

Operações Lógicas Bit a Bit

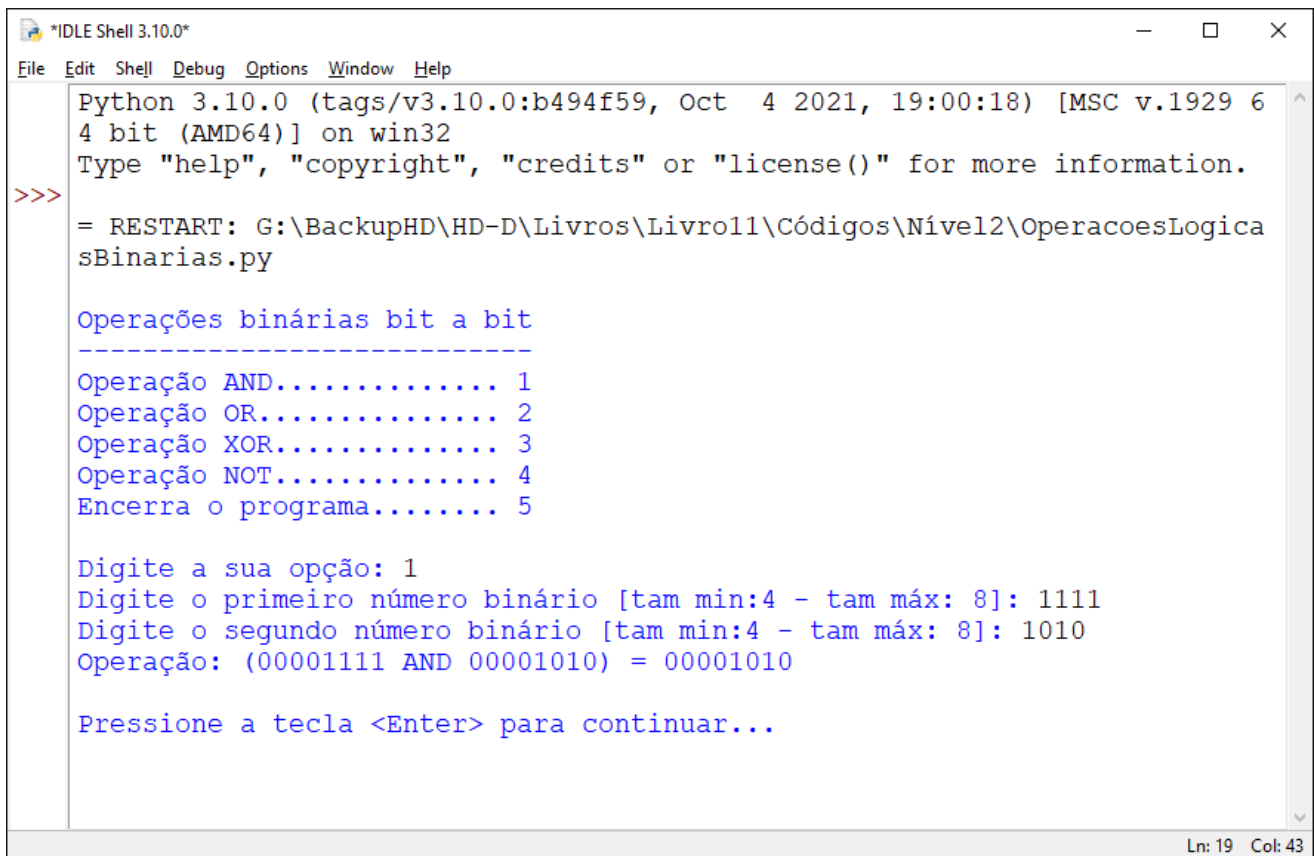
Mário Leite

...

