

Introdução à Organização de Computadores

Memória Principal

Sistemas da Computação

Prof. Rossano Pablo Pinto, Msc.

rossano at gmail com

2 semestre 2007

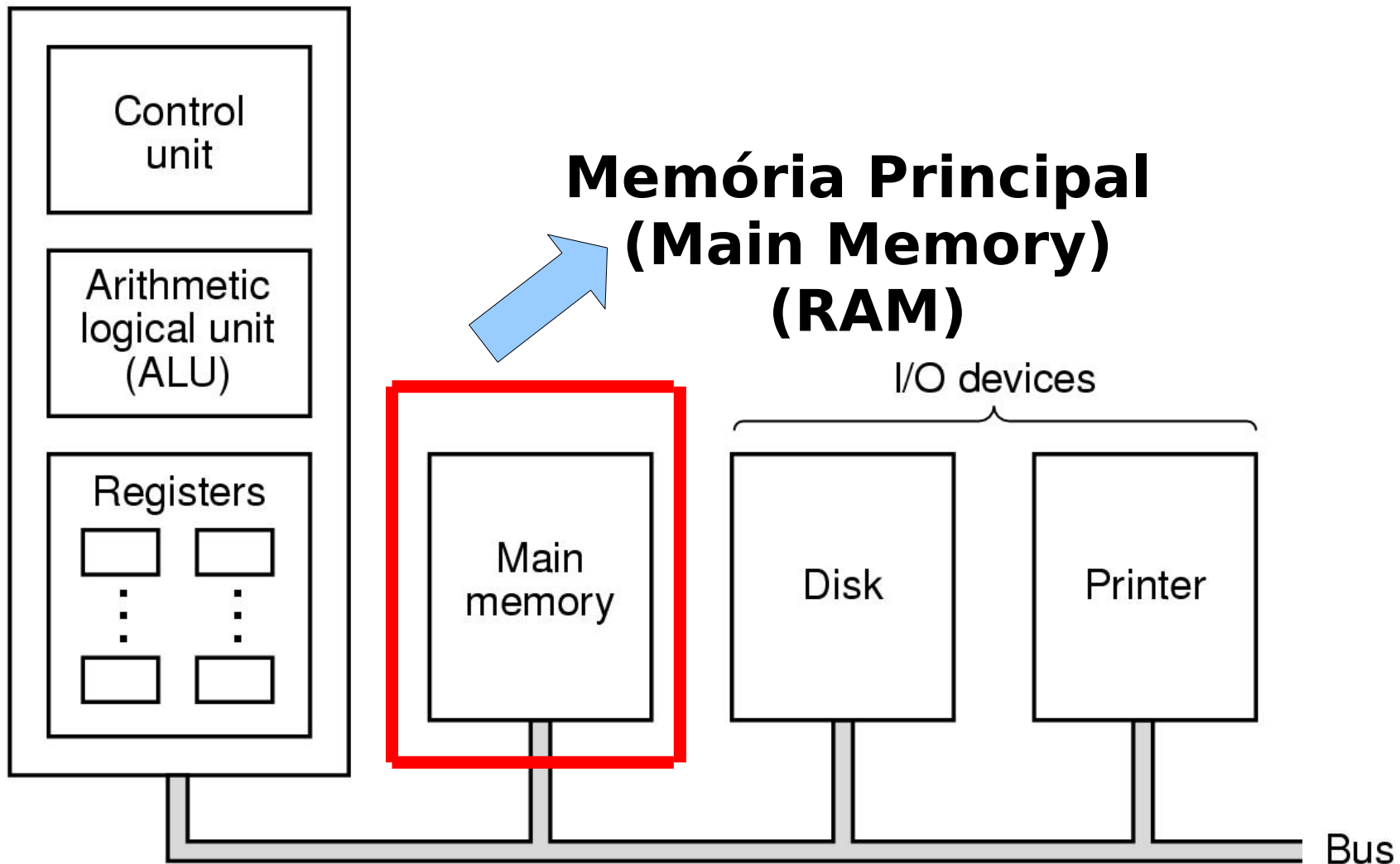
Tópicos

- Processadores
- **Memória Principal (seção 2.2)**
- Memória Secundária
- Entrada e Saída

Memória Principal

Memória Principal

Central processing unit (CPU)



Memória Principal

- Definição: “*Parte do computador onde **programas e dados** são armazenados*”.
- Bit (Binary Digit): Unidade básica de memória. 1 Bit pode armazenar os valores 0 ou 1.
- É a unidade mais simples possível - Um dispositivo que pode armazenar apenas ZERO não seria de muita utilidade - SÃO NECESSÁRIOS PELO MENOS 2 VALORES.

