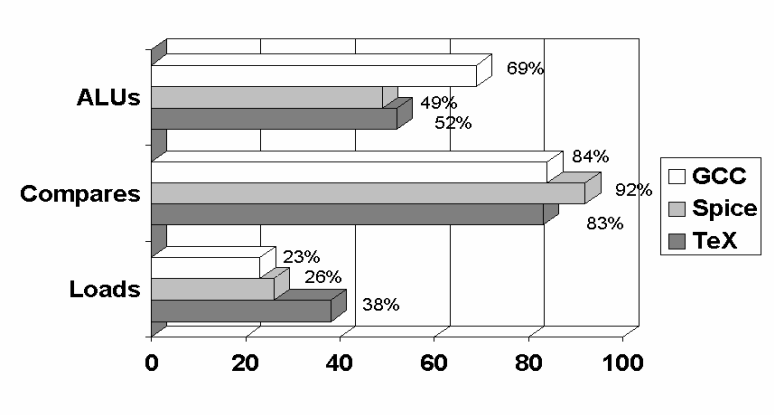
# Aula 05: Arquitetura do Conjunto de Instruções

## Distribuição dos Deslocamentos

- Altamente concentrado em 0 bits
  - 26-27% dos deslocamentos são 0
- 1% dos valores requerem mais que 16 bits
- Maioria dos deslocamentos > 14 bits referem-se a números negativos
- Deslocamentos de 12 bits: 75 %
- Deslocamentos de 16 bits: 99 %

## Imediatos em Operandos são Comuns?




## Qual deve ser o Tamanho de um Imediato?

- Podem ser usados em operações aritméticas, comparações ou moves
- Operações em ALU: 1/2 a 3/4 possuem argumento constante
- Limites dos valores usados:
  - Maioria são valores positivos
  - 50% a 70% cabem em 8 bits
  - 75% a 80% cabem em 16 bits

## Operações do Conjunto de Instrução

- Tipos de operação:
  - aritmética e lógica
  - transferência de dados
  - controle
  - sistema
  - FP
  - decimal
  - string
  - gráfica

## Exemplo do Conjunto de Instruções: 80x86

| | | |
|---|---|---|
| 1 | load | 22% |
| 2 | cond. branch | 20% |
| 3 | compare | 16% |
| 4 | store | 12% |
| 5 | add | 8% |
| 6 | and | 6% |
| 7 | sub | 5% |
| 8 | move r-r | 4% |
| 9 | call | 1% |
| 10 | return | 1% |
| | | 96% |

Velocidade de execução para essas instruções deve ser levado em consideração!

## Controle de Fluxo

- 4 tipos de instruções de controle de fluxo
  - branch para desvios condicionais (81% / 87%)
  - jump para desvios incondicionais (6% / 4%)
  - Chamada e retorno de procedimentos (13% / 11%)
- Maneiras de se especificar endereço destino
  - Relativo ao PC
    - Usa menos bits de instrução
    - Independente da carga do código
  - Qualquer modo de endereçamento definido anteriormente

## Branches

- 75% dos branches são para frente
- Branches mais freqüentes saltam 4-7 instruções de distância
- Comparação precendendo um branch
  - EQ ou NE (86%) vs. GT/LE (7%) ou LT/GE (7%)
  - Maior parte com imediatos (87%), dos quais 50% são a constante “0”

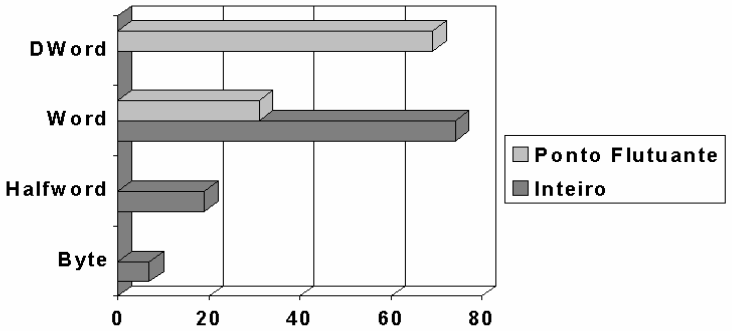
# Mecanismo para Controle de Salto

| | | + | - |
|---|---|---|---|
| *Condition Code* | setado pela ALU | setada p/ instr | s/ controle |
| *Condition register* | testa reg com resultado | simples | 1 instr a mais |
| *Compara e salta* | parte da instr. de branch | 1 instr | muito trabalho / instr |

