PROVA 1 - ARQUITETURA DE COMPUTADORES

Aluno: Daniel Sant' Anna Andrade
Matrícula: 20200036904

1)

```
INTERVALO .equ 0x0000c @ define "INTERVALO" com um valor de "0xc"

    .org 0x10          @ inicia o código na posição de memória 0x10
_start:
    set r2, 0          @ inicia "r2" com o valor de 0
    set r1, lista      @ inicia "r1" com a posição de memória da "lista" (0x200)
    set r3, 1          @ inicia "r3" com o valor de 1

loop:
    ld r0, [r1]        @ carrega em "r0" o valor da posição de memória de "r1"
    tst r0 , r3        @ testa "r0" com "r3" vendo se o bit mais significativo dos dois
                       @ são iguais ou não

    jz xxxx            @ desvia para "xxxx" se a comparação de teste for 0, ou seja, "r0"
                       @ seja um número par

    add r0, 1          @ adiciona 1 em "r0"
    st [r1], r0        @ seta o valor de "r0" no valor da posição de memória de "r1"

xxxx:
    add r2, 1          @ adiciona 1 em "r2"
    add r1, 4          @ adiciona 4 em "r1" para acessar o próximo valor da lista
    cmp r2, INTERVALO      @ compara "r2" com "INTERVALO"
    jge yyyy           @ pula para "yyyy" caso a "r2" seja maior ou igual a "INTERVALO"
    jmp loop           @ pula para loop

yyyy:
    hlt                @ termina o programa

    .org 0x200         @ inicia a lista na posição de "0x200"
lista: .word 39, 1, 2, 2, 10, 81, 16, 10, 332, 1, 19, 0, 1
```

Esse programa verifica cada um dos números da lista, quando um dos números for ímpar, ou seja o bit mais significativo dele for 1, esse número será modificado na lista somando mais um, o transformando em um número par. No final todos valores de lista serão números pares. A nova lista será 40, 2, 2, 2, 10, 82, 16, 10, 332, 2, 20, 0, 2.

2)

```
TAMANHOMAX .equ 0x00064      @ inicia o "TAMANHOMAX" com o valor de 100 ("0x64")

  .org 0x10            @ inicia o código em "0x10"
_start:
  set r1, list         @ inicia "r1" com a posição de "list", ou seja "0x200"
  set r2, sum          @ inicia "r2" com a posição de "sum", ou seja "0x190"
```

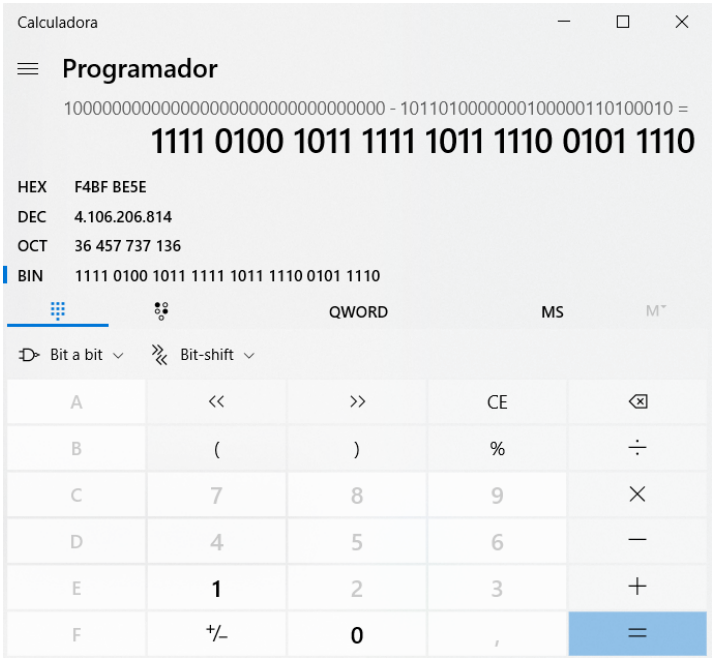
```
  set r3, 0             @ inicia "r3" com o valor de 0

loop:
  ld r0, [r1]           @ carrega em "r0" o valor da posição de "r1"
  add r3, r0            @ adiciona "r0" em "r3"
  cmp r3, TAMANHOMAX   @ compara "r3" com "TAMANHOMAX"
  jg end                @ caso "r3" seja maior que "TAMANHOMAX" pula para "end"

next_value:

  add r1, 4             @ adiciona 4 em "r1" para acessar o próximo item da "list"
  jmp loop              @ pula para "loop" e reinicia a soma e comparação

end:
  stb [r2], r3          @ envia o valor de "r3" para o endereço de "r2" que é a "sum"
                        @ iniciado em "0x190"

  hlt                   @ termina o programa

  .org 0x190
sum:  .word 0
  .org 0x200
list: .word 5, 39, 1, 2, 2, 10, 15, 16, 10, 20, 10, 1, 20, 25, 10, 5, 39, 1, 2, 2, 10, 15, 16, 10, 20
```

O programa adiciona todos os valores da lista no r3, quando o valor de r3 for maior que 100, é colocado esse valor no endereço 0x190 (soma) e termina o programa.

