PDP-8/X System Reference Manual

dgc@spies.com (David G. Conroy)

Last Updated on Monday, August 14, 2000 at 5:29:52 PM

1. Introduction

The PDP-8/X is a new implementation the Digital Equipment Corporation PDP-8.

The PDP-8/X system is implemented by two FPGAS. The first FPGA contains the processor. The second FPGA contains the serial line interface and the IDE disk interface. The memory (which consists of a 32K word normal memory used by standard software, and a 4K word panel memory which us used only by the software which emulates the front panel) is implemented by four 32K x 8 static RAMS.

2. CPU and Memory

The PDP-8 architecture survived for twenty-five years, and during those years thirteen distinct models were implemented (the PDP-8 was introduced in 1965, and the DECMATE-III/III+ were discontinued in 1990). These models differed from each other in minor ways. The PDP-8/X most closely emulates the PDP-8/I; the PDP-8/I was chosen because it eliminates all of the restrictions imposed on the operate instruction in earlier models, and does not implement the more complex instruction set of later models.

The PDP-8 uses big-endian bit numbering (that is, bit 0 is the most significant). This specification does the same. The PDP-8/X hardware, however, uses little-endian bit numbering (that is, bit 0 is the least significant bit) so that the processor, the static RAMS, the EPROM, and the IDE disk can number the bits the same way. Think of this as preserving one of the less-than-wonderful traditions of DEC designs.

2.1. CPU State

PC[00..11] — The PC register holds the address of the next instruction to be executed.

AC[00..11] — The AC register is the primary arithmetic and logical register. It can be loaded and stored, and serves both as operand and result for logical AND and ADD operations, The AC register can also be cleared, complemented, incremented, tested, and rotated left or right with the LINK flag.

LINK — The LINK flag is a logical extension of the AC register. It is complemented if an addition operation involving the AC register results in a carry. The LINK flag can also be cleared, complemented, tested, and rotated left or right with the AC register.

IF[00..02] The IF (“instruction field”) register holds the three most significant bits of the instruction field address.

IB[00..02] The IB (“instruction buffer”) register holds the three most significant bits of a new instruction field address. It is explicitly loaded by a program which wants to change the IF register, and implicitly loaded into the IF register by instructions which change the flow of control.

DF[00..02] The DF (“data field”) register holds the three most significant bits of the data field address.

SF[00..05] The SF (“save field”) register saves the contents of the DF register and the IF register during an interrupt. The contents of the DF register and the IF register are implicitly loaded into the SF register at the beginning of an interrupt, and the contents of the SF register can be explicitly loaded into the DF register and the IB register at the end of the interrupt service routine.

IE The IE (“interrupt enable”) flag is 1 if the CPU is running with interrupts enabled, and is 0 if it is running with interrupts disabled.

II The II (“interrupt inhibit”) flag is 1 if the CPU is running with interrupts inhibited, independent of the state of the IE flag, because the program is in the middle of an update to the IF register. The II flag is set to 1 when the IB register is loaded, and set to 0 when the value in the IB register is transferred into the IF register.

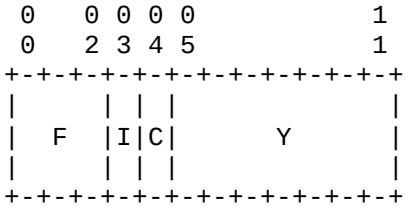
SW[00..11] The SW (“switch”) register emulates the switches normally on the front panel of a PDP-8. The console program writes this register with an I/O instruction, and any program can read this register with an operate instruction.

IM The IM (“instruction mode”) flag is 1 when the CPU is running, and is 0 when the CPU is halted. In both states the CPU is actually executing instructions, but when IM is 0 the CPU is running the console program.

DM The DM (“data mode”) flag determines if data field references should use the normal memory fields or the panel memory fields. Normally it is forced to contain the same value as the IM flag, but the control program can arrange that it contains a 1 (which specifies normal memory) for the execution of a single instruction, so that the console program can read and write normal memory upon user commands.

Very little of the PDP-8/X state is initialized by reset; the 4K word EPROM is copied into the lowest 4K locations of the panel memory, the PC is set to 0, and the IM flag is set to 0. The front panel program, written in ordinary PDP-8 code, finishes the initialization of the entire system.

