Ordem de Armazenamento (Endianess)

Big Endian e Little Endian

Alex Dias Gonsales

Big endian e Little endian

- Imagine a seguinte situação
  - Utiliza-se uma memória para armazenar números binários puros (inteiros positivos) de 8 bits. Essa memória está organizada a bytes, ou seja, cada posição de memória pode armazenar 8 bits. Vamos considerar o exemplo a seguir

Memória organizada a bytes

Endereço	Conteúdo
0	00001100
1	10000001
2	10000110
3	11100010
4	00000000
5	11111111
...	

Memória organizada a bytes

Endereço	Conteúdo
0	00001100
1	10000001
2	10000110
3	11100010
4	00000000
5	11111111
...	

Endereço 0

Conteúdo = 00001100

= 8 + 4 = 12

Portanto

Posição 0 de memória tem o valor 12

`MEM[0] = 12`

Memória organizada a bytes

Endereço	Conteúdo
0	00001100
1	10000001
2	10000110
3	11100010
4	00000000
5	11111111
...	

Endereço 1

Conteúdo = 10000001

= 128 + 1 = 129

Portanto

Posição 1 de memória tem o valor 129

`MEM[1] = 129`

Memória organizada a bytes

Endereço	Conteúdo
0	00001100
1	10000001
2	10000110
3	11100010
4	00000000
5	11111111
...	

Endereço 2
Conteúdo = 10000110
= 128 + 4 + 2 = 134
Portanto
Posição 2 de memória tem o valor 134
`MEM[2] = 134`

Memória organizada a bytes

Endereço	Conteúdo
0	00001100
1	10000001
2	10000110
3	11100010
4	00000000
5	11111111
...	

Endereço 3

Conteúdo = 11100010

= 128 + 64 + 32 + 2 = 226

Portanto

Posição 3 de memória tem o valor 226

`MEM[3] = 226`

Memória organizada a bytes

Endereço	Conteúdo
0	00001100
1	10000001
2	10000110
3	11100010
4	00000000
5	11111111
...	

Endereço 4
Conteúdo = 00000000
= 0
Portanto
Posição 4 de memória tem o valor 0
`MEM[4] = 0`

Memória organizada a bytes

Endereço	Conteúdo
0	00001100
1	10000001
2	10000110
3	11100010
4	00000000
5	11111111
...	

Endereço 5

Conteúdo = 11111111

= 128+64+32+16+8+4+2+1 = 255

Portanto

Posição 5 de memória tem o valor 255

`MEM[5] = 255`

Como armazenar números maiores que 8 bits?

Como armazenar números maiores que 8 bits?

- SIMPLES: utilizando mais de uma posição de memória para armazenar os números
  - Exemplo: utilizar duas posições para armazenar cada número, ou seja, iremos considerar que cada número armazenado possui 16 bits

Como armazenar números maiores que 8 bits?

- Exemplo:
- Utilizamos as posições 0 e 1 para armazenar um número X
- Utilizamos as posições 2 e 3 para armazenar um número Y
- Utilizamos as posições 4 e 5 para armazenar um número Z
- E assim sucessivamente...

Endereço	Conteúdo
0	xxxxxxxx
1	xxxxxxxx
2	yyyyyyyy
3	yyyyyyyy
4	zzzzzzzz
5	zzzzzzzz
...	

Exemplo

- Como armazenar o número 773 a partir da posição 2 da memória? O número 773 = 1100000101_2
- Ou seja, o número 773 necessita 10 bits para ser representado
- Então vamos utilizar 16 bits para representar o número
- Preenchemos com zeros à esquerda
- 773 = 0000001100000101_2
- Separamos o número em duas unidades de 8 bits, portanto
- 00000011 e 00000101
- Armazenar cada unidade em uma posição de memória, mas levar em conta a convenção (ordem de armazenamento) que estiver sendo utilizada (big endian ou little endian)

Exemplo

$773 = 1100000101_2$

O número 773 necessita 10 bits, logo, será utilizado 16 bits para armazená-lo.

Exemplo

$773 = 1100000101_2$

$773 = 0000001100000101_2$

Expandir o número para 16 bits.
Preencher com zeros à esquerda

Exemplo

$773 = 1100000101_2$

$773 = 0000001100000101_2$

Separar o número em duas unidades de 8 bits.

