Organização de computadores

Prof. Moisés Souto

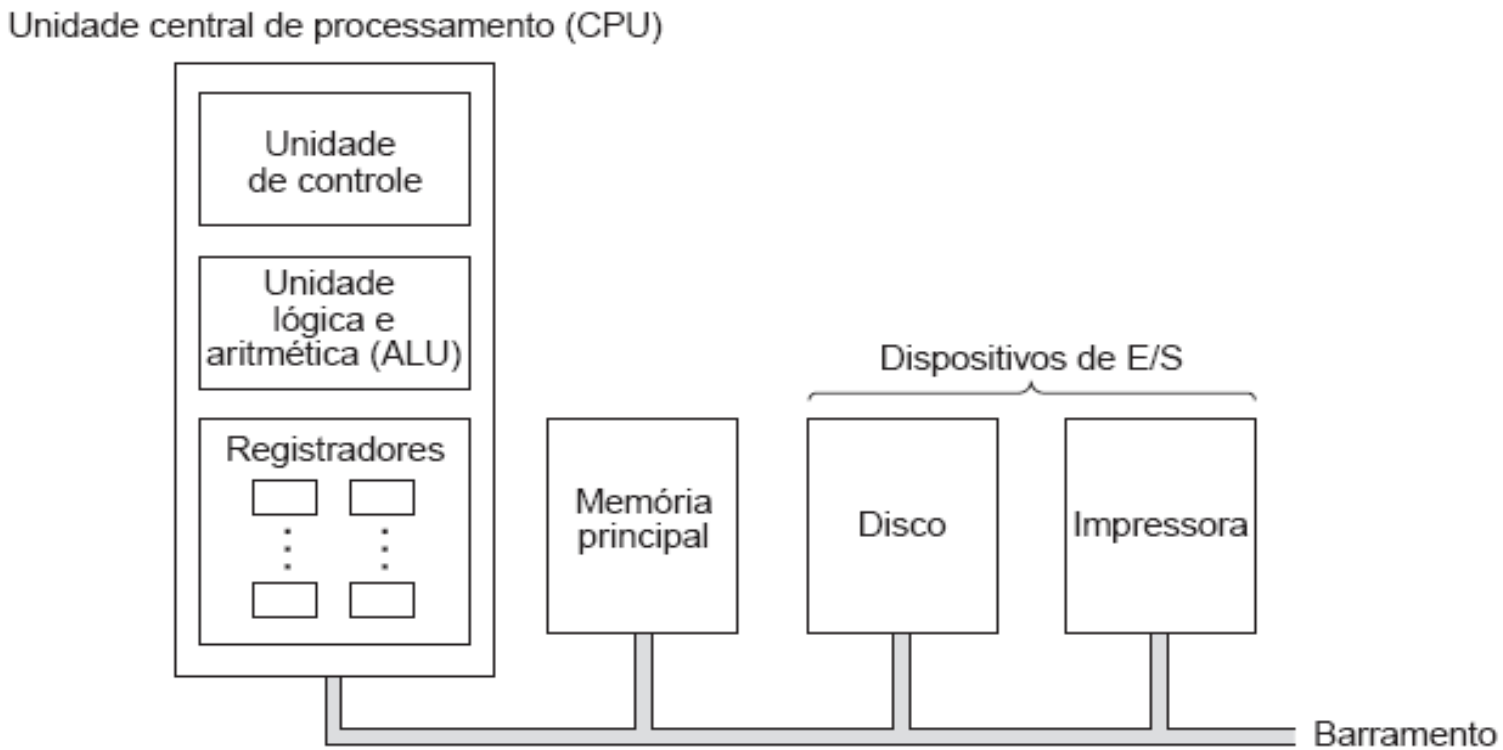
Moisés Souto
docente.ifrn.edu.br/moisessouto

professor.moisessouto.com.br
moises.souto@ifrn.edu.br
@moises_souto

Aula 06

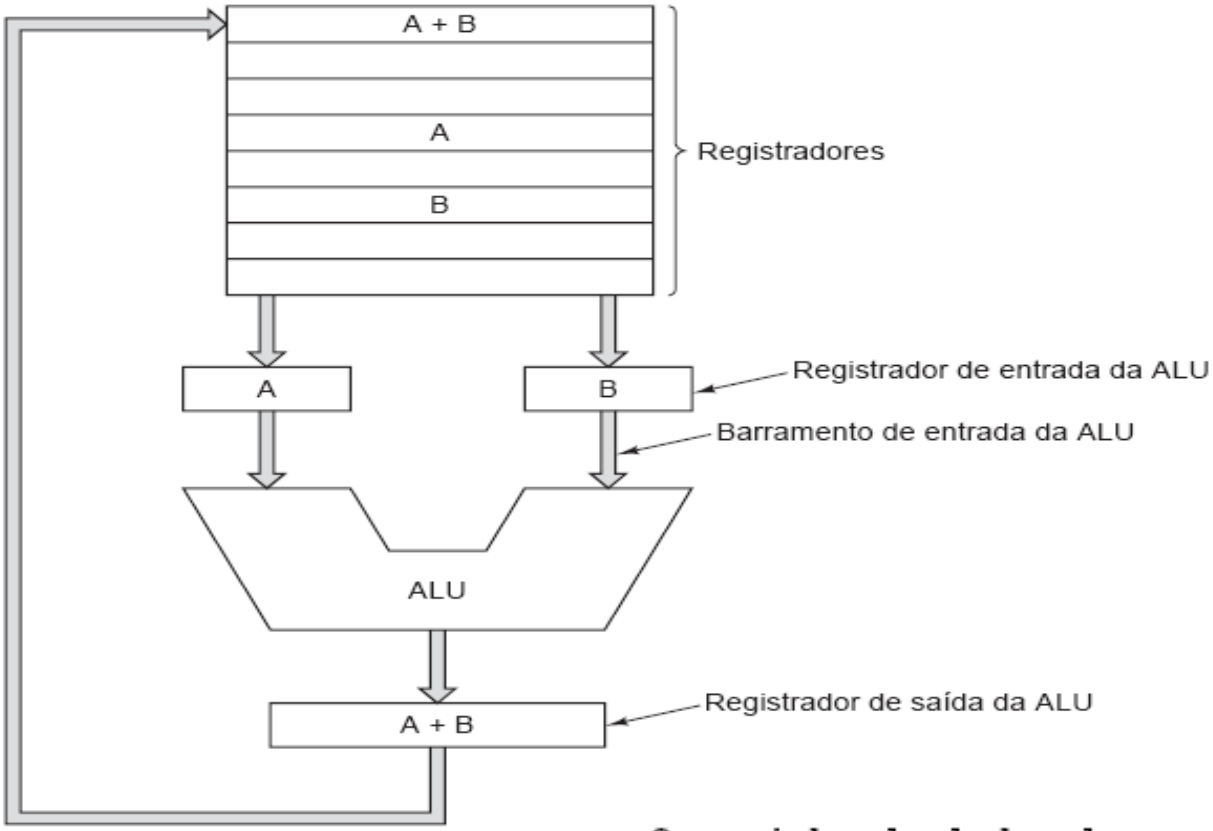
CICLO DE INSTRUÇÕES

Unidade Central de Processamento



A organização de um computador simples com uma CPU e dois dispositivos de E/S.

