The Charles Stark Draper Laboratory, Inc.
68 Albany Street, Cambridge, Massachusetts 02139 Telephone (617) 258-

2 July 1976

Professor Donald E. Knuth
Computer Science Department
Stanford University
Stanford, California 94305

Dear Professor Knuth:

I was delighted to receive the paper you sent me recently, and hope that my reply is timely since I was on a trip when your paper arrived. Although I have not yet read the entire paper, it is extremely interesting to see our work in the perspective of other contemporary activities. I have of course read the "Laning and Zierler" section meticulously, and find it at least as accurate as my memory. I have two typos to report: on page 53, paragraph 1, last line, I cannot locate any reference "GO 54" in your references at the end, and on page 56 your quote from my 1965 letter reads rather awkwardly after "Whirlwind" (i.e., it looks as though something was opaqued out and not rewritten). Otherwise, it's great!

As you can see from the enclosures, I have been motivated by your very kind letter to attempt to expose some of the history of Draper Lab computing before it is too late. I think we have probably made many interesting contributions, despite the fact that these are not widely known in computing circles. I think you also need an explanation as to why our work has not had greater impact on the community at large.

First of all, let me explain that the Charles Stark Draper Laboratory (CSDL) is the same organization that used to be known as the MIT Instrumentation Laboratory; since 1973 we have existed as a non-profit corporation independent of MIT. I will use the term "CSDL" because it is shorter to write. At the times of interest historically, CSDL was a departmental laboratory under the MIT Aeronautics Department, and despite the fact that we were an academic (and teaching) laboratory, computer science was merely a tool for the accomplishment of hard-nosed engineering objectives, rather than an end in itself. Our main product has always been instrument systems, generally in fields such as fire control and inertial guidance which rely heavily on gyroscopic instruments. Our products include flight hardware for guidance of ballistic missiles (Thor, Polaris, Poseidon) and space vehicles (Apollo) as well as numerous smaller projects. During the 1950's and early 1960's, a rather small group of colleagues and I were not only programming computers, but also developing guidance theory, designing computer hardware, and generally pressing on with the task of getting effective computation into flight hardware. This is at least one clue to our lack of publications in computer science; we were very much project oriented.

A second factor is the fact that during the early development of Apollo (1962-1967) we strayed from the beaten path by relying on the Honeywell H800, and later the H1800, as our primary computers. To compound our deviance, we wrote our own operating system. Thus during the period when MAC was beginning to become a mature language, there was nobody else who could use it. Only after we acquired a couple of 360/75's in 1967-1968 and converted back to the IBM fold was there any real purpose to publicizing our work, apart from gaining credit. Various of us felt badly about all this from time to time (when we were not too busy with project deadlines) and eventually Jim Miller, in 1970, was moved to write the overview of MAC (Enclosure 3,--my co-authorship is honorary), which was presented at a NASA symposium as I remember it.

Returning to the history of computing, I would like to clarify one issue. Sometime before the 1954 conference, Charlie Adams called me and said he was presenting a paper on the Whirlwind "Summer Session" system, and asked if I would mind if he mentioned the Laning-Zierler work also. I of course assented, and heard nothing more from it. It was not until ten or fifteen years later that I learned I had "co-authored" this paper. What makes this interesting is that I have had to repudiate a couple of times since a completely undeserved credit as the "inventor of the operating system". Not so!

We did do other things, however, which may or may not have been pioneering but which certainly were interesting. E.g., circa 1957-1959 Phil Hankins, Charlie Werner, and I used the IBM 650 to do completely automated digital logic layout, including not only automated partitioning of logic amongst logical "sticks" (they would have been printed circuit boards these days) but also generation of explicit wiring diagrams on an IBM 402 or 407 printer. One experimental Polaris guidance computer was actually built from such diagrams; however, the logic utilization was only 50% to 60% efficient as I remember it. We always suspected a basic program bug, since our (Monte Carlo) generation procedure produced poorer computers the harder we tried. As a second instance, around 1961-1962 I wrote the interrupt-driven, asynchronous, multiprogrammed executive system for the Apollo guidance computer, which I am told was essentially unchanged from the time I wrote it. This is the one which complained audibly of a computer overload during the final seconds of the first Lunar landing, because an astronaut had left a switch in the wrong position, and which successfully by-passed low priority tasks to accomplish the main mission successfully.

