Sistemas de Numeração

Os números Naturais são utilizados para resolver problemas de contagem e foram os primeiros a serem utilizados pela humanidade. Já no início da civilização, havia uma certa “correspondência binária” neste processo, pois as tribos primitivas costumavam contar as coisas Como “um, dois e muitos”, ou seja, tudo conjunto ou coleção que tinha mais do que dois elementos era considerado como muitos ou infinito.

Essa herança primitiva permanece até hoje: em francês, por exemplo, a palavra “muitos” se traduz como “très” e este radical está presente em vários idiomas modernos para representar tal número: no próprio francês, três é traduzido como trois, em inglês, three, em Italiano, tre e assim por diante. Com a evolução histórica, social e intelectual da humanidade, evoluiu também o conceito de infinito mas, embora seja matematicamente incorreto, é comum pessoas dizerem que uma quantidade enorme ou incontável é infinita.

Embora a matemática tenha um alto nível de exigência no tocante ao rigor das demonstrações lógicas, por muito tempo o conjunto N foi tido apenas como intuitivo e não demonstrável, tanto que o famoso matemático Leopold Kronecker (1823 – 1891) chegou a afirmar que Deus criou os números Naturais e o homem fez todo o resto. Foi apenas em 1889 que o matemático italiano Giuseppe Peano(1858 – 1932) conseguiu caracterizar os números Naturais de forma axiomática, através dos famosos Axiomas de Peano, Posteriormente, seguiram-se outras demonstrações mais complexas e rigorosas deste conjunto.

Para entendermos os sistemas de numeração, precisamos de dois conceitos importantes no conjunto N: divisibilidade e divisão euclidiana.

Divisibilidade em N

A divisibilidade é a versão multiplicativa da relação de ordem e é definida como: dados dois números naturais a e b, com b diferente de 0, dizemos que b divide a (ou, de forma recíproca, que a é um múltiplo de b) e anotamos b|a se e somente se existe um outro número natural c tal que a = b . c.

Por exemplo: 2|6 pois 6 = 3 . 2; 7|63 pois 63 = 7 . 9, mas 3 não divide 8, pois não existe nenhum número natural que multiplicado por três resulte em 8.

Percebamos, aqui, que o conceito de divisibilidade nada mais é do que achar uma solução para a equação bx = a . Por este motivo, tratamos deste assunto apenas no conjunto dos números Naturais e Inteiros, pois ela poderá ou não ter uma solução; em outros conjuntos, como Q, I ou R, sempre existirá algum x que satisfaça a equação. Se trouxéssemos o último exemplo para Q, teríamos que x=8/3. Notemos com isso, b é sempre menor ou igual a a.

O Algorítimo de Euclides Para a Divisão

Euclides foi um famoso matemático grego que viveu em aproximadamente 300 a.C. Foi ele quem escreveu a obra Os Elementos, que definem os alicerces para a Geometria que vemos nos ensinos Fundamental e Médio e que também é utilizada em várias áreas, como engenharia, design e construção civil.

Fora da Geometria, Euclides descobriu um teorema o qual nos permite dividir qualquer número Natural por outro – e você certamente o conhece desde o início de sua vida escolar. Ele se baseia na ideia de divisibilidade a qual acabamos de ver e tem por objetivos responder a uma simples pergunta: Se a e b são números Naturais e b não divide a, qual será o múltiplo de b que mais se aproxima de a? A resposta a esta pergunta está em um teorema o qual afirma que, se a e b são números Naturais, então existem e são únicos os números Naturais q e r tais que a = q . b + r, sendo que r é menor do que b.

Chamamos a q de quociente e a r de resto. A condição de que o resto seja menor do que o divisor b nos garante que o quociente será único. Caso não fizéssemos essa restrição, a divisão de a por b poderia ter infinitos resultados.

Como vimos no exemplo anterior, 3 não divide 8. Assim, podemos utilizar o algorítimo da divisão euclidiana e concluir que 8 = 2 . 3 + 2, ou seja: 6, 3 . 2, é o múltiplo de 3 que mais se aproxima de 8.

Nesta expressão, perceba que q = 2 e r = 2 e, ainda, que o valor de r é menor do que o divisor 3. Caso não considerássemos a condição r < b, também poderíamos escrever que 8 = 1 . 3 + 5 ou que 8 = 0 . 3 + 8 e, se considerássemos o conjunto dos números inteiros, Z, teríamos mais infinitas possibilidades. Desta forma, garantimos a unicidade do resultado ao considerar que o resto sempre deverá ser menor do que o divisor. Perceba, ainda, que se a for múltiplo de b, o resto será igual a 0 e a divisão será dita exata.

Sistemas de Numeração

Agora que temos toda a base matemática de que precisamos, podemos tratar dos sistemas de numeração em si. Um sistema de numeração nada mais é do que um sistema onde um conjunto de números é representado por numerais de forma consistente.

Atualmente, nós utilizamos um sistema de numeração decimal, isto é, de base 10, que é dito posicional, ou seja, cada algarismo, além de seu valor, possui um peso dado através da posição que ocupa. Desta forma, embora os números 518 e 851 sejam compostos pelos mesmos algarismos, sabemos que eles representam quantidades diferentes.

A princípio, a base do sistema de numeração pode ser um número qualquer. Os babilônios antigos usavam um sistema de base 60 cujos vestígios encontramos ainda hoje na medição de ângulos e nos relógios, por exemplo. Quando a base do sistema é menor ou igual a 10, utilizamos os algarismos indo arábicos para representar os numerais; quando é maior, devemos utilizar outros símbolos, geralmente letras do alfabeto latino.

