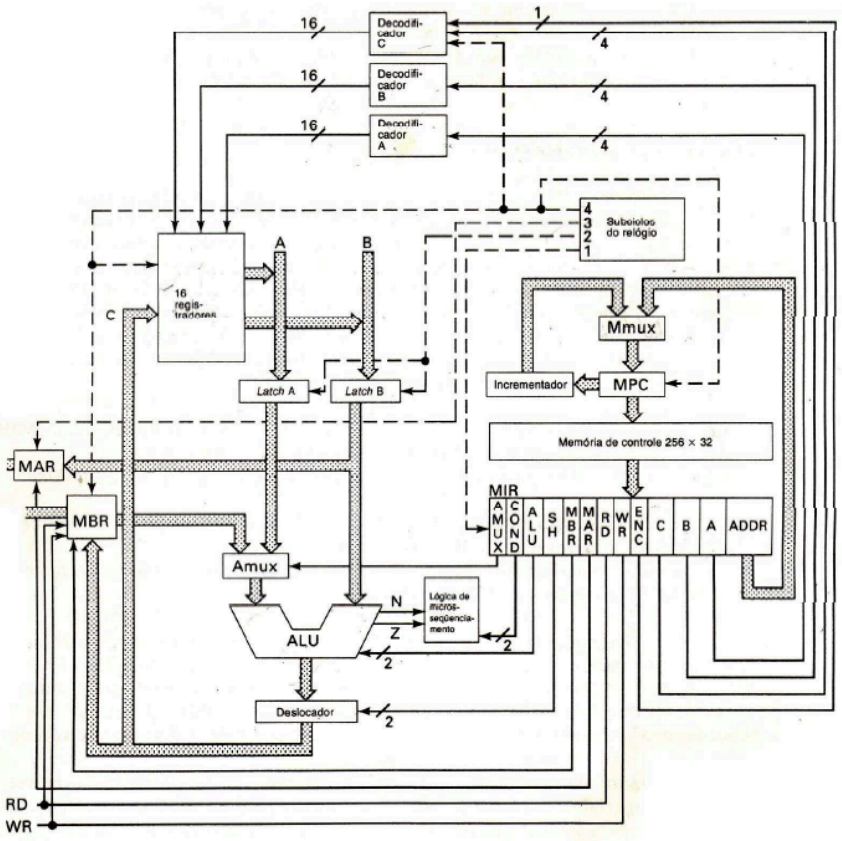
Microinstruções MAC-I

Encoding	Mnemonic	Instruction	Semantics
0000xxxxxxxxxxx	LODD	Load Direct	ac := m[x]
0001xxxxxxxxxxx	STOD	Store Direct	m[x] := ac
0010xxxxxxxxxxx	ADDD	Add Direct	ac := ac + m[x]
0011xxxxxxxxxxx	SUBD	Subtract Direct	ac := ac - m[x]
0100xxxxxxxxxxx	JPOS	Jump on non-negative	if ac ≥ 0, pc := x
0101xxxxxxxxxxx	JZER	Jump on zero	if ac = 0, pc := x
0110xxxxxxxxxxx	JUMP	Jump	pc := x
0111xxxxxxxxxxx	LOCO	Load Constant	ac := x ((0 ≤ x ≤ 4095)
1000xxxxxxxxxxx	LODL	Load Local	ac := m[sp + x]
1001xxxxxxxxxxx	STOL	Store Local	m[sp + x] := ac
1010xxxxxxxxxxx	ADDL	Add Local	ac := ac + m[sp + x]
1011xxxxxxxxxxx	SUBL	Subtract Local	ac := ac - m[sp + x]
1100xxxxxxxxxxx	JNEG	Jump on negative	if ac < 0, pc := x
1101xxxxxxxxxxx	JNZE	Jump unless zero	if ac ≠ 0, pc := x
1110xxxxxxxxxxx	CALL	Call	sp := sp - 1; m[sp] := pc; pc := x
1111000000000000	PSHI	Push Indirect	sp := sp - 1; m[sp] := m[ac]
1111001000000000	POPI	Pop Indirect	m[ac] := m[sp]; sp := sp + 1
1111010000000000	PUSH	Push	sp := sp - 1; m[sp] := ac
1111011000000000	POP	Pop	ac := m[sp]; sp := sp + 1
1111100000000000	RETN	Return	pc := m[sp]; sp := sp + 1
1111101000000000	SWAP	Swap AC , SP	tmp:= ac;ac :=: sp;sp=tmp
11111100yyyyyyyy	INSP	Increment SP	sp := sp + y (0 ≤ y ≤ 255)
11111110yyyyyyyy	DESP	Decrement SP	sp := sp - y (0 ≤ y ≤ 255)
xxxxxxxxxxxx is a 12-bit machine address; in column 4, it is called x. yyyyyyyy is a 8-bit constant; in column 4 it is called y.			

