Trabalho Final GCC-108 - Teoria da Computação

Prof.: Douglas H. S. Abreu

Nome: Marco Antônio Magalhães

Turma: 10A

Link do repositório GitHub

● O trabalho deve ser feito em grupos de no máximo 2 componentes

● Trabalhos entregues após a data limite não serão aceitos

● Data limite de entrega: 29 de Abril de 2022 : 23h59m

● Enviar o trabalho para o campus virtual, do seguinte modo: Notebook exportado em PDF contendo o código e também o link do repositório GitHub para acesso aos arquivos. A Documentação deve estar no readme

● O trabalho deve ser desenvolvido no modelo Notebook utilizando a linguagem Python

---

Introdução

Este trabalho propõe a utilização de operações da aritmética computacional por meio de uma Máquina de Turing. A máquina que foi desenvolvida recebe como entrada dois números em binário e gera como saída o resultado da adição desses números.

Números binários e adição em números binários

Os números binários são utilizados para representar dados em um meio digital, como por exemplo, a representação no meio analógico com presença ou ausência de carga elétrica e no meio digital por meio de zeros e uns. Essa representação com dois símbolos utiliza-se da mesma técnica do telégrafo, que transmitia mensagens por código Morse, sendo os símbolos curto e longo análogos ao zero e um (1).

Utilizando-se a notação binária é possível representar uma faixa de valores diferentes de acordo com a quantidade de bits. Por exemplo, com dois bits pode-se representar quatro valores distintos, sendo eles 00, 01, 10 e 11. Ou seja, com n bits, podemos representar 2n valores distintos.

Para a notação de números inteiros usando a base binária de zeros e uns, podemos representar os números utilizando as seguintes representações: de binário puro, de binários em sinal magnitude e a representação em complemento de 2 (1).

Tomando como base a representação de números inteiros na base binária pura, que também é a representação utilizada neste trabalho, pode-se observar na Tabela 1, que com quatro bits temos as seguintes possibilidades para números inteiros.

Tabela1.png

As operações matemáticas de adição e subtração feitas na base binária seguem as mesmas regras da base decimal, contando com a diferença que temos apenas dois dígitos. Para a adição de dois números, temos quatro possibilidades de valores, sendo elas: 0 + 0, 0 + 1, 1 + 0, e 1 + 1. As três primeiras têm os mesmos resultados de uma operação em decimal, já para a operação de 1 + 1 temos como resultado zero, gerando um “vai um” para a coluna da esquerda (1).

Máquina de Turing

Turing descreve um computador digital como sendo formado por: uma unidade de armazenamento, uma unidade de execução e uma unidade de controle. A unidade de armazenamento é formada por uma fita, dividida em células, com um cabeçote apontando para a célula atual, a qual pode ser lida/escrita de acordo com a unidade de execução. Por sua vez, a unidade de execução tem como objetivo fazer a leitura do caractere representado na célula atual, analisar o que deve ser feito e alterar quando necessário. Já a unidade de controle faz as movimentações do cabeçote de acordo com o que a unidade de execução deseja, movendo o cabeçote para esquerda ou direita (2).

Exercício 1)

Descreva com suas palavras uma estratégia para o desenvolvimento de uma maquina de Turing que compute a soma de 2 numeros binário.

```
A maquina basicamente realiza a soma algarismo por algarismo e mantém o resultado invertido no final da fita.
Invertido pois o crescimento da fita tem que ser direcionado ao lado direito, manténdo assim o inicio da fita
```

```
na mesma posição. O inicio será alterado para 'b' durante a execução e voltará ao seu estado inicial ao fim
da execução. Os algarismos que já forem somados serão substituidos por alguma letra, no caso a letra 'x'. Um
sinal de '=' será colocado ao final das entrada para indicar o inicio da sequencia que constitui o resultado.
Nas situações em que ocorre o "vai um" na soma de 1+1 = 10, o algarismo extra será substido pela letra 'u'.
Se ao final da execução da soma ainda existir algum 'u' ele será substituido por 1 e o resultado será
invertido para a configuração correta e será movido para o inicio da fita. Os caracteres extras 'x' e '='
serão apagados.
```

Exercício 2)

Faça o esboço por meio de desenho da máquina de Turing proposta.

