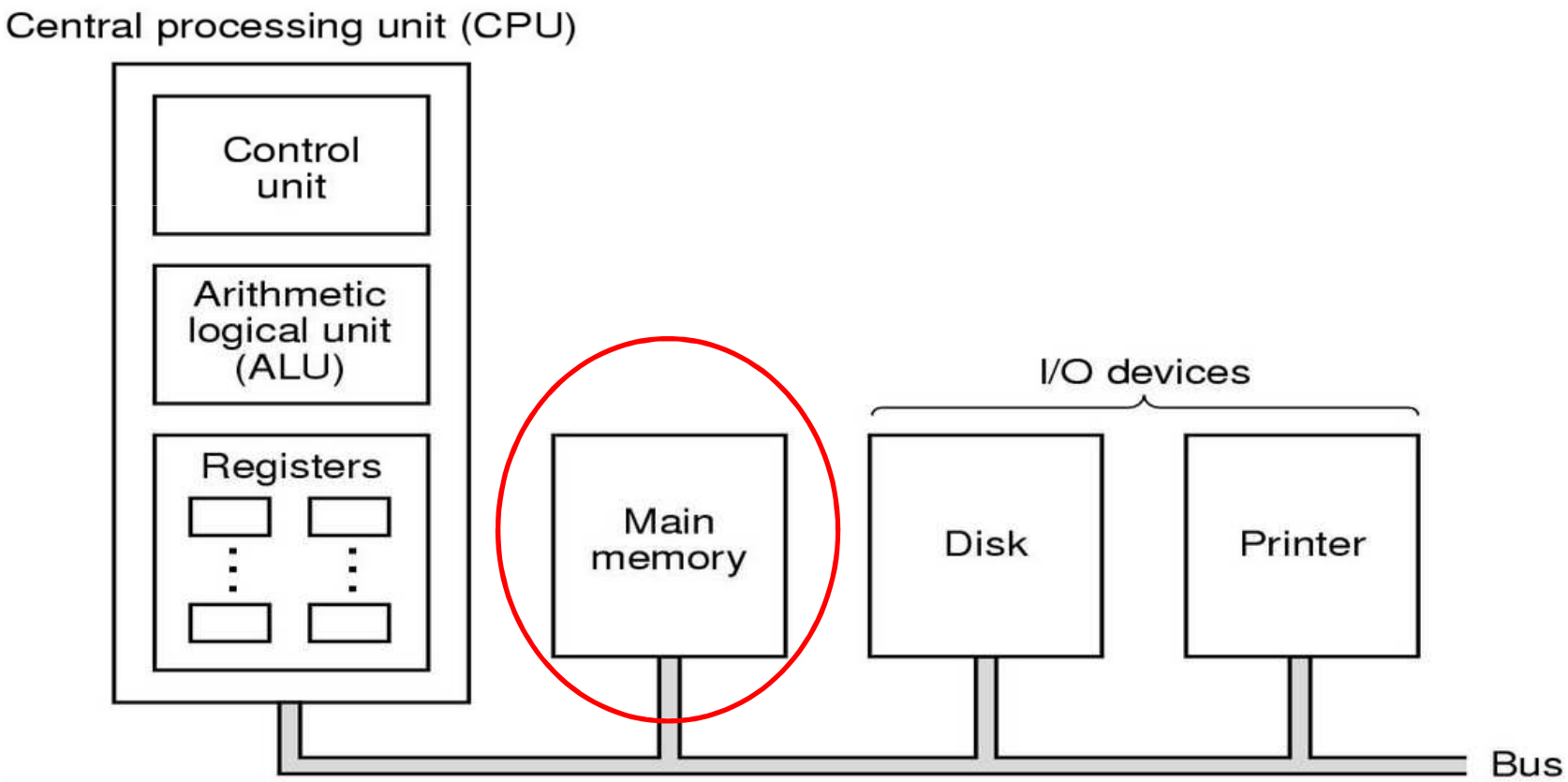
Memória Primária

Prof. Tiago Gonçalves Botelho

Hierarquia de Memória

Localização da memória primária



Memória Primária

- Armazena programa e dados.