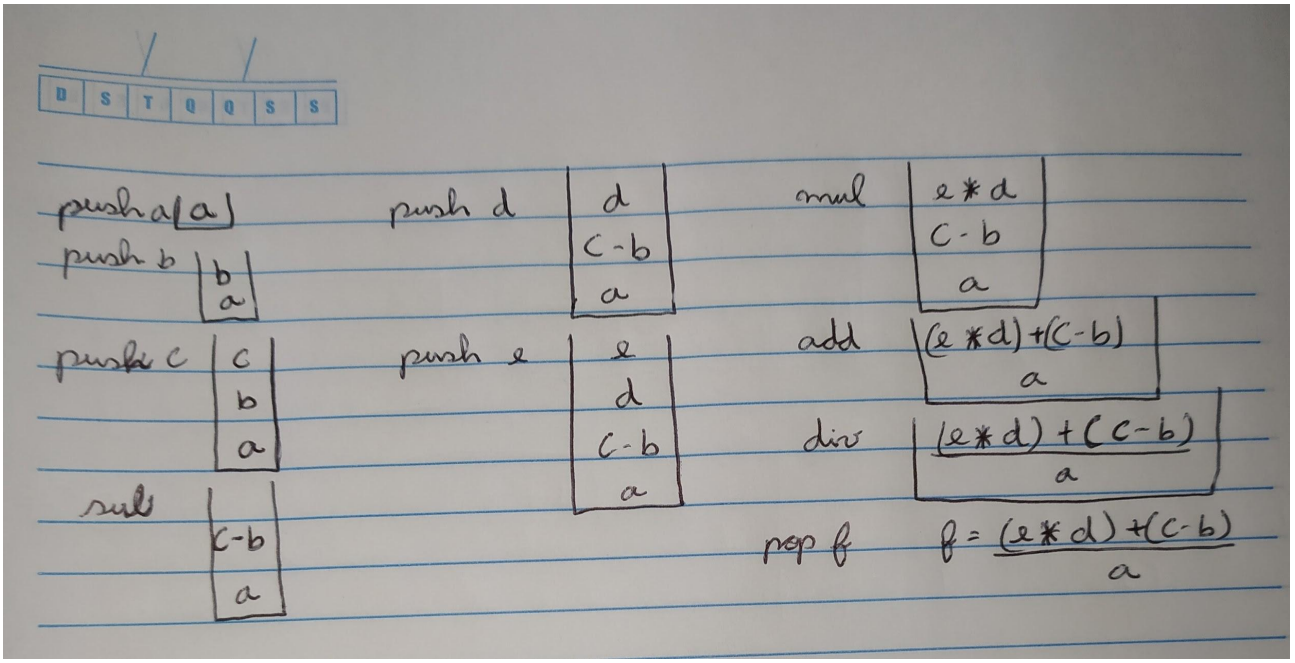
3)
Matrícula -> 20200036904
Em binário fica -> 100 1011 0100 0000 0100 0001 1010 0010 1000

Como só é para utilizar 32 bits, será utilizado o 1011 0100 0000 0100 0001 1010 0010 1000. Para ocorrer overflow a faixa deverá ultrapassar os 32 bits, ou seja, gerar um número de 33 bits (um número maior ou igual a 4.294.967.296). Diminuindo então 1 0000 0000 0000 0000 0000 0000 0000 0000 (4.294.967.296) de 1011 0100 0000 0100 0001

1010 0010 (188.760.482) irá resultar no valor de 1111 0100 1011 1111 1011 1110 0101 1110 (4.106.206.814).