Organização da CPU

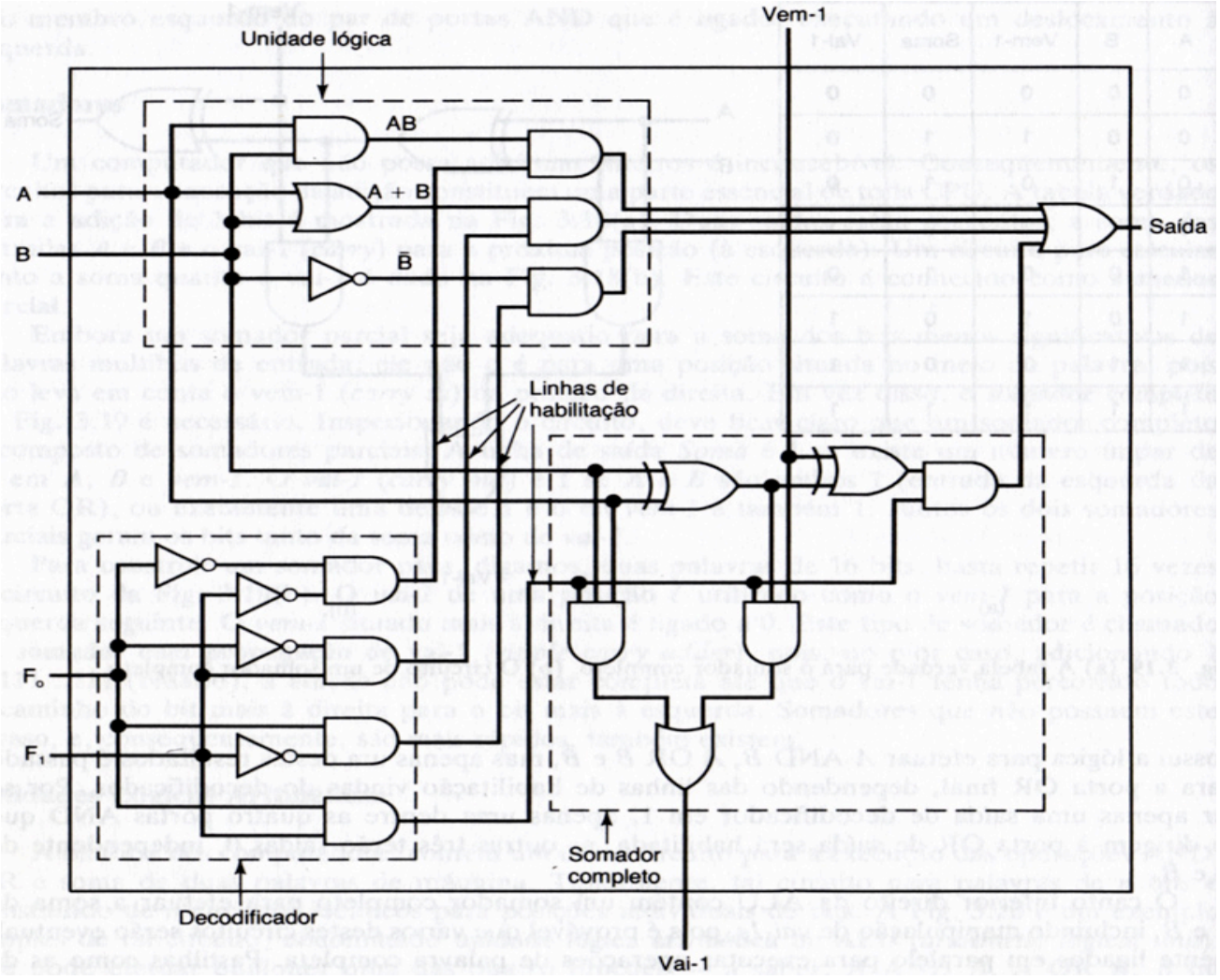


O caminho de dados de uma típica máquina de Von Neumann.

Conceito de Programa

- Sistemas baseados em hardware são inflexíveis
- Um hardware de uso geral pode executar diferentes tarefas de acordo com sinais de controle
- Ao invés de alterar o hardware, altera-se apenas o conjunto de sinais de controle

Unidade Lógica e Aritmética



O que é um programa?

- Uma seqüência de passos
- Para cada passo, uma operação lógica ou aritmética é realizada
- Para cada operação um diferente conjunto de sinais de controle é utilizado

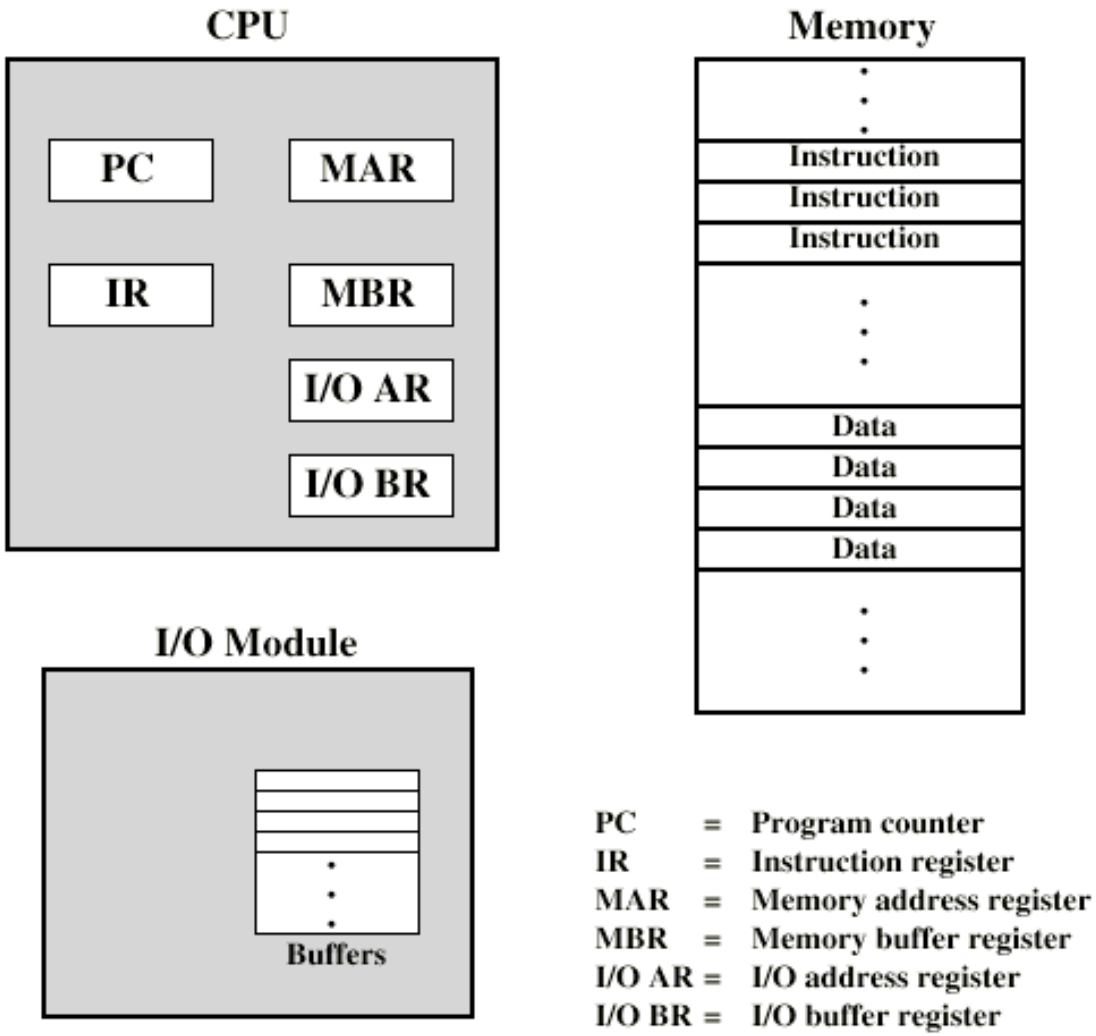
Função da Unidade de Controle

- Para cada operação um código único é fornecido
  - ex.: ADD, MOVE
- Um circuito de hardware aceita o código e emite os sinais de controle
- Pronto: Temos um Computador!

Componentes

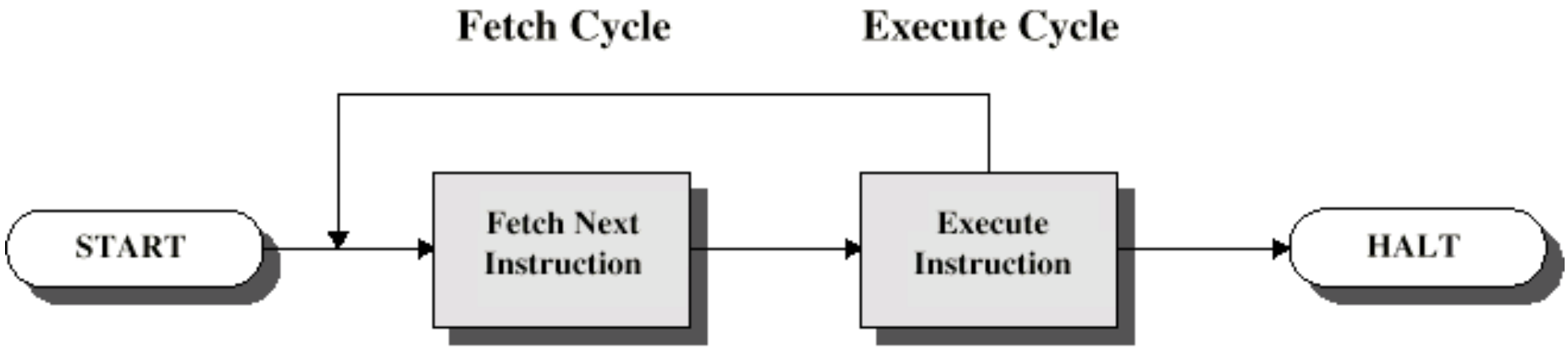
- A Unidade de Controle e a Unidade Lógica e Aritmética constituem a Unidade Central de Processamento – CPU.
- Dados e instruções precisam chegar ao sistema e resultados precisam sair.
  - Entrada/Saída
- É necessário armazenamento temporário do código, dados e resultado.
  - Memória

Componentes do Computador: Visão de Alto Nível



Ciclo de Instrução

- Dois Passos:
  - Busca, Captura - Fetch
  - Execução - Execute



Execução de instrução

1. Trazer a próxima instrução da memória até o registrador.
2. Alterar o contador de programa para indicar a próxima instrução.
3. Determinar o tipo de instrução trazida.
4. Se a instrução usar uma palavra na memória, determinar onde essa palavra está.
5. Trazer a palavra para dentro de um registrador da CPU, se necessário.
6. Executar a instrução.
7. Voltar à etapa 1 para iniciar a execução da instrução seguinte.

