Alunos:

- **Gabriel Vinicius de Menezes Gama**
  - **RA 134997**
- **Lucas Xavier**
  - **RA 132996**

1) Programa Tradutor IAS:

```
from sys import *
from math import *



############################## MEMÓRIA - REGISTRADORES ##############################

MEMORIA = []

AC = 0          #ACCUMULATOR
MQ = 0          #MULTIPLAYER QUOTIENT
IR = 0          #INSTRUCTION REGISTER
PC = 0          #PROGRAM COUNTER
MAR = 0         #MEMORY ACESS REGISTER
MBR = 0         #MEMORY BUFFER REGISTER




############################### CICLO DE INSTRUÇÃO ################################

def busca():
    global MEMORIA, PC, MAR, MBR
    #
    MAR = PC
    MBR = MEMORIA[MAR]

def decodificacao():
    global IR, MAR
    #
    IR, MAR = separaMBR()

def buscaDosOperandos():
    global MEMORIA, MBR, MAR
    #
```

```
    MBR = MEMORIA[MAR]

def execucao():
    global MEMORIA,AC,MQ,IR,PC,MAR,MBR
    #
    intCheck()
    #
    match IR:
                    ########## TODAS FUNÇÕES LOAD ##########
        case 'LOADM':
            AC = MBR
        case 'LOADMQ':
            AC = MQ
        case 'LOADMQM':
            MQ = MBR
        case 'LOAD-M':
            AC = MBR*(-1)
        case 'LOAD|M':
            AC = sqrt(MBR**2)
        case 'LOAD-|M':
            AC = (sqrt(MBR**2)) * (-1)
                    ########## FUNÇÕES ARITMÉTICAS ##########
        case 'ADDM':
            AC += MBR
        case 'ADD|M':
            AC += sqrt(MBR**2)
        case 'SUBM':
            AC -= MBR
        case 'SUB|M':
            AC -= sqrt(MBR**2)
        case 'DIVM':
            MQ = AC // MBR
            AC = AC % MBR
        case 'MULM':
            AC = MQ * MBR
        case 'LSH':
            AC *= 2
        case 'RSH':
            AC /= 2
                    ########## STOR E FUNÇÕES JUMP ##########
```

```
        case 'STORM':
            MEMORIA[MAR] = AC
        case 'JUMPM':
            PC = MAR - 1
        case 'JUMP+M':
            if AC >= 0:
                PC = MAR - 1




############################## FUNÇÕES AUXILIARES ################################

def carrega_memoria(arquivo):
    with open(arquivo,'r') as fin:
        mem = fin.readlines()
    mem = [item.replace('\n','') for item in mem]
    return mem

def inicia_PC():
    global MEMORIA
    for i in range(len(MEMORIA)):
        try:
            int(MEMORIA[i])
        except:
            return i

def printDadosMemoria():
    for i in range(inicia_PC()):
        print(f'    M({i})  {MEMORIA[i]}')

def separaMBR():
    global MBR
    parenteses = False
    MBR = MBR.replace(' ','')
    for i in range(len(MBR)):
        if MBR[i] == '(':
            parenteses = True
            break
    #
```

```
    if parenteses == True:
        MBR_OPCODE = MBR[0:i]
        MBR_OPCODE = MBR_OPCODE.replace(',','')
    elif parenteses == False:
        MBR_OPCODE = MBR[0:i+1]
    #
    MBR_LEAST_SIG_BITS = MBR[i:]
    MBR_LEAST_SIG_BITS = MBR_LEAST_SIG_BITS.replace('M','')
    MBR_LEAST_SIG_BITS = MBR_LEAST_SIG_BITS.replace(',','')
    MBR_LEAST_SIG_BITS = MBR_LEAST_SIG_BITS.replace('(','')
    MBR_LEAST_SIG_BITS = MBR_LEAST_SIG_BITS.replace(')','')
    MBR_LEAST_SIG_BITS = MBR_LEAST_SIG_BITS.replace('|','')
    MBR_LEAST_SIG_BITS = MBR_LEAST_SIG_BITS.replace(' ','')
    #
    try:
        MBR_LEAST_SIG_BITS = int(MBR_LEAST_SIG_BITS)
    except:
        MBR_LEAST_SIG_BITS = 0
    #
    return MBR_OPCODE, MBR_LEAST_SIG_BITS

def intCheck():
    global AC, MQ, MBR, PC
    try:
        AC = int(AC)
    except:
        pass
    try:
        MQ = int(MQ)
    except:
        pass
    try:
        MBR = int(MBR)
    except:
        pass
    try:
        PC = int(PC)
    except:
        pass
```

```
################################### PROCESSADOR ###################################

def processador():
    global PC
    PC = inicia_PC()
    while PC < len(MEMORIA):
        busca()
        decodificacao()
        buscaDosOperandos()
        execucao()
        PC += 1



################################ INICIO DO PROGRAMA ################################

MEMORIA = carrega_memoria(argv[1])

processador()

printDadosMemoria()
```

