ARGONNE NATIONAL LABORATORY
Applied Mathematics Division

IPL-VC
A Proposal for a Computer System having the IPL-V Instruction Set

Donald Hodges

Technical Memorandum No. 66

January 1964

Contents

1. Introduction

In the field of digital computer engineering most work has gone into the construction of faster processing units for the performance of normal numerical arithmetic. However, there are classes of problems that 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 a special processor built to solve the particular problem. Only one area, namely business data processing, has received any serious attention, because of its commercial applications. Another area which is arousing interest in scientific data processing is that of the heuristic or gestalt type of 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 of the applications most easily understood by the layman. However, it now begins to appear that the same techniques developed for this type of programming have application in the organization of large collections of information such as large business files and libraries; details are given in ref. 13.

In using arithmetic computers for such applications it has been found convenient to define new programming languages which are classed under the name of Symbol Manipulation Languages. In order to achieve a working machine obeying such languages it has been necessary to write simulation programs of such machines on conventional arithmetic computers. There have been four major languages so far developed and these are reviewed in ref. 3. They are COMIT, a system for helping in automatic translation (ref. 14) and three general information processing languages, LISP 1.5 (refs. 5,6,7) FLPL (ref. 4) and IPL-V (refs. 1,2). 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. A review of the uses of the language and references to their papers is given by A. Newell in ref. 15. Other applications or adaptations of the principles are given in the previously mentioned ref. 13, and a use at Argonne itself is given in ref. 16. However, since all these applications use simulators on other machines (than an IPL-V machine), they are slow and it is also 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. It will be assumed that the reader has a working knowledge of the IPL-V language as described in refs. 1,2.

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 7000 series, UNIVAC-1107, etc., but they 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 very obvious to an IPL-V programmer. The simulator programs containing excellent tracing, dumping, and snap shot procedures. It will be assumed that the reader is familiar with the CDC-3600 as described in ref. 12. The CDC-3600 has the following convenient features for converting it into an IPL-V hardware system, henceforth called IPL-VC. They are:

1. A fast (1-1/2 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 in the process of being converted to be suitable for the CDC-3600.

Several papers have been written on the possibility of the hardware realization of a list processing computer. Reference 8 describes the possible modifications to the 7090 in order to make the Fortran List Processing process more efficient. Reference 9 is a memory organization for a list processing computer with no special reference to a particular language. Reference 10 details a complete design of a list processing machine which is in no real relationship to any of the previously described languages. References 13,17,8 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 there has been a proposal for a similar piece of equipment called "ADAM", described in ref. 19. Reference 11, written by the original authors of the IPL-V system, describes a possible hardware

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 this last paper, and it is interesting to note that it appeared as early as 1958.

2. General Description

To convert an arithmetical computer into an IPL-VC system it is necessary to provide a processor which operates with certain basic IPL-V primitives as its instruction set. This processor requires direct access to memory for its data and instructions in order to be able to operate at as fast a speed as the memory will allow. The basic primmtives that this processor would perform are the operations on lists and the testing of them. The remaining list processing primitives would be executed in this processor, built up as routines from the basic primitives. All of the arithmetical and input/output primitives would be performed in the original arithmetical processor, with the list processor taking care of any "bookkeeping" necessary. 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 it may be possible to be sophisticated later, it will initially be assumed 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 schematic diagram is shown in fig. 1. The CDC-3600 memory system allows 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 at all. For the control communication between the two processors one of the data channels could be used, and for programming convenience the hardware could be made to appear like a CDC type 3682 "satellite coupler". However, the actual hardware would be much simpler as in this case no actual data transfer would take place through the coupler, other than a 6-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 of one of the flags in the satellite coupler with a function instruction.