Ciclo de Captura

- O Contador de Programa (PC) tem o endereço da próxima instrução a executar.
- O Processador captura a instrução da memória na posição apontada pelo PC
- A UC incrementa o PC
  - A menos que seja instruído ao contrário
- A instrução é colocada no Registrador de Instruções (IR)
- O Processador interpreta a instrução e executa as ações requisitadas

Ciclo de Execução

- Processador -> Memória
  - Transferência de dados entre CPU e Memoria
- Processador -> I/O
  - Transferência de dados entre CPU e I/O
- Processamento de dado
  - Operações lógicas ou aritméticas
- Controle
  - Alteração da seqüência de operações
  - ex.: jump (desvio)
- Combinações das tarefas acima

Exemplo de Execução de Programa

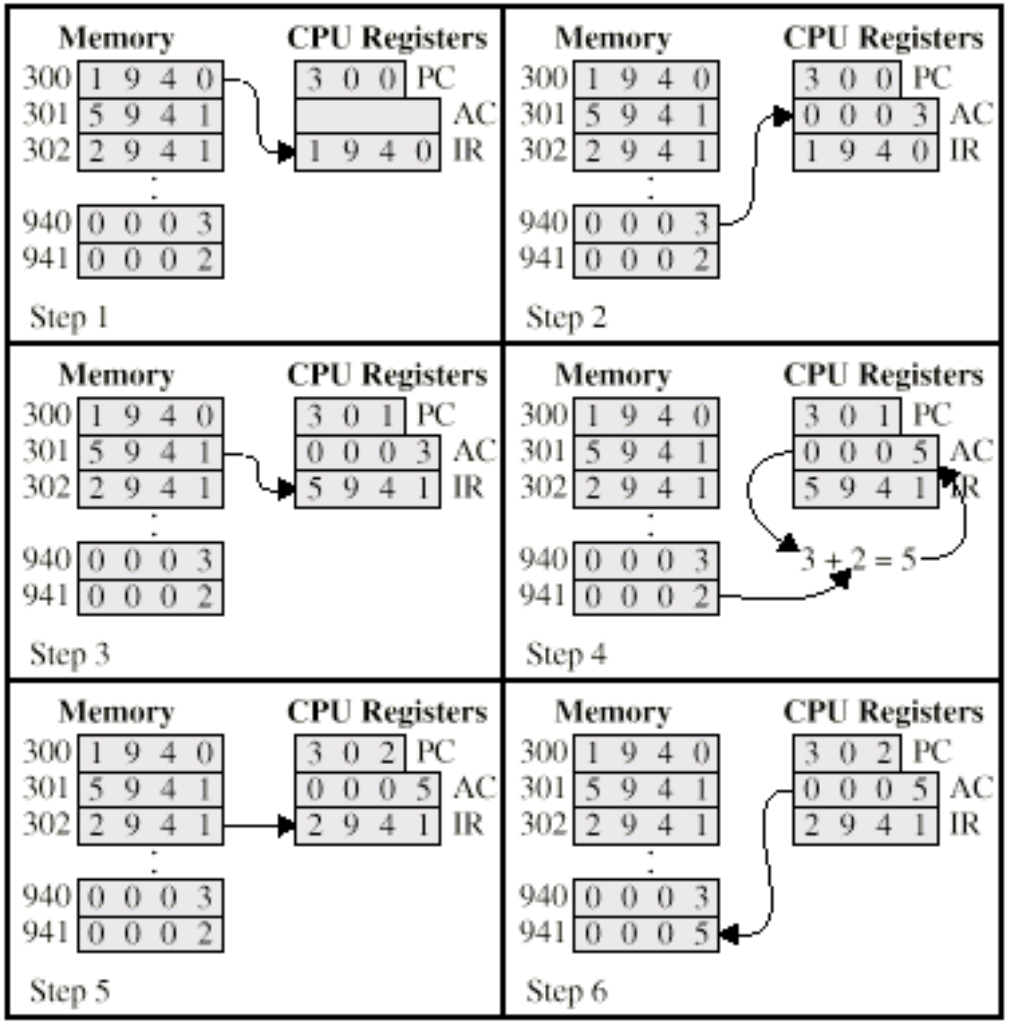
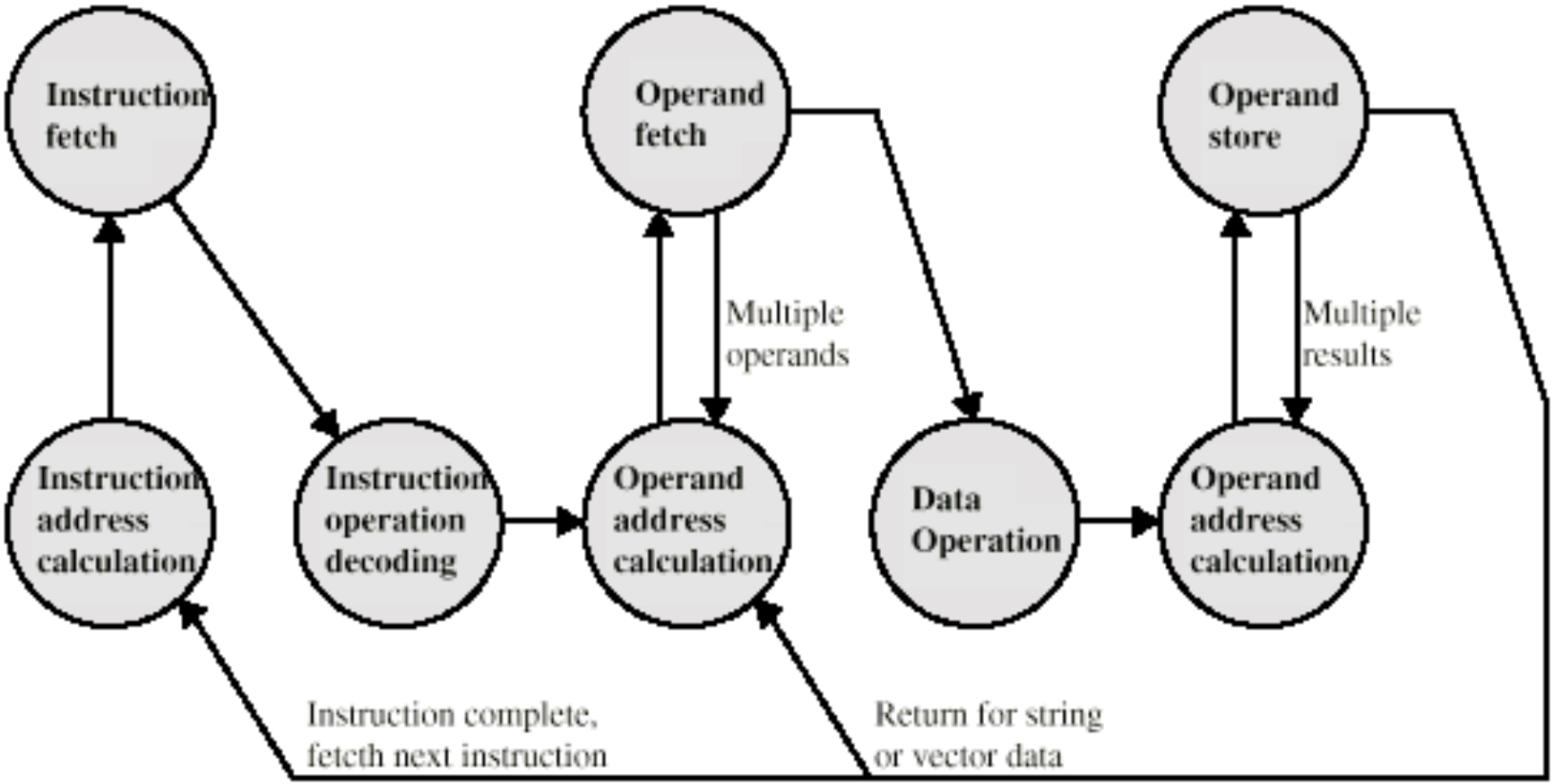


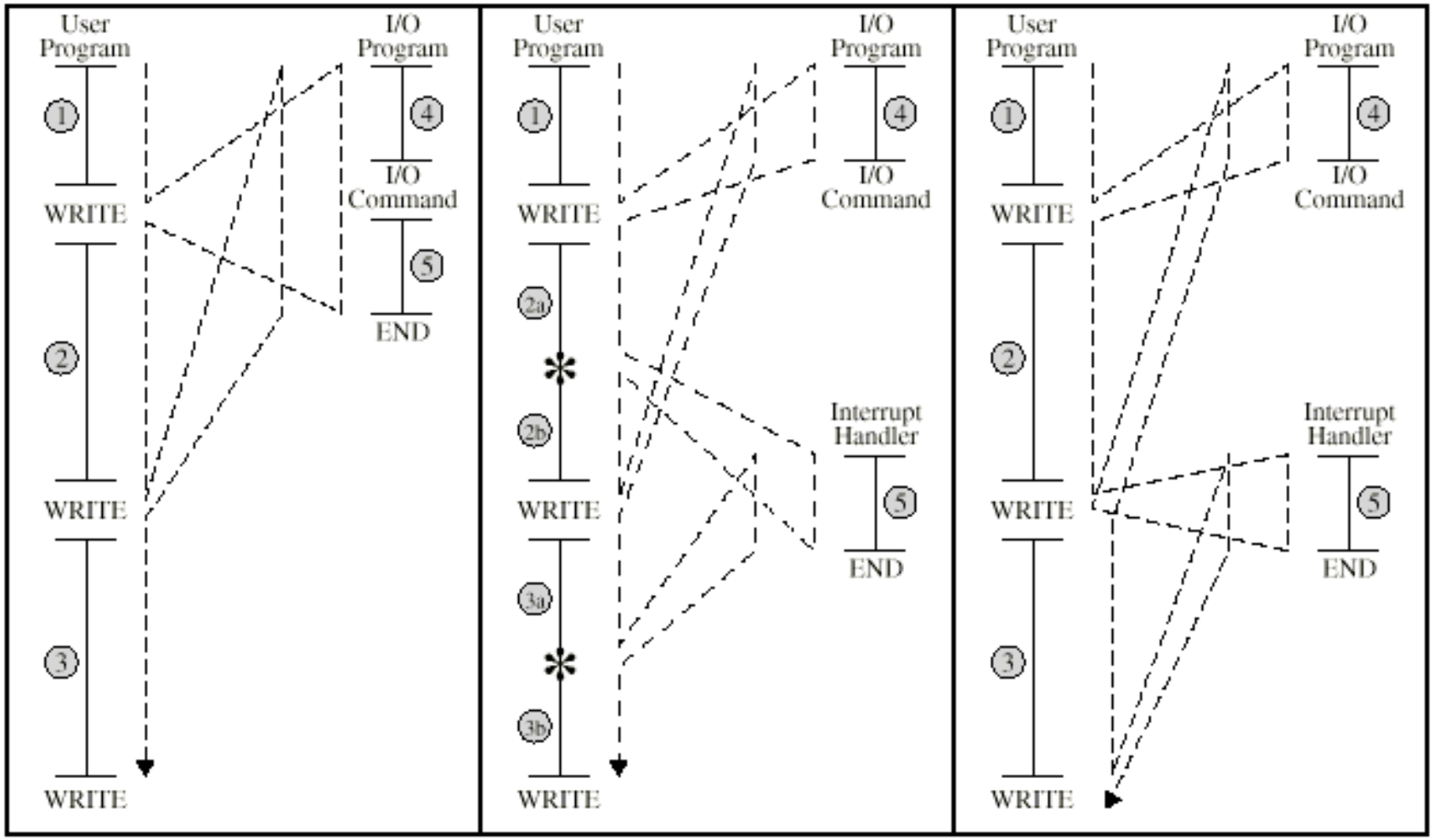
Diagrama de Estados do Ciclo de Instrução