2) Programa com todas as instruções:

```
2
-2
2
-2
2
-2
2
2
1
-1
LOAD MQ,M(0)
LOAD MQ
ADD M(1)
STOR M(0)
LOAD M(1)
ADD |M(1)|
STOR M(1)
LOAD M(2)
SUB M(3)
STOR M(2)
LOAD M(3)
SUB |M(3)|
STOR M(3)
LOAD MQ,M(4)
MUL M(5)
STOR M(4)
LOAD |M(4)|
LSH
STOR M(5)
LOAD -|M(6)|
RSH
STOR M(6)
LOAD -M(5)
DIV M(6)
LOAD MQ
STOR M(7)
JUMP M(40)
```

```
LOAD M(2)
ADD M(2)
STOR M(2)
LOAD M(9)
JUMP +M(5)
LSH
STOR M(9)
LOAD M(8)
JUMP +M(49)
LOAD M(2)
ADD M(2)
STOR M(2)
LSH
STOR M(8)
```

3) **Programa da prova 1 (Bhaskara):**

```
[valor de a]
[valor de b]
[valor de c]
4
-1
0
0
0
2
0
1
LOAD MQ,M(0)
MUL M(2)
STOR M(5)
LOAD MQ,M(5)
MUL M(3)
STOR M(5)
LOAD MQ, M(1)
MUL M(1)
SUB M(5)
STOR M(5)
```

```
JUMP M(26)
LOAD M(7)
SUB M(10)
STOR M(7)
JUMP M(34)
LOAD M(5)
ADD M(10)
STOR M(9)
LOAD MQ,M(5)
LOAD MQ()
DIV M(8)
LOAD MQ()
STOR M(7)
LOAD MQ, M(7)
MUL M(7)
SUB M(9)
JUMP +M(22)
LOAD MQ, M(0)
MUL M(8)
STOR M(0)
LOAD MQ, M(1)
MUL M(4)
STOR M(1)
ADD M(7)
DIV M(0)
LOAD MQ()
STOR M(6)
LOAD M(1)
SUB M(7)
DIV M(0)
LOAD MQ()
STOR M(1)
LOAD M(6)
STOR M(0)
```

4) **Programa multiplicação matrizes:**

0 a 3 é a matriz produto, 4 a 7 é a primeira matriz, 8 a 11 é a segunda matriz
EXEMPLO:
[-1 3] * [1 3] = [8 10]
[4 2] [2 4] [10 16]

```
0
0
0
0
-1
3
4
2
1
2
3
4
0
LOAD MQ, M(4)
MUL M(8)
STOR M(12)
LOAD MQ, M(5)
MUL M(10)
ADD M(12)
STOR M(0)
LOAD MQ, M(4)
MUL M(9)
STOR M(12)
LOAD MQ, M(5)
MUL M(11)
ADD M(12)
STOR M(1)
LOAD MQ, M(6)
MUL M(8)
STOR M(12)
LOAD MQ, M(7)
```

```
MUL M(10)
ADD M(12)
STOR M(2)
LOAD MQ, M(6)
MUL M(9)
STOR M(12)
LOAD MQ, M(7)
MUL M(11)
ADD M(12)
STOR M(3)
```