BIT

- É a unidade básica da memória;
- Dígito binário: compreende os valores 0 e 1 bit;
- Em um byte podemos armazenar $2^8 = 256$ combinações diferentes, em 2 bytes, $2^{16} = 65536$, e assim por diante.

BIT

- Existem empresas que anunciam seus computadores com aritmética decimal, bem como binária, código BCD.

Ex.: representação do número1944 usando 16 bits.

Decimal: 0001 1001 0100 0100

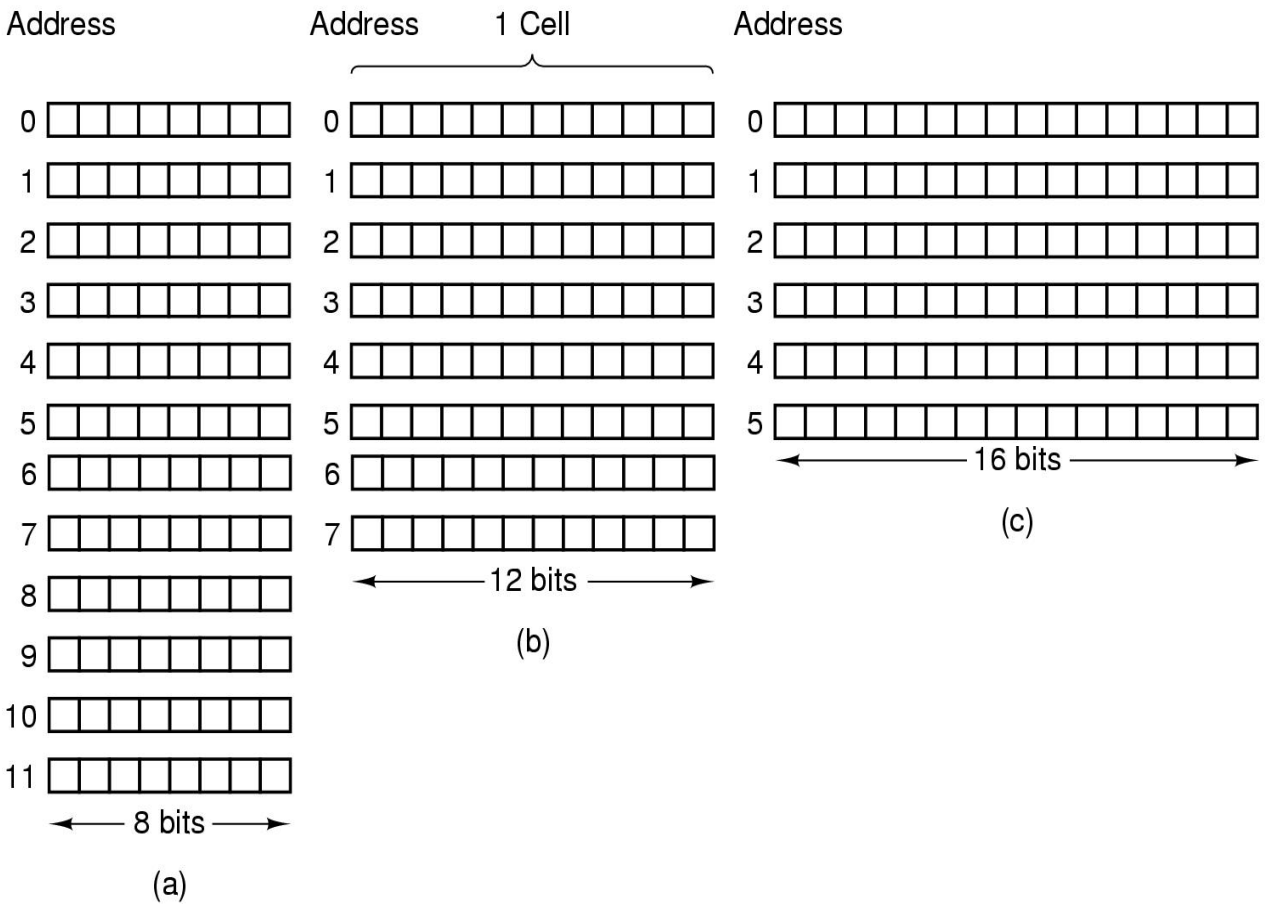
Binário: 0000011111001 1000

Qual representação é mais eficiente?

Endereços de memória

- A memória é composta por células que podem armazenar informações;
- Cada célula tem 1 número que é seu endereço;
  - Se a memória tiver n células, elas terão endereços de 0 a n-1.

3 Maneiras de organizar uma memória de 96 bits



Bits por célula para alguns computadores comerciais

Computador	Bits/célula
Burroughs B 1700	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

Palavras

- Célula é a menor unidade endereçável da memória;
- Os bits são agrupados em bytes que são também agrupados e chamados de **palavras**:
  - Um computador com uma palavra de 32 bits tem 4 bytes por palavra.
