Kolsky . Memo 9

November 20, 1958

Project 7000

FILE MEMORANDUM

SUBJECT: SIGMA Performance compared to other computers.

## I. Introduction:

In discussing the last SIGMA Timing Simulator memo (November 7, 1958) with various people, I realize that there is some confusion as to what a simulated comparison of internal computing speeds of two different machines really is. As a result perhaps it would be worth while to draw a comparison in detail for the purpose of showing how difficult it is to compare computers with different organizations in general, yet how simple it is to compare them on specific jobs. The Timing Similator code does no more than this, it just works out a larger sample of jobs for more variations of assumptions.

## II. General Limitations on Computer Speed

In machines of the von Neumann stored program type, the execution of a typical indexed, data fetching instruction has the following steps which must be done in sequence; (single address instructions assumed).

1. advance instruction counter
2. send fetch request to instruction memory
3. instruction memory reads out
4. decode instruction
5. perform index addition, if required
6. send fetch request to data memory
7. data memory reads out
8. perform arithmetic operation.

Each of these steps takes time, however. Only the last item represents "useful" work. The other steps although they are necessary may be considered as "parasitic" operations.

There are essentially two ways of speeding up a computer: (1) build it out of faster components, and (2) make its organization more efficient. The latter is our effort to reduce the number of logical levels in the arithmetic operations and to cut down the percentage of time spent on the parasitic operations.

Since the above steps must be done in sequence, getting 100% useful time would seem to be impossible. However by overlapping the starting of instructions with the execution of earlier ones it is theoretically possible to overlap all the parasitic operations so that they are done in "no time".

The 704 and 709 are examples of machines where the above steps are done sequentially with no overlap. (Actually, there is some overlap -- the instruction counter is advanced as part of the previous instruction.)

The 7090 is an example of increasing machine performance by using faster components with no change in organization.

LARC and STRETCH both use faster components and overlapped organization to achieve increased performance. Roughly speaking, STRETCH has about four times as much overlap as LARC.

Making simple comparisons between machines with overlap is difficult, however one can still learn much by looking at maximum and average rates for similar operations.

## III. Memory Speed Limitations

It has been often said, "Specify the memory, and you specify the machine." Of course this is an over-simplification of the situation, but it is still true that if one knows the memory speed and the degree of overlapping to be attempted, many of the other details of the machine are also specified.

In Table I, there are listed maximum rates at which instructions and data may be furnished to various computers. These, especially the instruction rate, limit the ultimate performance of the machines on short operations. Having separate data and instruction memories in LARC doubles the effective rate of the memories. STRETCH gains another factor of 4 by having 2 instructions per word with 2 memory boxes for instructions, and 4 memory boxes for data.

If the rates at which instructions or data can be fetched are lower that the rate at which the instructions can be executed in the arithmetic unit, then the machine would be strictly memory limited. Fortunately none of the machines listed in Table I are in this category.

The column "old STRETCH" is included to compare our estimates of a year or two ago with present realities.

Table I: Memory cycle times and maximum possible Data Rates and Instruction Rates for various Machines, including the effect of Overlap.

| | 704 | 7090 | LARC | SIGMA | "old STRETCH" |
|---|---|---|---|---|---|
| Instr. Memory cycle time | 12. usec | 2. 4 usec | 4. usec | 2. 2 usec | 0. 6 usec |
| Data Memory cycle time | 12. | 2. 4 | 4. | 2. 2 | 2. 0 |
| Maximum Instruction Rate (times 704) | 1. | 5. | 6. | 43. 6 | 160. |
| Maximum Data Rate (times 7 04) | 1. | 5. | 6. | 43. 6 | 48. |

## IV. Indexing Arithmetic Speed Limitations.

In sequential machines such as the 704, one often speaks of the indexing time being "buried" in the instruction, giving the impression that no time is required for it. Actually the true situation is that one always spends this time whether it is used or not. Indexing is much more truly buried in an overlapped design even though one is more conscious of the time it takes.