maquina.png

Exercício 3)

Defina a MT como uma quíntupla M=(Q,Σ,Γ,δ,q_0):

```
Q = conjunto de estados (padrão q[0-9]+)

Σ = alfabeto de entrada

Γ = alfabeto da fita

δ = função de transição no formato (q_i,x)→(q_j,y,D); assim, estando no estado q_i, lendo x, vai para o
estado q_j, escreve y e movimenta na direção de D. D será L para esquerda ou R para direita.

q_0 = estado inicial
```

$M = (Q, \Sigma, \Gamma, \delta, q0)$
$Q = \{q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17, q18, q19, q20, q21, q22, q23, q24, q25, q26, q27, q28, q29, q30\}$
$\Sigma = \{B, 0, 1\}$
$\Gamma = \{B, =, u, 0, x, b, 1\}$
$\delta(q0, B) \to (q1, b, R), \delta(q1, 0) \to (q1, 0, R), \delta(q1, 1) \to (q1, 1, R), \delta(q1, x) \to (q2, x, L), \delta(q1, B) \to (q2, B, L), \delta(q2, 0) \to (q4, x, R), \delta(q2, x)$
$\to (q3, x, R), \delta(q2, b) \to (q4, b, R), \delta(q3, x) \to (q3, x, R), \delta(q3, B) \to (q12, B, R), \delta(q4, B) \to (q13, B, R), \delta(q4, x) \to (q4, x, R), \delta(q5, B) \to ($
$\to (q6, x, R), \delta(q5, 0) \to (q7, x, R), \delta(q6, =) \to (q11, =, R), \delta(q6, x) \to (q6, x, R), \delta(q7, x) \to (q7, x, R), \delta(q7, =) \to (q9, =, R), \delta(q8, B) \to ($
$\to (q9, 1, R), \delta(q9, 0) \to (q9, 0, R), \delta(q9, u) \to (q10, 1, L), \delta(q9, B) \to (q10, 0, L), \delta(q10, b) \to (q1, b, R), \delta(q10, B) \to (q10, B, L), \delta(q10, x)$ -
$\to (q10, =, L), \delta(q10, 0) \to (q10, 0, L), \delta(q10, 1) \to (q10, 1, L), \delta(q11, u) \to (q14, 0, R), \delta(q11, 0) \to (q11, 0, R), \delta(q11, 1) \to (q11, 1, R), \delta(q1$
$\delta(q12, B) \to (q15, =, L), \delta(q12, 0) \to (q12, 0, R), \delta(q12, 1) \to (q12, 1, R), \delta(q12, x) \to (q15, x, L), \delta(q12, =) \to (q15, =, L), \delta(q13, =) \to (q2$
$\to (q5, =, L), \delta(q13, 0) \to (q13, 0, R), \delta(q13, 1) \to (q13, 1, R), \delta(q13, x) \to (q5, x, L), \delta(q14, B) \to (q10, u, L), \delta(q15, 0) \to (q6, x, R), \delta(q15,$
$\delta(q15, =) \to (q6, =, R), \delta(q15, B) \to (q6, B, R), \delta(q16, =) \to (q17, =, R), \delta(q16, x) \to (q16, x, R), \delta(q17, B) \to (q8, 0, R), \delta(q17, 1) \to (q17,$
$\to (q17, 0, R), \delta(q17, u) \to (q18, 1, R), \delta(q18, B) \to (q10, u, L), \delta(q19, x) \to (q19, x, R), \delta(q19, =) \to (q20, =, R), \delta(q20, 1) \to (q20, 1, R), \delta($
$\delta(q20, B) \to (q22, B, L), \delta(q20, u) \to (q21, 1, R), \delta(q21, B) \to (q22, B, L), \delta(q22, 1) \to (q23, B, L), \delta(q22, =) \to (q28, B, L), \delta(q22, 0) \to (q$
$\to (q23, 1, L), \delta(q23, x) \to (q23, x, L), \delta(q23, B) \to (q23, B, L), \delta(q23, =) \to (q23, =, L), \delta(q23, 0) \to (q23, 0, L), \delta(q23, b) \to (q24, b, R), \delta(q$
$\delta(q24, B) \to (q25, 1, R), \delta(q24, 1) \to (q24, 1, R), \delta(q24, 0) \to (q24, 0, R), \delta(q25, =) \to (q20, =, R), \delta(q25, B) \to (q25, B, R), \delta(q25, x) \to (q2$
$\to (q27, b, R), \delta(q26, =) \to (q26, =, L), \delta(q26, x) \to (q26, x, L), \delta(q26, 1) \to (q26, 1, L), \delta(q26, 0) \to (q26, 0, L), \delta(q26, B) \to (q26, B, L), \delta(q$
$\delta(q27, 1) \to (q27, 1, R), \delta(q27, 0) \to (q27, 0, R), \delta(q27, B) \to (q25, 0, R), \delta(q28, x) \to (q28, B, L), \delta(q28, 0) \to (q28, 0, L), \delta(q28, B) \to (q28$
$\to (q28, 1, L), \delta(q28, b) \to (q29, B, R), \delta(q29, 1) \to (q30, 1, L), \delta(q29, 0) \to (q30, 0, L), \delta(q29, B) \to (q30, B, L)$
$q0 = q0$

