Inter-Office Memorandum

To	David Liddle	Date	October 18, 1978
From	John Wick	Location	Palo Alto
Subject	**D0 Timing Simulation**	Organization	SDD/SS/DE

XEROX

Filed on: [Iris] < Thistle > Doc > ThistleReport.bravo

DRAFT

This report describes a series of experiments conducted to evaluate the performance of a number of proposed display configurations running on a D0, the OIS processor. The primary objective of this study was to verify the expected performance of the processor with a single large format display, and to discover the effects of adding a second display.

Because the eventual hardware, firmware, and software configurations are not presently available, a simulation approach was adopted. A program called Thistle was written to simulate the timing characteristics of the D0 processor at the micro instruction level. Instruction traces of a number of real programs (such as Apex and DeskTop) running on Alto/Mesa 4.1 were used to drive the simulation. A dozen experiments were run simulating the current hardware/firmware configuration to verify correct operation. Six program samples were then run with five different display configurations to predict their expected performance.

Simulator Input

Thistle requires two inputs to perform a simulation. The first is a trace of Mesa byte codes to be executed. The second is a description of the microcode which implements those instructions; provision for describing the display and memory refresh microcode is also included.

Instruction Traces

To obtain the instruction traces, a modified version of the Alto/Mesa 4.1 microcode was written which traps to the RAM at the beginning of each Mesa instruction. The RAM microcode records the opcode and its parameters in a trace buffer, which is written to the disk periodically; normal execution is then resumed. In a number of cases, additional information about the machine state is also captured. For example, all control transfers (XFERs and jumps) record the destination PC, so that buffer refill can be properly simulated. The alignment of operands was also recorded for some opcodes. The details of the trace format are described in [JohnssonITF].

There was little attempt to compensate for the differences between the current Alto/Mesa instruction set and the set proposed in the PrincOps [ThackerOIS]. The data contained here is therefore mildly pessimistic.

Instruction Profiles

Thistle also requires a description of the emulator and display microcode to be simulated. Because only timing charateristics of the processor are simulated, a rather terse description of the microcode is sufficient. It need only include processor and I/O memory references (and their alignment), memory interlocks and aborts, instruction buffer refill, and task switching. Microinstructions that are not otherwise interesting are grouped together into a count of execution cycles.

To arrive at this microcode description, we expanded on the idea of *instruction profiles* described in [Garner]. Instructions are divided into classes which exhibit the same memory and timing behavior. An instruction profile is then assigned to each class, as well as to the display and memory refresh tasks. Details of the instruction profile description can be found in [JohnssonTIP].

(Although we considered the possiblity of a program which compiled actual microcode source files into their profiles, it became clear that this would be much too big a project given the time constraints. Therefore all instruction profiles were produced by hand, and are subject to transcription errors.)

The important data dependicies were handled by including extra data in the instruction trace (for example, buffer refill depends heavily on alignment constraints; hence, the trace includes the PC value after each XFER and jump). Most other data dependencies result in very small differences in execution time (e.g., shifting right requires one more cycle than shifting left); these differences were ignored by the simulator. However, instructions like BLT and BITBLT required special casing. Their profiles were based on knowledge of the types of BITBLTs used in the test cases (as well as on an analysis of the microcode). For the display experiments, the profile for BLT assumed a four word block, and the profile for BITBLT assumed that a character was being painted.

Simulator Operation

The principle design objective of Thistle was to accurately simulate the interaction of the microprocessor and the memory. The instruction profiles for the Mesa emulator show the pattern of memory use that occurs while executing a given Mesa opcode. The main power of Thistle is that the interactions between adjacent opcodes and interactions between the emulator and other tasks (such as the display) can also be simulated, not for some abstract instruction mix, but for actual typical code sequences.

Automata

In addition to the microprocessor, the D0 contains two additional automata: the memory controllers MC1 and MC2. Thistle simulated the operation of MC1 and MC2 as described in [ThackerMT]; it also simulates the various kinds of aborts described in [ThackerD0]. Thus, if the profile calls for a PFETCH1 while MC1 is still active, the processor will undergo an MC1 abort for as many cycles as MC1 remains active. Likewise, referencing the data from a recent fetch will abort until MC2 finishes. Thistle keeps track of which task most recently used the memory, so the right thing happens if a task switch occurs between a fetch and use of the data.

Tasks

Most returns occurring within microinstructions will cause a task switch if another microtask of higher priority is ready to run. The display task is special in that it will also allow a lower priority task to run when it tasks. Thistle simulates this situation using coroutines.

