Mines Branch Information Circular IC 189

AN INTRODUCTION TO THE PDP-8 COMPUTER, ITS OPERATION AND PROGRAMMING

by

C. A. Josling*

- -

A description of the functioning and programming of a small computer (Digital PDP-8) which may be used to control experiments (on-line) is given in this circular.

In part one the principal components and their operation are discussed. Special features that enable it to be used on-line are described. The arrangement of the memory into pages is explained.

Part two deals with the programming of the computer in a symbolic language known as "PAL" or program assembly language. A sample program which adds two numbers together is given. It is assumed that the reader has no previous knowledge of computers.

*Technician, Mineral Physics Section, Mineral Sciences Division, Mines Branch, Department of Energy, Mines and Resources, Ottawa, Canada.

Direction des mines

Circulaire d'information IC 189

UNE INTRODUCTION À L'ORDINATEUR PDP-8, SON FONCTIONNEMENT ET SA PROGRAMMATION

par

C.A. Josling*

- - - -

RÉSUMÉ

Cette circulaire décrit le fonctionnement et la programmation d'un petit ordinateur (Digital PDP-8) qui peut être employé pour contrôler des expériences.

Dans la première partie, on discute des principales composantes et de leurs opérations. On décrit également des caractéristiques spéciales qui permettent de l'employer "sur ligne" (on-line). On explique aussi l'arrangement de la mémoire en pages.

Dans la deuxième partie, on traite de la programmation de l'ordinateur dans un langage symbolique connu sous le nom de "PAL". On donne l'addition de deux nombres comme exemple de programmation. L'auteur suppose que le lecteur n'a pas de connaissances préalables des ordinateurs.

---

*Technicien, Section de la physique minérale, Division des sciences minérales, Direction des mines, ministère de l'Energie, des Mines et des Ressources, Ottawa, Canada.

CONTENTS

= = =

FIGURES

- - - - - -

INTRODUCTION

Computers have been in use for over thirty years. The first computers were huge machines using thousands of vacuum tubes and relays, and were, of course, very expensive. When transistors were introduced, the size and cost of computers dropped considerably and in the past few years small, high-speed, very reliable, low-cost computers have been available.

Large computers are still necessary for complex mathematical problems, bookkeeping, etc., while the small ones may be used in the laboratory to control experiments. When a computer is used for this purpose it is referred to as "on-line".

There are instruments used to control variables in experiments, e.g. temperature, pressure, pH, etc., but these are limited to the use for which they were designed. Computers may be programmed (soft ware) to control practically any experiment, and, in fact, may look after several at the same time. For example, the PDP-8 computer* at the Mines Branch is programmed to control the Mössbauer effect spectrometer and an X-ray diffractometer simultaneously.

There are two types of computers, the digital and the analogue. There are many "analogue computers" in use which are generally not thought of as computers. For example, a temperature controller is essentially an analogue computer. Basically, an analogue computer is a device which works on a variation of voltage or speed.

The digital computer is the one generally used for mathematical computation, etc. It works with pulses and is therefore more accurate than an analogue device.

This report will deal only with the latter type of computer, using the PDP-8 machine and its peripheral equipment (see Figures 1 and 2) as an example.

---

*Program Data Processor 8, manufactured by Digital Equipment of Canada, Ltd., Carleton Place, Ontario.