# Fluxo de Controle Conclusões

- Importância de instruções simples
- Branches são comuns
  - Saltos são próximos (+/- 100 instruções)
  - Relativo a PC com deslocamento > 8 bits
  - Necessidade de relativo a registrador também
    - Como *switch* são implementados?

# Qual o Tamanho do Dado Acessado?



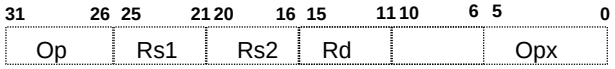

DEC Alpha AXP não possui byte ou halfword para reduzir circuito no caminho crítico. Uma boa escolha?

# A "Typical" RISC

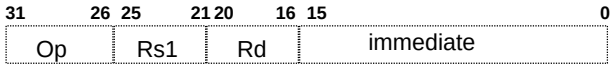
- 32-bit fixed format instruction (few formats)
- 32 32-bit GPR
- 3-address, reg-reg arithmetic instruction
- Single address mode for load/store: base + displacement
  - no indirection
- Simple branch conditions
- Pipelined implementation
- Separate Instruction and Data level-1 caches
- Delayed branch ?

# Exemplo: MIPS

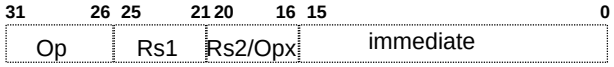
**Register-Register**




**Register-Immediate**

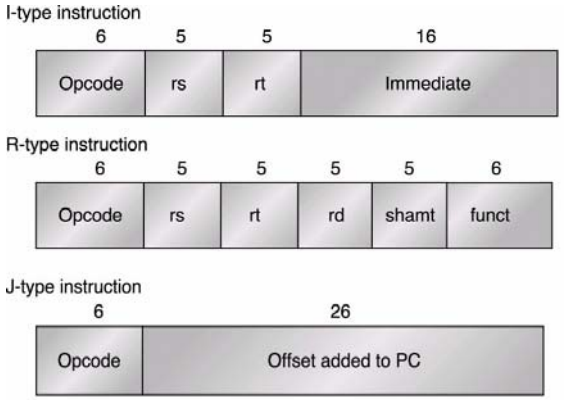



**Branch**




**Jump / Call**


# Examplo: DLX




# Comparison MIPS with 80x86

- How would you expect the x86 and MIPS architectures to compare on the following:
  - CPI on SPEC benchmarks
  - Ease of design and implementation
  - Ease of writing assembly language & compilers
  - Code density
  - Overall performance
- What other advantages/disadvantages are there to the two architectures?

# ISA Mais Popular de Todos os Tempos: O Intel 80x86

- 1971: Intel inventa o microprocessador 4004/8008, 8080 em 1975
- 1975: Gordon Moore percebeu mais uma chance de construir novo ISA antes do ISA não se alterar mais por uma década
  - Contratou pessoal de CS em Oregon
  - Não estavam prontos ainda em 1977 (fizeram o 432 em 80)
  - Iniciaram esforço para fazer processador de 16 bits
- 1978: 8086 com registradores dedicados, espaço de endereçamento segmentado, 16 bits
  - 8088; versão de barramento de 8 bits do 8086
  - Presente no núcleo de memória do DCC (DCC2600)

