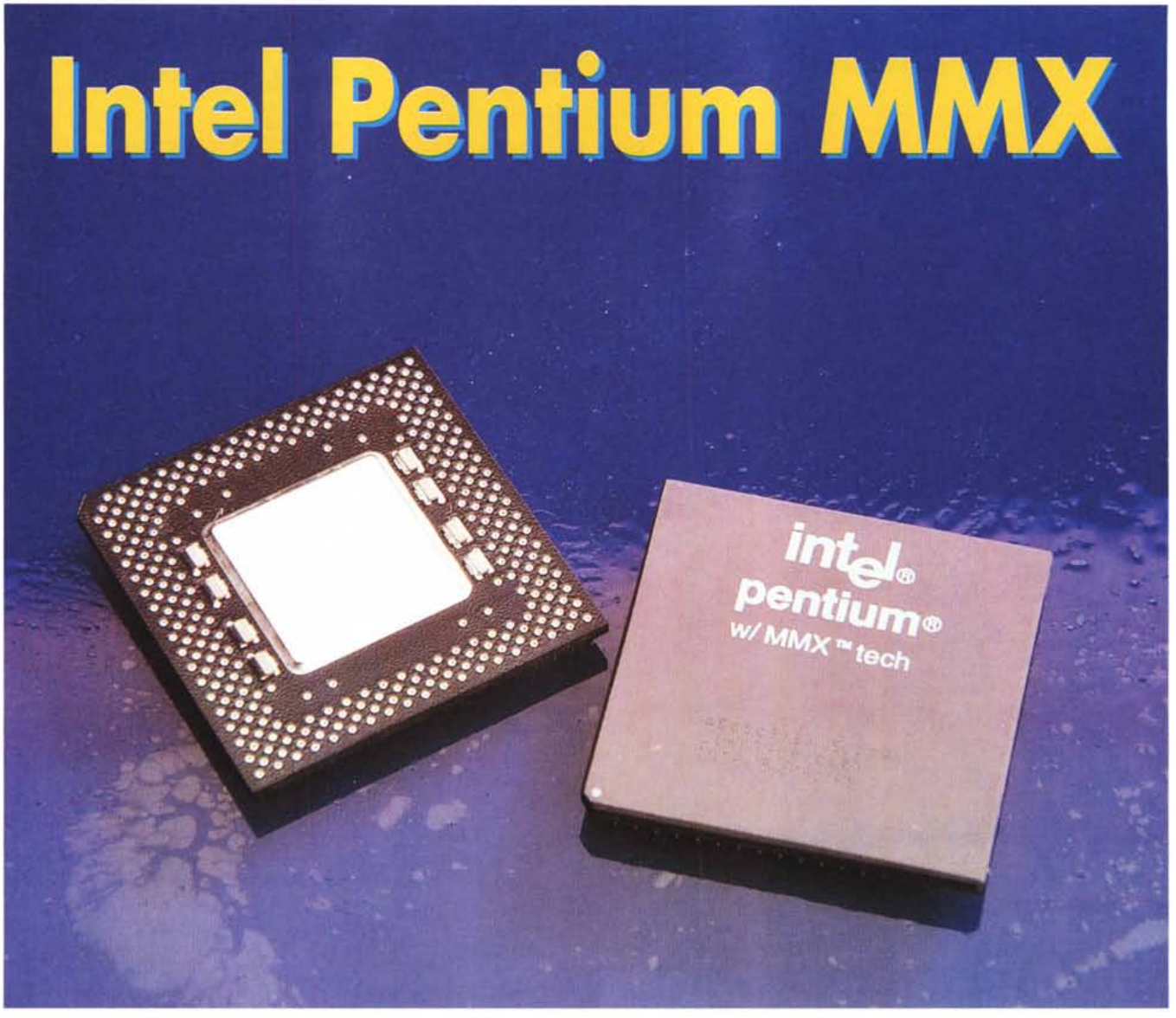
Prova

di Andrea de Prisco

Intel Pentium MMX



"La compatibilità è quella cosa con la quale e per la quale... si rimane tale e quale". Ai tempi del liceo ci divertivamo ad etichettare in tal senso (e a turno) più o meno tutte le materie scolastiche. C'era chi ce l'aveva a morte con la filosofia e il latino, chi (non certo il sottoscritto) diceva la stessa cosa per la matematica, la fisica, la chimica. A ben guardare, la compatibilità potrebbe diventare presto anch'essa una materia scolastica o addirittura un corso d'esame universitario. Quando, invece, mi va di fare il polemico (per usare un eufemismo...) mi diverte ricordare che sul vocabolario, compatibile, sta per "può essere compatito". E me la rido sotto i baffi e sopra la barba...

