Processing Power on the IBM Personal Computer

John K. Gotwals
Computer Technology Department
South Campus Courts, Bldg. C
Purdue University
West Lafayette, IN 47907

I. Introduction

In the last 20 years there has been nearly a six-order-of-magnitude increase in computing power, and it is predicted that another five-to-six-orders-of-magnitude increase will occur by the year 2000 [1]. Although supercomputers such as the Cray-1 and Cyber 205 can perform certain vector operations with burst rates of up to 100 million floating-point operations per second (Mflops) [2], it is remarkable that a popular and affordable personal computer is only three-orders-of-magnitude slower. Indeed, with suitable software, the IBM Personal Computer (PC) represents a significant advance in usable memory size and floating-point computing power when compared with previous popular microcomputers.

It is the purpose of this paper to discuss several floating-point compute-intensive benchmark programs which have been run on the IBM PC. These programs utilize the Intel 8087 Numeric Data Processor (NDP), and when coded in FORTH, can approach the NDP maximum compute rate of 0.05 Mflops.

II. A New Generation Personal Computer?

While it has been stated that the IBM PC "is not a technological breakthrough but rather a nice combination of the best features of the current generation of single user microcomputers" [3], I argue here that in terms of memory address range and numeric processing capability, the IBM PC represents a new generation of personal computers. Of course, it should be noted that increased address space and floating-point coprocessing capability are useful only to the extent that software takes advantage of these enhanced features.

A. One Megabyte of Memory Space

The only safe observation that can be made about memory address range, is that a desired application program exists which requires more memory space than is available. In 1970, when the first PDP-11 minicomputer was designed, the PDP-11/20 designers noted that a weakness of prior minicomputer designs was limited memory addressability. Nevertheless, within two years, the PDP-11 had to have its 16-bit processor-generated addresses expanded into 18-bit Unibus addresses [4].

8-bit personal computers such as the immensely successful Tandy Radio Shack TRS-80 and Apple Computer Corporation Apple have a 16-bit memory address space, i. e., the central processing unit (CPU) can not conveniently address more than 2^{16} (65536 = 64k, where k = 1024) bytes. While this may seem like a large amount of memory, memory space is needed by the operating system (IBM DOS 2.0 uses 24k) and possibly by "memory mapped" input/output devices. In addition, the TRS-80 has one-fourth (the first 16k) of the memory space allocated permanently to the read only memory

(ROM) required by the BASIC interpreter and to I/O services. For many applications, the 64k address limit is a serious constraint. The inadaquacies of the 8-bit microcomputers for numeric-intensive applications will be discussed in a later section.

The IBM Personal Computer uses a third generation microprocessor, the INTEL 8088. The 8088 can be classified as a 16-bit processor, because it can internally operate on 16-bit data items. Since the 8088 has a 20-bit memory address range, the IBM PC can address 1,048,576 (1024k) bytes, which is a factor of 16 times greater than the previous generation of personal computers. This megabyte of memory space is divided into logical segments of up to 64k bytes each, and the CPU has direct access to four segments at a time. The segment registers are accessible to programs and can be modified by several instructions [5,6].

An address space of one megabyte allows, for example, the development of "integrated management" systems packages such as Context MBA [7] and VisiON [8]. These types of applications were not developed for earlier personal computers because the memory required was not available. We thus see a whole new class of applications emerging for this new generation of small computers.

B. Numeric Computation Power

In the same manner that the extended memory address space of the IBM Personal Computer permitted a new generation of large management system packages to be developed, it is my assertion that the IBM PC's numeric capabilities will encourage the development of new classes of applications which have significant computational requirements. It is only fair to point out that much of the software which is currently available has been translated from 8-bit microcomputers and consequently shows little, if any, speed improvement when run on the IBM PC.

In terms of raw computational capability, the IBM PC has the ability to outperform by a wide margin the previous generation of personal computers. A computer graphics benchmark which scales 16,384 pairs of 16 bit integers by a fractional scale factor runs ten times faster on a 5 MHz 8088 than on a 6 MHz Z80B [9]. (A Z80 microprocessor is used in the TRS-80 8 bit personal computer.) Similarily, a 16-bit multiply benchmark which reads two 16-bit integers from memory, multiplies them and returns the 32-bit product to memory, runs six times faster on the 8088 [9].

Floating-point performance comparisons are even more impressive. If an 8087 Numeric Data Processor is plugged into the coprocessor socket on the system board of an IBM PC, floating-point performance can be increased up to 100-fold. As will be shown below, this allows a new class of serious computation-intensive applications to be performed on the new generation of personal computers.

