PROGRAMMER'S

HANDBOOK

Pacific Data Systems, Inc.

PROGRAMMERS' HANDBOOK

Preliminary Edition

PACIFIC DATA SYSTEMS
PUBLICATIONS DEPT.
MARCH 1964

TABLE OF CONTENTS

The recent trend in computer development has been almost entirely toward bigger and faster machines. Computer manufacturers, pressured on the one hand by the military with requirements for faster computers to be used in real time applications, and on the other hand by industry and business with ever increasing work loads for computers, have pushed the computer speed from the millisecond to the microsecond range and are now talking about the nano-second computer. The penalty for this increase in speed and size has been in complexity and cost. It is true that the computer today represents a better bargain than it did ten years ago in terms of the amount of computations per dollar, but it is equally true that the computer still represents a sizeable investment which few companies can easily afford.

Furthermore, the cost of this equipment and its speed of operation, make it almost prohibitive to use as an open shop device. It becomes mandatory to employ the computer in the most efficient manner possible, and consequently any proposed use made of it must be carefully planned and scheduled. The net result is that in certain areas of potential computer applications, computers have made relatively few inroads. Open shop usage in large companies, full time use of the computer within a small or medium company, automatic checkout and quality control applications for non-military goods, to mention only a few examples, are some of the areas where the use of the computer has been impractical and too expensive up to now.

Pacific Data Systems has developed a general purpose, digital, stored program computer specifically designed to be used in such applications. The computer is offered in two basic configurations, the PDS 1020 and the PDS 1068.

The PDS 1020 is a desk mounted computer designed mainly for engineering and scientific computations. The PDS 1068 is a rack mountable unit, designed essentially to fit as a control logic module within a larger system. From a point of view of programming the two machines are identical, and this handbook is designed to serve both. It might be useful, however, to look at these machines separately for a moment, from the point of view of the user: at the advantages which each machine offers and the applications for which it was designed.

PDS 1020   The PDS 1020 is designed principally as a calculating tool for use by engineers scientists, researchers, and other technical personnel, whose work includes a great deal of numerical data manipulation. Since it is primarily designed for the engineering department, the computer can be used at many levels of sophistication. To the non-professional programmer it offers a powerful interpreter, simplified keyboard entry of instructions and data, special functions which can be used to perform a variety of complicated routines at the push of a button, and most important the ability to perform at more sophisticated levels as the user improves his understanding of computer programming techniques. To the sophisticated programmer the PDS 1020 offers a powerful machine language, paper tape input and output, a hardware index register, automatic word length control, capabilities for doing both decimal and binary arithmetic, a numeric register display, and many other important features. To any user, regardless of how much or how little he knows about computers, the PDS 1020 offers a convenient means of solving problems which can not be readily solved manually, yet cannot economically be solved on a large scale system. In a word it offers fast direct answers to immediate engineering and scientific problems.

PDS 1068   The PDS 1068 is designed primarily for systems applications. It provides the user with a complete general purpose computer control capability, at a price normally associated with limited special purpose equipment. Among the many features that make the 1068 particularly useful for such applications, are the parallel input and output capabilities, multiple input and output channels, all the programming features already enumerated above and external as well as program interruption capabilities. Most important, the 1068 allows the inclusion of an operator in the control system, to any degree desired by the user. By employing the sense switches, the special function switches, the override feature, etc., the operator can exercise as much or as little influence on the process under control as is required by the particular application. The PDS 1068 can be used effectively for process control, data logging, automatic checkout operations, on line quality control, and many, many others.

THE PDS 1020
AND PDS 1068
COMPARED

There are three basic differences between the PDS 1020 and the PDS 1068 units. The principal difference is mechanical - - the PDS 1020 is a desk mounted computer easily portable and convenient for office installation. The PDS 1068 is rack mountable, so that it may be installed as a module within a larger system in a minimum amount of space. Both computers use ordinary household current and require no special installation or environmental control.

Internally the two machines are identical, using the same command language, the same logic, and identical input and output channels. The only difference here is in basic memory size: the PDS 1020 is equipped with 2048 words of memory in its basic configuration, whereas the PDS 1068 is equipped with 1024 words of memory. In either case memory is expandable in modules of 1024 words.

Finally, the PDS 1020 includes a paper tape reader and punch, a type-writer, and a numeric keyboard, for input and output. The PDS 1068 includes channels for communicating with these devices, but the actual units are not included in the basic configuration. Thus, the systems designer has leeway to include whatever communications equipment he chooses for input and output in the 1068 system.

With these exceptions, the two machines are identical. All references in this handbook, to commands, logic, execution, input and output character-istics, and the like, can be understood to apply equally well to both machines.

COMPUTER MEMORY

The PDS computer is equipped with a magnetostrictive delay line memory. The principal of such a memory consists of introducing a time delay into the transmission of electrical signals representing binary information. Information is stored in the delay line by coding it into a series of electrical pulses which are transmitted along the delay line to a receiving station at the other end. There the signals are received, amplified, and recirculated. By synchronizing this activity with a pulsating clock, it is possible to retrieve any segment of information desired.

From the point of view of programming, the mechanics of this operation are immaterial. The important thing to remember is that information is sequentially stored in the delay line, much as it would be in any other storage medium.

Memory is divided into words, and each word in the PDS computer consists of 16 bits, a sign bit, and a marker bit. The basic memory capacity of the 1068 is 1024 words; the basic capacity of the 1020 computer is 2048 words. Memory is expandable in units of 1024 words. Memory addresses begin with zero and end with either 1023, 2047, or 4095, in a 1024 word machine, a 2048 word machine, or a 4096 word machine respectively. Addressing is modulo the memory size of the particular computer. Thus, the address 1024 in a computer which has a 1024 word memory, would be interpreted by the computer as an address of zero, an address of 1025 would result in an effective address of 1, etc.

HEXADECIMAL NUMBERING

Memory locations are addressed in binary format. These locations are referred to, for convenience sake, in hexadecimal notation. The hexadecimal system, as the octal system, bears a direct relationship to binary in that 4 binary bits represent one hexadecimal digit. This representation was chosen in preference to octal, because normal machine arithmetic in the PDS computer is performed in decimal. Thus, the machine looks at 4 bits at a time to arrive at a decimal or hexadecimal equivalent. The relationship of decimal, hexadecimal, and binary numbers are shown in Table A. Note that the digits corresponding to the decimal values 10 through 15 are L, C, A, S, M, and D respectively.

DECIMAL	HEXADECIMAL	BINARY
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	L	1010
11	C	1011
12	A	1100
13	S	1101
14	M	1110
15	D	1111
16	10	10000
1023	3DD	001111111111
1024	400	010000000000
2047	7DD	011111111111
2048	800	100000000000
4095	DDD	111111111111
4096	1000	1000000000000

To avoid confusion and distinguish between the various representations, the nomenclature in this handbook will be as follows: a bit will always refer to a binary digit; a digit will always refer to a decimal digit which is equivalent to 4 binary digits; a character will be any specified number of bits which is interpreted by the computer as a single item of information. For example, a hex character of 4 bits, an alphanumeric character of 6 bits, an alphanumeric character of 8 bits, etc.

WORD
CONTENTS

A word in the PDS computer consists of 4 digits plus sign and a marker bit, or 18 bits. The 17 bits are accessible to the program and can assume meaning according to the following formats:

1. A word may contain 4 signed digits of data.

2. A word may contain a partial data value of 4 decimal digits within a larger data field.

3. A word may contain an instruction. The instruction may consist of 4 digits and sign, or may consist of a signed hexadecimal character plus a 3 hexadecimal character operand address.

DATA AND
INSTRUCTION
MODES

The PDS computer operates in one of two modes: the data mode and the instruction mode. In the data mode, information (data or instructions) is placed in the accumulator and is manipulated by the instructions. In the instruction mode information is placed in the instruction register and executed as an

instruction. The information, in either mode of operation, may come from memory, from the keyboard, from the paper tape reader, or from a device connected to the parallel input channel.

Note that it is thus possible to execute instructions either from memory, or directly as they are entered from an input device, or in any combination of memory and external devices.

MACHINE
REGISTERS

There are 7 machine registers which are of interest to the programmer. A brief description of each follows, to acquaint the programmer with their general functions. More detailed explanations of how each register works, how it may be accessed by the program, etc., are reserved and will be discussed in context with the functions which they perform and the instructions which relate to them.

The registers are numbered G 0 through G 6, and may be displayed in the register display light by setting the display switch to the appropriate setting.

G 0 Next Instruction Register. This register holds the address of the next instruction to be accessed by the program for execution.

G1 Instruction Register. This register holds the instruction currently being executed by the computer.

G2 Word Length Register. The word length register controls the number of machine words which are considered by the computer as a single data word.

G3 Index Register. This is a hardware index register which can be used to index arithmetic and storage instructions or any other instructions which require sequential memory access.

G4 Link Address Register. The contents of this register are interchangeable with the contents of the G0, next instruction, register. It is used to hold the return address when the program exits to a sub-routine.

G5 Sign Register. This register holds the sign of the accumulator, which is created separately from the value of the accumulator.

G6 Accumulator. The machine accumulator is 25 decimal digits long, and is used with the arithmetic register as well as for most input and output operations.

INPUT/OUTPUT
EQUIPMENT

The PDS 1020 is equipped with a numeric keyboard, a paper tape reader and punch and a typewriter. Additionally a parallel input channel and a parallel output channel are provided, and may be connected to devices of the user's choosing.

The PDS 1068 is equipped with input and output channels, the characteristics of which are identical to those of the 1020 although the devices are not included in the standard configuration.

CONTROL
CONSOLE

The PDS computer console contains a variety of push buttons, switches, and indicators, which are used to initiate the operations of the equipment, for program checkout purposes, or, where desired, for operator control of machine operations.

Functionally speaking the console controls and indicators fall into four classifications:

1. Control Switches and Indicators. These include controls for turning power on and off, starting the computer, putting the computer in a single cycle mode, a parity error clear button, an override button for manual interruption of machine operations, a reset button, an input data indicator, and other indicators associated with the above switches.