Each task has a profile to execute. Every task except the emulator task has a "next wakeup" time associated with it. After every tasking return in a profile, control is passed to the coroutine executing the profile of the highest priority task willing to run (the emulator is always willing to run). When a task is finished for a while, it updates its wakeup time.

For the display experiments, the only tasks simulated were the emulator, the display, and memory refresh. Other tasks can be added to Thistle without much difficulty.

Simulator Output

While the primary use of Thistle in this study is for large batch runs, it also has an interactive mode for debugging purposes. (We expect Thistle to continue to be of use in fine tunning the microcode with very little overhead.) The current state of the processor and memory controller, as well as accumulated statistics on all of the tasks (emulator, display, and memory refresh) are displayed continuously if desired, and Thistle has various forms of "single-stepping" at the micro and macro instruction level. Complete information on the operation of Thistle and its output format can be found in the *Thistle User's Guide* [JohnssonTUG].

For the purposes of this report, Thistle accumulates the number of cycles spent in each of the three tasks (emulator, display, and memory refresh). Cycles are assigned to tasks based on the value of the processor's current task register. The time in each task is broken down into running and waiting; the waiting time is further broken down into MC1, MC2, suspend, and (for the emulator task) NEWINST aborts. Details of these states can be found in the *D0 Functional Specification* [ThackerD0].

Thistle also records the number of Mesa instructions executed as well as the total cycles expended (the sum of the run and wait times discussed above). These together with the processor clock speed (85*ns*) are used to calculate a Kip rate (kilo instructions per second).

Benchmarks

Our first step was to verify correct operation of the simulator. These were run under current conditions, and should be carefully distinguished from the experiments described in the next section. We chose eight benchmark tests to match against actual D0 elapsed time. We also made several probes of a running D0 with a digital voltmeter to verify the various wait times reported by Thistle.

Integer and String Sorting

Our primary benchmarks were the sort programs which have been in use for measuring Mesa performance since 1976; they were extended slightly to operate optionally with a full page display of random data (they perform no display related operations themselves). A total of eight tests were run: small and large integer and string sorts with the display on and off. A set of instruction profiles was derived from (the then current) microcode Version 1.5' (with the clock bug fixed -- PCR#20.53). All tests were run on EM016 after verifying its board revision levels. Note that these tests and their corresponding simulations were run with an IUTFP driving the 850 display and with old microcode which is known to have unacceptable diplay performance.

The results of these benchmarks are described in [Wick]. They show accuracy of execution time well within 10%, with the simulator running slightly faster than a real D0. Some possible explanations for this discrepancy can be found in the reference.

Wait Times

To verify proper modeling of the memory controller and its interaction with the processor, a set of four signals (MC1 active, MC2 active, suspend, and abort) were measured and compared with corresponding figures produced by Thistle. Four cases were compared using the benchmark programs: integer and string sort with the display on and off.

The results of this benchmark are presented and discussed in [JohnssonTBV]. While comparisons with the actual voltages are not very meaningful (because the signals cannot be measured accurately), both the real D0 and Thistle exhibited the same behavior with respect to these four signals as the display was turned on and off, and this behavior was consistent across all of the test cases.

Experiments

Several changes to the input were made before running the experimental data (the simulator itself was not changed after running the benchmark tests). New microcode was written for each display configuration; several hardware fixes were enabled, and key parts of the emulator microcode were rewritten. These modifications are described in more detail below.

Display Configurations

The hardware (UTVFC) is described in [Cameron]; [JarvisPDC] contains a functional specification for the device driver, including cursor, mouse, and keyboard support.

Three display devices were involved in the experiments, in a total of five different configurations. They are identified as follows:

LF One and two 17" Large Format displays
FP One and two 850 Full Page displays
QP Four Quarter Page displays

Detailed characteristics of these devices are described in [JarvisDC], which also contains a description of the microcode used to support each device and the assumptions made about it (particularly regarding scanline alignment).

Hardware

We assumed the presence of a number of fixes to the hardware which have not yet been installed (although most have been tested on Thacker's D0).

NEWINST aborts will be reduced from the end of MC1 (six to seventeen cycles) to completion of the mapping operation (four to six cycles) [Memory control board revision K].

A change to NEXTINST/NEXTDATA will result in tasking between Mesa instructions and eliminate the need for the "time to task" counter [Control board revision I].

A change in the Misc board will allow the test for pending interrupts to be moved from the buffer refill code to NOOP [Misc board revision G]

LONGJUMP will be added to allow changing the current page and performing a jump in the same instruction [Control board revision I].