As quatro operações aritméticas: Adição, Subtração, Multiplicação e Divisão são as primeiras operações básicas da Matemática que todos aprendem a fazer depois de conhecer os dígitos numéricos da Base Decimal. Por outro lado, existem outros tipos de operações *bit a bit* que são as mais importantes do ponto de vista computacional, simplesmente pelo fato de serem as que são, efetivamente, executadas pelo computador. Na verdade, o computador não “conhece” nenhuma das operações aritméticas do ponto de vista “humano”; ele simplesmente faz deslocamentos de *bits* (**0** e **1**) para compor o resultado de uma operação matemática OU DE QUALQUER OUTRO TIPO DE OPERAÇÃO; não só operações matemáticas.

Deste modo, são as operações lógicas que definem o trabalho de um sistema computacional. Então, como é bem dito: “o computador só entende ZERO/UM”, que traduzindo para a lógica são **NÃO** e **SIM**, respectivamente. Por exemplo **SIM E SIM = SIM**, **SIM E NÃO = NÃO**; **SIM OU NÃO = SIM**, e assim por diante, seguindo a famosa “Tabela Lógica” onde **E** e **OU** são operadores lógicos conhecidos como conjunção (**AND**) e disjunção (**OR**), respectivamente: **1 AND 1 = 1**, **1 AND 0 = 0**; **1 OR 0 = 1**, etc.

Para explicar isto o programa “OperacoesLogicasBinarias”, codificado em Python e em C, mostra como fazer as quatro operações lógicas básicas *bit a bit*, lendo dois números binários (máximo oito bits) com os operadores: **AND**, **OR**, **XOR** e **NOT**.



```
*IDLE Shell 3.10.0*
File Edit Shell Debug Options Window Help
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 6
4 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:\BackupHD\HD-D\Livros\Livro11\Códigos\Nível2\OperacoesLogica
sBinarias.py

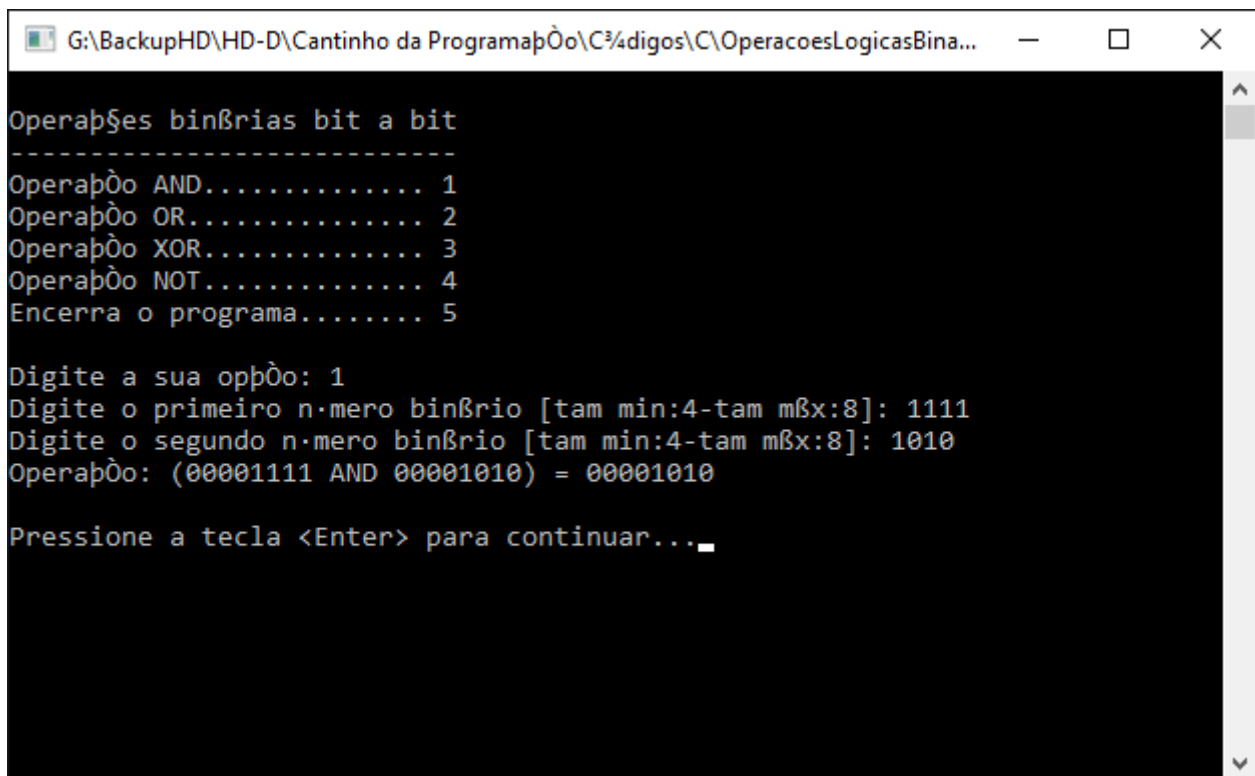
Operações binárias bit a bit
-----
Operação AND..... 1
Operação OR..... 2
Operação XOR..... 3
Operação NOT..... 4
Encerra o programa..... 5

Digite a sua opção: 1
Digite o primeiro número binário [tam min:4 - tam máx: 8]: 1111
Digite o segundo número binário [tam min:4 - tam máx: 8]: 1010
Operação: (00001111 AND 00001010) = 00001010

Pressione a tecla <Enter> para continuar...

Ln: 19 Col: 43
```