- Significância: grande parte das instruções efetuam operações com palavras inteiras:
  - Uma máquina de 64 bits terá registradores de 64 bits e instruções para manipular palavras de 64 bits.

Ordenação de Bytes

- Ordenação Big Endian
  - bytes são numerados da esquerda para a direita 0,1,2,...,n-1
  - usadas por sistemas Unix (arquiteturas SPARC, IBM Mainframe)
  - exemplo numérico com 4 bytes: representação numérica do número decimal 6

0	1	2	3
00000000	00000000	00000000	00000110

Ordenação de Bytes

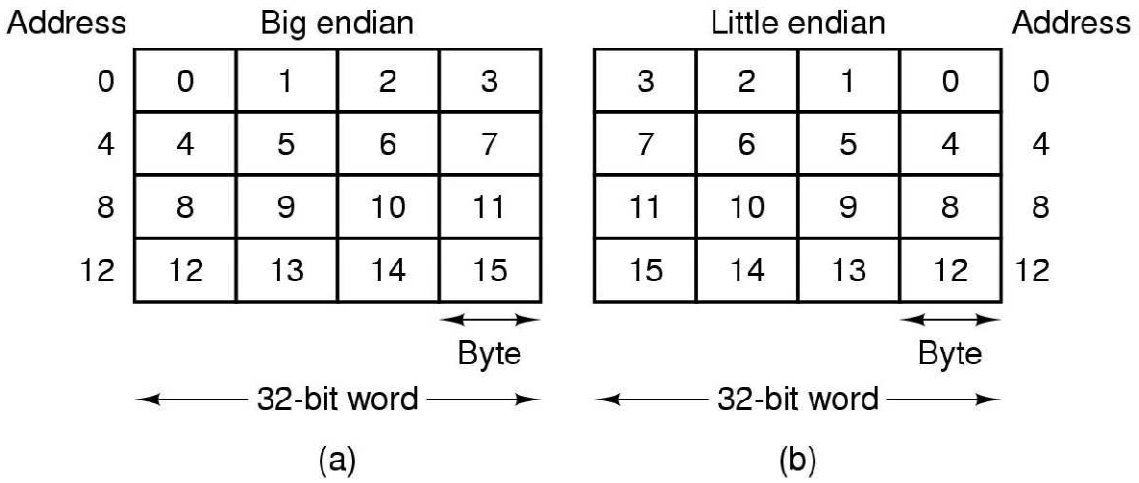
- Ordenação Little Endian
  - bytes são numerados da direita para esquerda n-1, ...,2,1,0
  - usado por IBM PCs (arquiteturas INTEL)
  - exemplo numérico com 4 bytes: representação numérica do número decimal 6

3	2	1	0
00000000	00000000	00000000	00000110

Ordenação de Bytes

(a) Memória big endian

(b) Memória little endian



Ordenação de Bytes

- Exemplo: Considere um simples registro de pessoal composto de uma cadeia e de dois inteiros:
  - Nome do Empregado = “Jim Smith” ;
  - Idade=21;
  - Número do Departamento=260.