These changes are described in the documentation on D0 board revision levels maintained by ED.

Firmware

The current D0 microcode (version 1.5) was rewritten (on paper) to take advantage of the hardware changes and to include a number of known but as yet unimplemented improvements suggested by Chuck Thacker. The rewrite concentrated on three areas: XFER, jumps, and buffer refill. Quadword code alignment and proper code byte ordering were assumed, as was a hardware stack error check, and numerous TASKs were added throughout the microcode. We incorporated as many changes as we could track from the 2.0 microcode, which is still under development.

Due to time constraints, we were not able to implement the PrincOps microcode. The simulations were run with the Alto/Mesa instruction set as it currently exists (version 4.1), with process bytecodes implemented in Nova code, and an Alto compatible BITBLT.

Experimental Data

Six sample instruction traces were taken from three Alto/Mesa application programs; all samples involved display manipulation. One sample of each program focused on the inner loop containing the code to paint characters on the display.

> DTest: a test program for the Alto/Mesa system display package. It writes characters on the display as if it were a Teletype, while also maintaining a typescript file.
>
> DeskTop: Advanced Design/User Prototype's experimental Star like environment. Two traces involving opening a document and painting the screen were taken.
>
> Apex: Product Software's applications executive. The three samples obtained involved moving a document into a folder, opening a document, and painting characters in a window.

The samples ranged from 0.48 to 2.86 seconds of simulated execution time; they varied from 121k to 468k Mesa instructions. More details on the samples can be found in [Sandman].

Results

The thirty test cases -- six instruction traces and five display configurations -- were run in about 56 hours of elapsed Alto time (about 36 seconds of simulated time). The raw data is summarized in Table 1; it shows the percentage of time running and waiting in the display and emulator tasks, followed by the sum of running and waiting for each task. (The memory refresh task accounts for a constand 2% of the cycles in all test cases.) The table also shows the instruction rate in Kips.

One display configuration was eliminated from the rest of the analysis. While running two Full Page displays, the simulator reported a large number (about 45%) of "misses", in which the display had missed a wakeup for a new scan line because it had not finished processing the previous one (this would show up as screen tearing). This explains why the Kip rates for the two FP case are only slightly smaller than with a single Full Page display.

Figures 1-4 summarize the run and run plus wait time (as a percentage of total cycles) for the display and emulator tasks. Figure 5 summarizes the Kip rates for all display configurations.

As we expected, one LF display consumes about 20% of the cycles, and two LF displays need just under 40%. One FP falls inbetween, at just under 30%, and four QP displays require a bit more

(just over 30%). The simulation indicates that two Full Page displays cannot be supported.

References

[Cameron] Cameron, J., Thacker, C., Tseng, C. *User Terminal Variable Format Controller (UTVFC) Specification.* Revision 5.0. September 28, 1978.

[Garner] Garner, B. *Mesa Opcode Timing.* June 21, 1978.

[JarvisPDC] Jarvis, J. P. *Functional Specification for the Prototype Display Controller.* October 13, 1978.

[JarvisDC] Jarvis, J. P. *Display Characteristics.* October 12, 1978.

[JohnssonITF] Johnsson, R., Sandman, J., Wick, J. *Instruction Trace Format.* October 11, 1978.

[JohnssonTUG] Johnsson, R., Sweet, R., Wick, J. *Thistle User's Guide.* October 11, 1978.

[JohnssonTIP] Johnsson, R., Sweet, R., Wick, J. *Thistle Instruction Profiles.* October 11, 1978.

[JohnssonTBV] Johnsson, R., Wick, J. *Thistle Benchmark: Voltages.* October 12, 1978.

[Sandman] Sandman, J. *Thistle Trace Data.* October 13, 1978.

[ThackerOIS] Thacker, C. *OIS Processor Principles of Operation.* Version 2.0. April 9, 1977.

[ThackerD0] Thacker, C. *D0 Processor Functional Specification.* January 16, 1978.

[ThackerMT] Thacker, C. *MemTiming.sil.* July 14, 1978.

[Wick] Wick, J. *Thistle Benchmark: Elapsed Time.* October 12, 1978.