2. Special Function Switches. These switches can be used by the operator to transfer control to a selected subroutine. The operator can therefore perform ten different functions, of any desired complexity, by merely pushing a button.

3. Sense Switches. A group of four sense switches is provided, along with four external sense lines. These may be tested under program control to determine their present state. The computer may be programmed to take appropriate action whenever one or more of these switches is turned on.

4. Register Display. A group of 17 display lights and an associated switch are provided to display the contents of machine registers. By setting the register display switch to a selected position number 0 through 6, the contents of the register correspondingly numbered will be displayed in the lights.

More detailed descriptions of these controls, indicators, and special function switches will be discussed in the following chapters when their functions and manner of operating can be more readily understood.

DECIMAL
ARITHMETIC

The normal arithmetic operations of the PDS computer are decimal.
A capability for performing binary arithmetic is provided, as will
be seen at the end of this chapter. The basic arithmetic commands
however, ADD, SUBTRACT, MULTIPLY, and DIVIDE, do their
arithmetic operations in decimal.

A data value is therefore represented in the machine as a series of
decimal digits, each digit consisting of four bits. The sign of the value
is separately handled and separately treated during the arithmetic oper-
ations, so that the actual calculations are performed on the absolute values
of the data.

Arithmetic operations are performed serially. In an ADDITION operation
for example, where two values A & B are to be added together, each digit
of A is separately added to the corresponding digit of B until all digits have
been added. Since 16 possible digits rather than ten can be represented in
four bits, a direct binary addition of the digits would not yield decimal
results. The hardware is so constructed, however, as to compensate for
this difference between the decimal and hexadecimal system, as will be
seen.

Arithmetic operations are performed on two values at a time, one of
which is always in the accumulator. The accumulator may be loaded
from memory or directly from an input device. A special register, the
G5 register, houses the sign of the value currently in the accumulator.
A second data value is retrieved from a memory cell specified by the
operand address of the arithmetic instruction. This value is loaded into
a separate register, not accessible by programming, which we will
call the B accumulator. The sign of the second data value is also loaded
into the G5 register. After the specified arithmetic operation has been
performed, the result replaces the original contents of the accumulator.

The accumulator is 25 digits long, and may therefore contain up to 6
machine words or a total of 24 decimal digits. The 24 digit length of the
accumulator, can be regarded as a single data field for certain arithmetic
operations, if desired. The 25th digit is used to hold overflow in such
cases and is also used by two of the output commands. The number of

digits which are affected by the arithmetic operations is determined by the word length register, which provides automatic control of the nunber of machine words which are treated as a single data field.

VARIABLE WORD LENGTH

The word length register, G2, is used to specify the number of machine words which make up a single data word. Since the accumulator length is six machine word lengths (24 digits not counting overflow), up to six words may be used as a single data field for arithmetic operations. In the case of ADD & SUBTRACT 24 decimal digits may be added or subtracted from an equal or smaller length value. In the case of MULTIPLY & DIVIDE the data word may be up to three machine words, 12 decimal digits,in length, yielding a 24 digit product in the case of DIVIDE.

The G2 register is 17 bits long, but only the least significant three bits are used in controlling the word length operations. The three least significant bits of the G2 register are loaded with the desired word length, by executing an SWL (Set Word Length) or an EWL (Exchange Word Length) command. This instruction will be further discussed under Indexing, in the chapter on Control Operations.

When the word length register is loaded with a desired field length, from 1 to 6 words, this field length is treated as though it were a single data word. This means that all arithmetic commands, as well as the load and storage commands,are automatically applied to as many machine words as make up the selected data field. For example, if a field length of 2 has been selected, a copy command would copy 8 digits from the accumulator into two consecutive memory cells. Only one operand address is required for the instruction, regardless of the word length used. To illustrate, C100+ if executed in double word length mode, would copy 8 digits from the accumulator to memory cells 100 and 101. The most significant 4 digits would reside in memory cell 100. The least significant 4 digits would reside in memory cell 101. The same situation, in reverse, exists when a load command is executed. The contents of 2 consecutive memory cells is loaded into the 8 least significant digits of the accumulator, with the digits which occupy the memory cell with the lower address loaded into the most significant digit position. Thus the command L100+, if executed in double length mode,

would cause the contents of memory cell 100 to be loaded into digit positions 5, 6, 7, and 8 of the accumulator, and the contents of memory cell 101 to be loaded into digit positions 1, 2, 3, and 4 of the accumulator.

The sign of the accumulator is copied into the sign position of every word within a data field when a storage instruction is executed. During a load operation, the sign of the word with the greatest memory address in the data field is loaded into the sign position of the accumulator; the signs of other words in the data field are ignored.

In the case of multiply and divide a double length accumulator is used. That is to say, if the data field length is two word lengths, 8 digits, the product of a multiplication will be 16 digits. Division also yields a 16 digit answer, the least significant 8 digits being the quotient of the division and the most significant 8 digits being the remainder (see below under DIVIDE). It therefore follows that the maximum word length allowable in MULTIPLY and DIVIDE operations is three word lengths, 12 decimal digits, yielding a 24 digit result.

The word length register also affects indexing operations, automatically adjusting address increments to correspond to the data field length chosen. This feature will be discussed under indexing in the chapter on control operations.

INSTRUCTIONS	MNEMONIC	MACHINE CODES
COPY	CPY	Cnnn+
	CPX	Cnnn–
LOAD	LDA	Lnnn+
	LDX	Lnnn–

G0   Incremented by 1.
G1   Contains the instruction
G2   Determines field length
G3   Indexing where specified
G4   Not affected
G5   Sign loaded or copied
G6   Value loaded or copied

RETRIEVAL & STORAGE

COPY

The COPY instruction (Cnnn+) copies the contents of the accumulator into location nnn. When multiple word length is specified by the word length register, the instruction will copy the contents of the data field specified from the accumulator into successive memory cells starting with memory cell nnn. The most significant 4 digits of the data field specified, will be copied into location nnn, successive digits will be copied into memory locations nnn+1, nnn+2, etc. The sign of the accumulator will be copied into each memory location affected by the instruction. The instruction may be indexed by terminating it with a negative sign (Cnnn–). When indexed the operand address will be modified by the index register, as described under Indexing in the chapter on control operations.

The contents of the accumulator is not destroyed by the COPY instruction.

The LOAD command (Lnnn+) clears the accumulator and replaces the contents of the 4 least significant digits of the accumulator with the contents of memory cell nnn. When multiple word length operations are performed, the load command accesses as many memory cells as are specified by the word length register. For example, if the specified word length is 3, the load command would replace the contents of the 12 least significant digits of the accumulator with the contents of memory cell nnn, nnn+1, and nnn+2. The contents of location nnn is always copied into the most significant 4 digits of the data field specified. The sign of the accumulator is replaced with the sign of the last memory cell accessed. The LOAD command does not destroy the contents of the memory cells.

LOAD

The LOAD command may be indexed by terminating it with a negative sign (Lnnn–). When indexed, the operand address of the command will be modified according to the information contained in the index register as described under Indexing, in the chapter on control operations.

	INSTRUCTIONS	MNEMONIC	MACHINE CODES

G0 Incremented by 1

G1 Contains the instruction

G2 Determines data field length

G3 Indexing affected where specified

G4 Not affected

G5 Sign algebraically determined

G6 Holds one of the operators before
      execution and the result after
      execution.

INSTRUCTIONS	MNEMONIC	MACHINE CODES
ADD	ADD	Annn+
	ADX	Annn-
SUBTRACT	SUB	Snnn+
	SUX	Snnn-
MULTIPLY	MPY	Mnnn+
	MPX	Mnnn-
DIVIDE	DVP	Dnnn+
	DVX	Dnnn-

ARITHMETIC OPERATIONS

The ADD command (Annn+) decimally adds the contents of memory cell nnn to the contents of the accumulator and leaves the result in the accumulator. Example 1 shows two adding operations,   to illustrate how the computer arrives at decimal answers.   The decimal numbers are of course coded in binary (8, 4, 2, 1 BCD Code) and the example shows three columns: In the first column the true decimal calculation is shown, in the second column the binary coded decimal operation as it is performed by the computer, and in the third column the true binary calculation.   To simplify the example a 2 digit calculation is shown; the same principle however applies to the entire length of the accumulator.

ADD

EXAMPLE 1. DECIMAL ADD

Decimal	Binary Coded Decimal	True Binary
A. 07	0000 0111	0000 0111
+ 08	+ 0000 1000	0000 1000
= 15	110	0000 1111
	0001 0101	
B. 05	0000 0101	0000 0101
+ 04	+ 0000 0100	+ 0000 0100
= 09	110	0000 1001
	0000 1111	
	- 110	
	0000 1001	

To compensate for the difference between the binary representation and the binary coded decimal, the machine automatically adds 6 to the digits added since 6 is the difference between the maximum decimal value which can be represented in 4 bits, and the maximum binary value which can be represented in 4 bits. If a carry into the next digit is generated by adding the 6, then the machine leaves the value untouched as can be seen in Example A: The most significant digit of the answer is 1 and the least significant digit is 5 which represents the decimal value 15 in BCD coding. In a binary machine this answer would be 1111, which is a binary representation of 15. In Example B a carry has not been generated into the next digit.

The machine therefore knows that the results can be represented correctly in 4 bits and subtracts the 6 to arrive at the correct answer.

Variable field length arithmetic may be performed by the ADD instruction, as specified by the word length register. The permissible data field for addition is from 1 to 6 words, or up to 24 decimal digits. During variable field operations the operand address is interpreted as nnn, nnn+1, nnn+2, etc. up to the specified number of memory cells which make up the single data field. The ADD command may be indexed by terminating it with a negative sign (Annn-). When indexed the operand address will be modified by the index register. If the result of the addition exceeds the field length specified by the word register, the overflow indicator will be turned on.