Graças a um teorema matemático, sabemos que cada número Natural possui uma representação única em uma base qualquer. Este teorema pode ser enunciado como: “Seja b um número Natural com b ? 2. Então, para todo a Natural diferente de 0, existem e são únicos números Naturais c0, c1, c2, …, cn tais que a = cnbn+ cn-1bn-1+ … + c1b + c0 para os índices de c menores do que b e o último diferente de 0“.

Esta expressão, acima, chama-se de “expansão de a na base b” e também pode ser descrita como a = (cncn-1...c1c0)b.

Com isso, a expansão do número 524 na base decimal é 5 . 102+ 2 . 10 + 4. Perceba que o número que multiplica cada algarismo é a base numérica elevada a um expoente que corresponde à posição do numeral menos 1 e o último algarismo está, na verdade, sendo multiplicado pela base elevada a 0.

Convertendo de Uma Base Qualquer Para a Base 10

Com o que vimos anteriormente, para convertermos um número que está em uma base qualquer para a base decimal simplesmente devemos escrever e calcular a expansão deste número na base dada.

Por exemplo, vamos calcular quanto vale (52024)7 na base 10:

(52024)7= 5. 74+ 2 . 73+ 0 . 72+ 2 . 7 + 4 = (12709)10

Portanto, a representação de 52024 na base 7 é 12709 na base 10.

Convertendo da base 10 para uma base qualquer

Para convertermos um número que esteja na base decimal para outra base b, devemos encontrar uma expressão da forma a = cnbn+ cn-1bn-1+ … + c1b + c0 para o mesmo.

Como exemplo, vamos escrever o número 139 na base 3. Para isso, precisamos encontrar para ele uma expressão do tipo 139 = an 3n+ an-13n-1+ … + a13+ a0.

Ao utilizarmos a propriedade distributiva da multiplicação, temos que

139 = 3 . (an 3n-1+ ...+ a1) + a0

Desta forma, concluímos que a0 é, na verdade, o resto da divisão de 139 por 3. Se realizarmos tal

divisão, teremos que 139 = 46 . 3 + 1, ou seja, a0 é igual a 1. Com isso, 46 . 3 = 139 - 1. Mas nós sabemos que 139 = 3 . (an 3n-1+ ...+ a1) + a0= 3 . (an 3n-1+ ...+ a1) + 1 . Temos, então, que 139 = 3 . (an 3n-1+ ...+ a1) , ou seja, eliminamos o termo independente. Vamos repetir essa operação até eliminarmos todos os a's e, então, teremos que:

139 = 46 . 3 + 1

46 = 15 . 3 + 1

15 = 5 . 3 + 0

5 = 1 . 3 + 2

Agora, vamos fazer uma série de substituições:

139 = 46 . 3 + 1 = (15 . 3 + 1) . 3 + 1 = 15 . 32 + 1 . 3 + 1 = 5 . 33 + 1 . 3 + 1 = (1 . 3 + 2) . 33 + 1 . 3 + 1 = 1 . 34 + 2 . 33 + 0 . 32 . 1 . 3 + 1

Assim, concluímos que a expansão de 139 na base 3 é (12011)3. Note quem apesar de esta ser a explicação matematicamente correta, muitos cursos técnicos preferem simplificar o processo simplesmente dizendo que, para converter um número na base decimal para outra base devemos realizar sucessivas divisões euclidianas do número pelo algarismo que representa a base até que o divisor seja igual a 0 e pegarmos os restos de baixo para cima. Este procedimento, muito mais simples, está correto e é justificado pelo exemplo anterior.

Sistemas de Numeração Utilizados na Informática

Os sistemas de numeração mais utilizados na informática são:

Base 2: também conhecido como sistema binário. É um sistema posicional composto pelos numerais 0 e 1 e, além da Informática, é utilizado na Eletrônica Digital na implementação de circuitos de portas lógicas. Uma de suas primeiras aplicações na informática surgiu quando da utilização de cartões perfurados para representar informações e programas.

Base 8: o sistema octal também é um sistema posicional e foi utilizado na Informática como alternativa ao sistema binário. É composto pelos numerais 0, 1, 2, 3, 4, 5, 6 e 7.

Base 16: o sistema hexadecimal é, talvez, um dos mais conhecidos da atualidade. É composto de 16 algarismos, representados por 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F. Trabalha-se com ele como qualquer outro sistema, mas deve-se prestar atenção ao valor dos caracteres alfabético na hora de fazer operações e conversões.

É atualmente a maior alternativa ao sistema binário por ser extremamente compacto e é utilizado para representar portas, interrupções e endereços de memória, além de cores no desenvolvimento web, em substituição ao sistema RGB. Para representar as cores, é utilizada uma notação de seis dígitos, onde cada dupla, da esquerda para a direita, representa o valor da intensidade do vermelho, do verde e do azul, respectivamente, variando de 00 até FF, que representa o valor decimal 255.

Assim, as cores variam de 000000, que representa o preto, até FFFFFF, que corresponde ao branco. Os tons de cinza são representados por valores iguais nas três posições, como por exemplo 666666, DEDEDE ou CCCCCC. Quanto mais próximo de FFFFFF, mais clara é a tonalidade de cinza.

Ao todo, essa notação hexadecimal permite a representação de mais de 16 milhões e meio de cores. Alguns programas de desenho vetorial e de tratamento de imagem incluem, ainda, uma quarta dupla de valores na notação para representar o nível de transparência da cor selecionada.

Graças à base hexadecimal, as rotinas de tratamento de imagem foram em muito facilitadas, Por exemplo: para fazer o efeito de negativo em uma foto, isto é, inverter suas cores, basta subtrair de FF cada valor da tripla que representa a cor de cada pixel.

