

CURSO DE GRADUAÇÃO CIÊNCIA DA COMPUTAÇÃO.
DISCIPLINA: ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES II.
PROFESSOR: THIAGO FELSKI PEREIRA

**ALUNOS: ROBERT CZELEN VITORINO, LUCAS OLIVEIRA PRADO E VICTOR
DA SILVA.**

TRABALHO M1

ITAJAÍ -SC

AGOSTO,2022


SUMÁRIO

1.	Programa 1	3
1.1.	Código fonte linguagem alto nível.....	3
2.	Caso de teste	6
2.1.1.	Arquivo txt utilizado para teste e instruções realizado no MARS.	6
2.1.2.	Saída do teste realizado com programa.....	7

1. PROGRAMA 1


1.1. Código fonte linguagem alto nível.

1.1.1. Início do programa,



```
1  int main() {  
2      lerArquivo();  
3  }
```

1.1.2. Estrutura utilizada para representar código binário,



```
1  struct OPcode{  
2      char formato;  
3      string nome;  
4      string code;  
5  };
```

1.1.3. Leitura do arquivo e inicio do programa, onde é chamado metodo buscaInstrução onde ocorre o processamento dos dados de um txt.

```

1 void lerArquivo(){
2     ifstream arquivoLer;
3     string linha = "";
4     OPcode listaOPcode[100];
5     inicializarOPcode(listaOPcode);
6     int ciclosTotal=0;
7     int instrucoes=0;
8     char letra;
9     arquivoLer.open ("texto.txt");
10    while(!arquivoLer.eof()){
11        arquivoLer.get(letra);
12        if(letra=='\n' || arquivoLer.eof()==1){
13            buscarInstrucao(pegarCodInstrucao(linha), listaOPcode, ciclosTotal);
14            instrucoes++;
15            linha = "";
16        }else{
17            linha += letra;
18        }
19    }
20    cout<<"Ciclos Totais: "<<ciclosTotal<<endl;
21    cout<<"CPI(media): "<<ciclosTotal / instrucoes<<endl;
22 }

```

1.1.4. Metodo de busca das instruções realizado no arquivo,

```

1 void buscarInstrucao(OPcode opcode, OPcode listaOPcode[100], int &ciclosTotal){
2     for(int i=0; i<100; i++){
3         if(opcode.formato=='r'){
4             if(opcode.code==listaOPcode[i].code && listaOPcode[i].formato=='r'){
5                 cout<<listaOPcode[i].nome<<" : "<<listaOPcode[i].formato<<" : "<<listaOPcode[i].code<<" : 4 ciclos"<<endl;
6                 ciclosTotal += 4;
7                 return;
8             }
9         }else{
10            if(opcode.code==listaOPcode[i].code && listaOPcode[i].formato!='r'){
11                cout<<listaOPcode[i].nome<<" : "<<listaOPcode[i].formato<<" : "<<listaOPcode[i].code;
12                if(opcode.code=="100011"){//caso seja lw
13                    cout<<" : 5 ciclos"<<endl;
14                    ciclosTotal+=5;
15                }
16                else if(opcode.code=="101011"){//caso seja sw
17                    cout<<" : 4 ciclos"<<endl;
18                    ciclosTotal+=4;
19                }
20                else{//caso qualquer outra instrução do formato i
21                    cout<<" : 3 ciclos"<<endl;
22                    ciclosTotal+=3;
23                }
24                return;
25            }
26        }
27    }
28 }

```

1.1.5. Método responsável localiza opcode de cada instrução,

```
1  Opcode pegarCodInstrucao(string code){
2      //cout<<code<<endl;
3      Opcode OPcode;
4      OPcode.code = "";
5      OPcode.formato = ' ';
6      for(int i=0; i<31; i++){
7          OPcode.code += code.at(i);
8          if(i==5){
9              if(OPcode.code=="000000"){//caso formatoo r
10                 code.erase(0, 26);
11                 OPcode.code = code;
12                 OPcode.formato = 'r';
13                 return OPcode;
14             }
15             else{
16                 code.erase(6, 31);
17                 OPcode.code = code;
18                 return OPcode;
19             }
20         }
21     }
22     return OPcode;
23 }
```