SUBTRACT

The SUBTRACT command (Snnn+) decimally subtracts the contents of location nnn, and subsequent memory locations where multiple word length is specified, from the contents of the accumulator. The result replaces the contents off the accumulator. The difference between the numbers is algebraic, and where numbers with different signs are subtracted from each other the difference may exceed the word length specified in which case an overflow condition will result. The numeric representation is binary coded decimal, and the machine compensates for the difference between binary and decimal numbers in much the same way as described under the ADD command.

The SUBTRACT command may be indexed by terminating it with a negative sign (Snnn-). When indexed the operand address will be modified by the index register. Up to six machine words, 24 decimal digits, may make up the data field for a SUBTRACT operation.

MULTIPLY

The MULTIPLY command (Mnnn-) multiplies the contents of the accumulator by the contents of memory location nnn. If multiple word length arithmetic is specified, the specified data field of the accumulator will be multiplied by the contents of memory locations nnn, nnn+1, and nnn+2. The result of the computation replaces the previous contents of the accumulator, and is always double the field length called for by the word length register.

Thus a single field length multiplication, 4 digits by 4 digits, will yield a double field product of 8 decimal digits. For this reason, a maximum of 3 word lengths can be used as a data field for a multiply operation, yielding a 6 word product of 24 digits. Digits outside the double data field length are set to 0 by the MULTIPLY command.

The MULTIPLY instruction may be indexed by terminating it with a negative sign (Mnnn-) in which case the operand address will be modified by the index register.

The DIVIDE command (Dnnn+) decimally divides the value in the accumulator by the value contained in memory location nnn. The value in the accumulator which is contained in twice the data field

DIVIDE .specified is decimally divided by the contents of memory location nnn and subsequent memory locations.

The results of the division are left in the accumulator in the following format: The quotient is in the least significant field length specified and the remainder is in the next significant field length specified. For example, if double length arithmetic is being performed, the value in the 16 least significant digits of the accumulator will be divided by the contents of memory locations nnn and nnn+1; when the division has been executed, an 8 digit quotient will be found in the least significant 8 digits of the accumulator and an 8 digit remainder will be found in digits 9 through 16 of the accumulator. Thus, the result is always double the field length specified by the word length reigster For this reason a maximum data field of 3 words, 12 decimal digits, may be used in division, yielding a 24 digit result.

Two precautions must be taken when executing a divide operation. First, the accumulator must not contain any non zero information outside twice the data field specified. If any such information is present in the accumulator while a division is being executed, even if this information is outside the double field length specified, the DIVIDE command cannot be correctly executed; the overflow indicator is turned on, and the accumulator is shifted one digit to the left.

Additionally, the denominator must be great enough, so that the resulting quotient will fit in the specified field length. If the denominator is too small, a quotient which is larger than the field length specified would result. Under such conditions however, the computer will not execute the division but will instead shift the contents of the accumulator one digit (4 bits) to the left, and turn on the overflow indicator. The following formula shows the conditions under which the command will not operate:

IF $\qquad$ $|N| \geq 10^m \times D$

Where   N is the numerator (the value in the accumulator)
          m is the number of digits in the data field specified
          D is the denominator (the value retrieved from memory)

THEN   overflow is turned on, and N is shifted left one digit (4 bits)

Note that the machine does not stop after it has turned on the overflow indicator and shifted the accumulator left, but continues on computing. It is therefore good practice where any doubt exists as to whether the denominator is large enough, to precede and follow the DIVIDE instruction with a test of the overflow indicator and a transfer to a corrective routine where necessary. The Transfer on Overflow instruction is discussed in the chapter on Control Operations.

The DIVIDE command may be indexed by terminating it with a negative sign (Dnnn-) in which case the operand address will be modified by the index register.

REGISTERS AFFECTED:	INSTRUCTIONS	MNEMONIC	MACHINE CODES
G0 Incremented by 1	ACCUMULATOR LEFT SHIFT	ALS	40XX+
G1 Holds the instruction being executed			
G2 Not affected	ACCUMULATOR RIGHT SHIFT	ARS	50XX+
G3 Not affected			
G4 Not affected	BINARY LEFT SHIFT	BLS	4100+
G5 Not affected			
G6 Shifted as directed by the instruction	BINARY RIGHT SHIFT	BRS	5100+

ACCUMULATOR SHIFTS

All shifts in the PDS computer are open shifts; bit positions vacated by the shifts are zeroed and bits shifted out of the accumulator, at either the high or low end, are lost. Left shifts into the 25th digit of the accumulator will turn on the overflow indicator. The word length register, however, is not effective during shift operations; a shift outside of the specified data field will not turn on the overflow indicator, unless the data is shifted into digit 25 of the accumulator. Shifts may be in binary or decimal units, and may be right or left shifts as specified by the command used.

LEFT SHIFTS

Accumulator Left Shift (40XX+) causes the contents of the accumulator to be shifted left XX decimal digits. XX are hexadecimal characters in the range 00 through DD. Note that the instruction must contain the correct hexadecimal representation of the number of decimal digits to be shifted. For example, if a left shift of 12 decimal digits is desired, the instruction should read 400A+, NOT 4012+. The latter instruction would cause a left shift of 18 decimal digits. It is possible to shift the accumulator left a total of 255 digits (DD). Needless to say, after a shift of a maximum of 25 digits to the left, the accumulator will contain nothing but zeros, and the overflow indicator will be on if a digit had been in the accumulator. This command may therefore prove useful in applications where desirable to have the computer wait a specified period of time before it continues in its execution.

The command 4000+, will be interpreted by the computer as a left shift of zero digits. This instruction disturbs none of the registers, and may be considered a "NO-OP" instruction.

RIGHT SHIFT

The Accumulator Right Shift (50XX+) operates exactly as described for the left shift, in the opposite direction. XX are hexadecimal characters, which determine the number of decimal digits to be shifted right. Vacated positions will be zeroed, and data shifted out of the accumulator is lost. A maximum of DD (255) digits may be shifted, and a right shift of zero digits constitutes a "NO-OP" instruction. An Accumulator Right Shift instruction will not turn on overflow.

BINARY SHIFT

Binary Left Shift (4100+) shifts the value in the accumulator one bit position to the left. The vacated bit position is zeroed one bit. A shifted into the 25th digit of the accumulator will turn on the overflow indicator.

Binary Right Shift (5100+) shifts the contents of the accumulator one bit position to the right. The vacated bit position is zeroed.

Note that the binary shift commands, unlike the decimal shift, are not flexible as to the number of bit positions shifted. A separate instruction must be issued for each bit position to be shifted. By combining the decimal and binary shifts, however, it is possible to shift any number of desired bit positions with maximum of 3 instructions. For example, a left shift of 3 bits, could be executed by using a decimal left shift of 1 digit (4 bits) and a binary right shift of 1 bit, resulting in a net shift of 3 bits to the left.

REGISTERS AFFECTED:		INSTRUCTIONS	MNEMONIC	MACHINE CODES
G0	Incremented by 1	LOAD MEMORY WORD	LMW	8nnn+
G1	Contains the instruction	STORE ADDRESS	STA	1nnn–
G2	Effective only for AND & ANX			
G3	Effective only for ANX	BINARY ADD	BAD	8nnn–
G4	Not affected.	COMPLEMENT	COM	7000+
G5	Affected only by AND & ANX			7001+
G6	Affected as specifically described.	EXTRACT & COMPARE	AND	9nnn+
			ANX	9nnn–

BINARY ARITHMETIC

While the normal arithmetic operations of the PDS computer are decimal, the instructions discussed here provide the capability for performing binary arithmetic where desired. This capability is particularly useful in performing address arithmetic, so that an address may be loaded, modified and stored during the execution of the program.

LOAD

LOAD MEMORY WORD (8nnn+) loads the contents of memory location nnn into the least significant 16 bit positions of the accumulator. The sign of memory word nnn is not loaded, so that the sign of the accumulator remains unchanged and other bit positions of the accumulator remain undisturbed. The Load Memory Word instruction cannot be indexed, nor is it affected by the contents of the word length register, so that only one memory location may be accessed by the instruction.

STORE

The STORE ADDRESS command (1nnn–) copies the contents of the 12 least significant bits of the accumulator into the 12 least significant bit positions of memory cell nnn. The 5 high order bits of memory cell nnn are undisturbed by this instruction, so that it is possible to change the operand address of an instruction residing in nnn without disturbing the instruction itself. The contents of the accumulator is undisturbed by this instruction.

ADD

BINARY ADD (8nnn–) adds the contents of memory location nnn to the least significant 16 bits of the accumulator. The sign of the accumulator remains unchanged and the sign of the memory location is ignored by this instruction. The adding process is binary, not decimal, as in the normal ADD instruction of the computer.

This instruction operates over a single word length only, 16 bits, and does not disturb any of the other digits in the accumulator.

The Binary Add instruction will not set overflow under any conditions, and if the result of the addition exceeds the maximum value which can be expressed in 16 bits (65,535) the accumulator at the end of the computation will contain the results as an excess of 65,536. 65,536 will be expressed as a zero, 65,537 will be expressed as a 1, etc.

COMPLEMENT

The COMPLEMENT command provides further binary capabilities, allowing subtraction and other binary arithmetic operations. 7000+ forms the 1's complement of the value contained in the least significant 16 bits of the accumulator. 7001+ forms the 2's complement of the value in the least signi-

ficant 16 bits of the accumulator. These instructions do not require an operand address since no memory cell is accessed. They do not change the sign of the accumulator or any of the information contained in other data fields of the accumulator. The Complement commands cannot be indexed, and are not affected by the contents of the word length register, so that they are operative only over a single data field, one word length, of the accumulator.

AND

COMPARE & EXTRACT is by rights a logic command, rather than an arithmetic command, and should properly be discussed in the chapter on control operations. Its usefulness, however, is mainly in manipulating binary data, and for this reason it is described here. This command performs two separate functions. It tests a data value contained in the accumulator against a test word in memory; if a match is found the overflow indicator is turned on; if no match is found the instruction performs an extract operation.