Distribution:
Jarvis
Lampson
Lynch
Metcalfe
Thacker
Weaver
Mesa Group

	1LF		2LF		1FP		2FP		4QP	
ApexA	56.4	14.4	43.8	27.4	51.5	18.7	50.5	19.3	50.9	21.2
	22.5	4.5	16.5	10.1	18.8	8.8	18.2	9.8	16.8	8.9
	78.9	18.9	60.3	37.5	70.3	27.5	68.7	29.1	67.7	30.1
	271.2		210.5		247.7		243.0		244.9	
ApexB	54.7	14.4	42.4	27.4	49.9	18.7	48.9	19.4	49.3	21.2
	23.9	4.8	17.3	10.6	19.9	9.3	19.2	10.4	17.8	9.5
	78.6	19.2	59.7	38.0	69.8	28.0	68.1	29.8	67.1	30.7
	275.4		213.6		251.6		246.3		248.5	
ApexC	50.7	14.4	39.0	27.4	46.2	18.7	44.3	20.6	45.7	21.2
	27.3	5.3	19.3	11.9	22.4	10.5	20.9	11.9	20.1	10.8
	78.0	19.7	58.3	39.3	68.6	29.2	65.2	32.5	65.8	32.0
	324.6		249.7		295.4		283.7		292.4	
DeskTopA	58.7	14.4	45.7	27.4	53.5	18.7	52.3	19.5	52.8	21.2
	20.0	4.6	14.2	10.4	16.7	9.0	15.8	10.1	14.8	9.0
	78.7	19.0	59.9	37.8	70.2	27.7	68.1	29.6	67.6	30.2
	265.3		206.5		241.5		236.4		238.6	
DeskTopB	60.5	14.4	47.3	27.4	55.1	18.7	53.9	19.5	54.4	21.3
	18.4	4.4	13.0	9.9	15.4	8.6	14.6	9.7	13.7	8.5
	78.9	18.8	60.3	37.3	70.5	27.3	68.5	29.2	68.1	29.8
	253.3		198.1		231.0		225.9		227.9	
DTestA	52.2	14.4	40.7	27.4	47.8	18.7	46.9	19.4	47.3	21.2
	25.9	5.2	18.4	11.1	21.5	9.8	20.5	10.9	19.1	10.2
	78.1	19.6	59.1	38.5	69.3	28.5	67.4	30.3	66.4	31.4
	210.5		164.0		192.6		189.3		190.6	