Figura 1 - Saída do programa em Python



```
G:\BackupHD\HD-D\Cantinho da Programação\C³4digos\C\OperacoesLogicasBina...  —  □  X

Operações binárias bit a bit
-----
Operação AND..... 1
Operação OR..... 2
Operação XOR..... 3
Operação NOT..... 4
Encerra o programa..... 5

Digite a sua opção: 1
Digite o primeiro n.º binário [tam min:4-tam mx:8]: 1111
Digite o segundo n.º binário [tam min:4-tam mx:8]: 1010
Operação: (00001111 AND 00001010) = 00001010

Pressione a tecla <Enter> para continuar..._
```

Figura 2 – Saída do programa em C

```
'''
```

OperacoesLogicasBinarias.py

Lê dois números binários (máximo oito bits) e faz as operações lógicas básicas: AND, OR, XOR e NOT, bit a bit.

Data: 24/09/2023

Autor: Mário Leite

```
'''
```

```
import time
```

```
TamBin = 8 #define o padrão da palavra para 1 byte (8 bits)
```

```
def FazerOperacaoAND(bin1, bin2):  
    retStr = ""  
    for j in range(TamBin):  
        bit1 = int(bin1[j])  
        bit2 = int(bin2[j])  
        resultado_bit = bit1 & bit2  
        retStr += str(resultado_bit)  
    return retStr
```

```
#-----  
def FazerOperacaoOR(bin1, bin2):  
    retStr = ""  
    for j in range(TamBin):  
        bit1 = int(bin1[j])  
        bit2 = int(bin2[j])  
        resultado_bit = bit1 | bit2  
        retStr += str(resultado_bit)  
    return retStr
```

```
#-----  
def FazerOperacaoXOR(bin1, bin2):  
    retStr = ""  
    for j in range(TamBin):  
        bit1 = int(bin1[j])  
        bit2 = int(bin2[j])  
        resultado_bit = bit1 ^ bit2  
        retStr += str(resultado_bit)  
    return retStr
```

```
#-----  
def FazerOperacaoNOT(binario):  
    binario = binario.zfill(TamBin) # Preenche com zeros à esquerda até ter tamanho 8  
    retStr = ""  
    for j in range(TamBin):  
        bit = int(binario[j])  
        resultado_bit = 1 - bit  
        retStr += str(resultado_bit)  
    return retStr
```

```
#-----  
def ValidarBinario(bin_str):  
    if len(bin_str) < 4 or len(bin_str) > TamBin:  
        return False  
    for bit in bin_str:  
        if bit != "0" and bit != "1":  
            return False  
    return True
```

```

#####
#Proghrama principal
while(True):
    print("\nOperações binárias bit a bit")
    print("-----")
    print("Operação AND..... 1")
    print("Operação OR..... 2")
    print("Operação XOR..... 3")
    print("Operação NOT..... 4")
    print("Encerra o programa..... 5")
    print()
    op = abs(int(input("Digite a sua opção: ")))
    if((op<1) or (op>=5)):
        print(); print() #ATENÇÃO: mesmo sendo válido isto deve ser evitado
        print("Encerrando...")
        time.sleep(3) #aguarda três segundos
        break

    if(op != 4):
        while(True):
            bin1=input(f"Digite o primeiro número binário [tam min:4-tam máx:{TamBin}]: ")
            if(ValidarBinario(bin1)):
                break

        while(True):
            bin2=input(f"Digite o segundo número binário [tam min:4-tam máx:{TamBin}]: ")
