Conversão de Little-endian para Big-endian

João P. da Silva Dantas, Jorge L. F. Costa, Marcelo S. Antunes

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul (UFMS) – Campo Grande, MS – Brasil

joao.dantas@ufms.br, myrdiaclonix@proton.me, marcelo.antunes@ufms.br

Abstract. *This report describes a program in C++ format that performs the conversion of an unsigned integer 32-bit binary number with little-endian endianness to its representative with big-endian endianness, presenting a matter about computer architecture in memory processing.*

Resumo. *Este relatório descreve um programa em formato C++ que realiza a conversão de um número inteiro sem sinal binário de 32 bits com endianness do tipo little-endian para o seu representante com endianness big-endian, apresentando uma questão sobre a arquitetura dos computadores sobre o tratamento da memória.*

1. Introdução

Os dados armazenados na memória de computadores obedecem uma certa formatação para facilitar a troca dessas informações. A menor unidade do dado é o bit, podendo ter valor 0 ou 1. Um conjunto de 8 bits forma um byte. No cenário de representação de números, o bit mais à esquerda de um byte é tido como o mais significativo, pois representa o maior valor. Assim, o bit mais à direita tem o menor valor e é o menos significativo.

Especificamente, little-endian é quando os bytes menos significativos são armazenados antes dos bytes mais significativos, e big-endian quando os bytes mais significativos são armazenados antes dos bytes menos significativos. Quando escrevemos um número (em hexadecimal), ou seja 0x12345678, escrevemos primeiro com o byte mais significativo (a parte 12). De certa forma, big-endian é a maneira "normal" de escrever as coisas.

Exemplo número 12345678:

Tabela, Big-Endian

Endereço	Valor
1000	12
1001	34
1002	56
1003	78

Tabela, Little-endian

Endereço	Valor
1003	78
1002	56
1001	34
0000	12

Porque a conversão é importante? Supõe que esteja armazenando valores de inteiros numa máquina e queira passar as informações para outra que usa extremidade oposta a medida que lê o valor, isso causará problemas por causa da endianness, os valores são totalmente invertidos.

Para fazer a conversão do little-endian para big-endian utilizamos a linguagem em C++, assim retornando o arranjo de memória em nosso computador, assim, facilitando a leitura.

2. Metodologia de Desenvolvimento

Para realizar a conversão do endianness de um número binário, fizemos um programa em C++. A entrada consiste em um vetor de valores a serem convertidos. O primeiro elemento desse vetor deve ser seu tamanho. A partir do segundo elemento, se houver, os elementos deverão ser números binários sem sinal de 32 bits. A metodologia utilizada foi ler cada entrada a ser convertida como um dado do tipo string e depois inverter os bytes ordenadamente. A função utilizada foi a que segue.

```
string convert(string s) {
    string r = "";
    for (int i = 3; i >= 0; i--) {
        r += s.substr(8*i, 8);
    }
    return r;
}
```

Assim, cada número de 32 bits será dividido em 4 partes, onde cada parte será a representação de um byte. O retorno da função é uma string criada por meio da interação inversa de cada uma dessas partes, alterando o endianness do número.

3. Resultados

Um exemplo de caso de teste é o que segue:

```
4
10010010100101010011000001110111
00000010101101100000101110101010
11100010010000000000000000000000
```

Cada linha representa um elemento do vetor de entrada. Como o primeiro elemento é 4, temos um restante de 3 números em little-endian para ser convertidos. Vamos considerar o primeiro.

```
10010010100101010011000001110111
```

Separando-o em grupos de 8 bits, temos a divisão:

```
10010010 – 10010101 – 00110000 – 01110111
```

Invertendo a ordenação, conseguimos:

```
01110111 – 00110000 – 10010101 – 10010010
```

E por fim, agrupando os bytes obtemos o valor numérico representado em big-endian.

```
01110111001100001001010110010010
```

Isso acontece com cada valor de entrada. Em versões posteriores, o código poderá armazenar as strings de saída em um vetor e transformá-las em números, devendo haver um decodificador para considerar o endianness do próprio.

Existem certas limitações no programa proposto. O não cumprimento da escrita de 32 bits para cada caso acarreta em uma falha na divisão dos bytes.

4. Conclusão

Ademais, através da implementação de algumas funções e lógica de programação foi possível concluir o objetivo do desenvolvimento para realizar a conversão do little-endian para o big-endian. Portanto, concluímos que é de suma importância realizar essa conversão para analisar e definir os impactos dos dados de um sistema para o outro, principalmente pela diferença de sistemas, que trabalham entre little-endian e big-endian.

Referências

Kalid, (2006) “Understanding Big and Little Endian Byte Order”, https://betterexplained.com/articles/understanding-big-and-little-endian-byte-order/, September.

Harsha Adiga, (2007) “How to write endian-independent code in C”, https://developer.ibm.com/articles/au-endianc/, Abril.

Zack Jorquera, (2020) “What Is Little-Endian And Big-Endian Byte Ordering?”, https://www.section.io/engineering-education/what-is-little-endian-and-big-endian/, March.

James L. Frankel, Harvard University, (2016) “Endianness”, https://cscie93.dce.harvard.edu/spring2020/slides/Endianness.pdf, March.