III. The 8087 Numeric Data Processor

A. The Famous Empty Socket

From the day of the IBM PC's announcement, there has been intense interest in the empty 40 pin socket adjacent to the 8088 processor on the system board. The first technical reference manual for the PC made reference to "AUX PROCESSOR SOCKET" on the system board data flow diagram, and the system board diagram referred to the same item as "SOCKET" [10]. As Ray Duncan so aptly describes the situation,

> "Within a matter of hours, it became an open secret in microcomputer circles that the mystery socket was properly wired to accept an 8087 numeric processor. There was an immediate upsurge of interest in the 8087 among scientists, engineers, speed freaks, and of course hungry software developers" [11].

B. NDP Overview and Architecture

The Intel 8087 is a coprocessor extension to the 8088 which adds hardware support for floating-point and extended precision integer data types. The NDP's arithmetic operations and numeric data formats conform to the IEEE Floating Point Standard [12], and the 8087 is designed to deliver stable and standard results for programmers who are not numerical analysis specialists [13,14]. Although the NDP allows results to be calculated with 24, 53 or 64 bits of precision, this paper will always assume that the 64 bit precision mode is being used. Specifying less precision does not improve execution speed, but the option is required by the IEEE standard.

Internally, the 8087 employs a register stack of eight 80-bit registers. Regardless of an external operand's data type, the operand is converted to temporary real before being stored in the register stack. A temporary real is organized in a three-field binary format consisting of a 1-bit sign field, a 15-bit biased exponent and a 64-bit significand. Even though the NDP usually operates with the significand in normalized form (leading bit contains 1), the leading significand bit of a temporary real number is physically present.

Externally, the NDP supports seven data types; 16, 32 and 64-bit signed integers, 80-bit signed packed decimal integer (18 significant digits), 32 and 64-bit real and 80-bit temporary real. For "scientific" applications, the 8087's temporary range of + or - 3.4×10^{-4932} to + or - $1.2 \times 10^{+4932}$ should be adequate for most calculations! For "commerical" applications, the coprocessor can be used to process decimal numbers of up to 18 digits without round-off errors. Exact arithmetic can be performed on integers as large as 2^{64}.

In addition to the four floating-point operations of addition, subtraction, division and multiplication, the NDP instruction set includes square root, partial tangent, partial arctangent, 2^x-1, y*(log base 2 of x) and y*[log base 2 of (x+1)].

C. 8087 Exception Handling

During execution, the 8087 will detect six different types of exception conditions:

- Invalid Operation, e.g., stack overflow/underflow, or an operand is in the class of values called NAN (not-a-number), or the result of an operation is indeterminate. This type of exception is usually caused by a program error.
- Overflow - The exponent of the result is too large for the destination real format.
- Underflow - The exponent of the result is too small for the destination real format.
- Zerodivide
- Denormalized Operand - The 8087 attempted to operate on an operand which was not normalized.
- Precision - The result of an operation can not be exactly represented in the destination format. This exception is provided for exact arithmetic applications.

By setting or clearing the exception masks in the NDP control word, the application can field any, all or none of the six exception classes. The 8087's exception responses have been chosen to deliver "reasonable" results, although an invalid operation should normally be detected by the application so that the program error can be corrected.

Since the exception flags in the NDP status word are "sticky", a program can clear the flags, run a calculation, and then test the flags to see if an exception was encountered. If perfomance requirements do not permit exception checking overhead, then an exception must cause the 8087 coprocessor to "interrupt" the host processor and force a "trap" to a user exception handler.

In the case of the IBM Personal Computer, switch 2 of system switch block SW1 must be set to the off position before an application can be interrupted by the NDP. If this switch is set off, then the 8087 interrupt will be "ORed" with the memory parity error interrupt, and either type of interrupt will force the 8088 host processor to execute a type 2 (non-maskable) interrupt. The application program must save the original type 2 interrupt vector, replace it with a pointer to the new interrupt handler, run the application and restore the original interrupt vector when the application is finished. At least one vendor offers

FORTH 8087 based software with an interrupt driven exception handler [15].

D. NDP Performance

The "number crunching" performance of the 8087 is impressive. Intel has published comparative execution times for the NDP and an 8086 executing the equivalent operations in highly optimized assembly language routines [13]. Both processors were driven by a 5 MHz clock.

	Microseconds		Fractional
Operation	8087	8086	Speedup
Multiply (DP)	27	2,100	78
Divide (SP)	39	3,200	82
Add	17	1,600	94
Compare	9	1,300	140
Tangent	90	13,000	140
Exponentiation	100	17,100	170
Square Root	36	19,600	540
Log	190	NA	

Before using this information to predict calculation times on the IBM PC, the following items should be noted:

- The IBM PC clock frequency is 4.77 MHz; the 8087 times must be increased by a factor of 1.05
- The IBM PC host processor is an 8088; the 8086 times must be increased by some unknown amount to compensate for the 8088 8-bit external data bus.
- For most practical situations, the IBM PC will not be able to keep the 8087 in a 100% duty cycle computational mode; this will be shown in the benchmark results.

