

# UC Berkeley Computer Science

**Subject:** The Size of a Record  
**Date:** August 7, 2006  
**From:** Ivan Sutherland  
**UCIES** #2006-is29

## 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# 2006-is23: FIFO Register File, Ivan Sutherland, 30 June 2006  
UCIES# 2006-is24: Parallel Switch Fabrics, Ivan Sutherland, 10 July 2006  
UCIES# 2006-is25: Instruction Sequence in FLEET, Ivan Sutherland & Igor Benko, 19 July 2006  
UCIES# 2006-is26: Record SHIPs, Ivan Sutherland, Igor Benko & Mark Greenstreet, 29 July 2006  
UCIES# 2006-is27: Scatter and Gather, Ivan Sutherland, 2 August 2006  
UCIES# 2006-is28: How Long Is A Record, Ivan Sutherland, 2 August 2006

## INTRODUCTION

We want FLEET to send a record of several words through its switch fabric as a unit. However, we have yet to say exactly how and where to define the size of a record, i.e. the number of words in it. Memo ucb-28, How Long is a Record, deals nicely with how to mark the end of a record passing through the switch fabric. However, ucb-28 is weak about how to define the record's size.

One alternative defines the size of a record as the number of occupied words in a record-length store. In effect, this makes the size of a record be the occupancy of the source record-length store at the time the record moves. This definition suffers from ambiguity if the number of words in a record-length store increases just as the record leaves. If a new word arrives just as the existing record departs, does the new word count in the "size" of the record? Will the new word be sent as a part of the record or not?

Another alternative defines the size of a record in the program. This alternative uses the count in a "counting move" (CMOVE) instruction to specify how many words to move from source to destination. This definition suffers from stalling the switch fabric if too few words are available at the source to complete the CMOVE. If the program calls for moving five words, but only two are available, does the CMOVE instruction stay valid for the three missing values? What does the switch fabric do during the delay?

## A PROPOSAL – start only when ready

I propose that the program state how many words to move, but that FLEET delay starting a CMOVE instruction until enough data are available. The CMOVE instruction will wait at the source until the source contains enough data to complete the action. This is consistent with single word MOVE instructions; a single word MOVE is exactly a CMOVE with count one. Moreover, with a CMOVE instruction pending at its output, the number of words of data present in a record-length store can only increase until enough data are available.

---

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.

There are two problems with this proposal. First, we may also want an “indirect count” move instruction (IMOVE). Second, FLEET must be able to report how full a record-length store is, so that a program can save and restore records. I shall deal with these two problems in turn.

## **THE INDIRECT COUNT MOVE - IMOVE**

Sometimes the number of words to move is data dependent. For example, the length of a vector may be a variable rather than a program constant. If such a vector is treated as eight-word records, the program will use CMOVE instructions to move many records of eight words each. However, the size of the final record will depend on the initial length of the vector and may be any number of words from zero to eight. We need a way to move the final record no matter what its length.

I propose to have “size” registers into which the program can store the size of records. An indirect count move instruction (IMOVE) will use the number of words indicated in a size register up to a maximum of eight. This contrasts with the CMOVE instruction that moves the number of words specified in a count field internal to the CMOVE. The count of an IMOVE is provided indirectly from the corresponding size register up to eight maximum.

## **SIZE REGISTERS**

It may be good to have a size register associated with each record-length source. For example, imagine 16 record-length registers as the “record file” for FLEET. Such a record file takes the place of the “register file” in a conventional machine. In addition, the record file might have 16 size registers, each associated with one record-length register. An IMOVE instruction will move the number of words contained in the corresponding size register up to a maximum of eight words. The IMOVE instruction will wait until enough data is available at the record-length source to complete the move. Any SHIP that serves as a record-length source would likewise have a size register.

A size register should be treated as a source and a destination. A size register may be empty. It will ordinarily be the programmer’s responsibility to put an appropriate value in a size register before issuing an IMOVE instruction.

I believe that an IMOVE instruction from a source whose size register contains zero will be a no-op. However, an IMOVE instruction to a source whose size register is empty should wait to start its action until both the size register has a value and enough data elements are present to complete the move.

## **REPORTING OCCUPANCY**

FLEET can use a size register to report the occupancy of a record-length store. Such a report can safely enter an empty size register as the result of a special input to the record-length store. Because the size register is initially empty, no IMOVE instruction can decrease the content of the record-length store. Because the command to issue such a report comes into the input of the record-length store, the command itself blocks other values from entering. Thus, given an initially empty size register, the result of seeking such a report and an IMOVE instruction to use it is independent of the issue sequence of the two instructions. We can use such a sequence to save later to restore the record store.

Some ALU SHIPs might set their output size register automatically. For example, an ALU SHIP might automatically set its size register to match the number of values most recently processed. Moreover, reporting the size of output records would

permit overflow to produce multiple precision output. The size register would report the number of words required to represent the value.

## **MOVE AND DECREMENT**

If eight is the maximum number of words that can move as a single record, what does it mean to have size values greater than eight? I favor moving the fewer of eight words or the number specified in the size register.

One might include a decrement capability in MOVE instructions. A simple move and decrement instruction (MOVD) would wait until the size is defined and there is at least one word of data to move. Then it would move a single word from the source and decrease the value of the size register by one. A counting move and decrement instruction (CMOVD) would wait until the size is defined and the specified number of words are available, and then move the specified number of words and decrement the size register by the number moved. An indirect move and decrement (IMOVD) instruction would wait until the size is defined and enough data are available and then move the smaller of eight or the value of the size register words. It would also decrement the size register by eight regardless of how many words moved, possibly leaving the size negative. The IMOVD instruction might simplify loops for moving long vectors.

If zero or negative integer values map into FALSE, one might read the value of the size register to detect the end of a loop.

## **AN OVERVIEW**

What's happening here? I'm adding features based on the size register that may or may not save program steps. If, indeed, wires are the problem and transistors are free, such features may prove valuable in saving energy or time. They may also be difficult to harness in compiled code.

The size registers are a bit like PDP-4's auto increment registers. In the PDP-4 certain memory registers would increment after use as indirect addresses or as counts. That made it quick and easy to step through certain tables of data. More general cases, however, were not possible. Where should we draw the line on such features?