2. CASO DE TESTE

2.1.1. Arquivo txt utilizado para teste e instruções realizado no MARS.

```

1 .data
2 msg1 : .asciiz "entre com o valor de X: "
3 msg2 : .asciiz "entre com o valor de Y: "
4 resultado : .asciiz "NDC: "
5
6 .text
7 j main
8
9 ndc:
10 add $t0, $a0, $zero
11 add $t1, $a1, $zero
12
13 loop:
14 beq $t0, $t1, x_maior_y
15 bit $t0, $t1, x_menor_y
16 sub $t0, $t0, $t1
17 j x_maior_y
18
19 x_menor_y:
20 sub $t1, $t1, $t0
21
22 x_maior_y:
23 bne $t0, $t1, loop
24
25 add $v1, $zero, $t0
26
27 jr $ra
28
29 main:
30 #IMPRIME PRIMEIRA MENSAGEM
31 la $a0, msg1
32 addi $v0, $zero, 4
33 syscall
34
35 #SALVA VALOR 1
36 addi $v0, $zero, 5
37 syscall
38 add $t0, $zero, $v0 # t0 = x
39
40 #IMPRIME SEGUNDA MENSAGEM
41 la $a0, msg2
42 addi $v0, $zero, 4
43 syscall
44
45 #SALVA VALOR 2
46 addi $v0, $zero, 5
47 syscall
48 add $t1, $zero, $v0 # t1 = y
49
50 # BACKUP DOS DADOS
51 addi $sp, $sp, -8
52 sw $t0, 4($sp)
53 sw $t1, 0($sp)
54
55 # PASSAGEM DOS ARGUMENTOS
56 add $a0, $zero, $t0
57 add $a1, $zero, $t1
58
59 jal ndc
60
61 # RETORNO DOS VALORES
62 la $a0, 4($sp)
63 lw $t0, 0($sp)
64 addi $sp, $sp, 8
65
66 # RESULTADO
67 la $a0, resultado
68 addi $v0, $zero, 4
69 syscall
70
71 addi $v0, $zero, 1
72 add $a0, $zero, $t0
73 syscall
74
75 #FINALIZA PROGRAMA
76 addi $v0, $zero, 10
77 syscall


```

```

1 00001000000100000000000000001100
2 0000000010000001000000000100000
3 0000000010100001000100000100000
4 0001001000010001000000000000101
5 00000010000100010000100000101010
6 000101000010000000000000000010
7 00000010000100011000000000100010
8 00001000000100000000000000001001
9 00000010001100001000100000100010
10 00010110000100011111111111111001
11 0000000000100000001100000100000
12 0000001111100000000000000001000
13 001111000000001000100000000001
14 001101000010010000000000000000
15 001000000000010000000000000100
16 0000000000000000000000000001100
17 001000000000010000000000000101
18 0000000000000000000000000001100
19 000000000000101000000000100000
20 00111100000001000100000000001
21 001101000100100000000000011001
22 001000000000100000000000000100
23 0000000000000000000000000001100
24 001000000000100000000000000101
25 0000000000000000000000000001100
26 00000000000010100010000100000
27 0010001110111011111111111111000
28 10101111011000000000000000100
29 101011110110001000000000000000
30 000000000010000001000000100000
31 0000000000100010010100000100000
32 000011000001000000000000000001
33 100011110110000000000000000100
34 100011110110001000000000000000
35 00100011101110110100000000001000
36 0011110000000100010000000001
37 00110100001001000000000000110010
38 001000000000010000000000000100
39 0000000000000000000000000001100
40 00100000000001000000000000001
41 000000000000011001000000100000
42 0000000000000000000000000001100
43 00100000000010000000000001010
44 0000000000000000000000000001100

```

2.1.2. Saída do teste realizado com programa.



```
1  j : j : 000010 : 3 ciclos
2  add : r : 100000 : 4 ciclos
3  add : r : 100000 : 4 ciclos
4  beq : i : 000100 : 3 ciclos
5  uslt : r : 101010 : 4 ciclos
6  bne : i : 000101 : 3 ciclos
7  sub : r : 100010 : 4 ciclos
8  j : j : 000010 : 3 ciclos
9  sub : r : 100010 : 4 ciclos
10 bne : i : 000101 : 3 ciclos
11 add : r : 100000 : 4 ciclos
12 jr : r : 001000 : 4 ciclos
13 ori : i : 001101 : 3 ciclos
14 addi : i : 001000 : 3 ciclos
15 addi : i : 001000 : 3 ciclos
16 add : r : 100000 : 4 ciclos
17 ori : i : 001101 : 3 ciclos
18 addi : i : 001000 : 3 ciclos
19 addi : i : 001000 : 3 ciclos
20 add : r : 100000 : 4 ciclos
21 addi : i : 001000 : 3 ciclos
22 sw : i : 101011 : 4 ciclos
23 sw : i : 101011 : 4 ciclos
24 add : r : 100000 : 4 ciclos
25 add : r : 100000 : 4 ciclos
26 jal : j : 000011 : 3 ciclos
27 lw : i : 100011 : 5 ciclos
28 lw : i : 100011 : 5 ciclos
29 addi : i : 001000 : 3 ciclos
30 ori : i : 001101 : 3 ciclos
31 addi : i : 001000 : 3 ciclos
32 addi : i : 001000 : 3 ciclos
33 add : r : 100000 : 4 ciclos
34 addi : i : 001000 : 3 ciclos
35 Ciclos Totais: 120
36 CPI(media): 2
```