

Universidade Federal do Rio Grande do Norte Centro de Tecnologia - CT

Departamento de Engenharia Elétrica - DEE

Disciplina: ELE1717 - Sistemas Digitais Período: 2018.1 Aluno: Data: 27/03/2018

- $\mathbf{1}$ (0,5) Desenvolva um código fonte em Assembly para retornar o valor e a posição do maior número em uma lista de 20 valores possíveis (em caso de números iguais retorne a posição de qualquer um deles). O valor do número deverá ser colocado no endereço 232 da memória e a sua posição na lista (entre 1 e 20) no endereço 233 da memória.
- **2** (0,5) Desenvolva um código fonte em Assembly para verificar se uma palavra (de no máximo 20 caracteres) é ou não palíndromo. Caso a palavra seja palíndromo, escreva no endereço 232 da memória o valor 48, do contrário escreve no endereço 232 da memória o valor 49.
- 3 (0,5) Desenvolva um código fonte em Assembly para multiplicar dois números de 8bits e obter um resultado em um número de 16bits. Os valores de 8bits devem ser colocados nos endereços 0 e 1 da memória e o resultado de 16bits deve ser colocado nos endereços 232(MSB) e 233(LSB).
- 4 (0,5) Desenvolva um código fonte em Assembly para calcular o número de Fibonacci (que resulte em um valor máximo de 16bits). O valor de entrada para a sequência de Fibonacci deverá ser colocado no endereço 0 da memória. O resultado de 16bits deve ser colocado nos endereços 232(MSB) e 233(LSB).

$$F(n) \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ F(n-1) + F(n-2), & n > 1 \end{cases}$$
 (1)

Observações

A interpretação das questões faz parte da prova;

Mov reg, regX reg=regX reg=mem regX	Id	Instrução	Dosariaño
MOV reg, address reg=mem regX		Instrução MOV reg. regX	Descrição
MOV reg, loaddress reg=mem[address]			
MOV reg, constant reg=constant		MOV reg. [address]	
5 MOV regl, regX mem[reg] = regX 6 MOV address , reg mem[address] = reg 7 MOV address , reg mem[address] = reg 8 ADD reg, regX reg=reg+mem regX 9 ADD reg, regX reg=reg+mem regX 10 ADD reg, address reg=reg+mem regX 11 ADD reg, constant reg=reg+regX 12 SUB reg, regX reg=reg-mem[address] 13 SUB reg, regX reg=reg-mem[address] 14 SUB reg, address reg=reg-mem[address] 15 SUB reg, address reg=reg-mem[address] 16 MUL reg A = A ^*rem address 17 MUL reg A = A ^*rem address 18 MUL address A = A ^*rem address 19 MUL constant A = A ^*rem address 10 AND reg, regX reg=reg AND regX 10 AND reg, regX reg=reg AND mem[regX] 12 AND reg, regX reg=reg AND mem[regX] 14 ADD reg, regX reg=reg AND mem[regX] 15 SUB reg, address reg=reg AND mem[regX] 16 MUL constant A = A ^*rem address 19 MUL constant A = A ^*rem address 10 AND reg, regX reg=reg AND mem[regX] 21 AND reg, regX reg=reg NO regX 22 AND reg, regX reg=reg NO regX 23 AND reg, regX reg=reg NO regX 24 OR reg, regX reg=reg NO regX 25 OR reg, regX reg=reg OR mem[regX] 26 OR reg, regX reg=reg OR mem[regX] 27 OR reg, constant reg=reg XOR regX 28 XOR reg, regX reg=reg XOR mem[regX] 30 XOR reg, regX reg=reg XOR mem[regX] 31 XOR reg, regX reg=reg XOR mem[regX] 32 SHL reg, regX reg=reg XOR mem[regX] 33 SHL reg, regX reg=reg XOR mem[regX] 34 SHL reg, regX reg=regx XOR mem[regX] 35 SHL reg, regX reg=regx XOR regX 36 SHR reg, regX reg=regx XOR regx 37 SHR reg, regX reg=regx XOR regx 38 SHR reg, regX reg=regx XOR regx 39 SHR reg, regX reg=regx XOR regx 40 CMP reg, regX reg=regx XOR regx 41 CMP reg, regX reg=regx XOR regx 42 CMP reg, regX reg=regx XOR regx 43 CMP reg, regX reg=regx XOR regx 44 INC reg regX reg=regx XO			
MOV address , constant mem address =cng mem address =cnstant	$\overline{}$		
MOV address . Constant mem address =constant			
