Programação em *Assembly*

SIMD

IA32

Single Instruction Multiple Data

1994 – Pentium II e Pentium with MMX –
Intel introduz a primeira extensão SIMD ao conjunto de instruções (MMX - MultiMedia eXtensions)

1995 – Introdução de Streaming Simd Extensions (SSE) no Pentium III

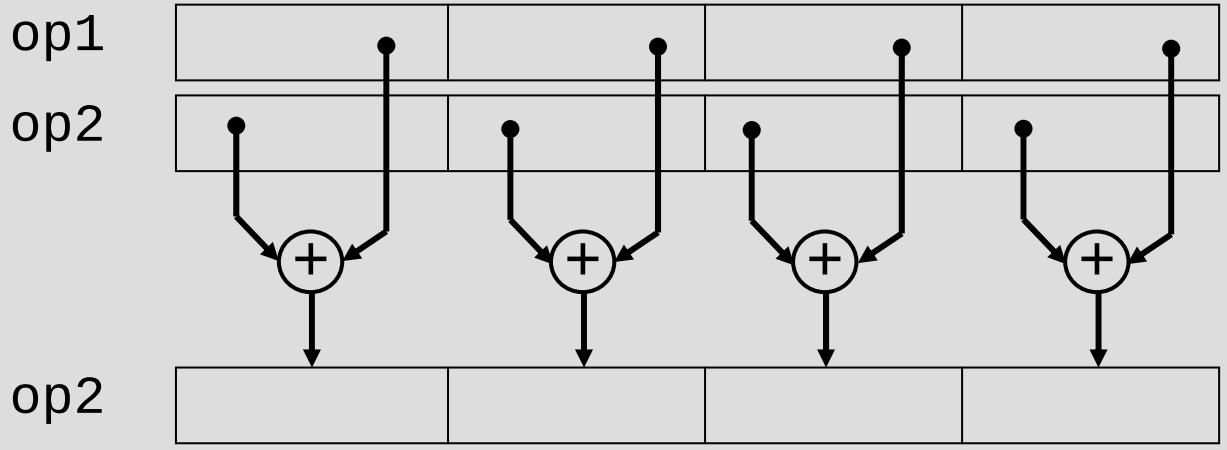
2000 – Introdução de SSE2 no Pentium IV

200… - Introdução de SSE3 no Pentium IV HT

Single Instruction Multiple Data

- A mesma instrução é aplicada a múltiplos elementos de dados

```
add_simd op1, op2
```



Single Instruction Multiple Data

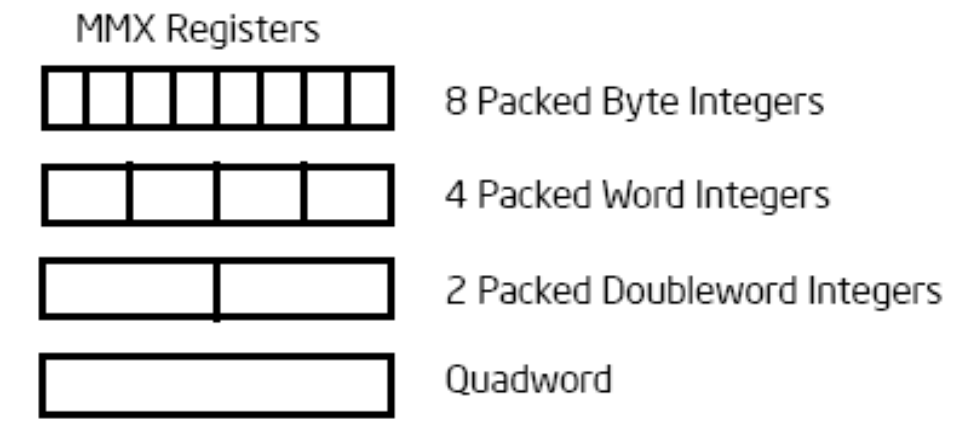
Vantagens

- Uma única instrução para processar n elementos de dados
- Acesso coerente à memória
  Os dados devem estar em posições consecutivas de memória
  Se os dados estiverem alinhados ao parágrafo (múltiplos de 16) o acesso é mais eficiente (SSE)
- Paralelismo declarado explicitamente
  Não existem dependências de dados entre as operações;
  Se o *hardware* possuir as unidades funcionais as operações podem ser realizadas em paralelo