As a first step in execution, the overflow indicator is turned off. Next, the instruction, 9nnn+, examines the contents of memory cell nnn and compares it to the contents of the accumulator. The sign of memory cell nnn is compared to the sign of the accumulator. If an exact match is found, over the entire accumulator length including sign, the overflow indicator is turned on and the instruction is terminated. If an exact match is not found, an Extract operation, a logical And is performed. That is to say, the memory word is compared bit by bit to the word in the accumulator; matching bits are left the same, non-matching bits are zeroed. Example 2 shows three instances where the accumulator is compared to a memory word.

EXAMPLE 2. COMPARE & EXTRACT

A. Memory Cell   1 1 1 1 +
    Accumulator   0 0 . . . 0 0 1 1 1 1 +
    Result: Accumulator unchanged: Overflow indicator on.

B. Memory Cell   1 1 1 1 +
    Accumulator   0 0 . . . 1 1 1 1 1 1 +
    Result in Accumulator   0 0 . . . 0 0 1 1 1 1 +

C. Memory Cell   1 1 1 1 +
    Accumulator   0 0 . . . 0 0 1 0 0 1 -
    Result in Accumulator   0 0 . . . 0 0 1 0 0 1 +

Note that the Compare and Extract instruction works over the entire length of the accumulator regardless of the data field length specified. In instance A, Example 2, the accumulator contains the identical word as contained in the memory cell and the high order digits of the accumulator contain zeros; in this instance a complete match is found, the accumulator is left unchanged and the overflow indicator is turned on. In instance B, the accumulator contains an identical word to the test word in the specified word length, but the high order digits contain non zero information; since the entire accumulator is compared to the memory cell, an exact match is not found and an extract is performed, leaving the matching bit unaltered but zeroing the extraneous bits since there is no match for them in memory. Note that this is a convenient way of assuring

that the accumulator contains no information outside of the specified data field: an extract using a test word of all one bits, will always result in the specified data field of the accumulator remaining unchanged and of accumulator digits outside of the specified data field being zeroed. In instance C a simple extract operation is performed leaving the data word in the accumulator much as it was; note however that the sign of the accumulator has changed from a minus to plus by this operation, since a positive sign is represented by a zero and a negative sign is represented by a 1.

The Compare and Extract instruction may operate over a variable data field. As many as six words in memory may be compared simultaneously to the contents of the entire accumulator. Where multiple word length is specified by the word length register, memory cell nnn is compared to the most significant digits of the data field in the accumulator, and subsequent memory locations are compared to the least significant digits in the accumulator. The Compare and Extract instruction may also be indexed, by terminating it with a negative sign (9nnn-), causing the operand address of the instruction to be modified by the contents of the index register.

		INSTRUCTIONS	MNEMONIC	MACHINE CODES
G0	Incremented by 1	SET WORD LENGTH	SWL	600X+
G1	Contains the instruction	SET SIGN POSITIVE	SSP	7002+
G2	Set by SWL	SET SIGN POSITIVE	SSP	7002+
G3	Not directly affected	SET SIGN NEGATIVE	SSN	7003+
G4	Not affected			
G5	Affected by SSP and SSN only			
G6	Not affected			

SET UP INSTRUCTIONS

SET SIGN

These are not arithmetic instructions but instructions which are used in setting up arithmetic operations, as well as for other programming and housekeeping chores. SET SIGN POSITIVE (7002+) and SET SIGN NEGATIVE (7003+) set the accumulator sign either positive or negative regardless of its previous condition. Note that the accumulator sign is contained in register G5 and that the contents of the G6 register, the accumulator proper, is not affected.

SET WORD LENGTH

SET WORD LENGTH (600X+) is an instruction used to note the word length register with the desired word length which will be used by the computer in arithmetic operations as a single data field. X is a number from 1 to 6, which will be loaded by this instruction into the 3 least significant bits of the word length register, G2. This instruction must be executed prior to any multiple word length arithmetic; the 3 least significant bits of the G2 register normally contain a 1, so that single length arithmetic will be performed unless otherwise specified.

The SET WORD LENGTH command is executed only once for a given data word length. To change the data field length with which the computer is working, a second SET WORD LENGTH instruction may be used, resetting the word length register as desired. The command which can make use of the variable data field are LOAD, COPY, ADD, SUBTRACT, MULTIPLY, DIVIDE, and COMPARE & EXTRACT.

The SET WORD LENGTH command, and the word length register, are also used in index operations as will be described in the chapter on Control Operations.

CONTROL OPERATIONS

So far we have discussed only those commands which manipulate data, operate on data, or set up conditions for arithmetic operations. This chapter will be concerned with instructions per forming control operations - that is to say those instructions which change the sequence of execution, and which give the machine the ability to perform logical decisions based on previous results of computations, external conditions or a variety of other conditions set by the programmer.

Generally speaking control operations can be classified into four broad areas: unconditional transfers change the sequence of execution without regard to any conditions which may exist, either externally or internally in the machine; conditional transfers test for the presence of a condition and change the sequence of execution only where such a condition is present; indexing instructions permit the machine to perform the same sequence of instructions over and over again, using different data for each execution; finally, some instructions fall into the housekeeping category, and are used mainly in setting up necessary conditions for the operating program.

REGISTERS AFFECTED:

		INSTRUCTIONS	MNEMONIC	MACHINE CODES
G0	Loaded with Jump Address	JUMP	JMP	2nnn+
G1	Contains the instruction	HALT	HLT	2nnn-
G2	Not affected			
G3	Not affected	JUMP-LINK	JPL	3nnn-
G4	Affected by JPL, RTN, & ELA instructions	RETURN	RTN	3007+
G5	Not affected	EXCHANGE LINK	ELA	3004+
G6	Affected only by ELA	ADDRESS		

UNCONDITIONAL TRANSFERS

JUMP

The unconditional jump instruction (2nnn+) disrupts the sequential execution of instructions and executes the instruction contained in memory cell nnn. When this instruction is executed, the contents of the G1 register, the instruction register, is copied into the G0, next instruction register. The computer then looks at the G0 register to see where it is to go for its next instruction, finds there the address nnn, and loads the contents of that memory cell into the G1 register to be executed. The G0 register is then incremented in the normal manner so that its address is now nnn+1, and after the instruction from memory cell nnn has been executed, the computer will proceed to nnn+1 to get the next instruction. The Jump instruction makes no provision for preserving the address in the program from which it has transferred. If a return is desired to the point of the program where the transfer occurred, the Jump-Link instruction should be used.

Note that the computer has no means of identifying whether a given memory cell contains data or an instruction. Care must therefore be exercised in transferring control, to avoid picking up a data word by mistake and loading it into the instruction

register can be interpreted as an instruction. For example, if a Jump instruction were given to a memory location which contained nothing, this would be interpreted by the instruction register as being an Input Instruction from Tape command. Careful attention must therefore be given to Jump addresses to assure that the memory cells referred to contain instructions, not data.

HALT

The Halt instruction (2nnn-) is essentially a Halt and Transfer instruction. When this instruction is executed the computer will stop, and when the start button is depressed, the instruction in location nnn will be executed. Note that this instruction stops the computer after it has accomplished the transfer of the instruction from the instruction register and into the G0, next instruction, register. If the registers are displayed during this Halt, the instruction 2nnn- will be seen both in the G0 and the G1 registers. When the start button is depressed the machine will look at the G0 register to determine where it is to take its next instruction and proceeds to read the next instruction from location nnn.

JUMP-LINK

The Jump Link instruction (3nnn-) operates in much the same way as the Jump instruction with one important difference. This instruction preserves the previous contents of the G0 register after incrementing it by 1, and stores it for future use in the G4 register. Thus a return address to a program is automatically provided by this instruction. As an example, suppose that at location 100 a transfer to a subroutine is desired, located in memory cell 500.

The instruction 3500- would accomplish the transfer to location 500, and store the address 101 in the G4 register. By using the return instruction described below, it is then possible for the programmer to return to location 101 and resume the sequential execution of his program when the subroutine is terminated.

RETURN

Return (3007+) is the instruction used to transfer back from the subroutine into the main program. This instruction exchanges the contents of the G4 register with the contents of the next instruction register. If the subroutine was entered with a Jump-Link instruction, the G4 register contains the address of transfer +1, as described above. This address is now transferred back into the G0, next instruction register, and the computer then resumes the normal execution of the main program. Note that the Jump-Link and Return instructions provide the programmer with a convenient means of entering and returning from a subroutine regardless of where in the main program transfer instructions occur.

In the example above, if the subroutine at location 500 is frequently used, transfers to it could occur in locations 25, 45, 93, and 100 in the main program. In each case these locations will contain a Jump-Link instruction to location 500 (3500-). In each case the G4 register would be loaded with the address following the jump instruction, i. e. 26, 46, 94, or 101. The subroutine itself is terminated with a Return instruction (3007+) which in each case

would load the appropriate address into the G0 register and return control to the point in the program where execution is to be resumed.

EXCHANGE
LINK
ADDRESS

To add one more level of complexity, assume that the subroutine itself must transfer to another subroutine as part of its execution. For example, assume this to be an exponential subroutine which refers back to a log subroutine. In this case the subroutine must contain provisions for saving the address in the main program to which it is to return control when it is through. The instruction EXCHANGE LINK ADDRESS (3004+) accomplishes this purpose. This instruction exchanges the contents of the Link Address register with the contents of the least significant field of the accumulator. Thus a return address to the main program is transferred to the accumulator where it may be stored in memory by the subroutine.