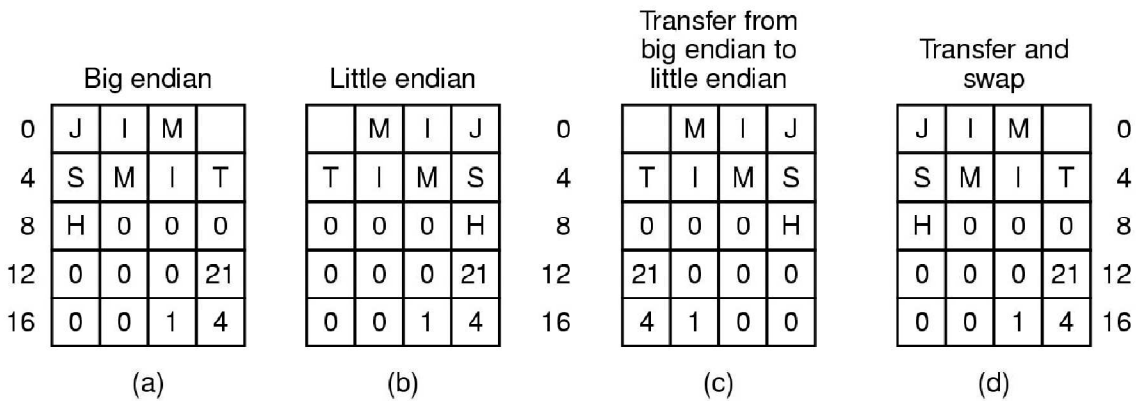
Ordenação de Bytes

a) Registro de pessoal para uma máquina big endian.

b) O mesmo registro para uma máquina little endian.

c) Resultado da transferência do registro de uma máquina big endian para uma little endian.

d) Resultado da troca de bytes



Ordenação de Bytes

- Solução possível:

Um modo que funcionaria, mas de maneira ineficiente, é incluir um cabeçalho informando o tipo de dado (cadeia, inteiro, ou outro) e seu comprimento. Assim o destinatário faria as conversões necessárias.

Códigos de correção de erros

- Memórias de computador podem cometer erros de vez em quando devido a picos de tensão na linha elétrica ou por outras causas.

  Solução: código de detecção de erros ou código de correção de erros, que são bits extras adicionados a cada palavra de memória.

Códigos de correção de erros

- Ex.: Dadas duas palavras de código quaisquer, por exemplo:

1000 1001 e 10110001. Utilizando OU EXCLUSIVO verifica-se 3 bits diferentes.

Códigos de correção de erros

Tamanho da palavra	Bits de verificação	Tamanho total	Sobrecarga percentual
8	4	12	50
16	5	21	31
32	6	38	19
64	7	71	11
128	8	136	6
256	9	265	4
512	10	522	2

- Número de bits de verificação para um código que pode corrigir um erro *(m+r bits)*

