruções.
  - EXEMPLO de uma instrução de 16 bits e duas referencias a operando.



Representação de instruções

- Processamento de dados:
  - ADD –> Adição
  - SUB –> Subtração
  - MPY –> Multiplicação
  - DIV –> Divisão
  - ...

Representação de instruções

- Processamento de dados:
  - ADD –> Adição
  - SUB –> Subtração
  - MPY –> Multiplicação
  - DIV –> Divisão
  - ...

- Movimentação de dados:
  - LOAD –> Carregar dados da memória (ou LDA)
  - STOR –> Armazenar dados na memória (ou STA)
  - ...

Representação de instruções

- Processamento de dados:
  - ADD –> Adição
  - SUB –> Subtração
  - MPY –> Multiplicação
  - DIV –> Divisão
  - ...

- Movimentação de dados:
  - LOAD –> Carregar dados da memória (ou LDA)
  - STOR –> Armazenar dados na memória (ou STA)
  - ...

- Desvio:
  - *Necessário para as instruções de alto nível de desvio condicional (if) e loops (for, while).*
  - J –> Jump (Salto incondicional)
  - ...

Representação de instruções

- Número de referencias à operandos
  - ***Operação de Soma (ADD):***
  - Referencia a 1 operando:
    - ADD X X -> Operando de origem e destino
      - Soma o valor armazenado no registrados X ao valor atual do registrador ACUMULADOR. Armazena o resultado em X.

Representação de instruções

- Número de referencias à operandos
  - ***Operação de Soma (ADD):***
  - Referencia a 1 operando:
    - ADD X X -> Operando de origem e destino
      - Soma o valor armazenado no registrados X ao valor atual do registrador ACUMULADOR. Armazena o resultado em X.
  - Referencia a 2 operandos:
    - ADD X, Y X e Y - > Operando de origem; X -> Op. de destino
      - Soma o valor ATUAL do registrador X com o valor armazenado no registrador Y. Armazena o resultado no registrador X

Representação de instruções

- Número de referencias à operandos
  - ***Operação de Soma (ADD):***
  - Referencia a 1 operando:
    - ADD X X -> Operando de origem e destino
      - Soma o valor armazenado no registrados X ao valor atual do registrador ACUMULADOR. Armazena o resultado em X.
  - Referencia a 2 operandos:
    - ADD X, Y X e Y - > Operando de origem; X -> Op. de destino
      - Soma o valor ATUAL do registrador X com o valor armazenado no registrador Y. Armazena o resultado no registrador X
  - Referencia a 3 operandos:
    - ADD, X, Y, Z X -> Operando de destino Y e Z -> Ops. de origem.
      - Soma os valores armazenados nos registradores Y e Z e armazena o resultado em no registrador X.

Representação de instruções

1 operando	2 operandos	3 operandos
Y = (A - B) / (C + D * E);	Y = (A - B) / (C + D * E);	Y = (A - B) / (C + D * E);

```
LOAD D
MPY  E
ADD  C
STOR Y
LOAD A
SUB  B
DIV  Y
STOR Y
```

Representação de instruções

1 operando	2 operandos	3 operandos
Y = (A - B) / (C + D * E);	Y = (A - B) / (C + D * E);	Y = (A - B) / (C + D * E);

LOAD D
MPY E
ADD C
STOR Y
LOAD A
SUB B
DIV Y
STOR Y
	MOVE Y, A
	SUB Y, B
	MOVE T, D
	MPY T, E
	ADD T, C
	DIV Y, T	

Representação de instruções

1 operando	2 operandos	3 operandos

Y = (A - B) / (C + D * E);
 	
LOAD D
MPY E
ADD C
STOR Y
LOAD A
SUB B
DIV Y
STOR Y
	Y = (A - B) / (C + D * E);
	
	MOVE Y, A
	SUB Y, B
	MOVE T, D
	MPY T, E
	ADD T, C
	DIV Y, T
		Y = (A - B) / (C + D * E);
		
		SUB Y, A, B
		MPY T, D, E
		ADD T, T, C
		DIV Y, Y, T

Conjunto de instruções

EXEMPLO:

Linguagem de Montagem

Programa em Assembly			*Programa Simbólico*			*Programa em hexadecimal*		*Programa em binário*	
Rot.	**Opr.**	**Opn.**	**Mem.**	**Opr.**	**Opn.**	**Mem.**	**Operador e Operando**	**Mem.**	**Operador e Operando**
FORM:	LDA	I	101	LDA	201	101	2201	101	0010 0010 0000 0001
	ADD	J	102	ADD	202	102	1202	102	0001 0010 0000 0010
	ADD	K	103	ADD	203	103	1203	103	0001 0010 0000 0011
	STA	N	104	STA	204	104	3204	104	0011 0010 0000 0100
	...	...	...	...	...	...	...	...	...
...	DAT	2	201	DAT	2	201	0002	201	0000 0000 0000 0010
I	DAT	3	202	DAT	3	202	0003	202	0000 0000 0000 0011
J	DAT	4	203	DAT	4	203	0004	203	0000 0000 0000 0100
K	DAT	0	204	DAT	0	204	0000	204	0000 0000 0000 0000
N									

Rot = Rótulo; Opr = Operação; Opn = Operando; Mem = Memória.

