DISASSEMBLER INSTRUCTIONS

VERSIONS DIS-1D and DIS-1P

The disassembler is memory mapped at 3000 and will not run at any other location. If you are using the issue 12 printer interface, the disassembler ROM can be fitted into the on-board ROM socket. You will have to shift the CS line that goes to the 74LS138 output to the second top output. To check if the disassembler is fitted properly, address 3000. The bytes there should be 21 C0 08. Without the printer interface, the disassembler can be stacked as described in issue 14.

The disassembler is designed to work with the JMON ROM, but if you are not using JMON, the disassembler can be used if you provide the start address at 0898 and the finish address at 089A (finishing address not required for the LCD version) The display buffer is located at 08C0. Make certain nothing valuable is there.

For those of you using JMON, the disassembler is entered at 37B0. You will be prompted by the JMON PERIMETER HANDLER to enter the address from where to start disassembling from. Enter the address and hit the "+" key. You are now asked for the end address. Enter this and hit "GO"

If using the disassembler without JMON, enter the start and end address at 0898 and 089A. To enter the disassembler, CALL 37BE. The disassembler will now work the same as if a JMON were fitted. (*PRINTER VERSION ONLY*)

If you have the printer version (DIS-1P) and you wish to output the disassembly to the LCD, Use the routine below. It is entered at 0F00 and provides the same functions as described above if used with JMON. If you are not using JMON, then enter the routine at 0F0E after placing the start address at 0898.

Below is actual example of disassembler output after being up-loaded to a IBM fake from the TEC using the serial routine in the TEC-pack. The disassembly is passed straight into the desk-top-publisher and printed with the rest of the article. No more cut and paste!

0F00	21 3B 0F	LD HL,0F3B
0F03	11 80 08	LD DE,0880
0F06	01 0A 00	LD BC,000A
0F09	ED B0	LDIR
0F0B	C3 44 00	JP 0044
0F0E	CD 00 30	CALL 3000
0F11	F7	RST 30
0F12	3E 01	LD A,01
0F14	D3 04	OUT 04,A
0F16	21 C0 08	LD HL,08C0
0F19	06 10	LD B,10
0F1B	F7	RST 30
0F1C	7E	LD A,(HL)
0F1D	D3 84	OUT 84,A
0F1F	23	INC HL
0F20	10 F9	DJNZ 0F1B
0F22	F7	RST 30
0F23	3E C3	LD A,C3
0F25	D3 04	OUT 04,A
0F27	06 10	LD B,10
0F29	21 D2 08	LD HL,08D2
0F2C	F7	RST 30
0F2D	7E	LD A,(HL)
0F2E	D3 84	OUT 84,A
0F30	23	INC HL
0F31	10 F9	DJNZ 0F2C
0F33	76	HALT
0F34	18 D8	JR 0F0E
0F36	DB 04	IN A,04
0F38	CB 7F	BIT 7,A
0F3A	20 FB	JR NZ 0F37
0F3C	C9	RET

0F3D 47 0F 99 08 00 01 0E 0F
0F47 04 A7 04 C7

The routine below is for those who have the LCD version of the disassembler and wish to print out a disassembly on the VZ200 printer.

If using JMON, then enter the routine at 0F00. Without JMON, enter the start and end addresses before CALLing at 0F0E.

0F00	21 40 0F	LD HL,0F40
0F03	11 80 08	LD DE,0880
0F06	01 0A 00	LD BC,000A
0F09	ED B0	LDIR
0F0B	C3 44 00	JP 0044
0F0E	CD 00 30	CALL 3000
0F11	2A A0 08	LD HL,(08A0)
0F14	36 FF	LD (HL),FF
0F16	21 C0 08	LD HL,08C0
0F19	7E	LD A,(HL)
0F1A	FE FF	CP FF
0F1C	28 05	JR Z 0F23
0F1E	D3 06	OUT 06,A
0F20	23	INC HL
0F21	18 F6	JR 0F19
0F23	3E 0A	LD A,0A
0F25	D3 06	OUT 06,A
0F27	3E 0D	LD A,0D
0F29	D3 06	OUT 06,A
0F2B	2A 98 08	LD HL,(0898)
0F2E	ED 4B 9A 08	LD BC,(089A)
0F32	03	INC BC
0F33	B7	OR A
0F34	ED 42	SBC HL,BC
0F36	38 D6	JR C 0F0E
0F38	C7	RST 00

0F40 FF FF 4A 0F 99 08 00 01
0F48 0E 0F 04 A7 04 C7

FOR ADVANCED USERS

The output of the disassembler is an ASCII string located at 08C0. The end of the string is easy to find. It is pointed to by the address located at 08A0. This address points to the next available display buffer address, so if you wish to place an end marker there for your own output routines, use the following:

2A A0 08	LD HL,(08A0)
36 FF	LD (HL),FF

For your own custom master routines, the disassembler can be called as a sub-routine at address 3000.

Two differences occur between the standard Zilog syntax and the disassembler output. They are: Disassembler leaves of brackets from IN A,(00) and OUT (01),A; and omits a comer from conditional relative jumps.

Possibly, they may be corrected in a future up-date.