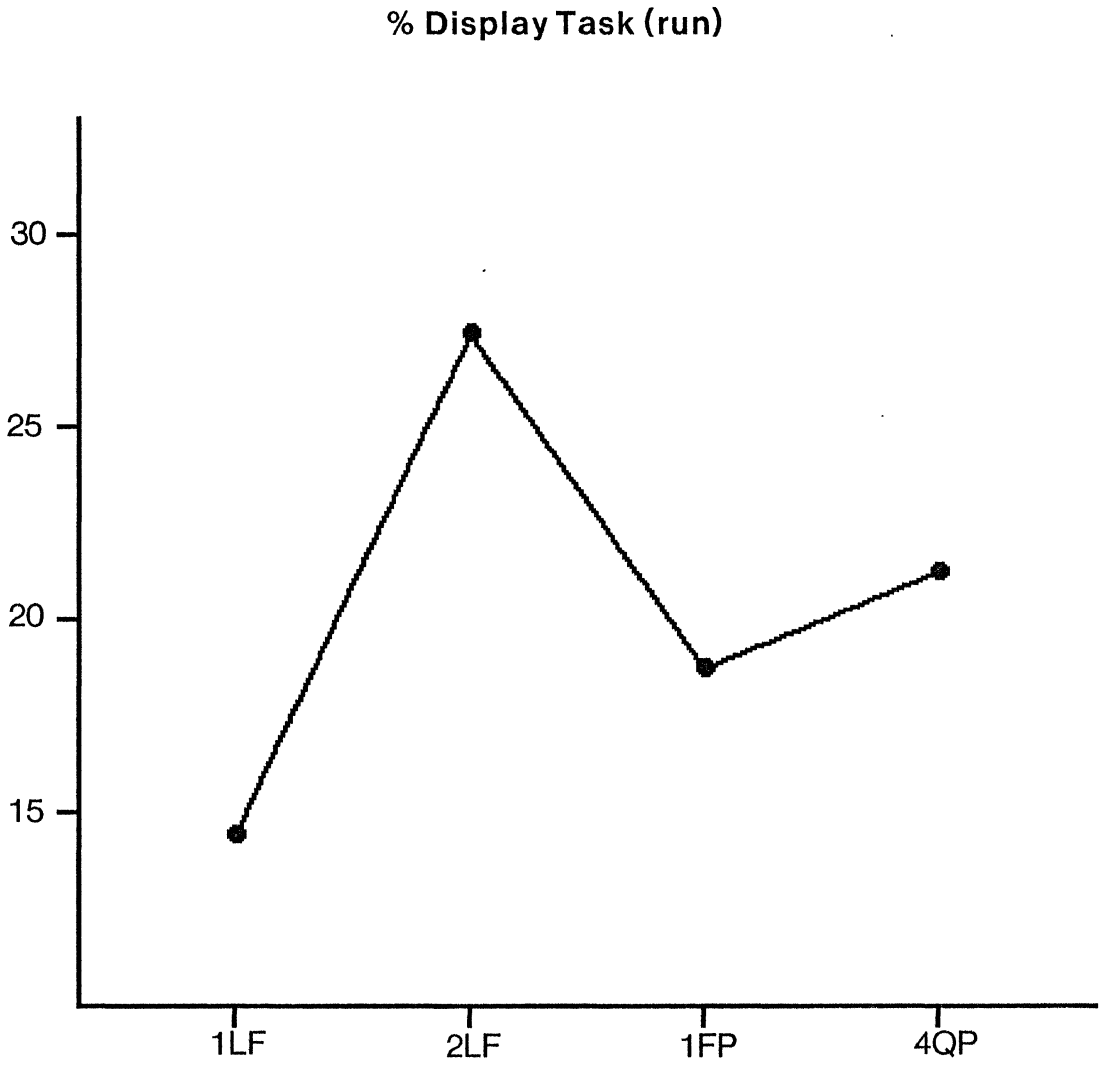
Display Configuration

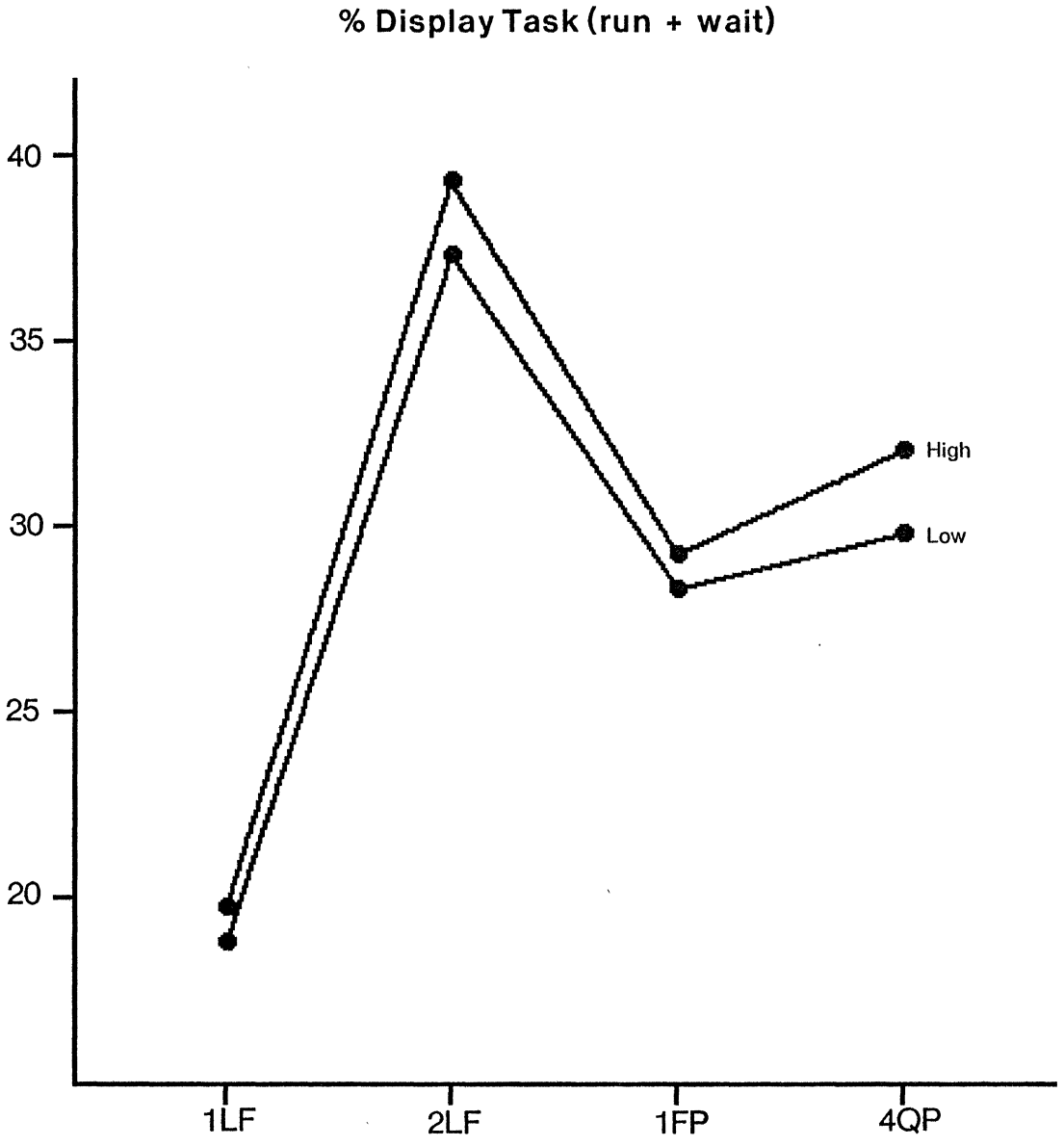
Emulator	Display
running	running
waiting	waiting
total	total

