ANL-6888

Argonne National Laboratory

IPL-VC
A COMPUTER SYSTEM HAVING THE IPL-V INSTRUCTION SET

by

Donald Hodges

ANL-6888
Mathematics and
Computers
(TID-4500, 32nd Ed.)
AEC Research and
Development Report

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60440

IPL-VC
A COMPUTER SYSTEM HAVING THE IPL-V INSTRUCTION SET

by

Donald Hodges

Applied Mathematics Division

May 1964

Operated by The University of Chicago
under
Contract W-31-109-eng-38
with the
U. S. Atomic Energy Commission

TABLE OF CONTENTS

LIST OF FIGURES

IPL-VC
A COMPUTER SYSTEM HAVING THE IPL-V INSTRUCTION SET

by

Donald Hodges

ABSTRACT

Shaw et al. (Ref. 1) have described a possible hardware computer, called IPL-VI, which showed the important features such a machine should have for its IPL-V instructions, but the input/output and arithmetic instructions are not considered. However, with the arrival of the present-day module concept of arithmetic computer organization, a new possibility arises for the construction of an IPL-V hardware machine. To convert an arithmetical computer into an IPL-V system, it could be provided with a second processor which operates with certain basic IPL-V "J" processes as its instruction set. The second processor would require direct access to memory for its data and instructions in order to be able to operate at as fast a speed as the memory would allow. The instruction set of the second processor should be all the basic list operations. The remaining list operations could be built up as routines from these basic operations. All the arithmetic and input/output processes would be performed in the original arithmetic processor, with the List Processor taking care of any necessary list "bookkeeping." The necessary data and addresses would be communicated between the two processors by prelegislation of memory locations where the processors would find the relevant information when requested to perform an operation. Thus, there would have to be some means of transferring control back and forth between the two processors, presumably some form of interrupt system.

The advantages of such an approach to an IPL-V hardware system over building a completely new system are as follows:

1. A large, fast memory, as provided with present-day, large, arithmetic computer systems, would be available.

2. The arithmetic and input/output facilities would be immediately available and as fast as possible. In addition, the wealth of input/output devices on a modern, large, arithmetic computer installation is much greater than could be justified for an IPL-V computer itself.

3. An IPL-V simulator program would be available on the original computer. The actual IPL-V system presumably would not contain all the

excellent tracing, dumping, and snapshot procedures that are available in the simulator and are so useful for program debugging.

4. The cost of building a fairly simple List Processor would be considerably less than that of building a completely new computer.

For such an IPL-V computer to have universal use, its input should be program cards identical to those used for input to present-day IPL-V simulators, as specified in the IPL-V reference manual (Ref. 2). This would, however, require a new assembler to set up the program in storage in a suitable manner for the IPL-V system to be able to execute it.

I. INTRODUCTION

Most work in digital computer engineering has gone into the construction of faster processing units for the performance of normal, numerical arithmetic. However, some classes of problems do not use conventional arithmetic processors as efficiently (from the point of view of speed and, in general, the quantity of hardware) as they would use a special processor built to solve the particular problem. Because of its commercial applications, only the area of business data processing has received any serious attention. Another area which is arousing interest in scientific data processing is that of the heuristic, or gestalt, type problem. This may in general be classed as artificial intelligence, covering such areas as learning, pattern recognition, automatic translation, and problem solving; the game-playing machine is one application most easily understood by the layman. However, the same techniques developed for this type of programming now appear to have application in the organization of large collections of information, such as large business files and libraries; details are given in Ref. 3.

In arithmetic computers for the applications described here, it has been found convenient to define new programming languages which are classed as Symbol Manipulation Languages. To achieve a working machine obeying such languages, simulation programs of such machines have been written on conventional arithmetic computers. Five major languages have been developed, and these are reviewed in Refs. 4 and 5. They are COMIT, a system for helping in automatic translation (Ref. 6) and four general information-processing languages: LISP 1.5 (Refs. 7, 8, 9), FLPL (Ref. 10), IPL-V (Refs. 2, 11), and SLIP (Ref. 12). Although descriptions of the first two information languages exist, there is little published work involving their use. However, with IPL-V, not only the original designers of the language, but others have used the language as a working tool. The uses of the language are reviewed by A. Newell in Ref. 13. Other applications or adaptations of the principles are given in Ref. 3, and a use at Argonne

itself is given in Ref. 14. However, since all these applications use simulators on machines other than an IPL-V machine, they are slow, and it is difficult to gather information on their operation.

