Save

Kolsky

University of California
Los Alamos Scientific Laboratory
(Contract W-7405-ENG-36)
P. O. Box 1663
Los Alamos, New Mexico

March 27, 1959

Mr. Dura Sweeney
IBM Product Planning
P. O. Box 390
Poughkeepsie, New York

Dear Dura:

[illegible] 2 [illegible] + 16 VFL [illegible] 4 [illegible] each

We have noticed, with some alarm, that IBM has filled up the set of VFL Integer instructions. We have inferred from this that IBM does not intend so to construct the Stretch machine as to facilitate the later addition of new orders. If this inference is correct then now is presumably the time to fill as many as possible of the vacant slots with instructions of modest usefulness and negligible hardware cost. Accordingly, we have prepared lists showing space available, space which could be made available by dropping instructions the usefulness of which we do not understand, and new instructions desired by us. We believe that <u>some</u> of the new instructions <u>probably</u> require too much additional hardware; it should be understood that we list them only on the chance that they do <u>not</u>.

We are also taking this opportunity to list miscellaneous questions. Most of these questions are "whif" questions (<u>w</u>hat <u>h</u>appens <u>if</u>). If such questions are asked without comment, then we <u>only</u> want to know for the sake of curiosity; we are <u>not</u> requesting that anything be done about it. If we do want something done about it, we shall say so.

Require a lot of work

Finally, there are few general remarks and/or requests.

We intend to telephone you for replies and comments a few days after you receive this letter. We would then like all remaining questions to be placed on the agenda for the April 27th. meeting in Poughkeepsie.

Space Available. According to the Manuals in our possession, the following space is currently available. Floating Point: 3. Unconditional Branch: 1. Miscellaneous: 3. Direct and Immediate Index, Count-and-or Indicator Branch: 0. Integer and Connect: 1 Connect or 2 Integer. Input-Output: 6. Transmit-Swap: bit 55 unused except on Transmit half-words immediate. General Full Word: 8 with op-code xxx1000001111 in bits 51-63; also 32 with op-code xxxxx10001111, although these would not work with Store Address. In addition to these, there should probably be the following. Floating Point: 2 of the Add to Magnitude and Augment family*. Unconditional Branch: 1 (Set and Wait has never been justified). Miscellaneous: 1, if Store Zero is dropped, but this might be replaced by Store Floating Point Zero. Integer: 4 to 7 and 3/2 (1 Convert seems sufficient, Load N and Set seems unnecessary, Compare Field if Equal seems useless and perhaps two more Compare orders could be dispensed with, and Decimal Multiply, Multiply Cumulative, and Divide should probably be dropped).

being changed

no comment

STOP yes in many cases don't care

New Orders Desired. For any of the following orders which are not easy to put in, please consider the uses referred to and think whether there might be an easier order which would facilitate the same job.

too late to change EB.

*It is rumored that this has been done. But it is also rumored that there are now four compares, including compare /a/ with -/m/.

1, Swap with Accumulator. Floating Point order, half-word, with any interpretation of sign and normalization modifiers convenient to you. The equivalent program takes three half-word orders and an index word, or two half-word orders, a Transmit, and a memory word. (If the present Swap addressing the accumulator takes the old sign from, and puts the new sign in, the Sign Register, then that is satisfactory.)

Ld M N
STO M
Ld N

Same as VFL now

2, Store Floating Point Zero. See discussion below of Negative Exponent Spill and Floating Point Zero. Assuming this problem is going to be solved somehow, it would be useful to have a half-word order for storing a floating point word with a large negative exponent and a zero mantissa.

TMT preferable

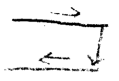
3, Set Exponent; Set Exponent Immediate. These could be Floating Point or Miscellaneous. They would first set the accumulator exponent to zero and then Add Exponent. The equivalent program is Connect 0000 followed by Add Exponent, or, if it works, Add Exponent Negative, Address 8, followed by Add Exponent. This has fairly low priority but is presumably very easy for your engineers.

ask McSorley ?

4, Reflect Accumulator. In its simplest form this changes accumulator bits according to the recipe bit n to position 127-n, for n = 0, 1, ...63. In its best form it does this job for a specified field in the accumulator, leaving the rest of the accumulator unchanged. Note that the primary reason for wanting this order is to develop a right zeros count. Perhaps you can solve that problem in a direct way, such as developing a right zeros count on connect orders. Incidentally, this inversion can be performed on Maniac type machines by connecting the low order ends of two registers and shifting one register right while the other shifts left.

ask Kane
Interesting? hard

Sto
connect
[illegible]
[illegible]
C+B
load



5, Assorted Shift Orders. There seem to be a number of jobs difficult to do with the connects alone. For simple displacements of fields shorter than 65 bits but much longer than 8 bits, connects will work but we are afraid they are too slow. Shifts of fields bigger than 64 bits are perhaps not too important. Circular shifts, especially of one field in the accumulator without disturbing the rest of the accumulator, are particularly difficult and may be rather useful (the equivalent program uses three connects; furthermore, a circular shift by n places of m bit field at offset o, potentially specifiable in one order, requires Connect to Memory with field m-n and offset o, Connect to Memory location 8.(127-o) with field n and offset o+m-n, Connect with field m-n and offset o+m-n; this is a mess if m, n, and o are computed quantities and it fails if n exceeds m). Note: by "shift" we only mean something giving the same result as an old fashioned shift, and with reasonable speed.