The early history of computing at CSDL, after the days of the CPC, started in 1955 with an interpretive system for the 650 called MITILAC (References 1, 2). MITILAC featured interpretive floating point arithmetic, and included software for solution to differential equations as well as a few standard functional subroutines. This was followed in 1956 by BALITAC (Reference 3), which was an assembler supplemented by operation codes for functional and vector-matrix operations; it also permitted indexing of basic 650 instructions. The lead role in these early developments was played by R. H. Battin, who was also responsible for proposing (probably in the summer of 1956) the 3-line format subsequently used in MAC. This format suggestion was passed on to IBM by Battin, but was turned down allegedly because of keypunching difficulties.

The remainder of this letter I will devote to rather random commentary on the enclosures. These include two documents relating to the Whirlwind compiler, two documents describing MAC essentially as it exists today for the IBM 360/370 (actually, we are acquiring an Amdahl 470 V/6 this month), a 1961 650/7090 MAC manual, two sample 650 MAC programs and an early memo. Unfortunately the 650 MAC compiler itself has long since disappeared.

Enclosure 1: Original Whirlwind Compiler

Here is what I believe to be the prototype Whirlwind algebraic compiler, complete with coffee stains plus a 23 year accumulation of dust. My knowledge of Whirlwind and of the assembler used at that time has become somewhat cloudy over the years; however, I might be able to make a few useful comments.

First, I believe the routine was the one designed to run in the 1024 word version of Whirlwind; as I remember it, the program left about 75 words available to store user program, using the rest for the compiler and executive functions. I did not then, and do not now, understand the global program structure including I/O and the interpretive floating point arithmetic routines. These were Zierler's responsibility, whereas I concentrated on the algebraic part per se. The floating point interpreter was called via "SP ax " or, apparently, "SP 852", and the interpretation of this same symbol by the interpreter caused it to exit from interpretive mode. The so-called "MRA" referenced in the annotation was the "multiple register accumulator" used by the floating point interpreter; unfortunately, I don't remember the data format used.

The program mostly generated a sequence of interpreted subroutine calls to a collection of elementary routines (cf. page 18) to operate on a dummy double precision accumulator. Nested parentheses (and " =...,") were handled by a sequence of generated branch instructions (sp(-)). In a one-pass operation the symbols were read and code generated a symbol at a time; the actual execution sequence used in-line sp orders to hop about from one point to another. The code used some rudimentary stacks, but was sufficiently

intricate that I didn't understand it without extreme concentration even when I wrote it. I have tried to play "human computer" for a while this afternoon, to give you a simple example to work from, but failed. Structured programs were not known in 1953!

The notion of operator precedence as a formal concept did not occur to me in 1953; I lived in fear that someone would write a perfectly reasonable algebraic expression that my system would not analyze correctly. In point of fact, one almost-debugged version which was believed correct for a while would not handle the minus sign properly in "X = -1". I mention this partly because I am not sure whether the version I am sending you does or does not contain this error. Perhaps the fact that page 18 is in a different handwriting (Zierler ?) is a clue that it has been fixed. I just don't remember.

A very much expanded Whirlwind compiler was initiated sometime during 1954-1955, but was abandoned when CSDL acquired its own IBM 650.

Enclosure 2: First Serious Compiled Program

This enclosure represents the first effort to use the Whirlwind compiler on a serious engineering problem. There are three pages of program and the first of several output pages is also attached. Unfortunately I have been unable to attach a date to this, but judging from the January 1954 date on the programming manual, would guess it to date from very early 1954.

The problem addressed is that of a three-dimensional lead pursuit course flown by one aircraft attacking another, including the fire control equations. What makes this personally interesting to me is tied in with the fact that for roughly five years previous to this time the Lab had managed and operated the MIT Rockefeller Differential Analyzer with the principal purpose of solving this general class of problem. Unfortunately, the full three dimensional problem required more integrators than the RDA possessed.