8 ADD reg, regX reg=reg+regX 9 ADD reg, [regX] reg=reg+mem [regX] 10 ADD reg, [address] reg=reg+mem [address] 11 ADD reg, constant reg=reg+constant 12 SUB reg, regX reg=reg-regx 13 SUB reg, [regX] reg=reg-mem [address] 14 SUB reg, [address] reg=reg-mem [address] 15 SUB reg, constant reg=reg-constant 16 MUL reg A = A ^*rem [regX] 17 MUL [reg] A = A ^*mem [regX] 18 MUL address A = A ^*mem [regX] 19 MUL constant A = A ^*mem [regX] 10 AND reg, regX reg=reg AND regX 11 AND reg, [regX] reg=reg AND mem [regX] 12 AND reg, [regX] reg=reg AND mem [regX] 13 AND reg, [regX] reg=reg AND mem [regX] 14 OR reg, regX reg=reg OR regX 15 OR reg, [regX] reg=reg OR regX 16 OR reg, [regX] reg=reg OR mem [regX] 17 OR reg, [regX] reg=reg OR regX 18 OR reg, [regX] reg=reg OR regX 19 OR reg, [regX] reg=reg OR regX 10 OR reg, [regX] reg=reg OR rem[regX] 10 OR reg, [regX] reg=reg OR rem[regX] 11 OR reg, [regX] reg=reg OR regX 12 OR reg, [regX] reg=reg OR regX 13 OR reg, [regX] reg=reg XOR mem [regX] 14 OR reg, constant reg=reg XOR mem [regX] 15 OR reg, [regX] reg=reg XOR mem [regX] 16 OR reg, [regX] reg=reg XOR mem [regX] 17 OR reg, [regX] reg=reg XOR mem [regX] 18 OR reg, [regX] reg=reg XOR mem [regX] 19 OR reg, [regX] reg=reg XOR mem [regX] 10 OR reg, [regX] reg=reg XOR mem [regX] 11 OR reg, [regX] reg=reg XOR mem [regX] 12 OR reg, [regX] reg=reg XOR mem [regX] 13 OR reg, [regX] reg=reg XOR mem [regX] 14 OR reg, [regX] reg=reg XOR mem [regX] 15 OR reg, [regX] reg=reg XOR mem [regX] 16 OR reg, [regX] reg=reg XOR mem [regX] 17 OR reg, [regX] reg=reg XOR mem [regX] 18 OR reg, [regX] reg=reg XOR mem [regX] 19 OR reg, [regX] reg=reg XOR mem [regX] 19 OR reg, [regX] reg=regX 10 OR reg, [regX] reg=regX 11 OR reg, [regX] reg=regX 12 OR reg, [regX] reg=regX 13 OR reg, [regX] reg=regX 14 OR reg, [regX] reg=regX 15 OR reg, [regX] reg=regX 16 OR reg,	7		
9 ADD reg, [regX] reg=reg+mem[regX] 10 ADD reg, constant reg=reg+mem[address] 11 ADD reg, constant reg=reg+mem[address] 12 SUB reg, regX reg=reg-regX 13 SUB reg, [regX] reg=reg-mem[regX] 14 SUB reg, loaddress reg=reg-mem[regX] 15 SUB reg, constant reg=reg-constant 16 MUL reg	8		
11	9	ADD reg, [regX]	reg=reg+mem[regX]
12 SUB reg., regX reg=reg.mem[regX] 13 SUB reg., [address] reg=reg.mem[regX] 14 SUB reg., [address] reg=reg.mem[regX] 15 SUB reg., constant reg=reg.constant 16 MUL reg [A]=A]*rem[reg] 17 MUL [reg [A]=A]*mem[reg] 18 MUL address [A]=A]*mem[reg] 19 MUL constant [A]=A]*constant 20 AND reg., regX reg=reg AND regX 21 AND reg., [regX] reg=reg AND mem[regX] 22 AND reg., [address] reg=reg AND mem[address] 23 AND reg., [catX] reg=reg AND mem[address] 24 OR reg., regX reg=reg OR mem[regX] 25 OR reg., [regX] reg=reg OR mem[address] 26 OR reg., [address] reg=reg OR mem[address] 27 OR reg., constant reg=reg OR constant 28 XOR reg., regX reg=reg OR constant 29 XOR reg., [regX] reg=reg XOR regX 29 XOR reg., [regX] reg=reg XOR mem[regX] 30 XOR reg., [address] reg=reg XOR mem[regX] 31 XOR reg., constant reg=reg XOR mem[regX] 32 SHL reg., regX reg=reg=x XOR mem[regX] 33 SHL reg., [regX] reg=reg≪mem[regX] 34 SHL reg., [regX] reg=reg≪mem[regX] 35 SHL reg., constant reg=reg≪mem[regX] 36 SHR reg., regX reg=reg=x constant 37 SHR reg., regX reg=reg=x constant 40 CMP reg., regX reg=reg=x constant 40 CMP reg., regX reg=reg=x constant 40 CMP reg., regX reg=reg=x constant 41 CMP reg., regX reg=reg=x constant 42 CMP reg., constant reg=reg x constant 43 CMP reg., constant reg=reg x constant 44 INC reg reg=x reg=reg x constant 45 DEC reg reg=x reg=reg x constant 46 DCP reg., regX reg=reg=x constant 47 CALA address if reg=mem[regX], Z=1 48 DVSH reg SP = reg. SP=SP-1 49 PUSH [reg SP = reg.] SP = reg. SP=SP-1 50 PUSH address SP = reg=reg. SP=SP-1 51 DVSH constant SP = reg=reg. SP=SP-1 52 DVP reg reg=reg. SP = reg=reg. SP=SP-1 53 JMP address PC=mem[address], if carry=false reg=reg. SP = reg. SP=SP-1	10	ADD reg, [address]	reg=reg+mem[address]