It is interesting to compare the IBM PC coprocessor approach to floating-point processing with the FP-11 microcode approach used in the Digital Equipment Professional 300 series of personal computers [16]. The FP-11 floating-point "processor" includes six 64-bit floating-point accumulators; square root and transcendental instructions are not included in the instruction set. A double precision multiply takes 193 microseconds, and a single precision add takes 37 microseconds [17]. These values are for the case of a 300 ns microcycle time.

IV. Compute-Intensive Benchmark Results

A. The Owen Benchmark

Owen has written a floating-point benchmark program which he has used to compare transcedental function calculation rates of various systems [18]. I have used his program, FPBENCH, to generate data for BASIC-PLUS and BASIC-PLUS-2 on a PDP-11/70 (with hardware floating-point option) running the RSTS/E operating system. I have also rewritten FPBENCH in FORTH, using an 8087 software package furnished by a local vendor [15]. Below is a selection of data from the Owen article along with some of the data I have accumulated. All times are for 500 iterations and are given in units of seconds.

		--- Precision --		
Hardware	Software	Single	Double	Notes
TRS-80 Model 3	BASIC (Interpreted)	240		[19]
Apple II	APPLESOFT	180		
IBM PC	UCSD-Pascal	152		
IBM PC	BASIC (Interpreted)	90	311	[20]
IBM PC	FORTRAN	86		
IBM PC	Pascal	70		
IBM PC (8087)	FORTRAN	21		[21]
IBM PC	BASIC (COMPILED)	20	50	
IBM PC (8087)	BASIC (COMPILED)	10	13	[21]
IBM PC (8087)	Pascal	6		[21]
PDP-11/70	BASIC-PLUS	3.3	4.6	
IBM PC (8087)	FORTH	2.8	2.9	
PDP-11/70	BASIC-PLUS-2	1.0	1.4	
VAX 11/780	VMS - Pascal	0.44		

The excellent showing by FORTH [22] is due to the fact that the NDP stack can be directly manipulated by the programmer.

B. Matrix Multiplication Benchmark

The IBM PC, with the 8087 option, can multiply two 128 by 128 matrices and store the result in a third 128 by 128 matrix in 126 seconds. All matrix elements were single precision (4 byte) floating-point, and the benchmark was written in FORTH using the classical algorithm for multiplying matrices. Because 65,536 bytes of memory are used to store each matrix, memory limitations may prevent the duplication of this benchmark on other computers. In order to scale execution times, it is important to note that the number of floating-point operations performed by the classical multiplication algorithm for square matrices is proportional to the cube of the number of rows/columns.

When using 90 by 90 double precision (8 byte) floating-point matrices, and the same hardware/software environment as above, the matrix multiplication occurred in 57 seconds. Note that 64,800 bytes of memory are used to store each matrix.

Using the data given above, it follows that the IBM PC can multiply floating-point matrices at the rate of 33 and 26 kflops (thousands of floating-point operations per second) for single and double precision modes respectively. This is two-orders-of-magnitude faster than the previous generation of personal computers!

As mentioned earlier, all NDP internal operations are done with 80 bit temporary reals regardless of the precision of the external data type. Apparently, the 8 bit data bus of the IBM PC does not allow the 8087 to load and store data at the same rate the NDP can process data. This is the reason the compute rate for matrix multiplication drops 21% when processing double precision reals.

C. Fourier Series Evaluation

A program was written in FORTH, for the IBM PC with the 8087 option, which displayed on the screen the sum of the first 5000 terms of a Fourier sine series. The jth term was

$$(2/pi) * \{[(-1)^{(j + 1)}]/j\} * [\sin(j * pi * x)]$$

To calculate and display (in tabular form) partial sums with 10, 100, 1000 and 5000 terms for 11 evenly spaced values of x between 0.0 and 1.0, took 35 seconds. This same calculation takes as long as 24 minutes on the previous generation of personal computers.

D. Mean and Standard Deviation

The mean and standard deviation of 16,384 points of data stored in an array containing single precision floating-point elements can be calculated in 1.4 seconds. Since three floating-point calculations are needed for each datum, the computation rate is calculated to be 35 kflops. For 8,192 points of double precision data, the calculation occurs in 0.9 seconds with a computation rate of 27 kflops. These results were obtained from an 8087 equipped IBM PC, programmed with FORTH. Earlier 8-bit personal computers can not store such large arrays in memory, and their floating-point computation rate for this type of problem is less by as much as two-orders-of-magnitude.