REGISTERS AFFECTED:		INSTRUCTIONS	MNEMONIC	MACHINE CODES
G0	Incremented by 1	TRANSFER ON OVERFLOW	TOV	4nnn-
G1	Contains the instruction			
G2	Used by TXH	TRANSFER ON ZERO	TZE	5nnn-
G3	Used by TXH			
G4	Not affected	TRANSFER ON MINUS	TMI	6nnn-
G5	Examined by TMI	TRANSFER INDEX HIGH	TXH	7nnn-
G6	Examined by TZE			

CONDITIONAL TRANSFERS

Conditional transfers are those transfers which test for the presence of a specified condition. If such a condition is present at the time the instruction is executed, then the transfer takes place. If the condition is not present when the instruction is executed, then no transfer takes place and the computer goes on with its sequential execution.

IF OVERFLOW

Transfer on Overflow (4nnn-) will test the overflow indicator to determine whether it is on. If the overflow indicator is on then a transfer to location nnn takes place. If the overflow indicator is on and the transfer is executed, the instruction will turn the overflow indicator off. The following instructions can, in course of their execution, turn on the overflow indicator:

    ADD
    SUBTRACT
    DIVIDE
    COMPARE AND EXTRACT
    TEST SENSE SWITCH
    ACCUMULATOR LEFT SHIFT
    BINARY LEFT SHIFT

IF ZERO

Transfer on Zero (5nnn-) examines the contents of the entire accumulator including the 25th, overflow digit. If any one bits are contained in the accumulator the next sequential instruction is executed. If the entire accumulator contains zeros only, then the next instruction is executed from location nnn. Note that the sign of the accumulator is not examined by this instruction and can be either positive or negative.

IF NEGATIVE

Transfer on Minus (6nnn-) examines the sign of the accumulator. If the sign is positive the next sequential instruction is executed. If the sign is negative the next instruction is sought at location nnn.

Transfer Index High (7nnn-) serves as the index testing instruction of the computer and merits discussion under a separate paragraph along with the other aspects of manipulating the index register.

INDEXING

Frequently it is desirable to perform a given sequence of instructions several times using different sets of data each time. For example it may be desirable to add 100 data values to a cumulative sum; this could be done by storing the data in 100 memory locations and then issuing 100 ADD instructions such as A000+, A001+, A002+, . . . etc. By using the index register only two instructions (ADX and TXH) would be necessary to perform this adding operation instead of 100 instructions. The indexed ADD instruction, Annn-, would be modified by the index register to pick up consecutively, first memory location 000 and, on the next execution, 001 and so on for each execution until all data values are added together in the accumulator.

Thus the purpose of the index register is to modify an operand address during the execution of the program (the address in memory is unchanged, it is modified during the execution of the command only). Two factors enter into the indexing operation: (1) The value by which the operand address is modified; (2) The number of times the indexed instruction will be executed. The first of these factors is controlled by the word length register G2. The second, the number of times the index instruction will be executed, is controlled by the initial value loaded into the index register, and by the TXH instruction. The TXH instruction serves a triple purpose. It decrements the contents of the index register, it tests the index register to see if the specified number of executions have been performed, and if the specified number has not been reached it transfers control to a specified location.

The operation of the index register is best illustrated by example. In Example 3, it is assumed that 4 data values have been stored in memory locations 000 through 003. The instruction sequence located in memory cells 500 and up, must load the value in memory cell 000 into the accumulator and add the values in cells 1, 2, and 3 to it, storing the accumulated result in memory cell 100. In instance A of Example 3, this operation is executed without the benefit of an indexed instruction; in instance B the index register is used. Note that only one instruction has been saved in this example, since only three data values must be added. In fact, the saving is illusory because a number of instructions are necessary to set up the index register. It is easy to see, however, that if 100 values were to be added, the saving could be considerable since the instruction sequence could add 3 numbers, 100 numbers, 300 numbers or any other number of data values. Only 3 are shown in the example for the sake of simplicity.

In preparation for executing these instructions, two things must be done: the word length must be set to the desired data field length, in this instance 1; the index register itself must be loaded with the value 2 for reasons which will be explained shortly.

EXAMPLE 3. INDEXING OPERATIONS

A. Location	Instruction	Op. Address
500	LDA	000
501	ADD	001
502	ADD	002
503	ADD	003
504	CPY	100
505	Continuation	

B.		
500	LDA	000
501	ADX	003
502	TXH	501
503	CPY	100
504	Continuation	

C. Contents of Word Length and Index Registers

	W. L.	Index
Initially	0001	0002
After TXH executed once	0001	0001
After TXH executed 2nd time	0001	0000
After TXH executed 3rd time	0001	0DDD

Let us see now what happens when the instructions in sequence B are executed. At location 500 the computer reads the LOAD command and loads the value contained in memory cell 0 into the accumulator. It then proceeds to cell 501 and finds there the indexed ADD command with an operand address of 003. The computer then subtracts the current contents of the index register from the operand address. The index register was loaded with the value 2 and this value is subtracted from the operand address, yielding an effective address of 001. The computer therefore executes the ADD command, by adding the contents of memory cell 001 to the contents of the accumulator.

Example 3 C shows the contents of the word length and of the index registers initially and after each pass of execution.

TRANSFER
INDEX
HIGH

The computer now gets to memory cell 502 and reads there the TXH instruction. This instruction does two things at this point: First, it compares the value currently in the index register with the value in the word length register; if the value in the index register is either larger than or equal to the value in the word length register then a jump is executed. Simultaneously with this comparison the TXH instruction subtracts the word length register from the index register and the result becomes the new contents of the index register. As can be seen in Paragraph C of Example 3, the index register contains a 1 after the TXH instruction has been executed the first time. In the meantime the jump has been executed to location 501 as specified by the TXH operand address. In location 501 the computer finds the indexed ADD instruction once again, and again subtracts the current contents of the index register which is now 1, from the operand address. The effective address is therefore 002 and the computer adds the value in location 002 to the contents of the accumulator. The TXH instruction is now executed a second time; once again a comparison is made between the contents of the index register and the contents of the word length register. At this point both registers contain an equal value (1001) and the instruction transfers again to location 501. At the same time a subtraction is also performed, setting the contents of the index register to 0. At location 501 the ADX instruction is executed and the contents of the index register is subtracted from the operand address; since the index register now contains a zero the operand address cited in the instruction, 003, will remain the effective address and the computer will add the contents of memory cell 3 to the contents of the accumulator. The TXH instruction is now executed a third time, but this time it finds that the contents of the index register (0000) is smaller than the contents of the word length register (1). The jump is therefore not executed. Instead the computer then goes on to the next sequential instruction, 504, and executes the copy command storing the result of the addition in memory cell 100.

The contents of G2 is subtracted from the index register, even though the jump is not executed. Since the subtraction is binary, not decimal, it is executed without a sign, by adding the 2's complement of G2 to G3. Since at this point G2 contains 1 and G3 contains 0, the final contents of G3 is

Initial contents of G3 (12 bits only)	0000	0000	0000
2's complement of G3 (12 bits only)	+ 1111	1111	1111
Final contents of G3 (12 bits only)	1111	1111	1111

Note that the subtraction takes place over the least significant 3 digits (12 bits) of the registers. The final contents of the index register is, therefore, ODDD.

To summarize, the TXH instruction when executed, compares the contents of the index register and the contents of the word length register; if the index register contains a value that is equal to or greater than the value contained in the word length register, a jump is executed to the location specified in the TXH instruction. If the value in the index register is smaller than that contained in the word length register the next sequential instruction is executed. Regardless of the result of the comparison, the index register is decremented by the value contained in the word length register. The current value of the index register is subtracted from the operand address of any instruction which is indexed.

EXCHANGE INDEX REGISTER

The word length register is loaded by the SET WORD LENGTH instruction (SWL) already discussed. The index register is loaded by using an Exchange Index Register instruction (EIR, 3002+). This instruction exchanges the contents of the index register with the contents of the least significant field of the accumulator. Thus the value to be loaded into the index register must first be loaded into the accumulator either from memory or directly from the input device. In single word length operations, the index register must be loaded with the number of times that the instruction is to be executed, less 1. The reason is, that the index command will be executed one more time after the contents of the index register has reached zero, as seen in Example 3.

Note that when operating with multiple word lengths, the indexed instructions will automatically compensate for the word length used, since the index register is always decremented by the contents of the word length register. When executing multiple word length operations, however, the index register must be loaded with a value according to the following formula:

$$X = WL (N - 1)$$

where X is the value to be loaded into the index register, WL is the current value of the word length register, and N is the number of times the instruction is to be executed. To illustrate, suppose that the ADD instruction is to be executed 4 times, double precision. Using the formula, the index register must be loaded with the value 8 - 2 = 6. When writing the operand address for the index instruction, it is necessary to adjust its value to the field length and to the value contained in the index register. For example, suppose that the

indexed ADD instruction in the illustration cited, must first add the contents of memory cells 1 and 2 to the accumulator, and then 3 and 4, then 5 and 6, and finally 7 and 8. If the instruction is written as A008- (ADD index modified) the contents of the index register which we have set at 6, will be subtracted from this number yielding an effective address of 002.

Since double length arithmetic is specified, the instruction would then access cells 002 and 003 which is the wrong set. It is necessary, therefore, to write the instruction as A007-; in this case the contents of the index register would be subtracted from 7 yielding an effective address of 001 and cells 1 and 2 in memory would be accessed, as is the intention of the program. The rule therefore is, when writing an index instruction, the address of the instruction less the contents of the index register should yield the first memory address to be accessed by that instruction.

Note that all addresses, and the values loaded into the index register must be in hexadecimal notation.