Memória Principal

- É correto dizer que um computador que utiliza **aritmética binária** é mais *eficiente* que um computador que utiliza **aritmética decimal**?
- Eficiente em termos do quê? Velocidade? Capacidade de armazenamento?

Memória Principal

- Informação digital pode ser armazenada por meio da distinção entre valores de algumas quantidades físicas contínuas.
 - Ex.: voltagem ou corrente.
- Quanto **MAIOR O NÚMERO DE ELEMENTOS** para distinguir, **MENOR A SEPARAÇÃO** entre valores adjacentes, e **MENOS CONFIÁVEL** a memória.

Memória Principal

- Sistema de números binários requer apenas a distinção entre dois valores.
- Conseqüentemente, *o sistema de numeração binária é o sistema com maior confiabilidade para armazenar informação digital.*

Memória Principal

- Mainframes IBM possuem aritmética decimal e binária !!!! Como?:
 - Representação decimal em BCD (Binary Coded Decimal)
 - Cada 4 bits armazenam os dígitos de 0 a 9:
 - 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001.
 - 6 de 16 combinações não utilizadas!!!

Memória Principal

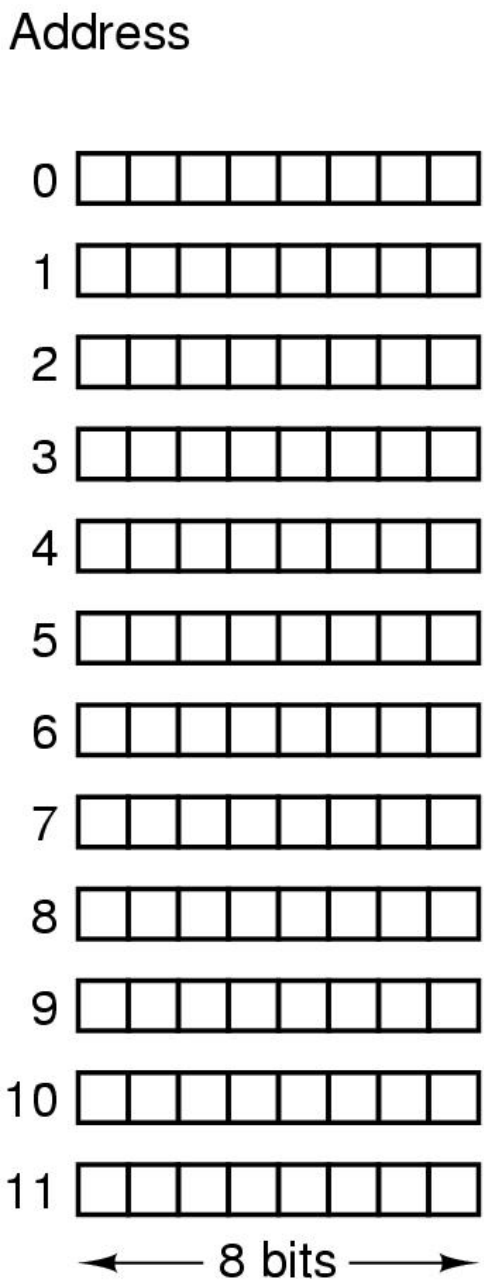
- Suponha que inventem um dispositivo **EXTREMAMENTE CONFIÁVEL** que possa armazenar diretamente os dígitos de 0 à 9 por meio da divisão da região de 0 à 10 volts em 10 intervalos:
 - 4 dispositivos deste tipo poderiam armazenar qualquer número decimal de 0 a 9999 (10.000 combinações)!!!