13 SUB reg., [regX] reg=reg-mem[regX] 14 SUB reg., [address] reg=reg-mem[address] 15 SUB reg., constant reg=reg-constant 16 MUL reg			
14 SUB reg, [address] reg=reg-constant			
15 SUB reg, constant reg=reg 16 MUL reg [A]=A]*reg [A]=A]*mem[reg] [A]=A]*mem[reg] [A]=A]*mem[reg] [A]=A]*mem[reg] [A]=A]*mem[reg] [A]=A]*mem[reg] [A]=A]*mem[address]	$\overline{}$		
[A] = [A]* Feg [A] = [A] * Feg [A] * Fe			
17	_		
18 MUL address [A] = [A]*constant [A			
19 MUL constant [A]=[A]*constant 20 AND reg, regX reg=reg AND regX reg=reg AND regX reg=reg AND mem[regX] reg=reg AND mem[regX] reg=reg AND mem[address] 22 AND reg, constant reg=reg AND constant reg=reg AND constant reg=reg AND constant reg=reg AND regX reg=reg OR regX reg=reg OR regX reg=reg OR mem[address] reg=reg OR mem[address] reg=reg OR mem[address] reg=reg OR mem[address] reg=reg OR constant reg=reg OR constant reg=reg OR constant reg=reg XOR regX reg=reg XOR mem[address] reg=reg XOR constant reg=reg XOR mem[address] reg=reg XOR mem[address] reg=reg XOR mem[address] reg=reg XOR constant r			[A]=[A]*mem[reg]
20 AND reg, regX reg=reg AND regX 21 AND reg, [regX] reg=reg AND mem[regX] 22 AND reg, constant reg=reg AND constant 23 AND reg, constant reg=reg OR regX 25 OR reg, regX reg=reg OR mem[regX] 26 OR reg, [address] reg=reg OR mem[regX] 26 OR reg, constant reg=reg OR constant 27 OR reg, constant reg=reg OR constant 28 XOR reg, [regX] reg=reg XOR mem[regX] 29 XOR reg, [regX] reg=reg XOR mem[address] 31 XOR reg, [address] reg=reg XOR constant 32 SHL reg, regX reg=reg @mem[regX] 33 SHL reg, regX reg=reg @mem[regX] 34 SHL reg, constant reg=reg @mem[regX] 35 SHL reg, constant reg=reg p=regX 36 SHR reg, regX reg=reg p=regX 37 SHR reg, [regX] reg=reg p=regX 38 SHR reg, [regX] reg=regx] 39 SHR reg, [regX] reg=regx]			
21			
22 AND reg, constant reg=reg AND constant 24 OR reg, cregX reg=reg OR regX 25 OR reg, [regX] reg=reg OR mem[regX] 26 OR reg, [address] reg=reg OR mem[regX] 27 OR reg, constant reg=reg OR constant 28 XOR reg, regX reg=reg XOR regX 29 XOR reg, [regX] reg=reg XOR mem[regX] 30 XOR reg, [regX] reg=reg XOR mem[regX] 31 XOR reg, constant reg=reg XOR mem[regX] 32 SHL reg, constant reg=reg XOR mem[regX] 33 SHL reg, (regX] reg=reg XOR mem[regX] 34 SHL reg, (regX] reg=reg X=regX 35 SHL reg, (constant reg=reg X=regX 36 SHR reg, [regX] reg=reg x=reg X=rem[regX] 37 SHR reg, [regX] reg=reg y=reg X=reg 38 SHR reg, [regX] reg=reg y=reg y=reg 40 CMP reg, regX if reg==mem[regX] 39 SHR reg, [regX] if reg==mem[regX] 41 CMP reg, (regX) if r		AND reg. [regX]	reg=reg AND mem[regX]
23 AND reg, constant reg=reg AND constant 24 OR reg, regX reg=reg OR regX 25 OR reg, [address] reg=reg OR mem[regX] 26 OR reg, [address] reg=reg OR constant 28 XOR reg, constant reg=reg XOR regX 29 XOR reg, [regX] reg=reg XOR mem[regX] 30 XOR reg, [address] reg=reg XOR constant 31 XOR reg, constant reg=reg XOR constant 32 SHL reg, regX reg=reg≪mem[regX] 33 SHL reg, [regX] reg=reg≪mem[regX] 34 SHL reg, [address] reg=reg≪mem[address] 35 SHR reg, regX reg=reg mem[address] 36 SHR reg, [regX] reg=reg mem[address] 37 SHR reg, [regX] reg=reg mem[address] 38 SHR reg, [regX] reg=reg mem[address] 39 SHR reg, [regX] if reg==mem[regX], Z=1 40 CMP reg, [address] if reg==regX, Z=1 41 CMP reg, [address] if reg=regX, Z=1 42 CMP reg, [regX]			