Interrupções

- Mecanismo pelo qual outros módulos (ex.: I/O) podem interromper a seqüência normal do processamento.
- Programa
  - ex.: overflow, divisão por zero
- Timer
  - Gerado pelo temporizador interno do processador
  - Usado em sistemas multi-tarefa pre-emptivos
- I/O
  - Do controlador de I/O
- Falha de Hardware
  - ex.: erro de paridade de memória

Controle do Fluxo do Programa



(a) No interrupts

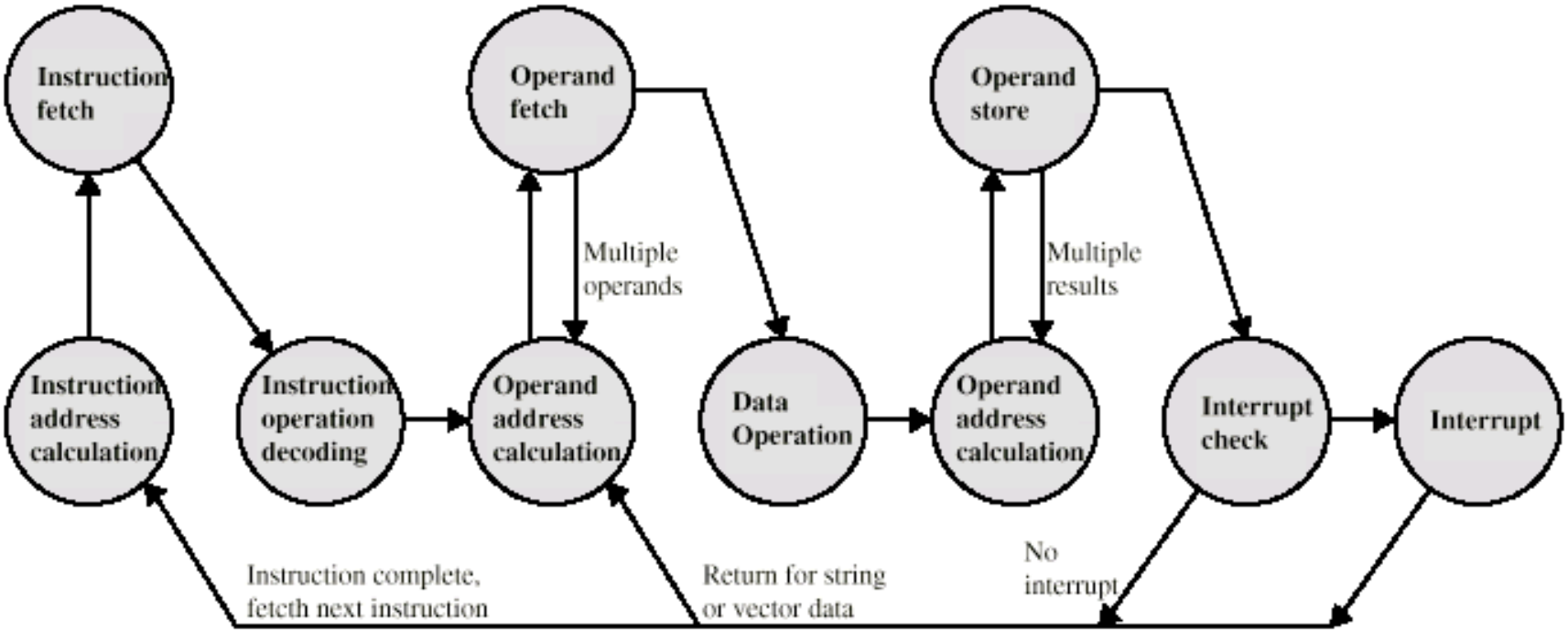
(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

Ciclo de Interrupção

- Adicionado ao Ciclo de Instrução
- O Processador verifica se houve pedido de interrupção.
  - Indicado por um bit sinalizador
- Se não houve interrupção captura a próxima instrução.
- Se existe interrupção pendente:
  - Suspende a execução do programa em andamento
  - Salva o contexto
  - Carrega o PC com o start address da rotina de interrupção
  - Processa a interrupção
  - Restaura o contexto e continua o programa interrompido

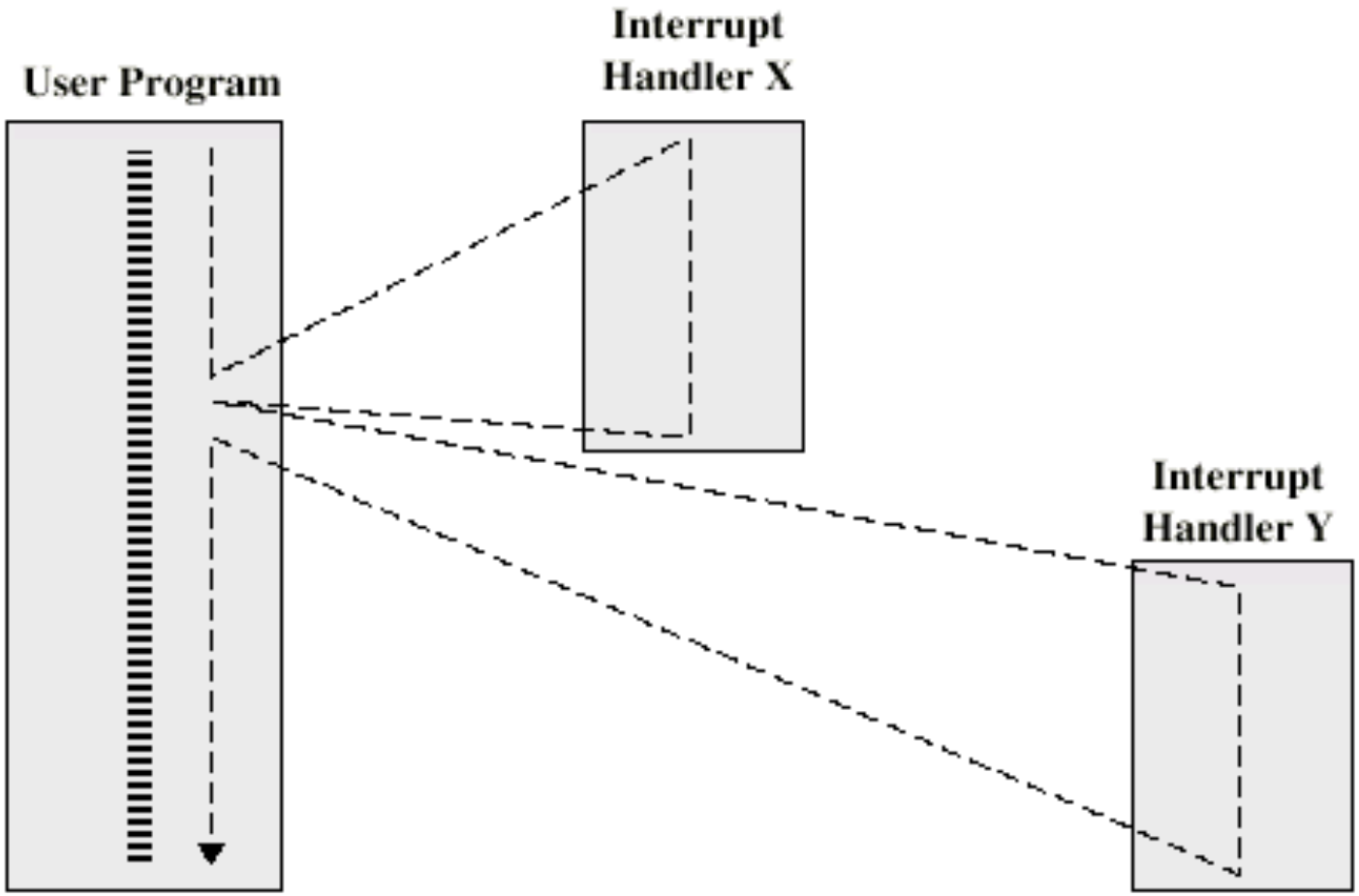
Diagrama de Estados do Ciclo de Instrução (com interrupções)