Interrupções

Interrupções

- Interrupções são mecanismos pelos quais componentes (Memória, E/S) podem interromper a sequência normal de instruções.

- O objetivo principal é melhorar a eficiência do processamento.

Interrupções

- Classes de interrupção:
  - Interrupção de software: overflow, divisão por zero, instrução ilegal ou referência a memória fora do espaço do programa;

Interrupções

- Classes de interrupção:
  - Interrupção de software: overflow, divisão por zero, instrução ilegal ou referência a memória fora do espaço do programa;
  - Interrupção de relógio: gerada pelo relógio interno do processador;

Interrupções

- Classes de interrupção:
  - Interrupção de software: overflow, divisão por zero, instrução ilegal ou referência a memória fora do espaço do programa;
  - Interrupção de relógio: gerada pelo relógio interno do processador;
  - Interrupção de E/S: gerada por um controlador de E/S;

Interrupções

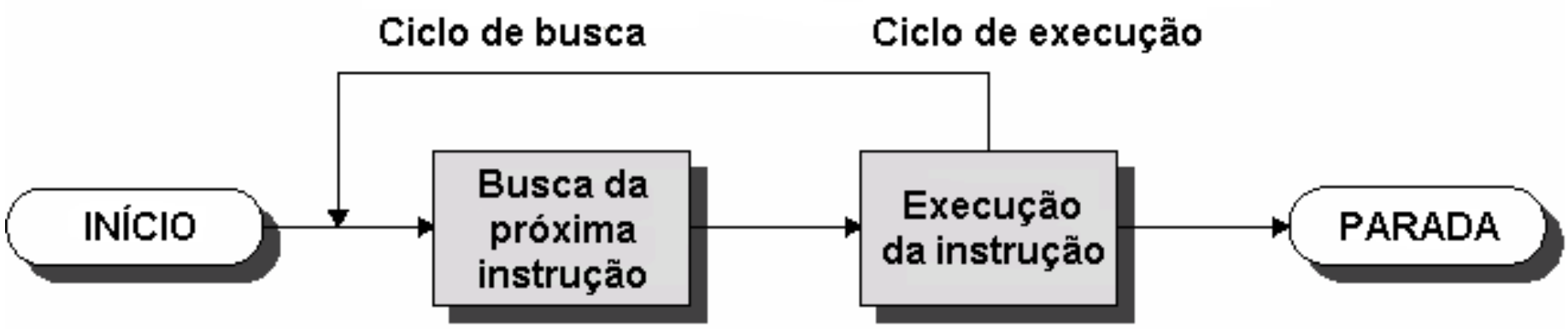
- Classes de interrupção:
  - Interrupção de software: overflow, divisão por zero, instrução ilegal ou referência a memória fora do espaço do programa;
  - Interrupção de relógio: gerada pelo relógio interno do processador;
  - Interrupção de E/S: gerada por um controlador de E/S;
  - Interrupção de falha de hardware: gerada pela falha de hardware (queda de energia, erro de paridade).

O ciclo de interrupção

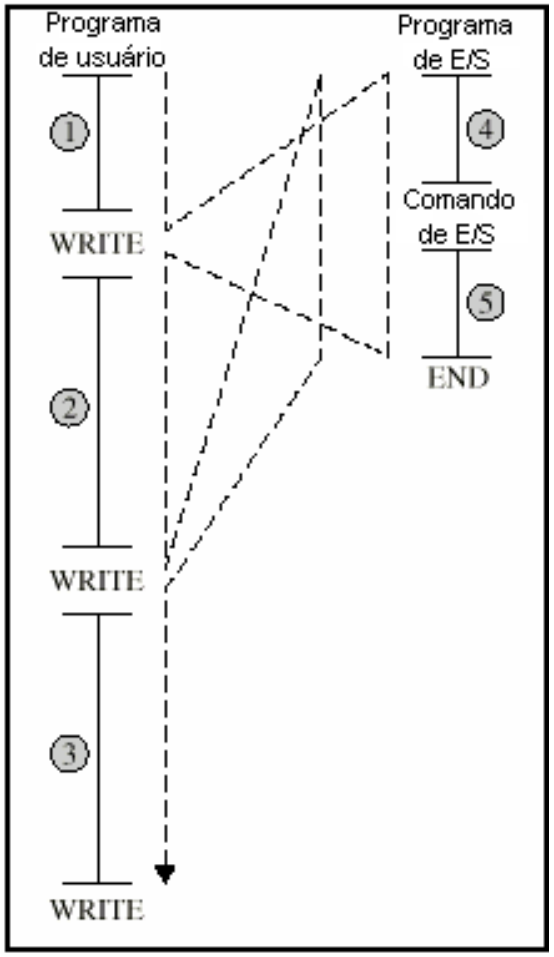
- O ciclo de interrupção é adicionado ao ciclo de instrução.
- Processador verifica interrupção, indicado por um sinal de interrupção.
- Se não houver interrupção, busca a próxima instrução.
- Se houver interrupção pendente:
  - Suspende a execução do programa atual.
  - Salva o contexto.
  - Define PC para endereço inicial da rotina de tratamento de interrupção.
  - Interrupção de processo.
  - Restaura contexto e continua programa interrompido.

