INTRODUCTION

The volume and complexity of data processed by today's personal computer is increasing exponentially, placing incredible demands on the microprocessor. New communications, games and "edutainment" applications feature video, 3-D, animation, audio and virtual reality, all of which demand ever increasing levels of performance.

MMX™ technology is designed to accelerate multimedia and communications software. The technology includes new instructions and data types that allow applications to achieve significant new levels of performance. It exploits the parallelism inherent in many multimedia and communication algorithms, yet maintains full compatibility with existing operating system and application software.

MMX technology is the most significant enhancement to the Intel Architecture since the Intel386™ processor, which extended the architecture to 32 bits. Processors enabled with MMX technology will deliver enough performance to execute compute-intensive communication and multimedia tasks with headroom left to run other tasks or applications. It allows software developers to design richer, more exciting applications for the PC. The volume of MMX technology-enabled systems will grow rapidly in 1997 as the technology is incorporated into multiple-processor generations from Intel.

The definition of MMX technology resulted from a joint effort between Intel's microprocessor architects and software developers. A wide range of software applications were analyzed, including graphics, MPEG video, music synthesis, speech compression, speech recognition, image processing, games, video conferencing and more. These applications were broken down to identify the most compute-intensive routines, which were then analyzed in detail using advanced computer-aided engineering tools. The results of this extensive analysis showed many common, fundamental characteristics across these diverse software categories. The key attributes of these applications were:

- Small integer data types (for example: 8-bit graphics pixels, 16-bit audio samples)
- Small, highly repetitive loops
- Frequent multiplies and accumulates
- Compute-intensive algorithms
- Highly parallel operations

MMX technology is designed as a set of basic, general purpose integer instructions that can be easily applied to the needs of the wide diversity of multimedia and communications applications. The highlights of the technology are:

- Single Instruction, Multiple Data (SIMD) technique
- 57 new instructions
- Eight 64-bit wide MMX registers
- Four new data types

The basis for MMX technology is a technique called Single Instruction, Multiple Data (SIMD). This allows many pieces of information to be processed with a single instruction, provide parallelism that greatly increases performance. This technology combined with the IA superscalar architecture will provide substantial performance enhancement to the PC platform. MMX technology is integrated into Intel Architecture processors in a way that maintains full compatibility with existing operating systems, including MS DOS*, Windows* 3.1, Windows 95, OS/2* and Unix*. In addition, the full base of Intel architecture software will run on MMX technology-enabled systems.

MMX technology was defined to be simple. MMX technology is general enough to address the needs of a large domain of PC applications built from current and future algorithms. MMX

instructions are not privileged; they can be used in applications, codecs, algorithms, and drivers. These instructions are also optimized for short arithmetic where the overhead for converting to and from floating-point is significant.

DATA TYPES

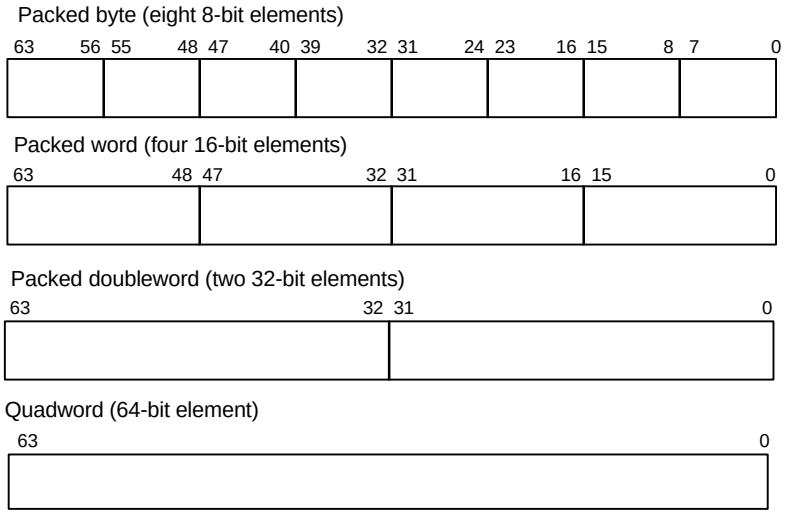
The principal data type of the IA MMX instruction set is the packed, fixed-point integer, where multiple integer words are grouped into a single 64-bit quantity. These 64-bit quantities are moved into the 64-bit MMX registers. The decimal point of the fixed-point values is implicit and is left for the programmer to control for maximum flexibility. The supported data types are signed and unsigned fixed-point integers, bytes, words, doublewords and quadwords.

The four MMX technology data types are:

Packed byte	Eight bytes packed into one 64-bit quantity
Packed word	Four 16-bit words packed into one 64-bit quantity
Packed doubleword	Two 32-bit double words packed into one 64-bit quantity
Quadword	One 64-bit quantity

As an example, graphics data are generally represented in 8-bit integers, or bytes. With MMX technology, eight of these pixels are packed together in a 64-bit quantity and moved into an MMX register. When an MMX instruction executes, it takes all eight of the pixel values at once from the MMX register, performs the arithmetic or logical operation on all eight elements in parallel, and writes the result into an MMX register.