Per quanto possa sembrare, ad una prima analisi, piuttosto strano, la compatibilità col passato è uno degli elementi chiave dello sfrenato consumismo personal-informatico degli ultimi anni. All'inizio ti vendo un computer e una collezione di programmi. Poi i programmi aumentano di numero, ma soprattutto la potenza di calcolo richiesta per la loro esecuzione ed ecco giunto il momento per venderti una nuova macchina, più potente della prima, sulla quale però è possibile "far girare", in modalità fulminea, tutti i tuoi vecchi programmi ormai inusabili per lentezza. Ovviamente non finisce qui: con le nuove potenze in gioco è facile, per così dire, programmare funzioni ancora più evolute che richiedono maggiori performance di calcolo... e nel giro di pochi mesi ci ritroviamo al punto di prima: 'sto computer è troppo lento!

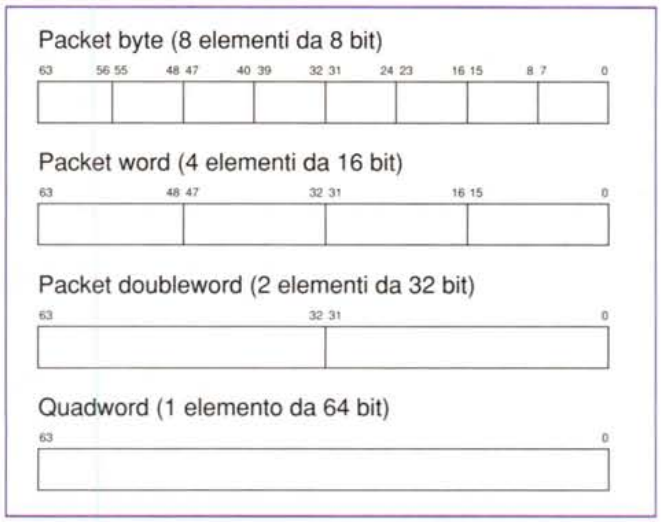
Nuovi microprocessori, nuove schede madri, nuovi hard disk, nuova spesa per l'acquisto del modello superturbo e, da lì a poche settimane, nuovi software stressa CPU, come prima più di prima.

Finiremo mai di giocare a Lascia o Raddoppia? La risposta è scontata: non solo continueremo in tal senso, ma sa-

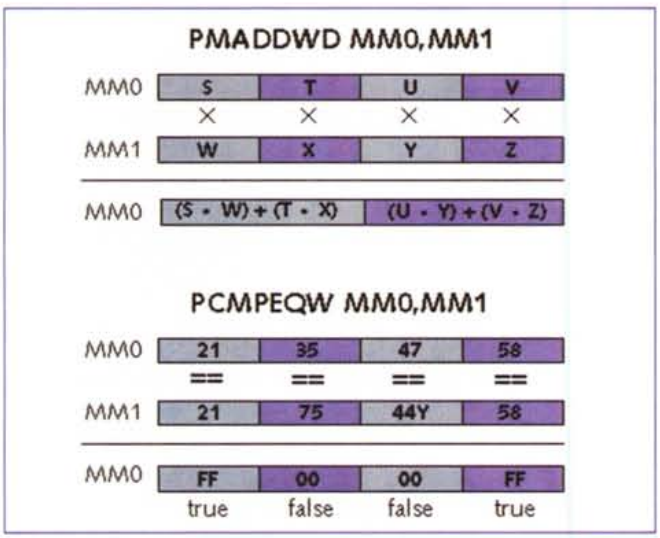
remo praticamente costretti a scegliere sempre la seconda ipotesi, almeno fintantoché sarà possibile offrire potenze di calcolo maggiori.

Nel corso degli ultimi anni abbiamo visto implementare nelle CPU dei personal computer ogni possibile bendidio tecnologico per il raggiungimento di performance di calcolo sempre più esagerate. Prima le pipeline a pochi stadi, poi quelle a molti stadi, doppie pipeline, meccanismi di branch-prediction tipo "sfera di cristallo" per tirare ad indovinare dove il programma, da lì a poche istruzioni macchina, andrà a parare in seguito a un'operazione di salto condizionato. CPU in tutte le salse talmente tanto veloci da avere grosse difficoltà di interfacciamento col mondo esterno, ma non solo con le periferiche più lontane (questo, tutto sommato, può anche essere ovvio) ma finanche con la memoria centrale che deve solo cedere e ricevere dati memorizzati o da memorizzare. E giù di cache più o meno grandi, secondo schemi di funzionamento più o meno evoluti, addirittura integrate dentro al "package" del microprocessore, come avviene per l'ancora sottosfruttato Pentium Pro (P6) e già, ovviamente, si parla di P7.