6, Set Indicator to One. If, as we are requesting below, we get some Programmer's Indicators (the Harvest indicators and/or the Transit register), then a fast half-word order to turn them on would be nice.

7, Extract. This has been brought up before and seemed too difficult. It should be looked at again, now that you know your circuits. The order is defined as follows: where a mask word, C, has zeros, leave A alone; where the mask word has ones, take for result the connection of A and M. This could be restricted to the connection 0011: Leave A where C is zero; use M where C is one. I believe that the logical statement of the general case is $A.\bar{C}$ v $(A \circledast M).C$. The chief use of extraction is to insert into A assorted, disjoint fields from M.

[illegible] "Splatter Transmit" very good what about A+C, B+R?

8. Generalized Swaps and Transmits. There are two sources of slack in the Transmit order: bit 55 is not used much, and for Direct mode the value and refill fields of the index word are not used. The first generalization which comes to mind is to use the value field of the index word to step the addresses, instead of using an implied +1. Example: if we now Swap n-1 words between α and α+1, the list x1, x2, ...xn becomes x2, x3, ...xn, x1. With a Value of -1, the list would become xn, x1, x2, ...x(n-1), i.e., a cyclic permutation in the other direction. There are also common cases, for certain approaches to storing arrays, for transmitting every kth word in one block to the corresponding positions in another block. The second generalization we have is to use bit 55 on Transmit direct to freeze the source address; the order then fills a block of memory with a constant value. This can be done by transmitting the constant to α and then transmitting n-1 words from α to α+1, but we don't know whether or not this is excessively slow. The third idea is to use the Index Flag and Refill Field to do chaining; this makes sense only if combined with the use of the value field mentioned above.

Expensive

9. Connect to Sign Register. The ability to move bits from memory to the flag and zone positions might be useful.

10. Locate List Element. This suggestion is a rather late response to the determination made in the early days that Stretch should incorporate instructions useful for automatic coding. The desire is to generalize the Load Indirect order, which now terminates due to a flag or mark in the list element itself (i.e., op-code not that of Load Indirect). It now seems that

this is less useful than an order which says Load Indirect to the nth Level. This, in turn, is less useful than an order which says Load Indirect until you find element E. For this last, think in terms of a set of n full words arranged in memory in a disordered way (as must happen if an original ordered list is constantly being added to and subtracted from). Each such word would contain the address of the next word (as in Load Indirect). The last word might be marked, say, with an index flag bit. But each word would also have a field containing its name, and the order would specify a name and continue loading until it got an equality between the list element name and the specified name (or got to the end of the list). If such a Load Indirect, guided by a comparison or a count, seems feasible, then we have some thoughts on field assignment which I am not including here. If this does not seem feasible, how about recording somewhere the number of levels used by the Load Indirect, so that a by-product result would be the ordinal number in the list of the element loaded?

here

Miscellaneous Comments and Questions.

require complete fetch

1. Does a VFL Store or Connect to Memory require a complete fetch of the old data word, or is the information merged at the memory unit? Does a two bit store to an address with bit address 63 require two full fetch bus times and two full store bus times?

no

yes,

2. What is the timing for a Swap of n words between α and α+1? In particular, when step two is reached, does Stretch have to re-fetch from α+1 the word it just stored there? We would like this to be made reasonably fast, if possible.

probably yes,

ask Jerry

4.8 slowest ?

[illegible]

3. If an ordinary floating point number is added to a zero with large negative exponent, is the Preparatory Shift Greater than 48 indicator turned on? If it is, then there is really no point in allowing interrupts on this indicator, since the above situation is very common. It would be much better if it were not turned on.

prob. O.K. ?

4. IBM has still never replied to our request, now almost a year old, to give us an example justifying the existence of the Set and Wait instruction.