# ISA Mais Popular de Todos os Tempos: O Intel 80x86

- 1980: IBM seleciona o 8088 como base para o IBM PC
- 1980: Coprocessador de ponto flutuante 8087: adiciona 60 instruções usando um híbrido de pilha/registrador
- 1982: 80286 endereçamento em 24 bits, proteção
- 1985: 80386 endereçamento em 32 bits, registradores de 32 bits GPR, paginação
- 1989: 80486 & Pentium em 1992: *pipelining* e *superescalar*, mais rápido, FP melhorado
  - Pentium com bug no algoritmo de divisão faz Intel perder um trimestre de lucros trocando chips defeituosos

# 80x86 Instruction Frequency

| *Rank* | *Instruction* | *Frequency* |
|---|---|---|
| **1** | **load** | **22%** |
| **2** | **branch** | **20%** |
| **3** | **compare** | **16%** |
| **4** | **store** | **12%** |
| **5** | **add** | **8%** |
| **6** | **and** | **6%** |
| **7** | **sub** | **5%** |
| **8** | **register move** | **4%** |
| **9** | **call** | **1%** |
| **10** | **return** | **1%** |
| **Total** | | **96%** |

# Relative Frequency of Control Instructions

| Operation | SPECint92 | SPECfp92 |
|---|---|---|
| Call/Return | 13% | 11% |
| Jumps | 6% | 4% |
| Branches | 81% | 87% |

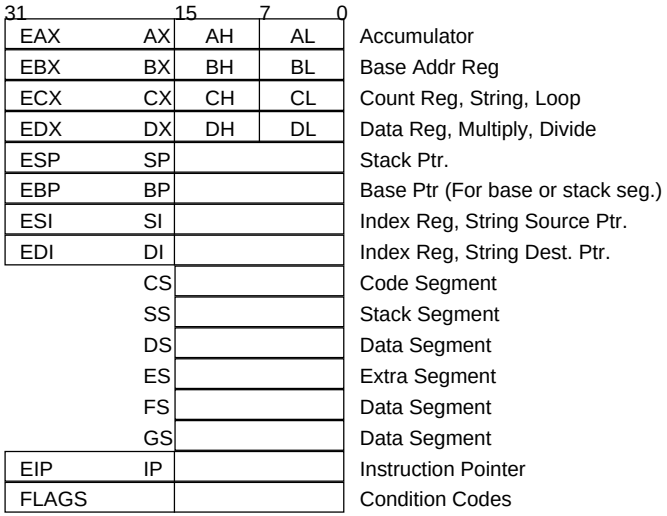
- **Design hardware to handle branches quickly, since these occur most frequently**

# Frequency of Operand Sizes on 32-bit Load-Store Machines

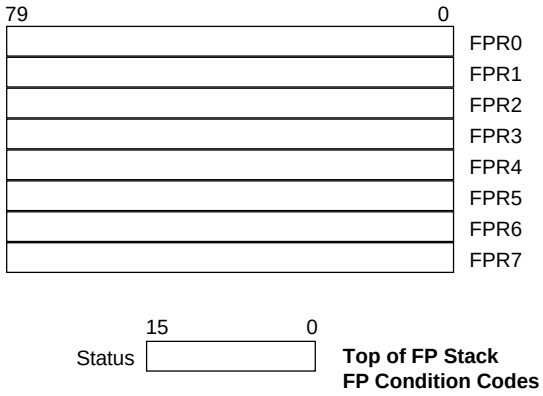
| Size | SPECint92 | SPECfp92 |
|---|---|---|
| 64 bits | 0% | 69% |
| 32 bits | 74% | 31% |
| 16 bits | 19% | 0% |
| 8 bits | 19% | 0% |

- For floating-point want good performance for 64 bit operands.
- For integer operations want good performance for 32 bit operands
- Recent architectures also support 64-bit integers