Base 62: Talvez você nunca tenha ouvido falar deste sistema, mas acredite: você já o utilizou. O

sistema de base 62 está se tornando cada vez mais popular porque seus 62 algarismos são representados pelos numerais de 0 a 9 e pelas letras de A a Z e de a a z. Uma de suas aplicações mais recorrentes está nos famosos encurtadores de URL: o código gerado pelo encurtador nada mais é do que a conversão de um número decimal – geralmente o número de identificação único para cada URL no banco de dados – nesta base. Assim, quanto mais caracteres um encurtador de URLs utilizar para gerar sua URL curta, mais URLs ele terá cadastradas.

Base 64: É um sistema numérico utilizado para codificação de dados binários que precisam ser armazenados e transferidos em meios que foram desenhados originalmente para lidar com dados textuais. É composto pelos algarismos de 0 a 9, pelas letras de A a Z e de a a z e pelos símbolos / e +. O caractere = é utilizado como sufixo especial.

O Sistema de Numeração Binário

Apesar das tentativas de utilizar-se outros sistemas de numeração mais compactos para substituir o sistema binário, é este que forma a “linguagem” dos computadores. Tudo, desde números, cores, palavras, textos e imagens é tratado pelo sistema operacional nesta base de numeração. No exemplo mais simples, se você fizer uma soma na calculadora de seu sistema operacional, ele irá converter os valores para sua representação base 2, realizará a operação e converterá, novamente, o resultado para a base decimal.

Para convertermos um número em base 2 para a base 10, basta fazermos o que já foi explicado, com a vantagem de que, como os algarismos são apenas 0 e 1, poderemos ter a oportunidade de cancelar várias potências.

Por exemplo: vamos converter o número binário 1010 para a base decimal. Escrevendo a expansão do número na base 2, temos:

1010 = 1 . 23 + 0. 22 + 1 . 2 + 0 = 23 + 2 = 8 + 2 = 10

Já o procedimento contrário pode ser feito pelo algoritmo explicado anteriormente ou, de maneira mais simples, pelo “truque” de realizar divisões sucessivas por 2 e pegar os restos de baixo para cima:

10 = 5 . 2 + 0

5 = 2 . 2 + 1

2 = 1 .2 + 0

1 = 0 . 2 + 1

Desta forma, 10 em base 2 se escreve como 1010.

Agora, o que devemos fazer se desejarmos converter um número que esteja na base 2 para a forma octal ou hexadecimal? Evidentemente, uma maneira seria converter o número em base 2para a base 10 e fazer a outra conversão, mas isso demoraria muito tempo. Assim, utilizamos um dispositivo mais prático: como a base octal possui 23 dígitos, basta agruparmos os algarismos do número em base 2 em grupos de três, completando com zeros, caso necessário, e convertendo estes grupos para sua forma decimal.

O mesmo procedimento pode ser utilizado para a base hexadecimal, com a diferença de que devemos agrupá-los em grupos de 4 e cuidar o valor correspondente às letras. Por exemplo, para convertermos 11101111 para a base octal, temos que separá-lo em grupos de três algarismos: 011 101 111. Agora, basta realizar a conversão e termos 3 5 7. 357 é a representação do número em base 8.

A adição de dois valores na base dois é realizada por transporte, da mesma forma do que a adição na base 10. Escrevemos os números um abaixo do outro e somamos por colunas da direita para a esquerda. Notemos que há quatro possibilidades:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

Perceba que na última possibilidade, o resultado é maior do que a quantidade máxima de algarismos que podem ser armazenados em cada posição e, neste caso, há o transporte, ou popular “vai um”.

A subtração ocorre de forma análoga, apenas com a diferença de que:

0 - 0 = 0

0 - 1 = 1

1 - 0 = 1

1 - 1 = 0

No segundo caso, o resultado será 1, mas ocorrerá um transporte para a próxima coluna, que será acumulado no subtraendo.

Logo no começo de sua utilização na informática, surgiram algumas questões conceituais sobre o sistema binário. Uma das principais era a de como fazer um computador representar números negativos. Para tal, reservamos o bit mais significativo de um byte para o sinal, ou seja, o primeiro. Ele será 0 caso o número positivo e 1 caso negativo e usamos o complemento para 1 ou o complemento para 2.

O primeiro consiste em simplesmente inverter zeros e uns: o que é zero vira um e vice-versa. O sinal é dado pelo bit mais significativo. Por exemplo: 3 em base 2 é 11. Representamos-lhe por 011. Para representarmos -3, simplesmente invertemos os bits e temos 100. Já no complemento para 2, invertemos os bits 0 e 1 e somamos 1 ao resultado. A vantagem de utilizar-se o complemento para dois é que ele possui um zero único e preserva as regras de adição e de subtração.

Outro ponto polêmico diz respeito à representação de números com vírgula. A questão apenas foi resolvida com o lançamento da Norma IEEE 754, publicada originalmente em 1985 e revisada em 2008. A norma original dita que os números reais conhecidos por números de ponto flutuante no meio informático – devem ser armazenados em sequências de 32 bits (a revisão de 2008 adicionou a representação de 64-bit para precisão dupla), onde o bit mais significativo é reservado ao sinal – 0se positivo e 1 se negativo -; os oito bits seguintes são reservados ao expoente e os 23 bits restantes são reservados à parte fracionária. O expoente em questão corresponde a 127 mais o expoente de base 2 do número em questão.

O Sistema Binário e a Eletrônica Digital

Enquanto a eletrônica digital trabalha apenas com dois valores possíveis de amplitude, a analógica admite que a amplitude de um sinal pode assumir qualquer valor entre um mínimo e um máximo dados através do tempo. O valor 1 representa o valor mais alto de amplitude, logicamente assumido como verdadeiro e o 0, o mais baixo, correspondente ao lógico falso. É um grande equívoco, portanto, acreditar que o 0 representa a ausência completa de energia Complemente o 1 a presença de energia.