Memória Principal

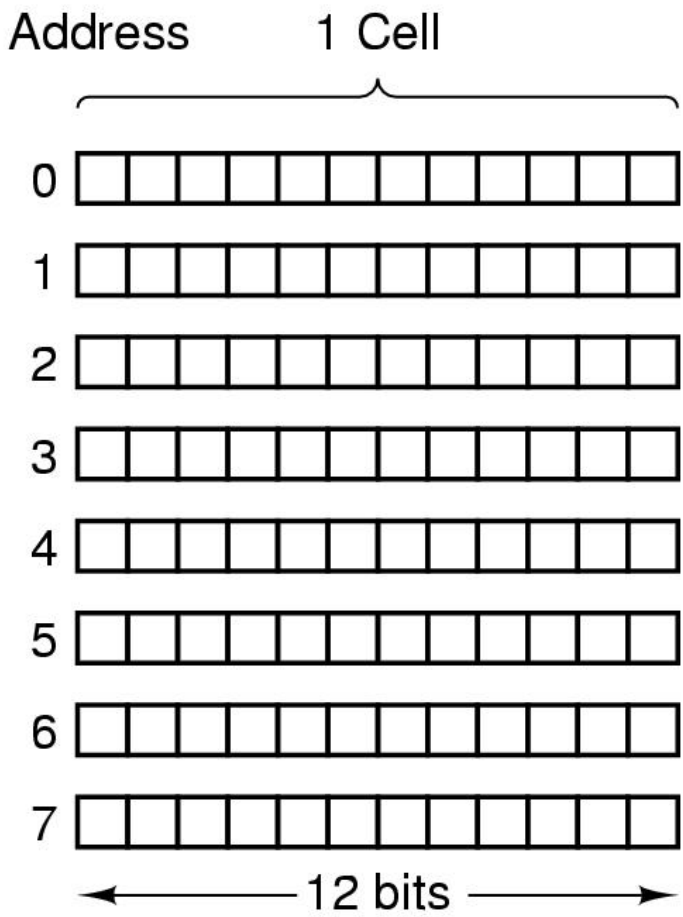
- Endereços de memória
 - Memórias são organizadas em células
 - Cada célula possui um número associado: **endereço**
 - Programas referenciam uma célula a partir deste endereço
 - Se uma memória possui **n** células, tais células possuirão os endereços **0** à **$n - 1$** .

Memória Principal

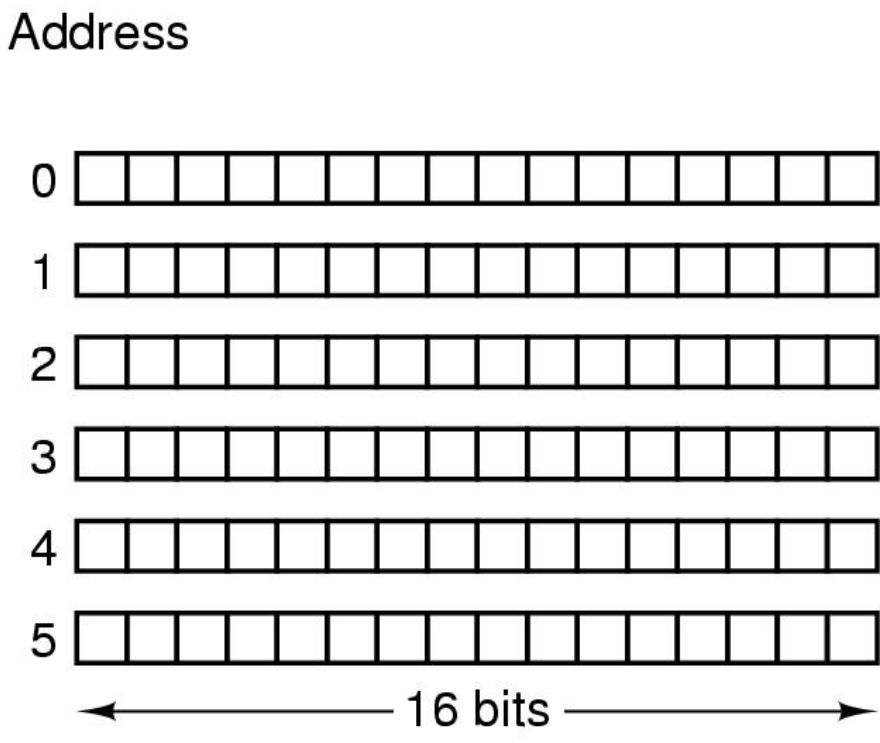
- Endereços de memória
 - Todas as células de uma memória possuem o mesmo número de bits.
 - Se uma célula é constituída de x bits, ela pode armazenar qualquer uma das 2^x diferentes combinações de bits.



(a)



(b)



(c)

Ex.: memória de 96 bits organizada em 3 maneiras distintas.