Interrupções

- Exemplo:
  - Um processador recebe uma instrução para transferência de dados para a impressora (WRITE):



Interrupções



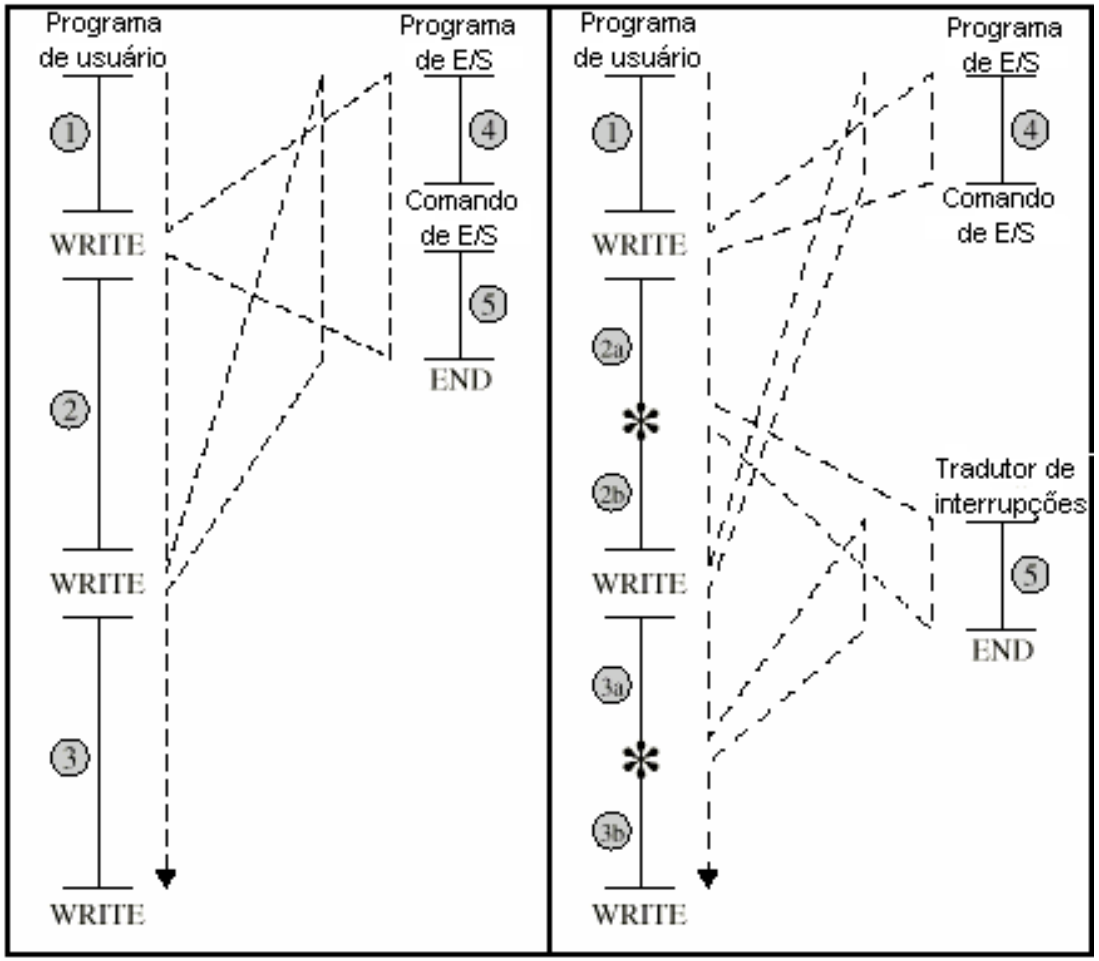
Sem interrupções