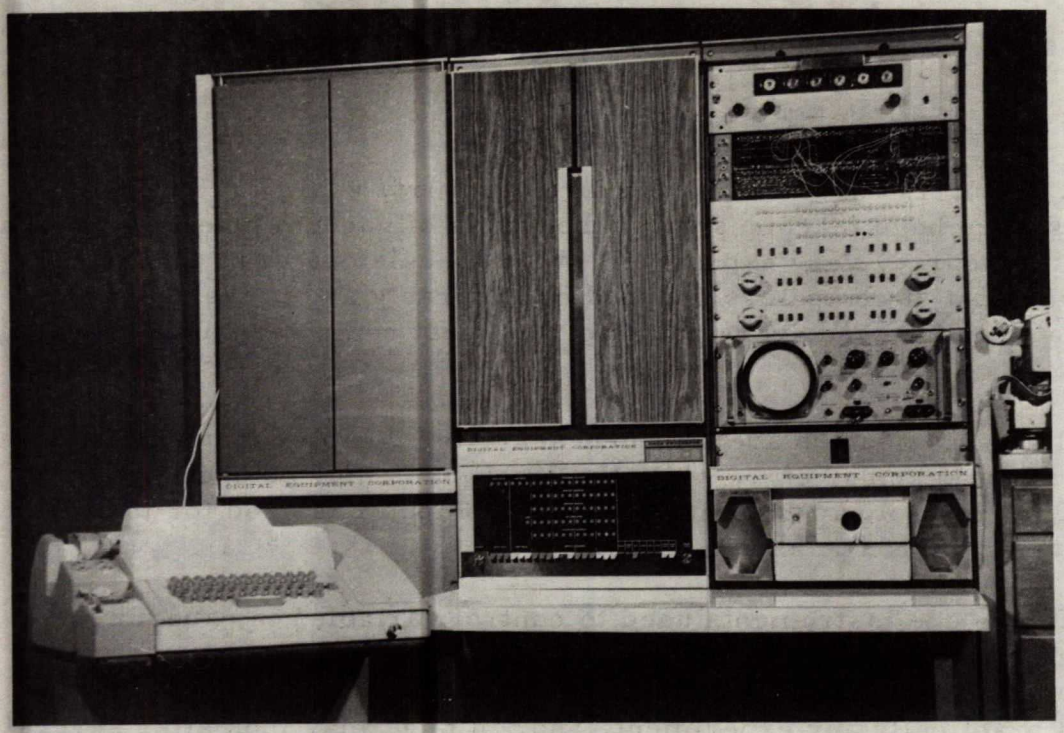


Figure 1. Overall view of computer.

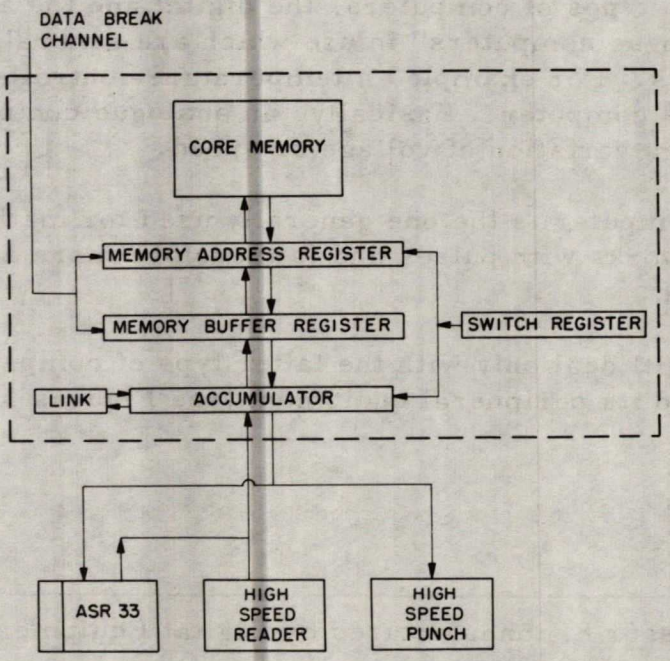


Figure 2. Schematic of computer and peripheral equipment.

PART 1 -- AN EXPLANATION OF DIGITAL COMPUTERS

OVERALL DESCRIPTION OF A COMPUTER

Most people think of a computer as a glorified desk calculator; this idea is entirely wrong. The correct definition of a computer is that it is a device which can automatically perform a series of operations on data presented to it in digital form. Outwardly it is a box with blinking lights and various devices coupled to it, as shown in Figure 1, but it might be compared to a hotel with many rooms.

In the case of the PDP-8, there are 4,096 rooms (Registers). Each room accommodates twelve people (Bits). The twelve people, as a group, would be a party (Word). It could be said to be a

4,096 12-person party hotel, or a

4,096 12-bit word computer.

The hotel has 32 floors (Pages). Every floor has 128 rooms (Registers). Each room has a number (Address) on the door. The occupant of one room may call the occupant of any room on the same floor (Current Page) or on the ground floor (Page Zero).

To call the occupant of a room on any other floor, a switchboard (Indirect Addressing) must be used. When called, the switchboard gives the number of the room whose occupant is to be contacted.

The hotel has a front door (Accumulator), through which most of the people (Bits) pass. It also has a back door (Data Break Channel), through which only certain people (Experimental Data) are allowed to pass. The back door is an express entrance which allows faster movement of traffic that has precedence over the data entering the front door.

A DESCRIPTION OF VARIOUS PARTS OF A COMPUTER

The analogy of the hotel has explained the computer in broad terms. In this section the parts will be dealt with in detail. Figure 2 is a block diagram showing the principal components of the computer and of the peripheral equipment.

Register

A Register is an electronic device which stores twelve binary bits of information called a "Word". A binary bit is either a zero or a one. Just as the occupant of a room may change, the twelve bits of a word stored in the register can be changed on command. There are 4,096 twelve-bit registers in the PDP-8, and these comprise what is commonly referred to as the "Memory" or "Core Memory".