Table II lists the indexing rates for the above machines. For the 704 and 7090 the time listed is the "I cycle" time. Their actual indexing time is about 1/4 of this, (See Section VI.)

Table II: Indexing Times and Speeds for various Machines.

| | 704 | 7090 | LARC | SIGMA | "old STRETCH" |
|---|---|---|---|---|---|
| IAU Time for non-indexed ops. | 12 usec | 2. 4 us | 3 usec | 0. 4 usec | 0. 1 usec |
| IAU Time for indexed ops. | 12. | 2. 4 | 3 usec | 0. 8 | 0. 2 |
| IAU Time for index modifying ops. | 24. | 4. 8 | 4 usec | 1. 6 | 0. 3 |
| IAU Speed for non-indexed ops. | 1. | 5. | 4. | 30. | 120. |
| IAU Speed for indexed ops. | 1. | 5. | 4. | 15. | 60. |
| IAU Speed for index modifying ops. (All times 704) | 1. | 5. | 6. | 15. | 80. |

## V. Arithmetic Unit Speed Limitations

As was mentioned above, the arithmetic unit represents the real useful work of the computer. Instruction sets vary considerably in complexity from one machine to another.

Machines which are aimed at scientific problems may best be rated in terms of their basic floating point arithmetic times. One may argue justly that this is a very narrow view point on which to compare complete systems, but recall that many people use only the floating multiply speeds to make the comparison.

Table III lists the arithmetic times and rates for the common floating point operations.

Table III. Floating Arithmetic Times and Speeds for various Machines.

| | 704 | 7090 | LARC | SIGMA | "old STRETCH" |
|---|---|---|---|---|---|
| Time for Load Store | 24. usec | 4. 8 usec | 4. usec | 0. 4 usec | 0. 2 usec |
| Time for Add | 84. | 16. 8 | 4. | 1. 0 | 0. 6 |
| Time for Mpy. | 204. | 40. 8 | 8. | 2. 4 | 1. 2 |
| Time for Div. | 216. | 43. 2 | 28. | 7. 5 | 1. 8 |
| Time for 6-6-3-1 Ave. | 92. 25 usec | 18. 45 usec | 6. 25 | 1. 44 usec | 0. 64 |
| Load, Store Speed | 1. | 5. | 6. | 60. | 120. |
| Add Speed | 1. | 5. | 21. | 84. | 140. |
| Mpy Speed | 1. | 5. | 25. 5 | 85. | 140. |
| Div. Speed | 1. | 5. | 7. 7 | 29. | 120. |
| Ave. 6-6-3-1 Speed | 1. | 5. | 14. 8 | 64. 1 | 144. |

(All times 704)

## VI. Comparison of the Computers on an actual Problem

If one studies the above tables, the reason why it is hard to quote a simple speed ratio for the computers becomes obvious. For example, SIGMA ranges from 15 to 85 times the 704 depending on the quantity compared. (This comparison difficulty does not exist for the 7090 which has a simple 5 times performance on everything.)

Clearly the only way to really compare the machines is on the basis of the time each takes to do a given job. To avoid discussions of different op. codes, I/O comparisons, etc., I have chosen a familiar old problem, the inner loop of a matrix inversion program. (Load, Mpy, Add, Store, TIX). This program is a short loop but heavy on arithmetic. It gives a good indication of performance both on instruction fetching and arithmetic.

The graphs at the end give the detailed timing charts on this problem for each of the computers being discussed.

Notice that I have purposely avoided memory conflicts in the STRETCH charts, so they represent the ideal best case.

Table IV lists some memory data concerning the performance as represented on the charts. (The times listed for the 704 and 7090 are the actual times spent on each of the jobs not the "I-cycle" times listed before.)

