Memória

S. W. Song

MAC 412 - Organização de Computadores

Memória

Veremos apenas dois tópicos:

- Hierarquia de memória
- Código de deteção e código de correção de memória

Hierarquia de Memória

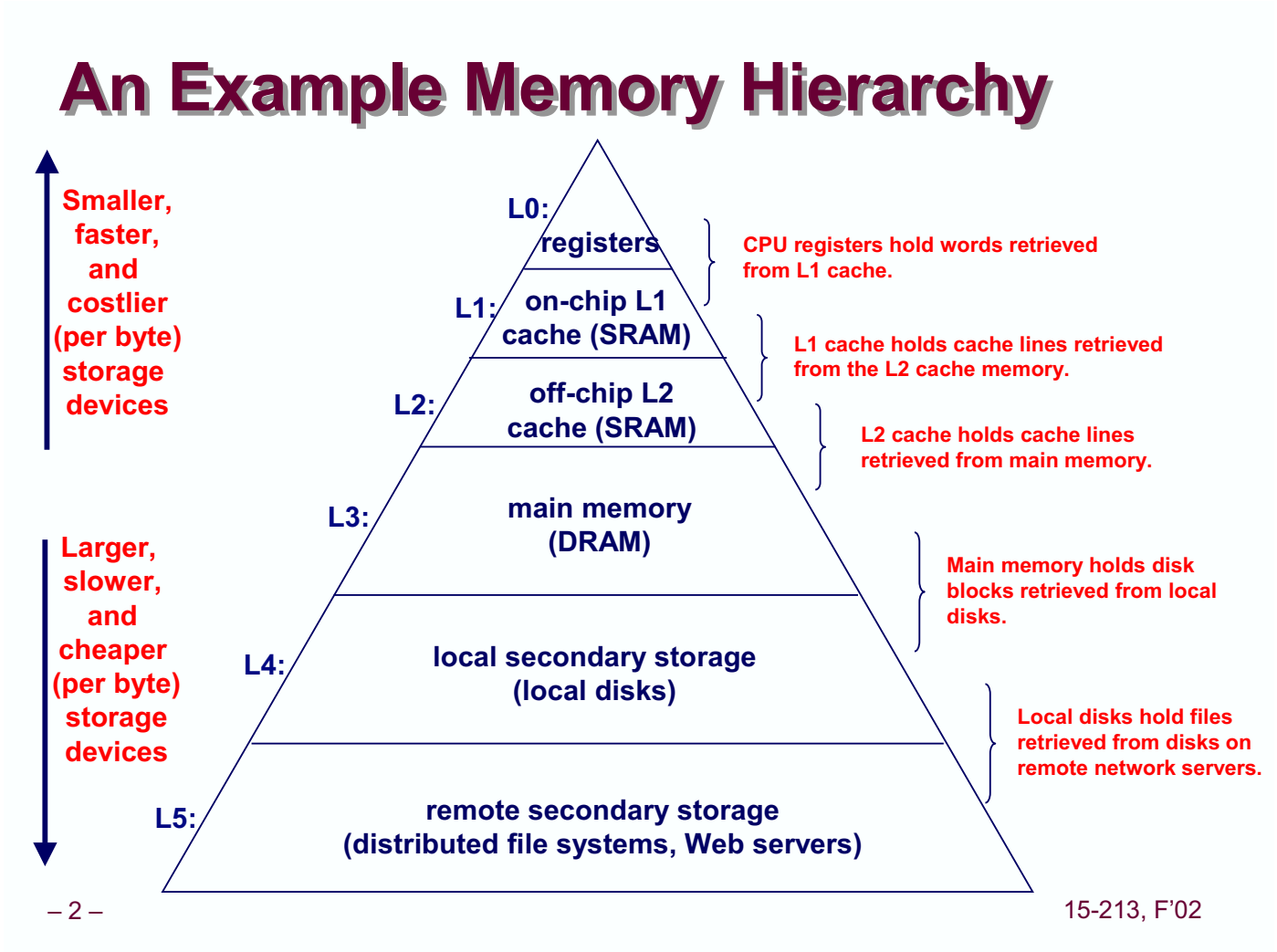
- Registradores
- on-chip cache
- off-chip cache
- memória principal (RAM)
- armazenamento secundário (discos)
- outros armazenamento remotos (arquivos distribuidos, servidores web).

