

August 29, 1958

PROJECT 7000  
FILE MEMO

SUBJECT: The Effect on SIGMA Performance of the Half-Microsecond Instruction Memory.

1. Introduction:

Recently a series of SIGMA Timing Simulator runs have been made to evaluate the present status of the SIGMA and HARVEST computers. One parameter studied was the speed of the instruction memory. Several runs were also made in which there was no separate instruction memory but instructions and data were stored in the same boxes.

The results of these runs were such that they seemed to warrant this separate report evaluating the importance of the half-microsecond memory to the STRETCH program.

2. Advantages and Disadvantages of the Half-Microsecond Memory:

A. The primary advantage of the half-microsecond memory is, of course, its speed. This speed is beneficial in the following two cases:

- (1) In programs which are instruction access limited, either because they consist of a series of short operations, or because they contain many branch orders, the half-microsecond memory used for instructions will help reduce the limitation by furnishing the instructions faster.
- (2) In programs which are data-access limited, putting the data in the faster memory will cut down the time required for fetching the data. VFL operations with short fields are in this category. (Ref: Project 7000 File Memo dated June 30, 1958 on the HARVEST Simulation Studies)

In both of the above cases it is the speed of the memory compared to the arithmetic speed which is the important ratio--the faster the arithmetic speed the faster the memory required to service it properly.

B. The main disadvantage of the half-microsecond memory is its size. Each memory box contains only one-sixteenth as many words as a comparable two microsecond memory box. This decreased size certainly must result in reduced performance because more time will be spent reallocating programs. Unfortunately this reduction cannot be evaluated quantitatively by simulation since it depends on the nature of the future problems, and on the nature of future methods of scheduling machine use.

There is another advantage in larger memories which is even harder to evaluate and that is the removal of programming restrictions which exist when programs must be cut to fit a small memory.

### 3. Test Calculations Used:

The test problems tried were the same five which have served as guinea pigs in many of our past runs. They were originally selected as being typical of different classes of problems. A brief description of each is repeated here for completeness.

- (1) Mesh Problem - Part of an hydrodynamics problem from Los Alamos. It contains a more or less "average" mixture of instructions for scientific problems: 85% Floating Point instructions, 14% index modification instructions, and 1% VFL. It is usually arithmetic unit limited.
- (2) MonteCarlo Branching Problem - Part of an actual Monte Carlo neutron diffusion code. It represents a chain of logical decisions with very little arithmetic in between. It contains 47% Floating Point, 15% index modification instructions, and 36% branches of the indicator and unconditional types. It is largely instruction-access limited.
- (3) Reactor Problem - The inner loop of a neutron diffusion problem from Westinghouse. It consists of 90% Floating Point arithmetic (39% of which are multiplies) and 10% index modification instructions. It is almost entirely arithmetic unit limited.

August 29, 1958

- (4) Computer Test Problem - The evaluation of a polynomial using computed indices. It was prepared by I. Ziller to compare various computers. It has 71% Floating Point, 10% index modification, 6% VFL and 13% indicator branches. It is usually arithmetic unit limited but not for all configurations.
- (5) Simultaneous Equations - The inner loop of a matrix inversion routine 67% Floating Point and 33% index modification. Arithmetic and logic are about equally important. It is limited both by arithmetic and instruction-access speeds.

#### 4. Simulator Input Data :

The above problems were run with most of the recent design changes simulated, including the 0.8 microsecond I-Box repetition rate and the 0.2 microsecond bus slots. The arithmetic speeds used were:

	<u>STANDARD</u>	<u>SIGMA</u>	<u>HARVEST</u>
Load, Store	0.2 us	0.4 us	0.4 us
Floating Add	0.6	1.0	1.0
Floating Multiply	1.2	2.5	7.5
Floating Divide	1.8	7.0	7.5
6-6-3-1 average	<u>0.64</u>	<u>1.43</u>	<u>2.40</u>

Standard are essentially the times given in the AEC contract. The SIGMA times are unofficial present estimates. The HARVEST times are those estimated for the version of the computer being built for BuShips. They were all run with the 0.8 us Indexing Arithmetic Unit rate (including the STANDARD cases). The average times are used for convenience of plotting only.

#### 5. Results:

Results of some of the runs are given in Table I. A short summary of the pertinent results are given in Table II.