Table IV. Performance Data from the timing charts for the various computers on the Matrix Inversion Problem. (Refer to the timing charts on back for details.)

| | 704 | 7090 | LARC | SIGMA | "old STRETCH" |
|---|---|---|---|---|---|
| Initial Start-up time | 20 usec. | 4 usec. | 9 usec | 5. 8 usec | 2. 6 usec |
| Arithmetic Time per cycle | 248. | 49. 6 | 20. | 4. 2 | 2. 0 |
| Indexing Time per cycle | 20. | 4. | 15. | 5. 2 | 1. 0 |
| Branch Time per cycle | 50. | 10. | 8. | 2. 6 | 0. 1* |
| Total Cycle Time | 360. | 72. | 28. | 7. 2 | 2. 2 |
| Initial Start-up Speed | 1. | 5. | 2. 2 | 3. 5 | 7. 7 |
| Arithmetic Speed | 1. | 5. | 12. 4 | 59. 0 | 124. 0 |
| Indexing Speed | 1. | 5. | 1. 3 | 4. 8 | 20. 0 |
| Branch Speed | 1. | 5. | 6. 3 | 19. 2 | 500. * |
| Total cycle Speed | 1. | 5. | 12. 9 | 50. 0 | 164. 0 |
| (All times 704) | | | | | *(time is overlapped) |

Note: "Total cycle Speed" represents the overall internal computer performance.

## VII. Discussion of Importance of Overlap

A good feeling can be had of the relative importance of two factors, increased circuit speed and improved organization, by comparing the following three machines against the 704:

(a) 7090
(b) LARC[1] -- a hypothetical LARC built of STRETCH circuits and memories
(c) SIGMA

Assuming that LARC[1] is faster than LARC by the ratio of their memory cycles, (2. 2 usec to 4 usec) one can construct Table V:

Table V: Comparison of Speeds of 7090, SIGMA, and LARC[1] (LARC with STRETCH circuits and memories) in terms of 704 speed.

| | 7090 | LARC[1] | SIGMA | Ratio: Sigma/LARC[1] |
|---|---|---|---|---|
| Fundamental Machine Speeds (including effect of overlap): | | | | |
| Max. Instruction Rate | 5. | 10. 9 | 43. 6 | 4. 00 |
| IAU Speed for indexed ops. | 5 | 7. 3 | 15. 0 | 2. 05 |
| AU Speed (6-6-3-1 ave.) | 5. | 26. 9 | 64. 1 | 2. 38 |
| Performance on Matrix Inversion Problem: | | | | |
| Start-up Speed | 5 | 4. 0 | 3. 5 | 0. 88 |
| IAU Speed | 5 | 2. 4 | 4. 8 | 2. 00 |
| AU Speed | 5 | 22. 5 | 59. 0 | 2. 62 |
| Branching Speed | 5 | 11. 5 | 19. 2 | 1. 67 |
| Total Job Speed | 5 | 23. 5 | 50. 0 | 2. 13 |

We now have three different machine organization all built from the same circuit elements. (Note: Mr. E. W. Coffin has examined a speeded up TRANSAC under the same assumptions in a Memo dated November 18, 1958 and found it to be about 11 times 704 Speed on this problem.)

The 7090 is not overlapped, so the factor of 5 represents circuit speed increase only. The LARC[1] has a factor of 2 overlap (instruction preparation and execution). SIGMA has a factor of 8 overlap (these two plus 4 from memory box overlap.)

The picture is complicated somewhat by the fact that the 7090 floating arithmetic is disproportionately slow for the rest of its capability, so LARC[1] picks up a factor of 4.7 in total job speed for only a factor of 2 in overlap.

The factor of 4 in overlap between SIGMA and LARC[1] nets SIGMA a factor of 2.13 increase in speed. This is not bad in view of the fact their arithmetic speeds differ only by a ratio of 2.62.

Overlap by its nature has certain limitations which prevent one from achieving its full factor of speed increase. For Data, the memory conflicts which occur when the data references to 4 boxes occur at random cut the average peak data rate to 40% of its value with no conflicts. (For two boxes the average is 75% of the maximum.) Fortunately, this is not as serious as it sounds since every instruction doesn't require data fetches, and the average time is still less than the average arithmetic time (1.37 usec compared to 1.44 usec for SIGMA.) The Look-Ahead smooths out the data references in time so that the average rate is the one which counts. Also the arithmetic unit is rarely busy more than 75% of the time.