E. Polynomial Evaluation

A polynomial approximation for the Bessel function $J_0(x)$ is [23]

$$J_0(x) = 1 - 2.24999\ 97(x/3)^2 + 1.26562\ 08(x/3)^4 - .31638\ 66(x/3)^6 + .04444\ 79(x/3)^8 - .00394\ 44(x/3)^{10} + .00021\ 00(x/3)^{12}$$

FORTH was used to write a program which takes data from a single precision floating-point array with 16,384 elements and stores the corresponding bessel function values in another array of the same size and type. When executed on an IBM PC equipped with the 8087 coprocessor, the program takes 7.0 seconds to run. Since each polynomial evaluation requires 14 floating-point calculations, the NDP computation rate is 33 kflops. The 8087 stack is used to store the polynomial coefficients, thereby minimizing memory accesses. Previous personal computers would not only require up to a 100-fold increase in time for the calculation, but their memory would not be large enough to contain the two arrays.

V. Summary

Compared with the previous generation of personal computers, the IBM personal computer has a much larger address space and significantly greater computational abilities. When equipped with the 8087 numeric data processor, a whole new class of compute-intensive applications can be executed on the IBM PC; the floating-point performance is up to two-orders-of-magnitude greater than earlier personal computers. Some representative computation rates are shown in the following table.

	----- Precision -----	
Description	Single	Double
Matrix multiplication	32 kflops	26 kflops
Polynomial evaluation	33 kflops	
Mean and standard deviation	35 kflops	27 kflops

References

[1] Morton E. Jones, William C. Holton and Robert Stratton, "Semiconductors: The Key to Computational Plenty", Proceedings of the IEEE, Vol. 70, No. 12, 1380-1409 (1982).

[2] Edward W. Kozdrowicki, "Supercomputers for the Eighties", Digital Design, Vol. 13, No. 5, 94-103 (1983).

[3] Arlin J. Brannstrom, "First Impressions of the IBM Personal Computer", NCCI Staff Paper Series, No. 4 (1982). North Central Computer Institute, 667 WARF Building, 610 Walnut St., Madison WI 53706.

[4] C. Gordon Bell and J. Craig Mudge, "The Evolution of the PDP-11", Computer Engineering, Digital Press, Bedford Massachusetts, 1978, pp. 381-382.

[5] Stephen P. Morse, The 8086 Primer, Hayden, Rochelle Park, New Jersey, 1980, pp. 11-16.

[6] iAPX 86,88 User's Manual, Intel Corporation, Literature Department, 3065 Bowers Avenue, Santa Clara, CA 95051, 1981, Chapter 2, pp. 7-14.

[7] "Context MBA", Context Management Systems, Torance CA, 1982.

[8] "ViSiON", VisiCorp, San Jose, CA 1983.

[9] iAPX 88 Book, Intel Corporation, Literature Department, 3065 Bowers Avenue, Santa Clara, CA 95051, 1981, Appendix A, pp. 4-6.

[10] Technical Reference Manual, IBM Corporation, Personal Computer, P.O. Box 1328, Boca Raton, Florida 33432, 1981, pp. 2-6 and D-2.

[11] Ray Duncan, "Intel's 8087 Numeric Data Processor", Dr. Dobb's Journal, Vol. 7, No. 8, 47-50 (1982).

[12] Jerome T. Coonen, "An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic", Computer, Vol. 13, No. 1, 68-79 (1980).

[13] iAPX 86,88 User's Manual, Intel Corporation, Literature Department, 3065 Bowers Avenue, Santa Clara, CA 95051, 1981, Supplement S.

[14] R. B. Simington, "The Intel 8087 Numerics Processor Extension", BYTE, Vol. 8, No. 4, 154-175 (1983).

[15] W L Computer Systems, 1910 Newman Road, West Lafayette, Indiana, 47906.

[16] Professional 300 Series Technical Manual, Digital Equipment Corporation, Accessories and Supplies Group, P.O.Box CS2008, Nashua, NH 03061, 1982, Chapter 5.

[17] Microcomputers and Memories, Digital Equipment Corporation, Maynard, MA, 1982, 687-688.

[18] G. Scott Owen, "Benchmarking the 8087 Numeric Coprocessor", Personal Computer Age, Vol. 2.3, 56-60 (1983).

[19] Gary Rantz, Private Communication (1983).

[20] The double precision time (182 seconds) published by Owen [18] is in error. To use square root and the transcendental functions in double precision mode, BASIC release 2.0 must be used, and /D must be specified on the BASIC command line.

[21] 8087 support software obtained by Owen [18] from Microware, P.O. Box 79, Kingston, MA 02364.

[22] Peter M. Kogge, "An Architectural Trail to Threaded-Code Systems", Computer, Vol. 15, No. 3, 22-32 (1982).

[23] F. W. J. Olver, "Bessel Functions of Integer Order", Handbook of Mathematical Functions, National Bureau of Standards, Washington, D.C., 369 (1964).