This report then presents a proposal for the realization of an information-processing computer having the IPL-V primitives as its instruction set. The reader is assumed to have a working knowledge of the IPL-V language as described in Refs. 2 and 11.

The principles of the equipment described here could well be applied to most of the present-day large computing systems, such as the AMD GUS system, IBM 360 series, Burroughs 35000, etc. However, these principles will be described in detail for the CDC-3600 system, as owned by the Argonne National Laboratory, because this would be the most suitable machine presently at the Laboratory for the attachment of such equipment. The other alternatives are the IBM-704 system and the Argonne-built GUS system. The former does not readily lend itself to the type of conversion to be described, while the latter is handicapped by small memory size, lack of suitable input-output equipment, and the absence of a written IPL-V simulator program. The convenience of having a simulator, even on a hardware IPL-V machine, is obvious to an IPL-V programmer. The simulator programs contain excellent tracing, dumping, and snapshot procedures. The reader is assumed to be familiar with the CDC-3600 as described in Ref. 15. The CDC-3600 has the following convenient features for converting it into an IPL-V hardware system (henceforth called IPL-VC):

1. A fast ($1\frac{1}{2}$-microsecond cycle time), 64,000-word, random-access memory.
2. Ease of attachment of equipment directly to the memory.
3. Working input-output equipment of all types.
4. A high-speed conventional arithmetic unit, with the provision of one list-processing instruction already built in, which is all that is needed in the arithmetic unit itself.
5. An IPL-V simulator program written for the CDC 1604, which is being converted to be suitable for the CDC-3600.

Several papers have been written proposing the hardware design of a list-processing computer. Reference 16 describes the possible modifications to the 7090 to make the Fortran List Processing process more efficient. Reference 17 is a memory organization for a list-processing computer with no special reference to a particular language. Reference 18 details the complete design of a list-processing machine which is in no real relationship to any of the previously described languages. References 3 and 19 describe the design of a processor which the authors believe represents an extension of the fundamental concepts of an associative

memory and the IPL-V language. The main point appears to be the use of variable-length data and a proposal for a similar piece of equipment, called ADAM, has been described in Ref. 20. Reference 1, written by the originators of the IPL-V system, describes a possible computer which they have called IPL-VI. However, they do not detail the arithmetic and input-output of IPL-VI. IPL-VC, now to be described, drew much inspiration from Ref. 1 (which appeared as early as 1958).

II. GENERAL DESCRIPTION

To convert an arithmetical computer into an IPL-VC system, a processor must be provided which operates with certain basic IPL-V Primitives as its instruction set. This processor requires direct access to memory for its data and instructions so that it can operate at as fast a speed as the memory will allow. The basic Primitives that this processor would perform are the operations on, and testing of, lists. The remaining list-processing Primitives would be executed in this processor, built up as routines from the basic Primitives. All the arithmetical and Input/Output Primitives would be performed in the original arithmetical processor, the List Processor taking care of any necessary "bookkeeping." The necessary data and addresses would be communicated between the two processors by prelegislation of where in memory the processors will find the relevant information when requested to perform a particular operation. Thus, there must be some means of transferring control back and forth between the two processors; it will be assumed to be some form of interrupt system. While the system may be sophisticated later, the initial assumption will be that only one or the other of the processors will be active during a run of a program; i.e., no concurrency of operation will take place.

