Division 6 - Lincoln Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts

SUBJECT: FOUR ALTERNATIVE PROPOSALS FOR ACCOMODATING THE 64 x 64 MEMORY IN MTC

To: N. H. Taylor, R. R. Everett, Programmers Generally

From: W. A. Hosier

Date: August 14, 1953

ABSTRACT: Four proposals are outlined for using a 64 x 64 = 2^{12} -register memory in MTC, which was designed to have only 11 address bits. The first three involve very little extra equipment:

I. Division of Memory into 32 x 32 quadrants selected by bank-switching instruction
II. Operating as a 2048-word memory, with 2048 additional registers subject to limited access on only two or three instructions
III. Use of a full 12-bit address by cutting instruction code to 16

The fourth, however, addition of a 17th digit to almost every register of the computer, is quite different from the standpoint of labor and equipment, being about a month's extra work for 4 or 5 people.

When the 64 x 64 magnetic memory is incorporated into MTC, one is confronted with the problem of how to select a 12-bit address in a computer which has at present only 11 bits available for this purpose. The proposals so far advanced to solve this problem boil down to the four distinct schemes here presented; they are being circulated in this manner to see whether any pronounced preference among them will be expressed by programmers.

The programmers' choice needs to be made between the first three proposals; no doubt all programmers would prefer the fourth, by which they can eat their cake and have it too. As between the first three modes of operation on the one hand and the fourth on the other, the decision is of a different sort, since it must weigh programmers' convenience against design and construction effort.

Any views on the subject of this memo should be expressed to the author or to Phil Bagley, preferably before September 1 if they are to affect any decisions.

I. Division of Memory into 32 x 32 quadrants selected by bank-switching instruction

A. Each bank-switch instruction effective until cancelled by the next one

In this mode, MTC would operate while in any one of the four quadrants exactly as it did with its old memory, the only difference being that addresses of instructions could refer to any one of the four quadrants, as indicated by a previous bank-switching instruction, bk. Somewhat more specifically, it would work as follows:

Instructions (as opposed to words manipulated by instructions) would be taken from the quadrant of memory designated by a 2-bit register, the "program timing bank selector" or ptbs. Addresses given by instructions, on the other hand, would always refer to the quadrant of memory designated by a second 2-bit register, the "operation timing bank selector" or otbs. The otbs, but not the ptbs, would be set by each bk instruction, and would remain so until the next bk. The ptbs, on the other hand, would be set to agree with the otbs either by a special "bank transfer" instruction or by an ordinary transfer instruction immediately following a bk. (Alternatively, a transfer instruction could always set ptbs to agree with otbs; but that would mean that most transfers would probably have to be preceded by an extra bk to keep the program from changing quadrants)

The number of bk instructions in a program may be kept to a minimum if one is willing to abide by the following restrictions:

(1) The program should not jump back and forth between quadrants.
(2) Registers referred to by successive instructions should be in one quadrant, as far as possible.

Since each quadrant is as large as the entire original memory, these restrictions should not prove terribly onerous.

B. Each bank-switch instruction effective for next instruction only

An alternative to the above, probably less desirable, is to have the otbs reset to agree with ptbs by all instructions other than transfer instructions, which would do the opposite. This solves the transfer problem posed by the above mode of operation, but of course necessitates a bk instruction for each address referred to outside the quadrant where the program happens to be. Using this mode to best advantage would thus require keeping resisters referred to by any part of the program in the same quadrant as the program itself.

One minor advantage of such operation over any of the other three proposals is that panel storage (64 registers, half plugboard and half toggle-switches) could continue to be addressed as it was with the eleventh address bit, needing neither to jettison 64 registers of magnetic memory nor to be pre-selected by a "switch" instruction.

A possible variation would be to switch halves instead of quadrants. This necessitates doing something special about panel storage: either having to switch to it separately, or giving it 64 of the addresses rightfully belonging to magnetic registers.

II. Operating as a 2048-word memory, with 2048 additional registers subject to limited access on only two or three instructions

In this mode, operation would be the same as before (except for the special treatment needed for panel storage) in one half of the memory (call it bank A); but the addresses for the other bank ("B") would be available only in a restricted category of instructions--say, as a minimum, "clear & add", "store", and "transfer". The thought behind such an arrangement is that bank B would hold only program, and that it is seldom desired to perform involved arithmetic operations on instructions. Any numbers, etc., to be manipulated arithmetically would be stored in bank A (which could of course also contain as must program as it had room for,

and the program counter would automatically index from the last register of one bank to the first register of the other). Arithmetic manipulation of words in bank B would of course be possible, if unavoidable, by storing them temporarily elsewhere.

This mode would seem to demand about as much of programmers as mode I, but with a different emphasis, namely that of avoiding unnecessary manipulation of instructions.

It does of course reduce the number of instructions available, since the extra address bit of each "unrestricted" instruction has to come out of the five instruction bits; but this would reduce the number of distinct instructions by at most four, to 28, a fairly generous number compared to the 16 of proposal III below, and quite adequate for any application of the machine that we can envisage.

It would be nice to have conditional transfer in the unrestricted category, but this could always be got around by a conditional transfer to a register containing unconditional transfer; besides, another handy instruction to put in the unrestricted category would be the identity check. All instructions in the in-out class would presumably have unrestricted memory access, having more bits at their disposal.

III. Use of a full 12-bit address by cutting instruction code to 16

Except for panel storage, which must be taken care of by one of the two means outlined above (separate switch instruction or jettison of 64 memory registers), the problem is "solved" by giving up half the available instruction code. If all in-out & display instructions, as well as "halt", "cr", "sr", are absorbed in two in-out instructions and some "free ride" instructions like "transfer & pulse" are pared off, we would be operating on 12 instructions. This would limit addition of such convenience instructions as "add one", "exchange", "logical multiply", "divide", "special add", etc.

There is certainly question whether the storage space necessary to synthesize some of these instructions would not be as much as would be sacrificed to bank-switching instructions in mode I, or to get around the limitations of mode II, not to mention the cramping of experimentation imposed by the limited instruction code.

IV. Addition of a 17th digit to all registers of the computer

Plainly this would have all the advantages of proposal III, with none of its disadvantages save the minor one of panel storage. It would however place a much bigger engineering and construction load on MTC personnel, necessitating as it would addition and relocation of components on some 23 panels of the main frame (including punching of tube sockets in about 10 panels) and considerable re-shuffling of diode networks in panel storage. Bill Papian has said that there is no significant additional effort required to accomodate 17 bits plus a parity bit in the memory and associated hardware, none of which is yet constructed anyhow. It seems quite feasible to add this digit, in general, to existing panels, with no disfigurement to the machine as a whole save occasional crowding or chipping of the paint.

The design and construction incidental to any of the first three proposals is almost negligible, and could simply be considered a part of the 64 x 64 memory installation; but the work for this proposal would almost certainly require a month's steady work by two engineers and two technicians, at least, plus the drafting time needed for all changes made, over and above the memory installation job.

There is a further consideration, namely, that if programmers discern a crying need for the 17th bit after trying out one of the first three arrangements, the labor of making the changeover will be about the same regardless of when it might be done. With careful planning, it should not even be necessary to rewrite programs written in another mode for operation in mode IV. Opposed to this is the fact that a month's shutdown now will not cut into operating time as it very likely would any time after the 64x64 memory is put into operation.

Signed W. A. Hosier
W. A. Hosier

WAH/jg

cc: N. Taylor
R. Everett
W. Hosier (10 copies)
P. Bagley
W. Ogden
W. Papian
W. Clark
B. Farley
J. Porter
S. Best