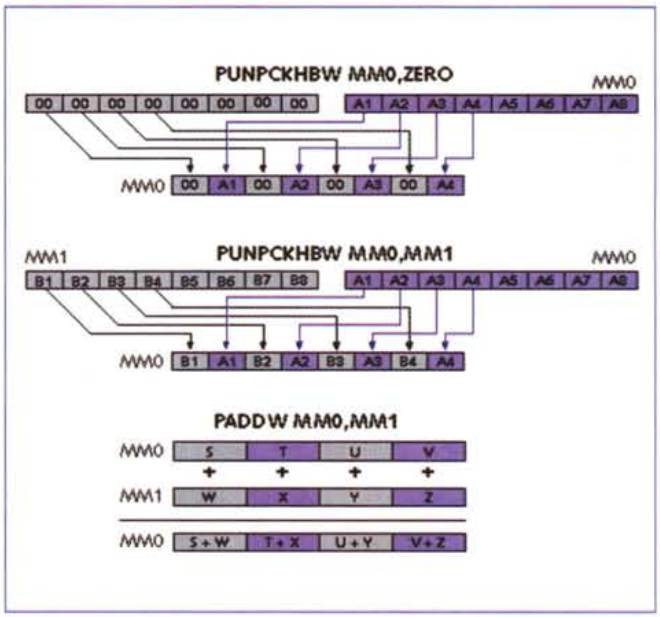
Nel mondo del software, ad agitare energicamente le acque, da qualche anno se ne stanno occupando principalmente le applicazioni multimediali. Lì fare il passo più lungo della gamba è molto facile: quando c'è di mezzo l'audio di qualità, il video idem, l'interattività, l'animazione e la grafica 3D più evoluta o l'accesso immediato a sempre più grandi masse di dati multimediali, non esistono più limiti "decenti" alla potenza di calcolo richiesta. Qualsiasi microprocessore, per quanto evoluto e stravolgente che sia, prima o poi (sicuramente prima...) mostra i suoi limiti specialmente nel caso in cui si utilizzi una tecnologia di calcolo tradizionale. E la ricerca, in più direzioni, di nuove fonti di potenza di calcolo ha indotto i progettisti Intel a rispolverare la ben nota architettura SIMD (Single Instruction-stream, Multiple Data-stream) con la quale è possibile eseguire la medesima istruzione su più dati contemporaneamente. Per inciso i processori tradizionali (dal più vetusto 8088 fino ai Pentium Pro, ma fanno parte del medesimo club anche i vari 6802, Z80, 680x0, PowerPC, Alpha, ecc.) seguono la linea SISD (Single Instruction-stream, Single Data-stream), ovvero una singola istruzione su un singolo flusso di dati, mentre le macchine multiprocessor appartengono alla categoria MIMD (Multiple Instruction-stream, Multiple Data-stream) che identifica la possibilità di eseguire istruzioni differenti su differenti flussi di dati. Per chi fosse maggiormente interessato all'argomento rimando al riquadro "Tassonomia di Flynn".

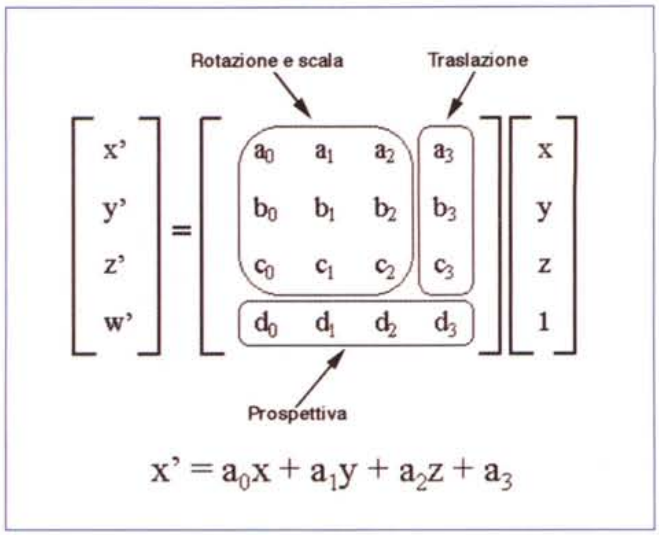


Gli otto registri del Pentium MMX possono essere utilizzati come "pacchetti" di otto byte, quattro word, due double-word o come interi a 64 bit (Quadword).



I dati "impacchettati" nei registri MMX (schema a lato) possono essere utilizzati dalla nuova unità per l'elaborazione in parallelo. Nello schema in alto l'istruzione "moltiplica & somma" e, in basso, un confronto parallelo.





Una tipica applicazione SIMD: il calcolo matriciale. Nell'esempio mostrato qui a lato, il vettore (X,Y,Z,1) è moltiplicato per una matrice 4x4 che comprende una rotazione e una variazione del fattore di scala, una traslazione e correzione prospettrica. Con la tecnologia MMX si riduce di circa due terzi il numero di istruzioni necessarie per il calcolo.