Having described the system in general terms, we will now describe the IPL-VC system using a CDC-3600 as the original computer (a block diagram is shown in Fig. 1). The CDC-3600 memory system provides for the connection of five devices which are served in a first-come, first-served, cyclic order. In the Argonne CDC-3600 system, only two devices are connected (the 3604 Processor and a 3602 Communication module); thus the attachment of a new device having direct access to memory is immediately possible with no modification to the memory system. For the control communication between the two processors, one of the data channels could be used, and for programming convenience, the hardware could be

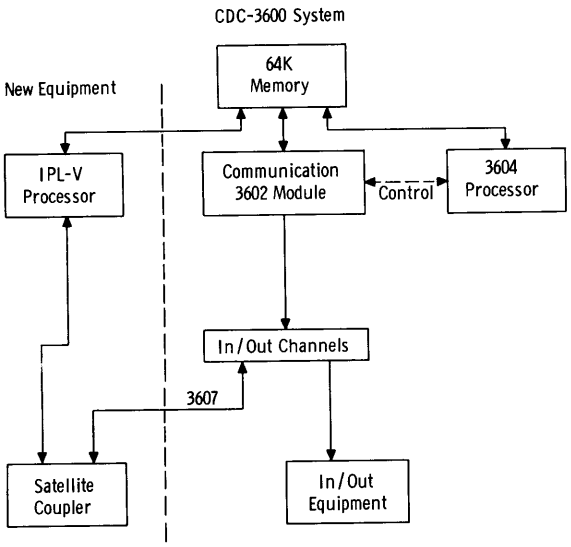


Fig. 1. Block Diagram of IPL-VC System

made to appear like a CDC type 3682 satellite coupler. However, the actual hardware would be much simpler since no actual data transfer would take place through the coupler, other than a six-bit piece of information which could be carried by the flags. Thus, when the list-processing processor wished the 3604 processor to do something, it would interrupt the 3604 processor by an interrupt from a particular equipment on a particular data channel. By performing a copy status instruction from that equipment, the 3604 would learn from the bit arrangement of the flags what it was being asked to do. When the 3604 had finished its operations, it would effectively interrupt the list-processing processor and tell it to continue, by setting one of the flags in the satellite coupler with a function instruction.

III. USERS' VIEW OF THE SYSTEM

The IPL-V programmer would code his program and have his cards punched identically to the ones used for input to present-day IPL-V simulators, as specified in Ref. 2. The assembler and control programs will probably have two levels of development. The first will be a two-stage process in which the assembler program will be run under the SCOPE monitor job system. The output of the assembler program would be a listing of the assembled program, with any errors indicated, and a binary magnetic tape, which would be the complete contents of the magnetic core memory for the initial state of the IPL-VC machine. This program would be run without the presence of the SCOPE monitor, and the first record would be autoloaded into the CDC-3600 memory. This record would then effectively bootstrap all the actual IPL-V code into memory and start the List Processor. The starting address would be placed by the 3604 in the current routine address list H1 (this list starts at a fixed location), and the 3604 would signal the List Processor (via the satellite coupler) to start. The 3604 would then halt, and the List Processor would have full usage of memory. When the List Processor required the attention of the 3604, either to perform a Primitive, or because the end of a program had been reached, it would interrupt the 3604, which would start running again. Normally the 3604 would perform what was asked of it before handing control back to the List Processor and halting itself again. Input and output of information to and from the outside world would take place just as with the IPL-V simulator.

The second program development would probably be to incorporate both the assembler and the running of the actual IPL-V program under one job in the SCOPE monitor system.

IV. WORD FORMAT AND INSTRUCTION SET

The formats of the types of words used throughout IPL-V expressions are shown in Fig. 2. The standard IPL word is a Symbol (some memory address), its type (given by P and Q), and a Link address which indicates where the next item in the list is to be found. Figure 2a shows the bit allocation that would be used in the CDC-3600. Eighteen bits are allowed for addressing, but only the lower 16 bits are needed to address a 64K memory. The upper two bits will be unused initially, but these bits may be used later to directly address auxiliary storage, perhaps after the manner of the Ferranti Atlas computer. Three bits each represent P and Q, and their labeling of the word type would be the same as in the IPL-V manual. The other six bits in a word are unused.