Straight averages of the percentage losses do not tell the whole story. There are abrupt changes in behavior for some of the problems from one case to another. Upon examination, the reason in each case was due to the problem becoming instruction-access limited where it had previously been arithmetic limited. Each problem crosses over under different circumstances because of its own particular combination of instructions.

Table II also lists the programs which seem to be instruction-access limited for each memory and arithmetic speed configuration.

The phenomena which has been observed so many times before, still holds here---the higher the machines overall performance, the more sensitive it becomes to each individual component's performance. Thus, all of the problems are prone to become instruction-access limited at STANDARD speeds, where only the faithful Monte Carlo code is limited at HARVEST speeds.

Graph I shows the percentage variations graphically for each configuration. The magnitude of the losses must be considered as well as the pattern. Clearly the memory interferences caused by not having a separate instruction memory is as large or larger than the speed of the memory. The average percentages are given in table III.

6. Rough Estimate of the effect of having a larger instruction memory on Computer Speed:

As was mentioned in section I, the favorable speed advantage gained by having a larger instruction memory is hard to assess quantitatively. The following is intended to be a rough order-of-magnitude estimate only.

In a given time T, assumed to be long enough to do several problems, the computer will divide its activities between the time spent on useful calculation and the time spent on swapping codes in and out of instruction memory. We may write

$$T = n \cdot t_c + n R t_c = n t_c (1 + R)$$

where n = the number of useful instructions executed

R = the ratio of the number of words swapped per useful instruction executed. (R should be much less than 1)

$t_c$  = average time per calculation executed.

(For simplicity the time for swapping an instruction is taken the same as  $t_c$ .)

The speed of the computer, S, is proportional to  $n/T$ , the number of useful operations per unit time. So we may write the ratios of the speeds of two systems as:

$$\frac{S_2}{S_1} = \frac{n_2}{n_1} = \frac{t_{c1} (1 + R_1)}{t_{c2} (1 + R_2)}$$

August 29, 1958

The  $tc_1 / tc_2$  factor is the regular speed-up caused by the faster memory. The term involving the  $R$ 's is the new factor resulting from the effect of swapping codes. As a guess, we can take  $R$  as being inversely proportional to the memory size, so that

$$R_1 = R_2 \frac{N_2}{N_1}$$

also, since the  $R$ 's are both much less than 1, we may write

$$\frac{S_2}{S_1} = \frac{tc_1}{tc_2} (1 + R_2 \frac{N_2}{N_1} - R_2)$$

In the present case, consider a 10% computer speed differential on  $tc$ 's between the half and two-microsecond memories, which differ in size by a 1 to 16 ratio. We can ask what value of  $R_2$  will be necessary to make the half microsecond memory result in an increase in speed over the two microsecond memory. The answer is approximately:

$$R_2 \leq 0.1$$

That is, each instruction in the half microsecond memory must be used at least 10 times in an average program before it is replaced in order that the half microsecond memory show a net increase in speed over the larger, slower 2 us memory.

Very roughly speaking, each instruction must be used at least once for each percent loss in speed under the configurations tested here to break even. It seems likely that this condition will be easily satisfied in practice, so that the faster memory will indeed result in a faster computer even though part of its advantage is lost.

The other factor mentioned which favors larger memories is the effect of being able to write less complicated codes when they need not be cut to size. One can express this factor as a  $(1 + f)$  term times the speed of the computer to give its effective speed. This speed gain is because the machine has to do a fraction  $f$  fewer instructions to accomplish the same job with a larger memory as it would take with the smaller. Since this fraction is so strongly a function of the problem involved, one can only guess what it will be as an average for all SIGMA problems. It should be in the 0 to 10% range, however.

**7. Conclusions:**

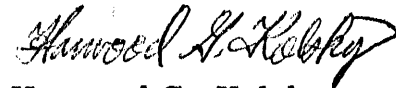
- (a) Whether a problem is instruction-access limited or not is the main property which determines its behavior under changes in instruction memory.
- (b) The property of being instruction-access limited depends considerably on the individual sequence of instructions in a problem itself, and on the relative speeds of the arithmetic unit and the instruction memory.
- (c) The higher the performance of the computer, the more sensitive is its speed to changes in instruction memory configuration. At the present SIGMA speeds, replacing the two 0.6 us memory boxes by two 2.0 us memories results in an average of 2.5% loss in performance in the cases tested.
- (d) At present SIGMA speeds, intermixing data and instructions causes an average loss of 3.9% in performance over having a separate 2.0 us instruction memory. This is because conflicts between data and instructions delay instruction accesses. Note that this is larger than the effect of memory speed itself.
- (e) The speed gains from having a faster memory are reduced somewhat by the fact that it is smaller and more time must be spent swapping codes. This seems to be a small effect timewise, however.