            if(ValidarBinario(bin2)):
                break

        difTam1 = TamBin - len(bin1)
        difTam2 = TamBin - len(bin2)

        if difTam1 < TamBin:
            bin1 = "0" * difTam1 + bin1

        if difTam2 < TamBin:
            bin2 = "0" * difTam2 + bin2

        if(op == 1):
            resultado = FazerOperacaoAND(bin1, bin2)
            print(f"Operação: ({bin1} AND {bin2}) = {resultado}")
        elif(op == 2):
            resultado = FazerOperacaoOR(bin1, bin2)
            print(f"Operação: ({bin1} OR {bin2}) = {resultado}")
        elif(op == 3):
            resultado = FazerOperacaoXOR(bin1, bin2)
            print(f"Operação: ({bin1} XOR {bin2}) = {resultado}")
        else:
            while(True):
                binario = input(f"Digite o número binário [tam min 4-tam máx {TamBin}]: ")
                if 4 <= len(binario) <= TamBin AND ValidarBinario(binario):
                    break

            resultado = FazerOperacaoNOT(binario)
            if resultado == "Tamanho inválido":
                print("Tamanho do número binário inválido.")
            else:
                print(f"Operação: NOT ({binario}) = {resultado}")

    input("\nPressione a tecla <Enter> para continuar...")
#Fim do programa "OperacoesLogicasBinarias" -----

```

```

/*
OperacoesLogicasBinarias.C
-----
Lê dois números binários (máximo oito bits) e faz as operações: gicas básicas:
AND, OR, XOR e NOT, bit a bit.
-----
Data: 24/09/2023
Autor: Mário Leite
-----
*/
#include <stdio.h>
#include <string.h>
#define TamBin 8
void FazerOperacaoAND(char *bin1, char *bin2, char *result) {
    for (int j = 0; j < TamBin; j++) {
        int bit1 = bin1[j] - '0';
        int bit2 = bin2[j] - '0';
        int resultado_bit = bit1 & bit2;
        result[j] = resultado_bit + '0';
    }
    result[TamBin] = '\0';
}

void FazerOperacaoOR(char *bin1, char *bin2, char *result) {
    for (int j = 0; j < TamBin; j++) {
        int bit1 = bin1[j] - '0';
        int bit2 = bin2[j] - '0';
        int resultado_bit = bit1 | bit2;
        result[j] = resultado_bit + '0';
    }
    result[TamBin] = '\0';
}

void FazerOperacaoXOR(char *bin1, char *bin2, char *result) {
    for (int j = 0; j < TamBin; j++) {
        int bit1 = bin1[j] - '0';
        int bit2 = bin2[j] - '0';
        int resultado_bit = bit1 ^ bit2;
        result[j] = resultado_bit + '0';
    }
    result[TamBin] = '\0';
}

void FazerOperacaoNOT(char *binario, char *result) {
    for (int j = 0; j < TamBin; j++) {
        int bit = binario[j] - '0';
        int resultado_bit = 1 - bit;
        result[j] = resultado_bit + '0';
    }
    result[TamBin] = '\0';
}

int ValidarBinario(char *bin_str) {
    if (strlen(bin_str) < 4 || strlen(bin_str) > TamBin) {
        return 0;
    }
    for (int i = 0; i < strlen(bin_str); i++) {
        if (bin_str[i] != '0' && bin_str[i] != '1') {
            return 0;
        }
    }
    return 1;
}