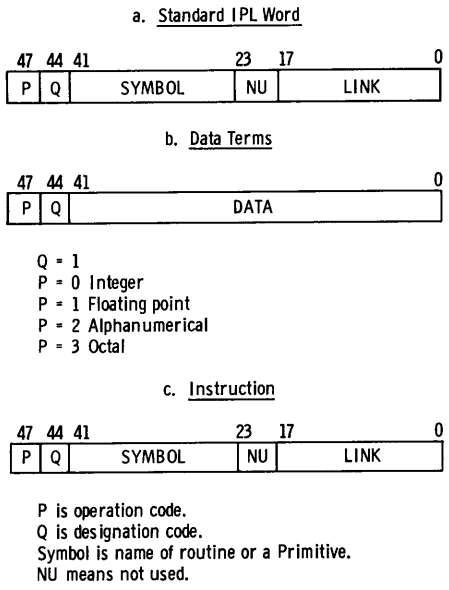


Fig. 2

Word Formats

A data word allows 39 bits for the data (either integer or floating point representation allowed), and uses the same P and Q bits as all other words to indicate the type of data contained in the word.

An instruction word contains the three-bit operation code (P), the three-bit designator (Q), and the 18-bit Symbol and Link fields. When P = 0, the Symbol part of the instruction will indicate a Primitive, if it is less than the address 128, and the name of a routine or a programmed primitive, if it is 128 or greater. In the IPL-VC hardware, the Primitives then are divided into the following three groups:

a) 0 to 63: Primitives actually executed by the hardware.

b) 64 to 127: Primitives wholly or partly executed by the 3604 arithmetic processor.

c) 128 upwards: The programmed Primitives and all programmer routines.

In the programmer's punched card input to the assembler, the IPL-V instructions would have their normal J notation as designated in the manual. Internal to the IPL-VC system, the instructions would in general have different numbers to make it easy to detect the type of Primitive. Figure 3 shows the way this decoding would take place within the 18 address bits of a Symbol.

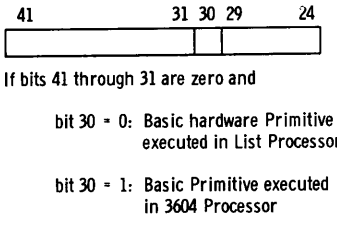


Fig. 3. Symbol Interpretation in an Instruction

The J-Primitives described in Ref. 2 are listed in Table I, together with how they would be executed. H denotes that these Primitives would be executed in the List Processor as a programmed routine, using those routines designated by H; C denotes execution of a Primitive in the 3604 processor. The double designation, CH, is used to indicate that some Primitives will need some list-processing "bookkeeping." A question mark following an entry indicates that there is doubt whether this will or should be executed by the IPL-VC system. The Primitives listed with a question mark fall into the following two categories:

a) Those that will not be allowed. No monitoring system would be built into the hardware so that the monitor Primitives (J147 to J149) would have no meaning.

b) Those that may be allowed. These are the generator housekeeping Primitives. An individual can program his own subroutine processes, and there is some doubt as to the need for the generator process.

Table I

HOW THE J PRIMITIVES ARE EXECUTED

J1	H	J2n	P	J75	H	J114	CH	J133	H	J152	C
J2	H	J3n	P	J76	P	J115	CH	J134	H	J153	C
J3	H	J4n	P	J77	P	J116	CH	J135	NU	J154	C
J4	H	J5n	P	J78	H	J117	CH	J136	H	J155	C
J5	H	J60	H	J79	H	J118	CH	J137	H	J156	C
J6	H	J61	P	J8n	P	J119	CH	J138	H	J157	C
J7	H	J62	P	J90	H	J120	C	J139	NU	J158	C
J8	H	J63	H	J9n(n≠0)	P	J121	CH	J140	C	J159	C
J9	H	J64	H	J100	?	J122	C	J141	C	J160	C
J10	P	J65	P	J101	?	J123	C	J142	C	J161	C
J11	P	J66	P	J102	?	J124	C	J143	C	J162	NU
J12	P	J67	P	J103	NU	J125	C	J144	C	J163	NU
J13	P	J68	H	J104	NU	J126	CH	J145	C	J164	NU
J14	P	J69	P	J105-8	?	J127	H	J146	C	J165	?
J15	P	J70	P	J109	NU	J128	CH	J147	?	J166	?
J16	C	J71	P	J110	CH	J129	CH	J148	?	J167	?
J17	?	J72	P	J111	CH	J130	H	J149	?	J168	NU
J18	?	J73	P	J112	CH	J131	H	J150	C	J169	NU
J19	?	J74	P	J113	CH	J132	H	J151	C	J170	?