Seven commands may be indexed, and will be affected by the index register when terminated with a negative sign. These commands are:

LOAD	LDX	Lnnn-
COPY	CPX	Cnnn-
ADD	ADX	Annn-
SUBTRACT	SUX	Snnn-
MULTIPLY	MPX	Mnnn-
DIVIDE	DVX	Dnnn-
COMPARE &	ANX	9nnn-
EXTRACT		

REGISTERS AFFECTED:		INSTRUCTIONS	MNEMONIC	MACHINE CODE
G0	Incremented by 1 and exchanges with G4 by RTN	EXCHANGE WORD LENGTH	EWL	3000+
G1	Contains instruction			
G2	Exchanged with G6 by EWL	EXCHANGE INDEX REGISTER	EIR	3002+
G3	Exchanged with G6 by EIR			
G4	Exchanged with G6 by ELA - RTN	EXCHANGE LINK ADDRESS	ELA	3004+
G5	Not affected			
G6	Exchanged with G2, G3, G3, or G4, as described.	RETURN	RTN	3007+

REGISTER
EXCHANGES

The register exchange commands are used mainly in setting up the registers to perform the programming functions already described. The EIR instruction is used to load the index register, the ELA instruction is used to save the contents of the G4 register whenever it is desired to store the return address from a sub-routine; the RTN instruction is used to return to the main program from a sub-routine, to the point where the sequential execution had been interrupted.

EXCHANGE
WORD
LENGTH

Exchange word length exchanges the contents of the word length register, G2, with the least significant 16 bits of the accumulator. The instruction may be used whenever it is desired to save the contents of the word length register for future use. For example, suppose that the subroutine makes use of the index register, using single length data. The program might be using multiple word length arithmetic and it is necessary to reset the word length register to single length. Before this is done it might be advisable to save the contents of the word length register so that when the subroutine returns to the main program it can reset the data word length to whatever it was before the subroutine was entered.

G0 Incremented by 1
G1 Contains the instruction
G2 Not affected
G3 Not affected
G4 Not affected
G5 Not affected
G6 Not affected

TEST SENSE SWITCH   TSW   0YX0-

SENSE SWITCHES

Four sense switches are provided in the control panel of the PDS computer. Additionally 4 sense lines are provided in the output channels of the computer, and may be connected to external devices. These sense switches and/or sense lines may be tested by programming to determine their condition at any given time. When a tested sense switch is on, or when a sense line under test is true, the overflow indicator will be turned on. The test instruction may be then followed by a Transfer on Overflow to transfer to the appropriate subroutine.

The instruction 0YX0-, tests a sense switch, a sense line, or any combination of switches or lines as specified by the hexadecimal characters YX. Each bit position in the least significant hexadecimal character (X) corresponds to a sense switch, and the corresponding sense switch will be tested when this bit is a 1. The most significant hexadecimal character (Y) represents the 4 sense lines and a 1 in any of the bit positions will cause one of the sense lines to be tested. Table A shows the test instruction as written to test the various sense switches and lines as well as the binary representation of the hexadecimal characters.

TABLE A

INSTRUCTION	HEXADECIMAL CHARACTERS	OPERATION
0010-	0000 0001	Test Switch 1
0020-	0000 0010	Test Switch 2
0040-	0000 0100	Test Switch 3
0080-	0000 1000	Test Switch 4
0100-	0001 0000	Test Line 1
0200-	0010 0000	Test Line 2
0400-	0100 0000	Test Line 3
0800-	1000 0000	Test Line 4
00D0-	0000 1111	Test All Switches
0D00-	1111 0000	Test All Lines
0DD0-	1111 1111	Test All Switches and All Lines

Note that a single instruction can test more than one switch or line; in fact the last instruction in the table, 0DD0-,will test all the sense switches and all the sense lines, since all eight bits of the hexadecimal characters are 1's. Table A shows 11 possible test instructions; actually it is possible to test all switches and all lines in various combinations for a total of 256 conditions, by using various combinations of sense switches and sense lines.

The overflow indicator is turned off as a first step in executing the test instruction; the instruction then proceeds to test the specified switches or lines, and if any of the switches is on or any of the lines is true when tested, the overflow indicator is turned on. If more than one switch or sense line is tested, overflow will be turned on if either switch or line is on or true.

The instruction 0000- will merely turn off the overflow light and cause no further action.

A computer is nothing more than a mass of useless wires and electrical components unless it has a means of communicating with the outside world. To do useful work, the computer must have input: Programs to instruct it what to do, and data with which it must work. It must then communicate the results of its computations through output devices.

The PDS 1020 computer includes a numeric keyboard for manual input, a paper tape reader, and a Selectric typewriter which may be optionally activated for input. The paper tape punch, and the Selectric typewriter are the standard output devices for the PDS 1020. Additionally, the computer features parallel input and output channels, which may be connected to any input or output device selected by the user.

The PDS 1068 offers these same devices as optional equipment; the basic configuration, however, includes only channels for communicating with these or like devices.

From a programming point of view, there is no difference between the PDS 1020 and the 1068; the programmer is merely concerned with the format of information input and output through the various channels, without regards to what devices originate these signals or will ultimately receive them.

The commands described here, even though discussed in terms of the PDS 1020 equipment, are therefore applicable to the PDS 1020 or 1068 input or output channels.

The major feature which characterizes the PDS computers' output and input structure, is the fact that it may operate in one of two modes. Instruc-tions and data may be input to the accumulator, as with any other computer, to be stored in memory and executed at a future time. Instructions may however, also be input directly into the instruction register for immediate execution. Likewise, certain parts of the instruction may be output from the instruction register as a special purpose code to a typewriter or a punch.

REGISTERS AFFECTED:		INSTRUCTIONS	MNEMONIC	MACHINE CODES
G0	Incremented by 1	INPUT INSTRUCTION FROM TAPE	IIT	0000+
G1	Serves as I/O register for IIT and OUT			
G2	Not affected	INPUT DATA FROM TAPE	IPT	0001+
G3	Not affected			
G4	Not affected			
G5	I/O register for sign of accumulator	INPUT DATA BINARY	IDB	0007+
G6	I/O register for IDP, IDB, OPU, OPB	OUTPUT	OUT	10XX+
		OUTPUT TO PUNCH	OPU	1200+
		OUTPUT BINARY	OPB	1L00+

PAPER TAPE INPUT/OUTPUT
The input from paper tape instructions specify one of three modes of operation:

1. Characters from paper tape are read, placed in the instruction register and executed.

2. Characters are read from paper tape and placed in the least significant digit of the accumulator.

3. Characters are read from paper tape and placed in the two least significant digits of the accumulator.

The output instructions similarly specify one of three modes of operation:

1. The least significant 8 bits of the instruction register are output to the punch.

2. The most significant digit of the accumulator is output to the punch.

3. The least significant 2 digits of the accumulator are output to the punch.

During input in modes 1 and 2, the 8 level input goes through an automatic conversion process and is entered as a single 4 bit digit. Similarly a single digit is output to the punch as an 8 level code when output mode 2 is used. This conversion is accomplished by hardware, and no programming conversion is necessary in these modes. The character formats are shown in Table B.

INPUT INSTRUCTION
Input Instruction from Tape (0000+) reads 8 level tape from the paper tape reader 1 character at a time, until a sign character, either plus or minus, is read. When the sign character is read the input is terminated and the last 5 characters read, including the sine, are placed into the instruction register and executed as an instruction. The 8 level input is converted to a single hexadecimal character as shown in Table B. Eight level codes not shown in Table B should not be used; they will either be ignored or cause wrong input.

INPUT DATA
Input Data From Tape. (0001+) reads 8 level tape, 1 character at a time, until a sign character is read. Eight level code is converted into hexadecimal characters, as shown in Table B, and each character entered is placed in the least significant digit of the accumulator; preceding characters are shifted 1 digit left. A sign character, when read, terminates the input and is placed in the sign position of the accumulator in register G5. This instruction will read correctly only the characters shown in Table B; illegal characters are ignored, or result on wrong input to the computer.

| | INPUT BINARY | Input Data Binary (007+) will read a single 8 level character without decoding it, and place it in the least significant 2 digits (8 bits) of the accumulator. The previous contents of the accumulator is shifted 4 digits to the left. This instruction reads only a single character; no decoding is done and all codes are legal. |

INPUT
BINARY
Input Data Binary (007+) will read a single 8 level character without decoding it, and place it in the least significant 2 digits (8 bits) of the accumulator. The previous contents of the accumulator is shifted 4 digits to the left. This instruction reads only a single character; no decoding is done and all codes are legal.

OUTPUT
TO
PUNCH
The output instruction (10XX+) will output the least significant 8 bits of the instruction register and punch them as an 8 level code on the paper tape punch. XX are hexadecimal characters which will be output to the punch. No decoding of any sort is done and all codes are legal. Output to Punch (1200+) will output the most significant digit of the accumulator, digits 25, to the punch. The single digit, 4 bit, hexadecimal character will be encoded and output to the punch as an 8 level code. The conversion is as shown in Table B. Note that only the 16 hexadecimal characters may be output by this command. The signs of the data, plus or minus, must be output from the instruction register using the 10XX+ command (10LC+ as +, 10LS+ as -).

OUTPUT
BINARY
Output binary (1L00+) will output the least significant 8 bits of the accumulator to the punch. These bits are punched exactly as output.

TABLE   B

HEXADECIMAL - 8 LEVEL CONVERSION

HEXADECIMAL CHARACTER	BINARY (4 BIT)	8 LEVEL	
0	0000	1011 0000	
1	0001	1011 0001	
2	0010	1011 0010	
3	0011	1011 0011	
4	0100	1011 0100	
5	0101	1011 0101	
6	0110	1011 0110	
7	0111	1011 0111	
8	1000	1011 1000	
9	1001	1011 1001	
L	1010	1100 1100	
C	1011	1100 1011	
A	1100	1100 0001	
S	1101	1101 0011	
M	1110	1100 1101	
D	1111	1100 0100	

Input conversion only (+ / + / 1010 1011) Input conversion only
(- / - / 1010 1101)

REGISTERS AFFECTED:	INSTRUCTIONS	MNEMONIC	MACHINE CODES

REGISTERS AFFECTED:

G0 Incremented by 1
G1 8 least significant bits output by OUT
G2 Not affected
G3 Not affected
G4 Not affected
G5 Contains parity check bit after ITY
G6 25th digit output by OTY; least
       significant field loaded by ITY

INSTRUCTIONS	MNEMONIC	MACHINE CODES
INPUT FROM TYPEWRITER	ITY	0006+
OUTPUT TO TYPEWRITER	OTY	1630+
OUTPUT	OUT	14XX+

TYPEWRITER INPUT/OUTPUT

The typewriter input channel may be enabled at the users' option by adding a special logic board. When this channel is activated the Input from Typewriter instruction (0006+) will cause a character from the typewriter to be entered and placed in the least significant field of the accumulator. Two types of information may be transmitted from the typewriter: An alphanumeric character or a machine function. An alphanumeric character is entered as a 6 bit code, and placed in the least significant 6 bit positions of the accumulator. Table C in the Appendix shows the hexadecimal equivalents of the 6 bit codes input for alphanumeric characters. Parity of the alphanumeric characters is checked and the 7th bit position of the accumulator will be set as an odd parity bit by a separate data line. If parity is correct, a check bit will be placed in the sign position of the accumulator. During typewriter input, therefore, the sign of the accumulator is not a mathematical sign but rather a parity check. It may be checked for correct parity, after each instruction by executing a Transfer on Minus instruction. If parity is correct the sign position will contain a 1 and the transfer will occur.