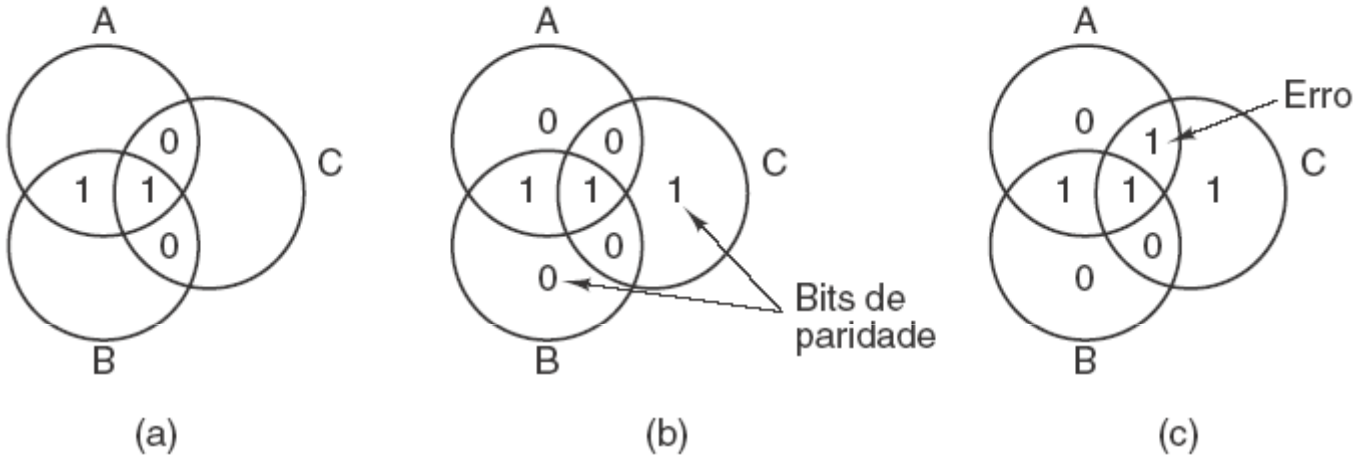
Bit de paridade

- Uma palavra de código de n (=m+r) bits conterá: m bits de dados + r bits de redundância (ou verificação).

- Técnica que segue duas regras simples:
  - **Número de bits "1" é ímpar:** adiciona-se o número "1" ao final;
  - **Número de bits "1" é par:** adiciona-se o número "0" ao final.
- Ex.: a) 1001100 = 10011001

  b) 100100 = 10010000

Códigos de correção de erros

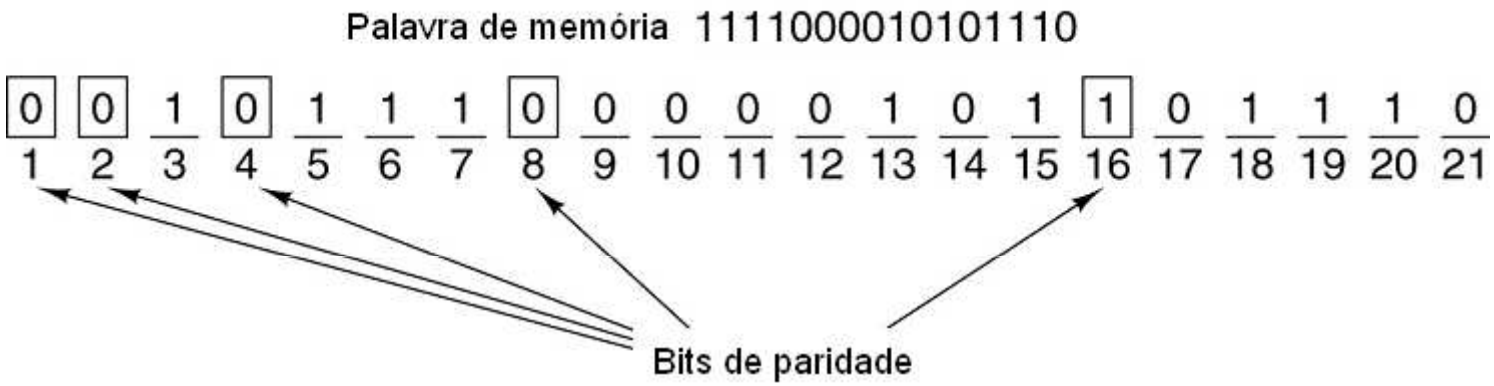


(a) Codificação de 1100

(b) Paridade par adicionada

(c) Erro em *AC*

Códigos de correção de erros



- Construção do código de Hamming para a palavra de memória 1111000000101011110 adicionando 5 bits de verificação aos 16 bits de dados.