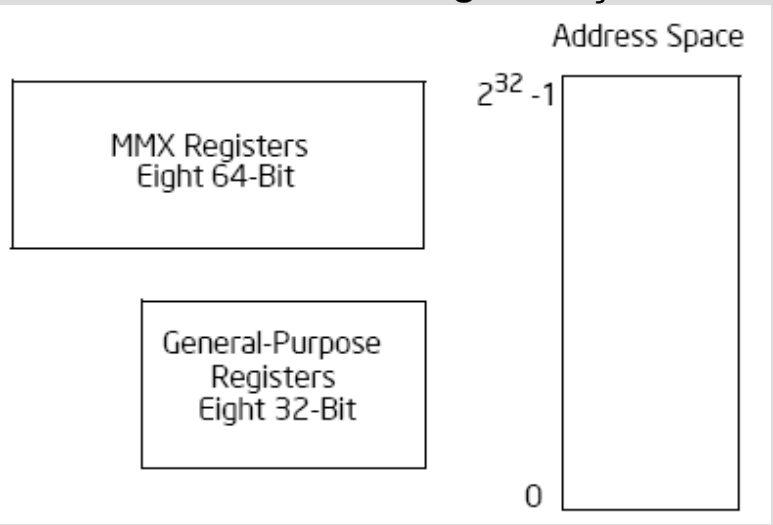
MultiMedia eXtensions (MMX)

- Operações sobre inteiros
- 8 registos de 8 *bytes* (64 *bits*): mmx0 .. mmx7
  Estes registos são os mesmos da FPU

Tipos de Dados



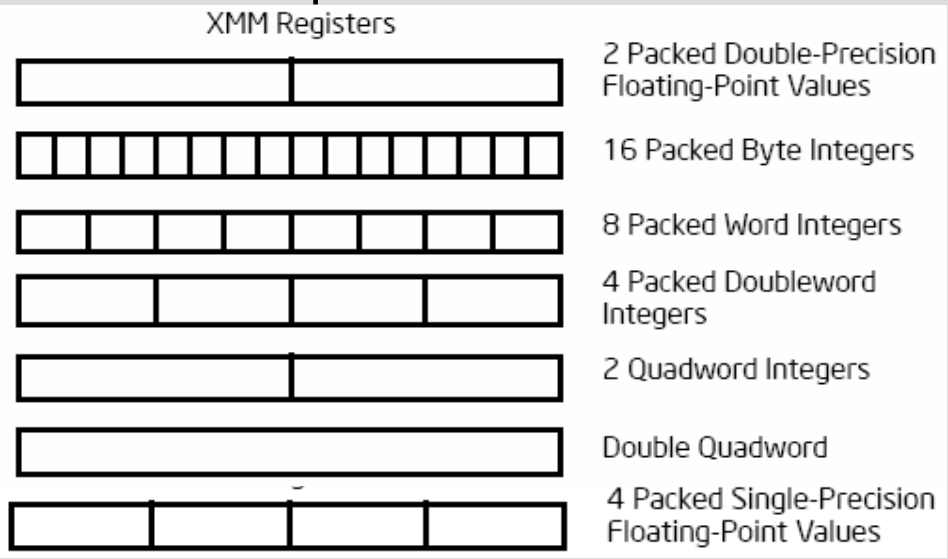
Ambiente de Programação



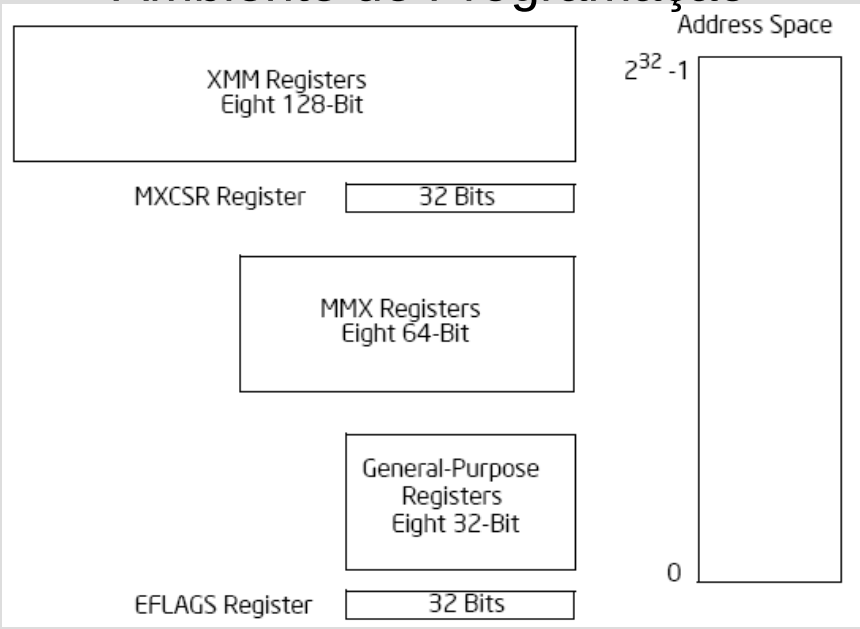
Streaming SIMD Extensions (SSE)