24 OR reg, [regX] reg=reg OR regX 25 OR reg, [regX] reg=reg OR mem[regX] 26 OR reg, constant reg=reg OR constant 27 OR reg, constant reg=reg OR constant 28 XOR reg, [regX] reg=reg XOR mem[regX] 30 XOR reg, [regX] reg=reg XOR mem[regX] 30 XOR reg, constant reg=reg XOR constant 32 SHL reg, constant reg=reg≪mem[regX] 34 SHL reg, [regX] reg=reg≪mem[regX] 35 SHL reg, constant reg=reg≪mem[regX] 36 SHR reg, [regX] reg=reg≫mem[regX] 37 SHR reg, [regX] reg=reg≫mem[regX] 38 SHR reg, constant reg=reg≫constant 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, regX if reg==mem[regX], Z=1 42 CMP reg, constant if reg==reg+1 43 CMP reg, constant if reg==reg+1 44 LOWP reg, regX if reg==reg+1 45 DEC reg reg=reg+1			
25 OR reg, [regX] reg=reg OR mem[regX] 26 OR reg, [address] reg=reg OR mem[address] 27 OR reg, [address] reg=reg OR constant 28 XOR reg, regX reg=reg XOR mem[regX] 30 XOR reg, [address] reg=reg XOR mem[regX] 31 XOR reg, [constant] reg=reg XOR constant 32 SHL reg, regX reg=reg≪mem[regX] 33 SHL reg, regX reg=reg≪mem[regX] 34 SHL reg, constant reg=reg mem[regX] 35 SHR reg, regX reg=reg mem[regX] 36 SHR reg, regX reg=reg mem[regX] 37 SHR reg, [regX] reg=reg mem[regX] 38 SHR reg, [constant] reg=reg mem[regX] 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, constant reg=reg = mem[regX], Z=1 42 CMP reg, constant if reg==mem[regX], Z=1 43 CMP reg, constant if reg==mem[regX], Z=1 44 INC reg reg=reg+1 45 DEC reg reg=regX	$\overline{}$		
26 OR reg, [address] reg=reg OR constant 27 OR reg, constant reg=reg XOR regX 28 XOR reg, [regX] reg=reg XOR mem[regX] 29 XOR reg, [regX] reg=reg XOR mem[regX] 30 XOR reg, [address] reg=reg XOR constant 31 XOR reg, [regX] reg=reg≪Rox constant 32 SHL reg, constant reg=reg≪mem[regX] 34 SHL reg, [address] reg=reg@mem[regX] 35 SHL reg, constant reg=reg≫mem[regX] 36 SHR reg, [regX] reg=reg≫mem[regX] 37 SHR reg, [regX] reg=reg≫mem[regX] 38 SHR reg, [address] reg=reg≫mem[regX] 39 SHR reg, [address] reg=reg=constant 40 CMP reg, [regX] if reg==mem[regX], Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[regX], Z=1 43 CMP reg, [constant if reg==mem[regX], Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg<			
27 OR reg, regX reg=reg XOR regX 29 XOR reg, regX reg=reg XOR regX 30 XOR reg, [regX] reg=reg XOR mem[regX] 30 XOR reg, [address] reg=reg XOR onstant 31 XOR reg, constant reg=reg ≪eregX 33 SHL reg, regX reg=reg≪mem[regX] 34 SHL reg, [address] reg=reg ≪mem[address] 35 SHL reg, constant reg=reg ≫regX 36 SHR reg, regX reg=reg ≫mem[regX] 38 SHR reg, [regX] reg=reg ≫mem[regX] 39 SHR reg, constant reg=reg ≫mem[regX] 40 CMP reg, constant reg=reg x, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, constant if reg==mem[regX], Z=1 43 CMP reg, constant if reg==reg+X, Z=1 44 CMP reg, constant if reg==regX, Z=1 42 CMP reg, constant if reg==regX, Z=1 43 CMP reg, constant if reg==regX, Z=1 44 INC reg reg=regX, Z=1 <	26		
28 XOR reg, [regX] reg=reg XOR mem[regX] 29 XOR reg, [regX] reg=reg XOR mem[regX] 30 XOR reg, [address] reg=reg XOR constant 31 XOR reg, constant reg=reg XOR constant 32 SHL reg, [regX] reg=reg≪mem[regX] 33 SHL reg, [regX] reg=reg mem[regX] 34 SHL reg, constant reg=reg mem[regX] 35 SHL reg, constant reg=reg mem[regX] 36 SHR reg, regX reg=reg mem[regX] 37 SHR reg, [regX] reg=reg mem[regX] 38 SHR reg, constant reg=reg mem[regX] 40 CMP reg, regX if reg==mem[regX], Z=1 41 CMP reg, regX if reg==mem[regX], Z=1 42 CMP reg, (address) if reg==mem[regX], Z=1 43 CMP reg, constant if reg==mem[regX], Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=reg-1 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address]	27		