For instructions there are two types of conflicts, memory conflicts and the loss of one or the other half of the two instruction word on a branch. The worst case is a series of branches always staying in one memory box which completely destroys the factor of 4 overlap.

Fortunately, instructions are rarely of this type. The computer will usually take the next instruction in order and branch only rarely. The average rate differs considerably from problem to problem, but a value between 50% and 80% of the theoretical maximum is typical.

This can still be serious because every instruction must have an instruction fetch and there is only limited Look-Ahead action on instructions. Time delays on instruction fetches tend to add directly to the problem time much more than data fetches do. Another penalty of overlap is the increased time required on initial start-up. This is apparent in the values listed for "Start-up Speed" in Table V.

In spite of these difficulties, overlapping clearly pays off in the performance of SIGMA. It does cost in increased hardware and machine complexity, but this will have to be evaluated separately.

## VIII. Discussion of "old STRETCH" vs. present SIGMA

Perhaps the most disturbing part of the comparisons in Tables I thru IV are those between SIGMA and the "old STRETCH" as pictured in the hand-drawn timing charts of a year or two ago. What are the reasons for this factor of 2 or 3 reduction in performance? The following seem to be the main causes:

(1) The fundamental transistor circuit speeds are slower by at least a factor of 2 than those originally postulated.

(2) The memories are all slower -- particularly the index registers. Another example, the read-out time of the 2 usec memory is presently 1.4 usec instead of 0.8 usec.

(3) The early arithmetic speed estimates were unrealistic even with the proposed circuit speeds. The "1.8 usec divide" is particularly hard to explain.

(4) The "old STRETCH" estimates were really based, perhaps unconsciously, on much simpler designs than the present ones. Nothing resembling the intricacies of the Interrupt system hardware, the VFL arithmetic, nor the present I-Box interlocks were ever considered in giving the "0.2 usec indexing time".

The fact that the overall performance has dropped only by a factor of 3 in view of these difficulties is greatly to the credit of the engineers.

Clearly it does no good to rail against over-optomistic early estimates nor to complain about present troubles. The honor and prestigue of the company is at stake in the performance of STRETCH. We must hold the line everywhere and wherever possible push it higher.