Accumulator

The Accumulator (AC) is a complex register through which the information normally passes to and from the computer. The exception is the Data Break channel.

The AC does much more than allow information to pass through it, for it is there that all the arithmetic operations are performed. The contents may be cleared, rotated left or right, or complemented (all zeros changed to ones and all ones to zeros). The contents of the Memory Buffer can be added to those of the AC and the results stored away. The contents of the AC may be seen on a set of 12 lights on the console panel (Figure 3).

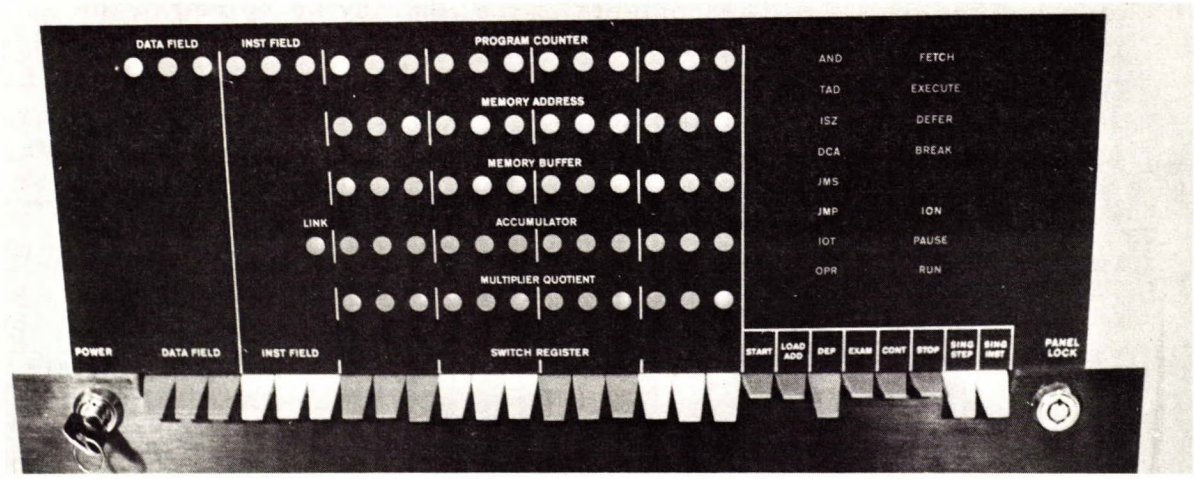


Figure 3. Console panel.

Link

To the left of the AC lights is a light by itself. This represents a one-bit register, known as the "Link", which may be cleared, complemented, or rotated as part of the AC. The Link detects accumulator overflow and is used for double precision arithmetic. It may be used as a "Signal Flag".

Program Counter

In a computer the instructions must be done sequentially. This sequencing (program control) is carried out by a 12-bit register known as the "Program Counter". The address of the next instruction is stored here.

Auto-Index Registers

Eight registers are reserved for Auto-Indexing. "Addresses" may be stored in these registers, and when called on indirectly, they are incremented by one. The contents of the AC may be stored in the new address. For example, if the address 1234 is stored in an Auto-Index register and the instruction "DCA I 10" is given, the computer will take the address found in core location 10, which is 1234, increment it to 1235, and DCA (Deposit Contents of Accumulator) in the new address.

Memory Buffer Register

All information being transferred to and from the memory must pass through a 12-bit register known as the "Memory Buffer Register".

Input-Output Devices

The input-output devices, which operate via the AC, use a binary code (0 or 1). Information must be encoded, for the computer can recognize only information given to it in this form. This code is known as the ASCII, or "asky", code (American Society for the Computer Interchange of Information). Most of the computer input and output consists of paper tape punched with this code. There are two types of paper tape. The first is a rolled tape used by the slow-speed devices. The second is a folded tape known as "fan fold" and is used by the high-speed reader and punch. Information from the tape enters the computer through the AC, 8 bits at a time; similarly, information from the computer passes through the AC and is punched on tape 8 bits at a time. The AC can handle 12 bits, but the tape requires only eight.

Input Devices

The devices that input information through the AC are:

1. Switch Register (SR)
2. Teletype (ASR33)
3. Slow-Speed Reader (SSR)
4. High-Speed Reader (HSR)

The switch register is a set of twelve switches on the console panel, Figure 3. Each switch represents a bit; if the tab is down the bit is a "zero", if the tab is up the bit is a "one". The SR is a slow means of communicating with the computer. Each switch has to be set for every "bit" of a word and the word deposited in the memory by lifting the "Deposit" tab. Only short messages, under special circumstances, are ever relayed to the computer by this means. An example of such a message is the RIM Loader, which sets up the computer to receive messages from other input devices.