Exemplo

$773 = 1100000101_2$

$773 = 0000001100000101_2$

Separar o número em duas unidades de 8 bits.

Exemplo

$773 = 1100000101_2$

$773 = 0000001100000101_2$

00000101

LSB (menos significativa)

A parte da direita é chamada
LSB (Least Significant Byte)
Byte menos significativo

Exemplo

$773 = 1100000101_2$

$773 = 0000001100000101_2$

00000011

MSB (mais significativa)

A parte da esquerda é chamada
MSB (Most Significant Byte)
Byte mais significativo

Exemplo

$773 = 1100000101_2$

$773 = 0000001100000101_2$

00000011 **00000101**

MSB (mais significativa) LSB (menos significativa)

Armazenar cada parte em uma posição de memória
Mas quem vai primeiro?
Depende da convenção utilizada.
Big Endian ou Little Endian

Exemplo

773 = 00000011 00000101

MSB (mais significativa) LSB (menos significativa)

Em qual ordem armazenar?

Endereço	Conteúdo
...	. . .
2	????????
3	????????
...	...

Exemplo

773 = **00000011** **00000101**

MSB (mais significativa) LSB (menos significativa)

Convenção Big Endian:

- Armazenar **primeiro** (no endereço menor) a parte **mais** significativa (Big=grande).

Endereço	Conteúdo
...	. . .
2	00000011
3	00000101
...	...

Exemplo

773 = **00000011** **00000101**

MSB (mais significativa) LSB (menos significativa)

Convenção Little Endian:

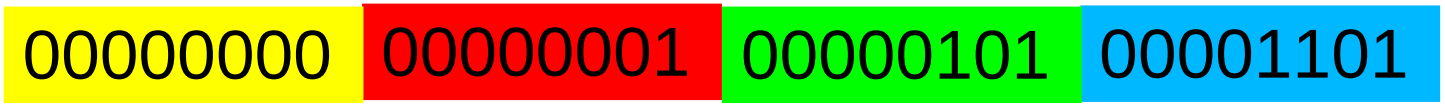
- Armazenar **primeiro** (no endereço menor) a parte **menos** significativa (Little=pequeno).

Endereço	Conteúdo
...	. . .
2	00000101
3	00000011
...	...

Exemplo: 66829

- Como armazenar o número 66829 a partir da posição 2 da memória? O número 66829 = 10000010100001101_2
- Ou seja, o número 66829 necessita 17 bits para ser representado
- Então vamos utilizar 32 bits para representar o número
- Preenchemos com zeros à esquerda
- 66829 = $00000000000000010000010100001101_2$
- Separamos o número em quatro unidades de 8 bits, portanto
- 00000000 00000001 00000101 00001101
- Armazenar cada unidade em uma posição de memória, mas levar em conta a convenção (ordem de armazenamento) que estiver sendo utilizada (big endian ou little endian)

Exemplo: 66829

66829 = 

MSB
(mais significativa)

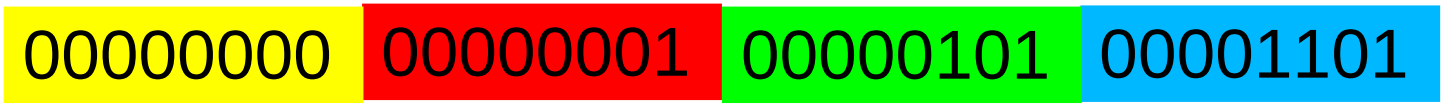
LSB
(menos significativa)

Convenção Big Endian:

- Armazenar **primeiro** (no endereço menor) a parte **mais** significativa (Big=grande).

Endereço	Conteúdo
2	00000000
3	00000001
4	00000101
5	00001101

Exemplo: 66829

66829 = 

MSB
(mais significativa)

LSB
(menos significativa)

Convenção Little Endian:

- Armazenar **primeiro** (no endereço menor) a parte **menos** significativa (Little=pequeno).

Endereço	Conteúdo
2	00001101
3	00000101
4	00000001
5	00000000