IPL-VC Schematic of Whole System

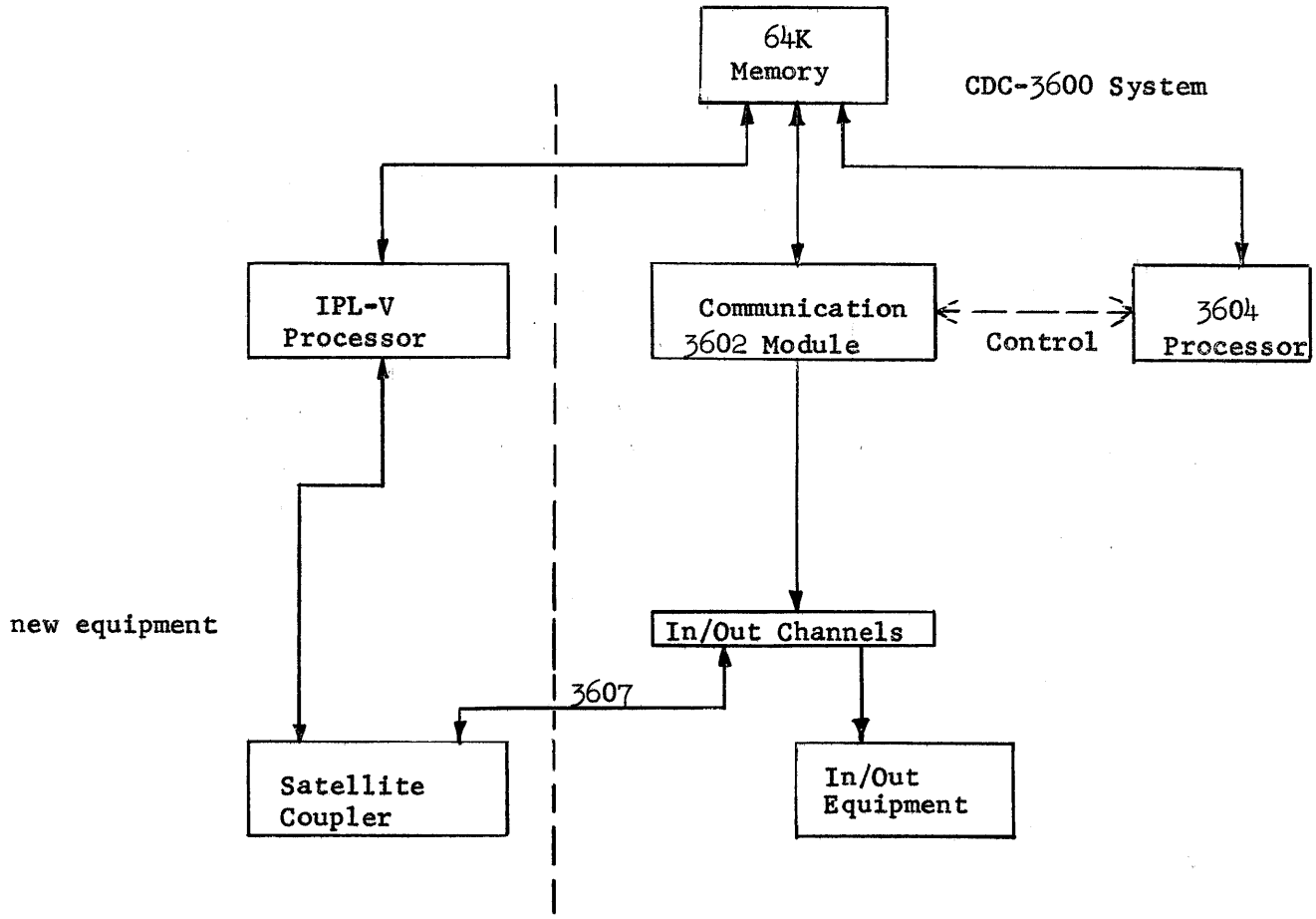


Fig. 1

3. Users' View of the System

The IPL-V programmer would code his program and have cards punched identical to the ones used for input to present day IPL-V computer simulators as specified in ref. 2. For debugging and check-out the programmer would probably wish to use the CDC-3600 simulator of IPL-V in order to allow himself full use of its monitor system. However, when he was satisfied the program was satisfactory then the IPL-VC system could be used. Initially an Assembler program would be in the CDC-3600 to read in the program and adjust it to be suitable for the IPL-VC system. The starting address would be placed by the 3604 in the current routine address list H1 (this list starts at a fixed location) and would then signal the list processor to start via the satellite coupler. 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.