H: Hardware executed; P: Program executed; C: 3604 executed; NU: Not used.

V. HARDWARE ORGANIZATION

The Register organization of the List Processor is shown in Fig. 4. This diagram indicates all the registers required, but no control is shown. There are two 38-bit registers, A and B; the former receives its input from memory, while the latter transmits its output to memory. Special single-address registers indicate the head address of the Communication List HOA, the head address of the Available Space List ASL, the Next Instruction Address NIA, and a Memory Register MR, which defines the address at which the current Read or Write operation is to take place. Two other short registers are required, one to indicate the current hardware Primitive that is being performed; the other would be a single-bit register which is really H5 and is used to encode and retain test results.

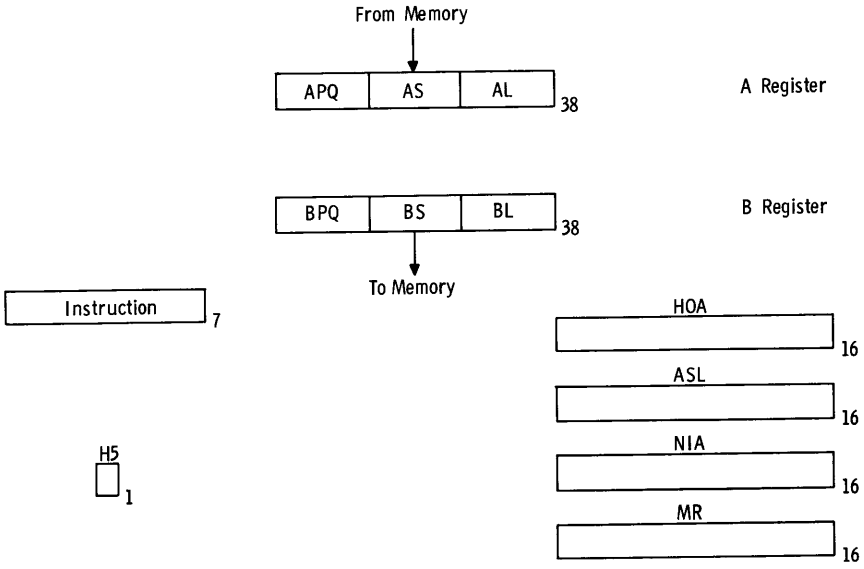


Fig. 4. Registers in IPL-VC List Processor

In practice, the hardware would execute the following functions, other than the basic hardware Primitives, as if they were Primitives:

1. The operations with $P \neq 0$.

2. The special case of P = 3 or 4, and the symbol is HO.

3. Restoring and Preserving of current routine address list H1.

Table II shows the actual instruction set of the list processor.

Table II

INSTRUCTION SET

Miscellaneous	Octal Operation Code	Memory Cycles for Execution*
Restore H1	10	5
Preserve H1	11	4
P = 1	01	2
P = 2	02	5
P = 3	03	5
P = 3HO, J8	12	2
P = 4	04	4
P = 4HO	13	2
P = 5	05	2
P = 6	06	3
P = 7	07	0

J No.	Octal Operation Code	Memory Cycles for Execution*
0	70	0
1	20	2+
2	21	3
3	71	0
4	72	0
5	73	0
6	22	5
7	74	0
9	23	3
60	24	2, 5, or 6
63	25	7
64	26	7 or 8
68	27	5 or 8
75	30	5
78	50	3
79	51	3
90	31	5
127	32	4
130	60	2
131	52	3
132	61	2
133	53	3
134	62	2
136	54	2
138	55	2
200	75	0
201	76	0
202	14	6

J No. Executed in 3604	Octal Operation Code
16	120
110	130
111	131
112	132
113	133
114	134
115	135
116	136
117	140
118	141
119	142
120	160
121	122
122	161
123	162
124	163
125	164
126	165
128	123
129	166
137	167
140	170
141	171
142	150
143	151
144	172
145	173
146	174
150	152
151	153
152	154
153	155
154	175
155	176
156	143
157	144
158	145
159	146
160	156
161	157
165	121
166	124
167	167

*Add one for instruction fetch cycle.
Add one cycle for Q = 1, and two cycles for Q = 2.
One cycle = 1-1/2 μsec.