Nasce così la prima significativa estensione dell'architettura Intel, denominata MMX (acronimo, non ufficiale, di MultiMedia eXtension) del dopo-386, periodo in cui si fece il grande salto verso l'architettura a 32 bit. Cinquantasette nuove istruzioni eseguite da una nuova sottounità logica, l'MMX extension, che operano in parallelo su insiemi di dati. Fino a otto byte trattati contemporaneamente ma anche quattro word o due double-word (rispettivamente da 16 o 32 bit) per effettuare somme, sottrazioni, moltiplicazioni, operazioni di "moltiplica e somma", confronti, shift logici ed altro. Il tutto nel rispetto (maledetto...) della compatibilità col passato: qui, forse, la vera e propria "fregatura"...

Qui comincia l'avventura

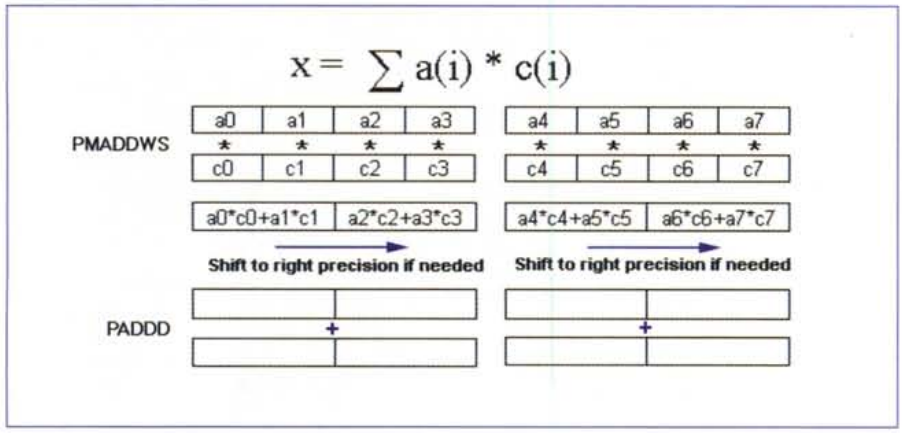
Freschi freschi com'eravamo di Windows 95 (per il quale, l'ottenimento di un vero sistema operativo 32 bit, abbiamo aspettato quasi una decina d'anni!), non era proprio il caso di stravolgere nuovamente la carte in tavola proponendo un prodotto "poco compatibile" con gli attuali sistemi operativi in circolazione. L'obiettivo di Mamma Intel doveva necessariamente essere quello di fornire un nuovo microprocessore che potesse essere assolutamente trasparente all'hardware e al software attuale, offrendo al contempo una significativa spinta tecnologica per il raggiungimento delle alte prestazioni in applicazioni multimediali. E così l'estensione MMX è una sorta di coprocessore vettoriale che entra in gioco invocando una delle sue nuove istruzioni che è in grado di ese-

Tassonomia di Flynn (e sai cosa bevi...)

di Andrea de Prisco

Dietro questo misterioso - quanto esotico! - titolo si cela una delle più valide e universalmente riconosciute classificazioni delle macchine parallele del tipo "von Neumann": questi sono i dati, questo è il programma... elabora!
Iniziamo col dire che essa risale addirittura ai primordi dell'elaborazione parallela e più precisamente alla seconda metà degli anni Sessanta, periodo in cui le calcolatrici da tavolo erano ancora meccaniche, gli articoli si scrivevano ancora con le macchine per scrivere, e dovevano ancora passare una quindicina d'anni prima dell'avvento dei microcomputer poi divenuti, grazie e IBM, personal computer.
La classificazione di Flynn si basa fondamentalmente sui due concetti tipici delle macchine "von Neumann": il flusso di istruzioni e il flusso di dati. Il modello "von Neumann" rappresenta in pratica l'architettura di calcolo più tradizionale possibile: un "qualcosa" contenente dei dati, un altro "qualcosa" contenente un programma (una sequenza di istruzioni), e un ultimo "qualcosa" in grado di eseguire le istruzioni sui dati per ottenere risultati. Tutti i personal computer finora prodotti si rifanno a tale modello, ma sono macchine von Neumann anche i supercomputer, i mainframe, le workstation e... le centraline d'iniezione elettronica dei motori, la logica di controllo e programmazione del vostro videoregistratore, nonché quella di qualsiasi altro dispositivo elettronico digitale in cui sia presente anche mezzo microprocessore.
Dicevamo, flusso istruzioni e flusso dati: a seconda della molteplicità dell'uno o dell'altro flusso individuiamo quattro ben precise architetture parallele: le macchine SISD, SIMD, MISD, MIMD. In questi quattro acronimi le "S" stanno per Single, "M" per Multiple, "I" per Instruction (nel senso di "flusso di istruzioni"), "D" per Data (nel senso di "flusso di dati"). Così una macchina SISD ha flusso di istruzioni e flusso di dati singolo (Single Instruction-stream, Single Data-stream) una macchina MIMD ha flusso dati e flusso istruzioni multiplo (Multiple Instruction-stream, Multiple Data-stream).