INPUT

In addition to alphanumeric characters, machine function codes may be entered by executing the proper function of the typewriter. For each machine function entered, a corresponding bit position in the first data field of the accumulator is set to 1. Thus the space function will enter a 1 into the 9th bit position, carriage return into the 10th bit position, tab into the 11th bit position, backspace into bit 12, index into bit 13, upper case into bit 14, lower case into bit 15. In addition a mode code is also used during typewriter input. The 8th bit position of the accumulator is set to 1 if the character entered is upper case, and is left zero if the character is lower case. Finally, an End of Line code is entered as a 1 bit into the 16th bit position of the accumulator, if the typewriter carriage has reached the end of the line.

Output to Typewriter (1630+) will output the most significant digit of the accumulator, digit 25, to the typewriter. The 4 bits of the 25th digit will be output as the appropriate hexadecimal character 0 through D on the typewriter. The conversion is done by hardware, and is automatic. Only the hexadecimal characters 0 through D may be output by this command.

OUTPUT

Output (14XX+) will output the hexadecimal digits XX from the instruction register to the typewriter. Any alphanumeric character, or machine function may be output by this instruction. XX are hexadecimal characters and will print on the typewriter the appropriate letter, or cause the appropriate function, as shown in Table C in the Appendix.

REGISTERS AFFECTED:	INSTRUCTIONS	MNEMONIC	MACHINE CODE
G0 Incremented by 1	INPUT INSTRUCTION		
G1 Loaded by IIK	FROM KEYBOARD	IIK	0002+
G2 Not affected			
G3 Not affected	INPUT DATA FROM		
G4 Not affected	KEYBOARD	IDK	0003+
G5 Sign loaded by IDK			
G6 Loaded by IDK			

KEYBOARD
INPUT

Input Instruction from Keyboard (0002+) will cause the computer to accept serial input of hexadecimal characters from the numeric keyboard channel. When a sign character is read, either a plus or minus, the input is terminated and the 5 characters last entered, including the sign, are placed in the instruction register and executed.

DATA
INPUT

Input Data from Keyboard (0003+) causes the computer to accept hexadecimal characters serially from the keyboard input channel. Characters are placed in the least significant digit of the accumulator, and as new characters are entered are shifted left 1 digit at a time. A sign character, either plus or minus, when entered, terminates the input. The sign character is not placed in the accumulator but in the sign position of the accumulator in Register G5.

SPECIAL
FUNCTION
SWITCHES

The execution of either of the keyboard input instructions enables the 10 special purpose switches on the control keyboard. The operator at his option, may input hexadecimal characters through the numeric keyboard or depress one of the special purpose switches, when an input from keyboard instruction is executed. Each of these special purpose switches, when depressed, will enter a Jump-Link instruction either to the instruction register or to the accumulator, depending on whether the input instruction is 0002+, or 0003+. Each of these Jump-Link commands has its own unique address as follows: the special function key marked A generates a Jump-Link instruction to location 6, the key marked B to location 7, C to location 8, D to location 9, E to location L, F to location C, G to location A, H to location S, I to location M, and J to location D. These locations may then contain the transfer instruction to an appropriate subroutine which will execute the special purpose function for which the switch was intended. The nature of such a subroutine, and thus the function of each switch, are entirely up to the programmer and can be determined to suit his particular application.

REGISTERS AFFECTED:		INSTRUCTIONS	MNEMONIC	MACHINE CODES
G0	Incremented by 1	INPUT INSTRUCTION PARALLEL	IIP	0004+
G1	Contains the instruction			
G2	Not affected	INPUT DATA PARALLEL	IDP	0005+
G3	Not affected			
G4	Not affected			
G5	Sign bit loaded by IDP, Output by OPP	OUTPUT PARALLEL	OPP	1800+
G6	Loaded by IDP, Output by OPP	TEST SENSE SWITCH	TSW	0YX0-

PARALLEL
INPUT/OUTPUT

The Parallel Input and Output channels may be used to input or output 17 bits simultaneously to or from the accumulator, or to and from the instruction register. Additionally both the input and the output channel contain the 4 external sense lines already mentioned under control operations, which may be tested by the TSW instruction to determine their condition at a given time.

INPUT

Input Instruction Parallel (0004+) will input 17 bits simultaneously to the instruction register. This information will then be interpreted as an instruction and executed.

Input Data in Parallel (0005+) shifts the contents of the accumulator 4 digits to the left and enters 16 bits of information from the parallel data lines into the 16 least significant bit positions of the accumulator. The 17th data line is connected to the sign position of the accumulator and register G5 and is used to input the sign of the entering character.

OUTPUT

The Output in Parallel instruction (1800+) transmits the contents of the least significant 16 bits of the accumulator and the contents of the sign bit of the accumulator in register G5, over the parallel output channel.

There are 4 sense lines in the parallel input channel and 4 sense lines in the parallel output channel. These are identical lines and are physically inter-connected so that either input or output devices may be tested by the program. Care should be taken however, that the same pair of lines not be connected both to an input and an output device since the Test Sense Switch instruction would then turn on the overflow if either of the lines was on, without any means of determining whether the true signal came from the input or from the output channel.

The operating procedure described here is intended for the PDS 1020 computer. It includes descriptions and operating procedures for the PDS 1020 input and output equipment. The PDS 1068 control panel, operates in the same way as described here except of course that the input and output equipment may be different than that which is standardly supplied with the PDS 1020 computer.

The computer control panel, exclusive of the numeric keyboard, contains 4 groups of indicators and switches:

1. The special function keys, already described under input/output.

2. The branch switches described under Control Operations.

3. Register display lights and an associated display switch.

4. The operating controls and indicators, which are used to control and monitor the computer operations.

LOADING THE PROGRAM

To operate the computer, first turn machine POWER on by pushing the POWER button. This button contains an indicator and will light up when on; if pushed a second time, the machine power will be turned off. Load the program tape into the tape reader and turn the tape reader power on. When the START button light comes on, push the START button and the tape will be read into the machine.

Internally, when power is first applied to the machine, the computer goes through an initializing cycle, setting up certain markers in memory and in the registers in preparation for machine operations. At the end of the initialize cycle, all the registers contain zeros, except for the G2, word length, register which contains a 1. Thus the machine is ready to operate on single word length data. When the initialize cycle is over, the START button indicator will be turned on, and the machine will be in a HALT or idle position.

When the START button is pushed, the computer starts execution in the normal manner. The G0 register is examined for the next instruction address; this register contains 0, so the computer proceeds to load the contents of cell 0 into the instruction register. Cell 0, of course, contains a 0, and this is loaded into the instruction register and interpreted as a command to input from tape (0000+). An instruction will now be read from tape, placed in the instruction register and executed. Suppose this instruction is 0001+. A second instruction will now be read from tape and placed in the accumulator. In the meantime the G0 register has been incremented by 1, and the computer will seek the next instruction from memory cell 1. Since nothing has as yet been copied into memory, cell 1 contains a 0. The instruction register will therefore again be loaded with an Input Instruction from Tape command, and will proceed to read another instruction from the tape. This instruction may now be a COPY instruction which will store the instruction previously read into the accumulator.

The program will normally be ended by a HALT instruction, with a transfer address to the first location in which the program has been stored. When the entire tape has been read, the computer will then halt and the operator may now take any other preparatory action that may be necessary. Sense switches may be set as called for in the program, the data tape may be loaded into the tape reader. Leader may be punched on the paper tape punch, data may be inserted into the typewriter, tab stops on the typewriter may be set, etc. etc. When the START button is depressed the computer will start the execution of the program read into memory.

TROUBLE-
SHOTTING

Several controls and aids are provided in the PDS computer, to help the programmer check out his program and correct any errors incorporated in it. These checkout and correction aids consist mainly of the register display, the SINGLE CYCLE button, the OVERRIDE button, the RESET button, and the ERROR CLEAR button.

REGISTER
DISPLAY

17 lights are provided in the register display, and may be used to display the contents of machine registers G0 through G6. An associated register display switch may be set to any number 0 through 6, and will cause the corresponding register to be displayed in the display lights.

The registers can only be displayed when the machine is in a halt condition, when the SINGLE CYCLE mode is used, or when the INPUT DATA light is on. The entire length of the G0, G1, G2, G3, G4, and G5 registers is displayed.

The G6 register which is the accumulator and contains 25 digits, cannot be displayed in its entirety. The least significant data field of the accumulator, 4 decimal digits or 16 bits, is displayed in the display lights when the display switch is set to 6, 7, 8, or 9. The sign of the accumulator is not displayed from bit position 17, but from the G5 register which contains the sign of the accumulator in its 10th bit position. When G5 is displayed, the 10th bit contains the sign of the accumulator.