2.2. Memory Reference Instructions



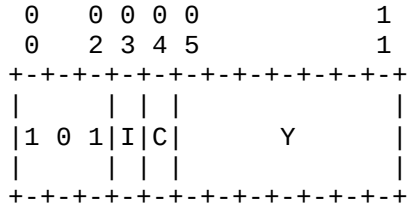
All memory reference instructions compute their effective address the same way. If the C field is 0 then the address is said to be in page zero, and the Y field specifies an address in the lowest 128 words. If the C field is 1 then the address is said to be in the current page, and the Y field specifies an address in the same 128 word page as the instruction (note that is not the page from the PC, which may have incremented to the first address of the next page). Then, if the I field is 1, a single level indirection is performed.

If an indirect word is fetched from locations 0010 through 0017 in any memory field (the “auto index” locations) then that indirect word is incremented (and rewritten to memory) before it is used.

The above algorithm generates the twelve-bit address which specifies a location within a 4K word memory field. The field is specified by the contents IF and DF registers (except when the processor is executing the front panel program, which will be explained later). The field is specified by the contents of the IF register unless the memory reference is the operand reference of an indirectly addressed AND, TAD, ISZ, or DCA instruction, when it is specified by the contents of the DF register.

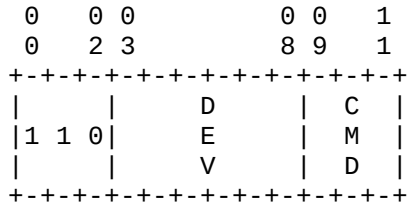
000	AND	The memory location specified by the effective address is combined with the AC register with a logical AND operation.
001	TAD	The memory location specified by the effective address is combined with the AC register with a twos-complement add. If there is a carry out of the most significant bit of the AC register the LINK flag is complemented.
010	ISZ	The memory location specified by the effective address is incremented, and if the result of the increment is 0, then the PC register is incremented, skipping the next instruction.
011	DCA	The AC register is stored into the memory location described by the effective address, and the AC register is cleared.
100	JMS	The IB register (which contains a potential update to the IF register) is loaded into the IF register, and the II flag is cleared (except when the processor is executing the front panel program, which will be explained later). Then the PC register, which has already been incremented to point to the instruction after the JMS instruction, is stored into the memory location described by the effective address, and the PC register is loaded with the effective address plus 1.

2.3. JMP Instruction



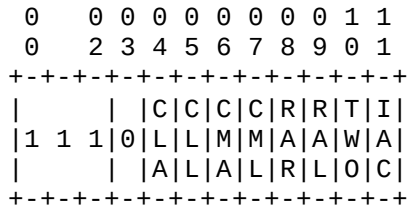
The JMP instruction evaluates its effective address in the same way as the memory reference instructions. Then the IB register (which contains a potential update to the IF register) is loaded into the IF register, and the II flag is cleared (except when the processor is executing the front panel program, which will be explained later). Then the effective address is loaded into the PC.

2.4. IOT Instruction



The IOT instruction is used to transfer data in an out of the processor via the AC register. By convention bits [03..08] specify the I/O device, and bits [09..11] specify the I/O command. All of the standard I/O devices comply with this convention, although the actual interpretation of the IOT instruction is controlled by the I/O device, so a custom I/O device could, in theory, violate these conventions.

2.5. Group 1 Operate Instruction



The group 1 operate instruction can clear, complement, increment, and rotate the AC register and the LINK flag.

The group 1 operate instruction is micro-coded, and any combination of micro-coded functions can be specified. The micro-coded functions are executed in a definite order; first the clears (CLA, CLL), then the complements (CMA, CML), then the increment (IAC), then the rotates (RAL, RAR, RTL, RTR).

[04] CLA The AC register is cleared.

[05] CLL The LINK flag is cleared.

[06] CMA The AC register is (ones) complemented.

[07] CML The LINK flag is complemented.

[08] RAR The AC register and LINK flag are rotated right by one or two bits, depending on the value of the TWO bit.

[09] RAL The AC register and LINK flag are rotated left by one or two bits, depending on the value of the TWO bit.

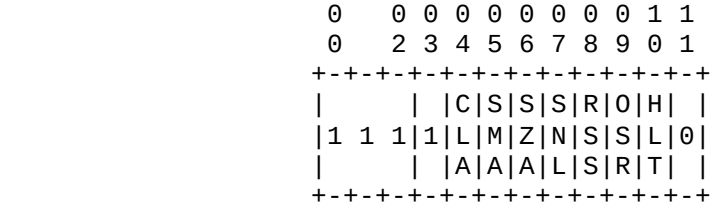
[10] TWO If a rotate is specified by bit [08] or bit [09], then that rotate is by two bits rather than by one bit.