```

```
//=====
//Programa principal
int main() {
    char resultado[TamBin + 1];
    int op;

    while (1) {
        printf("\nOperações binárias bit a bit\n");
        printf("-----\n");
        printf("Operação AND..... 1\n");
        printf("Operação OR..... 2\n");
        printf("Operação XOR..... 3\n");
        printf("Operação NOT..... 4\n");
        printf("Encerra o programa..... 5\n\n");

        printf("Digite a sua opção: ");
        scanf("%d", &op);
        getchar(); //limpa nova linha

        if (op < 1 || op >= 5) {
            printf("\nEncerrando...\n");
            return 0;
        }

        if (op != 4) {
            char bin1[TamBin + 1], bin2[TamBin + 1];
            printf("Digite o primeiro número binário [tam min:4-tam máx:%d]: ", TamBin);
            fgets(bin1, sizeof(bin1), stdin);
            bin1[strcspn(bin1, "\n")] = '\0';

            while (!ValidarBinario(bin1)) {
                printf("Número binário inválido. Digite novamente: ");
                fgets(bin1, sizeof(bin1), stdin);
                bin1[strcspn(bin1, "\n")] = '\0';
            }

            printf("Digite o segundo número binário [tam min:4-tam máx:%d]: ", TamBin);
            fgets(bin2, sizeof(bin2), stdin);
            bin2[strcspn(bin2, "\n")] = '\0';

            while (!ValidarBinario(bin2)) {
                printf("Número binário inválido. Digite novamente: ");
                fgets(bin2, sizeof(bin2), stdin);
                bin2[strcspn(bin2, "\n")] = '\0';
            }

            int difTam1 = TamBin - strlen(bin1);
            int difTam2 = TamBin - strlen(bin2);

            if (difTam1 < TamBin) {
                for (int i = 0; i < difTam1; i++) {
                    memmove(bin1 + 1, bin1, strlen(bin1) + 1);
                    bin1[0] = '0';
                }
            }

            if (difTam2 < TamBin) {
                for (int i = 0; i < difTam2; i++) {
                    memmove(bin2 + 1, bin2, strlen(bin2) + 1);
                    bin2[0] = '0';
                }
            }
        }
    }
}
```

```

    if (op == 1) {
        FazerOperacaoAND(bin1, bin2, resultado);
        printf("Operação: (%s AND %s) = %s\n", bin1, bin2, resultado);
    } else if (op == 2) {
        FazerOperacaoOR(bin1, bin2, resultado);
        printf("Operação: (%s OR %s) = %s\n", bin1, bin2, resultado);
    } else if (op == 3) {
        FazerOperacaoXOR(bin1, bin2, resultado);
        printf("Operação: (%s XOR %s) = %s\n", bin1, bin2, resultado);
    }
} else {
    char binario[TamBin + 1];
    printf("Digite o número binário [tam min 4-tam máx %d]: ", TamBin);
    fgets(binario, sizeof(binario), stdin);
    binario[strcspn(binario, "\n")] = '\0';

    while (!(4 <= strlen(binario) && strlen(binario) <= TamBin && ValidarBinario(binario))) {
        printf("Número binário inválido. Digite novamente: ");
        fgets(binario, sizeof(binario), stdin);
        binario[strcspn(binario, "\n")] = '\0';
    }

    FazerOperacaoNOT(binario, resultado);
    if (strcmp(resultado, "Tamanho inválido") == 0) {
        printf("Tamanho do número binário inválido.\n");
    } else {
        printf("Operação: NOT (%s) = %s\n", binario, resultado);
    }
}

printf("\nPressione a tecla <Enter> para continuar...");
getchar(); //Pressione <Enter> para continuar...
}

return 0;
} //Fim do programa "OperacoesLogicasBinarias

```