The teletype (ASR33), Figure 4, is used to write programs and give commands to the computer. It is an electric typewriter that encodes the typed character into binary form. The encoded character is held in a buffer register until it is transferred to the AC.

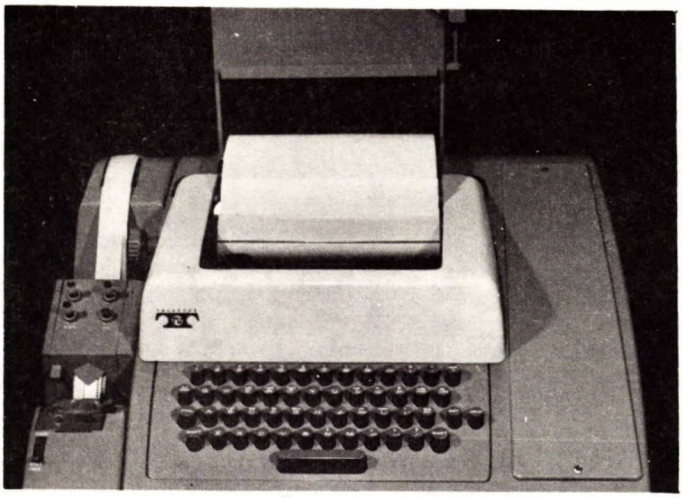


Figure 4. Teletype ASR33.

The next devices are readers which scan punched paper tape and convey the information to the computer. The first is a ten-character-per-second reader (SSR), Figure 4, which is mechanically coupled to the ASR33. The reading is done by eight small metal fingers. The second and more frequently used is a three-hundred-character-per-second photo-electric unit (HSR) (Figure 5), which mechanically feeds the tape over eight midget photo-diodes which read the data. Both of these devices store the information in buffer registers until it is accepted by the computer.

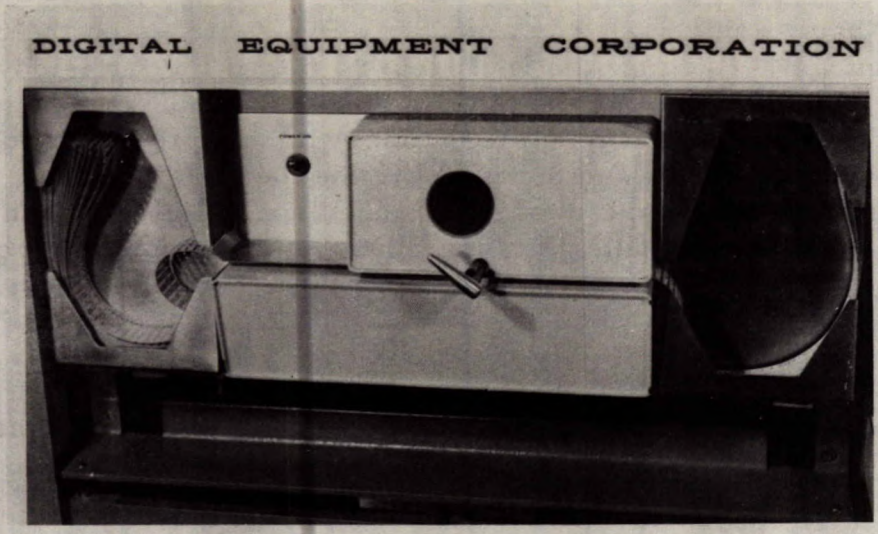


Figure 5. High-speed reader.

Output Devices

The system includes three output devices:

1. Teletype (ASR33)
2. Slow-Speed Punch (SSP)
3. High-Speed Punch (HSP)

The teletype can either send or receive messages from the computer, but is only used as an output device if a typed copy is required. The punched paper tape is by far the most useful. There are two devices which produce this: the first of these is the Slow-Speed Punch (SSP), Figure 4. This is the output equivalent of the SSR and punches out the contents of a section of memory on rolled paper tape at ten 8-bit characters per second.

The second of the paper tape output mechanisms is an accessory called the High-Speed Punch (HSP), Figure 6, also known as the "BRPE" (pronounced "Burpee" from the four letters in the identifying code). This produces fan fold tape at 63.5 8-bit characters per second.

Usually, paper tapes are punched in the ASCII code, but with programming any code may be produced. For example, if the data collected from experiments must be processed on machines which use

only IBM code, the PDP-8 computer can be programmed to translate ASCII to IBM. Figure 7 shows the alphabet in ASCII and IBM codes for comparison.

