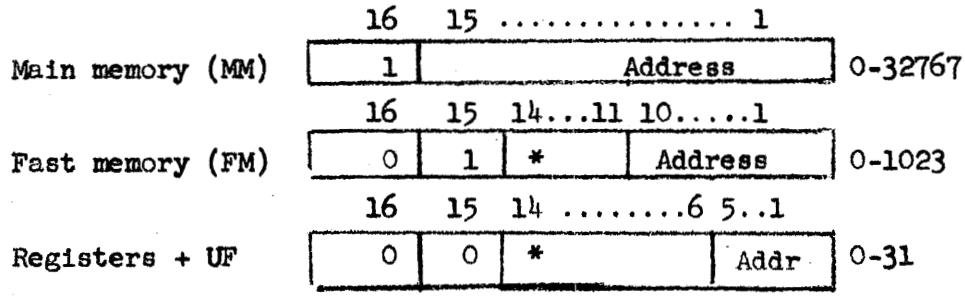
STRETCH ADDRESSING

Memory addressing. The following system for addressing the memories in Stretch seems to be both simple and satisfactory. We assume at this time an address length of 16 bits.



Bits 16 and 15 above are used to select the proper memory and a (*) indicates that these bits are not examined at present. This system does not, however, give continuity between FM and MM and if this is important the following arrangement is preferred: Registers + UF, 0-31; FM, 3072-4095; and MM. 4096-36863.

The FM positions should be addressed sequentially, unit #1 having the addresses 0-511, unit #2 the addresses 512-1023, etc. The MM positions should be addressed in a "4-ply" interlaced fashion, unit #1 having the addresses 0(4)32764. #2 the addresses 1(4)32765 etc. It would be preferable to have 8 units of 4096 each and hence two 4-ply interlaced sets.

We would like to single out the registers A. S. and A', and give them the addresses 0, 1, and 2. We would also like to recognize a set of index registers X_i addressable in the regular manner as well as in a special way to be described later. The X_i registers quickly consume most of the proposed 16 ultra-fast (UF) positions. It is, therefore, quite important to have at least another 16 made available. However, if there is any possibility of somewhat faster FM memory one could easily dispense with the 16 special UF positions.

Instruction format. The following format involving an operation part and three address parts will be discussed first. A 60-bit word is assumed for this purpose.

Field	Opn	T	X	B
Bits	20	12	12	16

where T is a "short address" capable of addressing storage positions up to 4095 incl., where X is a geometric address capable of addressing simultaneously up to 12 index registers, and where B is the "long address" or "base address" capable of addressing directly any position up to 32767 incl. The 20 operation bits include the "α-bits" to be used for assigning various functions to the three addresses besides those which we might regard as standard. The content of X, i.e., C (X), is to be interpreted as $\Sigma C(X_i)$ summed over those $X_i \neq 0$. If all X_i (all 12 bits) equal to zero then C (X) = 0.

For arithmetic operations B' = B + C (X) will normally be the address of the principal operand and T will be used to address a secondary operand. The following secondary operations seem to be extremely useful (1) Prestore, i.e., C (B') → T combined with the primary operation. (2) Preload, i.e., C (T) → A before the primary operation. (3) Poststore, i.e., C (A) → T after the primary operation. We may also want combinations of these such as "Preload, perform the primary operation, and Poststore". A few test examples indicate that 704 codes would be reduced by 25 to 35% by just these features. A special interpretation of the addresses would be B' = B + C (X) + C (T), i.e., "regional" addressing. Indirect addressing represents still another special case, in which C (B') rather than B' is taken as the address. C (B') should include the α-bits and hence, provide for, among other things, indirect addressing of higher orders.

In unconditional transfer instructions and also in transfers with explicit conditions (A $>$, $\geq$, or = 0 plus complements, the various TX-types with internal comparisons in the index registers, and the "transfer on trigger" types) the effective address B' would normally be given by B alone, with T calling for a secondary operation, and by B + C (T) for the regional case. X would then specify which index registers are to be reset, (arithmetic transfers), which index registers are to be modified (TX-type transfers) or which triggers in a fixed field of 12 are to be examined (trigger transfers). In implicit comparisons T would be needed to specify the location of the comparison number or the trigger field. TSX-type instructions would call for a Prestore of the instruction counter in the location specified by T and B' would normally be B + C (X). Again, indirect addressing is possible in all these instructions, with C (B') replacing B'.

Index registers. The format specifies four quantities, the first two forming a counter and the second pair an address modifier.

Field	N	n_i	with sign Δ	x_i
Bits	12	12	18	18

In reset status the register contains $n_i = 0$ and $x_i = 0$. The TX-type instructions "advance" the register, i.e., replace n_i by $n_i + 1$ and x_i by $x_i + \Delta$ unless $n_i + 1$ becomes equal to N in which case the register is reset. The status of the register determines whether or not the transfer takes place. These instructions should be able to refer to several registers at the same time. A set of instructions is needed to load (or store from) parts of or the entire register.

Double indexing. There are a number of important applications of double indexing, i.e., where both T and B are modified by X. T' = T + C (X) and B' = B + C (X). Such special addressing possibilities should be included for arithmetic operations.

Two index fields. More general two-address systems have been discussed without any definite conclusions. In these systems one would have four address parts, two 16-bit addresses B and $\overline{B}$, and two X-fields, one normally associated with B and the other, perhaps a 3-bit field $\overline{X}$, associated with $\overline{B}$. The first 3 positions of X would correspond to $\overline{X}$. In some problems one can definitely see use for such generalized systems.