MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts
Project MAC

Artificial Intelligence Project
Memo 72

Memorandum MAC-M-183
September 13, 1964

Proposed Instructions on the GE 635 for List Processing and Push Down Stacks

by Michael Levin

I. List Processing

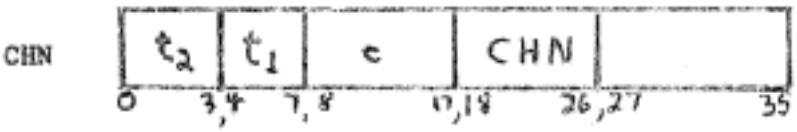
The instructions that transmit data between the index registers and the memory work only on the left half (address) portion of memory. These instructions are LDXn (load index n from address of storage word), and STXn (store the contents of index n in address of storage word). The effective address of both of these instructions includes modification by index registers.

A corresponding set of instructions for transmitting data to or from the right half of memory would facilitate list structure operations. The present order code makes it impossible to do list-chaining operations (car or cdr) without disturbing the A or Q registers.

Other instructions that operate between index registers and the left half of memory such as LXCn and ADXn do not require corresponding right half instructions.

It is frequently necessary to perform multiple car-cdr chains in order to access a particular substructure of a list. The ability to execute such a chain under the command of a single instruction would speed up list pro-

cessing considerably. The proposed instruction CHN permits one to do this.



The chain (c) field is scanned from left to right until the first one bit is encountered. The number of bits in the c field to the right of this bit determine the number of steps to be performed. For each step:

1. If this is the first step, then the contents of X_{t1} are used as the address for getting a data word from storage. In all succeeding steps, the contents of X_{t2} are used.

2. If the bit for this step is zero, the contents of the left half of the data word are placed in X_{t2}, otherwise the contents of the right half of the data word are placed in X_{t2}.

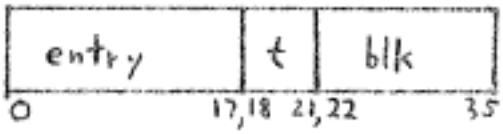
Extra bits in this instruction are available for specifying a program segment.

VI. Push Down Stack Instructions

A single instruction for entering a procedure and reserving a block of storage on the push down stack must be capable of incrementing the push down pointer, checking to determine if the push down stack has been exhausted, placing the saved instruction location and the size of the push down block at the end of the block, and transferring to the entry point of the procedure. The proposed instructions PUSHT, and POPT will probably have to be modified to accomodate the program segment scheme.

PUSHT:

The instruction has an address subject to all normal modifications. The address determines the location of a data word which is interpreted according to format I.

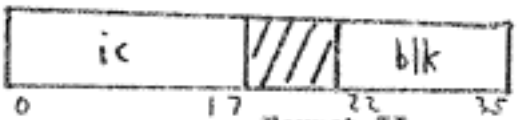


Format I

After the data word has been obtained, the instruction proceeds as follows.

1. The contents of blk are added to X_t.*

2. A word is formed using the contents of the instruction counter (ic) and the block size (blk) as shown by format II.



Format II

3. If $C(X_t) > C(X_{t+1})$ then the format II word is stored at the address determined by X_t, otherwise control goes to trap.

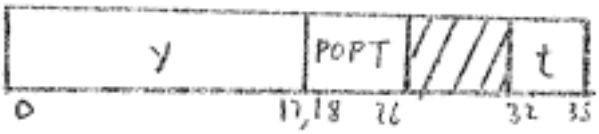
4. A transfer is made to the address specified in the entry field.

TRAP:

A trap is made to a fixed location relocated within the segment containing the PUSHT instruction. The processor does not leave slave mode.

*The field blk is made into an 18 bit field by preceding it with bits having the same sign as the left-most bit of blk. This permits negative indexing.

POPT:



1. The data word addressed by X_t is obtained and interpreted according to format II.

2. The blk field is subtracted from X_t.

3. A transfer is made to the location which is the sum of the y field of the instruction and the ic field of the format II word.