DRAFT

PROJECT MAC May 4, 1973

Computer Systems Research Division Request for Comments No.19

MEASUREMENTS OF HARDWARE SPEED OF THE 6180

by J. H. Saltzer

In comparing the performance of the 6180 Multics system with that of the 645 Multics, there are several changes in environment, each of which can be expected to affect performance. If one expresses the performance of the new system as a ratio relative to the old, it is then appropriate to separate out each of the environmental changes as a factor contributing to this ratio. Thus if we designate the overall performance improvement by F, we have

$$F = f_1 \cdot f_2 \cdot f_3 \cdot f_4$$

where f_1 = performance improvement due to speed up of the raw hardware

f_2 = performance improvement due to addition of ring crossing hardware

f_3 = performance improvement due to addition of EIS instructions

f_4 = performance improvement due to replacement of firehose drum with bulk store.

Such a separation of effects is useful, since it is probably possible to separately measure each of the effects, as well as the overall performance improvement, and thereby gain a cross-check as to whether or not the overall performance is correctly understood.

This memo reports an intensive series of measurements recently undertaken on the MIT 6180 to estimate the value of f_1, the effect of the raw hardware speed. In summary, these measurements indicate that:

1. When running a typical mix of Multics programs, the 6180 processor will run about 1.8 times as fast as the 645.
2. When running 6070 (non-EIS) programs, the 6180 processor will run about 95% as fast as a 6080 processor. Some EIS instructions are currently slower.
3. The instruction execution rate of the 6180, when running Multics programs, will be about .66 million instructions per second.

It should be noted that the MIT 6180 is operating under an intentional, but temporary, speed handicap estimated to be about 10%, in order to reduce the chance of associative memory errors (which are exceptionally hard to diagnose) during initial hardware shakedown. The numbers quoted above are predictions of the speed after the handicap is removed. Currently measured numbers are about 10% smaller.

The measurements

The basic measurements were made using an assembly language program and the microsecond calendar clock. The clock was read, a sequence of instructions executed, and the clock read again. The number of instructions between clock readings was adjusted to require between 500 and 600 microseconds of execution time, so that the measurements would be precise to within 1 part in 500, or 0.2%, on both the 645 and the 6180. After some initial measurements, it became apparent that different classes of instructions had been affected differently in the move from the 645 to the 6180, so a series of measurements on individual pure instruction classes was undertaken. In general, these were accomplished by placing 28 identical instructions in a sequence, then adding a loop index and a conditional transfer back at the end, and loading an index register with an appropriate value before entering the sequence. Thus a typical test run consisted of

```
load index register one
read clock
28 identical instructions
eaxl  -1,1
tnz   -29,ic
read clock
```

The test sequence was repeated 10,000 or 20,000 times, and the smallest observed running time of the sequence was taken to represent the maximum speed of the processor in executing the sequence.

Using this technique, the instruction execution times of table I were observed. The first column of numbers is simply the measured instruction time, in nanoseconds, on the 645. The second column is the corresponding number for the MIT 6180. The third column is obtained from the second column by subtracting 100 nanoseconds, an estimate of the amount of associative memory slow down. (The value of 100 nanoseconds was picked

	current 645 time	current 6180 time	adjusted 6180 time	book 6080 time	adjusted 645/6180 ratio	adjusted 6080/6180 ratio
ada	1920	832	735	700	2.62	0.95
sta	2250	1140	1038	1000	2.17	0.96
eapbp	2970	1505	1408		2.11	
eapbp*	5320	2960	2763		1.93	
stpbp	2870	2230	2133		1.35	
sarl		2230	2130	1660		.77
larl		1990	1893	1660		.87

TABLE I: Instruction timings for 645, 6180, and 6080. All times in nanoseconds.

* With indirect address. Adjusted 6180 time assumes two 100 nanosecond delays are removed.

---

since it results in an ada speed which is 95% of that of the 6080, which is the target value.) The fourth column contains the "book" time for the 6080, for which the ada and larl instruction times were verified on a 6080. The fifth column reports the speed ratio of the 645 compared with the (adjusted) 6180. The last column reports the 6080 to 6180 speed ratio.

As is apparent, not all instructions have been sped up by the same amount. The primary reason for the difference seems to be that although the nominal memory speed of the 6180 (500ns) is half that of the 645 (1000ns), the data access times observed inside the processor, after cable propagation and settling time, are 660 and 1100ns, respectively. Thus, although much of the basic CPU logic is more than twice as fast as the 645, the memory access time is only 1.7 times as fast. The effect of the memory access time is especially apparent on those instructions for which execution is not overlapped with address preparation for the next instruction, and also those using indirect addressing. Apparently, as a design simplification, all instructions which load or store the pointer registers have address preparation overlap inhibited.