4. 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. Fig. 2a shows the bit allocation that would be used in the CDC-3600. Eighteen bits are allowed for addressing, whereas only the lower 16 bits are needed to address a 64K memory. The upper two bits will initially be unused but it may be possible later to use these bits 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 6 bits in a word are unused.

A data word allows 42 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 3-bit operation code (P), the 3-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 three groups, namely:

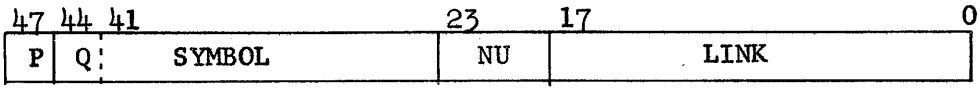
a) 0 to 63. Primitives actually executed by the hardware.

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

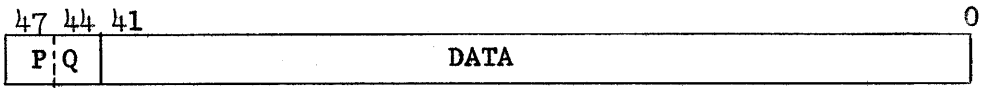
c) 128 upwards. The programmed primitives and all programmer routines.

In the Programmers Punched Card input to the Assembler the IPL-V instructions would have their normal J notation as designated in the manual, but internal to the IPL-VC system they would in general have different instruction numbers so as to make it easy to detect the type of primitive. Figure 3 shows the way in which this decoding would take place within the 18 address bits of a Symbol.

A list of the J-Primitives as described in ref. 2 is shown in table 1 together with how they would be realized. H denotes direct hardware execution in the List Processor, P 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. However, some of these primitives will need some list processing "bookkeeping", hence



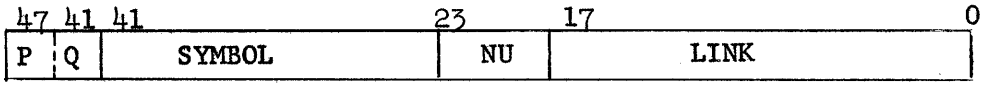
a. Standard IPL Word



Q = 1

P = 0 Integer
P = 1 Floating Point
P = 2 Alphanumerical
P = 3 Octal

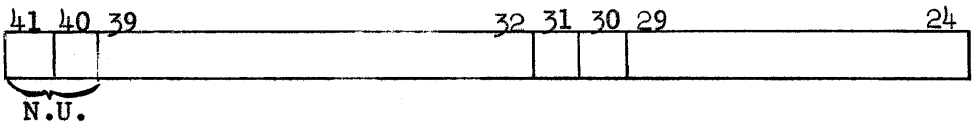
b. Data Terms



P is operation code
Q is designation code
SYMBOL is name of Routine or a Primitive

c. Instruction

Fig. 2 Word Formats



: 39 thru 32 are zero and

bit 31	bit 30	
0	0	: Basic Hardware Primitive
0	1	: Arithmetic or Input Output Primitive
1	0	: Programmed Primitive
1	1	: Programmed Primitive

Fig. 3 Symbol Interpretation in an Instruction

the double designation CH. A question mark following an entry indicates that there is some doubt whether these particular Primitives will or should be executed by the IPL-VC system. There are three classes:

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. One can program one's own Sub-routine processes and there is some doubt as to the need for the Generator Process.

c) Those that may be replaced by new Primitives, in particular the Input/Output instructions. It is more than likely that some new primitives will be introduced into the CDC-3600 IPL-V Simulator program and it would then be these that would be executed.

H - Hardware
P - Programmed
C - in 3604 A.U.