30 XOR reg, [address] reg=reg XOR mem[address] 31 XOR reg, constant reg=reg XOR constant 32 SHL reg, regX reg=reg≪regX 33 SHL reg, [regX] reg=reg≪mem[regX] 34 SHL reg, [address] reg=reg≪mem[address] 35 SHL reg, constant reg=reg≫constant 36 SHR reg, [regX] reg=reg≫mem[regX] 38 SHR reg, [regX] reg=reg≫constant 40 CMP reg, [regX] if reg==reg, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [constant reg=reg=-constant, Z=1 43 CMP reg, constant if reg==mem[regX], Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=reg-1 46 NOT reg reg=reg-1 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51	28		
31	29	XOR reg, [regX]	reg=reg XOR mem[regX]
32 SHL reg, [regX] reg=reg≪regX 33 SHL reg, [regX] reg=reg≪mem[regX] 34 SHL reg, [address] reg=reg≪mem[address] 35 SHR reg, constant reg=reg≫regX 36 SHR reg, regX reg=reg≫mem[regX] 37 SHR reg, [address] reg=reg≫mem[regX] 38 SHR reg, constant reg=reg≫mem[address] 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[regX], Z=1 43 CMP reg, [address] if reg==mem[regX], Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg+1 46 NOT reg reg=reg-1 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 50 PUSH freg [SP]=mem[reg]; SP=SP-1 51 PUSH address [SP]=mem[address]; SP=SP-1			
33 SHL reg, [regX] reg=reg≪mem[regX] 34 SHL reg, [address] reg=reg≪mem[address] 35 SHL reg, constant reg=reg≫constant 36 SHR reg, regX reg=reg≫mem[regX] 37 SHR reg, [regX] reg=reg≫mem[regX] 38 SHR reg, [address] reg=reg≫constant 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, [address] if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH reg [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=constant; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 <th>31</th> <td>XOR reg, constant</td> <td></td>	31	XOR reg, constant	
34 SHL reg, [address] reg=reg≪constant 35 SHL reg, constant reg=reg≪constant 36 SHR reg, regX reg=reg≫regX 37 SHR reg, [regX] reg=reg≫mem[regX] 38 SHR reg, [address] reg=reg≫mem[address] 39 SHR reg, constant reg=reg≫constant 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [address] if reg==mem[regX], Z=1 41 CMP reg, [address] if reg==mem[address], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, [address] if reg==mem[address], Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=reg-1 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 50 PUSH address [SP]=mem[reg]; SP=SP-1 51 PUSH constant [SP]=mem[address]; SP=SP-1			
35 SHL reg, constant reg=reg ≪ constant 36 SHR reg, regX reg=reg ≫ regX 37 SHR reg, [regX] reg=reg ≫ mem [regX] 38 SHR reg, [address] reg=reg ≫ mem [address] 39 SHR reg, constant reg=reg ≫ constant 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [address] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[regX], Z=1 43 CMP reg, [address] if reg==mem[regX], Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg+1 46 NOT reg reg=reg+1 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg, SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[reg]; SP=SP-1 51 PUSH constant [SP]=sp=SP+1 <t< td=""><th></th><td></td><td></td></t<>			
36 SHR reg, [regX] reg=reg≫mem[regX] 37 SHR reg, [regX] reg=reg≫mem[regX] 38 SHR reg, [address] reg=reg≫constant 40 CMP reg, regX if reg=regX, Z=1 41 CMP reg, [regX] if reg=mem[regX], Z=1 41 CMP reg, [address] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=reg-1 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 50 PUSH address [SP]=mem[reg]; SP=SP-1 50 PUSH dodress [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=sp=reg; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address [SP]=constant; SP=SP-1 54 JC PC=mem[address], if carry=false		SHL reg, [address]	