Microprograma MIC-1

```
0: mar := pc; rd;                          {loop principal}
1: pc := pc + 1; rd;                       {incremente pc}
2: ir := mbr; if n then goto 28;           {salva, dec. mbr}
3: tir := lshift (ir + ir); if n then goto 19;
4: tir := lshif t(tir); if n then goto 11; {000x ou 001x?}
5: alu := tir; if n then goto 9;           {0000 ou 0001?}

6: mar := ir; rd;                          {0000 = LODD}
7: rd;
8: ac := mbr; goto 0;

9: mar := ir; mbr := ac; wr;               {0001 = STOD}
10: wr; goto 0;

11: alu := tir; if n then goto 15;         {0010 ou 0011?}
12: mar := ir; rd;                         {0010 = ADDD}
13: rd;
14: ac := mbr + ac; goto 0;

15: mar := ir; rd;                         {0011 = SUBD}
16: ac := ac + 1; rd;                      { x - y = x + 1 + not y}
17: a := inv (mbr);
18: ac := ac + a; goto 0;

19: tir := lshift (ir); if n then goto 25; {010x ou 011x?}
20: alu := tir; if n then goto 23;         {0100 ou 0101?}

21: alu := ac; if n then goto 0;           {0100 = JPOS}
22: pc := band (ir, #mask); goto 0;        {faça o desvio}

23: alu := ac; if z then goto 22;          {0101 = JZER}
24: goto 0;                                {desvio falhou}
25: alu := ir; if n then goto 27;          {0110 ou 0111?}
26: pc := band(ir, amask); goto 0;         {0110 = JUMP}
27: ac := band(ir, amask); goto 0;         {0111 = LOCO}
28: tir := lshift(ir + ir); if n then goto 40;{10xx ou 11xx?}
29: tir := lshift(ir); if n then goto 40;  {100x ou 101x?}
30: alu := ir; if n then goto 33;          {1000 ou 1001?}
31: a := ir + sp;                          {1000 = LODL}
32: mar := a; rd; goto 7;

33: a := ir + sp;                          {1001 = STOL}
34: mar := a; mbr := ac; wr; goto 10;

35: alu := ir; if n then goto 38;          {1010 ou 1011?}

36: a := ir + sp;                          {1010 = ADDL}
37: mar := a; rd; goto 13;

38: a := ir + sp;                          {1011 = SUBL}
39: mar := a; rd; goto 16;

40: tir := lshift (tir); if n then goto 46; {110x ou 111x?}
41: alu := tir; if n then goto 44;         {1100 ou 1101?}

42: alu := ac; if n then goto 22;          {1100 = JNEG}
43: goto 0;
```

```
44: alu := ac; if z then goto 0;           {1101 = JNZE}
45: pc := band (ir, amask); then goto 0;

46: tir := lshift (tir); if n then goto 50;
47: sp := sp + (−1);                       {1110 = CALL}
48: mar := sp; mbr := pc; wr;
49: pc := band (ir, amask); wr; goto 0;

50: tir := lshift (tir); if n then goto 65; {1111, examine end.}
51: tir := lshift (tir); if n then goto 59;
52: alu := tir; if n then goto 56;

53: mar := ac; rd;                         {1111000 = PSHI}
54: sp := sp + (−1); rd;
55: mar := sp; wr; goto 10;

56: mar := sp; sp := sp + 1; rd;           {1111001 = POPI}
57: rd;
58: mar := ac; wr; goto 10;

59: alu := tir; if n then goto 62;
60: sp := sp + (−1);                       {1111010 = PUSH}
61: mar := sp; mbr := ac; wr; goto 10;

62: mar := sp; sp := sp + 1; rd;           {1111011 = POP}
63: rd;
64: ac := mbr; goto 0;

65: tir := lshift (tir); if n then goto 73;
66: alu := tir; if n then goto 70;
67: mar := sp; sp := sp + 1; rd;           {1111100 = RETN}
68: rd;
69: pc := mbr; goto 0;

70: a := ac;                               {1111101 = SWAP}
71: ac := sp;
72: sp := a; goto 0;

73: alu := tir; if n then goto 76;
74: a := bund (ir, smask);                 {1111110 = INSP}
75: sp := sp + a; goto 0;

76: a := band (ir, smask);                 {1111111 = DESP}
77: a := inv (a);
78: a := a + 1; goto 75;
```



Via de Dados - MIC-1