A seguir apresentamos o slide 2 e slide 3 de

`http://web.cecs.pdx.edu/~jrb/cs201/lectures/cache.friendly.code.pdf`

Hierarquia de Memória

slide 2 of http://web.cecs.pdx.edu/~jrb/cs201/lectures/cache.friendly.code.pdf



Hierarquia de Memória

slide 3 of http://www.cs.pdx.edu/~robboy/CLASSES/CS201/slides/cache.friendly.code.pdf

Why Cache-Friendly Code is Important

Cache type	Size of item (bytes)	Latency (cpu cycles)
Registers	4 bytes	0
L1 Cache	32 bytes	1
L2 Cache	32 bytes	10
Main Memory	4-KB pages	100
Disk		millions

On ia32 processor, with few registers, even local variables are likely to spill to memory.
We want them in cache!

Avanços nos vários tipos de memória

Os vários tipos de memória vêm avançando em capacidade e velocidade. Quando foi inventado pela IBM em 1956, o disco RAMAC - Random Access Method of Access and Control (Fonte: *Newsweek*, Aug 14, 2006, p. 8.)

- Pesava uma tonelada
- Era alugado por US$ 250.000,00 por ano
- Tinhas capaciade de 5 Megabytes

Hoje os discos têm capacidade e velocidade bem maiores.

Entretanto, os demais tipos de memória também estão avançando.

Assim, persiste a hierarquia de memória.

Futuro do disco rígido

- Em setembro de 2005, ao lançar a 16 GBytes NAND flash memory, o dono da Samsung prevê o fim do disco rígido.

  http://www.techworld.com/storage/news/index.cfm?NewsID=4387&inkc=0

  Samsung boss predicts death of hard drives.

- Confirmação preliminar pela notícia de 15/03/2007: "Memória flash começa a substituir HDs e promete deixar PC mais rápido."

  http://tecnologia.uol.com.br/especiais/cebit/2007/ultnot/2007/03/15/ult4473u17.jhtm

  SanDisk lança SSD (solid state drive) de 32GB, 100 vezes mais rápido que o HD.

Futuro do disco rígido

Qual a situação mais recente?

- Em 2009, Kingston lançou um flash drive (Kingston DataTraveler 300) de 256GB.
- Em 2013, Kingston anunciou o lançamento de DataTraveler HyperX Predator (USB 3.0) de 1 TB.
- (Em 2015 voce pode comprar esse drive pela Amazon por US$ 772,74 :-)

Dimensão: $2,8 \times 1,1 \times 0,8$ polegadas.

Futuro do disco rígido

Em agosto de 2015, na Flash Memory Summit, Samsung anunciou o SSD (solid state drive) de 16 Tbytes, chamado PM1633a.

Samsung mostrou um servidor com 48 desses drives, totalizando 758 Tbytes.

(Fonte: *Golem.de*)

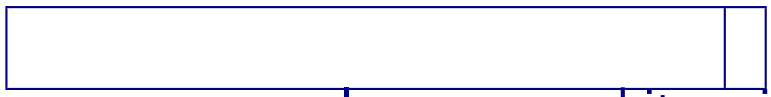
http://www.dpreview.com/articles/5938341907/samsung-introduces-pm1633a-world-first-2-5-16tb-ssd

Códigos de deteção e correção de erros

- Erros de leitura e escrita de memória podem ocorrer, e.g. por problemas de voltagem nas linhas.
- Códigos de deteção e de correção são usados para detectar ou corrigir erros de memória.
- Bits adicionais são acrescentados a cada palavra de memória.
- Algumas dessas técnincas podem também ser usadas para transmissão de dados (e.g. em modens).

Código de deteção

Um bit paridade é acrescentado a cada palavra da memória.



O bit paridade é escolhido de tal modo que o número de 1's do código resultante (palavra+paridade) é par (ou ímpar).

Esse código detecta erros de 1 só bit no código.

Seja o código 00110 (último bit é paridade)

se lido como 01110 (erro de 1 bit: erro detectado)
se lido como 00111 (erro de 1 bit: erro detectado)
se lido como 01111 (erro de 2 bits: nao detectado)

Códigos de deteção e correção de erros

O bit paridade (paridade par) pode ser obtido fazendo o ou-exclusivo dos bits do dado original.

Dado $= x_1x_2x_3x_4$

o bit paridade $x_5 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$

onde $\oplus$ representa a operação *ou exclusivo*.