Os circuitos integrados digitais mais utilizados atualmente podem ser de dois tipos: TTL e CMOS. Os circuitos TTL (Transistor-Transistor Logic) são comercializadas em duas séries de modelos: aqueles que começam com 54 são destinados ao uso militar e os que começam com 74 à utilização comercial. Nesta família de circuitos lógicos, o sinal de entrada pode variar em uma amplitude que vai de 0V a 5V.

Quando o sinal está entre 0V e 0,8V, o circuito o interpreta como sendo o nível mais baixo de amplitude, isto é, o 0 binário. Já quando o sinal recebido está entre 2V e 5V, este é interpretado como sendo o nível alto 1. As tensões entre 0,8V e 2V não são reconhecidas pelo circuito e deve-se,

portanto, evitar seu uso em circuitos digitais. Já na família CMOS (Complementary Metal Oxid Semiconductor), tensões que estejam entre 0V e 1,5V são interpretadas como o baixo nível de energia, ou o Falso lógico. Já as tensões de 3,5V a 5V são reconhecidas como o nível alto ou 1. As tensões intermediárias não são reconhecidas.

Assim, cai por terra o mito de que o 0 binário representa a ausência completa de energia: o zero, na verdade, representa o nível lógico correspondente ao falso. Isso não significa, necessariamente, que haja ausência de energia.

Portas lógicas são circuitos digitais que recebem uma ou mais tensões de entrada e devolvem apenas uma tensão de saída. Através da associação dessas portas, podemos construir circuitos capazes de realizar operações aritméticas simples, utilizando como base a álgebra booleana. Essas portas lógicas são constituídas por transistores que atuam como chaves ao receber um sinal elétrico, permitindo ou não sua passagem.

Hoje, porém, existem circuitos integrados que já desempenham este papel. É desnecessário dizer que elas também estão presentes em processadores, em escala muito reduzida. As principais portas lógicas disponíveis hoje, com exemplos entre parêntesis, são:

NOT (TTL 7404) – também conhecida como inversora. Caso receba um nível lógico 1, devolverá 0 e caso receba um nível lógico 0, devolverá 1;

AND (TTL 7421 / CMOS 4081) – Possui duas entradas e uma saída. Retorna 1 apenas se receber o nível lógico 1 em ambas as entradas;

OR (TTL 7432 / CMOS 4075) – Também possui duas entradas e uma saída, mas devolve 1 se pelo menos uma das entradas informar o nível lógico 1.

NAND (TTL 7400 / CMOS 4011) – Equivale a uma porta AND seguida de uma porta NOT. Retorna 0 se receber o nível lógico 1 em ambas as entradas e 1 nos demais casos;

NOR (TTL 7402 / CMOS 4001) – Equivale à porta OR seguida de NOT. Retorna 1 apenas se as duas entradas informarem o nível lógico 0;

XOR (TTL 7486 / CMOS 4070) – A porta Ou Exclusivo retornará o nível lógico 1 se e somente se apenas uma das entradas for igual a 1 e a outra igual a 0.

Embora seja possível combinar livremente as portas lógicas, as empresas buscam economizar recursos quando de sua utilização em projetos eletrônicos calculando circuitos equivalentes, isto é, circuitos que produzirão o resultado esperado com um menor número de portas. Para isso, os engenheiros utilizam o Teorema de Morgan ou o Mapa de Karnaugh.

Os sistemas de numeração transcenderam o âmbito matemático e são utilizados de várias formas na Informática e na Eletrônica.

Aritmética Computacional

As palavras de um computador são compostas por bits e podem representar números armazenados na memória. Estes números podem ter diferentes significados, como inteiros ou reais, serem positivos ou negativos. A manipulação dos números inclui operações de soma, subtração, multiplicação e divisão.

O objetivo deste texto é mostrar como o hardware implementa a representação dos números, os algoritmos adequados para operações aritméticas e sua implicação no conjunto de instruções da máquina.

Números com Sinal e Números sem Sinal

Os números podem ser representados em qualquer base. Porém, a base 2 é a mais adequada para

os computadores porque tratam com somente dois valores: 0 e 1. Estes valores são implementados facilmente através de circuitos elétricos.

Da aritmética temos que, em qualquer base, o valor do i-ésimo dígito d de um número é dado por: d x basei, onde i começa em 0 e cresce da direita para a esquerda, de acordo com a posição ocupada pelo dígito. Por exemplo, o número 1011 na base dois é igual a:

(1x23) + (0x22) + (1x21) + (1x20) = (1x8) + (0x4) + (1x2) + (1x0) = 8 + 0 + 2 + 1 = 11

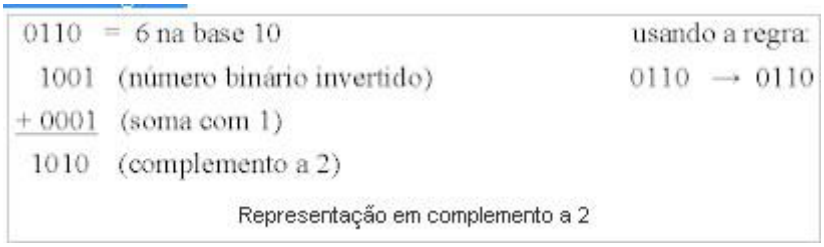
Portanto, os bits são numerados como 0,1,2,3,... da direita para a esquerda em uma palavra. Utilizamos a expressão bit menos significativo para designar o bit 0, e a expressão bit mais significativo para designar o bit de mais alta ordem, como por exemplo o bit 31 numa palavra de 32 bits.

