intel

October 1982

iAPX 186, 286 BENCHMARK REPORT

Source. Microcomputer Application Engineering

Order Number: 210826

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

BXP, CREDIT, i, ICE, I^2-ICE, iCS, iLBX,i_m, iMMX, Insite, INTEL, int_el, Intelevision, Intellec, $int_eligent$ Identifier™, $int_eligent$ Programming™, Intellink, iOSP, iPDS, iRMS, iSBC, iSBX, iSXM, Library Manager, MCS, Megachassis, Micromainframe, MULTIBUS, Multichannel™ Plug-A-Bubble, MULTIMODULE, PROMPT, Promware, RMX/80, RUPI, System 2000, and UPI, and the combination of ICE, iCS, iRMX, iSBC, MCS, or UPI and a numerical suffix.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

* MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Department
3065 Bowers Avenue
Santa Clara, CA 95051

iAPX 186/286 BENCHMARK REPORT

CONTENTS

iAPX 186 and iAPX 286 (Real Address Mode)

Benchmark Report

In the past, many people have made attempts to characterize the relative performance of 16-bit microprocessors by using a number of different programs called benchmarks. These programs show how quickly the processor is able to handle certain processing tasks. Each type of task uses different processor instructions with differing frequencies. Because some processor architectures may lend themselves with greater facility to certain classes of tasks or to the implementation of higher level languages, the relative performance between processors measured among different benchmarks varies greatly, just as it does between processors running different application programs.

Many previously published benchmarks show the relative performance of the 8086 with the 68000. These benchmarks were written both in assembly language and in higher level languages (e.g. Pascal or C). With the introduction of the iAPX 186 and iAPX 286, requests have been made to show how well these new processors perform the same benchmark tasks. All of these benchmark programs have been run in the Intel applications lab in Santa Clara on iAPX 186 and iAPX 286 processor evaluation systems. All iAPX 286 numbers quoted in this report were generated with the iAPX 286 running in real address mode. This is the mode that is 100% code compatible with the 8086 and iAPX 186. In this mode it does not perform memory management and protection. The evaluation systems allow the processors to run at full speed with no wait states, or allow wait states to be inserted to measure how the performance changes when wait states are inserted. The actual programs themselves were compiled or assembled, and real time measurements were made. The results of this work show that in both assembly language and in higher level languages, the iAPX 186 and iAPX 286 have superior performance compared to the 68000.

The benchmark programs used are:

These benchmarks compare the performance of the iAPX 186 and iAPX 286 for a large number of tasks with other 16-bit microprocessors.

SYSTEM COMPARISONS

Although the exact system configurations used to generate the published benchmark numbers for non-Intel processors is frequently not known, the speeds of system components used in 8 MHz systems for maximum performance (i.e. no wait states) can be compared for the iAPX 186, iAPX 286 and 68000.

A small iAPX 186 configuration is shown in figure 1. Notice that it requires address latching, and the system shown allows for data bus buffering. If no wait states are to be inserted into a memory fetch, the system will require a memory access time from address valid of:

$$\begin{aligned} \text{ACCESS TIME} &= (3 \times t_{CLCL}) - t_{CLAV} - t_{DVCL} - t_{IVOV} - t_{IVOV} \\ &= 375 - 44 - 20 - 30 - 30 \\ &= 251 \text{ ns} \end{aligned}$$

These numbers include the time for addresses going valid (t_{CLAV}), data input setup times (t_{DVCL}) and buffer and latch propagation delays (t_{IVOV}).

A small 68000 configuration is shown in figure 2. The system shown allows for address and data buffering. Since the 68000 does not have a multiplexed address/data bus, address latching is not required. For a memory system of any reasonable size, however, the addresses will require buffering. If no wait states are to be inserted into a memory fetch, the system will require a memory access time from address valid of:

$$\begin{aligned} \text{ACCESS TIME} &= (3 \times t_{CYC}) - t_{CLAV} - t_{DICL} - (2 \times t_{IVOV}) \\ &= 375 - 44 - 20 - 30 - 30 \\ &= 230 \text{ ns} \end{aligned}$$

These numbers include the address valid time (t_{CLAV}), the data input set up time (t_{DICL}) and the buffer delay time.

A small iAPX 286 system is shown in figure 3. This system allows for address and data buffering. ALthough the iAPX 286 does not have a multiplexed address/data bus, addresses must be latched to insure that valid addresses for one bus cycle are held to the memory components until the end of the bus cycle. If no wait states are to be inserted into a memory fetch, the system will require a memory acceses time from address valid of:

$$\begin{aligned} \text{ACCESS TIME} &= (4 \times \text{system clock period}) - (\text{read data set up time}) - \\ &\quad t_{SHOV} - t_{IVOV} - (\text{STB delay time}) \\ &= 250 - 10 - 45 - 30 - 10 \\ &= 155 \text{ ns} \end{aligned}$$

These numbers allow for the address strobe to be generated by TTL right at the beginning of a bus cycle on the iAPX 286, and include the strobe to output valid latch delay time (t_{SHOV}) and the buffer delay time (t_{IVOV}).

A larger iAPX 286 system can use memory interleaving to allow the use of slower memory components. If the memory system is divided into two banks, each with their own address latches, and is controlled by logic requiring 5TTL packages, the memory device access time is 200 ns. This time is derived by:

$$\begin{aligned} \text{ACCESS TIME} &= (5 \times \text{systems clock period}) - (\text{address out delay}) - 2 \times t_{IVOV} - \text{data setup} \\ &= (62.5 \times 5) - 40 - 60 - 10 \\ &= 202.5 \text{ ns} \end{aligned}$$

With interleaving, a wait state will be added during about 10% of the memory cycles.