SINGLE CYCLE

The SINGLE CYCLE button may be used to stop the computer during the normal execution of the program. When the button is depressed the computer will finish executing the current instruction and will then halt. The SINGLE CYCLE light will come on and the computer will continue to operate in the SINGLE CYCLE mode, until the button is depressed a second time. While in the SINGLE CYCLE mode, the computer will execute one instruction each time the START button is depressed. After each instruction has been executed the computer will halt, allowing the programmer to check the results of the instruction by displaying the contents of the accumulator or of other registers.

OVERRIDE

The OVERRIDE button is provided to allow the programmer to make the corrections or additions to the program manually, through the keyboard, whenever desired. When the OVERRIDE button is pushed a 0002+ command (Input Instruction from Keyboard) is forced into the instruction register. The

programmer may now enter an instruction through the keyboard; the instruction will be executed and the computer will return to await further instructions from the keyboard. This condition will persist as long as the computer is in the OVERRIDE mode, and the OVERRIDE light is on. When the OVERRIDE button is pushed a second time the mode is terminated and the computer will resume execution of the stored program.

RESET   The RESET button is provided mainly for maintenance purposes, and should not be used during the normal operation of the computer. The RESET button will reset some of the machine registers and may destroy parts of the program. The only condition under which it might be useful to the programmer, is when the computer is unable to execute an input or output instruction. For example, an input instruction may be given such as 0005+ which calls for input from the parallel data lines. If the input device has been turned off, the computer is unable to execute the instruction, nor is it able to proceed to the next instruction. Under these circumstances the RESET button may be used to free the computer from this condition.

ERROR CLEAR   The ERROR CLEAR light will come on whenever the computer detects a memory parity error. When the ERROR CLEAR button is pushed, the light will be turned off, and the programmer can proceed to search for and correct the parity error.

OVERFLOW
AND
INPUT DATA   In addition to the controls and indicators described, two other indicators are provided on the computer control panel: the OVERFLOW indicator and the INPUT DATA light. OVERFLOW is turned on whenever an overflow condition is discovered by the computer. Testing the sense switches and the sense lines can also turn on the overflow indicator as previously shown. The overflow indicator cannot be turned off from the control panel. It can only be turned off by an instruction, such as TOV or TSW as shown in the preceding chapters. The INPUT DATA light will come on whenever the computer has executed an 0003+ command (Input Data from Keyboard), to inform the operator that the computer is ready to accept data from the keyboard.

TABLE C

HEX CODE	U.C.	L.C.		HEX CODE	U.C.	L.C.
19	A	A		3M	Z	Z
20	B	B		3D	±	1
29	C	C		3L	@	2
2S	D	D		3C	#	3
2A	E	E		35	$	4
0C	F	F		3A	%	5
0D	G	G		38	¢	6
24	H	H		3S	&	7
18	I	I		39	※	8
0M	J	J		30	(	9
28	K	K		34	)	0
25	L	L		00	_	-
1D	M	M		0L	+	=
2L	N	N		1M	°	!
15	O	O		05	?	/
0A	P	P		1L	.	.
08	Q	Q		09	,	,
1S	R	R		13	SPACE	
14	S	S		02	INDEX	
2M	T	T		03	CAR RTN	
2C	U	U		06	U.C.	
1C	V	V		07	L.C.	
10	W	W		23	TAB	
2D	X	X		17	BACK SP.	
04	Y	Y		0S	:	;

APPENDIX

ALPHANUMERIC CODES

	INSTRUCTION	MNEMONIC	MACHINE CODE	PAGE NO.
STORAGE	Copy	CPY*	Cnnn+	15
	Load Accumulator	LDA*	Lnnn+	15
DECIMAL ARITHMETIC	Add	ADD*	Annn+	16
	Subtract	SUB*	Snnn+	17
	Multiply	MPY*	Mnnn+	17
	Divide	DVP*	Dnnn+	18
SHIFTS	Accumulator Left Shift	ALS	40XX+	20
	Accumulator Right Shift	ARS	50XX+	20
	Binary Left Shift	BLS	4100+	20
	Binary Right Shift	BRS	5100+	20
BINARY ARITHMETIC	Load Memory Word	LMW	8nnn+	21
	Store Address	STA	1nnn-	21
	Binary Add	BAD	8nnn-	21
	Complement	COM	7000+	21
	Two's Complement	COM	7001+	21
	Compare & Extract	AND*	9nnn+	22
SET-UP	Set Word Length	SWL	600X+	24, 31
	Set Sign Positive	SSP	7002+	24
	Set Sign Negative	SSN	7003+	24
TRANSFERS	Jump	JMP	2nnn+	25
	Halt	HLT	2nnn-	26
	Jump-Link	JPL	3nnn-	26
	Return	RTN	3007+	26
	Transfer on Overflow	TOV	4nnn-	28
	Transfer on Zero	TZE	5nnn-	28
	Transfer on Minus	TMI	6nnn-	28
	Transfer Index High	TXH	7nnn-	28
EXCHANGES	Exchange Link Address	ELA	3004+	27
	Exchange Word Length	EWL	3000+	33
	Exchange Index Register	EIR	3002+	33
	Test Sense Switch	TSW	0YX0-	34
INPUT AND OUTPUT	Input Instruction from Tape	IIT	0000+	36
	Input Data from Tape	IDT	0001+	36
	Input Data Binary	IDB	0007+	37
	Output (to typewriter)	OUT	14XX+	38
	Output (to punch)	OUT	10XX+	37
	Output to Punch	OPU	1200+	37
	Output Binary	OPB	IL00+	37
	Input from Typewriter	ITY	0006+	38
	Output to Typewriter	OTY	1630+	38
	Input Instruction from Keyboard	IIK	0002+	39
	Input Data from Keyboard	IDK	0003+	39
	Input Instruction Parallel	IIP	0004+	40
	Input Data Parallel	IDP	0005+	40
	Output Parallel	OPP	1800+	40

*These operations may be indexed.

The bootstrap routine appears on tape as follows: (the instruction numbers are only for reference, and do not appear on tape)

1.	0001+	*
2.	27D8+	
3.	27D8+	*
4.	C7D9+	*
5.	0001+	*
6.	L7D8+	
7.	C7DL+	*
8.	L7D8+	*
9.	0001+	*
10.	0001+	
11.	C7DC+	*
12.	0001+	*
13.	C0LS-	
14.	C7DA+	*
15.	0001+	*
16.	77DL-	
17.	C7DS+	*
18.	0001+	*
19.	2000-	
20.	C7DM+	*
21.	L7D8+	*
22.	0001+	*
23.	LS+	
24.	3002+	*
25.	27DL+	*

The bootstrap routine takes advantage of the unique ability of the PDS computer to execute instructions directly as they are input, as well as from a stored program.Instructions marked with an asterisk are executed as they are read from paper tape;instructions 2, 6, 10, 13, 16, 19 and the data value at Step 23, are stored in memory and used by the computer to load the user's program.

Briefly, the operation of the program is as follows:

When power is turned on, the initialize cycle completed, and the start button depressed, the computer looks at G0 for the address of an instruction. G0 contains 0, and the computer loads the contents of Cell 0, which is 0, into G1. This is interpreted as an instruction (0000+, ITT), and the computer reads instruction 1 from paper tape, and executes it. This instruction (0001+, IDT) causes the next instruction to be read from tape and placed in the accumulator. This is a jump to location 7D8, which we will call A for convenience. In the meantime G0 is incremented and the computer now reads the instruction in Cell 1. This is again 0, and instruction 3 is executed from tape. This is a jump to A. G0 now contains A, and the computer seeks the next instruction there. A contains 0, and instruction 4 is executed, copying the "Jump to A" instruction into cell A+1. Prior to each of the asterisked instructions the computer goes to A+1 (since G0 is always incremented) jumps to A, finds a 0, and executes the IIT instruction, taking the next instruction from tape and executing it.

When the program as punched is finished, memory contains the following:

Location A	contains	0
Location A + 1	contains	Jump to A
Location A + 2	contains	Load Accumulator with A (Clear Accumulator)
Location A + 3	contains	Input Data from Tape (reads an instruction into the Accumulator)
Location A + 4	contains	Copy Indexed to B
Location A + 5	contains	Transfer Index High to A + 2
Location A + 6	contains	Halt at C
Index Register	contains	N

This sequence will clear the accumulator, read information from tape and store it, indexed, starting at C, where C=B-N, until N instructions have been read from tape and stored. The computer will then halt and, when START is pushed, start execution at location C. The jump at A+1 allows the bootstrap to be used as a loader, with other programs present in memory. All that is necessary is to set A and A+1 to zero. Every loader or bootstrap written as the one shown can be entered with the following sequence.

PUSH OVERRIDE

Enter From Keyboard:	6001+	(Sets G2 to 1)
	0003+	(Input Data From Keyboard)
	0000+	
	C7D8+	(Sets A to 0)
	C7D9+	(Sets A+1 to 0)
Push OVERRIDE	27D8-	(Halt and go to A)
Push START		

The machine can now execute the bootstrap, or any similarly written loader.

HEXIDECIMAL CONVERSION TABLE

HEX.	DEC.	HEX.	DEC.	HEX.	DEC.
0	0	0	0	0	0
1	1	10	16	100	256
2	2	20	32	200	512
3	3	30	48	300	768
4	4	40	64	400	1024
5	5	50	80	500	1280
6	6	60	96	600	1536
7	7	70	112	700	1792
8	8	80	128	800	2048
9	9	90	144	900	2304
L	10	L0	160	L00	2560
C	11	C0	176	C00	2816
A	12	A0	192	A00	3072
S	13	S0	208	S00	3328
M	14	M0	224	M00	3584
D	15	D0	240	D00	3840
10	16	100	256	1000	4096

EXAMPLES:

	Decimal		Hex
	198 = 192	=	A0
	+ 6		+ 6
			A6

	1246 = 1024	=	400
	+ 16		+ 10
	6		6
			416

	Hex		Decimal
	MSA = M00	=	3584
	+ S0		+ 208
	A		12
			3804

APPENDIX