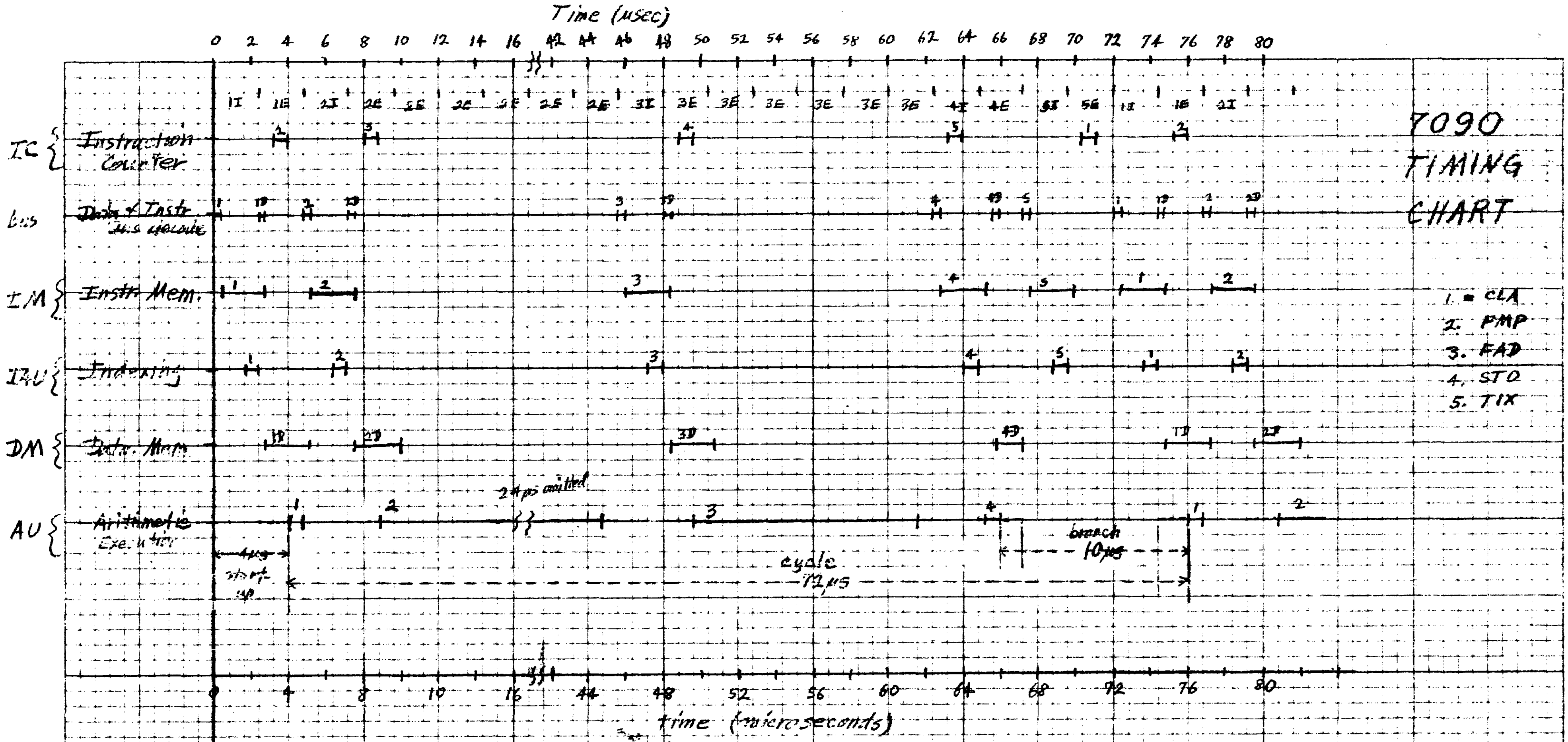
H. G. Kolsky
Product Planning Representative
Project 7000