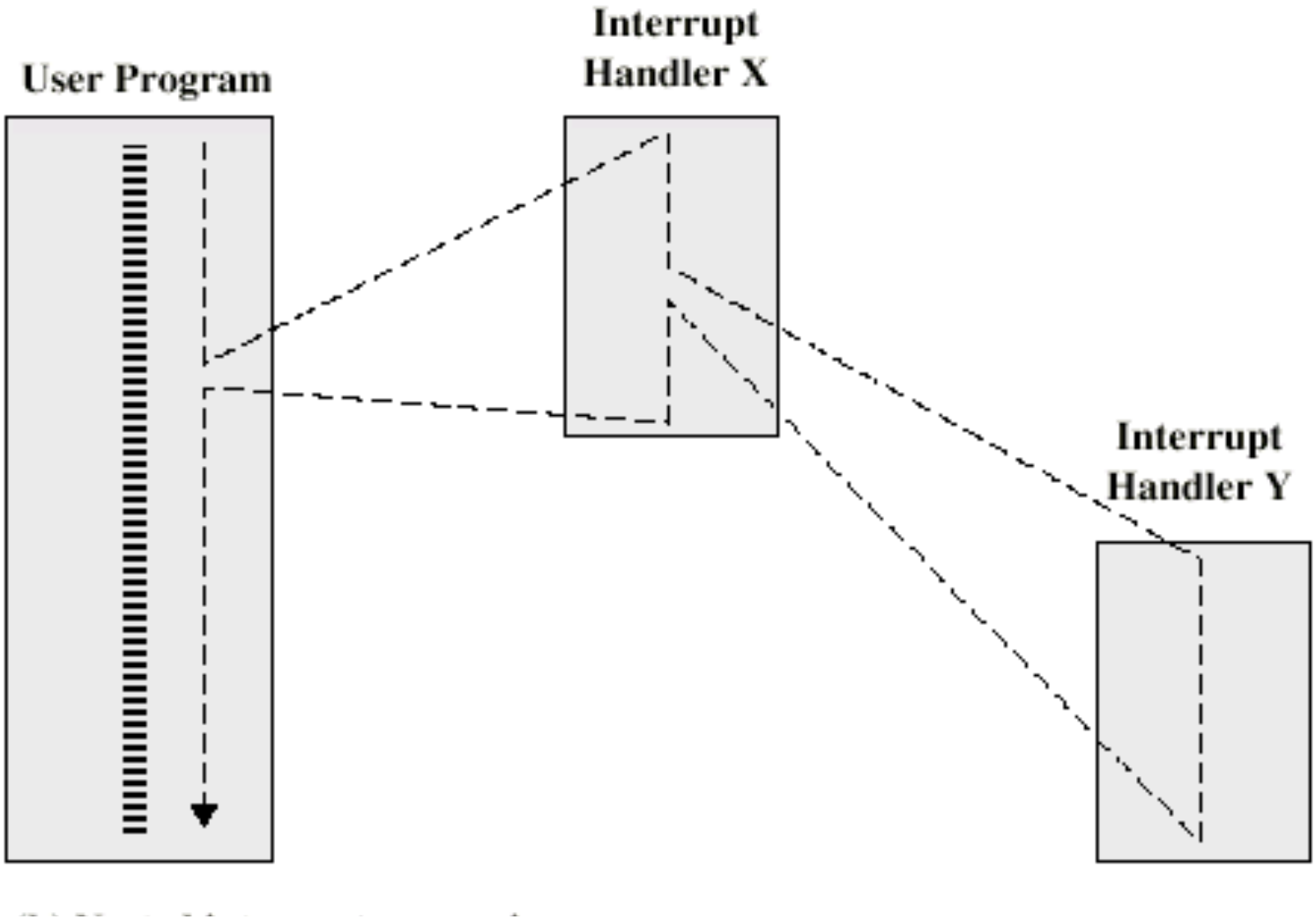
Interrupções Múltiplas

- Com Desabilitação Interrupções
  - O processador irá ignorar outras interrupções enquanto estiver processando uma interrupção
  - As interrupções ficarão pendentes e serão atendidas após o término da interrupção em curso
  - Interrupções pendentes são atendidas na seqüência em que ocorreram
- Com Definição Prioridades
  - Interrupções de menor prioridade podem ser interrompidas por interrupções de maior prioridade
  - Quando uma interrupção de maior prioridade acaba de ser processada o processador retorna à interrupção anterior

Interrupções Múltiplas Seqüenciadas



Interrupções Múltiplas Aninhadas



Interpretador (1)

```
public class Interp {
  static int PC;                    // contador de programa contém endereço da próxima instr
  static int AC;                    // o acumulador, um registrador para efetuar aritmética
  static int instr;                 // um registrador para conter a instrução corrente
  static int instr_type;            // o tipo da instrução (opcode)
  static int data_loc;              // o endereço dos dados, ou -1 se nenhum
  static int data;                  // contém o operando corrente
  static boolean run_bit = true;    // um bit que pode ser desligado para parar a máquina

  public static void interpret(int memória[ ], int starting_ address) {
    // Esse procedimento Interpretador para um computador simples (escrito em Java).
    // um operando da memória. A máquina tem um registrador AC (acumulador), usado para
    // aritmética. A instrução ADD soma um inteiro na memória do AC, por exemplo.
    // O interpretador continua funcionando até o bit de funcionamento ser desligado pela instrução HALT.
    // O estado de um processo que roda nessa máquina consiste em memória,
    // contador de programa, bit de funcionamento e AC. Os parâmetros de entrada consistem
///
    // na imagem da memória e no endereço inicial.
```

Interpretador para um computador simples (escrito em Java).

Interpretador (2)

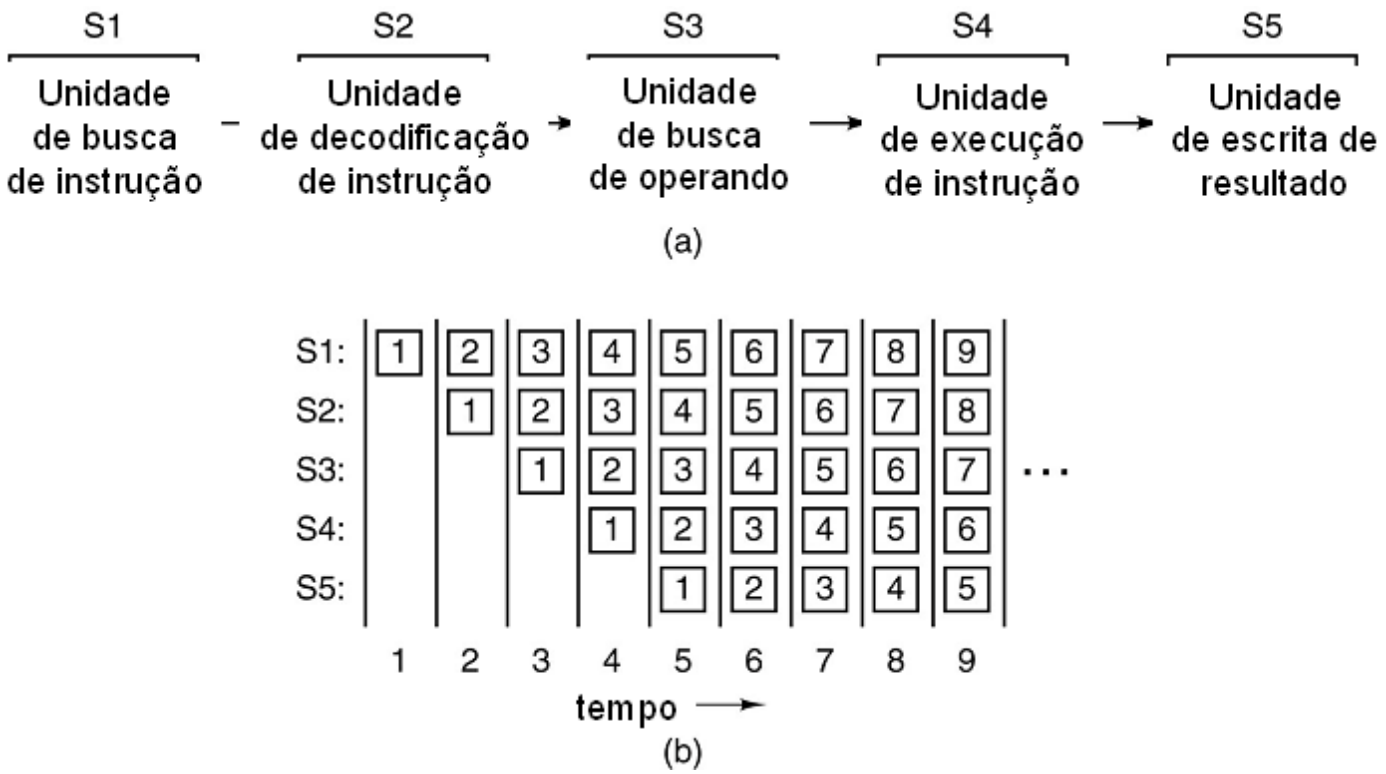
```
PC = starting_address;
while (run_bit) {
   instr = memory[PC];                          // carrega próxima instrução em instr
   PC = PC + 1;                                 // incrementa contador de programa
   instr_type = get_instr_type(instr);          // determina tipo de instrução
   data_loc = find_data(instr, instr_type);     // localiza dados (-1 se não houver)
   if (data_loc >= 0)                           // se data_loc é –1 não há operandos
      data = memory[data_loc];                  // busca os dados
   execute(instr_type, data);                   // executa instrução
}

}

private static int get_instr_type(int addr) { ... }
private static int find_data(int instr, int type) { ... }
private static void execute(int type, int data) { ... }
}
```

Interpretador para um computador simples (escrito em Java).