VI. LOGICAL DESIGN OF THE IPL-V PROCESSOR

The basic principle of the hardware construction would be of the dc synchronous logic type. The complete sequence that would take place during the execution of an instruction is shown in Fig. 5. The operations in Fig. 5 are divided into the following two phases:

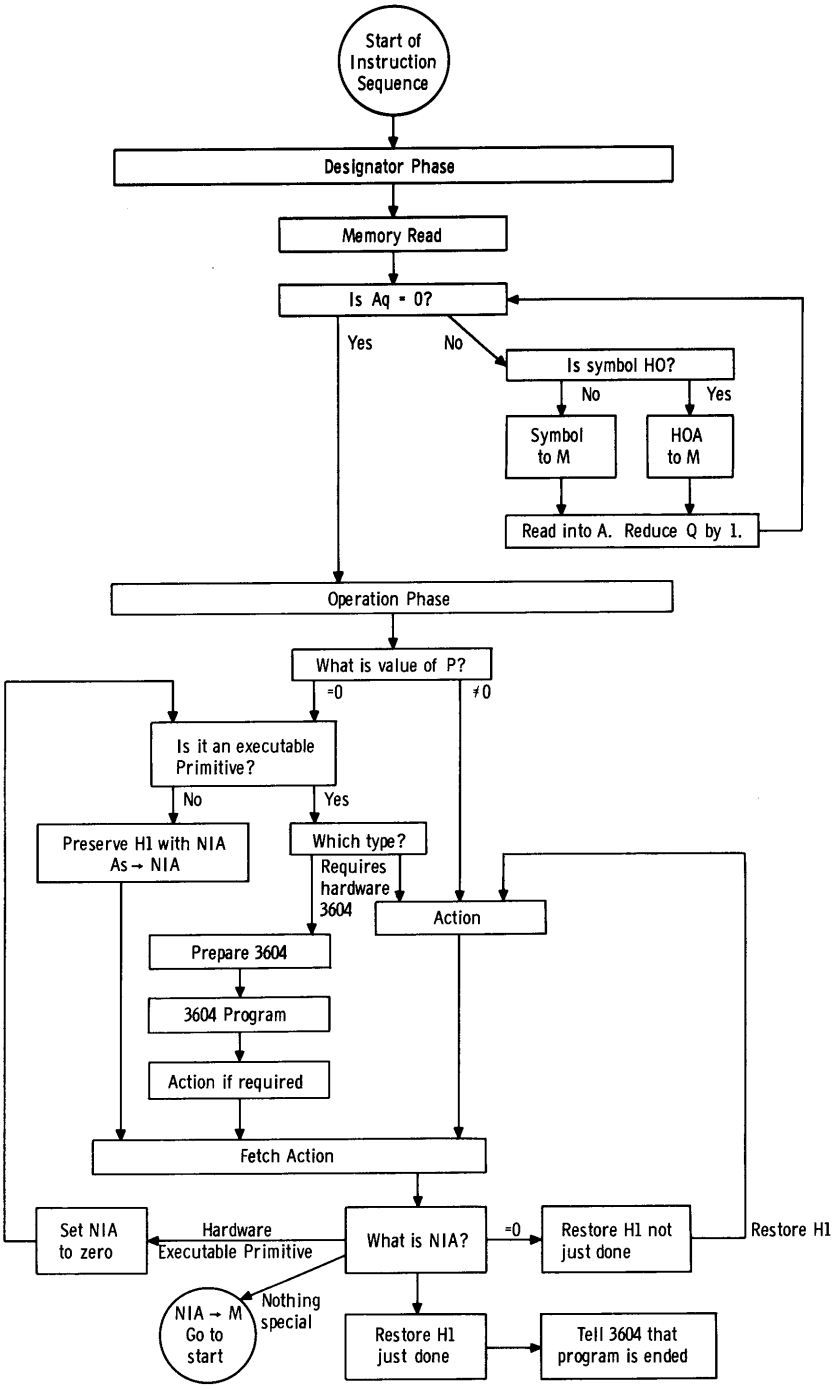


Fig. 5. Instruction Sequence Chart

A. Designator Phase

The IPL-V manual describes how Q in an instruction denotes the depth of indirect addressing used. Only values of 0, 1, and 2 are allowed (the other values of Q are concerned with monitoring and are meaningless in IPL-VC). In the Designator phase, memory is read using the contents of the Symbol portion of the instruction as the address, and Q is reduced by one each time until zero. If Q was initially zero, then no memory reference would be made and the instruction would pass into the Operation phase. Whether HO itself is being addressed must be detected; in that case, HOA is sent to MR. In addition, at the start of the Designator phase, the Link address of the instruction is sent to NIA.