Da qui una prima, probabile, obiezione: se un computer è SISD (lavora con un flusso singolo di dati e un altrettanto singolo flusso di istruzioni) perché classificarlo tra le architetture parallele? La risposta è molto semplice: il parallelismo può benissimo essere presente solo all'interno del processore nel modo in cui lo stesso elabora le istruzioni. Tutti i personal computer finora prodotti sono certamente macchine SISD, ma quelli più recenti dispongono comunque di un'architettura parallela in quanto la pipeline interna al processore fa sì che in ogni istante vi siano più istruzioni in esecuzione, pur appartenenti allo stesso flusso. Altrettanto singolo è il flusso dei dati in quanto il processore (unico) accede ai dati in memoria più o meno così: prendo questo dato qui, prendo quest'altro dato lì, scrivo il risultato in questa determinata cella, ecc. ecc.
Il primo vero salto di qualità lo incontriamo con le macchine SIMD (Single Instruction-stream, Multiple Data-stream). A questa classe appartengono principalmente i computer vettoriali, più propriamente detti "array processor" e, da qualche settimana anche i moderni Pentium MMX. In queste macchine una singola istruzione viene eseguita contemporaneamente su un insieme di dati. Immaginiamo ad esempio di eseguire la somma di due vettori: il processore avrà al suo interno un'unica unità istruzioni (che si occupa della decodifica delle stesse) più un certo numero di unità esecutive capaci di lavorare in parallelo. Dovendo ad esempio effettuare la somma di due vettori (come vedete il flusso delle istruzioni è singolo, in ogni istante una ed una sola istruzione "entra" nel processore), le unità esecutive riceveranno in parallelo (il flusso dati è invece multiplo) ognuna un elemento del primo vettore, altrettanto in parallelo il corrispondente elemento del secondo vettore, per eseguirne la somma di tutti gli elementi in un colpo solo ed avere come risultato un terzo vettore.
Nelle architetture MIMD, tanto i dati quanto le istruzioni vengono processati secondo flussi multipli. In ogni istante più istruzioni (differenti) operano parallelamente su più dati (differenti). E' questo il



L'istruzione PMADD esegue quattro moltiplicazioni e due somme in un solo colpo. Grazie al suo utilizzo risulta molto semplificato il calcolo delle sommatorie.

guire e, per quel che riguarda l'interfacciamento con la rimanente CPU, prende letteralmente il posto del coprocessore matematico. Detto in altre parole, la sezione MMX dei nuovi microprocessori Pentium utilizza gli stessi registri del coprocessore matematico integrato per il passaggio dei dati e quindi se utilizziamo l'estensione multimediale dobbiamo dimenticare momentaneamente l'unità floating point.

Per dirla alla Lubrano, la domanda, a questo punto, nasce spontanea: Che c'entra la compatibilità degli attuali sistemi operativi con la sovrapposizione logica delle due unità di processo (MMX e FP)? Semplice: è tutta colpa del multitasking. Cerchiamo, nella maniera più semplice possibile, di capire dov'è l'inghippo. Nei moderni sistemi operativi multitasking (come Windows 95) la "commutazione di contesto" tra l'esecuzione di differenti task (i programmi in esecuzione) avviene in modalità preemptive, ovvero quando meno se l'aspettano. Il processore, a causa di un evento asincrono indipendente dall'esecuzione del programma in corso, sospende la sua elaborazione e inizia quella di un altro task. Lo stesso si ripete da lì a pochi ulteriori millesimi di secondo e dunque il processore, per mandare avanti in parallelismo simulato l'esecuzione di più task, suddivide il suo tempo macchina tra i vari lavori da eseguire dedicandosi ora ad un programma, ora ad un altro, poi ad un altro ancora e così via. Ogni volta che si ha una commutazione di contesto (l'abbandono momentaneo di un task per iniziare/continuare l'esecuzione di un lavoro successivo) il sistema operativo deve salvare lo stato del processo attualmente in esecuzione per poterlo riprendere esattamente dove l'aveva lasciato al suo successivo riutilizzo. Per salvare lo stato

caso delle macchine multiprocessor che a loro volta si suddividono in due grosse categorie: le macchine ad ambiente globale e quelle ad ambiente locale. Nelle prime la memoria principale è unica per tutti i processori i quali accedono a questa attraverso un ben preciso meccanismo di arbitraggio che evita collisioni sugli accessi in memoria. Nelle seconde ogni processore ha la sua memoria "privata" e colloquia con gli altri processori attraverso una struttura di interconnessione. In più, mentre per le macchine a memoria unica non si può aumentare più di tanto il numero di processori poiché oltre un certo valore l'overhead determinato dall'arbitraggio fa sì che le prestazioni peggiorino invece di migliorare, per le macchine ad ambiente locale possiamo collegare tra loro quanti processori vogliamo (anche migliaia o milioni) fino a costruire quelle che comunemente sono dette macchine a parallelismo massiccio.