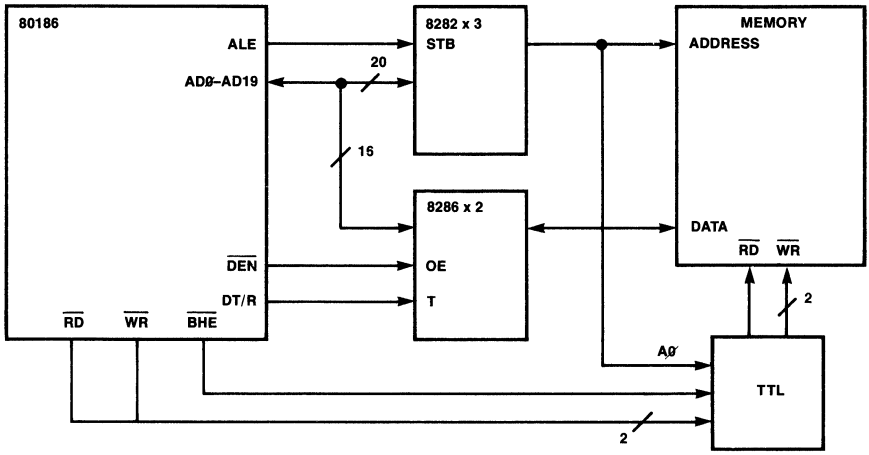

Figure 1A. Typical Simple 80186 System

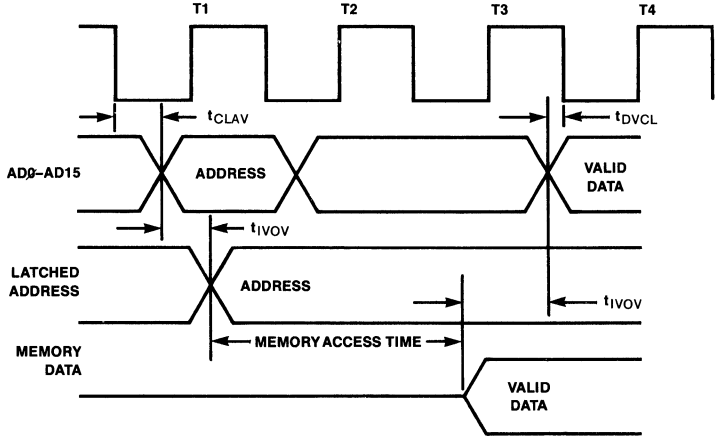

Figure 1B. 186 System Memory Timing Diagram

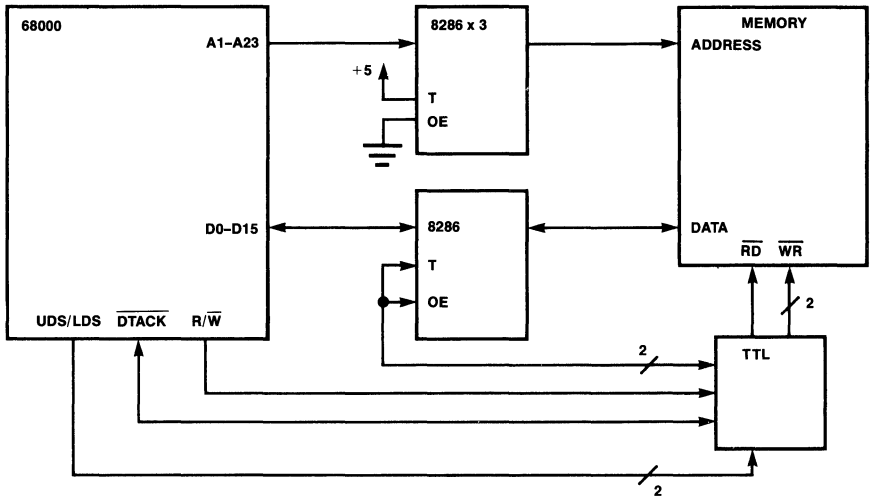

Figure 2A. Typical Simple 68000 System

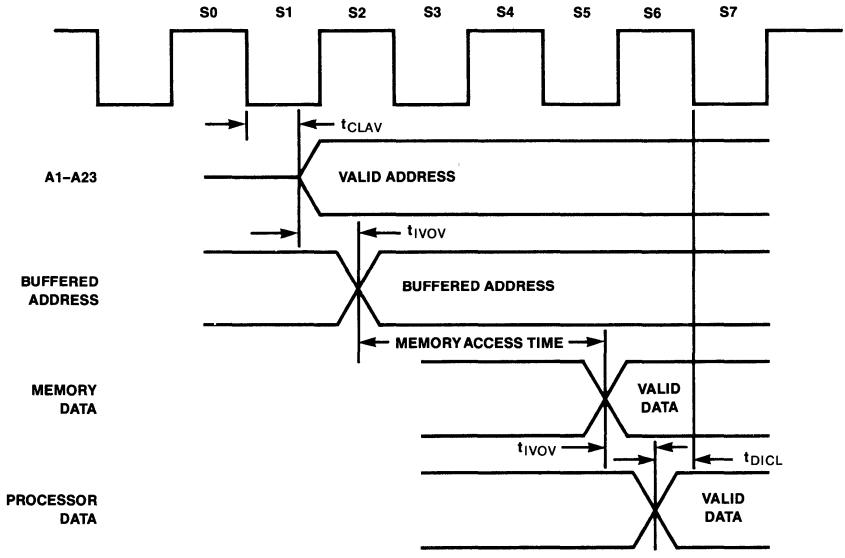

Figure 2B. 68000 System Memory Timing Diagram

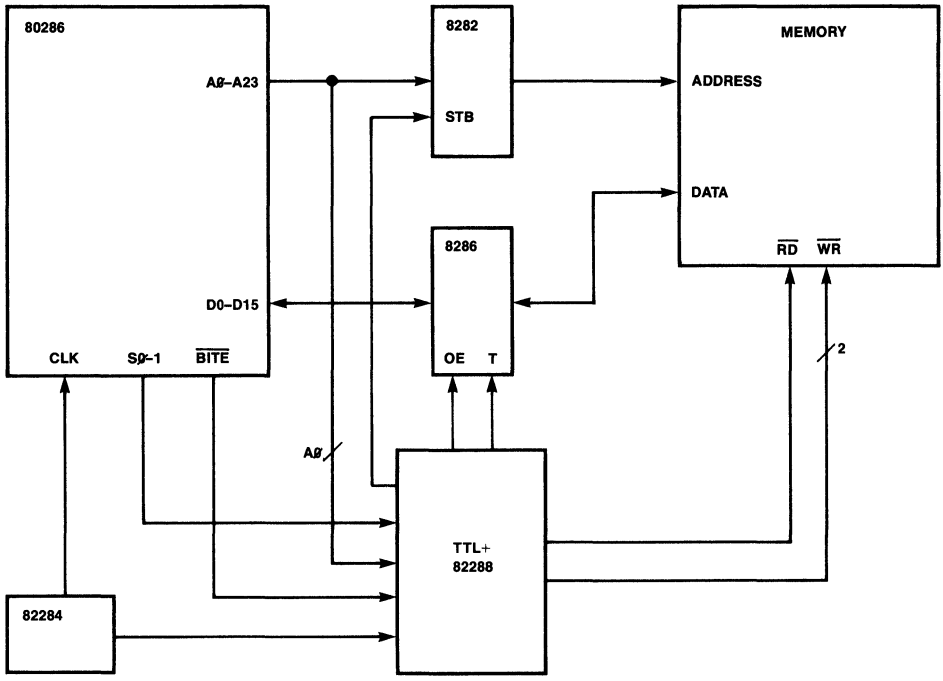

Figure 3A. Typical 286 System

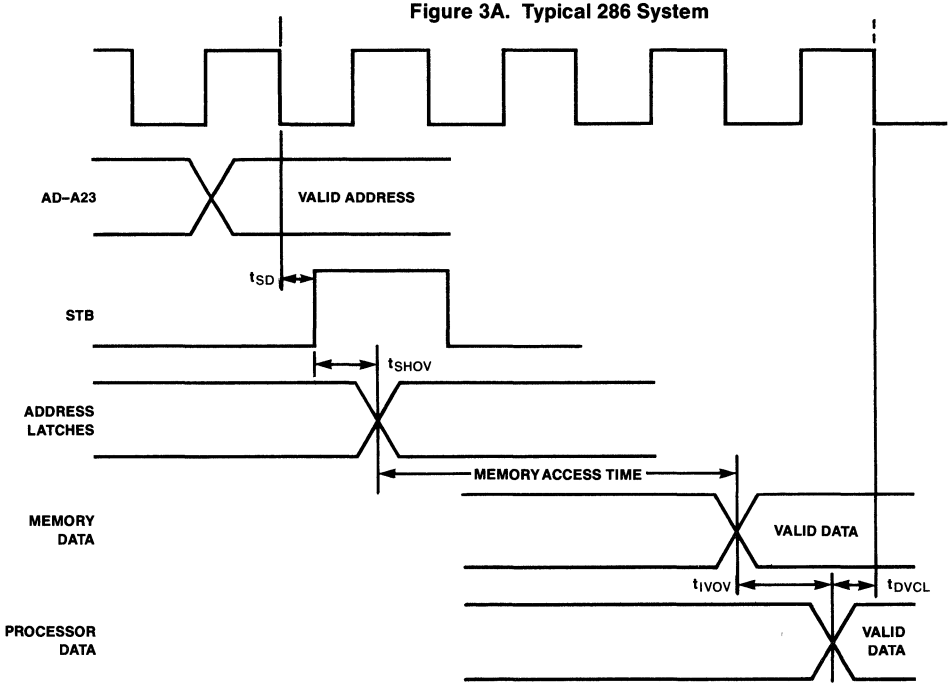

Figure 3B. 286 System Memory Timing Diagram

DIGITAL FILTER BENCHMARKS

In the February, 1981 edition of IEEE Micro Magazine (pp. 23-41), H. T. Nagle and V. P. Nelson at Auburn University presented the results of some work they had been doing using 16-bit microprocessors in digital filters. They discussed methods and algorithms for various types of digital filters, and presented benchmark data for a particular instance of an eighth-order cascaded filter. In the article, performance measurements were run on the 8086, the Motorola 68000, the Texas Instruments 9900, the Zilog Z8000, and the Fairchild 9445 (a bipolar machine), all running (presumably) with no wait states. All programs for this benchmark were written in assembly language.

These benchmark programs have also been run in the Intel applications lab on both an iAPX 186 evaluation board (with no wait states) and on an iAPX 286 evaluation board (with no wait states, a similar configuration to that shown in figure 1). The clock speeds for the various processors used are:

68000	8 MHz
Z8000	4 MHz
8086	5 MHz
9900	3.3 HMz
9445	15 MHz
186	8 MHz
286	8 MHz

The performance of the iAPX 186 and iAPX 286, along with the performance quoted of the processors in the article for the procedures called by the digital filter program (in microseconds) are shown in the table below.