37 SHR reg, [regX] reg=reg≫mem[regX] 38 SHR reg, [address] reg=reg≫mem[address] 39 SHR reg, constant reg=reg≫mem[address] 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg+1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address] 55 JNC PC=mem[address] if carry=false <td< td=""><th></th><td></td><td></td></td<>			
38 SHR reg, [address] reg=reg≫mem[address] 39 SHR reg, constant reg=reg≫constant 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH reg [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false e zero=false <tr< td=""><th></th><td></td><td></td></tr<>			
39 SHR reg, constant reg=reg≫constant 40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[reg]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address] 54 JC PC=mem[address] 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60			
40 CMP reg, regX if reg==regX, Z=1 41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 50 PUSH address [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false 60			
41 CMP reg, [regX] if reg==mem[regX], Z=1 42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address], if carry=true 54 JC PC=mem[address], if carry=false 54 JC PC=mem[address], if carry=false 54 JC PC=mem[address], if carry=false e zero=false 55 JNZ PC=mem[address], if carry=false e zero=false 56 JAE PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address],	$\overline{}$		
42 CMP reg, [address] if reg==mem[address], Z=1 43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address], if carry=true 54 JC PC=mem[address], if carry=false 54 JC PC=mem[address], if zero=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=true <th></th> <td></td> <td></td>			
43 CMP reg, constant if reg==constant, Z=1 44 INC reg reg=reg+1 45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address] 54 JC PC=mem[address] 54 JC PC=mem[address] 55 JNC PC=mem[address] 56 JZ PC=mem[address] 57 JNZ PC=mem[address] 58 JA PC=mem[address] 59 JNBE PC=mem[address] 60 JAE PC=mem[address] 61 JNB PC=mem[add			if reg==mem[address]. Z=1
44 INC reg reg=reg-1 45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if carry=false e zero=false 57 JNZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false 60 JAE PC=mem[address], if carry=true 61 JNB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if zero=false			
45 DEC reg reg=reg-1 46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address], if carry=false 54 JC PC=mem[address], if carry=false 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC	- 1		
46 NOT reg reg=not(reg) 47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if zero=true 57 JNZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=true 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=m	45		
47 CALL address [SP]=PC; SP=SP-1; PC=mem[address] 48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if zero=false 57 JNZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=false 66 JE	$\overline{}$	9	reg=not(reg)
48 PUSH reg [SP]=reg; SP=SP-1 49 PUSH [reg] [SP]=mem[reg]; SP=SP-1 50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if zero=true 56 JZ PC=mem[address], if zero=false 59 JNZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=false 66 JE PC=mem[address], if zero=false 67 JNE PC=mem[address],	47	CALL address	[SP]=PC; SP=SP-1; PC=mem[address]
