

UC Berkeley Computer Science

Subject: Notation Questions
Date: September 12, 2005
From: Ivan Sutherland
UCIES #2005-is05

NAME _____

References:

UCIES# 2005-is02: FLEET – A One-Instruction Computer, Ivan Sutherland, 24 August 2005
UCIES# 2005-is03: Defining Some SHIPs, Ivan Sutherland, 24 August 2005
UCIES# 2005-is04: A Dozen Problems, Ivan Sutherland, 6 September 2005

PURPOSE

This memo intends to collect your ideas about notation. Please write your name at the top of the paper and write thoughtful comments in the blank spaces below.

MOVE INSTRUCTIONS

Because there's only one instruction, does it need a herald, e.g. the "MOVE" in "MOVE a to b"? Might an infix notation be better? Please suggest some alternatives.

SOURCES AND DESTINATIONS

How shall we name sources and destinations? Should we use a two-part name that references the SHIP and its input or output, e.g. ADDER input A? Will a short-hand notation be better? Should we declare a name for a particular SHIP and then use that name? Should we declare names for particular inputs and outputs? How about labeled pictures?

Please record your idea for source and destination notation.

This document is a product of a collaboration between Sun Microsystems and the University of California at Berkeley.. The ideas contained herein are freely available for any academic purpose.

LITERAL NOTATION

In most programming languages one can make assignment statements like
`foo := whip + 5;` (I pronounce “:=” as “gets”.)
 Here the “5” is to be taken literally as the fifth integer, and not as a location in memory. The names “foo” and “whip”, on the other hand, designate variables whose locations in memory are known to the compiler but not necessarily to the programmer.

In some assembly languages a number refers instead to a particular location in memory. For example, `356 := 239 + 101;` might mean add the contents of location 239 to the contents of location 101 and put the result into location 356.

How shall we distinguish numbers that are to be taken literally from those that designate locations in memory or the inputs and outputs of SHIPs? Shall we permit string literals, and if so, how shall we distinguish them from names?

LITERAL MECHANISM

We have yet to define how FLEET gets literals into use. In most computers a literal is stored as part of the program code, often occupying the bits that might otherwise indicate the name of a register. This makes sense for small integers, but makes no sense for character strings.

Do we need a literal mechanism for FLEET at all? Why not just store all literals in main memory and fetch them from there. That choice would eliminate the atomic operation “5” goes into ADDERinputA.
 Surely we could fetch the value 5 from main memory.

Can we live without any literals in our code bags? If not, please explain why not.