The various subroutines shown below perform the following tasks:

input: performs writes to a memory-mapped I/O device and polls the device until it receives acknowledgement (the benchmark assumes that the processor receives positive acknowledgement immediatly).

outp_ld: multiplies samples by a constant, then adds another constant to the result of this multiplication

delay_ld: shifts a memory array from one location to another

pre_ld: performs multiple multiplies and adds as part of the filter implementation

tot sample time: the time it takes for all filter processing between samples

procedure	68000	Z8000	8086	9900	9945	186	286
input	10.25	15.5	14.8	24.2	2.8	9.5	7.9
outp_ld	65.5	127.25	174.4	211.5	40.8	57	34.6
delay_ld	32	30.25	32.8	135.8	33.2	13.5	9.9
pre_ld	194.5	380.25	582.4	559.4	108	149.5	80.4
tot sample time	327usec	584usec	857.6usec	1000usec	194.9usec	273usec	139.9

The most important number shown is the total sample time. It indicates the amount of time it takes the processor to process a single data sample. Because each data sample must be processed sequentially, this number sets an upper limit to the sampling speed, and therefore the upper frequency limit of the filter. The higher this limit, the greater the number of applications to which the filter may be applied. With the performance numbers generated, the upper frequency limit of this digital filter for the 68000 is 3050 kHz, while for the iAPX 186 it is 3660 kHz (20% better than the 68000) and for the iAPX 286 it is 7148 kHz (134% better than the 68000).

The reasons for the improvements of the iAPX 186 and iAPX 286 are:

1. the outp_ld and pre_ld programs use the multiply instruction extensively, which is much faster on the iAPX 186 and iAPX 286 than it is on the 8086 and 68K

2. the delay_ld routine merely copies one area of memory to another area of memory using a string move instruction. Since string moves go at the bus bandwidth on the iAPX 186, this routine was much faster with the iAPX 186. The reason the 68000 number for this routine is so large is that the 68000 does not have a string move instruction, therefore requiring a software loop to perform this operation.

SIEVE OF ERATOSTHENES

In the September, 1981 edition of BYTE, Jim Gilbreath presented the results of benchmarking the "Sieve of Eratosthenes" on various microcomputers in various languages. This algorithm computes prime numbers without performing divisions or multiplications by selectively eliminating non-prime numbers from an array of all numbers within the number range selected. In the article, the author quoted performance numbers for both 8 and 16 bit machines (from the lowly 6502 to the lofty VAX11/780).

These benchmark programs have been run in the Intel applications lab on the iAPX 186 evaluation board (with no wait states) and on the iAPX 286 evaluation board (with no wait states). All the other microprocessor numbers are quoted directly from the BYTE article. For the iAPX 186 and iAPX 286, the Mark Williams C complier was used. The benchmark was changed slightly to use register variables.

The performance of iAPX 186 and real address mode iAPX 286, along with the performance given for a few of the more significant processors (in seconds) are given in the table below.

processor	language	time
68000(8MHz)	assembly	1.12
68000	Moto Pascal	14.0
8086(8MHz)	assembly	1.90
8086(5MHz)	Intel Pascal	9.05
PDP11/70	C	1.52
PDP11/40	C	6.10
186(8 MHz)	assembly	1.065
186(8 MHz)	C	2.241
186(8 MHz)	Intel Pascal(V2.0)	3.405
286(8 MHz)	assembly	.517
286(8 MHz)	C	1.14
VAX11/780	C	1.4

These performance numbers show that with equivalent clock speeds, the iAPX 186 outperforms the 68000 in both assembly language programs and Pascal programs. It also shows that the 8086 (and hence the iAPX 186 and iAPX 286) Pascal performance is better than the 68000 Pascal performance. This benchmark is not computationally intensive, rather it is memory intensive. This is apparent when comparing the iAPX 186 and iAPX 286 numbers. It is also interesting to note that the performance of the iAPX 286 is actually better than the VAX11/780!

EDN BENCHMARKS

In the April, 1981 edition of EDN, Robert Grappel and Jack Hemenway present the results of running a subset of the Carnegie-Mellon benchmark programs on the 8086, the 68000, the Z8000 and the LSI11/23. Each of these seven programs was written entirely in assembly language. Each of these programs has since been enhanced (to use the same algorithms as used on the 68000) and re-run on the 8086, the iAPX 186, and the iAPX 286 using evaluation systems in the Intel applications lab with a variety of wait states. The seven programs are:

A. I/O interrupt kernel: shows interrupt response time for a four level interrupt system (i.e. there are four different interrupt sources with different priority levels).

B. I/O kernel with FIFO processing: queues interrupt processing requests and does a small amount of processing.

E. Character-string search: searches a string for a certain character pattern

F. Bit set, reset, test: checks the bit manipulation capabilities of the processor; sets, resets, and tests arbitrary bits in an area of memory

H. Linked-list insertion: inserts five items into a linked list

I. Quicksort: implements a quicksort algorithm sorting 100 16-byte long records

K. Bit-matrix transposition: exercises bit manipulation capabilities, transposes a 7x7 bit array

The number of clock cycles (with no wait states) determined by running the routines on all the processors required to execute the benchmarks are given in the table below.

Benchmark	8086	80186	80286	68000	Z8002
A	456	392	226	320	260
B	3816	3448	1838	3216	2250
E	2053	2024	1050	2255	1140
F	1401	1128	727	696	740
H	1936	1504	865	1210	1220
I	246,417	203,552	118,117	173,482	133,000
K	3945	2888	1879	3660	3380

None of these routines use multiplies or divides, they mainly manipulated bits or words in memory. The speedup of the iAPX 186 over the 8086 is caused mainly by the speedups of the effective address calculation, speedups of multiple bit rotations, and by the new instructions saving and restoring the general purpose registers. The iAPX 286 adds a faster memory interface in addition to these other enhancements.

Benchmark F was mainly bit manipulation where the bit checking, setting and clearing instructions of the 68000 could be directly used. Note, however, that the 68000 did much worse in comparison with benchmark K, where it had to do bit swapping. This shows that even though an architecture may contain a class of instructions which allow for higher performance in a small number of applications, these instructions must be of sufficient generality to cause an overall performance improvement.

BERKELEY ARCHITECTURE BENCHMARKS

In the June, 1982 edition of Computer Architecture News, Paul Hansen, et al. at Berkeley presented a series of four programs used to benchmark Intel's 432, the 8086, the 68000 and the VAX11/780 for programs written in Ada, C and Pascal. All the numbers shown herein are for programs written in Pascal. The four benchmarks used are:

string search:	searches 120 character string for a 15 character substring
sieve:	the Sieve of Eratosthenes (similar to the BYTE benchmark, mentioned earlier)
puzzle:	a "bin packing program that solves a simple puzzle"
acker:	a program which computes the Ackerman function. This function is heavily recursive, and is used to show procedure calling overhead.

The machines used for these benchmarks are:

A VAX11/780 running 4.1 BSD UNIX

An 8 MHz 68000 running in an EXORMACS (causing four wait states, two for memory management and 2 for slow memory) with Motorola Pascal.

A 16 MHz 68000 stand alone system with no wait states and Motorola Pascal.

A 5 MHz 8086 in the series III development system with 1-3 wait states.

The 8 MHz iAPX 186 evaluation board in the Intel applications lab with a variety of wait states.

The 8 MHz iAPX 286 design test bed, program running in real address mode.

the processing time for the various processors (in milliseconds) are:

machine	language	search	sieve	puzzle	acker
VAX11/780	Pascal	1.6	220	11900	7800
68000(8 MHz,4ws)	Pascal(V2.0)	5.3	810	32470	11480
68000(8 MHz,0ws)	Pascal(V2.0)	2.6	392	18360	5500
68000(16 MHz,0ws)	Pascal(V2.0)	1.3	196	9180	2750
8086(5 MHz,2ws)	Pascal(X125)	7.3	764	44000	11100
186(8 MHz,0ws)	Pascal(V2.0)	2.7	314	16012	4114
286(8 MHz,0ws)	Pascal(V2.0)	1.29	175	9157	2175

In all but one case, the speed values for an 8 MHz 68000 with no wait states (calculated by doubling the values quoted for the 16 MHz 68000) are worse than the iAPX 186 8 MHz 0 wait state value. This shows that Pascal performance of the 186 with Intel Pascal is better than even the newest Motorola Pascal on the 68000. Also, on 3 of the 4 benchmarks, the 4 ws performance of the 68000 is less than half the 0 ws performance quoted in the article, which makes the numbers suspect.

