Inter-Office Memorandum

To CSL, SSL

Date May 14, 1975

From P. Deutsch

Location Palo Alto

Subject Status report on Alto Lisp

Organization PARC/CSL

XEROX

File: <LPD>alispreport.memo (Bravo format)

Keywords: Alto, Lisp

There has been some recent confusion about the status of the Alto Lisp project. This memo attempts to dispel that confusion by presenting a brief history of the project, a summary of its present state, and our current plans for further work.

The project began in the Fall of 1972 as an outgrowth of some preliminary work on the design of an instruction set which would be tailored to compact representation and rapid execution of Lisp programs: this work resulted from, and the results encouraged, a long-standing interest of mine in Lisp systems for small machines. We decided to proceed by first constructing a system with only an S-expression interpreter (i.e. not using a new instruction set) by simply making a copy of the PDP-10 Interlisp implementation, converting it from MACRO-10 to BCPL and running it on a Nova until Altos became available; then adding the ability to run code which had been compiled into the new instruction set, but emulating the instruction set with a BCPL program; then writing microcode for the Alto to perform the emulation at high speed. The first instruction set design was completed in the Spring of 1973 and presented at the 3IJCAI conference at Stanford in August 1973.

In early 1975 we successfully completed the last stage of microcode implementation. It had become clear, meanwhile, that core space would be at a great premium, and that the best way to gain space for paging buffers would be to reduce the amount of BCPL code. Especially in view of the fact that Lisp code using the specialized instruction set and a microcoded emulator executes at speeds comparable to BCPL code, we decided to embark on a major program of simplifying the BCPL implementation code and replacing it by Lisp code. In this process we would approach much more closely the idea of a "Lisp machine" which was entirely self-sufficient (i.e. did not depend on a Nova/BCPL emulator), gain flexibility which might allow us to incorporate some of the interesting ideas from Smalltalk and Mesa, and make the system more easily maintainable. We are currently pursuing this effort. Our first major step, to replace the original S-expression interpreter (EVAL and APPLY) by compiled Lisp code, is nearing completion.

Throughout the course of the Alto Lisp project there has been conflict between the desire to produce a working system for the (relatively) large and demanding Parc Lisp user community, on the one hand, and the desire to experiment with system designs, on the other. It

has been my opinion that the project's chief justification has been in the latter activity, and consequently experimentation and redesign has generally taken precedence over concerted efforts to produce a working system. Recently, however, three factors have combined to give more weight to producing a usable system: the arrival of more Altos and RAMs; the increasing load on Maxc, especially by the Understander project (heavy Lisp users); the availability of a third person to work on the Lisp project. Consequently, starting in June 1975 we will be able to devote more resources to this goal without compromising our design investigations.

The project to date has had research results in four areas.

(1) Instruction set design: we have experimented with two significantly different instruction set designs for Lisp, and taken extensive statistics on their worth for encoding the present corpus of Lisp programs.

(2) Processor design: having completed the Lisp microcode, we have had several useful comments to make on the design of the successor to the Alto.

(3) Data encodings: we have collected numerous interesting statistics on the entropy of Lisp list structures and designed an efficient encoding for these structures.

(4) Storage management: we have developed an interesting new class of storage reclamation (garbage collection) techniques.

We have published papers on topics (1), (3), and (4).

The current status of the system may best be described as "demonstrable" but not "usable". It is possible to sit down at an Alto equipped with a RAM and certain minor ROM changes (of which there are only two at present), insert an appropriately initialized disk pack, and interact with a system that looks very much like Tenex Interlisp. (Willie Sue or I will gladly give short demos.) However, there are some basic elements missing (interrupt characters being the most noticeable), none of Warren's assistance features are available (editor, prettyprint, file package, break/trace, DWIM, CLISP, etc.), the system is 5 to 7 times slower than a completely unloaded Maxc (i.e. comparable to Maxc with load average of 5), and there is no garbage collector, so it is quite possible to run entirely out of space in the course of a few hours' work.

Our current work attacks all of these problems. Larry Masinter will be working on bringing over Warren's code this summer: since it is all written in Lisp, and since the Alto system is supposed to be source-compatible with the Tenex Interlisp system, we do not anticipate substantial difficulties (in fact, we have run Warren's editor experimentally in the past). Interrupt characters and a few other similar odds and ends will be implemented in order according to their difficulty in the near future. We hope for some speed improvement through further system tuning and analysis, but we are probably fairly close to the limit of the Alto's performance now. The garbage collector is a complex program and will be implemented in stages, but the first stage, which should reclaim a large part of casual garbage, has been written and largely debugged, and will be installed within the next few weeks.