Como o tamanho de uma palavra manipulada por um computador tem tamanho limitado, os números que podem ser representados também tem tamanho limitado. Se o tamanho de uma palavra é igual a n bits, o maior número possível de ser representado é igual a 2n. Se ao realizarmos operações sobre os números, elas gerarem resultados que não podem ser representados na quantidade de bits reservados ocorre o que denominados de overflow (números muito grandes) ou underflow (números muito pequenos). Tanto o overflow quanto o underflow geram exceções e são tratados pelo sistema operacional.

Os computadores manipulam tanto números positivos quanto números negativos, que são representados em complemento a 2. Nesta convenção os números que possuem zeros à esquerda são considerados positivos e os números com uns à esquerda são considerados negativos. O complemento a 2 é obtido invertendo-se o número binário e depois somando 1 a este valor. Porém, uma regra simples para transformar um número binário em sua representação em complemento a 2 é a seguinte:

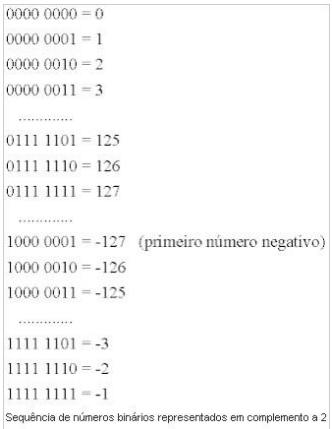
1) copie da direita para a esquerda todos os bits até encontrar o primeiro bit 1 inclusive e
2) inverta todos os demais bits.

A figura abaixo ilustra um exemplo da obtenção de representação em complemento a 2 de um número binário com 4 dígitos.



Representação em complemento a 2

A respresentação em complento a 2 tem a vantagem de representar números negativos sempre com o bit 1 em sua posição mais significativa. Assim, o hardware só precisa testar este bit para verificar se o número é positivo ou negativo. Este bit é conhecido como bit de sinal. Na figura abaixo está representada uma sequência de números binários (8 dígitos) representados em complemento 2.



Sequência de números binários representados em complemento a 2

Conversão entre Diferentes Bases

As bases octal e hexadecimal também são muito úteis em computação. A base octal é representada com 8 dígitos que variam entre 0 e 7. A base hexadecimal é composta por dígitos e letras da seguinte forma: 0 a 9 e as letras a, b, c, d, e e f. Na figura abaixo ilustramos como realizar a mudaça da base binária para as demais bases de forma simples. Lembre-se de que precisamos de 3 ou 4 dígitos binários para a mudança de base octal e hexadecimal, respectivamente.

Binário	Octal	Hexadecimal
9 8 7 6 5 4 3 2 1 0	1 000 111 101	10 0011 1101
1 0 0 0 1 1 1 1 0 1 = 573	1 0 7 5 = 1075	2 3 d = 23d

Mudança de base

Binário	Octal	Hexadecimal
9 8 7 6 5 4 3 2 1 0	1 000 111 101	10 0011 1101
1 0 0 0 1 1 1 1 0 1 = 573	1 0 7 5 = 1075	2 3 d = 23d

Mudança de base

Adição e Subtração

Numa soma os bits são somados um a um da direita para a esquerda, com os carries sendo passados para o próximo bit à esquerda. A operação de subtração usa a adição. O subtraendo é simplesmente negado antes de ser somado ao minuendo. Lembre-se que a máquina trata com números representados em complemento a 2. A figura abaixo mostra as operações de soma (6+7) e subtração (7-6) bit a bit entre dois números representados com 4 dígitos binários.

Representação binária	Soma	Subtração
7 = 0 1 1 1	1 1 (vai um)	1 1 1 (vai um)
6 = 0 1 1 0	0 1 1 1	0 1 1 1
13 = 1 1 0 1 (soma)	+ 0 1 1 0	+ 1 0 1 0 (complemento a 2)
1 = 0 0 0 1 (subtração)	1 1 0 1	0 0 0 1

Operações de soma e subtração (complemento a 2) com representção binária

Como citado anteriormente, tanto a soma como a subtração podem gerar overflow ou underflow, se o resultado obtido não puder ser representado pela quantidade de bits que formam uma palavra. Se somarmos ou subtrairmos dois números com sinais contrários, nunca ocorrerá overflow ou underflow. Isto porque operandos com sinais contrários nunca podem ser maior do que qualquer dos operandos.

O overflow ocorre quando somamos dois operandos positivos e obtemos um resultado negativo, ou vice-versa. Isto significa que utilizamos o bit de sinal, gerando um carry, para armazenar um valor pertencente ao resultado da operação. Raciocínio semelhante é realizado para detectar a ocorrência do underflow numa subtração. Neste caso, o bit de sinal também é usado para armazenar um valor pertencente ao resultado da operação.

Os projetistas de um sistema devem decidir onde tratar a ocorrência de overflow ou de underflow em operações aritméticas. Elas podem ser tratadas tanto por hardware quanto por software. Pode existir a detecção por hardware que gera uma exceção, e que depois é tratada por software.

Operações Lógicas

Os computadores manipulam palavras, mas é muito útil, também, manipular campos de bits dentro de uma palavra ou mesmo bits individuais. O exame de caracteres individuais (8 bits) dentro de uma palavra é um bom exemplo dessa necessidade. Assim, as arquiteturas de conjuntos de intruções incluem instruções para manipulação de bits.