Table 1

How the J Primitives are Interpreted

J 1	H	68	H	J120	C	J150	?
2	H	69	P	1	CH	1	?
3	H	J 70	P	2	C	2	?
4	H	71	P	3	C	3	?
5	H	72	P	4	C	4	?
6	H	73	P	5	C	5	?
7	H	74	P	6	CH	6	?
8	H	75	H	7	H	7	?
9	H	76	P	8	CH	8	?
J10	P	77	P	9	CH	9	?
11	P	78	H	J130	H	J160	?
12	P	79	H	1	H	1	?
13	P	J 8n	P	2	H	2	NU
14	P	J 90	H	3	H	3	NU
15	P	J 9n(n≠o)	P	4	H	4	NU
16	C	J100	?	5	NU	5	?
17	?	101	?	6	H	6	?
18	?	102	?	7	H	7	?
19	?	103	NU	8	H	8	NU
J 2n	P	104	NU	9	NU	9	NU
J 3n	P	105-8	?	J140	?	J170	?
J 4n	P	109	NU	1	?		
J 5n	P	J110	CH	2	?		
J60	H	11	CH	3	?		
61	P	12	CH	4	?		
62	P	13	CH	5	?		
63	H	14	CH	6	?		
64	H	15	CH	7	?		
65	P	16	CH	8	?		
66	P	17	CH	9	?		
67	P	18	CH				
		19	CH				

5. Hardware Organization

The Register organization of the IPL-V processor is shown in figure 4. This diagram indicates all of the registers required, but no control is shown. There are two full length 48-bit registers, A and B; the former receives its input from memory while the latter always transmits its output to memory. Special single address registers are used to indicate the head address of the Communication List H0, the head address of the Available Space List H2, the Next Instruction Address and a Memory Register 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.

In actual practice the hardware would execute certain functions, other than the basic hardware primitives, as if indeed they were primitives. They are as follows:

1. The operations with $P \neq 0$.
2. The special cases of $P = 3$ or 4 and the symbol is H0.
3. Restoring and Preserving of Current Routine address list H1.

Table 2 shows the actual instruction set of the list processor.