It is also interesting to note that the iAPX 286 once again shows superior performance the the VAX11/780, and the 8 MHz iAPX 286 shows higher performance than the 16 MHz 68000!

PASCAL BENCHMARKS

In the February, 1982 iAPX 86 System Benchmark Report (published by Intel), Mark Moore and John Crawford present the results of running various Pascal routines on both the 8086 (using Intel Pascal V2.0) and the 68000 (using Motorola Pascal V1.2).

These benchmark programs have been run in the Intel applications lab on the iAPX 186 evaluation board and on a iAPX 286 evaluation board using Intel Pascal V2.0. The hardware used to generate these results assumes an 8 MHz processor running with no wait states in each instance.

The routines implemented are:

GCD: This program computes the Greatest Common Denominator of two integers using a recursive function. Function overhead and the MOD oeprator are tested by this benchmark.

Integer Matrix Multiply: This program uses a simple row/column inner product method to compute the product of two matrices. The elements of the matrices are integers. The program tests control structures, array references and integer arithmetic. The timind data presented was taken using a 32x32 matrix.

Bubble Sort: This program performs a bubble sort on 1000 numbers. Bubble Sort extensively tests control structures, relational expressions and array references.

Queens: This program lists all possible combinations of non-attacking queens on an NxN chessboard. The timing data is based on a 9x9 chessboard. This program tests control structures, boolean expression evaluation and evaluates the code generated for certain commonly used statements (for example A := A + 1).

The performance of the iAPX 186 and real address mode 286 compared to the 8086 and 68000 are shown in the table below. The numbers reflect execution time in seconds.

routine	8086	68000	186	286
GCD	35.1	106.6	22.6	12.1
Integer Matrix Multiply	20.6	68.2	12.5	6.9
Bubble Sort	14.6	33.1	11.0	5.6
Queens	20.8	54.7	13.9	7.2

These numbers clearly show the Pascal implementation available on the iAPX 186, iAPX 286 and 8086 to be superior to the implementation on the 68000. For a description of each of these routines, please see the iAPX 86 System Benchmark Report, Feb. 1982 (Intel Literature Order No. 210352).

Other than the general performance advantage of iAPX 86, 186, 286 over the 68000 when executing high level languanes, specific reasons for the performance improvements of the iAPX 186, 286 over the 68000 are:

1 The GCD routine uses modulo division to determine the greatest common denominator of two numbers. Since it uses the processor's divide instruction (which is much quicker on the iAPX 186 than the 68000) the performance of the iAPX 186 is considerably better.

2. The MMULT routine uses integer multiplication extensively. Again, since this instruction is much quicker on the iAPX 186 than the 68000, the performance improvement of this routine is considerable.

INTEL STANDARD ASSEMBLY LANGUAGE BENCHMARKS

In the August, 1981 8086 16-bit Microprocessor Benchmark Report (Intel Literature Order No. 205931), Hal Kop developed a number of routines to be used to benchmark microprocessors. In a later revision, he quoted numbers for both the 10 MHz 8086 and the 8 MHz 68000. All of these routines are written in assembly language for the two processors.

These benchmark programs have been run in the Intel applications lab on the iAPX 186 evaluation board and on the iAPX 286 evaluation board. The performance of the iAPX 186 and the performance quoted in the benchmark report (with all numbers normalized to show 8 MHz performance of the processors with a variety of wait states) is shown in the table below.

The routines implemented are:

1. Automated Parts Inspection: The automated parts inspection program controls two 8-bit D/A converters (X and Y control) and reads a gray shade signal from a 12-bit A/D converter. Both D/A and A/D converters are interfaced to an image-dissector camera. For each of the 16,384 (128 x 128) points, the measured gray shade signal is compared with a known good gray shade signal (stored in memory) to determine if it is within tolerance. If it is, the inspection continues. Otherwise, a reject part signal is generated. This benchmark assumes that all 16,384 points are within tolerance.

2. Block Translation: The block translation software translates each character from an EBCDIC buffer to ASCII and stores the translated character in an ASCII buffer. The translation operation is terminated either when an EOT character is detected or when all characters in the EBCDIC buffer have been translated and stored. This benchmark assumes that the EBCDIC buffer contains 132 characters, none of which is an EOT.

3. Bubble Sort: The bubble sort program sorts a one-dimensional array containing 16-bit integer elements into numerically ascending order using the exchange (bubble) sort algorithm. This benchmark assumes that the array contains 10 integers which are initially arranged in descending order.

4. XY Graphics Transformation: The XY transformation software scales (expands or compresses) a selected graphics window containing 16-bit unsigned integer XY pairs. Each X data value is offset by X_0 and multiplied by a fractional scale factor while each Y data value is offset by Y_0 and multiplied by the same fractional scale factor. This benchmark assumes the selected window contains 16,384 XY pairs.

5. Reentrant Procedure: The reentrant procedure benchmark tests processor features which are useful to implement reentrant procedures. Three input parameters are passed by value to the procedure. Prior to the call, the first parameter is in one of the general registers while the second and third parameters are stored in memory locations. Upon entry, the procedure preserves the state of the processor. It assumes that the procedure uses all general registers. Next the procedure allocates storage for three local variables. The procedure then adds the three passed parameters and stores the result in a local variable. Upon exit, the state of the processor is restored.

routine	no. wait states	8086	68000	186	286
automated parts inspct(seconds)	0	.834	.696	.381	.217
	1	.883	.762	.432	.254
	2	.924	.827	.465	.289
	3	.979	.893	.504	.346
block translate(msec)	0	.93	.883	.870	.415
	1	1.04	1.083	.930	.466
	2	1.14	1.283	.997	.565
	3	1.27	1.483	1.130	.697
bubble sort(msec)	0	1.14	.979	.921	.494
	1	1.26	1.184	1.038	.632
	2	1.33	1.390	1.180	.778
	3	1.46	1.595	1.340	.945
X-Y transformation(seconds)	0	1.4	.975	.532	.285
	1	1.44	1.010	.582	.317
	2	1.48	1.044	.612	.346
	3	1.52	1.079	.651	.385
re-entrant procedure(usec)	0	39.0	56.50	32.4	17.0
	1	43.0	69.13	37.5	21.4
	2	49.0	81.75	43.4	27.5
	3	55.0	94.38	50.2	33.3

These numbers show that the iAPX 186 is greatly superior to the 68000 on computationally bound programs (those with many multiplies or divides), and is at least equivalent to the 68000 for many other classes of programs written in assembly language.

The reasons for the performance improvements of the iAPX 186 over the 68000 are:

1. Both the automated parts inspection and the X-Y transformation are very computationally bound and use the multiply instruction extensively. Because of the high-speed CPU, the iAPX186 and iAPX286 both show a significant performance advantage in these two benchmarks. Because the pre-fetch queue in the Intel processors allowing parallel instruction execution and fetching, their performance remains high even when wait states are added to memory accesses.

2. The block translate and the bubble sort routines perform mainly memory accesses, therefore the performance improvement of the iAPX 186 over the 68000 in these two routines are because of the improved effective address calculation. In the iAPX 286, the performance improvement is caused by the higher speed bus, allowing more bytes to be moved from one location to another in less time.

CONCLUSIONS

Each of these benchmarks (except the Intel standard benchmarks and Pascal system benchmarks) were generated by outside parties. In all but one instance, they show 186 Pascal performance to be much superior to the 68000 Pascal performance (even when the new Motorola Pascal is used). In addition, the 186 assembly language performance is shown to be superior or equivalent to the 68000 assembly language performance in most cases. Since the iAPX 286 is an even higher performance processor than the iAPX 186, it is not suprising to find that the iAPX 286 performance surpasses the 68000 by a considerable margin on pratically all benchmarks shown. Indeed, on the few occasions where benchmark comparisons were made, the iAPX 286 outperformed the VAX11/780 superminicomputer!

iAPX 286 PROTECTED SYSTEM BENCHMARK REPORT

This is a report on the relative performance of two protected microprocessor systems, the Intel iAPX 286 and Motorola MC68000 with MC68451. The iAPX 286 (80286 component number) is a microprocessor with memory management and protection integrated on chip. The MC68000 microprocessor requires the Motorola MC68451 memory management unit for protection and memory management.