Memória Principal

- Endereços de memória
 - Células adjacentes tem endereços consecutivos
 - Computadores que utilizam sistemas de numeração binária expressam endereços de memória como número binário
 - Se um endereço possui m bits, o número máximo de células endereçáveis é 2^m .
 - Ex.: Fig. anterior: (a) precisa de pelo menos 4 bits, (b) e (c) apenas 3 bits.

Memória Principal

- Endereços de memória
 - Uma memória com 2^{12} células de 8 bits cada e uma memória de 2^{12} células de 64 bits cada, precisam de endereços de 12 bits.
 - MP com endereços de 0 a $(N - 1)$. X representa número de linhas de endereço:
 - $N = 2^X$
 - $X = \log_2 N$

Memória Principal

- Endereços de memória
 - Tamanho das células utilizadas no IBM-PC são de 8 bits.
 - Bytes são agrupados em palavras (words):
 - computador de 32 bits - 4 bytes/palavra
 - computador de 64 bits - 8 bytes/palavra

Memória Principal

- Endereços de memória: capacidade da memória principal
 - $T = N \times M$
 - $T \rightarrow$ capacidade da memória em bits
 - $N \rightarrow$ número de endereços
 - $M \rightarrow$ número de bits por célula
 - $C = T / 8$
 - $C \rightarrow$ capacidade da memória em bytes

Memória Principal

- Endereços de memória

1) Numa MP com 1kbyte de capacidade, onde cada célula tem 8 bits:

a) quantas células tem a MP? b) quantos bits são necessários para representar um endereço de memória?

2) Um computador endereça 1k células de 16 bits cada uma. Pede-se:

a) sua capacidade de memória; b) o maior endereço que o computador pode endereçar;

3) A memória de um computador tem capacidade de armazenar 216 bits e possui um barramento de dados de 16 bits. Pede-se:

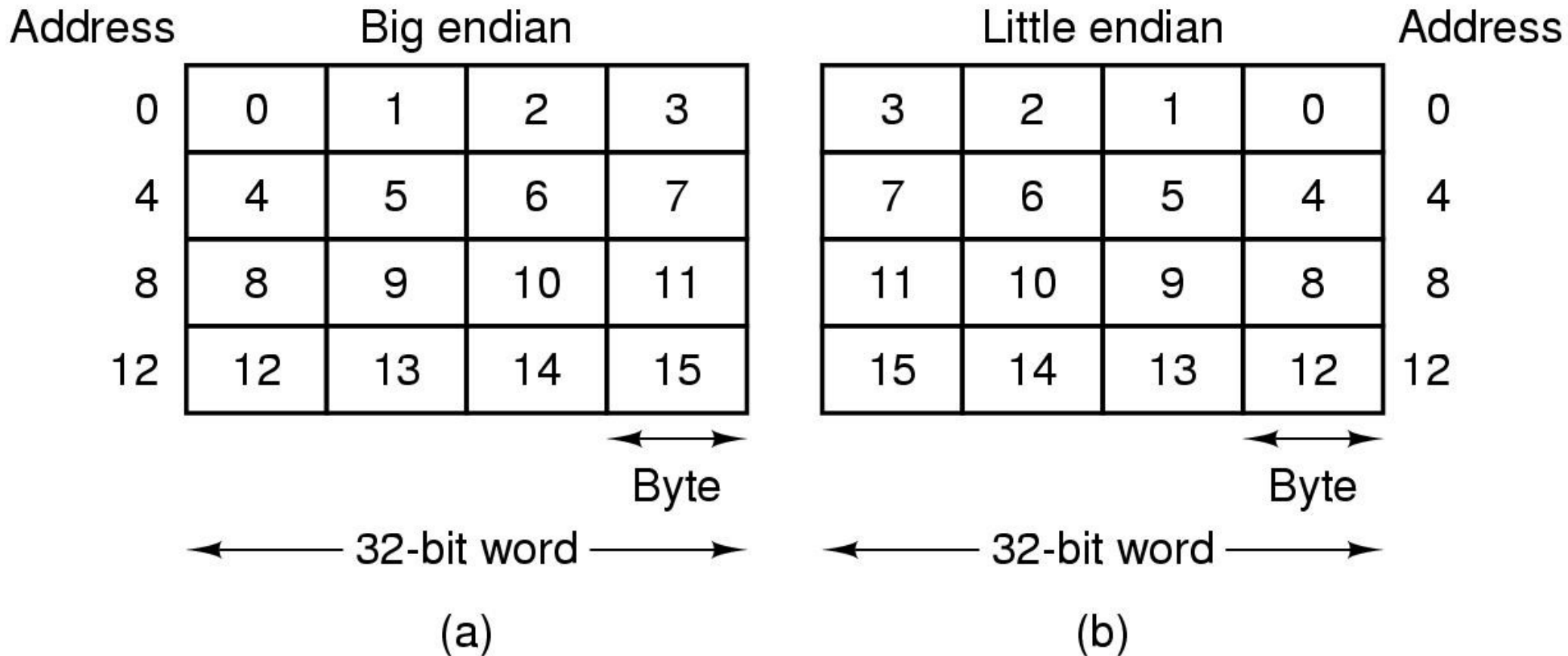
a) o tamanho da célula de memória;