- Operações em vírgula flutuante
- Operações adicionais sobre inteiros
- 8 novos registos de 16 *bytes* (128 *bits*): xmm0 .. xmm7

Tipos de Dados



Ambiente de Programação



Instruções: Transferência de Dados

Instruções	**Operandos** orig, dest	**Obs.**
MOVQ	mm/m64, mm mm, mm/m64	Mover palavra quádrupla (8 bytes) de memória para registo mmx ou de registo mmx para memória (Apenas para inteiros)
MOVDQA MOVDQU	xmm/m128, xmm xmm, xmm/m128	Mover 2 palavras quádruplas (2*8 bytes) Apenas para inteiros A - addr alinhado M16; U – addr não alinhado
MOVAP[S|D] MOVUP[S|D]	xmm/m128, xmm xmm, xmm/m128	Mover 4 FP precisão simples ou 2 FP precisão dupla A – addr alinhado M16 U – addr não alinhado

Instruções: Operações Inteiras

Instruções	**Operandos** orig, dest	**Obs.**
PADD? PSUB? PAND? POR?	mm/m64, mm xmm/m128, xmm	Adição, subtracção, conjunção ou disjunção do tipo de dados indicado Operação realizada sobre o número de elementos determinado pelo registo+tipo de dados Endereços em memória alinhados O resultado não pode ser em memória

```
? = B | W | D | Q
B – byte          W – 2 bytes
D – 4 bytes Q – 8 bytes
```

Instruções: Operações FP

Instruções	**Operandos** orig, dest	**Obs.**
ADDP? SUBP? MULP? DIVP? SQRTP? MAXP? MINP? ANDP? ORP?	xmm/m128, xmm	Operação sobre o tipo de dados indicado Operação realizada sobre o número de elementos determinado pelo tipo de dados (S = 4 ; D = 2) Endereços em memória alinhados O resultado não pode ser em memória

? = S | D
S – precisão simples
D – dupla precisão

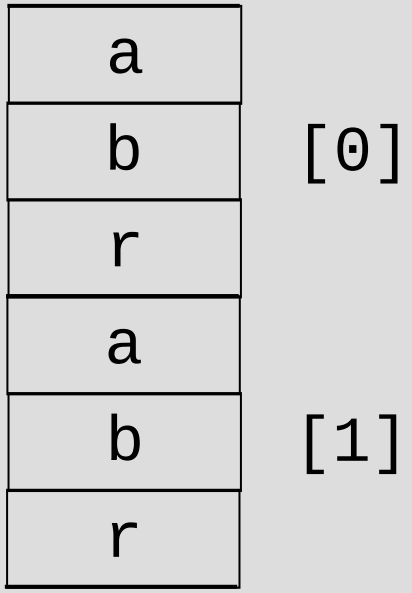
Exemplo MMX

```
int a[100], b[100], r[100];

func (int n, int *a, int *b, int *r) {
  int i;
  for (i=0 ; i<n ; i++)
    r[i] = a[i] + b[i];
}
```

```
func:
  …
  movl 8(%ebp), %edx
  movl 12(%ebp), %eax
  movl 16(%ebp), %ebx
  movl 20(%ebp), %esi
  movl $0, %ecx
ciclo:
  movq (%eax, %ecx, 4), %mm0
  paddd (%ebx, %ecx, 4), %mm0
  movq %mm0, (%esi, %ecx, 4)
  addl $2, %ecx
  cmpl %edx, %ecx
  jle ciclo
  emms
  …
```

SSE - Exemplo

```
struct {
  float a, b, r;
} arr[100];

func (int n) {
  int i;
  for (i=0 ; i<n ; i++)
    arr[i].r = arr[i].a*arr[i].b;
}
```



Vectores de estruturas resultam num *layout* em memória em que os diferentes elementos do mesmo campo não estão em endereços consecutivos!
Usar estrutras de vectores!

SSE - Exemplo

```
float a[100], b[100], r[100];

func (int n) {
  int i;
  for (i=0 ; i<n ; i++)
    r[i] = a[i] * b[i];
}
```

```
func:
  …
  movl 8(%ebp), %edx
  movl $a, %eax
  movl $b, %ebx
  movl $r, %esi
  movl $0, %ecx
ciclo:
  movaps (%eax, %ecx, 4), %xmm0
  mulps (%ebx, %ecx, 4), %xmm0
  movaps %xmm0, (%esi, %ecx, 4)
  addl $4, %ecx
  cmpl %edx, %ecx
  jle ciclo
  …
```

IA32 – SIMD

Tema		
IA32 – SIMD	Intel 64 and IA-32 Software Developer's Manual – Basic Architecture – Volume 1 Intel 64 and IA-32 Software Developer's Manual – Instruction Set Reference – Volume 2	