Princípios de Projeto para Computadores Modernos

- Todas as instruções são diretamente executadas pelo hardware
- Maximiza a taxa na qual as instruções são executadas
- Instruções devem ser fáceis de decodificar
- Somente leituras e armazenamentos devem referenciar a memória
- Fornece vários registradores

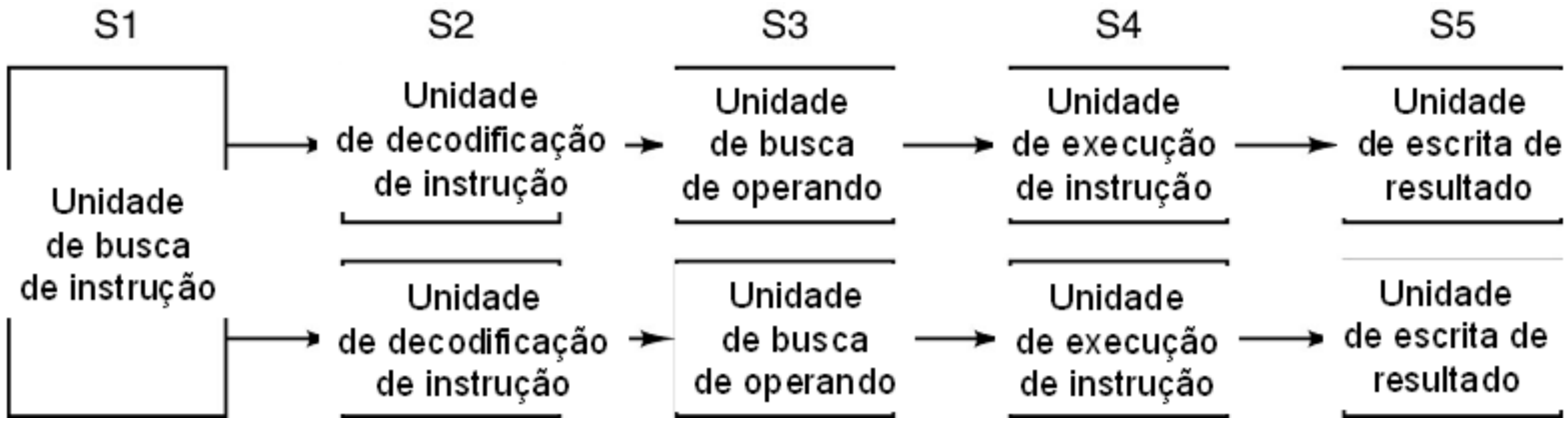
Paralelismo no Nível de Instrução



a) Pipeline de cinco estágios.

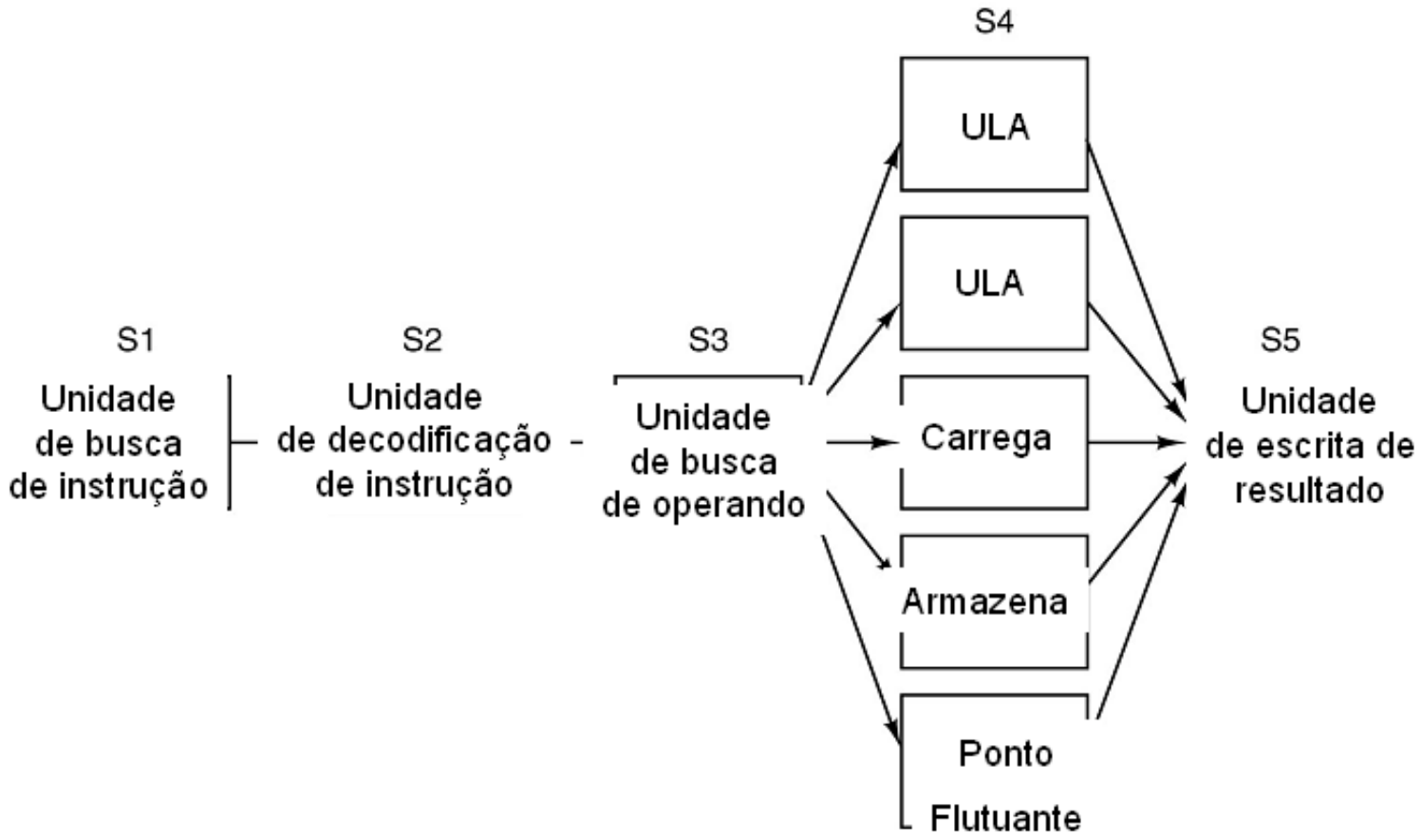
b) Estado de cada estágio em função do tempo. São ilustrados nove ciclos de relógio.

Arquiteturas Superescalares (1)



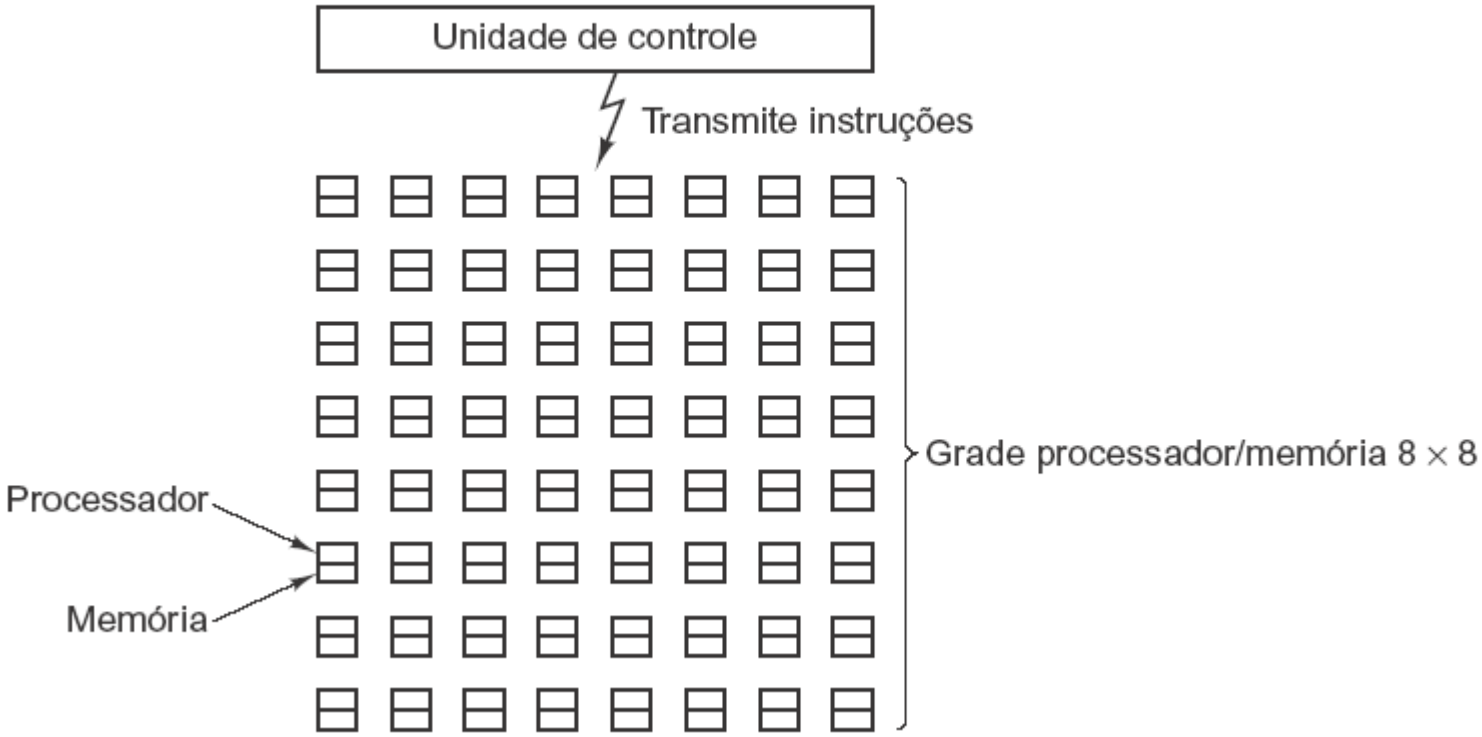
Pipelines duplos de cinco estágios com uma unidade de busca de instrução em comum.