# Intel 80x86 - Registradores Inteiros




# Intel 80x86 - Registradores de Ponto Flutuante

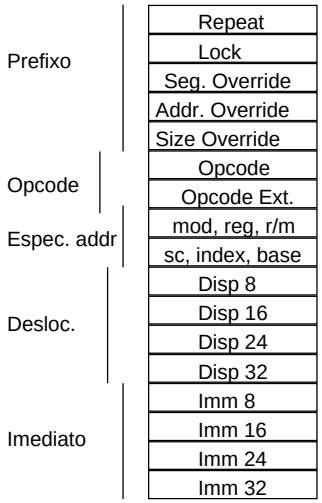



# Utilização dos Registradores de Ponto Flutuante do 80x86

| | NASA 7 | Spice |
|---|---|---|
| Pilha (2o. operando ST(1)) | 0.3% | 2.0% |
| Registr. (2o. operando ST(i), i>1) | 23.3% | 8.3% |
| Memória | 76.3% | 89.7% |

Pilha não é utilizada pelos compiladores do Solaris para execução mais rápida

# Formato das Instruções do 80x86




**(Base reg + $2^{Scale}$ x Index reg)**

# Codificação das Instruções do 80x86: Campos Mod, Reg, R/M

| r w=0 | w=1 | | r/m | mod=0 | | mod=1 | | mod=2 | | mod=3 |
|---|---|---|---|---|---|---|---|---|---|---|
| | 16b | 32b | | 16b | 32b | 16b | 32b | 16b | 32b | |
| 0 | AL | AX | EAX | 0 | addr=BX+SI | =EAX | *same* | *same* | *same* | *same* | *same* |
| 1 | CL | CX | ECX | 1 | addr=BX+DI | =ECX | *addr* | *addr* | *addr* | *addr* | *as* |
| 2 | DL | DX | EDX | 2 | addr=BP+SI | =EDX | *mod=0* | *mod=0* | *mod=0* | *mod=0* | *reg* |
| 3 | BL | BX | EBX | 3 | addr=BP+SI | =EBX | *+d8* | *+d8* | *+d16* | *+d32* | *field* |
| 4 | AH | SP | ESP | 4 | addr=SI | =(sib) | SI+d8 | (sib)+d8 | SI+d8 | (sib)+d32 | " |
| 5 | CH | BP | EBP | 5 | addr=DI | =d32 | DI+d8 | EBP+d8 | DI+d16 | EBP+d32 | " |
| 6 | DH | SI | ESI | 6 | addr=d16 | =ESI | BP+d8 | ESI+d8 | BP+d16 | ESI+d32 | " |
| 7 | BH | DI | EDI | 7 | addr=BX | =EDI | BX+d8 | EDI+d8 | BX+d16 | EDI+d32 | " |

r/m depende de mod e modo da máquina

w do opcode

**Tamanho dos campos: Reg=3 bits, R/M=3 bits, Mod=2 bits**

# Codificação das Instruções do 80x86: Campos Sc/Index/Base

| | *Index* | *Base* |
|---|---|---|
| 0 | EAX | EAX |
| 1 | ECX | ECX |
| 2 | EDX | EDX |
| 3 | EBX | EBX |
| 4 | no index | ESP |
| 5 | EBP | if mod = 0, d32<br>if mod != 0, EBP |
| 6 | ESI | ESI |
| 7 | EDI | EDI |

**Base + Modo Indexado Escalado**

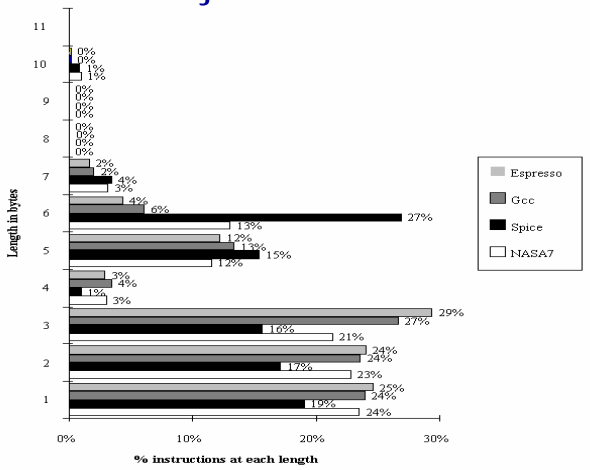
**Usado quando:**
- **mod = 0,1,2**
- **em modo de 32 bits**
- **e r/m = 4**