Exercício 4)

Faça a conversão de M em R(M)

R(M) =
000101011011111110110011011110110111101100110111111101101111111011001101111101110111110100110101110101001110111101111101111

Exercício 5)

Desenvolva uma função MTU que receba R(M) acrescido de uma entrada w, onde w é um arquivo csv que contem dois números binário. A saída da função MTU deve ser a computação de M para uma entrada w.

In [42]:
```python
import pandas as pd

# Entrada de dados
entrada = pd.read_csv('exemplo2.CSV')
entrada = 'B'+'B'.join(str(entrada.columns[0]).split(';'))
print('Entrada: ')
print(entrada)
print('\n')


# Configuração inicial
inicial_code = 0
final_code = 30
inicial = '1'*inicial_code
final = '1'*final_code

fita_code = {'1':'B','2':'=','3':'u','4':'0','5':'x','6':'b','7':'1'}
direcao_code = {'1':'L','2':'R'}

entrada_w = ''
for char in entrada:
    char_code = '1'*int(list(fita_code.keys())[list(fita_code.values()).index(char)])
    entrada_w+=char_code+'0'
entrada_w+='00'

mr = "00010101101111110110011011110110111101100110111111101101111110111111101100110111101110111110100110101110101001110111101111101111
mrw = mr+entrada_w

# função MTU = {R(M)| R(M) aceite w}
# O código converte a sequencia de r(M) para que a MTU execute as operações mais simplificadamente
# A converção é baseada na configuração inicial
def MTU (mr,w):
    x = mr.split("000")
    #print(x)
    trans = x[1].split("00")
    fita = list(w+'B'*(len(w)+5))
    a = dict()
    b = dict()
```

```python
    for tra in trans:
        #print(tra.split("0"))
        ori,rec,des,res,dir = tra.split("0")
        b[ori,rec] = des,res,dir
        a[len(ori)-1,fita_code.get(str(len(rec)))] = len(des)-1,fita_code.get(str(len(res))),direcao_code.get(str(len(dir)))
    #print(b)
    #print(a)
    state = inicial_code
    i=0
    while(state!=final_code):
        #print(fita)
        read = fita[i]

        state2,write,direction = list(a.get((state,read)))

        state = state2
        #print(state2,write,direction)
        fita[i] = write
        if direction == 'R':
            i+=1
        else:
            i-=1

    #print(fita)
    try:
        while True:
            fita.remove('B')
    except ValueError:
        pass

    print('Resultado: ')
    print(''.join(str(io) for io in fita))


MTU(mr,entrada)
```

```
Entrada:
B1011000B1101


Resultado:
1100101
```

Exercício 6)

A) Explique a Tese de Chuch-Turing de forma sucinta

```
  Sucintamente, a tese pode ser considerada uma tentativa de se determinar precisamente o alcance e os
limites da computação teórica. O enunciado estrito diz que toda função algoritimica é turing-computável e o
enunciado extendido diz que toda função parcial algoritimica é parcialmente turing-computável.
```