Arquiteturas Superescalares (2)



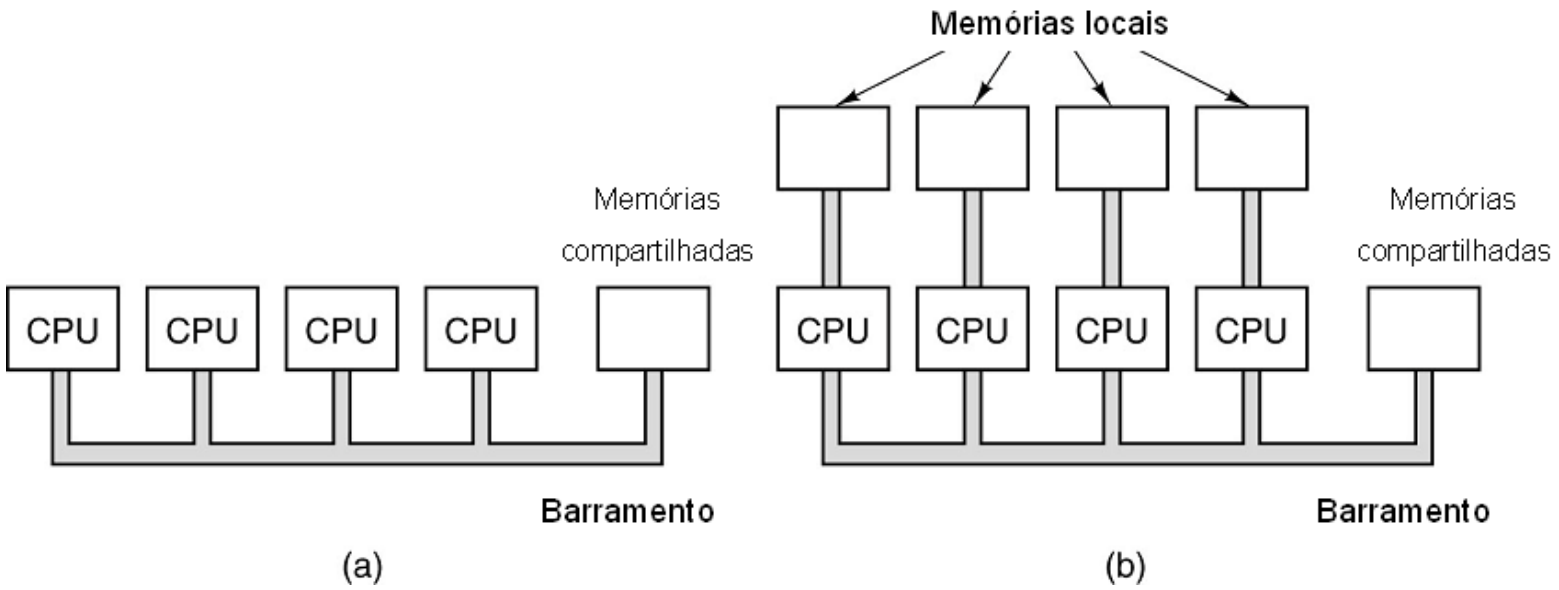
Processador superescalar com cinco unidades funcionais.

Paralelismo de Processador (1)



Processador matricial do tipo ILLIAC IV.

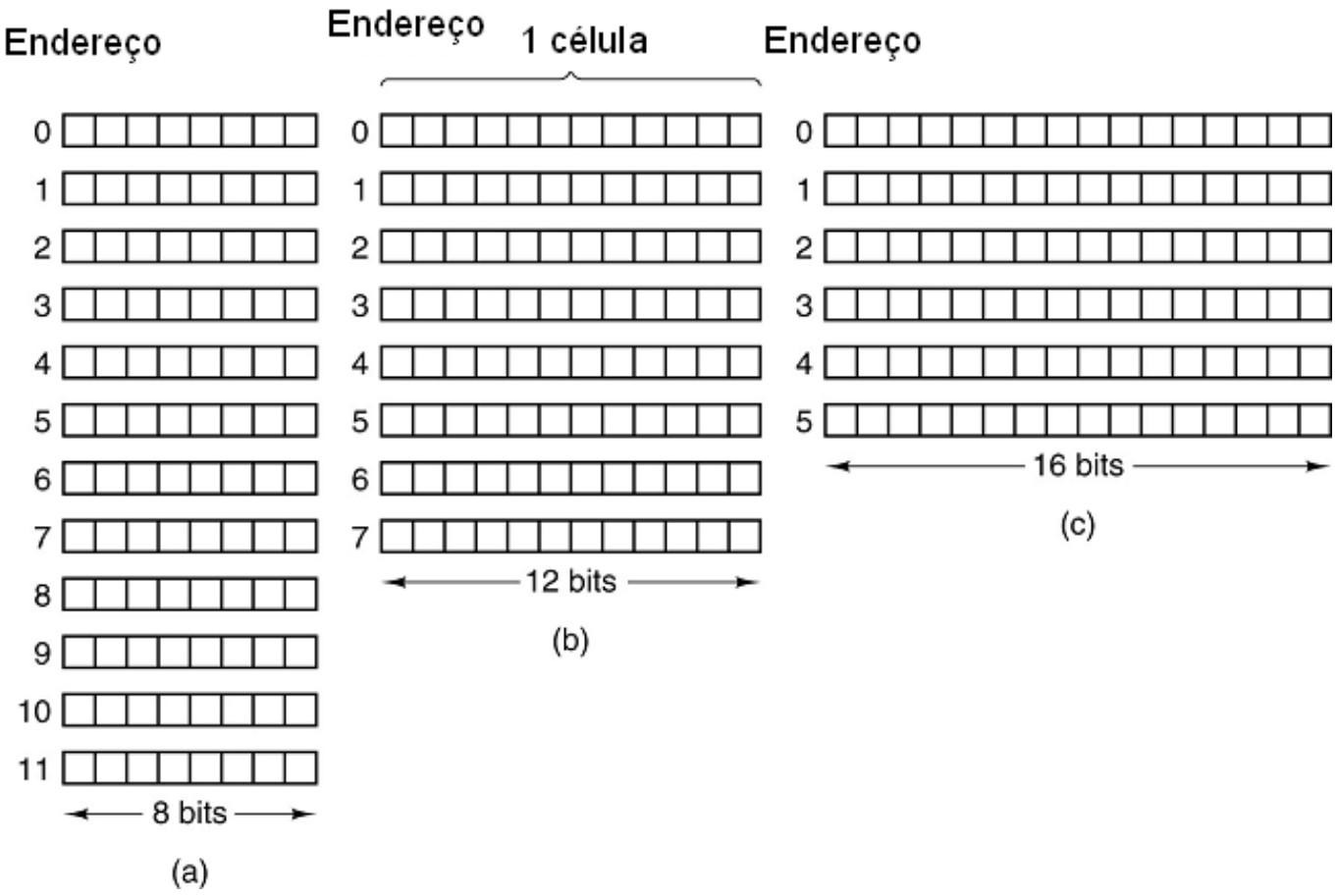
Paralelismo de Processador (2)



a) Multiprocessador com barramento único.
b) Multicomputador com memórias locais.