[11] IAC The AC register is incremented. If there is a carry out of the most significant bit of the AC register the LINK flag is complemented.

Three of the eight possible combinations of bits [08..10] are undefined. Two of these (110, 111) try and rotate the AC register and LINK flag left and right at the same time; this rotate results in the logical OR of both rotations. The third (001) attempts to do no rotate by two bits; this rotate does nothing.

I need to look very carefully at the engineering drawings of the PDP-8/I and see if this is the correct interpretation of codes 110 and 111; there are a lot of low-true signals in the data path, and that might cause the result to actually be the logical AND of both rotations.

2.6. Group 2 Operate Instruction



The group 2 operate instruction can test the contents of the AC register and/or the LINK flag by conditionally skipping the next instruction. It can also clear the accumulator, read the switch register, and explicitly halt the processor (which forces control back into the console program).

The group 2 operate instruction is micro-coded, and any combination of micro-coded functions can be specified. The micro-coded functions are executed in a definite order; first the skips (SMA, SZA, SNL, RSS), then the clear (CLA), then the switch register read (OSR), and finally the halt (HLT).

[04] CLA The AC register is cleared.

[05] SMA The test condition is true if the AC register is less than zero (that is, if AC[00] is 1).

[06] SZA The test condition is true if the AC register is 0.

[07] SNL The test condition is true if the LINK flag is 1.

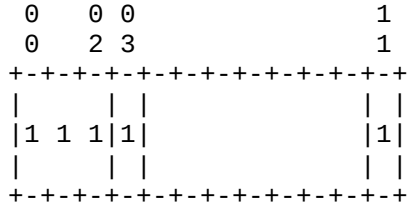
[08] RSS If this bit is 0 then the processor skips if any of the test conditions are true. If this bit is 1 then the processor

skips if none of the test conditions are true. Another way to think about this bit is that it reverses the test conditions, and the skip happens if all of the (reversed) test conditions are true.

[09] OSR The AC register is replaced with the bit-by-bit inclusive-or of the AC register and the SW register.

[10] HLT The processor halts. A halt interrupt is generated. Control passes to the front panel program.

2.7. Group 3 Operate Instruction



The group 3 operate instruction is used to control the extended arithmetic element (EAE). The PDP-8/X does not implement the EAE, so all group 3 instructions are no-operation instructions. This is the correct interpretation for a system which wants to be compatible with the PDP-8/I (the original PDP-8 honored the CLA bit of a group 3 operate instruction, even if the EAE was not installed, and the PDP-8/E always implements the MQ register and the group 3 operate instructions which move its contents to and from the AC register, even if the EAE is not installed).

2.8. Operating Modes

The PDP-8/X does not implement the traditional switches and lights console of the PDP-8/I. Instead, it implements a console program, which performs all of the functions normally provided by the switches and lights console (and more), driven by commands from the console terminal.

Implementing a console program is tricky on the PDP-8 because the architecture has considerable “off to the side” state which is difficult to save and restore. In addition, there is really no place to hide the console program in the physical address space; PDP-8 software does not believe that any part of the 32K word memory is reserved for a console program. The PDP-8/X solves this problem in a way which is similar in spirit, but different in detail, to the INTERSIL 6100 and 6120 processor chips used in the VT78 and DECMATE products.

The PDP-8/X operates in one of two modes, controlled by the IM flag. If the IM flag is 0 then the CPU is architecturally halted, and the console program is running. If the IM flag is 1 then the CPU is architecturally running, and a user program is running. In both cases the processor is executing normal PDP-8 instructions.

When the IM flag is 0 all instruction field and data field references, with one exception (which will be described later), are diverted into a special 4K word memory dedicated to the console program; it does not matter what values are in the IF, IB, and/or DF registers. The one exception involves data field references. The console program can execute a special IOT instruction which

makes any data field reference made by the next instruction operate normally; it accesses the normal memory field specified by the DF register. This lets the console program examine and deposit normal memory when necessary.

When the IM flag is 0 interrupts and halts are blocked (even if the state of the IE and II flags would allow interrupts). Furthermore, the implicit changes to the system state which normally happen on JMP and JMS instructions (the contents of the IB register being loaded into the IF register, and the II flag being set to 0) are blocked as well (although a special IOT instruction can be used to explicitly perform this operation, which is needed to completely update the state of the processor). The idea is that when then console program is running most of the “off to the side” state can be left sitting in the CPU; the only things which need to be saved are the AC register, the LINK flag, and the DF register.