B. Operation Phase

During the Operation phase, the hardware Primitives are actually executed, plus some special sequence of events which, to the hardware, look just like other instructions. The following is a list of what may occur during the Operation phase:

a) The operation called for by $P \neq 0$.

b) The execution of a hardware Primitive if $P = 0$.

c) The preparation for the 3604, the operation of the 3604, and any further list "bookkeeping" after the 3604 has finished, if the Primitive was an arithmetic or Input/Output type.

d) The preservation of H1, and the storing away of NIA, if a routine is named with $P = 0$.

e) The restoring of H1 and output to NIA, if a routine has just terminated.

A fetch action occurs at the end of each Operation phase.

The address in NIA would be examined and the appropriate action taken. The following possibilities arise:

a) NIA = 0, and a "Restore H1" instruction had just been performed. Then, the List Processor would halt and inform the 3604 that it had completed the program.

b) NIA = 0, and a "Restore H1" was not the last instruction. Then a "Restore H1" instruction would be set up, and the Operation phase would be re-entered.

c) NIA = a hardware Primitive. NIA would be transferred to the function register and then set to zero. The Operation phase would be re-entered.

d) NIA = no special case. Then NIA is sent to M, the next instruction is requested, and a new instruction sequence is entered.

The control sequencer would essentially be a two-pulse system in which the first of the pair of pulses has two functions, namely to move a byte or bytes of data around in the IPL-V processor and to decide the type of memory cycle that would be called for by the second pulse. The second pulse would do one of the following four things:

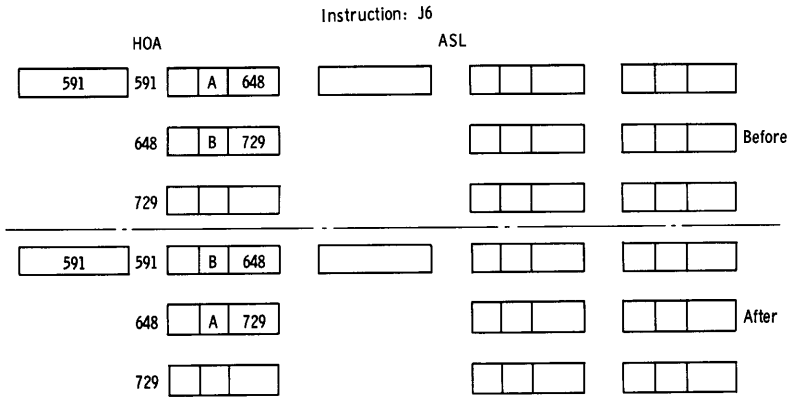
a) Request read memory action (memory data sent to Register A).

b) Request write memory action (contents of Register B written into memory).

c) Return to first pulse (A and B unchanged).

d) Terminate the current phase.

When the memory signalled that a or b was complete, then the first pulse would reappear; for c and d the first of the pulse pair would reappear some short fixed time after the no-memory-action second pulse occurred. A construction sheet which shows the Operation phase for the hardware-executed Primitive J6 [reverse (0) and (1)] is shown in Fig. 6.



Cycle	P1 Action	P2 Type
Start		
0	HOA → M	R
1	AL → M, A → B	R
2	AL → BL	W
3	HOA → M, AS → BS	R
4	AL → BL	W
5	~~~~	T
6		
7		
8		
9		
10		
11		

M	Register A			Register B		
	PQ	S	L	PQ	S	L
591		A	648			
648		B	729		A	648
648		B	729		A	729
591		A	648		B	729
591		A	648		B	648

R Request read memory action
W Request write memory action
N Return to first pulse
T Terminate Operation phase.