Código de correção - código de Hamming

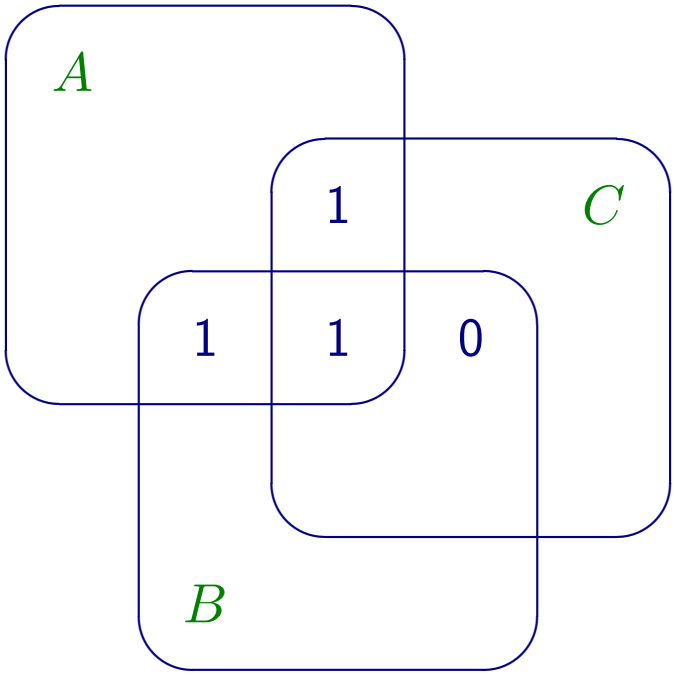
- O código de deteção pode detectar erro, mas não se sabe onde ocorreu o erro, isto é, qual o bit errado. O dado precisa ser lido de novo da memória ou retransmitido no caso de transmissão de dados.

- O código de correção permite saber onde está o erro. Assim é possível corrigi-lo.

Seja um dados de 4 bits. Vamos acrescentar nesse caso mais 3 bits adicionais.

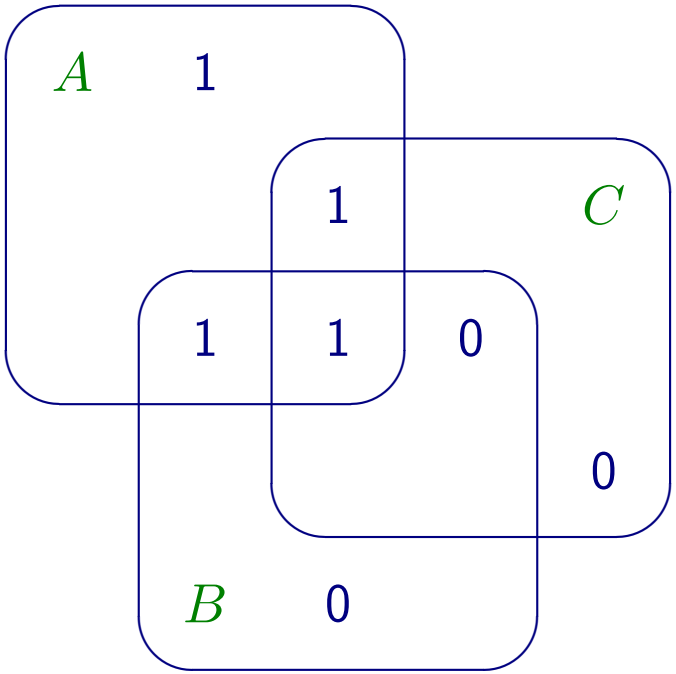
Usamos o diagrama abaixo apenas para fim didático, no caso específco de dado de 4 bits. O método com diagrama não serve para o caso geral em que o dado possui bastante bits. Mostramos como tratar do caso geral mais tarde.

Palavra de 4 bits - 3 bits adicionais



Seja o dado 1101. Para obter os 3 bits extras, vamos inicialmente encarar 1101 como os bits nas regiões de interseção AB, AC, BC, ABC, onde A, B, C são diagramas de Venn.