**2-bit Campo de Escala**
**3-bit Campo da Índice**
**3-bit Campo da Base**

# Utilização dos Modos de Endereçamento para 80x86 em Modo de 32 bits

| *Addressing Mode* | *Gcc* | *Espr.* | *NASA7* | *Spice* | *Avg.* |
|---|---|---|---|---|---|
| Register indirect | 10% | 10% | 6% | 2% | 7% |
| Base + 8-bit disp | 46% | 43% | 32% | 4% | 31% |
| Base + 32-bit disp | 2% | 0% | 24% | 10% | 9% |
| Indexed | 1% | 0% | 1% | 0% | 1% |
| Based indexed + 8b disp | 0% | 0% | 4% | 0% | 1% |
| Based indexed + 32b disp | 0% | 0% | 0% | 0% | 0% |
| Base + Scaled Indexed | 12% | 31% | 9% | 0% | 13% |
| Base + Scaled Index + 8b disp | 2% | 1% | 2% | 0% | 1% |
| Base + Scaled Index + 32b disp | 6% | 2% | 2% | 33% | 11% |
| 32-bit Direct | 19% | 12% | 20% | 51% | 26% |

# Distribuição do Comprimento das Instruções do 80x86



## *Instruction Counts*: 80x86 v. DLX

| *SPEC pgm* | *x86* | *DLX* | *DLX/86* |
|---|---|---|---|
| **gcc** | 3,771,327,742 | 3,892,063,460 | 1.03 |
| **espresso** | 2,216,423,413 | 2,801,294,286 | 1.26 |
| **spice** | 15,257,026,309 | 16,965,928,788 | 1.11 |

## Compilador da Intel vs. Compiladores Você Compra

- 66 MHz Pentium

| | SpecInt92 | SpecFP92 |
|---|---|---|
| Compilador interno da Intel | 64.6 | 59.7 |
| Melhor compilador 486 (Junho 1993) | 57.6 | 39.9 |
| Compilador típico p/ 486 em 1990, quando Intel iniciou o projeto | 41.0 | 32.5 |

- Inteiro: Intel 1.1X mais rápido, FP 1.5X mais rápido

- 486

| | SpecInt92 | SpecFP92 |
|---|---|---|
| Compilador interno da Intel | 35.5 | 17.5 |
| Melhor compilador 486 (Junho 1993) | 32.2 | 16.0 |
| Compilador típico p/ 486 em 1990 , quando Intel iniciou o projeto | 23.0 | 12.8 |

- Inteiro: Intel 1.1X mais rápido, FP 1.1X mais rápido

## Graphics and Multimedia Instruction Set Extensions

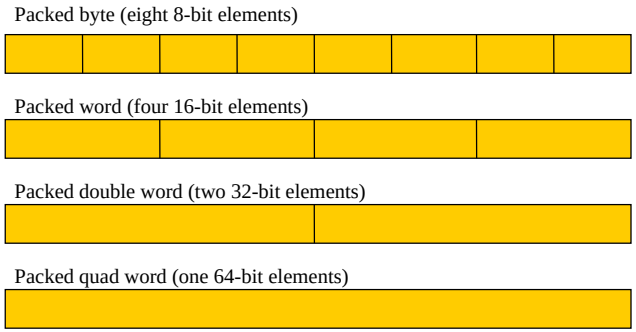
- Several companies have extended their computer's instruction sets to better support graphics and multimedia applications
  - Intel's MMX Technology
  - Intel's Internet Streaming SIMD Extensions
  - AMD's 3DNow! Technology
  - Sun's Visual Instruction Set
  - Motorola's and IBM's AltiVec Technology