For an equivalent system's level comparison, both processors ran the same programs with similar memory systems. To keep the programs the same, either the same PASCAL source or same assembly language algorithm was used. Both systems use the highest clock frequency available. System timing was determined by manufacturer data sheets.

The 80286 execution times were measured by Intel on an 80286 component system operating at 0-3 wait states. The 68000 system execution times were taken from the published benchmark reports.

Operating the MC68000L8 with the MC68451L8 MMU at 8 MHz requires two wait states to function with the same memory system as the 8 MHz 80286. The effect of two wait states on the 68000 system has been extrapolated from earlier studies and incorporated in the numbers given here.(1) Both 0-wait and 2-wait execution times are shown for the MC68000 system execution times.

Five different suites of programs were used which cover a wide range of application processing requirements from numerics, subroutine calls, data manipulation, and sorting. Both assembly language and PASCAL programs were used. Four of the benchmark suites were developed outside Intel. The results were published in EDN magazine, Computer Architecture News, Byte Magazine, and IEEE Micro Magazine. The fifth suite is an Intel standard benchmark. All the sources of the benchmark suites included execution times for the MC68000 microprocessor.

BENCHMARK DESCRIPTION

The benchmark programs used are:

MC68000 WITH MC68451 SYSTEM CONFIGURATION

The Motorola MC68451 MMU component provides memory protection and management to the MC68000 microprocessor. The fastest currently available speed selection of the MC68451 is 8 MHz. Since the CPU must operate at a frequency no faster than that allowed by the MMU, the 68000 microprocessor was operated at 8 MHz also. Both components are packaged in .9x3.1 inch 64-pin DIPs.

All timing, clock counts, and operation definitions for the 68000 system were taken from the Motorola MC68000L8 advanced data sheet #ADI-814R2 and Motorola MC68451L8 advanced data sheet #ADI-872, both published in 1981.

As shown in figure 3, the MC68451 MMU is connected in the address path between the CPU and the memory system. Address latches are required at the physical address outputs of the MMU since these pins also serve as a bidirectional data bus on memory cycles directed to the MMU after address translation. The MMU HAD signal controls use of the address latches.

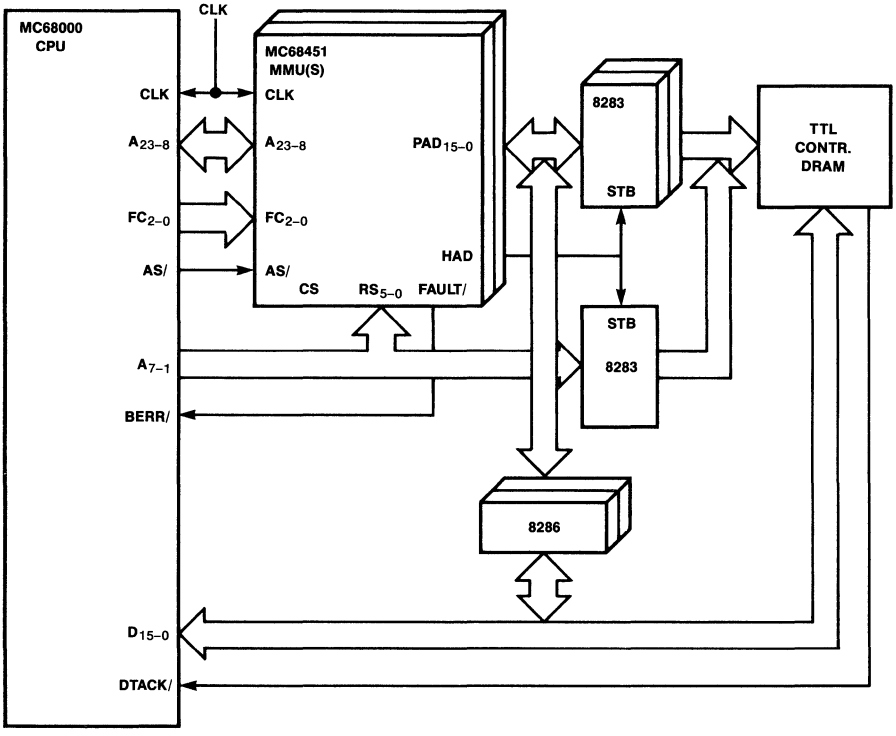

Figure 3. MC68000 with MC68451 Configuration

The MMU checks all addresses used by a program to see if they are valid. Valid addresses are then translated to a different address which is presented to the memory system. If an address is invalid, the MMU aborts the memory cycle and interrupts the program. Invalid addresses are normally assumed to be a result of programming errors.

The MC68451 MMU is a slave peripheral requiring the CPU to load all registers needed for its operation. The CS input tells the MMU when the CPU wants to send commands to it or inquire about its status. Data transfers directed towards the MMU require 8-20 wait states. Such delays are considered a part of the time required to program the MMU for a task switch.

Each MC68451 MMU supports up to 32 segments in the logical address space. Each segment may have a unique read/write and physical address attribute for its region of logical address space. The MMU allows segments to be of 16 sizes: all powers of two between 28 to 224 bytes. Segments start on addresses which are exact multiples of the segment size.

The MC68000 CPU must predefine all logical memory segments mapped by the MMU to a physical memory segment. If a logical address presented to the MMU by the CPU does not lie in a segment defined earlier, or if the segment does not allow the type of access requested, that address is considered invalid. Any such invalid access is reported to the CPU by the MMU FAULT/ signal connected to the CPU BERR/ input. The BERR/ input will terminate the current memory cycle and abort the associated instruction.

The address space for each task will normally require 5-7 segments in an MMU. A 68000 system with one MC68451 MMU component can execute about 5 tasks along with the operating system without requiring the MMU segment entries be changed during a task switch.

However, many multi-tasked systems have more than 5-7 tasks executing at one time. One MMU can not keep all the address space information for all the running tasks. Either the operating system must often transfer task address space information into and out of the MMU segment entries as tasks are run or more MMUs must be added to support the programs.

Most of the benchmarks for the 68000 system assume all memory management information is already in the MMU. This is a best case assumption for a system configuration using one MMU. With multiple tasks executing in a system with one MMU, it will be necessary to occasionally change the MMU contents when a new task is started. The benchmarks must reflect the time required to manage the MMU as well as execute the programs.

About 520 usec is required to load the MC68451 MMU with 5 segments and perform a MC68000 task switch at 8 MHz with two wait states. To account for this MMU reload time, two benchmarks which require task switchs include time to reload the MMU entries. The EDN interrupt benchmarks A and B were assigned an 8% MMU reload hit rate.

iAPX 286 CONFIGURATION

The iAPX 286/10 configuration assumed is shown in figure 4. The 80286 is packaged in a 68-pin JEDEC approved type A chip carrier requiring about 1 square inch of board area. No external MMU devices are required since the 80286 has the memory management hardware integrated into the component. The iAPX 286 information was obtained from the Intel iAPX 286/10 advanced data sheet (Order Number 210253-001) published in January 1982.

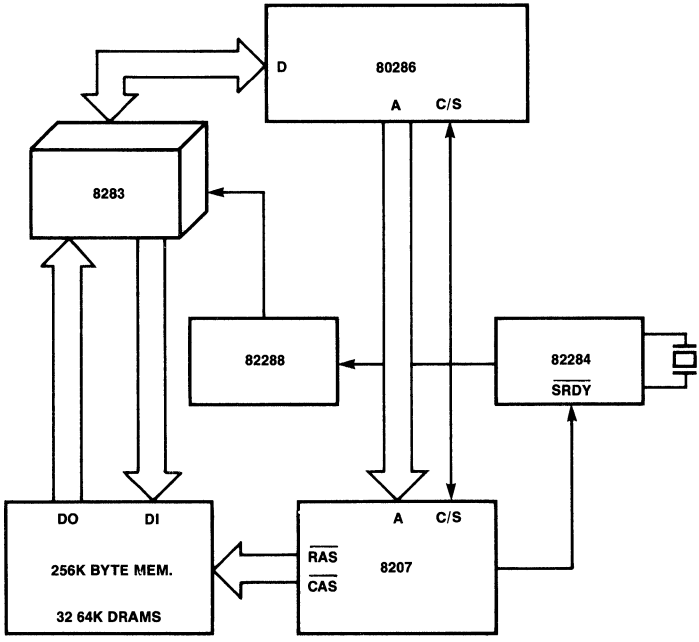

Figure 4. iAPX286 Configuration

The memory management and protection hardware of the iAPX 286 provides a superset of the functions of the MC68451 MMU. The 80286 integrated MMU and protection hardware checks and translates the logical addresses that programs deal with into physical addresses. The segmented structure of the logical address space is directly mapped into a segmented physical address space.