1 2 3 → Instruções que não envolvem E/S

4 → Preparo para a operação de E/S

5 → Instruções para conclusão da E/S

Interrupções

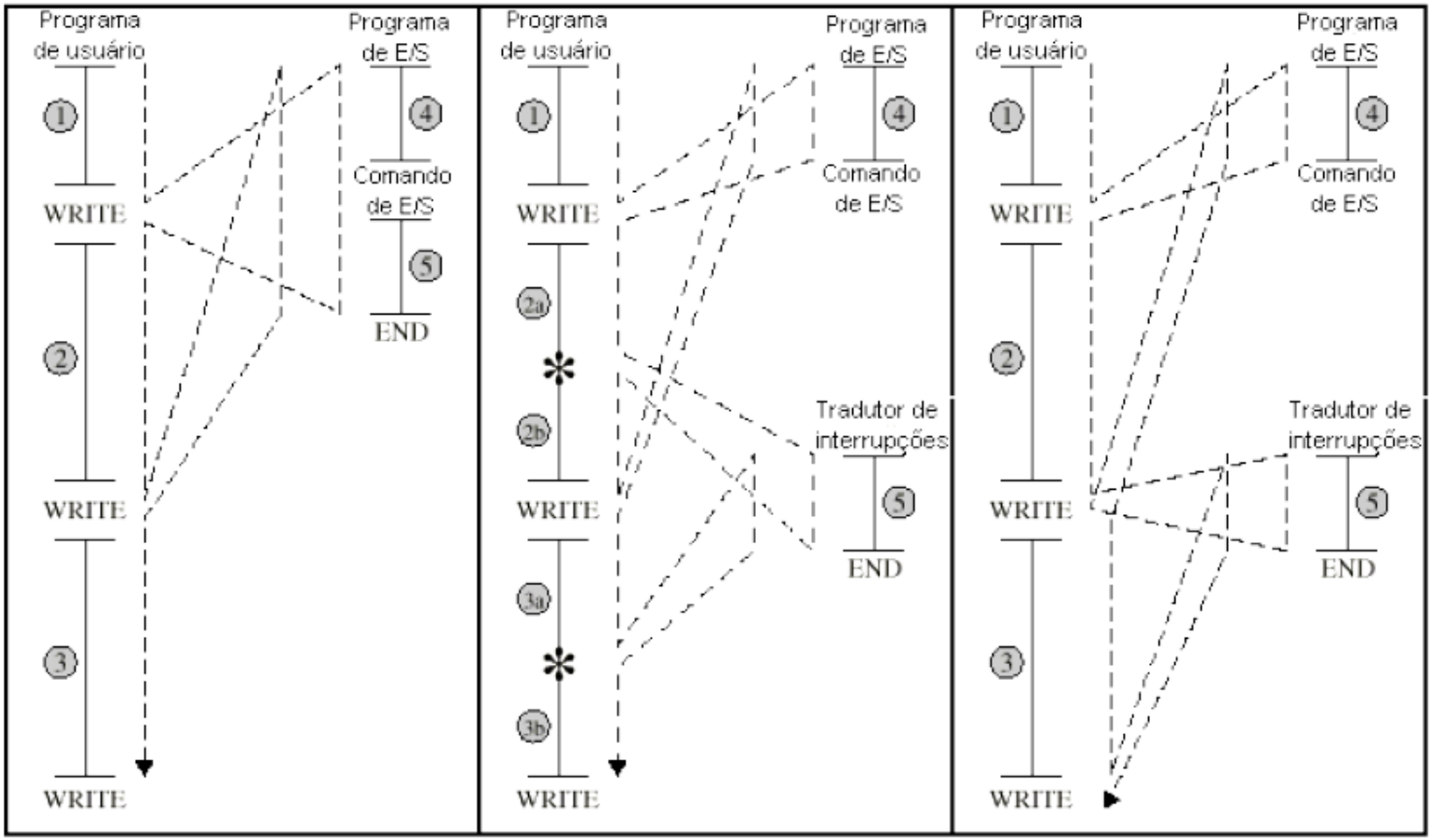


Sem interrupções

Com interrupções