Considerações sobre acesso a memória

Memória Primária
Endereços de Memória (1)



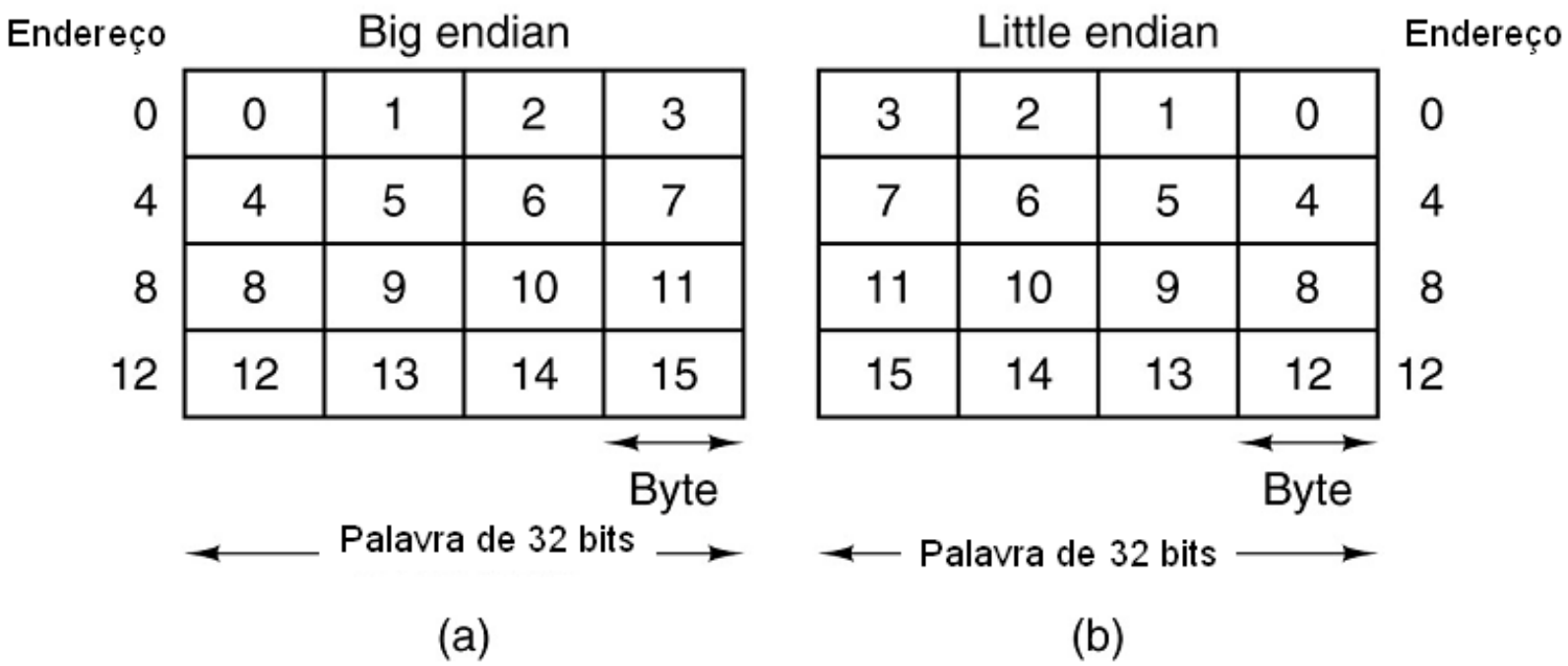
Três maneiras de organizar uma memória de 96 bits.

Memória Primária
Endereços de Memória (2)

Computador	Bits/célula
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Electrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

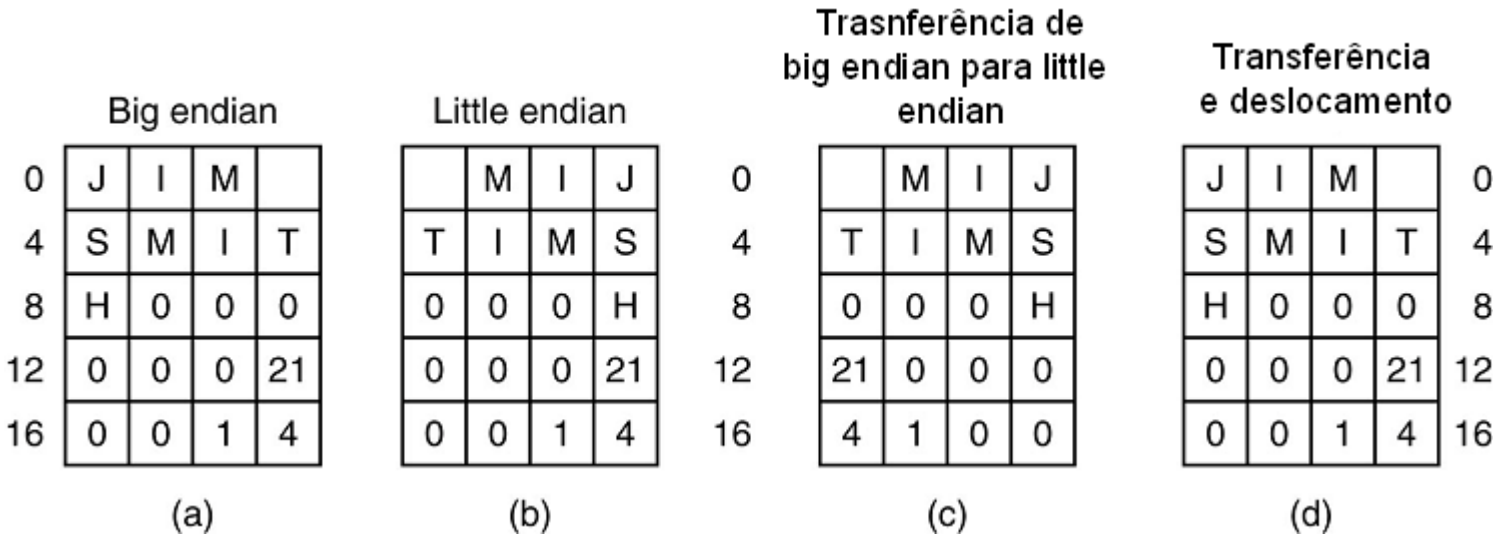
Número de bits por célula para alguns computadores comerciais historicamente interessantes.

Ordenação de bytes (1)



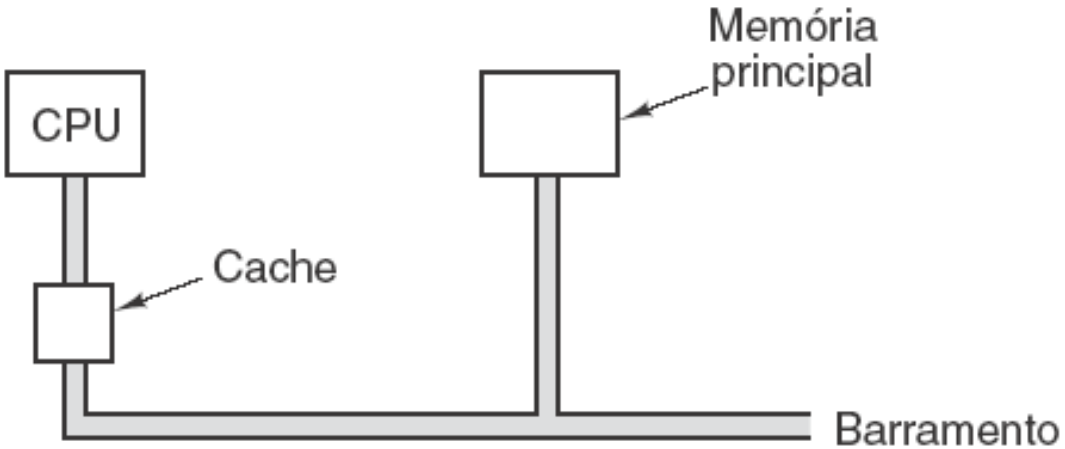
(a) Memória *big endian* (b) Memória *little endian*

Ordenação de Byte (2)



(a) Registro pessoal para uma máquina *big endian*.
(b) O mesmo registro para uma máquina *little endian*.
(c) Resultado da transferência de *big endian* para *little endian*.
(d) Resultado do deslocamento de bytes (c).

Memória Cache



O cache localiza-se logicamente entre a CPU e a memória principal. Fisicamente há vários locais onde ela pode ser colocada.

Conjunto de Caracteres ASCII (1)

Hex	Nome	Significado	Hex	Nome	Significado
0	NUL	Null	10	DLE	Data Link Escape
1	SOH	Start Of Heading	11	DC1	Device Control 1
2	STX	Start Of Text	12	DC2	Device Control 2
3	ETX	End Of Text	13	DC3	Device Control 3
4	EOT	End Of Transmission	14	DC4	Device Control 4
5	ENQ	Enquiry	15	NAK	Negative AcKnowledgement
6	ACK	ACKnowledgement	16	SYN	SYNchronous idle
7	BEL	BELl	17	ETB	End of Transmission Block
8	BS	BackSpace	18	CAN	CANcel
9	HT	Horizontal Tab	19	EM	End of Medium
A	LF	Line Feed	1A	SUB	SUBstitute
B	VT	Vertical Tab	1B	ESC	ESCape
C	FF	Form Feed	1C	FS	File Separator
D	CR	Carriage Return	1D	GS	Group Separator
E	SO	Shift Out	1E	RS	Record Separator
F	SI	Shift In	1F	US	Unit Separator

Conjunto de caracteres ASCII: caracteres 0 – 31.

Conjunto de Caracteres ASCII (2)

Hex	Car.	Hex	Car.	Hex	Car.	Hex	Car.	Hex	Car.	Hex	Car.
20	(Space)	30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

Conjunto de caracteres ASCII: caracteres 32 – 127.

PEARSON Prentice Hall

REFERÊNCIAS

Notas de aula. Arquitetura e organização de computadores. Glaucus Brelaz.

**Slides do livro Organização Estruturada de Computadores
Andrew S. Tanenbaum**

**Arquitetura e Organização de Computadores.
William Stallings**

**Organização Estruturada de Computadores
Andrew S. Tanenbaum**

Obrigado

Moisés Souto
professor.moisessouto.com.br
moises.souto@ifrn.edu.br