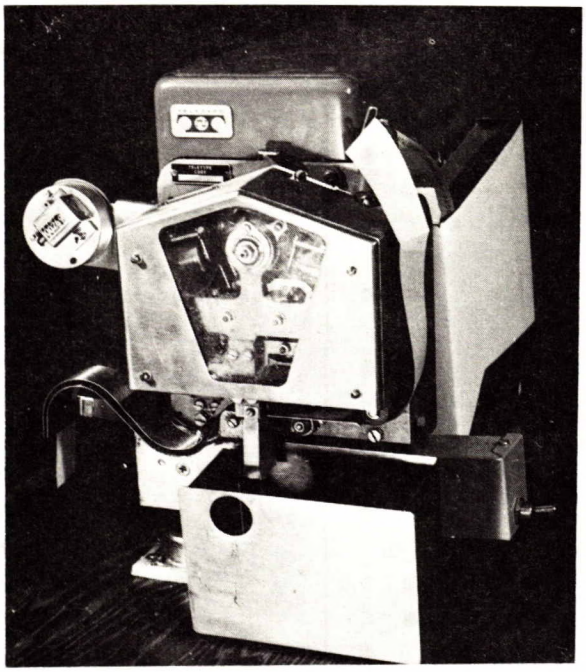


Figure 6. High-speed punch.

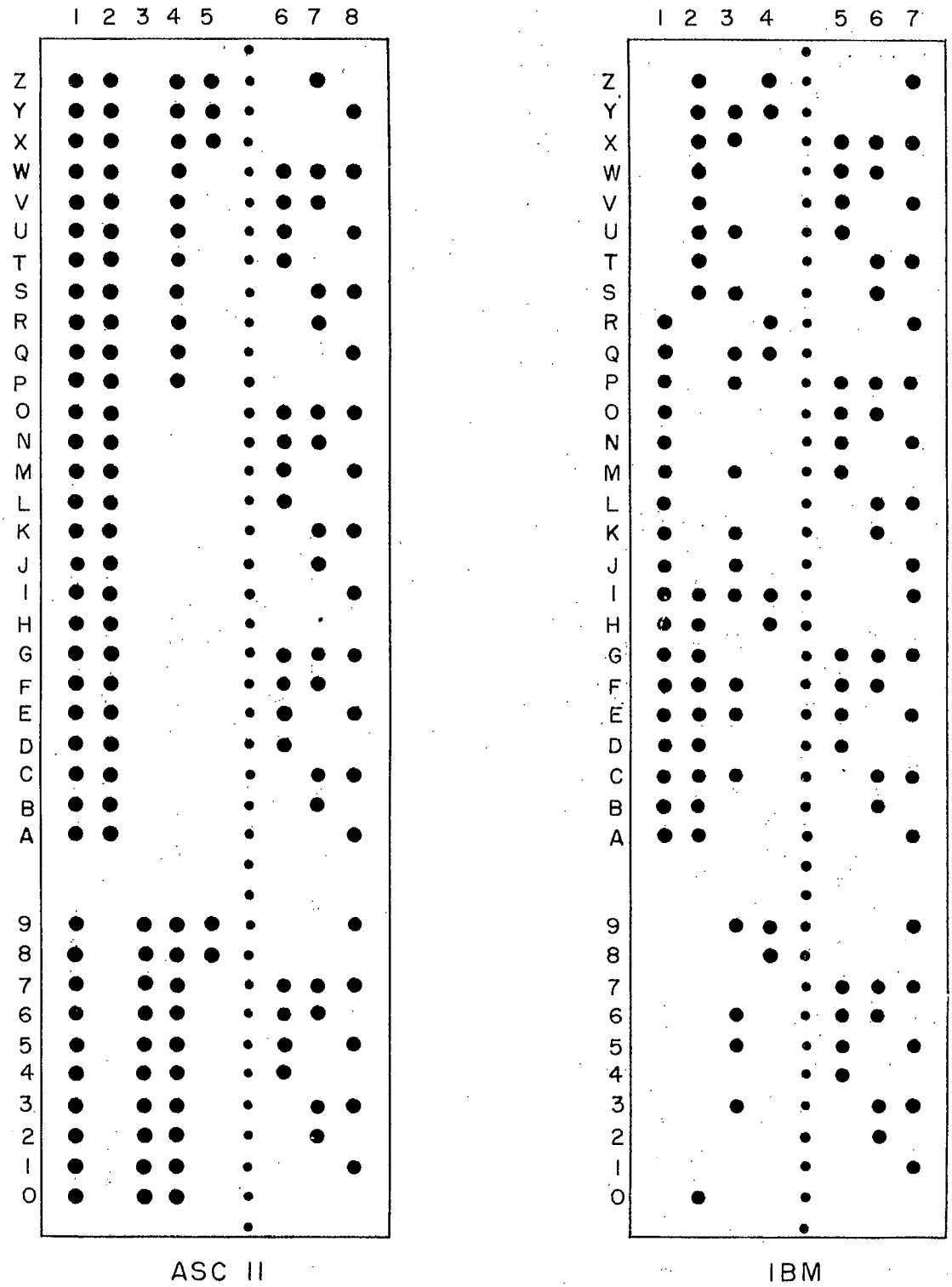


Figure 7. Alphabet in ASC II and IBM codes.