The 80286 CPU component allows up to 8191 segments to be accessed by any program without requiring extra components or operating system intervention while the program runs. All memory management information is automatically fetched from memory while the CPU executes a program. The 80286 execution times in all the programs described later include the time required for memory management information transfers.

MEMORY ACCESS TIME COMPARISONS

For most microprocessor systems, the memory costs regularly exceed the cost of the CPU. As a result it is vital to compare the relative performances of these processors when operating with comparable memory systems. The processor which uses its memory system most effectively will always have a significant price-performance advantage.

Figure 4 shows an 80286 system using the 8207 dynamic RAM controller and 64-2164A dynamic RAMs. This system can operate at 8 MHz with 0 wait states using 100ns DRAMs.

The 68000 system in figure 3 requires a TTL based dynamic RAM controller. Two wait states are required to operate with DRAMs to allow sufficient time to respond to a memory request. The translation delays of the MC68451L8 allow only 38 ns from translated address valid at the outputs of the MC68451 to return DTACK to the 68000 at 8 MHz and 1 wait state. This is not enough time for the address to be decoded and DRAM state logic to decide whether to begin the memory cycle or not. For most DRAM systems, two wait states will be required to allow sufficient ready response time.

The 80286 memory bus is pipelined. As a result, the 8207 memory controller has sufficient time to respond to memory requests with "not-ready" without requiring adding wait states to all memory cycles. The fast cycle time of the 80286 bus is automatically handled by the 8207 via interleaving. The 8207 controlled memory system is organized into 2 or 4 banks so that the RAS precharge time of one DRAM bank is overlapped with memory cycles occurring in another bank. The following table summarizes the performance of the two systems. The numbers shown below are measured from address valid at the CPU or MMU pins to either data or ready valid at the CPU pins.

Memory System Comparisons

	8 MHz 80286 0-wait	8 MHz MC68000L8 + MC68451L8 2-wait
Address access time	242 ns	293 ns
Ready response time	170 ns	163 ns
Maximum Bus Bandwidth	8 Mbyte/sec	2.66 Mbyte/sec

DIGITAL FILTER BENCHMARKS

In the February, 1981 edition of IEEE Micro Magazine (pp. 23-41), H. T. Nagle and V. P. Nelson at Auburn University presented the results of some work they had been doing using 16-bit microprocessors in digital filters. They discussed methods and algorithms for various types of digital filters, and presented benchmark data for a particular instance of an eighth-order cascaded filter. In the article, performance measurements were run on the 8086, the Motorola 68000, the Texas Instruments 9900, the Zilog Z8000, and the Fairchild 9445 (a bipolar machine), all running (presumably) with no wait states. This program was written in assembly language.

The performance of the 68000 with 68451 and 286 for the procedures called by the digital filter program (in microseconds) are shown in the table below.

The various subroutines shown below perform the following tasks:

input: Performs writes to a memory-mapped I/O device and polls the device until it receives acknowledgement (the benchmark assumes that the processor receives positive acknowledgement immediatly).

outp_ld: Multiplies samples by a constant, then adds another constant to the result of this multiplication

delay_ld: Shifts a memory array from one location to another

pre_ld: Performs multiple multiplies and adds as part of the filter implementation

total sample time: The time it takes for all filter processing between samples

Procedure	8 MHz 80286 0-wait	8 MHz (2) 68000+68451 0-wait	8 MHz (3) 68000+68451 2-wait
Filter	4.1 usec	24.7 usec	32.1 usec
Input	7.9 usec	10.3 usec	13.4 usec
Outp_ld	34.6 usec	65.5 usec	85.2 usec
Delay_ld	9.9 usec	32 usec	41.6 usec
Pre_ld	80.4 usec	194.5 usec	252.8 usec
Total Sample Time	139.9 usec	327.0 usec	425.1 usec
Normalized Performance	(1.0)	(.42)	(.32)

The most important number shown is the total sample time. It indicates the amount of time it takes the processor to process a single data sample. Because each data sample must be processed sequentially, this number sets an upper limit to the sampling speed, and therefore the upper frequency limit of the filter. The higher this limit, the greater the number of applications to which the filter may be applied. With the performance numbers generated, the upper frequency limit of this digital filter for the 68000 is 2352 Hz and for the 286 it is 7304 Hz (310% better than the 68000).

The reasons for the 286 performance advantage is:

1. The outp_ld and pre_ld programs use the multiply instruction extensively, which is much faster on the 286 than it is on the 68000.

2. The delay_ld routine merely copies one area of memory to another area of memory using a string move instruction. The string move instruction of the 80286 operates at the maximum bus bandwidth which is 3 times that of the 68000 system. The 68000 does not have a string move instruction, therefore requiring a software loop to execute this operation.

SIEVE OF ERATOSTHENES

In the September, 1981 edition of BYTE, Jim Gilbreath presented the results of benchmarking the "Sieve of Eratosthenes" on various microcomputers in various languages. This algorithm computes prime numbers without performing divisions or multiplications by selectively eliminating non-prime numbers from an array of all numbers within the number range selected.

This algorithm was coded up in assembly language and run on the 8 MHz 68000. An assembly language version of the benchmark was also run on the 80286. The execution times are shown below:

Processor:	8 MHz 80286 0 - Wait	8 MHz 68000 (2) 0 - Wait	8 MHz 68000+68451(3) 2 - Wait
Execution Time	.517 sec	1.12 sec	1.456 sec
Normalized Execution Time	(1.0)	(.46)	(.35)

The 286 higher performance in this benchmark reflects the general performance advantage which the 286 has over the 68000 in performing simple operations such as load, store, test, etc.

EDN BENCHMARKS

In the April and October, 1981 editions of EDN, Robert Grappel and Jack Hemenway present the results of running a subset of the Carnegie-Mellon benchmark programs on the 8086, the 68000, the Z8000 and the LSI-11/23. Each of these seven programs was written entirely in assembly language. The program for running each of these programs has since been enhanced for the protected mode iAPX 286 and to assure the same algorithm is used, and re-run on the 286. The seven programs are:

A. I/O interrupt kernel: shows interrupt response time for a four level interrupt system (i.e. there are four different interrupt sources with different priority levels).

B. I/O kernel with FIFO processing: queues interrupt processing requests and does a small amount of processing.

E. Character-string search: searches a string for a certain character pattern

F. Bit set, reset, test: checks the bit manipulation capabilities of the processor; sets, resets, and tests arbitrary bits in an area of memory

H. Linked-list insertion: inserts five items into a linked list

I. Quicksort: implements a quicksort algorithm sorting 100 16-byte long records

K. Bit-matrix transposition: exercises bit manipulation capabilities, transposes a 7x7 bit array

The two interrupt benchmarks, (A and B), require that the memory management hardware retain the address space context of both the program interrupted and that of the interrupt handler. Since the MC68451 can not normally retain all the memory management information required for all tasks, some task switches must change entries in the MMU.

To account for the required task switch time of the MC68451, both benchmarks A and B have a time added for reloading the MMU. The time required to change 5 entries in the MMU and perform a task switch is about 400 usec at 8 MHz and 0-wait or 520 usec at 8 MHz and 2-waits. An 8% MMU reload hit rate is assumed, or in other words, 92% of task switches don't require an MMU reload.

Since benchmark B requires that 12 interrupts be serviced, it is assumed one MMU reload (400 usec) will occur in it. This makes the 0-wait execution time of benchmark B 402 usec + 400 usec. Benchmark A is assigned one third of an MMU reload time (133 usec) since it has 4 interrupts. Benchmark A execution time becomes 40 usec + 133 usec.