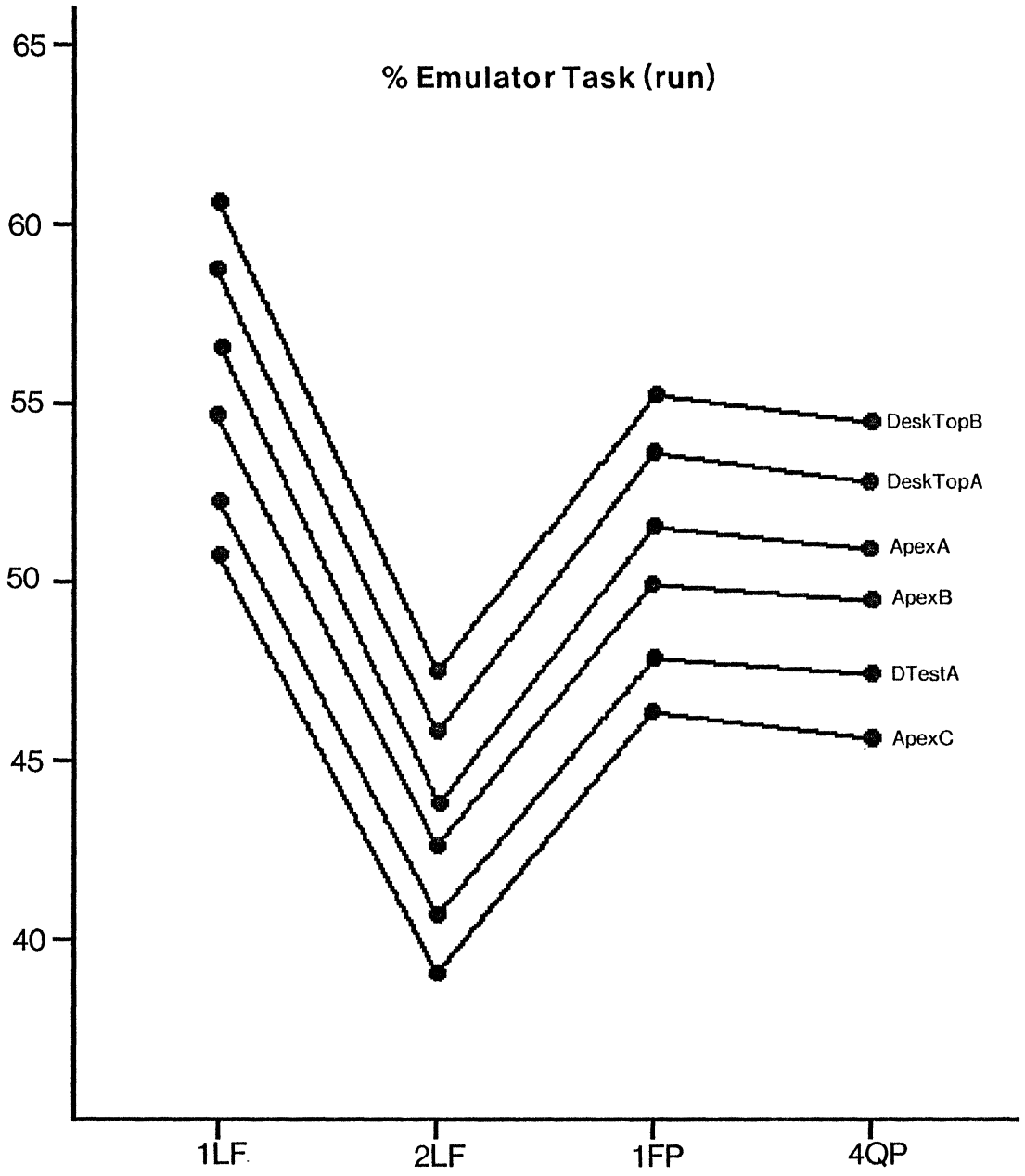
Kips

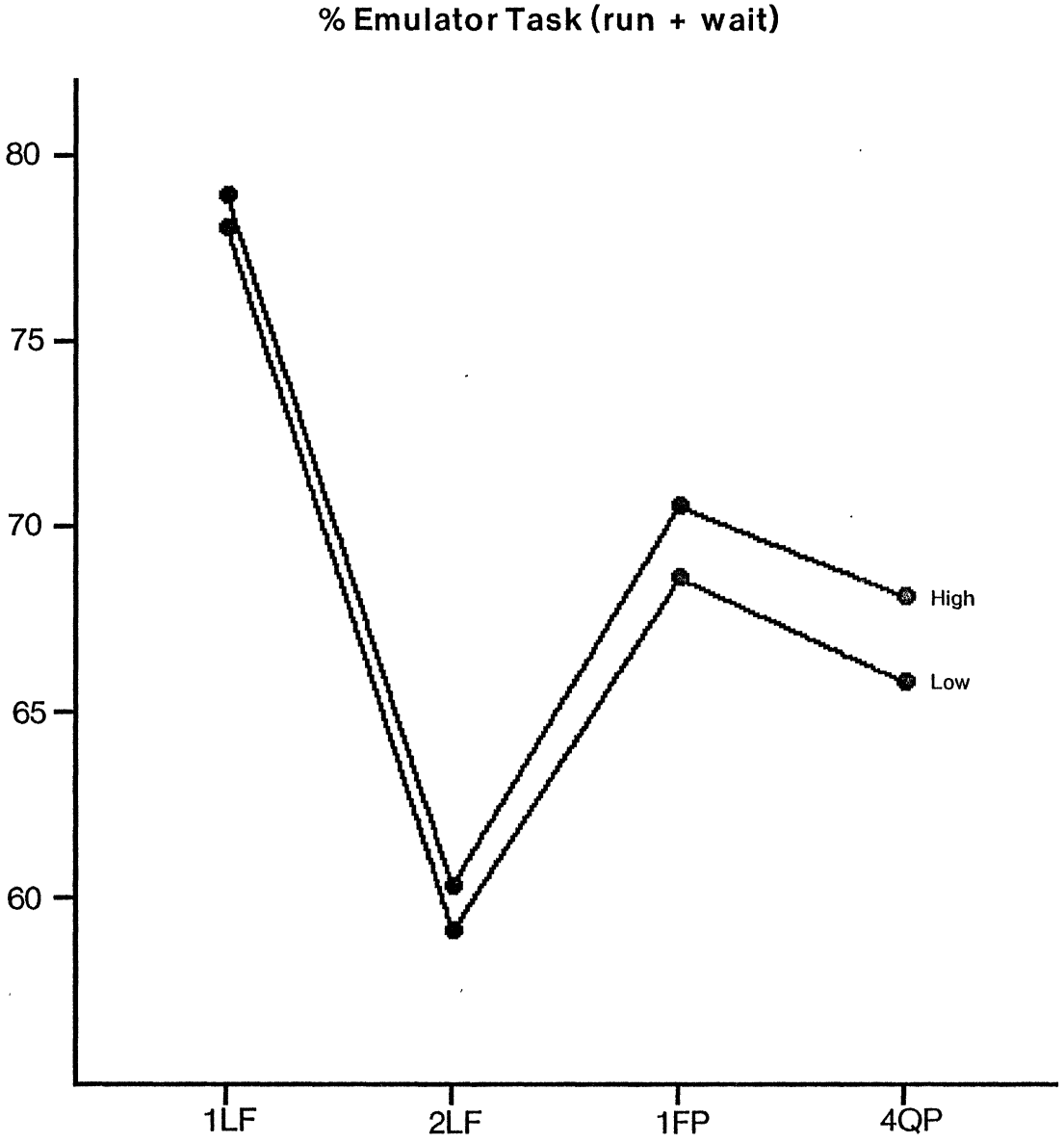
Displays

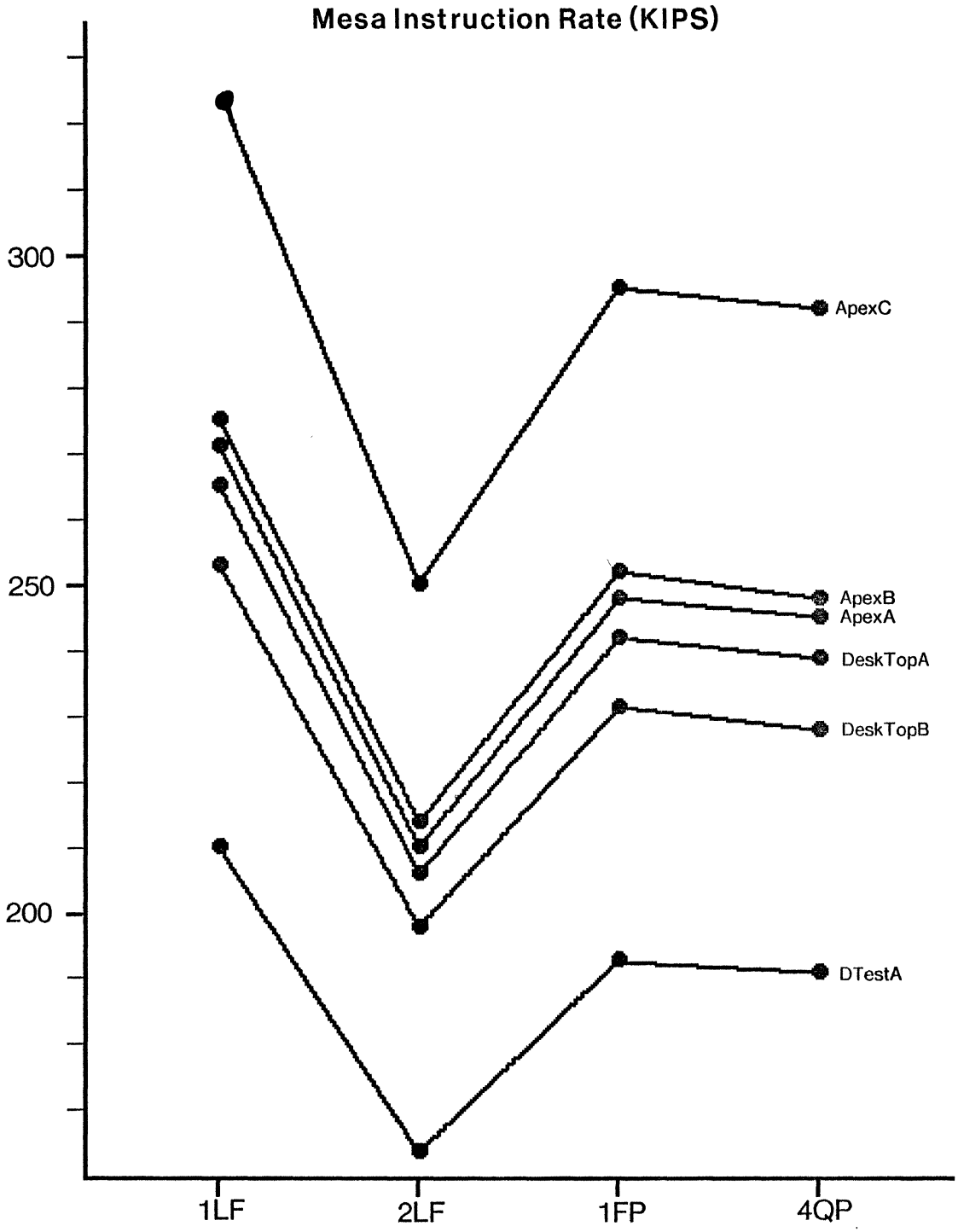
1LF: one large format (17") display
2LF: two large format (17") displays
1FP: one full page (850) display
2FP: two full page (850) displays
1QP: four quarter page displays


% Display Task (run)
30
25
20
15
1LF
2LF
1FP
4QP


% Display Task (run + wait)
40
35
30
25
20
High
Low
1LF
2LF
1FP
4QP


% Emulator Task (run)
65
60
55
50
45
40
1LF
2LF
1FP
4QP
DeskTopB
DeskTopA
ApexA
ApexB
DTestA
ApexC


% Emulator Task (run + wait)
80
75
70
65
60
High
Low
1LF
2LF
1FP
4QP


Mesa Instruction Rate (KIPS)
300
250
200
ApexC
ApexB
ApexA
DeskTopA
DeskTopB
DTestA
1LF
2LF
1FP
4QP