HGK:jcj

cc: Dr. W. Buchholz
Mr. D. W. Sweeney
Mr. E. Bloch
Mr. R. E. Merwin

Pendery
Dunwell

# 7090 TIMING CHART

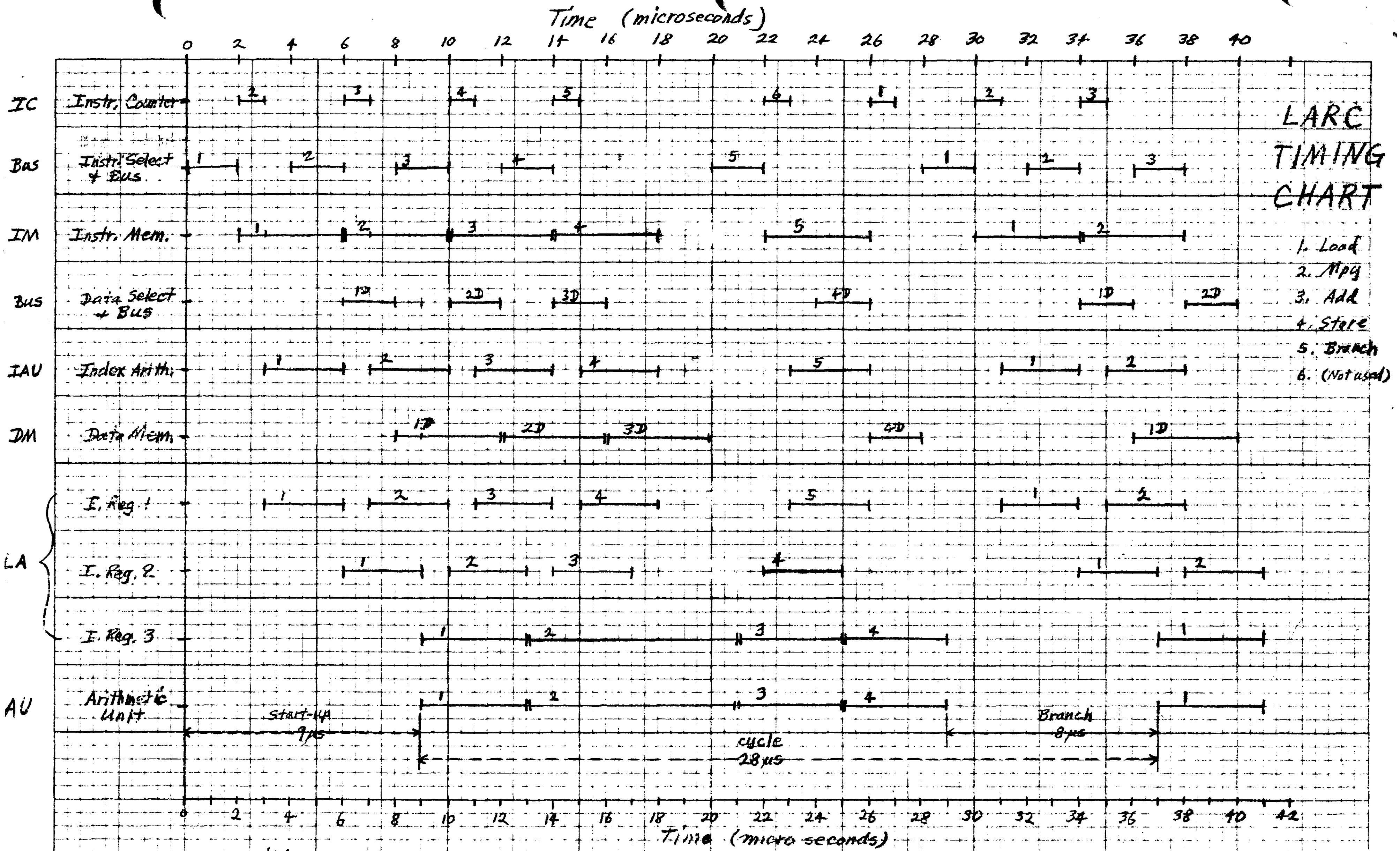


| | |
|---|---|
| Initial Startup | = 4. μs |
| Total Cycle | = 72 μs |
| AU Time per cycle | = 49.6 μs (+12.4 "lost time") |
| IAU Time per cycle | = 4. μs |
| Branch Time per cycle | = 10 μs. |

Note: The 7090 uses the same arithmetic unit for IC, IAU, and AU. Also IM and DM are the same. They are drawn on separate lines here for comparison with STRETCH.

The charts for the 709 or 704 are the same with all times multiplied by 5.