Data Break Channel

The data break is the feature of the PDP series of computers that makes them such a valuable laboratory tool. The data break stops the program at the end of the instruction being executed, enters the data from the experiment, and restarts the program at the next instruction.

Flags - ION

Since a computer can perform instructions faster than external devices can respond, "Flags" are used as a signal to the computer that a device is ready for further attention. The computer may be programmed to look for certain or all flags, but it can service only one request at a time. When the instruction ION (Interrupt On) is given, the computer will acknowledge sequentially any flags that occur. When the computer "sees" (senses) a flag, it completes the instruction, stores the address of the next instruction, services the device which signalled, then returns to the program. Thus more than one device may be in operation at one time. For example, the computer could sequentially display one point on the oscilloscope, type or punch a character, and execute one or more instructions of the program.

PART 2 -- PROGRAMMING

PROGRAM LANGUAGES AND INSTRUCTIONS

Programs for computers are written in a special language suited to the particular machine. The most popular computer language is Fortran, of which there are several varieties. Some of the less common ones are Cobol, Algol, Altac, Autocom, and Automath. In all cases the program is translated into a binary machine language by the computer.

The PDP computers use a language known as PAL or Program Assembly Language.

A program is a series of instructions which tell the computer what is required. There are six basic instructions, called "Memory Reference Instructions". These are coded as: AND, TAD, ISZ, DCA, JMS, and JMP, and are used as follows:

AND X — is a bit by bit multiplication between the contents of the AC and the contents in location X. The answer is in the AC and the contents of X are unchanged. This, in conjunction with other instructions, is useful for looking for specific numbers, or bits, by "Masking" the bits you are not interested in.

Example -					
	Contents of X	001	001	111	011
	AND	111	111	000	000
	The AC contents	001	001	000	000

TAD X — is the addition of the contents of the AC and the contents of location X. The AC will then contain the sum of its original contents and those of X.

Example -					
	Contents of AC are	000	010	010	010
	Contents of X are	000	100	100	100
	The instruction TAD X is given; the AC then contains	000	110	110	110

ISZ is increment and skip if X equals zero.

Example - The contents of X are -5.
The instruction ISZ is given.
One is added to -5, making it -4,
and when X equals zero the next
instruction is skipped.

This may be used in a programming technique, known as "Looping", which allows the programmer to have a given set of instructions performed a required number of times, in this case five.

DCA X is Deposit and Clear Accumulator.

Example - The contents of the AC are 000 010 010 010
The instruction DCA X is given.
The AC now contains 000 000 000 000
and location X contains 000 010 010 010

JMS X is Jump to Sub Routine X.

Example - The program arrives at the instruction stored in B, which is JMS X. The program jumps to location X, does these instructions, then returns to location B + 1.

JMP X is Jump to X.

Example - The program gets to the instruction stored in B, which is JMP X. The program jumps to location X and continues.

The next group of instructions are "Operate Instructions". They are listed below with their meanings:

NOP	No Operation
IAC	Increment Accumulator
RAR	Rotate Accumulator Right
RAL	Rotate Accumulator Left
RTR	Rotate Two Right
RTL	Rotate Two Left
CML	Complement Link
CMA	Complement Accumulator
CIA	Complement and Increment Accumulator

CLL	Clear Link
STL	Set Link
CLA	Clear Accumulator
STA	Set Accumulator
HLT	Halt
OR	Or With Switch Register
SKP	Skip Unconditional
SNL	Skip On Non Zero Link
SZL	Skip On Zero Link
SZA	Skip On Zero Accumulator
SNA	Skip On Non Zero Accumulator
SMA	Skip On Minus Accumulator
SPA	Skip On Positive Accumulator
CLA	Clear Accumulator

These instructions are built into the computer by the manufacturer and cannot be changed.

There are four instructions used only by the assembler, namely:

JMP . -X = Jump this point back X instructions.

JMP . +X = Jump this point ahead X instructions.
(X is the number of instructions to be jumped.)

* - sets the starting address of the program.

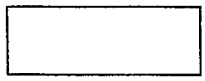
$ - signifies the end of the program.