A special IOT instruction is used to set the IM flag to 1, returning the CPU to the architecturally running state. There is a one instruction delay so that the final JMP is fetched from the special control panel memory. It is not possible to take an interrupt between the last instruction of the control panel program and the first instruction of the normal program, so that if the CPU is halted immediately after the execution of an ION, the one instruction grace period is (effectively) preserved.

2.9. Interrupts

A normal interrupt is generated at the end of any instruction if the CPU is architecturally running (the IM flag is 1), interrupts are enabled (the IE flag is 1), interrupts are not inhibited (the II flag is 0), and some device is making an interrupt request. If all of these conditions are true the CPU copies the current values of the IF and DF registers into the SF register, sets the IB register to 0, sets the IF register to 0, sets the DF register to 0, sets the IE flag to 0, stores the value of the PC in location 0000 of field 0 of normal memory, and sets the PC to 0001. The interrupt handler begins executing from location 0001 in normal memory, and future interrupts are disabled because the IE flag is 0.

A halt interrupt is generated at the end of any instruction if the CPU is architecturally running (the IM flag is 1), and there is a halt request (either the last instruction executed was a group 2 operate with the HLT bit set, or the external halt request signal is asserted). If both of these conditions are true the CPU sets the IM flag to 0, sets the DM flag to 0, stores the value of the PC in location 0000 of field 0 of panel memory, and sets the PC to 0001. The halt handler begins executing from location 0001 in panel memory, and future halts are disabled because the IM flag is 0.

If both a normal interrupt and a halt interrupt are being requested at the end of any instruction a halt interrupt is taken.

2.10. CPU Control I/O Instructions

The PDP-8/X CPU uses IOT instructions with device code 00 to control the interrupt logic, and, when architecturally halted, to control the front panel emulation logic. The nonstandard IOT instructions do nothing in normal mode.

6000	PWS	The contents of the AC register are loaded into the SW register. This instruction performs no operation if the IM flag is 1.
6001	ION	The IE flag is set to 1, enabling interrupts, after a one instruction delay.
6002	IOF	The IE flag is set to 0, disabling interrupts.
6003	PSI	Skip if the IE flag is 1. This instruction performs no operation if the IM flag is 1.
6004	PAI	The contents of the IB register are loaded into the IF register, and the II flag is set to 0, just as if a JMP or JMS was executed when the IM flag is 1. This instruction performs no operation if the IM flag is 1.
6005	PAS	The contents of the DF register are loaded into bits [03..05] of the SF register, and the contents of the IF register are loaded into bits [00..02] of the SF register, just as they would be loaded on an interrupt. This instruction performs no operation of the IM flag is 1.
6006	PDX	The DM flag is set to 1 for the execution of the next instruction. This instruction executes normally if IM is 1, but since DM is 1 anytime IM is 1, it effectively performs no operation.
6007	PEX	The IM flag is set to 1, switching the CPU into normal mode, after a one instruction delay. This instruction executes notmally if IM is 1, but effectively performs no operation.

2.11. Extended Memory I/O Instructions

The PDP-8/X CPU uses IOT instructions with device codes between 20 and 27 to control the extended memory control logic. The implementation is compatible with the MC8/I memory extension control, without the KT8/I timeshare control.

62x1	CDF	Bits [06..08] of the instruction are loaded into the DF register.
62x2	CIF	Bits [06..08] of the instruction are loaded into the IB register, and the II flag is set to 1. The contents of the IB register will be loaded into the IF register, and the II flag will be set to 0, when the next JMP or JMS instruction is executed. The reload happens after any instruction and/or indirect address words are read, but before the JMS writes its return address; the return address is written to the new instruction field. The II flag blocks interrupts between the CIF and the JMP/JMS (page 53 of the 1970 small computer handbook describes the effect of the II flag, even though the flag itself is not described clearly).
6214	RDF	The contents of the DF register are logically OR’ed into bits [06..08] of the AC register.

6224 RIF The contents of the IF register are logically OR'ed into bits [06..08] of the AC register.

6234 RIB The contents of the SF register are logically OR'ed into bits [06..11] of the AC register; if the AC register is initially 0, then bits [09..11] of the AC register get the value which was in the DF register at the time of the last interrupt, and bits [06..08] of the AC register get the value which was in the IF register at the time of the last interrupt.