## Graphics and Multimedia Instruction Set Extensions

- These extensions improve the performance of
  - Computer-aided design
  - Internet applications
  - Computer visualization
  - Video games
  - Speech recognition

# MMX Data Types

MMX Technology supports operations on the following 64-bit integer data types:




# SIMD Operations

- MMX Technology allows a Single **I**nstruction to work on Multiple pieces of Data (**SIMD**)

| | | | |
|---|---|---|---|
| A3 | A2 | A1 | A0 |
| B3 | B2 | B1 | B0 |
| A3+B3 | A2+B2 | A1+B1 | A0+B0 |

**PADD[W]: Packed add word**

- In the above example, 4 parallel adds are performed on 16-bit elements
- Most MMX instructions only require a single cycle

# Saturating Arithmetic

- Both wrap-around and saturating adds are supported
- With saturating arithmetic, results that overflow/underflow are set to the largest/smallest value

| | | | |
|---|---|---|---|
| a3 | a2 | a1 | FFFFh |
| + | + | + | + |
| b3 | b2 | b1 | 8000h |
| a3+b3 | a2+b2 | a1+b1 | 7FFFh |

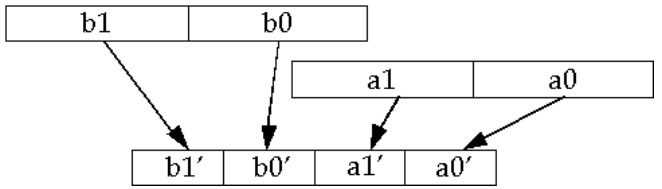
PADD[W]: Packed wrap-around add

| | | | |
|---|---|---|---|
| a3 | a2 | a1 | FFFFh |
| + | + | + | + |
| b3 | b2 | b1 | 8000h |
| a3+b3 | a2+b2 | a1+b1 | FFFFh |

PADDUS[W]: Packed saturating add

# Pack and Unpack Instructions

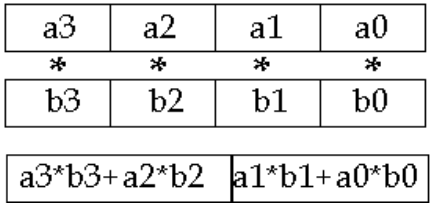
- Pack and unpack instructions provide conversion between standard data types and packed data types




PACKSS[DW]: Pack signed, with saturating, double to packed word

# Multiply-Add Operations

- Many graphics applications require multiply-accumulate operations
  - Vector Dot Products
  - Matrix Multiplies
  - Fast Fourier Transforms (FFTs)
  - Filter implementations

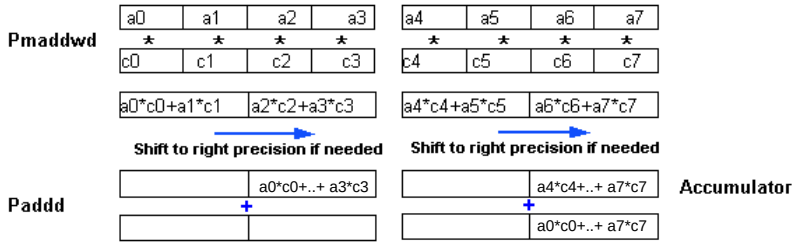



PMADDWD: Packed multiply-add word to double

# Vector Dot Product

- A dot product on an 8-element vector can be performed using 9 MMX instructions
- Without MMX 40 instructions are required