Data Types in 64-bit Registers



COMPATIBILITY

MMX technology retains its full compatibility with existing operating systems and applications by aliasing its registers and state upon the IA floating-point registers and state. No new registers or state are added to support MMX technology. This means the operating system uses the standard mechanisms for interacting with the floating point state to save and restore MMX code. For example, during a task switch, the operating system would use an FSAV and FRSTR to preserve either floating point or MMX code. Aliasing the MMXstate upon the floating-point state does not preclude applications from executing both MMX technology routines and floating point routines.

Floating-point instructions that save/restore the floating-point state also handle the MMX

state (for example, during context switching). The same techniques used by the floating-point architecture to interface with the operating system are used by MMX technology. MMX technology does not introduce any new exception or state information, so today's operating systems can enable applications using MMX instructions.

DETECTING THE PRESENCE OF MMX ™ TECHNOLOGY

Detecting existence of MMX technology on an Intel microprocessor is done by executing the CPUID instruction and checking a set bit. This gives software developers the flexibility to determine the specific code in their software to execute. During install or run time the software can query the microprocessor to determine if MMX technology is supported and install or execute the code that includes, or does not include, MMX instructions based on the result.

INSTRUCTIONS

The MMX instructions cover several functional areas including:

- Basic arithmetic operations such as add, subtract, multiply, arithmetic shift and multiply-add
- Comparison operations
- Conversion instructions to convert between the new data types - pack data together, and unpack from small to larger data types
- Logical operations such as AND, AND NOT,OR, and XOR
- Shift operations
- Data Transfer (MOV) instructions for MMX register-to-register transfers, or 64-bit and 32-bit load/store to memory

Arithmetic and logical instructions are designed to support the different packed integer data types. These instructions have a different op code for each data type supported. As a result, the new MMX technology instructions are implemented with 57 op codes.

MMX technology uses general-purpose, basic instructions that are fast and are easily assigned to the parallel pipelines in Intel processors. By using this general-purpose approach, MMX technology provides performance that will scale well across current and future generations of Intel processors.

MMX™ Instruction Set Summary

The instructions and corresponding mnemonics in the table below are grouped by categories of related functions.

If an instruction supports multiple data types—byte (B), word (W), doubleword (DW), or quadword (QW), the datatypes are listed in brackets. Only one data type may be chosen for a given instruction. For example, the base mnemonic PADD (packed add) has the following variations: PADDB, PADDW, and PADDD. The number of opcodes associated with each base mnemonic is listed.

Category	Mnemonic	Number of DifferentOpcodes	Description
Arithmetic	PADD[B,W,D]	3	Add with wrap-around on [byte, word, doubleword]
	PADDS[B,W]	2	Add signed with saturation on [byte, word]
	PADDUS[B,W]	2	Add unsigned with saturation on [byte, word]
	PSUB[B,W,D]	3	Subtraction with wrap-around on [byte, word, doubleword]
	PSUBS[B,W]	2	Subtract signed with saturation on [byte, word]
	PSUBUS[B,W]	2	Subtract unsigned with saturation on [byte, word]
	PMULHW	1	Packed multiply high on words
	PMULLW	1	Packed multiply low on words
	PMADDWD	1	Packed multiply on words and add resulting pairs
Comparison	PCMPEQ[B,W,D]	3	Packed compare for equality [byte, word, doubleword]
	PCMPGT[B,W,D]	3	Packed compare greater than [byte, word, doubleword]
Conversion	PACKUSWB	1	Pack words into bytes (unsigned with saturation)
	PACKSS[WB,DW]	2	Pack [words into bytes, doublewords into words] (signed with saturation)
	PUNPCKH [BW,WD,DQ]	3	Unpack (interleave) high-order [bytes, words, doublewords] from MMX™ register
	PUNPCKL [BW,WD,DQ]	3	Unpack (interleave) low-order [bytes, words, doublewords] from MMX register
Logical	PAND	1	Bitwise AND
	PANDN	1	Bitwise AND NOT
	POR	1	Bitwise OR
	PXOR	1	Bitwise XOR
Shift	PSLL[W,D,Q]	6	Packed shift left logical [word, doubleword, quadword] by amount specified in MMX register or by immediate value
	PSRL[W,D,Q]	6	Packed shift right logical [word, doubleword, quadword] by amount specified in MMX register or by immediate value
	PSRA[W,D]	6	Packed shift right arithmetic [word, doubleword] by amount specified in MMX register or by immediate value
Data Transfer	MOV[D,Q]	4	Move [doubleword, quadword] to MMX register or from MMX register
FP & MMX State Mgmt	EMMS	1	Empty MMX state