B) Dada uma máquina de Turing arbitrária M e uma string de entrada w, a computação de M com entrada w irá parar em menos de 100 transições? Descreva uma máquina de Turing que resolva esse problema de decisão.

```
Não é possível afirmar genericamente isso. A definição de uma máquina que diz isso sempre resulta em
contradições tornando impossível provar a existencia.
```

C) Motre a solução para cada um dos seguintes sitemas de correspondência de Post:

a) (a, aa), (bb, b), (a, bb)

```
a   |a   |bb |b b
a a| bb|  b|  b
1   3   2   2

Um inicio com 2 não teria continuidade.
```

b) (a, ab), (ba, aba), (b, aba), (bba, b)

```
a   |b b a|  ba
a b|  b   |a ba
1   4     3

Um inicio com 2 não teria continuidade
```

c) (abb, ab), (aba, ba), (aab,a bab)

```
ab b|  a ba|
ab  |b a   |
1    2      2...
```

```
Somente 1 pode iniciar e somente 2 pode sucede-lo. Mas somente 2 pode suceder 2, gerando assim uma sequencia
infinita inválida 1222...
```

d) (ab,aba), (baa, aa), (aba, baa)

```
ab  |a ba  |
ab a|  ba a|
1    3      3...
```

```
Mesmo problema da letra anterior mas agora com o 3. Uma sequencia infinita e incorreta da forma 1333...
```

e) (a, aaa), (aab, b), (abaaa, ab)

```
a   |aa b
a aa|   b
1    2
```

```
Somente 1 e 3 podem iniciar, mas um inicio com 3 gera uma sequencia infinita e incorreta.
```

f) (ab, bb), (aa, ba), (ab, abb), (bb, bab)

```
Como somente possui peças com 2/2 ou 2/3 algarismos, fica impossível de encontrar uma seguencia a não ser que
seja formada somente por 1 e 2. Mas 1 e 2 não podem se quer iniciar uma sequencia válida.
```

D)

a) Prove que a função é primitiva recursiva minimizacao.png

sempre que *p* e *u* são recursivas primitivas

```
O operador de minimalização limitada define uma função recursiva primitiva sempre que o predicado for
recursivo primitivo. Como esse é o caso, pois p é recursiva primitiva, então cai no caso similar a prova de
somatório ou produto limitado.
```

b) defina o valor "passo a passo" de *gn(4,1,0,2,1)* =

$$gn(4,1,0,2,1) = pn(0)^4 * pn(1)^1 * pn(2)^0 * pn(3)^2 * pn(4)^1$$
$$gn(4,1,0,2,1) = 2^4 * 3^1 * 5^0 * 7^2 * 11^1$$
$$gn(4,1,0,2,1) = 16 * 3 * 1 * 49 * 11$$
$$gn(4,1,0,2,1) = 25872$$

E)

a) Dado *f(x) = 3x² + 4x + 6* e *g(x) = 5x²*

Prove que *g(x) ∈ O(f)* e *f(x) ∈ O(g)*

```
f e g são polinomios de mesma ordem.
Em f especificamente, o fator decisivo é o 3x², sendo que 4x e 6 será menores que 3x² para qualquer x>1.
Tem-se: f(x) = 3x² + 4x + 6 <= 3x² + 4x² + 6x² = 13x². Que por sua vez é da forma 13x² = c1*g(x). Isso
implica em limites similares para as duas funções, e consequentemente que possuem mesmo O(f)=O(g).
```

b) Qual é a complexidade e o "*big O*" de M'?

MT.png

```
Contando o número de transições a cada passo, conclui-se que a complexidade
de tempo de M é
```

$tcM\ n = 3\ (n + 1) + 1$

A complexidade O(3n+4) é, por definição, análoga a O(n).

Referências

(1) *Ronald. J. Tocci, Neal. S. Widmer e Gregory L. Moss. 2011. Sistemas Digitais: Princípios e Aplicações. (11ª ed.). Pearson.*

(2) *Alan Turing. 1937. Computability and λ-definability. Journal of Symbolic Logic, 2, 4: 153–163.*

(3) *Sudkamp, T. A. 2006. Languages and machines: an introduction to the theory of computer science. 3rd Edition*

In [ ]:

```
```

Links úteis:

Link do site Jupyter

Link do site Anaconda

Link para ajuda com Markdown no Notebook