My colleagues who formulated the problem were very skeptical that it could be solved in any reasonable fashion. As a challenge, Zierler and I sat down with them in a 2-1/2 hour coding session, at least half of which was spent in defining notation. The tape was punched, and with the usual beginner's luck it ran successfully the first time! Although we never seriously capitalized on this capability, for reasons of cost and computer availability, my own ego probably never before or since received such a boost.

Enclosure 3: Laning-Miller Report on MAC

This report was written by Jim Miller in late 1970, giving an overview of MAC. It was issued as Instrumentation Laboratory Report R-681, November 1970, but I don't seem to have a copy of the final report; what I am sending you is substantially identical. A brief history is given on pages 14-15.

Enclosure 4: Current MAC Users Guide

MAC-360 was implemented in 1967, and has not changed since in any significant way; the 1973 update to the manual consisted primarily of editorial changes. At the present time, MAC and FORTRAN have about equal use at CSDL. A recent language which could be regarded as a derivative of MAC is the HAL language, conceived and implemented by Intermetrics, Inc., under contract to NASA-JSC for use in the Space Shuttle flight software.

Enclosure 5: MAC-650 Users Guide

MAC-650 was operational at CSDL from early 1958 through 1961 or 1962 when we switched to the H800 computer. The January 1961 manual therefore represents a somewhat mature version of what we were doing on the 650; in particular, we were by then generating code onthe 650 to be run on the 7090. A number of substantial improvements were made to the language between 1958 and 1961; unfortunately, we have no records to identify the details of this development.

The Users Guide should be more or less self explanatory; however, some remarks are perhaps appropriate concerning the operating system within which it is embedded.

For mass storage, we relied on the third RAMAC disc unit produced by IBM, and developed an operating system to match. As a matter of passing interest, for about a year prior to receipt of the RAMAC our operating system used magnetic tape to simulate the RAMAC; by separating valid tape blocks with tape marks and dummy blocks we could update any block in the middle of the tape without affecting others. MAC program was paged into main memory from RAMAC, and programs of 5,000 to 10,000 words in size were not unusual, on a 2,000 word machine. Named variables were stored in main memory; by 1959, however, we were using the disk file also, via the FILE WRITE and FILE READ commands.

The 3-line format of MAC is probably its most controversial feature. Those who have never used it claim it should be extremely difficult to key-punch. Clearly it is less easy to handle than single-line text. However, those of us who have used it extensively claim that this difficulty is greatly exaggerated, and is more than compensated by a presentation which is visually closer to normal mathematical notation, permitting a much easier detection of errors. A simultaneous virtue and vice is the fact that MAC has essentially only one data type: the floating point number. This obviously restricts its use and at the same time makes it much easier to learn than, say, FORTRAN or other more esoteric languages. It is designed for the engineer, who often regards programming as an occasional tool rather than as a full time job. In this regard, we feel it has served its purpose well. The professional programmer per se was essentially non-existent at CSDL during the 50's and early 60's.

Enclosures 6 and 7: Two MAC-650 Programs

I have enclosed two sample MAC-650 programs to give you the flavor of the language as it was used in 1961. The first routine (Enclsoure 6) calls the second as a subroutine (cf. line M0310: DO 97207); the main reason for including both, however, is that a wider variety of language features is illustrated thereby.

Enclosure 8: Computing Devices Memo #14

This simply establishes a date at which our MAC-650/704 capability was operational.

I hope that this material proves to be of interest to you; if not, feel free to junk it. I know that it has been an interesting experience for me to collect it, and has revived many old memories.

Cordially,

J. H. Laning

J.H. Laning

8 Enclosures

JHL:fh

References:

(1) Battin, R.H., O'Keefe, R.J., Petrick, M.B., "Programming Manual for MITILAC 650 Routine", Report R-87, M.I.T. Instrumentation Laboratory, September 1955.

(2) Battin, R.H., O'Keefe, R.J. Petrick, M.B., "The M.I.T. Instrumentation Laboratory Automatic Coding 650 Program", in IBM Applied Science Division, Technical Newsletter No. 10, October 1955.