EDN Benchmark Execution Times

	8 MHz 80286 0 - Wait	8 MHz 68000(2) 0 - Wait	8 MHz 68000+68451(3) 2 - Wait
A: I/O Int.	78.6 usec (1.0)	173 usec (.45)	223.8 usec (.35)
B: FIFO Int.	310 usec (1.0)	802 usec (.39)	1037.9 usec (.30)
E: String Search	131.3 usec (1.0)	281.9 usec (.47)	366 usec (.36)
F: Bit Manipulation	90.9 usec (1.0)	87 usec (1.04)	113 usec (.80)
H: Link List Insertion	108.1 usec (1.0)	151.2 usec (.71)	197 usec (.55)
I: Quicksort	20.5 ms (1.0)	21.68 ms (.95)	28.2 ms (.73)
K: Matrix Transpose	234.9 usec (1.0)	487.5 usec (.48)	634 usec (.37)
Normalized Execution Time	1.0	.64	.49

The performance advantage of the 286 was strongest in benchmarks A and B (interrupt intensive) because no software overhead is required to manage the 286's internal MMU. The 286's performance in the string search benchmark (E) was also helped by the 286 string scanning instructions.

The 68000 was strongest in benchmark F due to its bit manipulation instructions; however, when dealing with a more realistic bit addressing and manipulation program (benchmark K), the general performance superiority of the 286 shows through.

BERKELEY ARCHITECTURE BENCHMARKS

In the August 1982 edition of Computer Architecture News, Dave Patterson at Berkeley University present a series of four programs used to benchmark the iAPX 286, the 8086, the 68000 and the VAX 11/780 for programs written in Pascal. The programs used for the iAPX 286 were run in iAPX 86 real address mode. The following iAPX 286 execution numbers were measured in protected mode and use the enhanced instruction set of the iAPX 286. The four benchmarks used are:

string search:	Searches 120 character string for a 15 character substring
sieve:	The Sieve of Eratosthenes (similar to the BYTE benchmark, mentioned earlier)
puzzle:	A bin packing program that solves a simple puzzle.
acker:	A program which computes the Ackerman function. This function is heavily recursive, and is used to show procedure calling overhead.

The machines used for these benchmarks are:

A VAX 11/780 running 4.1 BSD UNIX

A 16 MHz 68000 stand alone system with no wait states and Motorola Pascal V2.0. The execution time for this system was doubled to account for slower maximum clock frequency of the 8 MHz MC68451.

The 8 MHz 80286 stand alone system with no wait states.

The processing time for the various processors (in milliseconds) are:

Machine	Language	search	sieve	puzzle	acker	Normalized Average
8 MHz 80286 0-wait	Pascal(V2.0)	1.29 (1.0)	175 (1.0)	9157 (1.0)	2175 (1.0)	(1.0)
8 MHz 68000(2) 0-wait	Pascal(V2.0)	2.6 (.50)	392 (.45)	18360 (.50)	5500 (.40)	(.46)
8 MHz 68000(3) 2-wait	Pascal(V2.0)	3.38 (.38)	509 (.34)	23868 (.38)	7150 (.30)	(.35)
VAX 11/780	Pascal	1.6 (.81)	220 (.80)	11900 (.77)	7800 (.28)	(.67)

This shows that Pascal performance of the 286 with Intel Pascal V2.0 is almost three times of Motorola Pascal V2.0 on the MC68000. It is also interesting to note that the 286 shows superior performance than the VAX 11/780 for these programs.

INTEL STANDARD ASSEMBLY LANGUAGE BENCHMARKS

In the August, 1981 8086 16-bit Microprocessor Benchmark Report, Intel developed a suite of programs to be used to benchmark microprocessors. In a later revision, numbers for both the 8 MHz 8086 and the 8 MHz 68000 were published. All of these routines are written in assembly language for the two processors.

The routines implemented are:

1. Automated Parts Inspection: The automated parts inspection program controls two 8-bit D/A converters (X and Y control) and reads a gray shade signal from a 12-bit A/D converter. Both D/A and A/D converters are interfaced to an image-dissector camera. For each of the 16,384 (128 x 128) points, the measured gray shade signal is compared with a known good gray shade signal (stored in memory) to determine if it is within tolerance. If it is, the inspection continues. Otherwise, a reject part signal is generated. One 16-bit multiply and one divide is performed for each point. This benchmark assumes that all 16,384 points are within tolerance.

2. Block Translation: The block translation software translates each character from an EBCDIC buffer to ASCII and stores the translated character in an ASCII buffer. The translation operation is terminated either when an EOT character is detected or when all characters in the EBCDIC buffer have been translated and stored. This benchmark assumes that the EBCDIC buffer contains 132 characters, none of which is an EOT.

3. Bubble Sort: The bubble sort program sorts a one-dimensional array containing 16-bit integer elements into numerically ascending order using the exchange (bubble) sort algorithm. This benchmark assumes that the array contains 10 integers which are initially arranged in descending order.

4. XY Transformation: The XY transformation software scales (expands or compresses) a selcted graphics window containing 16-bit unsigned integer XY pairs. Each X data value is offset by X_0 and multiplied by a fractional scale factor while each Y data value is offset by Y_0 and multiplied by the same fractional scale factor. One 16-bit multiply and divide is performed for each of the X and Y coordinates. This benchmark assumes the selected window contains 16,384 XY pairs.

5. Reentrant Procedure: The reentrant procedure benchmark demonstrates processor features which are useful to implement reentrant procedures. Three input parameters are passed by value to the procedure. Prior to the call, the first parameter is in one of the general registers while the second and third parameters are stored in memory locations. Upon entry, the procedure preserves the state of the processor. It assumes that the procedure uses all general registers. Next the procedure allocates storage for three local variables. The procedure then adds the three passed parameters and stores the result in a local variable. Upon exit, the state of the processor is restored.

The results when the same routines are implemented with the 286 and compared with the 68000 alone, and 68000 with 68451 are shown below:

Intel Assembly Language Benchmarks

	8 MHz 80286 0 - Wait	8 MHz 68000(2) 0 - Wait	8 MHz 68000+68451(2) 2 - Wait
Automated Parts Inspection	217 ms (1.0)	696 ms (.31)	827 ms (.26)
Block Translation	.415 ms (1.0)	.883 ms (.47)	1.283 ms (.32)
Bubble Sort	.494 ms (1.0)	.979 ms (.50)	1.39 ms (.36)
Computer Graphics XY Transform	285 ms (1.0)	975 ms (.29)	1044 ms (.27)
Reentrant Procedure	17 usec (1.0)	56.5 usec (.31)	81.75 usec (.21)
Normalized Average	(1.0)	(.38)	(.28)

The reasons for the performance improvements of the 286 over the 68000 are:

1. Both the automated parts inspection and the X-Y transformation are very computationally bound and use the multiply and divide instructions extensively. Because of the fast multiply and divide instruction execution time of 2.875-3.0 usec, the 286 shows a significant performance advantage in these two benchmarks. The pre-fetch queue in the Intel processor allows parallel instruction execution and fetching providing high performance even when wait states are added to memory accesses.

2. The block translate and the bubble sort routines perform mainly memory accesses. In the 286, the performance improvement is caused by the higher speed bus, allowing more bytes to be moved from one location to another in less time.

3. The Reentrant procedure benchmark uses the PUSHA and POPA instructions of the 80286 to quickly save the registers while the 68000 must save more and larger registers.

The two wait state execution times of the 68000 benchmarks were measured by Intel. Across the group, adding two wait states to the 68000 increased its 0-wait execution time by 30% (ranging from 1.1 to 1.5 with an average of 1.36). A 1.3X factor was used in the remaining benchmarks to derive a 2-wait execution time from the 0-wait execution time.

CONCLUSIONS

Based on an average of the results presented in the previous sections, the following table summarizes the relative performance of the iAPX 286 and MC68000-MC68451.

Relative Processor Performance

	Byte Sieve Program	Berkeley Pascal Programs	Intel Assembly Programs	EDN Assembly Programs	IEEE Micro Digital Filter Program
80286	1.0	1.0	1.0	1.0	1.0
68000+68451(3)	.35	.35	.28	.49	.32

Each of these benchmarks, except the Intel standard benchmark, were generated by outside parties. In all cases the iAPX 286 outperforms the MC68000 with MC68451. As a whole the iAPX 286 system was 2.8 times faster than the MC68000 system with comparable memory systems. Even without the MC68451 MMU, the 8 MHz 80286 at 0-waits outperforms an 8 MHz MC68000 at 0-waits by more than two to one.