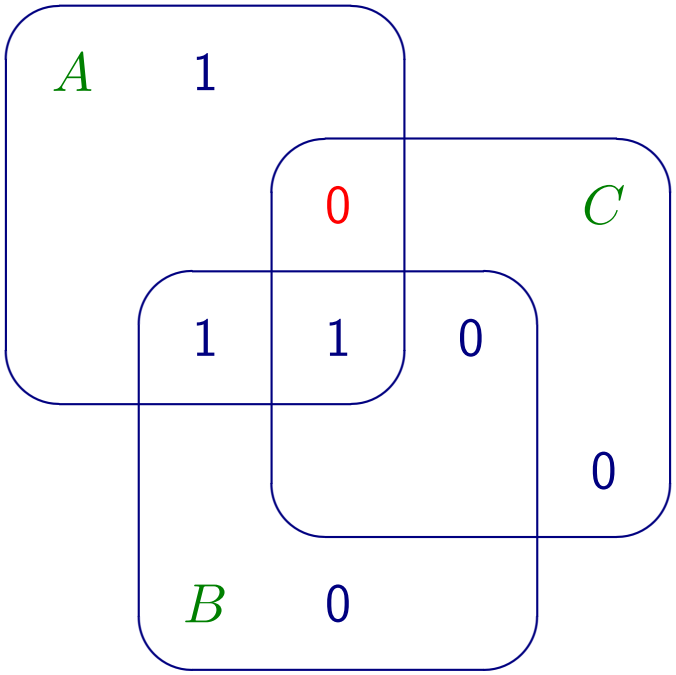
Acrescentando 3 bits adicionais



Agora vamos acrescentar um bit de paridade em cada uma das 3 regiões vazias acima para dar paridade par em A, B, e C.

Os 7 bits (4 do dado original e 3 adicionais) formam o código de Hamming.

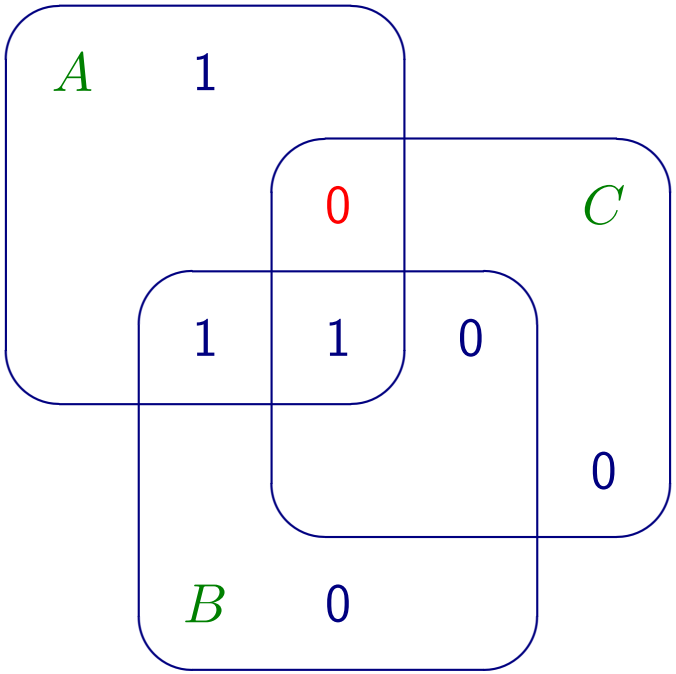
Correção de erro em 1 bit



Erro de 1 bit (qualquer um dos 7 bits) pode ser localizado e corrigido.

Tal erro pode ser detectado de modo simples, como se segue.

Correção de erro em 1 bit



- região A: paridade errada
- região B: paridade OK
- região C: paridade errada
- Conclusão: regiao AC errada. Logo o bit daquela região devia ser 1.

Caso geral: dado de n bits

Seja um dado de n bits. Mostramos um método para obter os k bits adicionais: $k = 1 + \lceil \log n \rceil$.

Palavra de n bits	k bits adicionais	Total bits	overhead %
4	3	7	75
até 8	4	12	50
até 16	5	21	31
até 32	6	38	19
até 64	7	71	11
até 128	8	136	6
até 256	9	265	4

Obs. Para n grande, o *overhead* é menor. Mas lembre-se que o código só funciona para erro de um só bit no código.

Código de Hamming

Vamos mostrar a obtenção do código de Hamming para uma palavra de $n = 8$ bits. Numere os bits de

$$m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8$$

A esse dado de 8 bits vamos acrescentar 4 bits adicionais, formando o código de Hamming de 12 bits.

