

Aula 11

SIMD

SIMD

- Taxonomia de Flynn
 - SISD – Single Instruction, Single Data (computadores convencionais)
 - SIMD - Single Instruction, Multiple Data (computadores vetoriais e multimídia)
 - MIMD – Multiple Instruction, Multiple Data (computadores paralelos)
 - MISD – Multiple Instruction, Single Data

SIMD

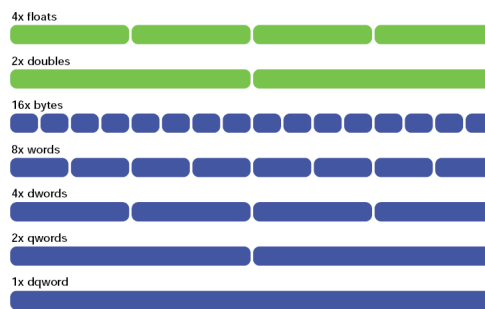
- Uma única instrução opera sobre vários valores
- Objetivo: Extrair **paralelismo** no nível de dados, diminuindo o tempo de execução
- Exemplos típicos onde o paralelismo no nível de dados é possível
 - Computação de alto-desempenho
 - Processamento de áudio
 - Processamento de imagem
 - Processamento de sinais

MMX, SSE, SSE2

- MMX – lançado no Pentium II, 1995
 - Oito registradores MMX de 64 bits (MM0 – MM7)
 - Quatro tipo de dados MMX (packed bytes, packed words, packed double words, e quad word).
 - 57 Instruções MMX

SSE, SSE2

- SSE – Pentium 3, 1999
- SSE2 – Pentium 4, 2001
- Oito registradores de 128 bits XMM0 – XMM7
- Tipos de dados:



Anything that fits into 16 bytes.

Declaração de variáveis

- Dados devem ser alinhados em 16 bits para que o desempenho não seja penalizado
- Utilize a diretiva `.align 16`
- Ex:

```
.section .data  
.align 16  
.vector: .fill 128, 4
```

Instruções – “Packed”

- ADDPD (Ponto flutuante 64 bits)
 - ADD packed double
- ADDPS (Ponto flutuante 32 bits)
 - ADD packed single
- PADDQ (Inteiro 64 bits)
- PADDD (Inteiro 32 bits)
- PADDW (Inteiro 16 bits)
- PADDB (Inteiro 8 bits)

Instruções – “Packed”

- ADDPS

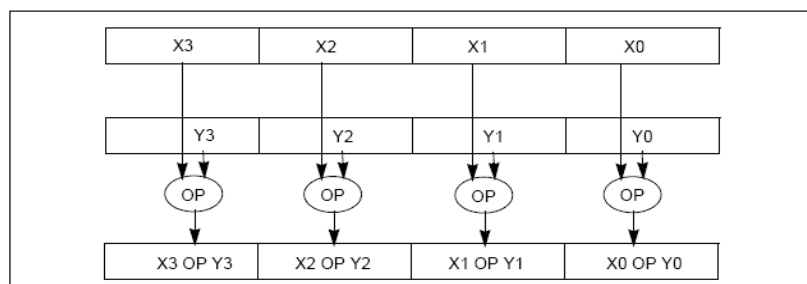


Figure 10-5. Packed Single-Precision Floating-Point Operation

Instruções – “Scalar”

- ADDSD (Ponto flutuante 64 bits)
- ADDSS (Ponto flutuante 32 bits)

Instruções – “Scalar”

- ADDSS

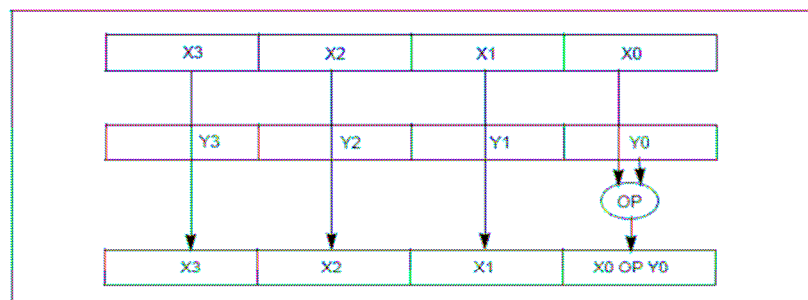


Figure 10-6. Scalar Single-Precision Floating-Point Operation

Operações aritméticas

Ponto Flutuante

- Subtração
 - SUBPD
 - SUBPS
- Multiplicação
 - MULPD
 - MULPS
- Divisão
 - DIVPD
 - DIVPS