6244 RMF Bits [03..05] of the SF register are loaded into the DF register, bits [00..02] of the SF register are loaded into the IB register, and the II flag is set to 1. The next JMP or JMS instruction completes the restore.

The extended memory control instructions are, in fact, microcoded, with bit [11] determining if the instruction should change the data field, bit [10] determining if the instruction should change the instruction field, and bit [09] determining if a special function should be executed. All combinations of these three bits function correctly, although instructions which combine a field change operation with a special function are probably not useful.

Since PDP-8/X does not implement the timeshare control, the instructions normally used by the timeshare control (6204, 6254, 6264, 6274) perform no operation.

3. External I/O

The PDP-8/X system contains a number of built in I/O devices; a serial port, an IDE disk port, and an expansion port. All of the these devices are accessed using IOT instructions. IOT instructions which specify nonexistent devices are completely ignored.

3.1. Serial I/O

The serial port interfaces with devices which comply with the RS-232 specification. The PDP-8/X controller is compatible with the controller which was used on the PDP-8/I (the controller seems to have been considered to be part of the processor, and so it does not have an option number).

The receive side of the serial port emulates the teletype keyboard on device code 03. The I/O instruction is micro-coded, with bit [11] of the instruction determining if the IOT instruction should skip if the keyboard flag is set, bit [10] determining if the AC register and the keyboard flag should be cleared, and bit [09] determining if the receive data buffer should be merged with the AC register using a logical OR operation. Although all combinations of these three bits functions correctly, some combinations are more common that others, and are assigned mnemonics. Use of IOT instructions outside of this set may cause compatibility problems with software written for the (somewhat more sophisticated) interface of the PDP-8/E.

6031 KSF Skip if the keyboard flag is set.

6032 KCC Clear the AC register and the keyboard flag.

6034	KRS	Inclusive-or the contents of the keyboard buffer into the AC register.
6036	KRB	Load the contents of the keyboard buffer into the AC register, and clear the keyboard flag. The flag will set when the next character is received.

Although the serial port interface is essentially data leads only, the state of the receive flag can be read on the RTS pin (the active-low RTS signal is true if the receive flag is false). This allows a terminal emulator, which may be sending a file, to avoid sending characters when the PDP-8/X is not ready to read then; think of this as an RS-232 version of “reader run”.

The transmit side of the serial port emulates the teletype printer on device code 04. The I/O instruction is microcoded, with bit [00] of the instruction determining if the IOT instruction should skip if the teleprinter flag is set, bit [01] determining if the teleprinter flag should be cleared, and bit [02] determining if the data buffer should be written. Although all combinations of these three bits functions correctly, some combinations are more common that others, and are assigned mnemonics. Use of IOT instructions outside of this set may cause compatibility problems with software written for the (somewhat more sophisticated) interface of the PDP-8/E.

6041	TSF	Skip if the teleprinter flag is set.
6042	TCF	Clear the teleprinter flag.
6044	TPC	Begin printing the character whose code is in the AC register.
6046	TLS	Begin printing the character whose code is in the AC register, and clear the teleprinter flag. The flag will set when the character has been printed.

The serial port data rate is fixed at 9600 baud, which is generated by dividing down a 1.832 MHz oscillator.

At reset the keyboard flag and the teleprinter flag are cleared. This is a useful initial state for the keyboard flag, but not a useful initial state for the teleprinter flag, which must be set. The front panel program sets the teleprinter flag by typing a NUL character; the TFL instruction, which sets the teleprinter flag, cannot be used because TFL is a PDP-8/E only instruction.

3.2. Disk I/O

The disk port interfaces with disks which comply with the ATA-2 (IDE) specification. The PDP-8/X controller is very simple, and only supports type-0 programmed I/O operations (the slowest mode). There is no provision for direct memory access. The 512-byte sectors of the ATA-2 disk mesh perfectly with the 256-word sectors used by the PDP-8/X.

Although the ATA-2 interface is defined to have a 16-bit data bus, the most significant 8 bits of the bus are only used for data transfers. All control registers are defined to be 8 bits wide, and all control transfers happen on the least significant 8 bits of the data bus. This allows for a very simple

mapping of the PDP-8/X 12-bit data bus to the ATA-2 16-bit data bus; the PDP-8/X data bus is connected to the least significant 12 bits of the ATA-2 data bus, and the most significant 4 bits of the ATA-2 data bus are ignored on reads and unpredictable on writes.