(3) Battin, R.H., "Programming Manual for BALITAC 650 Routine", Report R-126, M.I.T. Instrumentation Laboratory, August 1956.

Vol 2.113
J. W. Fanning Jr.

Enclosure 2

108-60-3

c|^{10}=2.4588,
c|^{11}=1.4759,
c|^{12}=-.1284,
c|^{13}=.51485,
c|^{20}=.93082,
c|^{21}=.021173,
c|^{22}=.0025995,
c|^{31}=9.9666,
c|^{32}=.36072,
c|^{33}=-.15711,
c|^{34}=.11966,
k=250.5,
c=.367,
v|^{2}=2870,
g=32.2,
v=880,
v|^{1}=750,
p|^{o}=1373.9,
i|^{o}=1.00589,
f|^{o}=1379.9,
d|^{o}=1,
x=6400,
t=0,
d=0,
a=1.2153,
j|^{1}=.00735.

```
j|²=.00689,
t|¹=3.007198,
h=.1,
i|¹=0,
k|¹=((v+v|²)/v|²)(d|°v/k),
SR2,
b=F⁷(j|¹/j|²),
SR3,

m=1,
z=0,
1 Dz=0,
7 p=xd|°/c+p|°,
8 q=.0001p,
u=10000/(c|¹⁰+c|¹¹q+c|¹²q²+c|¹³q³),
q|¹=F³(a),
q|²=F³(d),
s|¹=F²(a),
s|²=F²(d),
i=c|²⁰+10000c|²¹/u+10⁸c|²²/u²,
y|¹=c(i-i|°)/2d|°,
x|¹=-(v-v|¹q|¹q|²)/(1-v|¹q|¹q|²/u +y|¹s|²),
Dx=x|¹,
d|¹=-((v|¹q|¹s|²/u+y|¹q|²)x|¹+vj|²+v|¹q|¹s|²)/x,
z|¹=-(v|¹(1+x|¹/u)s|¹+vj|¹)/xq|²,
b=F⁷(j|¹/j|²),
Dd=d|¹,
Da=z|¹,
Dj|¹=-z|¹q|²-k|¹j|¹+d|°vi|¹F²(b)/k-gj|¹s|²/v+z|¹j|²s|²-(gj|¹/v-z|¹j|²)s|²,
Dj|²=-d|¹+gq|²/v-k|¹j|²+d|°vi|¹F³(b)/k-gj|²s|²/v-z|¹j|¹s|²-(gj|²/v+z|¹j|¹)s
Dt|¹=x|¹/u,
2 PRINT t.
3 PRINT x, d, a, j|¹, j|², t|¹, b.
e=.67-t,
CR4,
e=2.7-t,
CR5,
SP6,
```

```
4 h=.15,
5 h=.2,
6 m=-m,
CP1,
SR7,
SR8,
f=1000(c|31q+c|32q^2+c|33q^3+c|34q^4),
y=c^2(f-f|0-i|0d|0x/c)/2(d|0)^2,
r=F1(x^2+y^2+2xyF2(d)),
r|0=F1((v|1t|1)^2+r^2-2v|1t|1xF3(d)F3(a)),
a|1=F6(xF3(d)F3(a)/r),
a|0=F5(rF2(a|1)/r|0),
PRINT r, r|0, a|0.
e=t-4.9,
CP1,
ST
```

ZIELRER

t=+.16155871-27

x=+.64000003+04 d=+.16155871-27 a=+.12152999+01 j| 1 =+.73500000-02 j| 2 =+.68899998-02 t| 1 =+.30071979+01
b=+.81769049-00

t=+.10000000-00

x=+.63214701+04 d=+.51502749-03 a=+.12112091+01 j| 1 =+.79314811-02 j| 2 =+.68757068-02 t| 1 =+.29432733+01
b=+.85684836-00

t=+.20000000-00

x=+.62434354+04 d=+.10123356-02 a=+.12069029+01 j| 1 =+.84715420-02 j| 2 =+.68808826-02 t| 1 =+.28808076+01
b=+.88879823-00