Códigos de correção de erros

- Posições a serem verificadas pelos bits de paridade:
  - Bit1 verifica bits: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21.
  - Bit 2 verifica bits: 2, 3, 6, 7, 10, 11, 14, 15, 18, 19.
  - Bit 4 verifica bits: 4, 5, 6, 7, 12, 13, 14, 15, 20, 21.
  - Bit 8 verifica bits: 8, 9, 10, 11, 12, 13, 14, 15.
  - Bit 16 verifica bits: 16, 17, 18, 19, 20, 21.

- Qual seria o resultado da verificação dos bits de paridade, para a Saída de m+r= 00100110000010110111 10? Qual bit estaria incorreto através da verificação?

Códigos de correção de erros

- Bit de paridade 1 incorreto: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 – contém 5 uns
- Bit de paridade 2 correto: 2, 3, 6, 7, 10, 11, 14, 15, 18, 19 – contém 6 uns
- Bit de paridade 4 incorreto: 4, 5, 6, 7, 12, 13, 14, 15, 20, 21 – contém 5 uns
- Bit 8 verifica bits: 8, 9, 10, 11, 12, 13, 14, 15 – contém 2 uns
- Bit 16 verifica bits: 16, 17, 18, 19, 20, 21 – contém 4 uns

- Se a os Bits de paridade 1+4 estão incorretos, logo o resultado de sua soma, 5 é o bit que inverteu o valor...

Tipos de memória

Tipo de memória	Categoria	Apagamento	Mecanismo de escrita	Volatilidade
Memória de acesso aleatório (RAM)	Memória de leitura-escrita	Eletricamente, em nível de byte	Eletricamente	Volátil
Memória somente de leitura (ROM)	Memória somente de leitura	Não é possível	Máscaras	Não volátil
ROM programável (PROM, do inglês *programmable ROM*)			Eletricamente	
PROM apagável (EPROM, do inglês *erasable PROM*)	Memória principalmente de leitura	Luz UV, nível de chip		
PROM eletricamente apagável (EEPROM, do inglês *electrically erasable PROM*)		Eletricamente, nível de byte		
Memória flash		Eletricamente, nível de bloco		

RAM (random access memory)

- Leitura e escrita de forma aleatória;
- Memória volátil;
- Ler e escrever novos dados na memória de modo fácil e rápido;
- Dois tipos: DRAM e SRAM.

DRAM x SRAM

- São memórias voláteis => potência deve ser continuamente fornecida para manter o valor;
- DRAM
  - Célula é mais simples e menor => memória mais densa e barata;
  - Requer suporte de um circuito de recarga (refresh);
  - Adequadas para requisições de grande capacidade => memória principal;

DRAM x SRAM

- SRAM
  - Mais rápidas que as DRAM, mas armazenam menor quantidade de dados;
  - Usadas em memória cache (no chip e fora dele).

ROM (read-only memory)

- Contém um padrão permanente de dados, que não pode ser alterado;
- Memória não-volátil => não requer fonte de energia;
- Aplicações
  - Microprogramação, bibliotecas de funções de uso frequente, programas do sistemas e tabelas de funções;
- Gravação de dados é parte do processo de fabricação da ROM
  - Custo fixo grande, independente da quantidade de memórias fabricadas;
  - Não permite erros => descarte do lote inteiro.

ROM (read-only memory)

- ROM programável (Programmable ROM)
  - Não-volátil;
  - Pode ser escrita uma vez, de forma elétrica, através de equipamentos específicos;
  - Pequeno número de ROM com determinado conteúdo.
- Memória principalmente de leitura
  - Aplicações com muitas operações de leitura e poucas operações de escrita;

Bibliografia:

- Tanembaum, A. S. Organização Estruturada de Computadores. 5 ed – Editora Pearson, 2007.
- Stallings, W. Arquitetura e Organização de Computadores. 8 ed – Editora Pearson, 2009.