Registers in IPL-VC, List Processor

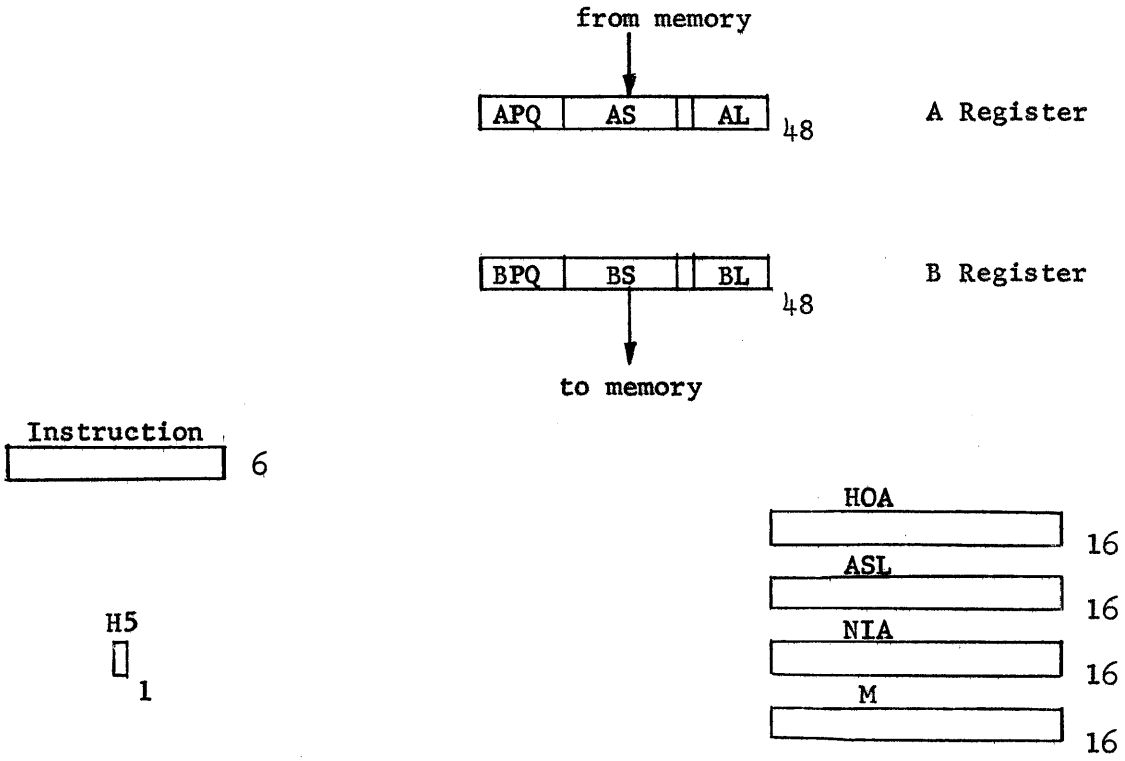


Fig. 4

Table 2: Instruction Set

Misc	Memory Cycles Execution*	J No.	Memory Cycle Operation*	J No. Executed in 3604
Restore H1	5	0	-	16
Preserve H1	4	1	2+	110
P = 1	2	2	3	111
P = 2	5	3	-	112
P = 3	5	4	-	113
P = 3HO	2	5	-	114
J8		6	5	115
P = 4	4	7	-	116
P = 4HO	2	9	3	117
P = 5	2	60	2 or 5 or 6	118
P = 6	3	63	7	119
P = 7	-	64	7 or 8	120
		68	5 or 8	121
		75	5	122
		78	3	123
		79	3	124
		90	5	125
		127	4	126
		130	2	128
		131	3	129
		132	2	
		133	3	
		134	2	
		136	2	
		137	6	
		138	2	

*NOTE: Add 1 cycle for Instruction Fetch
Add 1 cycle for Q = 1 and 2 cycles for Q = 2
1 cycle = 1-1/2u secs.

6. Logical Design of the IPL-V Processor

The basic principle of the hardware construction would be of the "d.c. synchronous logic" type. The complete sequence that would take place during the execution of an instruction is shown in fig. 5. It will be noticed that the operations in fig. 5 are divided into three phases, which are:

A. Designator Phase

The IPL-V manual details how "Q" in an instruction denotes the depth of indirect addressing used. Only values of 0,1,2 are allowed (the other values of Q are concerned with monitoring and are meaningless in IPL-VC) and 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 1 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. It will be noted that it must be detected whether HO itself is being addressed; in that case it is HOA that is sent to the Memory Address register. Also, at the start of the designator phase the link address of the instruction is sent to the next instruction address register (NIA).

B. Operation Phase

It is during this phase that 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 it was an arithmetic or Input/Output hardware primitive.
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.