Um dos tipos de instrução utilizados são as de deslocamento de bits. As instruções podem deslocar bits tanto à direita quanto à esquerda. Todos os bits são movidos para o lado determinado e os bits

que ficam vazios são preenchidos com zeros. Outras instruções lógicas muito úteis que são implementadas na unidade lógica e aritmética de um processador são as operações NOT, AND, OR e XOR. A figura abaixo mostra as operações lógicas, bit a bit, de deslocamento à direita, à esquerda, NOT, AND, OR e XOR.

Desl. à direita	Desl. à esquerda	NOT	AND	OR	XOR
1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1
0 1 1 0	1 0 1 0	0 0 1 0	0 1 0 1	0 1 0 1	0 1 0 1
			0 1 0 1	1 1 0 1	1 0 0 0

Operações lógicas

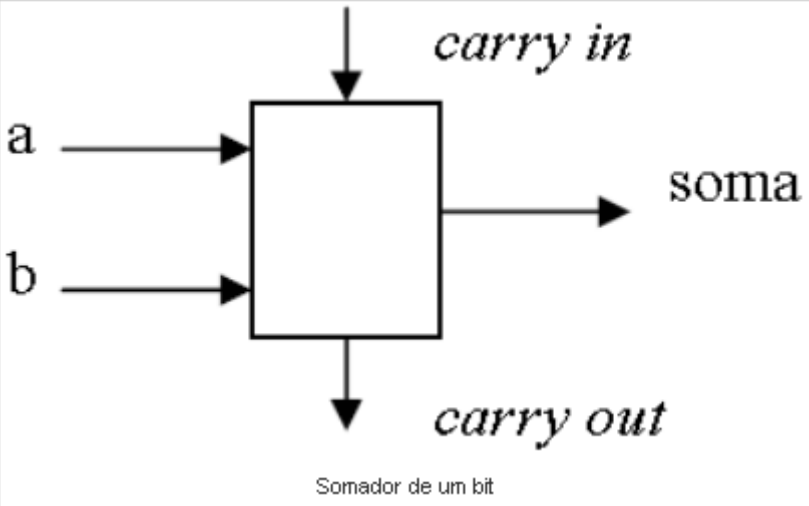
Construção de uma Unidade Lógica Aritmética

A unidade lógica aritmética (ALU – Arithmetic Logic Unit) é o dispositivo que realiza as operações lógicas e aritméticas, definidas pelo conjunto de instruções, dentro do processador.

A ALU é construída basicamente por quatro blocos básicos de hardware: portas AND, portas OR, NOT (inversores) e multiplexadores.

As implementações de operações lógicas são as mais simples de serem realizadas, pois elas são mapeadas diretamente com componentes do hardware.

A próxima função a ser incluída é a adição. Supondo que temos apenas um bit para ser somado, necessitamos de um circuito com duas entradas para os operandos, uma saída para a soma resultante, uma entrada relativa ao carry in e uma saída para o carry out. A figura abaixo mostra este somador.



Somador de um bit

Podemos especificar as saídas soma e carry out através de equações lógicas, que podem ser implementadas a partir dos blocos de hardware mencionados anteriormente. A equação lógica para gerar o bit carry out é dada por:

CarryOut = (b.CarryIn) + (a.CarryIn) + (a.b)

E, a equação lógica para gerar o bit soma é dada por:

Soma = (a . b . CarryIn) + (a . b . CarryIn) + (a . b . CarryIn) + (a . b . CarryIn)

Para completar o projeto de uma ALU de n bits podemos conectar n somadores de um bit. Os carry outs gerados pelos bits menos significativos da operação podem ser propagados por toda a extensão do somador, gerando um carry out no bit mais significativo do resultado da operação. Este somador é denominado somador de carry propagado.

A operação de subtração pode ser realizada somando-se o minuendo com a negação do subtraendo. Este efeito é realizado acrescentando uma entrada complementada de b ao somador e ativando o carry in do bit menos significativo para um. O somador então calcula a + b + 1. Ao escolhermos a versão invertida de b obtemos:

a + b + 1 = a + (b + 1) = a + (-b) = a - b

A simplicidade do projeto do hardware de um somador para números de complemento a 2 demonstra porque esta representação tornou-se um padrão para operações aritméticas inteiras em computadores.

O problema com o somador de carry propagado está relacionado a velocidade de propagação do carry, que é realizada sequencialmente. Num projeto de hardware a velocidade é um fator crítico. Para solucionar este problema existem diversos esquemas para antecipar o carry. Porém, nestes esquemas são utilizadas mais portas lógicas, o que provoca um aumento no custo.

Um dos esquema para antecipar o carry é denominado carry lookahead. Os somadores que utilizam o esquema de carry lookahead baseiam sua implementação em vários níveis de abstração. Utilizando a abreviação ci para representar o i-ésimo bit de carry, podemos escrever a equação do carry como:

ci = (bi . ci) + (ai . ci) + (ai . bi) = (ai . bi) + (ai + bi) . ci

Os termos (ai . bi) e (ai + bi) são tradicionamente chamados de gerador (gi) e propagador(pi), respectivamente. Usando estas relações para definir ci + 1, obtemos:

ci + 1 = gi + pi . ci

Qualquer equação lógica pode ser implementada com uma lógica de dois níveis. Mesmo esta formulação mais simplificada pode gerar equações muito grandes e, portanto levar a circuitos lógicos relativamente grandes e caros, dependendo do número de bits a serem somados.

Multiplicação

Para realizar a multiplicação são necessários dois operandos, um multiplicando e um multiplicador, para gerar um operando produto. O algoritmo da multiplicação diz que os dígitos do multiplicando devem ser multiplicados pelos dígitos do multiplicador um de cada vez, da direita para a esquerda, deslocando os produtos intermedários um dígito à esquerda em relação ao imediatamente anterior.