r=+.62442894+04 r| o =+.58358421+04 a| o =+.15602501+01

t=+.30000001-00

x=+.61659007+04 d=+.14920736-02 a=+.12023821+01 j| 1 =+.89851245-02 j| 2 =+.68982471-02 t| 1 =+.28197651+01
b=+.91612988-00

t=+.40000003-00

x=+.60888709+04 d=+.19544472-02 a=+.11976463+01 j| 1 =+.94818426-02 j| 2 =+.69231553-02 t| 1 =+.27601116+01
b=+.94017356-00

r=+.60897108+04 r| o =+.56727342+04 a| o =+.15444216+01

t=+.50000006-00

x=+.60123519+04 d=+.23996916-02 a=+.11926958+01 j| 1 =+.99678896-02 j| 2 =+.69526485-02 t| 1 =+.27018139+01
b=+.96177488-00

t=+.60000008-00

x=+.59363499+04 d=+.28280557-02 a=+.11875306+01 j| 1 =+.10447267-01 j| 2 =+.69848443-02 t| 1 =+.26448407+01
b=+.98148244-00

r=+.56274702+04 r| o =+.55116121+04 a| o =+.15279202+01

COMPUTING DEVICES NOTES

M. I. T. Instrumentation Laboratory

April 6, 1960 No. 14

3-way Output Mode Option in MAC 704

Output from MAC 704 may now be specified in one of three modes. These are the Punch, Print, and Write modes. A tape prepared in the Punch mode may be converted to cards in the usual way. A tape prepared in the Print mode may be printed at the 704 installation. CAUTION: the programmer is responsible for seeing that security is not violated by use of the Print mode. A tape prepared in the Write mode may be read by the 650 with the aid of Flad subroutine Z25-A or converted to cards by the 650 using the 704 Write Mode Tape Dump (Routine XXXIV), or the tape may be processed by some other suitable program.

Operation to select output mode is as follows:

Sense switch 5 down: Punch mode (same as old system),

Sense switch 6 down: Print mode,

Sense switches 5 and 6 down: Write mode.

At the end of every MAC 704 run, control must be sent manually to 3564 octal. If sense switch 4 is up, the termination routine starting in 3564 will complete the file in a manner appropriate to the output mode, rewind tape 4, and rewind tape 2 (if any). If sense switch 4 is down, the rewinding will be omitted. Thus several run outputs may be stacked on one tape. However, a tape that has any Write mode files must have only Write mode files. The termination routine stops with 042000777777 in the storage register. If the START button is then pushed with sense switch 4 down, the next program will be started immediately. If the START button is pushed with sense switch 4 up, the 704 will stop with 000000003573 in the storage register. Pushing START again will cause sense switch 4 to be re-examined, and the decisions about rewinding will be made as before. This sequence stops with 042000777777 (the same stop described above).

Auxiliary Converter for MAC 704 Output Tapes

The Auxiliary Converter accepts as input a tape prepared in either the Punch or Write mode, and produces a tape in any mode. Operation is as follows:

Output mode selection is the same as in MAC 704, using sense switches 5 and 6.

Sense switch 1 down for Punch mode input, up for Write mode input.

Sense switch 4 down prevents rewinding of input tape after a file is converted. (Note, however, that the binary program deck must be reloaded for each logical file of input).

Input tape is on logical 4, output on logical 5. Only one output file (or part of an output file) may be written on an output tape. Program stop 042000477777 in 10440 is the final stop, and indicates that output tape 5 has been rewound. Binary deck must be reloaded for conversion of next file if any. Program stop 042000333333 in 10444 means end of file on input tape. Since this does not necessarily mean end of logical file, a branch is provided here. If there is another tape with more of the input file on it, mount it as logical 4, put sense switch 3 down, push START. Otherwise put (or leave) sense switch 3 up, and push START. The latter procedure quickly leads to the final stop in 10440. It is possible to come to the final stop without passing through the stop in 10444, because a Write mode input block of from 1 to 8 cards is recognized as a final block.