Non abbiamo ancora parlato delle macchine MISD, ugualmente classificate da Flynn per motivi di simmetria, che però hanno ben poco a che fare con la realtà informatica: non si è ancora riusciti a trovare (per la verità non credo che qualcuno ci abbia pensato per più di tre minuti di seguito) una possibile applicazione di una macchina MISD. A costruirla, infatti, non ci vorrebbe nulla: il problema rimarrebbe solo "per farci cosa?". Verrebbe fuori una macchina nella quale in parallelo più programmi diversi tra loro elaborano lo stesso flusso di dati per ottenere, parimenti, un unico flusso di risultati.

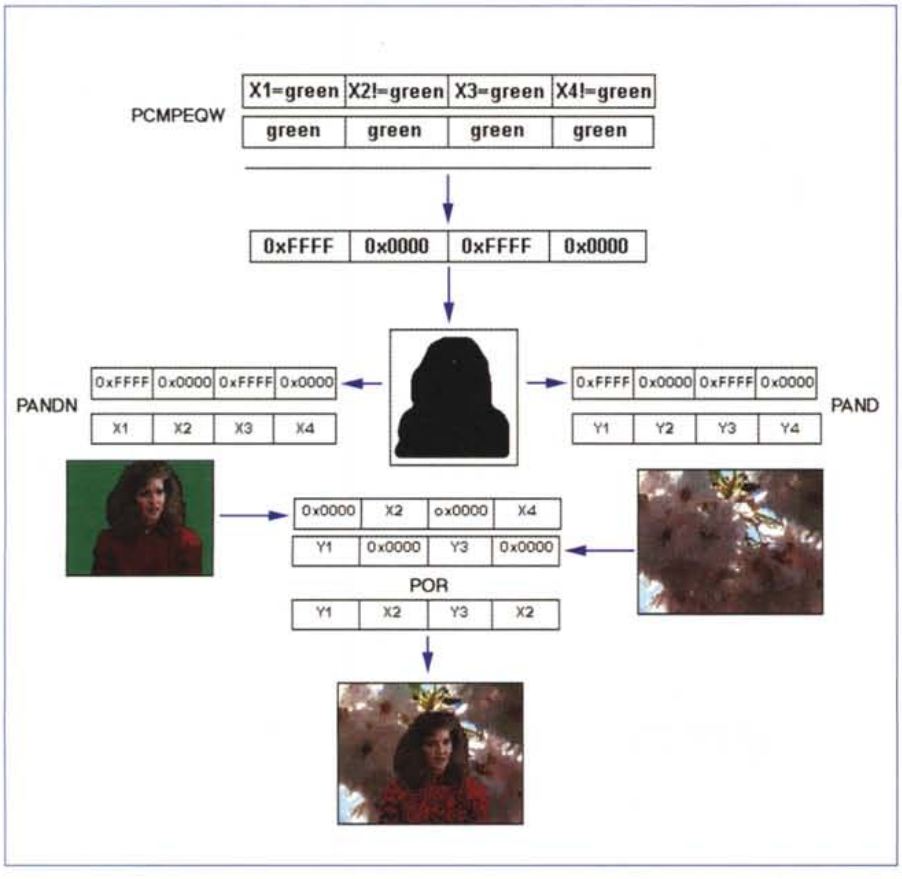
Per concludere questa breve carrellata sulle architetture parallele è necessario sottolineare il fatto che i risultati ottenibili dalle architetture parallele sono fortemente dipendenti dalle applicazioni. Per fare calcolo vettoriale non serve una macchina a parallelismo massiccio utilizzabile solo ed esclusivamente per problemi intrinsecamente paralleli. Le macchine multiprocessor a memoria comune vanno molto bene per le applicazioni parallele in cui più processi diversi cooperano per portare a termine l'elaborazione su dati condivisi. Non appena ci spostiamo dai rispettivi campi di applicazione potremmo avere risultati a dir poco deludenti: inutile tentare di fare fuoristrada con una Testarossa o partecipare al Gran Premio di Monza con un camper.