The effective performance increase possible because bigger programs may be put into the larger memory at once is hard to assess. It is probably also in the 1 to 10% area.

August 29, 1958

- (f) The three main factors resulting in the present insensitivity of SIGMA to instruction memory speeds are: (1) the two instructions per word feature, (2) the present slow arithmetic speeds, and (3) the actual small differences in times between the "half" and "two" microsecond memories.

HGK:jcv  
Attachment



Harwood G. Kolsky  
Product Planning Representative  
Project 7000

cc: 7000 Product Planning  
7000 Engineering Planning  
Mr. D. W. Pendery  
Mr. S. W. Dunwell  
Mr. R. E. Merwin  
Mr. H. A. Mussell  
Mr. H. K. Wild  
Mr. J. H. Pomerene  
Mr. J. J. Kenney, Jr.  
Mr. E. D. Foss  
Dr. J. Cocke  
Mr. E. Bloch  
Dr. P. S. Herwitz

GRAPH 1.

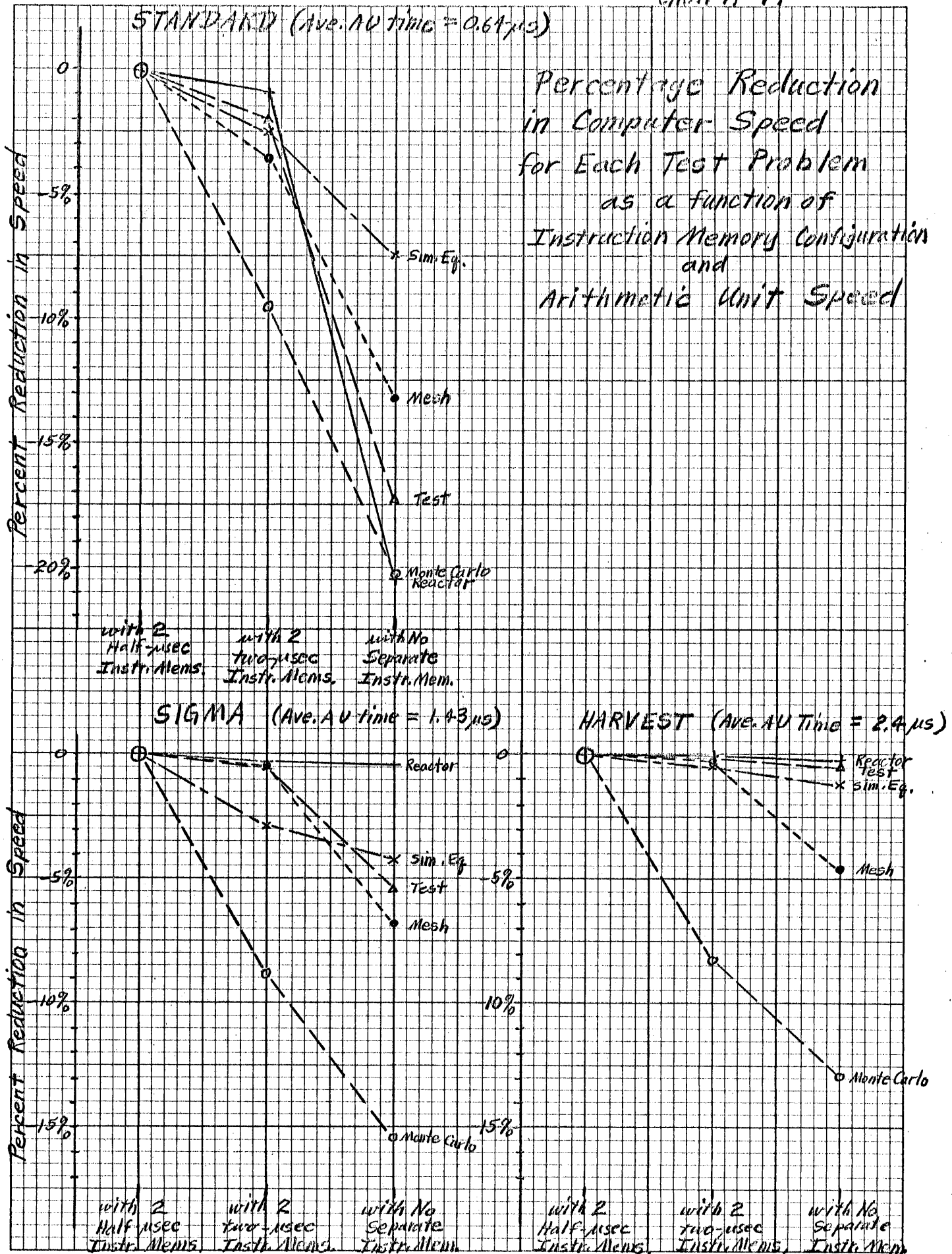




TABLE I

Computer Performance as functions of Memory Configurations and Arithmetic Unit Speeds.

## SPEED OF TEST PROBLEMS (times 704 Speeds)

Configuration Instruction Memories	Mesh		Monte Carlo		Reactor		Computer Test		Simultaneous Equations	
	Speed	%	Speed	%	Speed	%	Speed	%	Speed	%
<u>STANDARD</u>										
1. 2 0.6 us Mem.	85.9	0	45.5	0	122.4	0	89.5	0	48.7	0
2. 2 2.0 us Mem.	82.9	-3.5	41.2	-9.5	121.3	-0.9	87.6	-2.1	47.5	-2.5
3. No. Instr. Mem.	74.7	-13.1	36.3	-20.2	97.5	-20.3	73.2	-18.3	45.0	-7.4
<u>SIGMA</u>										
1. 2 0.6 us Mem.	59.2	0	41.9	0	75.9	0	58.1	0	45.1	0
2. 2 2.0 us Mem.	59.0	-0.4	38.2	-8.8	75.7	-0.3	57.9	-0.4	43.8	-2.8