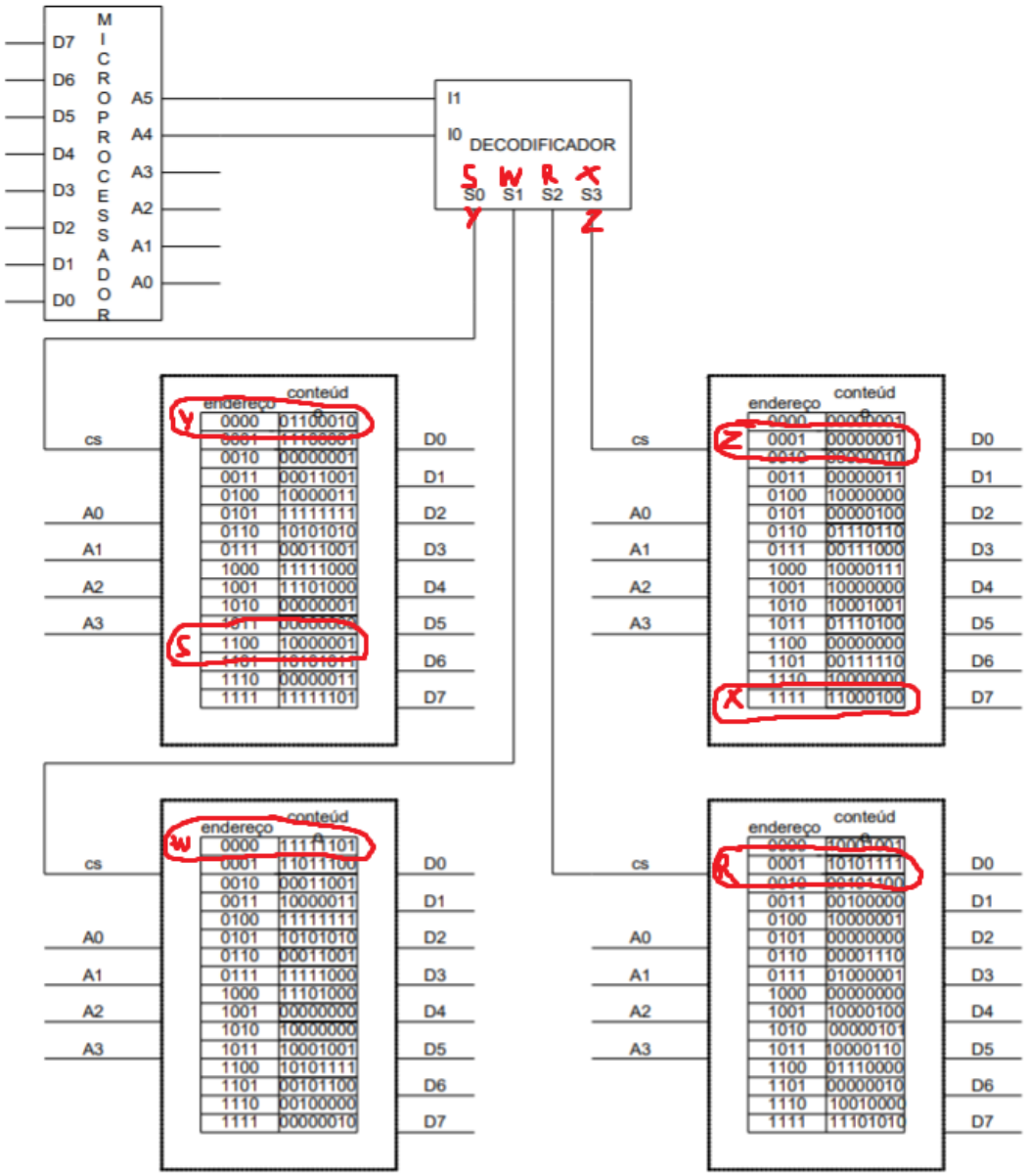
4)
fórmula



Arquitetura de 1 endereço

e1	lda e
e1+1	mul d
e1+2	add c
e1+3	sub b
e1+4	div a
e1+5	sta f
e1+6	hlt

5)



Decod	Endereço	Conteúdo
W 10_{16} →	$0001\ 0000_2$ →	$1111\ 1101_2$
X $3F_{16}$ →	$0011\ 1111_2$ →	$1100\ 0100_2$
R 21_{16} →	$0010\ 0001_2$ →	$1010\ 1111_2$
S $0C_{16}$ →	$0000\ 1100_2$ →	$1000\ 0001_2$
Y 00_{16} →	$0000\ 0000_2$ →	$0110\ 0010_2$
Z 31_{16} →	$0011\ 0001_2$ →	$0000\ 0001_2$

$$\begin{array}{r} 1111\ 1101_2 \\ +1100\ 0100_2 \\ \hline 11100\ 0001_2 \end{array}$$

W + X dá overflow em sinal magnitude

$$1010\ 1111_2 - 1000\ 0001_2 \Rightarrow$$

$$\begin{array}{r} 1010\ 1111_2 \\ +0111\ 1110_2 \\ \hline 10010\ 1101_2 \\ \longrightarrow\ 1+ \\ \hline 0010\ 1110_2 \end{array}$$

R-S não da overflow em compelemento de 1

$$\begin{array}{r} 0110\ 0010_2 \\ +\ 0000\ 0001_2 \\ \hline 0110\ 0011_2 \end{array}$$

Y + Z não da overflow em complemento de 2