L'unica cosa da aggiungere, a questo punto, riguarda naturalmente la possibilità di avere architetture miste ad esempio macchine MIMD nelle quali ogni processore è una macchina SIMD (computer multiprocessor parallelo) o semplicemente SISD ma con parallelismo interno come visto prima. E così via...

di un processo, principalmente, il sistema operativo conserva in un'opportuna zona di memoria il contenuto dei registri dei microprocessore in quel momento utilizzati, il valore del Program Counter (l'indirizzo di memoria della prossima istruzione da eseguire) più il contenuto della/delle Processus Status Word ovvero tutti i bit di stato del processo in esecuzione. Se l'estensione MMX dei nuovi Pentium avesse avuto dei propri registri di interfacciamento diversi da quelli già presenti nei precedenti microprocessori sarebbe stato necessario modificare anche i sistemi operativi in modo tale da permettere il salvataggio dello stato anche relativamente a queste nuove microlocazioni interne alla CPU. Mappando, viceversa, i registri MMX negli stessi registri dell'unità floating point, il sistema operativo non dovrà essere modificato in quanto una commutazione di contesto durante l'esecuzione di codice MMX non è (dal suo punto di vista) assolutamente differente dal caso (già noto) del codice relativo al coprocessore matematico. Un "trucco", tutto sommato pulito, grazie al quale è assicurata la compatibilità

Category	Mnemonic	Number of Different Opcodes	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	Subtract 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 MMXTM 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]	4	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

Il set di istruzioni MMX comprende funzionalità aritmetiche, di comparazione, conversione, logiche e di trasferimento dati. In tutto 57 nuovi opcode che operano, in parallelo, su dati impacchettati.

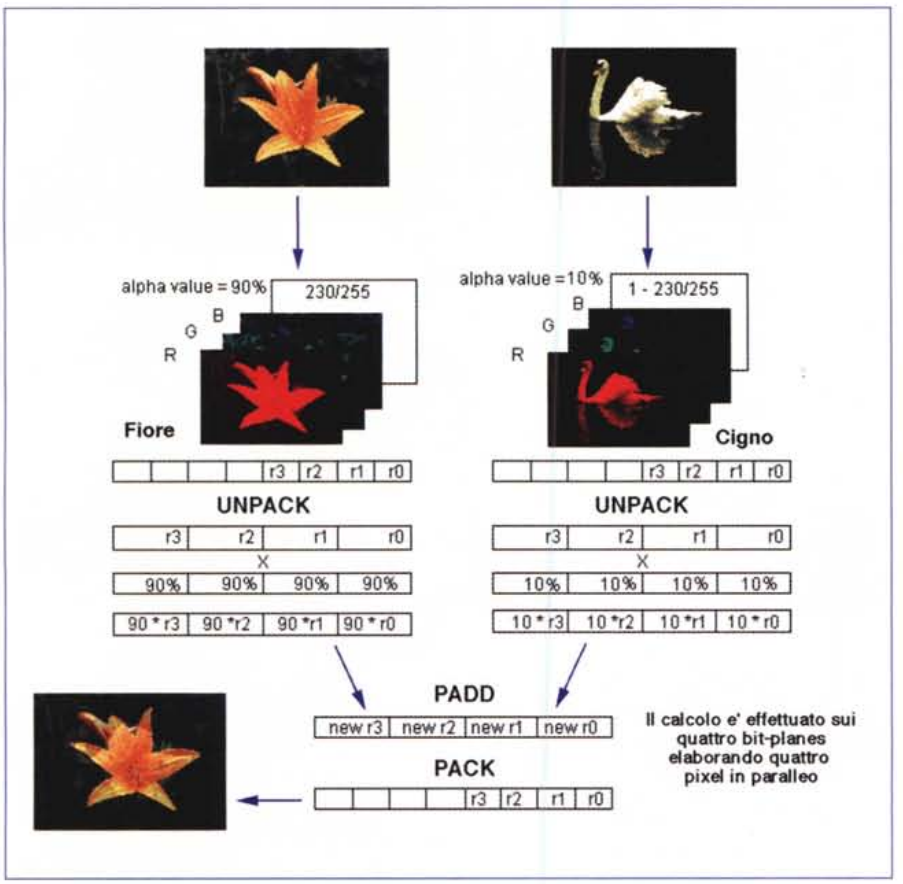


Il Chroma-key digitale è una vera e propria passeggiata software per l'architettura Intel MMX. Dall'immagine di partenza si preleva una maschera logica che è utilizzata "bucare" il fondo. Il risultato è il soggetto sovrapposto al nuovo sfondo: il tutto, ovviamente, in tempo reale su immagini in movimento.

software anche in tal senso, ma che ha come non trascurabile handicap il fatto che è impossibile utilizzare l'unità floating point contemporaneamente all'unità MMX. E' quindi necessario ingegnerizzare opportunamente il codice dei nuovi programmi specificamente realizzati per i nuovi Pentium MMX per evitare continue (e gravose) commutazioni tra i due dispositivi di coprocesso.