5. What is the present indicator set? Is there a Noisy Mode indicator which can be masked on or off? We feel that there must be, for safety. We request very strongly that all unused indicators (in particular, the "Harvest" indicators) be wired up as programmers' indicators, for two reasons: to serve as high speed toggles, and to permit a limited class of half-word bit-branching (for use with STICI).

it [illegible]

6. When a memory operand is at an address less than 16, is there a gain in speed? We would like things to be arranged so that one can use the miscellaneous registers for data and get fast access.

yes

7. What happens if there are two op-code 8's in succession? If this gives Invalid Operation, then we are wasting full word op-codes.

check on? have invalid op

s[illegible] [illegible]

8. How much time is spent for the two boundary register tests? Is this slowing down the machine during otherwise fast loops? Where is this comparison made?

no

mem bus unit

9. What is the present status of the floating point Add to Magnitude and Augment orders? Have you made the changes analagous to those made in the integer set? We feel that this should be done.

lookups 5

same intrupt system would need extra 1A level

10. What are the reasons for not having fast one-word fixups? Can we have them if we prohibit the programmer from changing data that is used in the next few succeeding instructions? This may or may not be important, depending upon what is being done about Negative Exponent Spill.

ask ~~Flook~~ as?

11. What happens if a Rename instruction has an effective address identical to the address in the current XO Refill field? It is essential that the store effectively precede the fetch, as it says in the Manual; otherwise we lose the fool-proof updating feature which is the chief (only?) justification for the Rename instruction.

Little in 0's to right;

12. What happens if a VFL Immediate instruction, such as Load, specifies a Field Length greater than 24 bits?

13. What are the properties of Floating Point Zeros in connection with Exponent Underflow? If there is a mode of machine operation in which any result having negative exponent spill is replaced quickly and automatically by a number with exponent all ones and mantissa zero, then we have no further worries (but see comments on Prep. Shift Greater than 48). If there is not such a mode, then the fix-up must be via interrupt. In the days of the fast one-word fixup, it was checked that everything was all right. If fix-ups always run the look-ahead dry, then things would still be all right if, and only if, a negative exponent spill indication is not given by the product of an ordinary number with negative exponent and a zero with exponent all ones. Similarly, no indication should be given when dividing a zero with exponent all ones by an ordinary number with positive exponent. The reason for this is that the decay of ordinary numbers to the bottom of the range is not nearly so frequent as the use of numbers which have already decayed to zero.

ask McSorley

A & B are split
and no put back as a ll
TMT
ask about

14. Why are accumulator bits 60-63 cleared in some single precision floating point operations? This can be irritating in some cases, ? and we should like to know whether it is really necessary. If it is not necessary, please explain why it is desirable, or else change things so as to leave 60-127 of the accumulator alone.

being changed

15. Is there any reason why bit 55 is not doing more work in the Transmit orders?

not an internal register

16. We are delighted to see a new full word internal register and two new indicators. The register could be a handy temporary storage spot, presumably fast, especially if it were equipped with its own Load from and Store to memory. It would also be tempting to fulfill one more piece of the contract and make it an indicator register (note that we would probably be ahead if we used bit 29 of Indicator Branch to select one of two Indicator Registers, instead of using it for turning off indicators; this would give us 64 more bits testable in a half-word, in exchange for requiring the use of a full word to test an indicator and turn it off). However, we do not understand the somewhat limited and peculiar use to which this new register is now being put. The whole decimal arithmetic business now looks hard to justify. There used to be a justification based on exact reproduction of accounting work, including rounding procedures, but this presumably disappears when conversion to binary is necessary for multiplication and division (if it doesn't disappear, then it was never there). Branching to general subroutines by means of interrupts seems quite unattractive.

commercial reasons

it does

fast as any branch

add takes place as usual turns on address invalid. if restore is also below 32 goes into loop.

17. What happens on interrupt if the Interruption Address is less than 32?

if gets [illegible] value no ind turned on

18. What happens if the interrupt table is in a protected area? Is the Instruction Fetch indicator set?

[illegible] to 0, loop

19. Is there any special protection for the boob who has an interrupt table of all binary zeros?

[illegible] α in acc too? old value in used $\alpha+\beta$ (talk to [illegible])

20. What happens when compound orders modify themselves? We could deduce the answers here if the descriptions in the Manual were precise as to sequencing. For example: at address α there is a VFL Store with Progressive Indexing; the value field of index I has α and the address field of the VFL Store has β; is the final value field of index I 2α or $\alpha+\beta$? Another example: in α there is a STICI to $\alpha+1/2$; in $\alpha+1/2$ there is a half-word Branch to β; do we branch to β all right? If so, suppose β is not protected, but $\alpha+1$ is; do we get any false indicators?

[illegible] as 1 instr yes

originally [illegible]

no IF

will do 1st ignore 2nd

21. What happens if you put a pair of half-word orders into the interrupt table? Are they both performed, or only the first? If both, what happens if the first causes an interrupt?

α [illegible] protected either ~~data~~ (DF) or instr (IF)

22. What happens if there is a data store to, or fetch from, $\alpha.63$, say, of a field longer than one bit, and α is not protected but $\alpha+1$ is? Or a full word instruction fetched from $\alpha+1/2$?

I apologize for the length of this letter. I will probably phone you on Thursday, April 2.

As ever,

Roger

Roger B. Lazarus

RBL:pae