In MC68000 systems there is a cost vs. performance tradeoff in deciding on the number of MMU components to use. Adding more MMU components will reduce the number of MMU reloads which occur but at the expense of high system cost and more board space required. The deciding factor will be the number of tasks run and the required level of system performance.

With a fixed number of MMU entries, an MC68000 system will have more overhead with more tasks active since the MMU must be reloaded more often. More overhead for more work is one requirement for thrashing to occur.

In contrast, the 80286 performance will not be affected by how many tasks are currently active. No extra overhead arises to switch tasks if more tasks are currrently running.

For more information on these benchmarks or on the iAPX 286 contact your local Intel sales office.

(1) Two wait state performance of the 68000 + 68451 is estimated at 1.3X 68000 0 wait state performance. This multiplier is estimated from the Intel Assembly Language benchmarks that showed 1.36X 0 wait execution for a 2 wait state 68000 system. This multiplier was confirmed by the Berkeley Pascal Benchmark article which measured a 1.9X multiplier for 4 wait state 68000 execution compared to 0 wait state 68000 performance. These two measures indicate that the 1.3X multiplier is realistic for two wait state 68000 performance.

(2) Performance numbers as reported in indicated publication.

(3) Based on estimated 2 wait state performance (reference (1)).

U.S. SALES OFFICES

October 1982

ALABAMA

Intel Corp
303 Williams Avenue, S W
Suite 1422
Huntsville 35801
Tel (205) 533-9353

ARIZONA

Intel Corp
10210 N 25th Avenue,
Suite 11
Phoenix 85021
Tel (602) 869-4980

CALIFORNIA

Intel Corp
1010 Hurley Way
Suite 300
Sacramento 95825
Tel (916) 929-4078

Intel Corp
7670 Opportunity Road
Suite 135
San Diego 92111
Tel (714) 268-3563

Intel Corp *
2000 East 4th Street
Suite 100
Santa Ana 92705
Tel (714) 835-9642
TWX 910-595-1114

Intel Corp *
3375 Scott Boulevard
Santa Clara 95051
Tel (408) 987-8086
TWX 910-339-9279
910-338-0255

Intel Corp *
5530 Corbin Avenue
Suite 120
Tarzana 91356
Tel (213) 708-0333
TWX 910-495-2045

COLORADO

Intel Corp
4445 Northpark Drive
Suite 100
Colorado Springs 80907
Tel (303) 594-6622

Intel Corp *
650 S Cherry Street
Suite 720
Denver 80222
Tel (303) 321-8086
TWX 910-931-2289

CONNECTICUT

Intel Corp
36 Padanaram Road
Danbury 06810
Tel (203) 792-8366
TWX 710-456-1199

EMC Corp
48 Purnell Place
Manchester 06040
Tel (203) 646-8085

EMC Corp
393 Center Street
Wallingford 06492
Tel (203) 265-6991

FLORIDA

Intel Corp
1500 N W 62nd Street
Suite 104
Ft Lauderdale 33309
Tel (305) 771-0600
TWX 510-956-9407

Intel Corp
500 N Maitland
Suite 205
Maitland 32751
Tel (305) 628-2393
TWX 810-853-9219

GEORGIA

Intel Corp
3300 Holcomb Bridge Road
Suite 225
Norcross 30092
Tel (404) 449-0541

ILLINOIS

Intel Corp *
2550 Golf Road
Suite 815
Rolling Meadows 60008
Tel (312) 981-7200
TWX 910-651-5881

INDIANA

Intel Corp
9100 Purdue Road
Suite 400
Indianapolis 46268
Tel (317) 875-0623

IOWA

Intel Corp
St Andrews Building
1930 St Andrews Drive N E
Cedar Rapids 52402
Tel (319) 393-5510

KANSAS

Intel Corp
8400 W 110th Street
Suite 170
Overland Park 66210
Tel (913) 642-8080

LOUISIANA

Industrial Digital Systems Corp
2332 Severn Avenue
Suite 202
Metairie, LA 70001
Tel (504) 831-8492

MARYLAND

Intel Corp *
7257 Parkway Drive
Hanover 21076
Tel (301) 796-7500
TWX 710-862-1944

Intel Corp
1620 Elton Road
Silver Spring 20903
Tel (301) 431-1200

MASSACHUSETTS

Intel Corp *
27 Industrial Avenue
Chelmsford 01824
Tel (617) 256-1800
TWX 710-343-6333

EMC Corp
381 Elliot Street
Newton 02164
Tel (617) 244-4740
TWX 922531

MICHIGAN

Intel Corp *
26500 Northwestern Hwy
Suite 401
Southfield 48075
Tel (313) 353-0920
TWX 810-244-4915

MINNESOTA

Intel Corp
3500 W 80th Street
Suite 360
Bloomington 55431
Tel (612) 835-6722
TWX 910-576-2867

MISSOURI

Intel Corp
4203 Earth City Expressway
Suite 131
Earth City 63045
Tel (314) 291-1990

NEW JERSEY

Intel Corp *
Raritan Plaza III
Raritan Center
Edison 08837
Tel (201) 225-3000
TWX 710-480-6238

NEW MEXICO

BFA Corp
P O Box 1237
Las Cruces 88001
Tel (505) 523-0601
TWX 910-983-0543

BFA Corp
3705 Westerfield, N E
Albuquerque 87111
Tel (505) 292-1212
TWX 910-989-1157

NEW YORK

Intel Corp *
300 Motor Parkway
Hauppauge 11787
Tel (516) 231-3300
TWX 510-227-6236

Intel Corp
80 Washington Street
Poughkeepsie 12601
Tel (914) 473-2303
TWX 510-248-0060

Intel Corp *
211 White Spruce Boulevard
Rochester 14623
Tel (716) 424-1050
TWX 510-253-7391

T-Squared
4054 Newcourt Avenue
Syracuse 13206
Tel (315) 463-8592
TWX 710-541-0554

T-Squared
7353 Pittsford
Victor Road
Victor 14564
Tel (716) 924-9101
TWX 510-254-8542

NORTH CAROLINA

Intel Corp
2306 W Meadowview Road
Suite 206
Greensboro 27407
Tel (919) 294-1541

OHIO

Intel Corp *
6500 Poe Avenue
Dayton 45414
Tel (513) 890-5350
TWX 810-450-2528

Intel Corp *
Chagrin-Brainard Bldg , No 300
28001 Chagrin Boulevard
Cleveland 44122
Tel (216) 464-2736
TWX 810-427-9298

OKLAHOMA

Intel Corp
4157 S Harvard Avenue
Suite 123
Tulsa 74101
Tel (918) 749-8688

OREGON

Intel Corp
10700 S W Beaverton
Hillsdale Highway
Suite 324
Beaverton 97005
Tel (503) 641-8086
TWX 910-467-8741

PENNSYLVANIA

Intel Corp *
510 Pennsylvania Avenue
Fort Washington 19034
Tel (215) 641-1000
TWX 510-661-2077

Intel Corp *
201 Penn Center Boulevard
Suite 301W
Pittsburgh 15235
Tel (412) 823-4970

Q E D Electronics
300 N York Road
Hatboro 19040
Tel (215) 674-9600

TEXAS

Intel Corp *
12300 Ford Road
Suite 380
Dallas 75234
Tel (214) 241-8087
TWX 910-860-5617

Intel Corp *
7322 S W Freeway
Suite 1490
Houston 77074
Tel (713) 988-8086
TWX 910-881-2490

Industrial Digital Systems Corp
5925 Sovereign
Suite 101
Houston 77036
Tel (713) 988-9421

Intel Corp
313 E Anderson Lane
Suite 314
Austin 78752
Tel (512) 454-3628

UTAH

Intel Corp
268 West 400 South
Salt Lake City 84101
Tel (801) 533-8086

VIRGINIA

Intel Corp
1501 Santa Rosa Road
Suite C-7
Richmond 23288
Tel (804) 282-5668

WASHINGTON

Intel Corp
110 110th Avenue N E
Suite 510
Bellevue 98005
Tel (206) 453-8086
TWX 910-443-3002

WISCONSIN

Intel Corp
150 S Sunnyslope Road
Brookfield 53005
Tel (414) 784-9060

*Field Application Location

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 (408) 987-8080

Printed in U.S.A./1182/5K/ME/CBM/BL