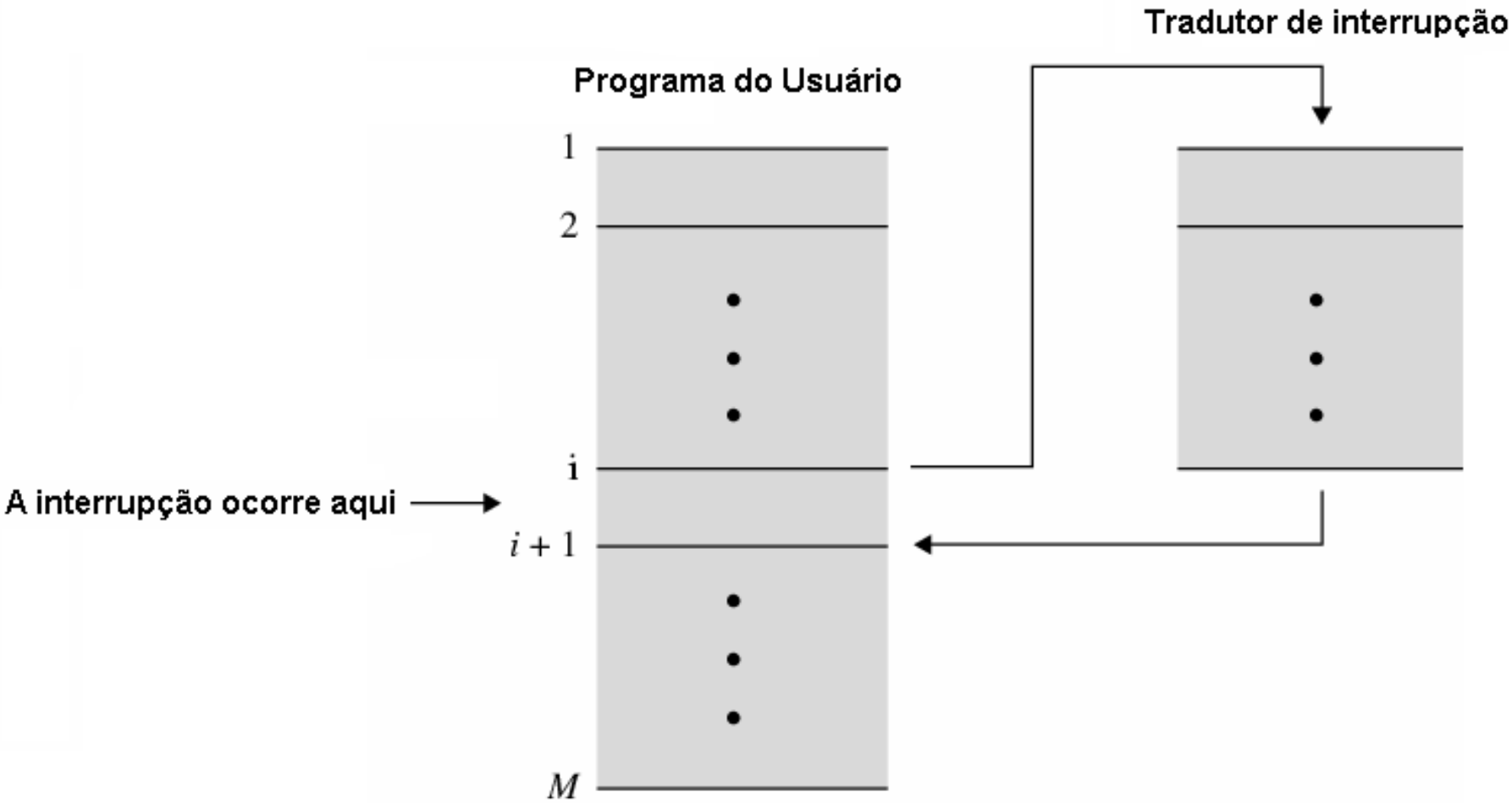
* → Ocorrência de Interrupção

Interrupções

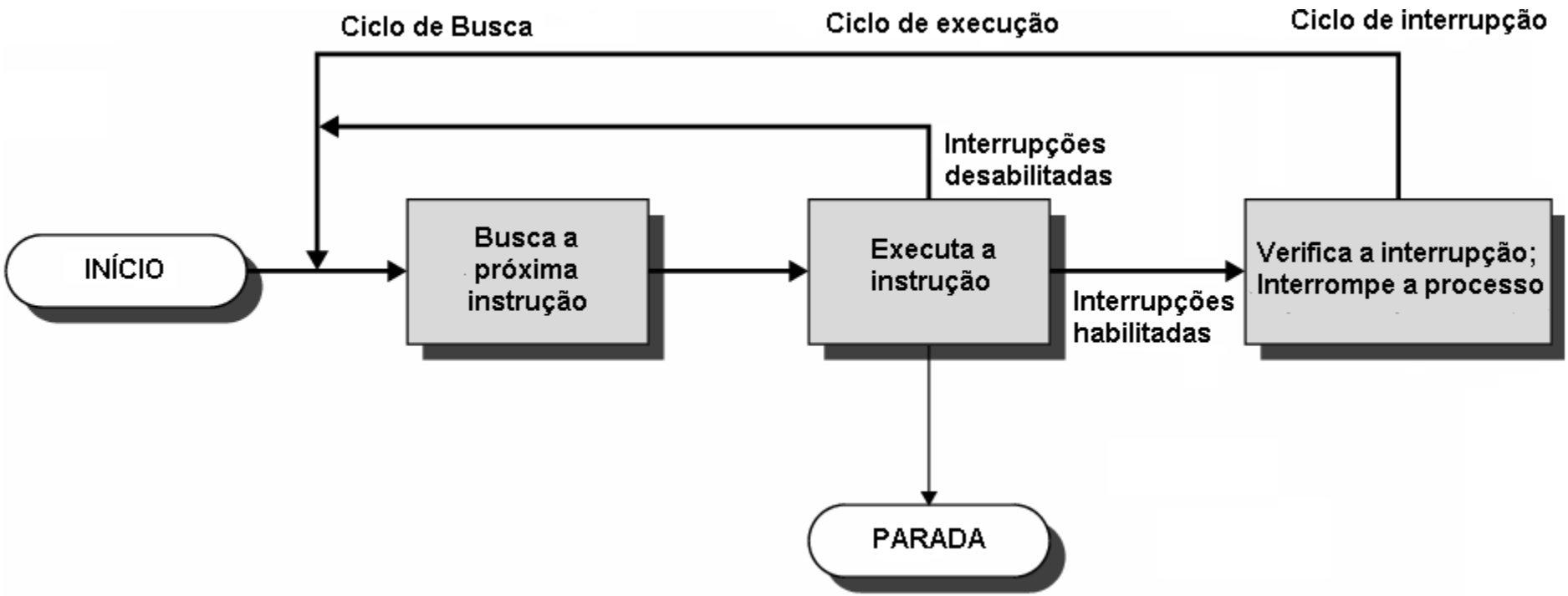


Interrupções

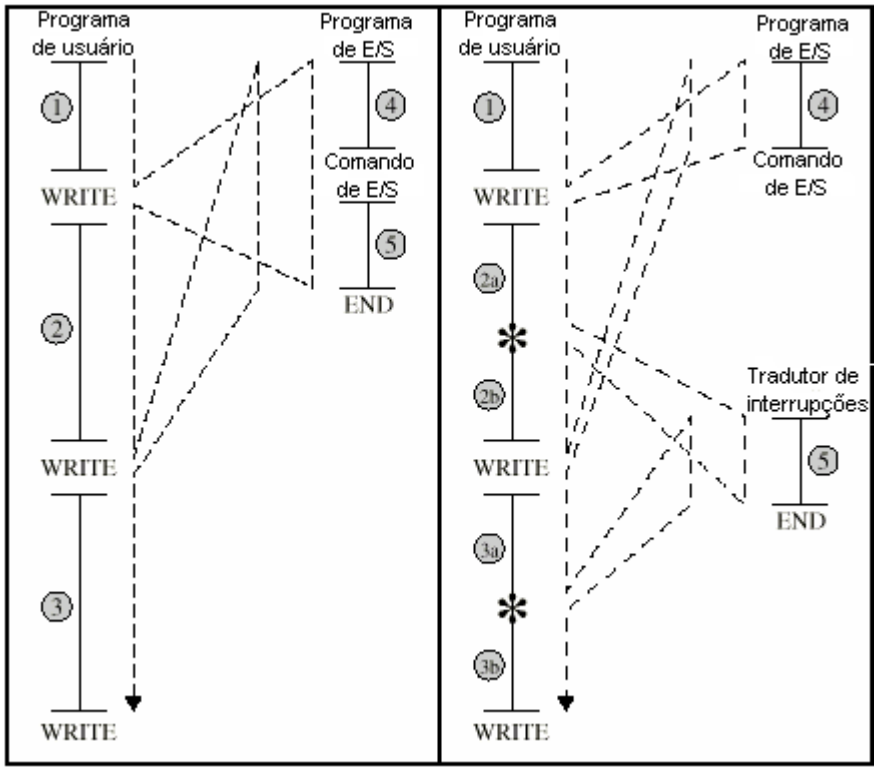
- Do ponto de vista do usuário:

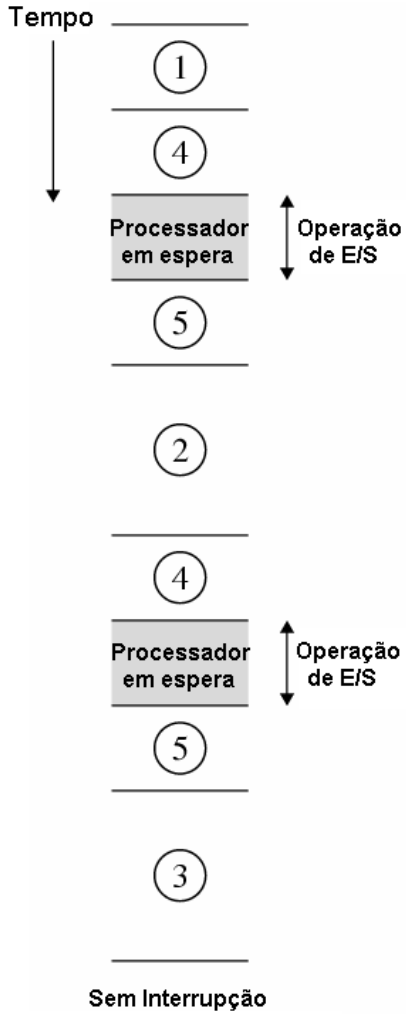


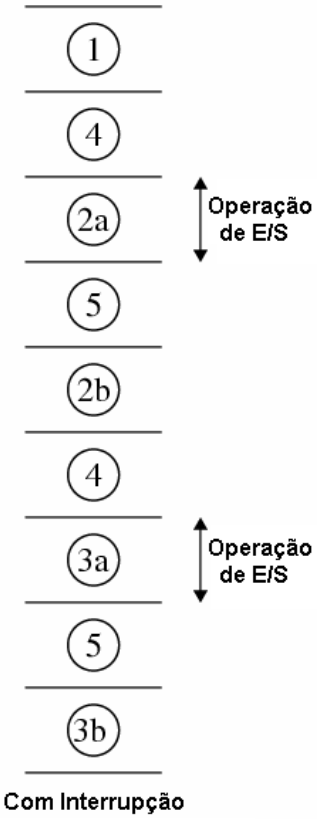
Ciclo de instrução com interrupção



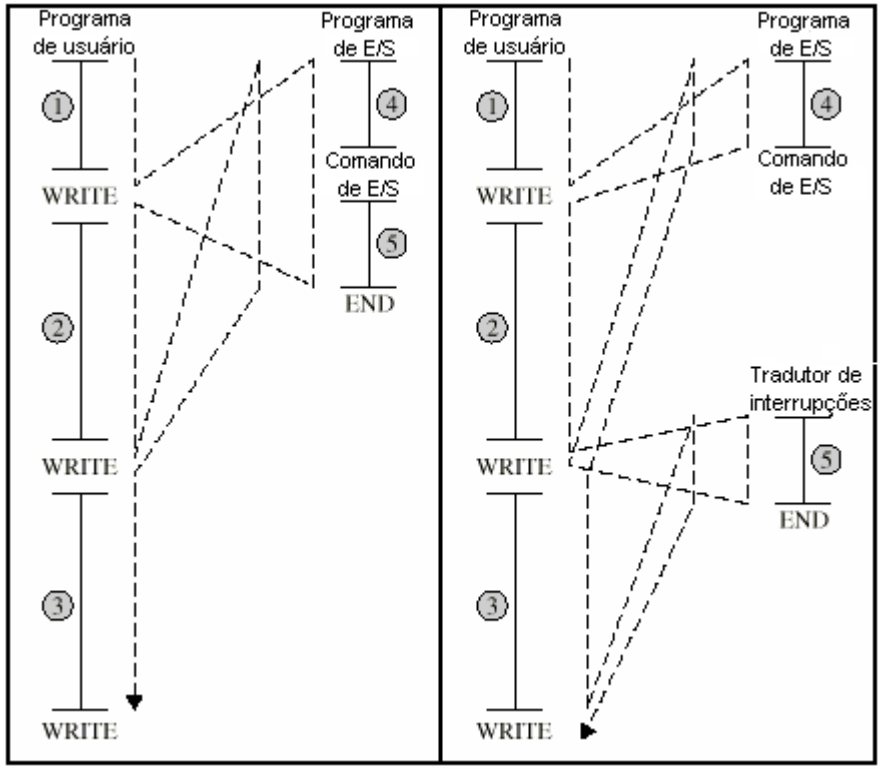
Ganho em eficiência

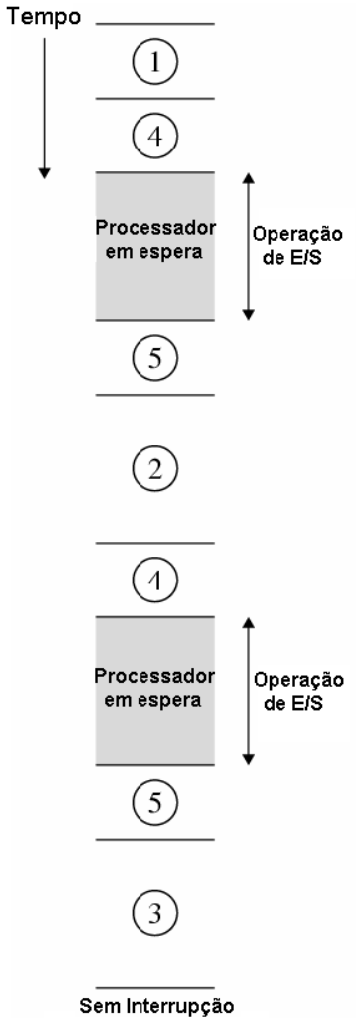


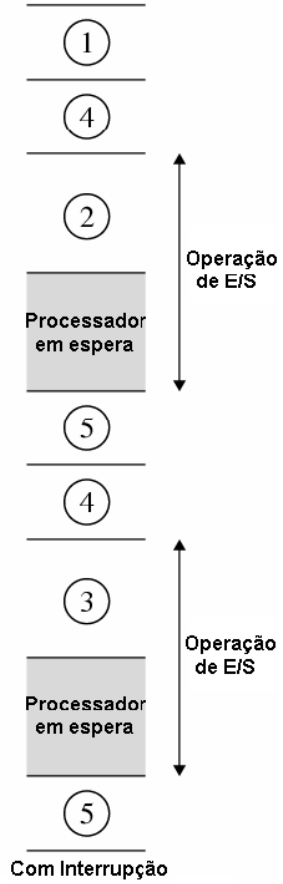




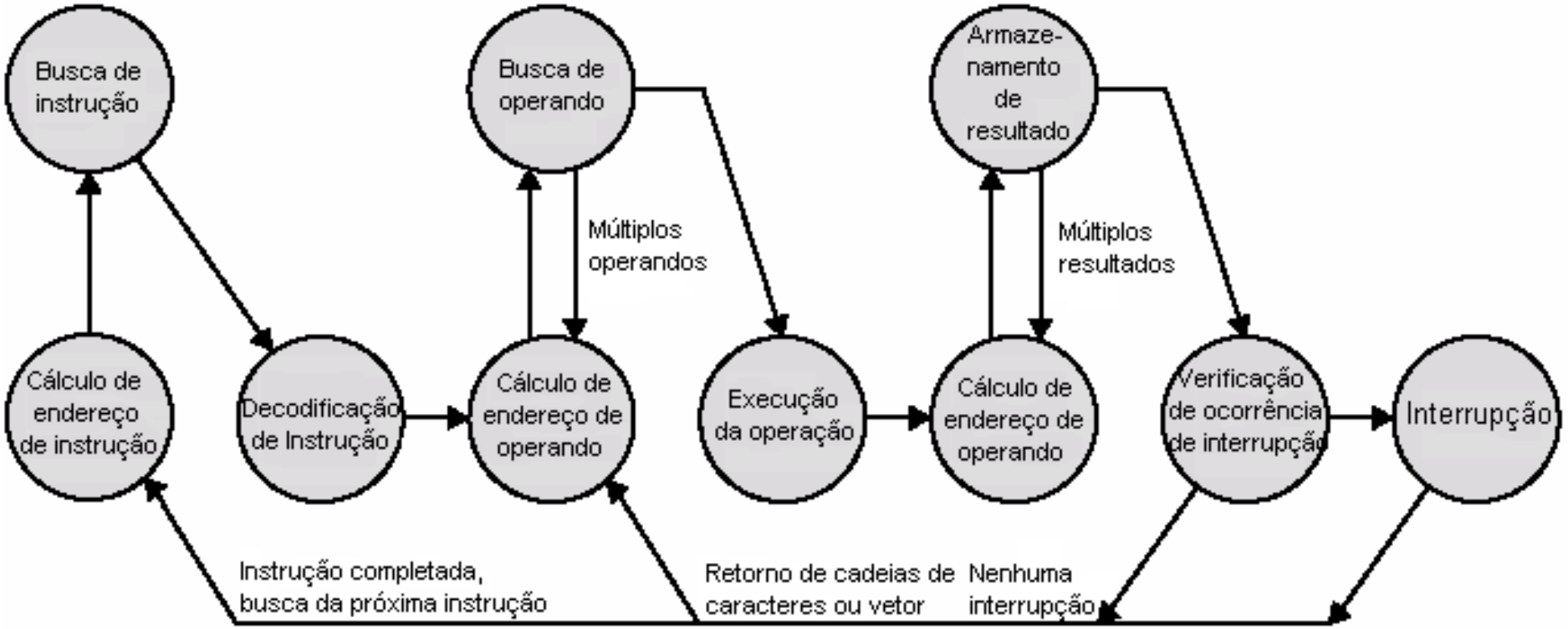
Ganho em eficiência







Ciclo de instrução com interrupção



Exercícios

Exercícios

1) Considere três máquinas A, B e C. As instruções da máquina A podem referenciar explicitamente 3 operandos, as instruções da máquina B podem referenciar 2 operandos e as da máquina C podem referenciar apenas 1 operando.

•Conjunto de instruções das máquinas A, B e C. Todas as instruções acessam as posições de memória representadas pelos rótulos X, Y e Z.

Máquina A	Máquina B	Máquina C	
ADD X, Y, Z	ADD X, Y	ADD X	Soma
SUB X, Y, Z	SUB X, Y	SUB X	Subtração
MPY X, Y, Z	MPY X, Y	MPY X	Multiplicação
DIV X, Y, Z	DIV X, Y	DIV X	Divisão
		LOAD X	Carrega registrador AC a partir da memória
		STOR X	Armazena na memória conteúdo de AC

INICIO	MPY	T, J, K
	ADD	A, I, T
	SUB	T, L, M