50 PUSH address [SP]=mem[address]; SP=SP-1 51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if zero=true 57 JNZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=false 66 JE PC=mem[address], if zero=false 67 JNE PC=mem[address], if zero=false	48		
51 PUSH constant [SP]=constant; SP=SP-1 52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if zero=true 56 JZ PC=mem[address], if zero=false 57 JNZ PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=false 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
52 POP reg reg=[SP]; SP=SP+1 53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if zero=true 57 JNZ PC=mem[address], if zero=false 58 JA PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
53 JMP address PC=mem[address] 54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if zero=true 57 JNZ PC=mem[address], if zero=false 58 JA PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=false 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
54 JC PC=mem[address], if carry=true 55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if zero=true 57 JNZ PC=mem[address], if zero=false 58 JA PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=false 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1	$\overline{}$	POP reg	
55 JNC PC=mem[address], if carry=false 56 JZ PC=mem[address], if zero=true 57 JNZ PC=mem[address], if zero=false 58 JA PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=true 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
56 JZ PC=mem[address], if zero=true 57 JNZ PC=mem[address], if zero=false 58 JA PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=false 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
57 JNZ PC=mem[address], if zero=false 58 JA PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true ou zero=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=false 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
58 JA PC=mem[address], if carry=false e zero=false 59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
59 JNBE PC=mem[address], if carry=false e zero=false 60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
60 JAE PC=mem[address], if carry=false 61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if carry=true ou zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1	$\overline{}$		PC—mem[address], if carry—false e zero—false
61 JNB PC=mem[address], if carry=false 62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if carry=true ou zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			PC=mem[address], it carry—talse e zero=talse
62 JB PC=mem[address], if carry=true 63 JNAE PC=mem[address], if carry=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if carry=true ou zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
63 JNAE PC=mem[address], if carry=true 64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if carry=true ou zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1	$\overline{}$		
64 JBE PC=mem[address], if carry=true ou zero=true 65 JNA PC=mem[address], if carry=true ou zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
65 JNA PC=mem[address], if carry=true ou zero=true 66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
66 JE PC=mem[address], if zero=true 67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
67 JNE PC=mem[address], if zero=false 68 RET PC=[SP]; SP=SP+1			
68 RET PC=[SP]; SP=SP+1		JNE	PC=mem[address], if zero=false
69 HLT parar o processador	68		PC=[SP]; SP=SP+1
	69	HLT	

Table 1: Conjunto de instruções do processador.