C. Fetch 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 it had completed the program.

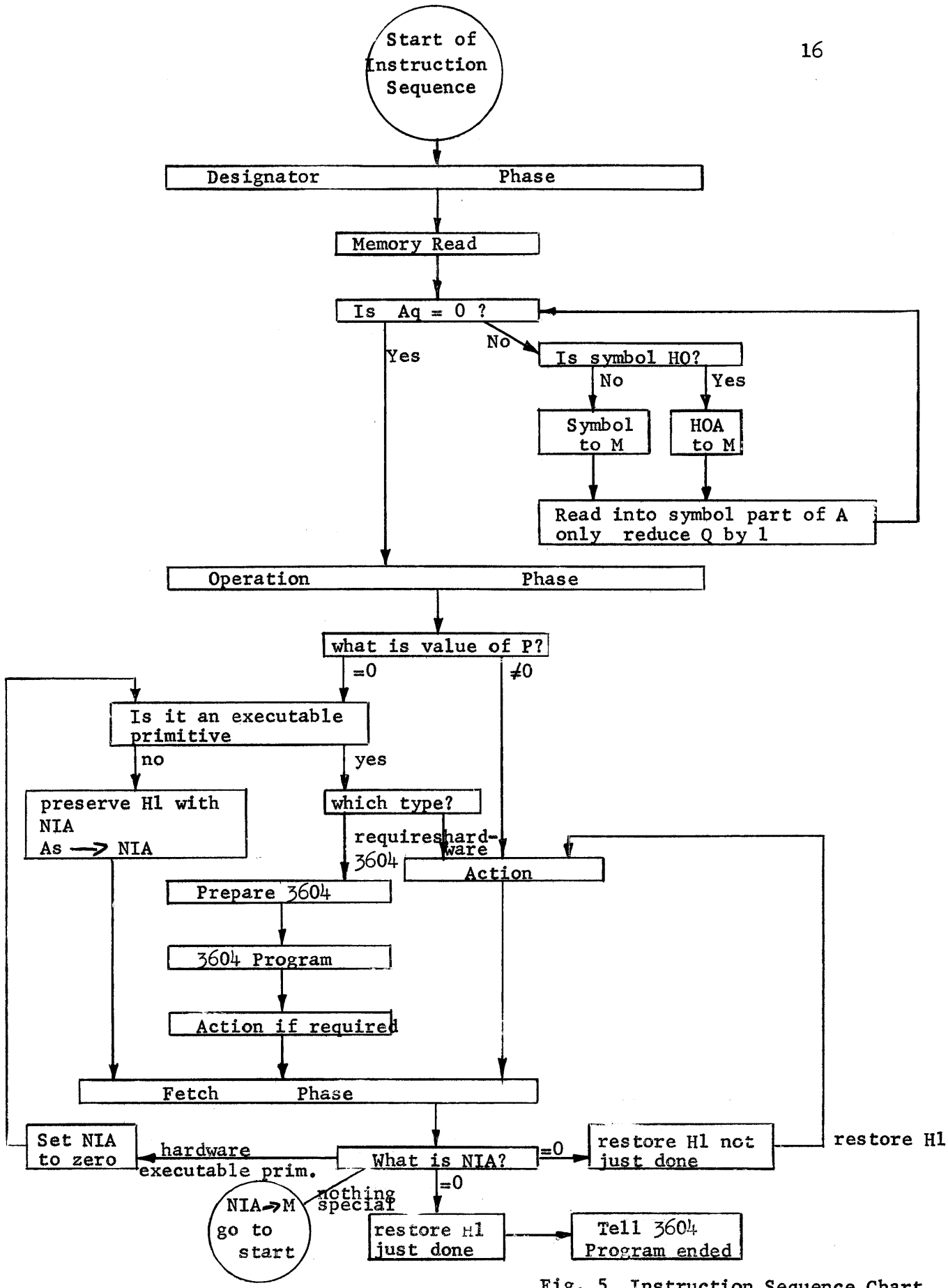


Fig. 5 Instruction Sequence Chart

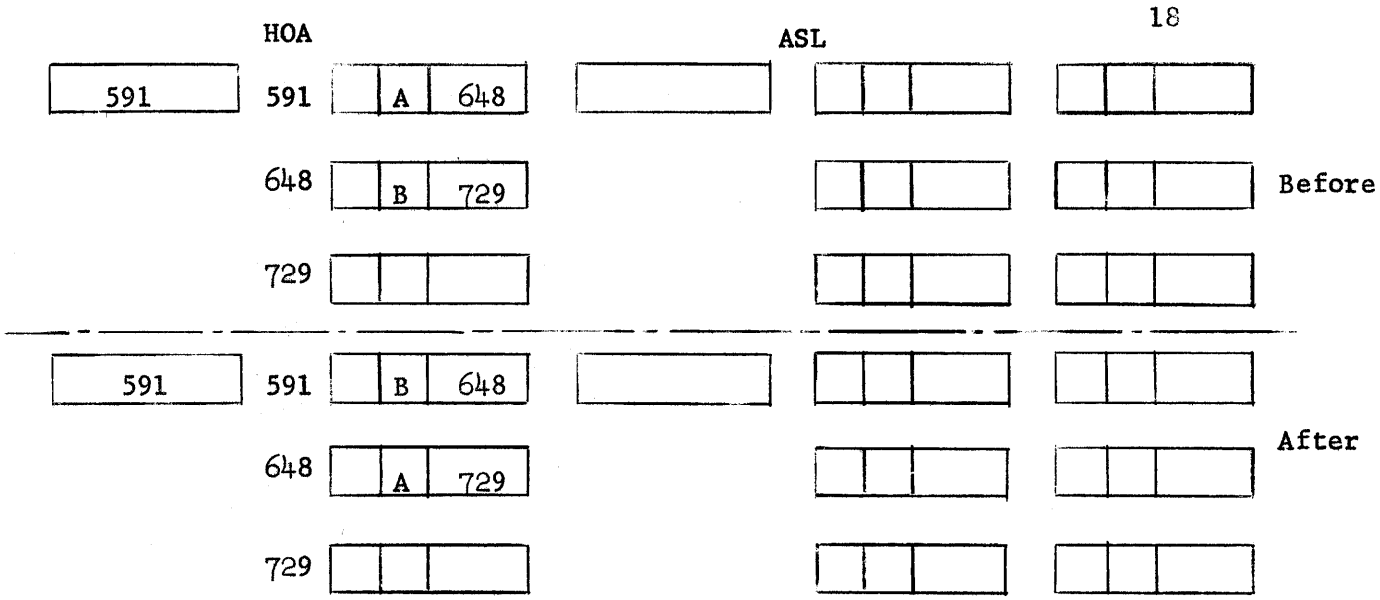
b) NIA=0 and a "restore H1" was not the last instruction. Then a "restore H1" instruction would be set up and the Operation phase would be re-entered.
c) NIA = a hardware primitive. NIA would be transferred to the function register and then set to zero. The Operation phase would be re-entered.
d) NIA = no special case, then NIA is sent to M, the next instruction requested and a new instruction sequence entered.