	DIV	A, A, T
A	DATA	0
I	DATA	50
J	DATA	20
K	DATA	15
L	DATA	30
M	DATA	5
T	DATA	0

2) Considerando o seguinte programa escrito em linguagem de montagem para a máquina A, responda as questões:

- a) Qual o valor contido na posição de memória A ao final da execução do programa?
- b) Reescreva o programa listado acima para que ele funcione na máquina B (instruções capazes de referenciar 2 operandos).
- c) Reescreva o programa listado acima para que ele funcione na máquina C (instruções capazes de referenciar 1 operando).

Exercícios

2) Escreva um programa em linguagem assembly (utilizando rótulos) considerando o conjunto de instruções simplificado apresentado na tabela do exercício anterior que execute as seguintes instruções de alto-nível. Para isso, considere que o conteúdo das posições de memória referenciadas por A, B, C, D e E armazenam os valores 20, 4, 52, 100 e 9 respectivamente.

- a) E = A + B / (C – D) * B;
- b) D = C * E + (A + B) / D;

3) Reescreva os programas escritos em linguagem de montagem (assembly) no exercício anterior para linguagem simbólica. Considere que a primeira instrução de cada programa seja "montada" na posição de memória 512 e que cada instrução ocupa exatamente 1 posição dessa memória. Os dados são armazenados na memória a partir da posição 640 considerando o rótulo das variáveis em ordem alfabética.

4)Reescreva os programas em linguagem simbólica do exercício 2 para em linguagem de máquina, porém em representação hexadecimal. O código de operação das instruções é:

Operação	Opcode (Hexadecimal)
ADD	1
SUB	2
MPY	3
DIV	4
LOAD	5
STOR	6

Exercícios

•5) Explique o funcionamento do ciclo básico de busca e execução de instruções, e qual é a sua relação com o funcionamento do computador.

•6) Explique a função dos registradores PC, IR, AC, MAR e MBR. Relaciona as funções desses registradores com o ciclo de busca e execução de instruções.

•7) A instrução mostrada na Figura faz parte de um Conjunto de Instruções em que o campo para o código da operação código da operação (Opcode) possui 4 bits, quantas instruções diferentes são suportadas por essa arquitetura.

–E se o opcode das instruções possuísse 8 bits?