Numere os bits do código de Hamming como sendo:

$$x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}$$

Inserir os n bits originais no código de Hamming

$$
\begin{aligned}
x_3 &= m_1 \\
x_5 &= m_2 \\
x_6 &= m_3 \\
x_7 &= m_4 \\
x_9 &= m_5 \\
x_{10} &= m_6 \\
x_{11} &= m_7 \\
x_{12} &= m_8
\end{aligned}
$$

Falta obter os 4 bits adicionais x_1, x_2, x_4, x_8.

(Note os seus índices são todos potências de 2.)

Obtenção dos bits adicionais

Os 4 bits adicionais x_1, x_2, x_4 e x_8 são assim calculados, onde $\oplus$ representa a operação *ou exclusivo*:

$$
\begin{aligned}
x_1 &= x_3 \oplus x_5 \oplus x_7 \oplus x_9 \oplus x_{11} \\
x_2 &= x_3 \oplus x_6 \oplus x_7 \oplus x_{10} \oplus x_{11} \\
x_4 &= x_5 \oplus x_6 \oplus x_7 \oplus x_{12} \\
x_8 &= x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12}
\end{aligned}
$$

Observe que a operação ou-exclusivo é equivalente à paridade par.

Regra para chegar às fórmulas

$$
\begin{aligned}
x_1 &= x_3 \oplus x_5 \oplus x_7 \oplus x_9 \oplus x_{11} \\
x_2 &= x_3 \oplus x_6 \oplus x_7 \oplus x_{10} \oplus x_{11} \\
x_4 &= x_5 \oplus x_6 \oplus x_7 \oplus x_{12} \\
x_8 &= x_9 \oplus x_{10} \oplus x_{11} \oplus x_{12}
\end{aligned}
$$

1 a 12 em binário	8	4	2	1
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

Correção de erro

Agora suponha que esses 12 bits são lidos como sendo:

$$y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12}$$

Se não houver erro, então cada y_i é igual seu respectivo x_i.

Se houver erro em um bit apenas, é possível detectar esse erro e corrigi-lo.

Para isso fazemos o seguinte cálculo de 4 bits, denominados k_1, k_2, k_3 e k_4.

Correção de erro

$$
\begin{aligned}
k_1 &= y_1 \oplus y_3 \oplus y_5 \oplus y_7 \oplus y_9 \oplus y_{11} \\
k_2 &= y_2 \oplus y_3 \oplus y_6 \oplus y_7 \oplus y_{10} \oplus y_{11} \\
k_3 &= y_4 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_{12} \\
k_4 &= y_8 \oplus y_9 \oplus y_{10} \oplus y_{11} \oplus y_{12}
\end{aligned}
$$

Se $k_1 = k_2 = k_3 = k_4 = 0$, então não há erro.

Senão o número binário codificado pelos 4 bits $k_4k_3k_2k_1$ determina a posição do bit errado.

Correção de erro

$$
\begin{aligned}
k_1 &= y_1 \oplus y_3 \oplus y_5 \oplus y_7 \oplus y_9 \oplus y_{11} \\
k_2 &= y_2 \oplus y_3 \oplus y_6 \oplus y_7 \oplus y_{10} \oplus y_{11} \\
k_3 &= y_4 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_{12} \\
k_4 &= y_8 \oplus y_9 \oplus y_{10} \oplus y_{11} \oplus y_{12}
\end{aligned}
$$

Exemplo, se $k_4k_3k_2k_1 = 0111$ entao o bit y_7 está errado.

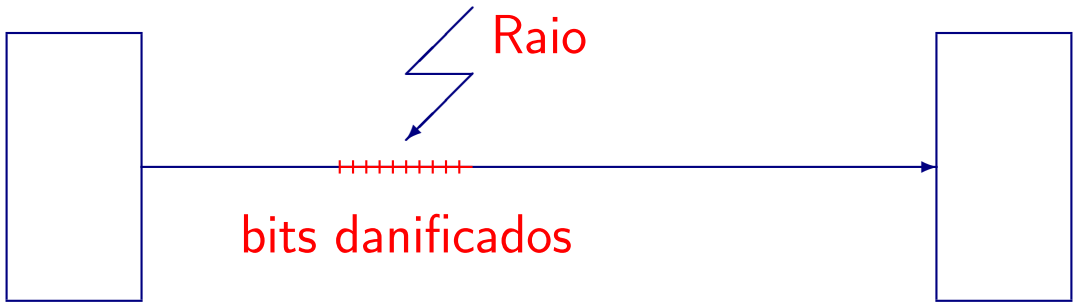
Uma referênciaa:

Vera Pless. *Introduction to the theory of error-correcting codes*. New York : Wiley, 1982, ISBN 0471086843

Erro em mais de 1 bit

- O código de Hamming não funciona para erro em mais de um bit no código.
- Em comunicação da dados, onde uma sequência longa de bits é transmitida de um local a outro, é comum uma série consecutiva de bits ser danificada.
- Veremos um truque que permite detectar e corrigir erros em uma sequência de bits.
- (Uau, que legal!, não posso perder essa dica! :-)

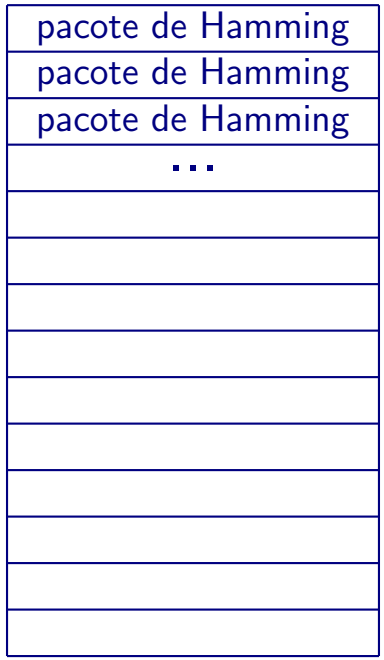
Erro em mais de 1 bit



Vamos ilustrar por um exemplo em comunicação de dados.

- Uma mensagem constituída de um número de pacotes (cada pacote tem n bits) deve ser enviada de um local a outro.
- O meio de transmissão é sujeito a chuvas e trovoadas :-) quando um raio pode danificar uma sequência de bits consecutivos.
- Não queremos apenas detectar erro de transmissão e pedir para retransmitir os pacotes errados. Queremos corrigir os erros.

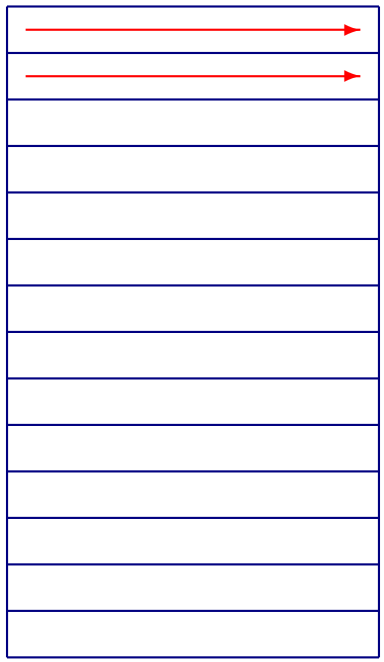
Erro em mais de 1 bit



- Vamos acrescentar a cada pacote de n bits os k bits adicionais conforme estudamos no código de Hamming. Chamamos cada pacote assim incrementado de pacote de Hamming.

- Vamos considerar os pacotes de Hamming em uma matriz onde cada elemento é um paconte de Hamming.

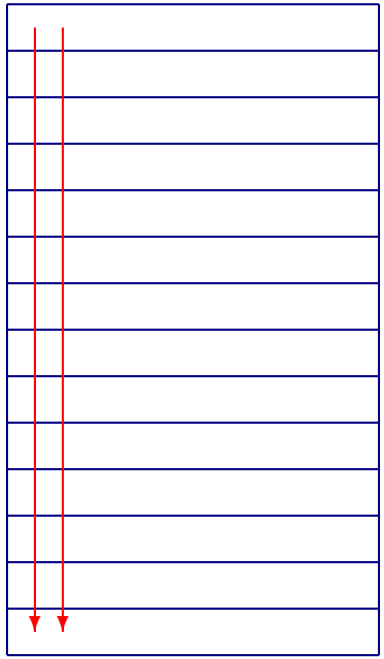
Erro em mais de 1 bit



- Se transimitirmos esses pacotes de Hamming sequencialmente, um a um, então o dano de um raio (que estraga uma série consecutiva de bits) pode ser irrecuperável. Nada adiatou :-(.

Agora vem a idéia brilhante :-).

A idéia brilhante



- Basta transmitirmos a matriz por coluna. No outro lado da recepção coletamos os bits recebidos para reconstruir a matriz.

- Agora aplicamos método de Hamming para cada pacote de Hamming recebido.