The disk instructions are:

67x0	HSI	Skip if the ATA-2 interface is requesting an interrupt; that is, if the INTRQ signal is asserted.
67x4	HRR	Read a register in the ATA-2 command register block (that is, a register in the space which is accessed with CS0 asserted and CS1 negated). Bits [06..08] of the instruction word supply DA[02..00] to the ATA-2 interface. Bits [00..11] of the AC register get whatever data is supplied by the ATA-2 device; bits [00..03] of the AC register will be unpredictable if anything other than the data register (DA[02..00] = 0) is read.
67x5	HWR	Write a register in the ATA-2 command register block (that is, a register in the space which is accessed with CS0 asserted and CS1 negated). Bits [06..08] of the instruction word supply DA[02..00] to the ATA-2 interface. Bits [00..11] of the AC register are supplied to the ATA-2 device, which ignores bits [00..03] if anything other than the data register (DA[02..00] = 0) is written. The AC register is cleared.
67x6	HRS	Read a register in the ATA-2 control block (that is, a register in the space which is accessed with CS0 negated and CS1 asserted). Bits [06..08] of the instruction word supply DA[02..00] to the ATA-2 interface, but the only sensible value is 6, which accesses the alternate status register (hence the instruction’s mnemonic). Bits [00..11] of the AC register get whatever data is supplied by the ATA-2 device; bits [00..03] of the AC register will be unpredictable.
67x7	HWC	Write a register in the ATA-2 control block (that is, a register in the space which is accessed with CS0 negated and CS1 asserted). Bits [06..08] of the instruction word supply DA[02..00] to the ATA-2 interface, but the only sensible value is 6, which accesses the device control register (hence the instruction’s mnemonic). Bits [00..11] of the AC register are supplied to the ATA-2 device, which ignores bits [00..03]. The AC register is cleared.

Any IOT instruction with bit [09] = 0 acts as an HSI, so the undefined instructions (67x1, 67x2, 67x3) act as if they were 67x0. There is some possibility that one or more of these instructions may be reassigned in the future.

3.3. Expansion I/O

The expansion port is a header attached to the same busses used by the serial and disk I/O devices. Software transfers data to and from a device attached to

the expansion port using programmed I/O. There is no provision for direct memory access.

An empty expansion port connector never requests an interrupt. Expansion port devices should be designed to clear any interrupt request on reset, otherwise the console program will need to be updated.

4. Console Program

The console program performs all of the low-level initialization of the PDP-8/X system, all of the functions of the switches-and-lights front panel of the original PDP-8/I, and more (such as symbolic disassembly of memory).

When the PDP-8/X is reset the 4K word ROM is copied into the special front panel memory, the PC is set to 0000, and the IM flag is set to 0. A JMP at location 0000 sends control to the cold start entry point of the debugging program. The cold start entry point initializes the CPU, initializes the I/O devices, and jumps to the command parser.

When the PDP-8/X halts the PC register is stored in location 0000 of the panel memory, and control passes to location 0001.

The console program's parser works character-at-a-time so it can be small. When an illegal character is detected “XXX” is typed, and the partially parsed command is discarded. All numbers are octal.

A [d]	Examine or deposit the AC register.
D a [d ...]	Deposit memory in octal.
E a [n]	Examine memory in octal.
F [d]	Examine or deposit field data. The “d” argument is of the form IDSS, where I represents the IF/IB registers, D represents the DF register, and SS represents the SF register.
G [a]	Go. If an “a” argument is specified the PC is set to “a”, the IF, IB, and DF registers are set to 0, interrupts are disabled, and the AC is set to 0000, before the program starts.
I [d]	Examine or deposit the IE flag.
L [d]	Examine or deposit the LINK flag.
P [d]	Examine or deposit the PC register.
R	Load a binary file in BIN loader format. Only one segment of a multi-segment file will be loaded; multi-segment files (for example, FOCAL) will need to be reformatted into a single segment. If a checksum error happens reading stops, and the residual checksum is displayed.
S [d]	Examine or deposit the SW register.

U a [n]	Examine memory in octal and as instructions.
X op [d]	Execute the instruction specified by the “op” argument. If a “d” argument is specified then that value is loaded into the AC register before the instruction is executed, otherwise 0 is loaded into the AC register before the instruction is executed. The value of the AC register after the instruction is executed, and an indication if the instruction ends in a skip, is printed. The instruction is usually an I/O instruction, although any instruction can be specified.
Y	Read physical block 0 (which corresponds to logical block 0 of partition 0) of the disk into location 0 or normal memory. Normally this will contain an OS/8 bootstrap block, which can be started by a “G 0”.