mpc-1 Microprogram to fetch, decode, and execute MAC-2 instructions

Adr: Microinstruction	Comment	Adr: Microinstruction	Comment
0: mar:=pc; rd;	fetch instr	50: tir:=lshift(tir); if n then goto 65;	decode ir_{11}
1: pc:=pc + 1; rd;	increment pc	51: tir:=lshift(tir); if n then goto 59;	decode ir_{10}
2: ir:=mbr; if n then goto 28;	decode ir_{15}	52: alu:=tir; if n then goto 56;	decode ir_9
3: tir:=lshift(ir + ir); if n then goto 19;	decode ir_{14}	53: mar:=ac; rd;	1111-0000 = **PSHI**
4: tir:=lshift(tir); if n then goto 11;	decode ir_{13}	54: sp:=sp + (-1); rd;	
5: alu:=tir; if n then goto 9;	decode ir_{12}	55: mar:=sp; wr; goto 10;	
6: mar:=ir; rd;	0000 = **LODD**	56: mar:=sp; sp:=sp + 1; rd;	1111-0010 = **POPI**
7: rd;		57: rd;	
8: ac:=mbr; goto 0;		58: mar:=ac; wr; goto 10;	
9: mar:=ir; mbr:=ac; wr;	0001 = **STOD**	59: alu:=tir; if n then goto 62;	decode ir_9
10: wr; goto 0;		60: sp:=sp + (-1);	1111-0100 = **PUSH**
11: alu:=tir; if n then goto 15;	decode ir_{12}	61: mar:=sp; mbr:=ac; wr; goto 10;	
12: mar:=ir; rd;	0010 = **ADDD**	62: mar:=sp; sp:=sp + 1; rd;	1111-0110 = **POP**
13: rd;		63: rd;	
14: ac:=mbr + ac; goto 0;		64: ac:=mbr; goto 0;	
15: mar:=ir; rd;	0011 = **SUBD**	65: tir:=lshift(tir); if n then goto 73;	decode ir_{10}
16: ac:=ac + 1; rd;		66: alu:=tir; if n then goto 70;	decode ir_9
17: a:=inv(mbr);		67: mar:=sp; sp:=sp + 1; rd;	1111-1000 = **RETN**
18: ac:=ac + a; goto 0;		68: rd;	
19: tir:=lshift(tir); if n then goto 25;	decode ir_{13}	69: pc:=mbr; goto 0;	
20: alu:=tir; if n then goto 23;	decode ir_{12}	70: a:=ac;	1111-1010 = **SWAP**
21: alu:=ac; if n then goto 0;	0100 = **JPOS**	71: ac:=sp;	
22: pc:=band(ir,amask); goto 0;	perform jump	72: sp:=a; goto 0;	
23: alu:=ac; if z then goto 22;	0101 = **JZER**	73: tir:=lshift(tir); if n then goto 76;	decode ir_9
24: goto 0;	else don't jump	74: a:=band(ir,smask);	1111-1100 = **INSP**
25: alu:=tir; if n then goto 27;	decode ir_{12}	75: sp:=sp + a; goto 0;	
26: pc:=band(ir,amask); goto 0;	0110 = **JUMP**	76: tir:=lshift(tir); if n then goto 80;	decode ir_8
27: ac:=band(ir,amask); goto 0;	0111 = **LOCO**	77: a:=band(ir,smask);	1111-1110 = **DESP**
28: tir:=lshift(ir + ir); if n then goto 40;	decode ir_{14}	78: a:=inv(a);	
29: tir:=lshift(tir); if n then goto 35;	decode ir_{13}	79: a:=a + 1; goto 75;	
30: alu:=tir; if n then goto 33;	decode ir_{12}	80: tir:=lshift(tir); if n then goto 84;	decode ir_7
31: a:=ir + sp;	1000 = **LODL**	81: alu:=tir; if n then goto 83;	decode ir_6
32: mar:=a; rd; goto 7;		82: ac:=f; goto 0;	1111-1111-00 = **FTAC**
33: a:=ir + sp;	1001 = **STOL**	83: f:=ac; goto 0;	1111-1111-01 = **ACTF**
34: mar:=a; mbr:=ac; wr; goto 10;		84: tir:=lshift(tir); if n then goto 97;	decode ir_6
35: alu:=tir; if n then goto 38;	decode ir_{12}	85: a:=rshift(smask);	1111-1111-10 = **RACR**
36: a:=ir + sp;	1010 = **ADDL**	86: a:=rshift(a);	
37: mar:=a; rd; goto 13;		87: a:=rshift(a);	
38: a:=ir + sp;	1011 = **SUBL**	88: a:=rshift(a);	
39: mar:=a; rd; goto 16 ;		89: a:=band(ir,a); if z then goto 0;	
40: tir:=lshift(tir); if n then goto 46;	decode ir_{13}	90: tir:=0 + 1;	
41: alu:=tir; if n then goto 44;	decode ir_{12}	91: alu:=band(ac,tir); if z then goto 96;	
42: alu:=ac; if n then goto 22;	1100 = **JNEG**	92: ac:=rshift(inv(ac));	
43: goto 0;		93: ac:=inv(ac);	
44: alu:=ac; if z then goto 0;	1101 = **JNZE**	94: a:=a + (-1); if z then goto 0;	
45: pc:=band(ir,amask); goto 0;		95: goto 91;	
46: tir:=lshift(tir); if n then goto 50;	decode ir_{12}	96: ac:=rshift(ac); goto 94;	
47: sp:=sp + (-1);	1110 = **CALL**	97: alu:=tir; if n then goto 99;	decode ir_5
48: mar:=sp; mbr:=pc; wr;		98: ac:=band(ac,f); goto 0;	1111-1111-110 = **ANDF**
49: pc:=band(ir,amask); wr; goto 0;		99: halt; goto 99;	1111-1111-111 = **HALT**

The execution cycle for each decoded MAC-2 instruction begins at the control store address whose line is labeled with a comment showing the assembly language mnemonic for the corresponding instruction (capitalized for emphasis). "Adr:" is the control store address. The instruction fetch cycle begins at control store address zero.