The control sequencer would essentially be a two-pulse system with the first of the pair of pulses having 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 of the two pulses would in fact do one of 4 things only, as follows:

a) Request Read Memory Action (Memory data sent to Register A).
b) Request Write Memory Action (Contents of Register B written into memory).
c) No Memory Action (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 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.

Instruction: J6



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

R read memory cycle
W write memory cycle
N no action
T terminate operation phase

Fig. 6 IPL-VC Operation Cycle Sequence Sheet

7. Implementation, Cost and Speed

There are four choices in the type of hardware that might be used to build the extra list processing processor. They are:

a) Buying PCB's made by a commercial firm.
b) Buying boards and rack mounting equipment from CDC, the same as that used to build the 3600 computer itself.
c) Use of the AMD Computer Engineering Section's standard NOR type circuitry.
d) Although not normally considered from an economical viewpoint, to build the system in Micro-logic elements in order that the group may obtain experience in building actual systems.

A rough count of the number of standard NORS that would be needed to implement the system is 2500 to 3000, which at $10. per NOR (this includes power supplies and cabinets) would run from $25,000. to $30,000. If outside boards were used (either a or b) then the cost could range from $20,000. to $50,000. Using Micro-logic elements at this time would probably double the cost of the system, i.e., it would be in the $40,000. to $100,000. area. It is of course not practical to quote the costing any more accurately until more detailed design is performed.

The speed of the execution of the hardware-executed primitives is indicated in table 2; the footnotes should be consulted as to the extra cycles to be added to represent the complete instruction time.

One further increase in speed at the expense of more hardware that is being investigated is the implementation of HO as an actual Hardware Stack. This would probably involve some limitation to the depth of HO (probably to 8 entries)

8. Conclusions

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

9. Acknowledgement

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 very gratefully acknowledged.

10. References

1) An Introduction to Information Processing Language-V A. Newell and F. M. Tonge, Communications of the ACM Vol. 3, April 1960.

2) Information Processing Language-V Manual, Allan Newell, Prentice Hall Inc., 1961.

3) Computer Languages for Symbol Manipulation, Bert F. Green, IRE Trans on Electronic Computers Vol. EC-10, Dec. 1961.

4) A Fortran - Compiled List-Processing Language, H. Gelernter, J. R. Hansen and C. L. Gerberich, Journal of the ACM Vol. 7 No. 2, April 1960.

5) Recursive Functions of Symbolic Expressions and Their Computations by Machine, J. McCarthy, Comm. of the ACM Vol. 3, April 1960.

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

7) LISP 1.5 Programmers Manual, J. McCarthy and Als., MIT Press, August 1962.

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

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

10) Preliminary Design of a Linked List Computer, C. C. Foster Internal Report, University of Michigan, May 1963.

11) A Command Structure for Complex Information Processing, J. C. Shaw, A. Newell, H. A. Simon, T. O. Ellis, Proc. Western Joint Comp. Conf., 1958.

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

13) The Multi-List System for Real-Time Storage and Retrieval, N. S. Prywes and H. J. Gray, IFIP 62 North American Holland, 1963.

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

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

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

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

18) A Growing Tree for Descriptor Language Translation, W. I. Landauer and N. S. Prywes, "Symbolic Languages in Data Processing", Gordon and Brench 1962.

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