Nov 13, 1958 HGK



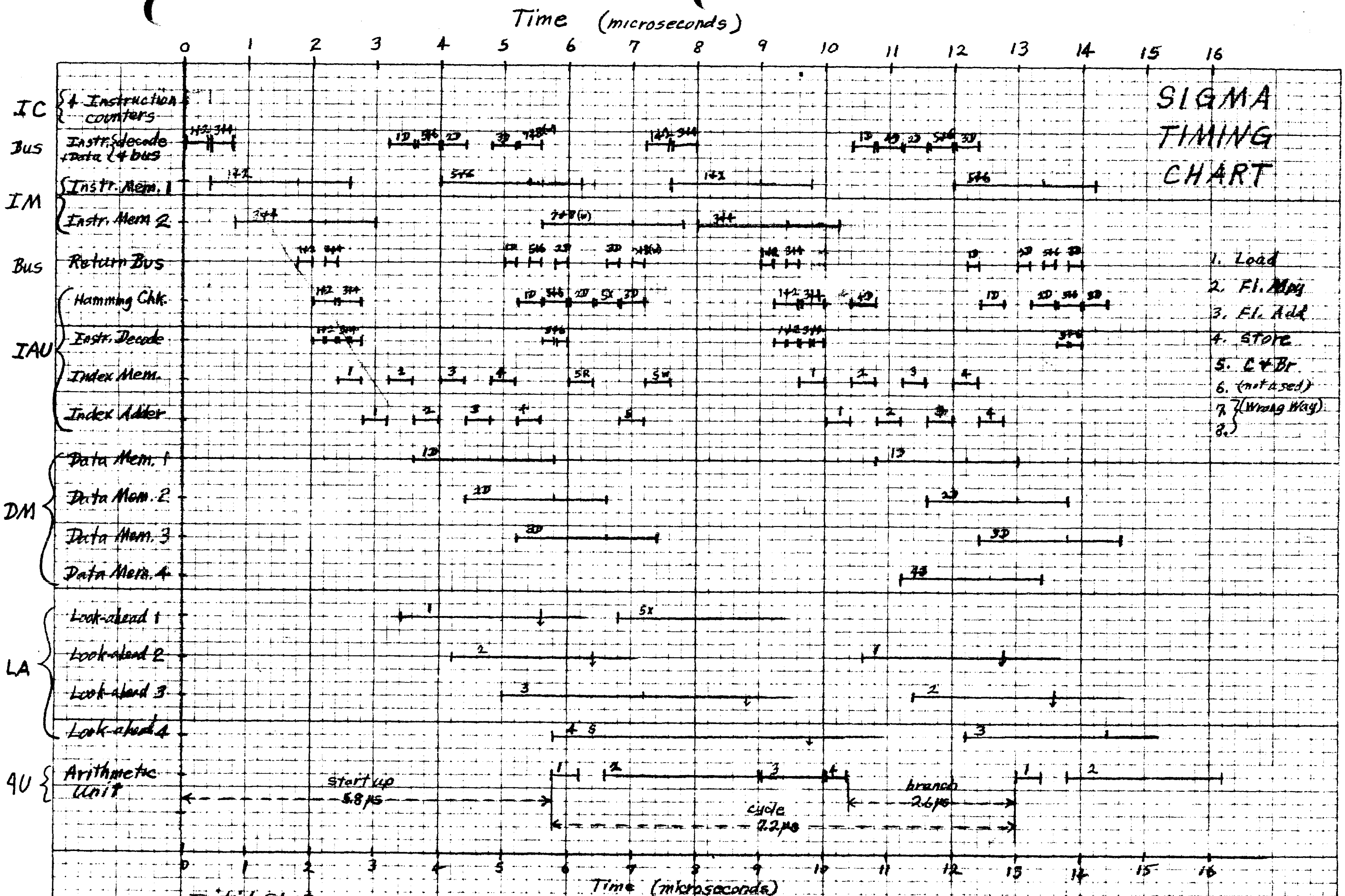
# LARC TIMING CHART

1. Load
2. Mpy
3. Add
4. Store
5. Branch
6. (Not used)

| | | |
|---|---|---|
| Initial Startup | = | 9. µs |
| Total Cycle | = | 28. µs |
| AU time per cycle | = | 20. µs |
| IAU Time per cycle | = | 15. µs |
| Branch time per cycle | = | 8. µs |