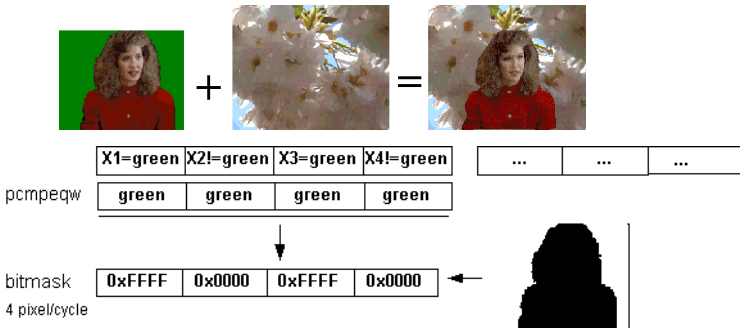
$$x = \sum a(i) * c(i)$$




**Note:** Input data and coefficients are 16-bit precision. If not, first unpack to 16 bit.

# Packed Compare Instructions

- Packed compare instructions allow a bit mask to be set or cleared
- This is useful when images with certain qualities need to be extracted




# MMX Instructions

- MMX Technology adds 57 new instructions to the x86 architecture.
- Some of these instructions include
  - PADD(b, w, d) Packed addition
  - PSUB(b, w, d) Packed subtraction
  - PCMPE(b, w, d) Packed compare equal
  - PMULLw Packed word multiply low
  - PMULHw Packed word multiply high
  - PMADDwd Packed word multiply-add
  - PSRL(w, d, q) Pack shift right logical
  - PACKSS(wb, dw) Pack data
  - PUNPCK(bw, wd, dq) Unpack data
  - PAND, POR, PXOR Packed logical operations

## Performance Comparison

- The following shows the performance of Pentium processors with and without MMX Technology

| **Application** | Without MMX | With MMX | Speedup |
|---|---|---|---|
| Video | 155.52 | 268.70 | 1.72 |
| Image Processing | 159.03 | 743.90 | 4.67 |
| 3D geometry | 161.52 | 166.44 | 1.03 |
| Audio | 149.80 | 318.90 | 2.13 |
| Overall | 156.00 | 255.43 | 1.64 |

## MMX Technology Summary

- MMX technology extends the Intel x86 architecture to improve the performance of multimedia and graphics applications.
- It provides a speedup of 1.5 to 2.0 for certain applications.
- MMX instructions are hand-coded in assembly or implemented as libraries to achieve high performance.
- MMX data types use the x86 floating point registers to avoid adding state to the processor.
  - Makes it easy to handle context switches
  - Makes it hard to perform MMX and floating point instructions at the same time
- Only increase the chip area by about 5%.

## Questions on MMX

- What are the strengths and weaknesses of MMX Technology?
- How could MMX Technology potentially be improved?
- How did the developers of MMX preserve backward compatibility with the x86 architecture?
  - Why was this important?
  - What are the disadvantages of this approach?
- What restrictions/limitations are there on the use of MMX Technology?

## Internet Streaming SIMD Extensions

- Intel's **I**nternet **S**treaming **S**IMD **E**xtensions (**ISSE**)
  - Help improve the performance of video and 3D applications
  - Are designed for streaming data, which is used once and then discarded.
  - 70 new instructions beyond MMX Technology
  - Adds new 128-bit registers
  - Provide the ability to perform parallel **floating point** operations
    - Four parallel operations on 32-bit numbers
    - Reciprocal and reciprocal root instructions - normalization
    - Packed average instruction – Motion compensation
  - Provide data prefetch instructions
  - Make certain applications 1.5 to 2.0 times faster

# Conclusões sobre Intel 80x86

- Projeto do conjunto de instruções do 8086
  - Endereçamento: 16 bits vs. 32 bits
  - Proteção: segmentação vs. paginação
  - Armazenamento temporário: acumulador vs. pilha vs. registradores
- "Algemas" da compatibilidade binária afeta a arquitetura depois de 20 anos
- Não é muito difícil de fazer mais rápido, como Intel tem mostrado
- HP/Intel anunciaram esforço conjunto para projetar novo conjunto de instruções para o ano 2000. Isto significa o fim da arquitetura 80x86?
- "*Beauty is in the eye of the beholder*"

  Com 50M de chips/ano vendidos, é a melhor arquitetura do mundo