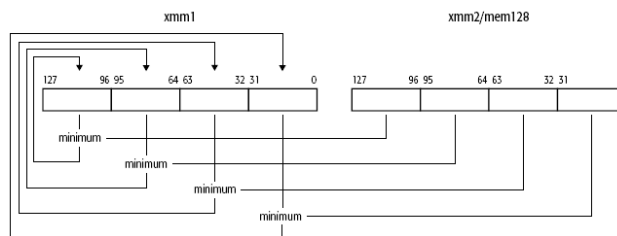
Operações lógicas

Ponto Flutuante

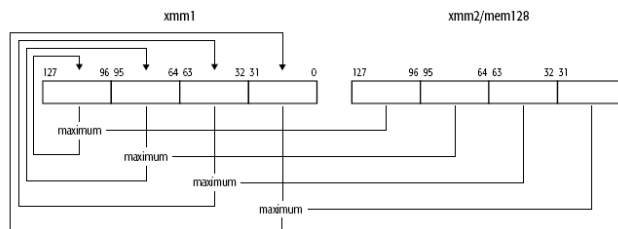
- ANDPD
- ANDPS
- ORPD
- ORPS
- XORPD
- XORPS

Funções aritméticas – Ponto flutuante

- MINPD
- MINPS



- MAXPD
- MAXPS



Funções aritméticas – Ponto Flutuante

- Raiz quadrada
 - SQRTPD XMM/MEM128, XMM
 - SQRTPS XMM/MEM128, XMM

Saturação

- As operações SSE não alteram os *flags*, portanto não é possível a detecção da ocorrência de *overflow* e *carry*.
- No processamento de imagens e áudio, é interessante que durante uma operação, o resultado seja o maior ou o menor valor possível
- $0xFF + 0x70 = 6F, CF = 1$
 - Inteiro não sinalizado MAX = $0xFF$ (255), MIN = $0x0$
 - Inteiro sinalizado MAX = $0x7F$ (+127), MIN = $0x80$ (-128)

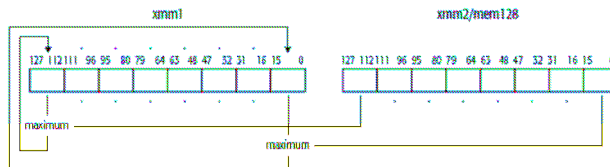
Saturação (Inteiros)

- PADD SB (packed add with saturation – 8 bits)
- PADD SW (packed add with saturation – 16 bits)
- PADD USB (packed add unsigned with saturation – 8 bits)
- PADDUSW (packed add unsigned with saturation – 16 bits)

Funções Matemáticas - Inteiros

- PMAWSW (max, signed word – inteiro 16 bits)

- PMAWSB



- PMINSW

- PMINUB

Movimentação de dados

(64 bits, ponto flutuante)

- MOVAPD XMM/Mem128, XMM
- MOVAPD XMM, XMM/Mem128

(32 bits, ponto flutuante)

- MOVAPS XMM/Mem128, XMM
- MOVAPS XMM, XMM/Mem128

Movimentação de Dados

```
.section .data
    .align 16
    dados: .fill 16, 128 #16 elementos de 128 bits

.section .text
.global main:
main:
    push %ebp
    mov %esp, %ebp
    push %esi
    mov $1, %eax
    cpuid # retorna os valores
    test $0x2000000, %edx # testa bit 25: processador implementa SSE
    jz NO_SSE
    xor %esi, %esi
    movapd dados(%esi), xmm0
    add $128, %esi
    movapd dados(%esi), xmm1
:NO_SSE
    pop %esi
    leave
    ret
```

Programa em C

```
__attribute__((aligned (16))) int v[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
    12, 13, 14, 15};
__attribute__((aligned (16))) float v1[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
    11, 12, 13, 14, 15};

__attribute__((aligned (16))) short v2[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
    11, 12, 13, 14, 15};

__attribute__((aligned (16))) char v3[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
    12, 13, 14, 0xFF};

void main()
{
    asm("movaps v3, %xmm7" );
    asm("paddusb %xmm6, %xmm7");
    asm("movaps %xmm7, v3");
}
}
```

CPUID

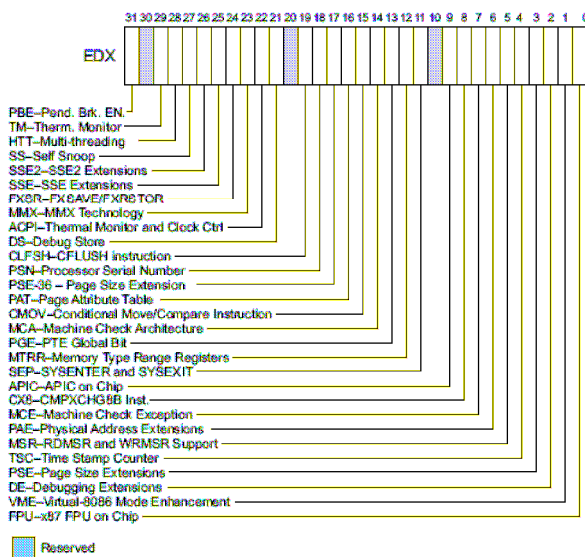
Table 3-12. Information Returned by CPUID Instruction

Initial EAX Value	Information Provided about the Processor	
	<i>Basic CPUID Information</i>	
0H	EAX EBX ECX EDX	Maximum Input Value for Basic CPUID Information (see Table 3-13) "Genu" "ntel" "Inel"
01H	EAX EBX ECX EDX	Version Information: Type, Family, Model, and Stepping ID (see Figure 3-5) Bits 7-0: Brand Index Bits 15-8: CLFLUSH line size (Value * 8 = cache line size in bytes) Bits 23-16: Maximum number of logical processors in this physical package. Bits 31-24: Initial APIC ID Extended Feature Information (see Figure 3-6 and Table 3-15) Feature Information (see Figure 3-7 and Table 3-16)
02H	EAX EBX ECX EDX	Cache and TLB Information (see Table 3-17) Cache and TLB Information Cache and TLB Information Cache and TLB Information

Fonte: <ftp://download.intel.com/design/Pentium4/manuals/25366621.pdf>

CPUID

- MOV \$1, %EAX
- CPUID



SSE2

- Ex: Transferência de dados usando SSE2
- Ex: SQRT usando SSE2 de 65536 elementos
- Ex: Soma de dois vetores de tamanho 65536
 - Ponto flutuante de 32 bits
 - Inteiro de 16 bits