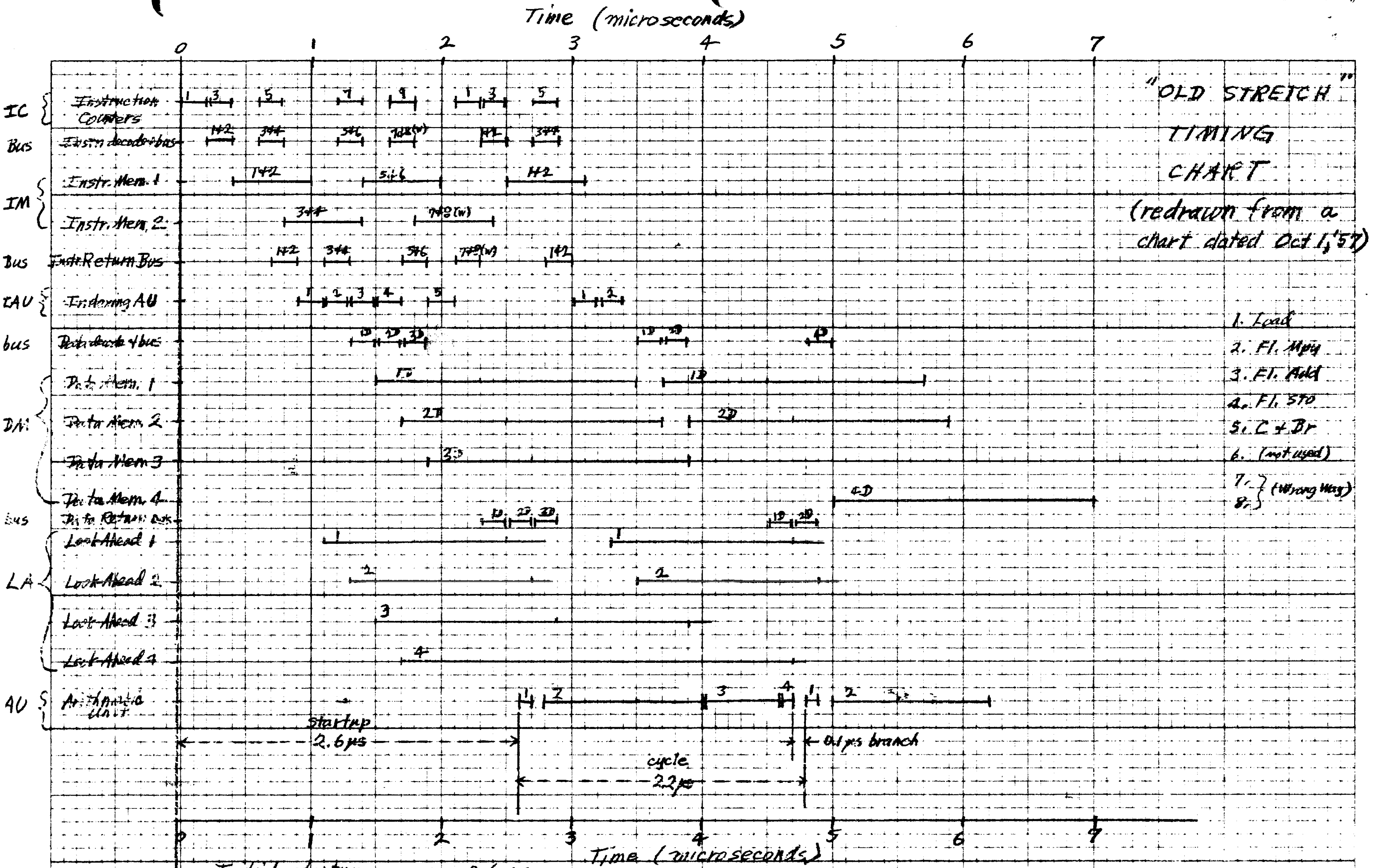
Nov 15 '58 HGK

# SIGMA TIMING CHART



Initial Startup = 5.8 µs
Total Cycle = 7.2 µs
AU time per cycle = [illegible] µs
IAU Time per cycle = 5.2 µs
Branch time per cycle = 2.6 µs

Nov 13, 58 HGK



"OLD STRETCH" TIMING CHART

(redrawn from a chart dated Oct 1,'57)

1. Load
2. Fl. Mpy
3. Fl. Add
4. Fl. STO
5. C + Br
6. (not used)
7. } (Wrong Way)
8. }

| | | |
|---|---|---|
| Initial startup | = | 2.6 µs |
| Total cycle | = | 2.2 µs |
| AU time per cycle | = | 2.0 µs |
| IAU Time per cycle | = | 1.0 µs |
| Branch Time per cycle | = | 0.1 µs |

Nov 15,'58 HGK