These are the only instructions the computer obeys, and with them all programs are written.

There is one other symbolism to be described, and this is a "tag". A tag is equivalent to an address location, and is a group of one to six letters and/or numbers, which must always start with a letter and be followed with a comma. Tags should be meaningful, if possible, to help keep track of them in programming. For example, a positive number may be designated as PXX or P123, and a negative or minus number MXX or M123.

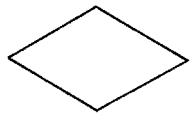
WRITING A PROGRAM

Before writing a program, a "FLOW CHART" should be drawn. This is a symbolism showing graphically the basic functions of the program. The symbols used are as follows:

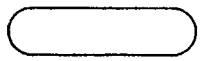


represents an operation

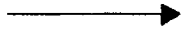
connects two program points



a decision is to be made



beginning or stopping point



direction of flow.

The various instructions and steps used in preparing a program will be described. The preparation of a short program which will add the first two numbers from a table designated as P 7, store away the sum, and halt the computer, is illustrated in Figure 8.

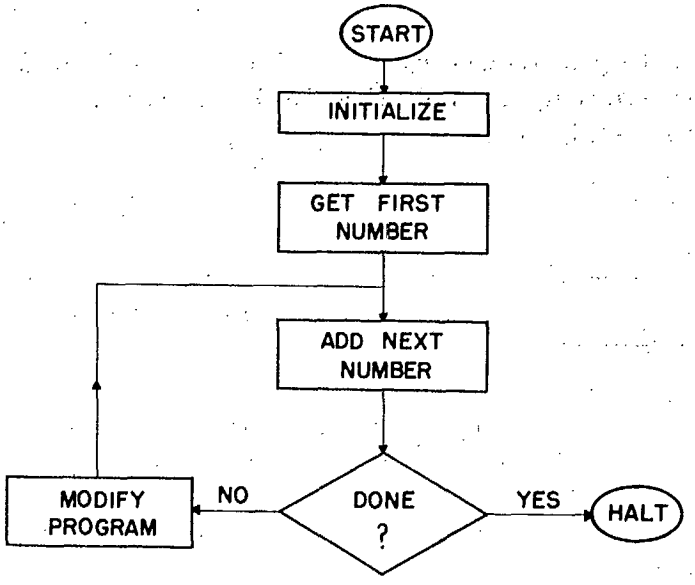


Figure 8. Flow chart.

The program may now be written, using the steps outlined in the flow chart.

```
*10
SUMM        P7
*400
            CLA
            TAD M2
            DCA CT
            TAD I SUMM
            ISZ CT
            JMP . -2
            DCA SUM
            TAD SUM
HLT
M2,         -2
```

```
P7,     7
        15
        20
CT,     0
SUM,    0
$
```

The analyses of the various steps of the program are as follows:

*10 - tells the computer to store the next instruction in location 10, in this case SUMM, P7.

SUMM, P7 This is stored in location 10, which is an Auto Index Register. When this location is called indirectly, it gives the address of the required information. In this case it gives SUMM, which is the address P7 whose contents are 7.

*400 The next instruction will be stored in location 400.

CLA "Clear the AC". That means put all zeros in the accumulator.

TAD M2 "Add M2". Put the contents of location whose address is M2 in the AC. M2 contains -2.

DCA CT "Deposit and clear the AC". Deposit -2 in the location CT and clear the AC.

TADI SUMM "Add the contents of the location given in SUMM". The location or address given in SUMM is P7; the contents of this address (7) are put into the AC.

ISZ CT "Increment and skip on zero". Add one to the contents of CT. CT contains -2 and with one added to it, it is now -1.

JMP . -2 "Jump this point minus 2". Jump the program back two locations to TAD I SUMM.

DCA SUM "Deposit and Clear the AC". At this point the computer has completed the addition of 7 and 15 and the answer is in the AC. The instruction DCA SUM tells the computer to store the total away in the location designated as SUM and clear the AC. Any time a result is stored the AC is automatically cleared. This is due to the limited set of instructions and cannot be changed. Therefore if one wishes to see the answer, or use it further, it must be called back by the next instruction.

TAD SUM	This puts the results back in the AC for further use and also leaves the same information in the Location SUM.
HLT	Tells the computer the task is completed and to stop.
M2, -2	M2 is a tag, and, as the rule says, the computer does not recognize a number alone as a tag -- it must be preceded by a letter. This instruction tells the computer to store -2 in the location the address of which is M2.
P7, 7 15 20	P7 is a tag, and the location designated by it contains 7. The 7, 15, 20 are a table, and the computer recognizes this and stores 15 in P7 + 1, 20 in P7 + 2 and would store as many numbers as there are in the table in this manner.
SUM, 0	This tag, SUM, sets aside a location and clears it, i.e., puts zeros in it.
$	All programs must end with a $ sign. This signals the computer that the end of the program has been reached and to begin assembling it.