Fig. 6

Typical IPL-VC Operation Phase Sequence Sheet

VII. IMPLEMENTATION, COST, AND SPEED

Since the List Processor would be attached to the CDC-3600, the most convenient method of implementation would be to use the same printed-circuit card (PCB) as was used to build the CDC-3600 itself. The Control Data Corporation do not sell their individual PCB, so a realistic cost of the List Processor is difficult to obtain, but it would probably be between $20,000 and $100,000.

The speed of the execution of the hardware-executed Primitives is indicated in Table II; the footnote indicates the extra cycles to be added to represent the complete instruction time. In general, these speeds represent at least a tenfold increase in speed of execution of the J Primitives over that in the IPL-V Simulator on the CDC-3600.

VIII. CONCLUSIONS

Although a more useful and efficient information-processing language than IPL-V could be developed, it does represent a well-documented, working language from which much can be learned. The actual physical realization of an IPL-VC system would provide both an education and an insight into the area of information processing that would probably lead to more useful further developments than if such a system were only a paper design.

IX. ACKNOWLEDGMENT

The patience and explanations of Dr. W. Cowell of the IPL-V language during the exploratory period of the design of the IPL-VC system are gratefully acknowledged.

REFERENCES

1. J. C. Shaw, A. Newell, H. A. Simon, T. O. Ellis, A Command Structure for Complex Information Processing, Proc. Western Joint Comp. Conf. (1958).
2. A. Newell, Information Processing Language-V Manual, Prentice Hall, Inc. (1961).
3. N. S. Prywes and H. J. Gray, The Multi-List System for Real-Time Storage and Retrieval, IFIP 62 North American Holland (1963).
4. B. F. Green, Computer Languages for Symbol Manipulation, IRE Trans. on Electronic Computers, EC-10 (Dec. 1961).

5. D. G. Bobrow and B. Raphael, A Comparison of List-Processing Computer Languages, Communs. ACM 7, 4 (April 1964).

6. V. Yngve, A Programming Language for Mechanical Translation, Mech. Translation, 5, p. 25 (July 1958).

7. J. McCarthy, Recursive Functions of Symbolic Expressions and Their Computations by Machine, Communs. ACM, 3 (April 1960).

8. P. M. Woodward and D. P. Jenkins, Atoms and Lists, Computer Journal, 4 (April 1961).

9. J. McCarthy et al., LISP 1.5 Programmers Manual, MIT Press (Aug. 1962).

10. H. Gelernter, J. R. Hansen, and C. L. Gerberich, A Fortran - Compiled List-Processing Language, J. ACM, 7, No. 2 (April 1960).

11. A. Newell and F. M. Tonge, An Introduction to Information Processing Language-V, Communs. ACM, 3 (April 1960).

12. J. Weizenbaum, Symmetric List Processor, Communs. ACM 6, 9 (Sept 1963).

13. A. Newell, Learning, Generality and Problem Solving, IFIP 62, North American Holland (1963).

14. W. R. Cowell and M. C. Reed, A Checker-Playing Program in IPL-V, AMD Technical Memorandum No. 57 (Sept 1963).

15. Control Data 3600 Preliminary Reference Manual, Control Data Corporation, 501 Park Avenue, Minneapolis, Minnesota.

16. H. Gelernter, A Note on the System Requirements of a Digital Computer for the Manipulation of List Structures, IRE Trans. on Electronic Computers, EC10 (Sept 1961).

17. V. O. Muth and A. K. Scidmore, A Memory Organization for an Elementary List-Processing Computer, IRE Trans. on Electronic Computers, EC12 (June 1963).

18. J. C. Reynolds, A Proposal for a Micro-Programmed List Processor, AMD Tech. Memo. No. 69 (Feb 1964).

19. N. S. Prywes and S. Litwin, "The Multi-List Central Processor," Chapter 8, Computer Organization, Eds. A. A. Barum and M. A. Knapp, Spartan (1963).

20. A. P. Mullery, R. F. Scauer, and R. Rice, ADAM - A Problem-Oriented Symbol Processor, 1963 Spring Joint Computer Conference.