A special mystery surrounds the timing of the stpbp and sarl instructions. They have not sped up significantly from the 645, yet they run 77% of the 6080 speed, which means that both the 6080 and the 6180 provide a slow implementation. The time of the 6080 sarl instruction is exactly that of an sta instruction (1000ns) with address preparation overlap inhibited (addition of 660ns). On the other hand, the 645 stp instruction time (2870ns) is much less than an sta instruction (2250ns) with address preparation overlap (addition of 1100ns) inhibited. Thus apparently the 645 uses some trick to obtain some address preparation overlap on the stp instruction. There are actually two mysteries:

1. Why does the 6080 perform sarl instructions in 73% of the time of the 6180, even after compensation for the 6180 associative memory slowdown? (About 470 nanoseconds in the 6180 time are unexplained.)
2. Why does the 645 perform stpbp instructions so rapidly? (A store with address preparation inhibited should take about 480 nanoseconds longer than measured.)

If the 6180 were first brought up to the 6080 speed, and the 645 speed up trick were then applied, the resulting stpbp instruction time should be around 1450 nanoseconds, which would produce a 645/6180 ratio of about 2.0 rather than 1.35.

Effect on Multics

To estimate the overall effect on Multics and its users, one must have some idea of the relative frequency of occurrence of the different types of instructions. Two different experiments were performed:

1. Call-save-return speed. The time to perform a PL/I call-save-return sequence with no arguments on the two machines was compared. Currently, the two instruction sequences are essentially the same. (The only difference is that the register store instructions of the 645 sequence have been removed, and also the 6180 entry sequence has been slightly lengthened, changes which should almost balance out.) The times were 170 μsec on the 645, and 110 μsec on the 6180. The speedup is 170/110, or 1.54. Examination of the 51-instruction sequence established that 39 of the instructions are

of the type for which address preparation of the next instruction is inhibited, 12 are pointer store instructions, and 13 have indirect addresses. Thus, the standard call-save-return sequence is a heavy user of instructions which, relatively, have not been sped up very much.

2. "Nothing" loop. A library program named "nothing", which when called does nothing but return, was called as a user command. The CPU time used to read the command line, decode it, locate the command, call it, and then print a report of time used was measured. Since the 6180/IOM system uses different channel control software then the 645/GIOC system, the experiment was performed by having I/O directed at a file rather than the typewriter. The following times were measured:

645:	25ms	total "nothing" time
6180:	14ms	

a ratio of 1.8. In both cases, there were no page faults. Since this "nothing" loop includes one wall-crossing, wall-crossing was measured. Because of the ring hardware, it is much smaller on the 6180:

645:	3.3ms	wall-crossing time with three arguments.
6180:	.4ms	

To determine the speedup of the "nothing" loop apart from wall-crossings, we subtract the measured cost of a wall-crossing from each:

645:	25 - 3.3 = 21.7ms	"nothing" time without wall-crossings
6180:	14 - 0.4 = 13.6ms	

Thus the ratio of 6180/645 in the "nothing" loop with wall-crossings discounted is 1.6. The effect of wall-crossings on the running system lies somewhere between these extremes -- the average program is probably affected about 1/3 as much as the "nothing" loop. Except for its higher than average use of wall-crossings, the "nothing" loop is probably representative of most system and library code.

Effect of the 100ns associative memory delay on Multics

From the figures in Table I, we may estimate the removal of the 100ns associative memory delay will speed up the 6180 by about 10%. After a 10% adjustment, the "nothing" loop would run about 1.8 times as fast as on the 645.

Conclusions

The primary conclusion is that since Multics makes very frequent use of a set of instructions which have not been speeded up very much, the 6180 does not provide nearly as much performance improvement over the 645 as was anticipated. Instead of the factor of 2.5 quoted for GCOS on the 6080 when compared with the 635 (and which will apparently be achieved by the 6180 ada-sequence when the speed handicap is removed), we seem to be dealing with an average factor closer to 1.8. Predictions as to the maximum number of users must be scaled down 30% below their former values, and predictions of cost per unit operation must be scaled up about 40%.

Recommendations

There are at least three obvious directions worth studying:

1. Examine the 6180 design to discover why the store pointer instructions speed up by less than a factor of 1.7 compared with the 645. Since the memory speed was up by 1.7, memory speed cannot be the limiting factor on those instructions. Some attention should be focused on understanding why the 6080 has EIS register instructions which execute much more rapidly than the corresponding 6180 instructions. Finally, the 645 design should be reviewed to discover why its store pointer instruction is so fast, and whether or not the same technique could be adapted for the 6180.
2. Since address preparation overlap is defeated on so large a fraction of instructions used by Multics, schemes for speeding up the memory should provide more leverage than in the 6080/GCOS system. For example, a buffer memory installed in each CPU which reduced average memory access times to, say, 70nsec. would be very effective. In the call-save-return sequence, for example, there are 39 inhibitions of address preparation overlap and 13 indirect addresses each of which add two memory cycle times. Thus a savings of 590 nsec. would be realized 65 times, for a total of 38 μsec. in

a sequence currently requiring 110 μsec. The call-save-return sequence would thus run 1.5 times as rapidly as it does at present.

3. As soon as possible, remove the 10% speed handicap in the M.I.T. 6180, so as to determine that it is actually 10%.

Acknowledgements

Mike Schroeder, Rich Feiertag, and Dave Gifford did much of the work of measurement. Helpful discussions were provided by F.J. Corbató and Riley Doberstein.