3. No. Instr. Mem.	55.2	-6.8	35.4	-15.4	75.6	-0.4	55.0	-5.3	43.2	-4.2
<u>HARVEST</u>										
1. 2 0.6 us Mem.	39.3	0	38.8	0	38.0	0	40.8	0	31.7	0
2. 2 2.2 us Mem.	39.2	-0.2	35.6	-8.2	37.9	-0.1	40.7	-0.3	31.6	-0.4
3. No. Instr. Mem.	37.5	-4.6	33.8	-12.9	37.9	-0.2	40.7	-0.3	31.3	-1.2

Each has 4 0.6 microsecond total cycle instruction memory  
2.0 microsecond total cycle data memories

TABLE II

Summary of Results: Average Computer speed changes caused by Instruction memory speeds and Arithmetic Speeds, straight averages for all five test problems.

<u>STANDARD AU Speeds</u>	<u>Average Percent Decrease</u>	<u>Problems*which are Instr. - access limited</u>
1. 2 1/2 us Mems.	0	(2)
2. 2 2 us Mems	-3.7%	(2) (4) (5)
3. No. Instr. Mem.	-15.9%	(1) (2) (3) (4) (5)
 <u>SIGMA AU Speeds</u>		
1. 2 1/2 us mems	0	(2)
2. 2 2 us Mems	-2.5%	(2) (5)
3. No. Instr. Mem.	-6.4%	(1) (2) (4) (5)
 <u>HARVEST AU Speeds</u>		
1. 2 1/2 us Mems.	0	(2)
2. 2 2 us Mems.	-1.8%	(2)
3. No. Instr. Mem.	-3.8%	(1) (2)

\*The Problem numbers are those given in Section 3.

TABLE III

Average Percentage Losses for all problems.

	<u>Arithmetic Speeds</u>		
	<u>STANDARD</u>	<u>SIGMA</u>	<u>HARVEST</u>
Ave loss caused by replacing 0.6 us Instr. Memory by 2.0 us Memory.	-3.7%	-2.5%	-1.8%
Average additional loss caused by having no separate Instr. Memory.	-12.2%	-3.9%	-2.0%
Maximum loss caused by replacing 0.6 us Instr. Memory by 2.0 us Memory	-9.5%	-8.8%	-8.2%
Max. additional loss caused by having no separate Instr. Memory.	-19.5%	-6.6%	-4.7%