EDITING AND ASSEMBLING

The program is typed on the ASR33 and stored in the memory of the computer. The computer is under control of a program known as the Symbolic Tape Editor or STE. This allows the programmer to call back any line or lines for correction. When the program has been typed and corrected it is punched out on paper tape. This is known as the Symbolic Tape.

The symbolic tape must be processed to produce a "binary tape", as the computer only understands information in this form. This is done by the computer under control of a program known as PAL, or Program Assembly Language.

The tape must be read into the memory of the computer twice. The first "pass" stores the program and types out certain error messages, i.e., unidentified addresses. The second pass assigns storage locations to the instructions and produces a paper tape of the program in machine language called the "Binary Tape".

A third pass may be made which lists the locations used in memory, the machine language equivalent of the instruction, and the instruction as written. The third pass as it is typed out is shown in Figure 9.

```
          *10
0010  0412  SUMM,        P7
          *400
0400  7200                CLA
0401  1211                TAD M2
0402  3215                DCA CT
0403  1410                TAD I SUMM
0404  2215                IZS CT
0405  5203                JMP .-2
0406  3216                DCA SUM
0407  1216                TAD SUM
0410  7402  HLT
0411  7776  M2,           -2
0412  0007  P7,           7
0413  0015                15
0414  0020                20
0415  0000  CT,           0
0416  0000  SUM,          0
```

Figure 9. Third pass, as typed by the ASR33.

To run the program, the binary tape is read into the memory with the SR set at 7777. The SR is then set to the starting address of the program (400) and the "LOAD" tab is depressed. The answer will be shown on the AC lights. But the answer is 26, and not 22 as one would expect. This is because the AC which does the mathematics can only operate in binary (zeros and ones) and the most convenient binary notation is octal. When an AC light is on, it represents a "one"; when it is off, it is a "zero". Each digit starting from the right represents a power of 2 greater than the one before it (Figure 10).

```
1    1    1    1    1    1    1    1    1
256  128  64   32   16   8    4    2    1
```

Figure 10. Binary Progression.

It can be readily seen that this is very cumbersome, so the digits have been grouped in sets of three:

```
111   111   111   111
421   421   421   421
```

If the three digits are all ones they total seven. By changing a one to a zero, any number from 0 to 7 may be obtained.

Number				
0	Binary	0 0 0		
	Decimal	0+ 0+ 0	=	0
1	Binary	0 0 1		
	Decimal	0+ 0+ 1	=	1
2	Binary	0 1 0		
	Decimal	0+ 2+ 0	=	2
3	Binary	0 1 1		
	Decimal	0+ 2+ 1	=	3
4	Binary	1 0 0		
	Decimal	4+ 0+ 0	=	4
5	Binary	1 0 1		
	Decimal	4+ 0+ 1	=	5
6	Binary	1 1 0		
	Decimal	4+ 2+ 0	=	6
7	Binary	1 1 1		
	Decimal	4+ 2+ 1	=	7

Eight is represented by the first digit in the next group of three. To convert a decimal number to octal (0_8), the decimal number is divided by 8, the remainder is put in the second column on the right, and the answer is placed in the third column on the right (see below).

15 converted to 0_8 - $15 \div 8$ = 1 with 7 over, or 17

56 " " " - $56 \div 8$ = 7 " 0 " or 70

57 " " " - $57 \div 8$ = 7 " 1 " or 71

The addition of 7 and 15 would be done as follows:

7 converted to octal,	$7 \div 8$	=	07 or, in binary,	000	111
15 " " "	$15 \div 8$	=	17 or, in binary,	001	111
			Total binary	010	110
			Total octal	2	6

To change 26 in 0_8 back to decimal, the procedure for changing to 0_8 is reversed:

6 is the remainder		6
2 is the answer	2 x 8 =	16
	Total	22

A programmer must always keep in mind that all numbers must be converted to octal and that in octal the numbers 8 and 9 do not exist.

ACKNOWLEDGEMENTS

I wish to express my thanks to Drs. R. H. Goodman and E. J. Gabe for their assistance in preparing this manuscript and to Mrs. V. Belanger for typing it. The guidance of P. E. Shannon, Mines Branch editor, is also gratefully acknowledged.

SUGGESTED READING

1. D. D. McCracken, "Digital Computer Programming" (John Wiley and Sons, Inc., New York), eighth printing, 1963, 253 pp. Library of Congress Catalog No. 57-8891.

= = = =

CAJ:(PES):vb