Uma observação importante é que o número de bits do produto final (n+m) é maior do que o número de bits do multiplicando (n) ou do multiplicador (m). Além disso, a multiplicação também precisa tratar a ocorrência de overflow.

Considerando os dígitos binários 0 e 1, temos apenas duas possibilidades de escolha, a cada passo da multiplicação:

1. Coloque uma cópia do multiplicando (multiplicando x 1) no lugar apropriado, se o dígito do multiplicador for igual a 1, ou

2. Coloque 0 (multiplicando x 0) no lugar apropriado, se o dígito do multiplicador for igual a 0.

Assim, é necessário desenvolver um algoritmo em hardware que seja eficiente para realizar a multiplicação.

Um método elegante de multiplicar números com sinal recebeu o nome de algoritmo de Booth. Ele foi elaborado a partir da constatação de que com a capacidade de somar e de subtrair números existem várias maneiras de se calcular um produto. Por exemplo, podemos substituir um string de uns no multiplicador por uma subtração quando encontramos o primeiro 1, e por uma soma ao encontrarmos o último 1 do string.

Booth buscou atingir maior velocidade de processamento utilizando operações de deslocamento, que ainda hoje são operações mais rápidas do que operações de soma. Baseado nesta observação, se desejarmos maior velocidade na multiplicação de números inteiros por uma potência de 2, basta que utilizemos operações de deslocamento indicando a quantidade de deslocamentos igual ao expoente.

A grande vantagem do algoritmo de Booth é tratar com facilidade os números com sinal. O raciocínio de Booth foi classificar os grupos de bits como início, meio e fim de um string de uns. Naturalmente um string de zeros não precisa ser considerado.

Algoritmo da Multiplicação

Este algoritmo precisa apenas de dois passos principais: o teste do produto e o seu deslocamento; pois os registradores Produto e Multiplicador podem ser combinados em um só. O algoritmo começa com o Multiplicador na metade à direta do registrador Produto, e 0 na metade à esquerda.

1. Testa se Produto é igual a 0 ou 1.

2. Produto = 0, passa ao item 4.

3. Produto = 1, soma o Multiplicando à metade esquerda do Produto e coloca o resultado na metade à esquerda do resgistrador Produto.

4. Desloca o registrador Produto 1 bit à direita.

5. Verifica se foram realizadas todas as repetições necessárias de acordo com o tamanho da palavra, se não volta ao item 1.

6. Fim.

Divisão

A divisão é a operação recíproca da multiplicação. Dentre as operações aritméticas é a que aparece menos frequentemente nos códigos dos programas.

No algoritmo da divisão são utilizados dois operandos, o dividendo e o divisor, e produzidos dois resultados o quociente e o resto. A relação entre os componentes da divisão pode ser expressa da seguinte forma:

Dividendo = quociente x divisor + resto,

Onde o resto é sempre menor que o divisor.

Às vezes, os programas usam a divisão simplesmente para obter o resto, ignorando o quociente. Além disso, é necessário que seja detectada a divisão por zero, que é matematicamente inválida.

Algoritmo da Divisão

Da mesma forma que foram combinados registradores na multiplicação, também na divisão são combinados dois registradores, o Resto e o Quociente. O algoritmo começa com o Resto na metade à esquerda do registrador Resto, e o Quociente na metade à direita.

1. Desloca o registrador Resto 1 bit à esquerda.

2. Subtrai o registrador Divisor da metade à esquerda do registrador Resto e armazena o resultado na metade esquerda do registrador Resto.

3. Testa se Resto é menor do que 0.

4. Resto < 0, restaura valor original com Divisor + metade esquerda do Resto, armazenando na metade esquerda do registrador Resto e deslocando 1 bit à esquerda, inserindo 0 no novo bit menos signifiativo, passa ao item 6.

5. Resto ≥ 0, desloca o registrador Resto 1 bit à esquerda, inserindo 1 no novo bit mais à direita.

6. Verifica se foram realizadas todas as repetições necessárias de acordo com o tamanho da palavra, se não volta ao item 1.

7. Desloca a metade a esquerda do registrador Resto 1 bit à direita, Fim.

Ponto Flutuante

Assim como os números decimais podem ser representados em notação científica normalizada os números binários também podem. A aritmética computacional que manipula os números binários em notação científica normalizada é denominada de aritmética de ponto flutuante.

Os projetistas do hardware devem encontrar um compromisso entre a mantissa e o expoente dos números em ponto flutuante. A relação entre a mantissa e o expoente é expressa do seguinte modo: o aumento do número de bits reservados à mantissa aumenta a precisão do número, enquanto o aumento do número de bits reservados ao expoente aumenta o intervalo de variação dos números representados.

Deve ser observado que as interrupções relativas ao overflow e ao underflow também ocorrem na representação em ponto flutuante. Porém, neste caso, overflow e o underflow ocorrem quando o expoente é muito grande ou muito pequeno, respectivamente, para ser armazenado no espaço reservado a ele.

Outra questão que os projetistas devem decidir é se vão ser utilizados os mesmos registradores tanto para números inteiros quanto para números de ponto flutuante. A adoção de registradores diferentes aumenta ligeiramente o número de instruções necessárias a execução do programa.

O impacto maior está na criação de um conjunto de instruções de transferência de dados para mover os dados entre os registradores de ponto flutuante e a memória. Os benefícios estão no fato de não precisar aumentar o tamanho do campo nas instruções para diferenciar os operandos e aumentar a banda passante dos registradores.

A partir de 1980 todos os computadores projetados adotam uma representação padrão para números em ponto flutuante denominada IEEE 754. A adoção deste padrão facilita a portabilidade de programas (precisão simples = 1 bit de sinal, 8 bits de expoente e 23 bits de mantissa + 1 implícito = 24, precisão dupla = 1 bit de sinal, 11 bits de expoente e 52 bits de mantissa + 1 implícito = 53).