Memória Principal

- Endereços de memória: ordenação de bytes (byte ordering)
 - Os bytes em uma palavra podem ser ordenados **da-esquerda-p-direita** ou **da-direita-p-esquerda**. **IMPORTANTE !!!**
 - SPARC – big-endian (byte 0 da palavra na posição mais significativa)
 - Intel – little-endian (byte 0 da palavra na posição menos significativa)

Memória Principal

- Endereços de memória: ordenação de bytes (byte ordering)



Memória Principal

- Endereços de memória: ordenação de bytes (byte ordering)

Big endian

| | | | | |
|----|---|---|---|----|
| 0 | J | I | M | |
| 4 | S | M | I | T |
| 8 | H | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 21 |
| 16 | 0 | 0 | 1 | 4 |

(a)

Little endian

| | | | |
|---|---|---|----|
| | M | I | J |
| T | I | M | S |
| 0 | 0 | 0 | H |
| 0 | 0 | 0 | 21 |
| 0 | 0 | 1 | 4 |

(b)

Transfer from
big endian to
little endian

| | | | | |
|----|----|---|---|---|
| 0 | | M | I | J |
| 4 | T | I | M | S |
| 8 | 0 | 0 | 0 | H |
| 12 | 21 | 0 | 0 | 0 |
| 16 | 4 | 1 | 0 | 0 |

(c)

Transfer and
swap

| | | | | |
|---|---|---|----|----|
| J | I | M | | 0 |
| S | M | I | T | 4 |
| H | 0 | 0 | 0 | 8 |
| 0 | 0 | 0 | 21 | 12 |
| 0 | 0 | 1 | 4 | 16 |

(d)

Memória Principal

- Memória Cache
 - Problema: CPUs são mais rápidas que memórias do sistema
 - CPUs devem esperar vários ciclos p/ que o dado requerido seja carregado
 - Problema de custo e não de engenharia: é possível construir memórias tão rápidas quanto a CPU (***Precisam ficar dentro do chip da CPU***). **CUSTO MUITO ALTO**

Memória Principal

- Memória Cache
 - Solução: um pouco de memória “rápida” e muita memória “lenta”
 - **combinação** do uso das duas + algumas **técnicas de uso = desempenho da memória mais rápida**
 - Memória cache: memória mais rápida, interna ao chip da CPU

Memória Principal

- Memória Cache

- Idéia básica da memória cache:

- palavras de memória altamente usadas são mantidas em cache

- CPU sempre verifica cache antes de buscar dado na memória principal

- A técnica funciona ? Sim, graças ao **princípio da localidade.**

Memória Principal

- Memória Cache

- Princípio da localidade

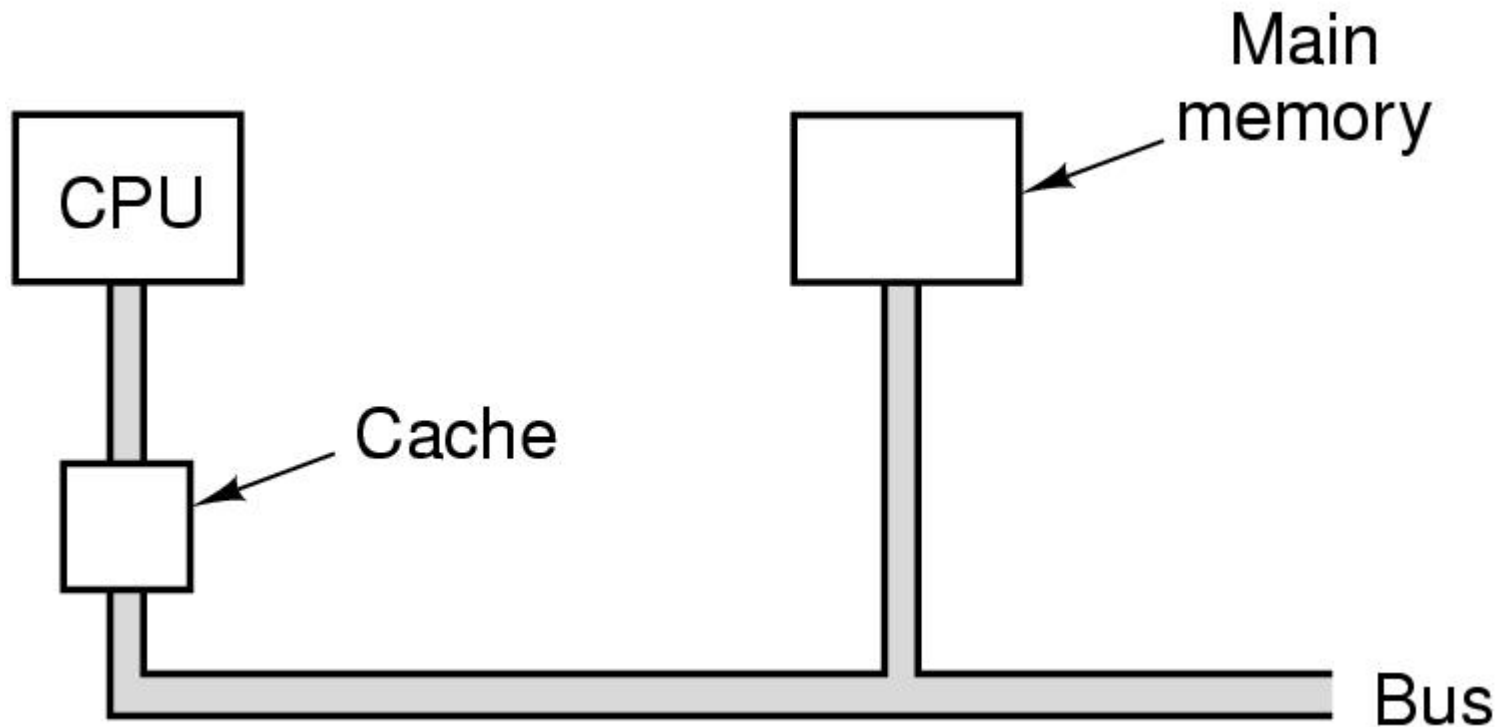
- referências de memória em um período de tempo curto tendem a acessar regiões de memória vizinhas.

- Base para todos os sistemas de cache:

- cada referência à memória principal traz à memória cache os vizinhos anteriores e posteriores ao endereço referenciado

Memória Principal

- Memória Cache



Memória Principal

- Memória Cache
 - Medidas de desempenho
 - **c** = tempo de acesso ao cache
 - **m** = tempo de acesso à memória principal
 - **h** = hit ratio (taxa de acerto) (miss ratio = $1 - h$)
 - fração de todas as referências que puderam ser satisfeitas pelo cache
 - **k** = número de vezes que a palavra é acessada
 - **h = (k - 1)/k**
 - tempo médio de acesso: **c + (1 - h) m**

Memória Principal

- Memória Cache

- Conclusões:

- Se $h = 1$, todas as referências são satisfeitas pelo cache
 - Se $h = 0$, todas as referências são satisfeitas pela memória principal e **tempo de acesso = $c + m$**
 - tempo de acesso ao cache (sem sucesso) + tempo de acesso à memória principal

Memória Principal

- Memória Cache
 - Influência do Princípio da Localidade
 - memória cache e principal são divididas em blocos de tamanho fixo - **linhas de cache** (cache lines)
 - quando ocorre um “*miss*” a *linha de cache* inteira é carregada no cache
 - Ex.: linha de 64-bytes
 - referência ao endereço 260
 - linha de cache: endereços 256 à 319

Memória Principal

- Memória Cache
 - Princípios de Projeto de cache
 - tamanho, tamanho da linha de cache, organização, **unificado x dividido**, número de caches (L1, L2, L3?)
 - Cache unificado x dividido (split cache)
 - unificado: dados + instruções
 - dividido: cache separados para dados e instruções (arquitetura de Harvard - utilizado no computador Mark III de Howard Aiken)