Il software che verrà

Alla luce di queste iniziali considerazioni, non possiamo far altro che augurarci di veder comparire, da qui a pochi mesi, vere e proprie valanghe di nuovi programmi e applicativi compatibili sia con la nuova architettura MMX che con le macchine tradizionali. E' supportata, infatti, la possibilità di riconoscere via software la presenza della nuova architettura ed è così possibile sia installare



Anche la dissolvenza incrociata tra immagini non rappresenta alcun problema per l'architettura Intel MMX. Il flusso di dati corrispondenti alle due sequenze video viene impacchettato in word (16 bit), moltiplicato "al volo" con i corrispondenti valori di alpha blending (tra loro complementari) e sommato per ottenere la sequenza video finale.

il codice specifico per la macchina sul quale dovrà "girare" sia confezionare opportunamente codice "fat" contenente entrambe le modalità elaborative. Le applicazioni in cui MMX la farà da padrone saranno senza dubbio quelle multimediali, intendendo con questo tutti quei programmi che fanno largo uso di elaborazione numerica su grossi insiemi di byte o word, come possono essere le funzionalità audio/video/grafica 3D. Non è escluso, però, che la tecnologia MMX verrà sfruttata anche in ambiente tecnico/scientifico, così come utilizzata da future estensioni di sistema operativo per l'elaborazione numerica di segnali digitali incorporando, ad esempio, funzionalità particolarmente evolute in futuri driver di stampa per pilotare al meglio anche le macchine più "stupide". Discorso analogo per il riconoscimento vocale, la sintesi musicale, l'elaborazione audio in genere.

Simpatica, infine, da segnalare la presenza (nascosta) di codice MMX in applicazioni già presenti da tempo sul mercato, come nel caso di Adobe PhotoDeLuxe (programma di fotoelaborazione digitale dedicato al mercato consumer) il quale offre performance decisamente superiori sulle nuove macchine, senza che l'utente debba reinstallare il pacchetto. Very good!

Dati e istruzioni

L'unità MMX dei nuovi Pentium utilizza otto registri da 64 bit mappati, come ripetuto più volte, all'interno dei registri dell'unità floating point. A seconda del formato dell'istruzione MMX utilizzata ogni registro a 64 bit può essere interpretato come un "pacchetto" di 8 byte (ognuno da 8 bit), di 4 word (da 16 bit), 2 double-word (da 32 bit) o il nuovo dato Quad-Word, intero a 64 bit, per i programmatori incontentabili. Una volta "impacchettati" più byte o più word (o double-word) nei registri è possibile effettuare istruzioni SIMD sugli stessi: un'unica istruzione che opera parallelamente su tutti i dati contenuti nei registri di origine e destinazione. Le 57 nuove istruzioni MMX sono suddivisibili in sette categorie. Alla prima categoria, delle funzionalità aritmetiche, fanno parte ben 17 istruzioni che riguardano la somma, la sottrazione, la moltiplicazione e l'operazione di "moltiplica & somma" tra registri MMX. Tutte le operazioni aritmetiche possono essere eseguite in modalità "wrap-around" o in modalità "saturation". Nel primo caso a seguito di un overflow (il risultato eccede le capacità di memorizzazione del tipo di dato utilizzato) è restituito il risultato troncato dell'operazione, nel secondo viene restituito il massimo intero rappresentabile con quella notazione. Le moltiplicazioni avvengono in due passaggi: con una prima istruzione si ottiene la parte bassa del risultato (i bit meno significativi), con una seconda istruzione la parte alta (i bit più significativi). Infine l'operazione "moltiplica & somma", come dice il ragionamento stesso, non fa altro che eseguire la moltiplicazione a due a due degli operandi effettuando successivamente la somma dei relativi prodotti: molto utile nel calcolo matriciale e, in particolare, nella trasformata di Fourier utilizzata soprattutto in ambiente audio digitale.

Alla seconda categoria di istruzioni MMX appartengono le operazioni di comparazione parallela tra registri. Il risultato è una maschera numerica True/False (composta da 0xFF e 0x00) riutilizzabile nei calcoli successivi ad esempio per selezionare con un singolo statement tutti i valori che soddisfano quel determinato criterio di confronto ("uguale" o "maggiore di").

Alla categoria "conversione" appartengono nove istruzioni per l'impacchettamento o lo spacchettamento dei dati con relativa conversione tra formati: da word a byte, da double-word a word, ecc.

Non mancano, naturalmente, categorie di istruzioni logiche, di shift, di trasferimento dati, con le quali è possibile effettuare operazioni sui bit (AND, OR, NAND, XOR), shiftare a destra o a sinistra dati impacchettati, spostare il contenuto da o verso i registri MMX.

Infine, la categoria "FP & MMX management" contempla un'unica istruzione, EMMS, che è utilizzata per rilasciare lo stato MMX e riattivare il coprocessore matematico. Come dire: rien ne va plus!