A adição e a multiplicação com números de ponto flutuante, na sua essência, utilizam as operações inteiras correspondentes para operar as mantissas, mas é necessária uma manipulação extra nos expoentes e para a normalização do resultado.

Algoritmo da Adição em Ponto Flutuante

1. Compare o expoente dos dois números. Desloque o menor número à direita até que seu expoente se iguale ao maior número.

2. Some as mantissas.

3. Normalize a soma, deslocando à direita e incrementando o expoente ou deslocando à esquerda e decrementando o expoente.

4. Teste se há overflow ou underflow.

5. Sim, gera exceção.

6. Não, arredonde a mantissa para o número de bits apropriado.

7. Testa se resultado está normalizado.

8. Sim, Fim.

9. Não, retorna ao passo 3.

Algoritmo da Multiplicação em Ponto Flutuante

1. Soma os expoentes com peso dos dois números, subtraindo o valor do peso da soma para obter o novo expoente.

2. Multiplique as mantissas.

3. Normalize o produto se necessário, deslocando à direita e incrementando o expoente.

4. Teste se há overflow ou underflow.

5. Sim, gera exceção.

6. Não, arredonde a mantissa para o número de bits apropriado.

7. Teste se resultado está normalizado.

8. Não, retorna ao passo 3.

9. Sim, faça o sinal do produto positivo se ambos os sinais dos operandos originais são os mesmos, caso contrário o sinal é negativo.

Características e Recursos dos Processadores

Apesar do processador ser o componente mais importante do micro, já que é ele quem processa quase todas as informações, ele não é necessariamente o maior responsável pelo desempenho. Na verdade, dependendo da aplicação à qual o micro se destina, o desempenho do processador pode ser menos importante que a quantidade de memória RAM, que o desempenho da placa de vídeo 3D, ou até mesmo que o desempenho do disco rígido.

Tenha em mente que o computador é um conjunto, cada componente depende dos demais para mostrar o seu potencial.

Dizemos que um micro é tão rápido quanto seu componente mais lento. Como estamos falando de um conjunto, apenas um componente que apresente uma baixa performance será suficiente para colocar tudo a perder. Assim como vemos em outras situações, num carro por exemplo, onde um simples pneu furado pode deixar o carro parado na estrada.

Se o micro tiver pouca memória RAM por exemplo, o sistema operacional será obrigado a usar memória virtual, limitando a performance ao desempenho do disco rígido, que é centenas de vezes mais lento que ela. Caso o micro não possua memória cache, o desempenho ficará limitado ao desempenho da memória RAM, que é muito mais lenta que o processador e por aí vai.

Dizemos neste caso, que o componente de baixo desempenho é um gargalo, pois impede que o conjunto manifeste todo o seu potencial. Às vezes, simplesmente aumentar a quantidade de memória RAM, operação que custa relativamente pouco, é capaz de multiplicar a velocidade do micro.

Mas, apesar de tudo, o processador ainda é o componente básico de qualquer PC. Com o avanço cada vez mais rápido da tecnologia, e várias empresas disputando o mercado, os projetistas vem sendo obrigados a desenvolver projetos cada vez mais ousados a fim de produzir os processadores com o melhor desempenho.

Isso é excelente para nós, mas também pode trazer armadilhas, já que com projetos tão diferentes, cada processador acaba saindo-se bem em algumas aplicações, mas muito mal em outras. Não dá para julgar o desempenho do processador apenas pela frequência de operação, como fazíamos na época do 486, os tempos mudaram.

Mas, já que está aqui, que tal conhecermos os avanços pelos quais os processadores passaram até chegar aos dias de hoje? Vamos discutir primeiro algumas características básicas dos processadores, conhecer os pioneiros da década de 70 e avançar pelos anos 80 e 90, até chegar nos dias de hoje.

Características Básicas dos processadores modernos

Existem no mercado vários modelos de processadores, que apresentam preços e desempenho bem diferentes. Este tópico inicial se destina a estabelecer os diferenciais básicos que determinam a performance de um processador, a parte teórica que vai lhe ajudar a compreender a diferença entre os processadores que vamos examinar com detalhes mais adiante.

Quando vamos comprar um processador, a primeira coisa que perguntamos é qual sua frequência de operação, medida em Megahertz (MHz) ou milhões de ciclos por segundo, frequência também chamada de clock.

Acontece, que nem sempre um processador com uma velocidade de operação mais alta é mais rápido do que outro que opera a uma frequência um pouco mais baixa. A frequência de operação de um processador indica apenas quantos ciclos de processamentos são realizados por segundo, o que cada processador é capaz de fazer em cada ciclo já é outra história.

Imagine um processador 486 de 100 MHz, ao lado de um Pentium também de 100 MHz. Apesar da frequência de operação ser a mesma, o 486 perderia feio em desempenho. Na prática, o Pentium seria pelo menos 2 vezes mais rápido. Isto acontece devido à diferenças na arquitetura dos processadores e também no coprocessador aritmético e cache.

Coprocessador Aritmético

Todos os processadores da família x86, usada em micros PC, são basicamente processadores de números inteiros. Muitos aplicativos porém, precisam utilizar valores de maior precisão, assim como funções matemáticas complexas, como Seno, Coseno, Tangente, etc., para realizar suas tarefas. Este é o caso dos programas de CAD, planilhas, jogos com gráficos tridimensionais e de processamento de imagens em geral.

A função do coprocessador aritmético é justamente auxiliar o processador principal no cálculo destas funções complexas, cada vez mais utilizadas, principalmente em jogos. É como um matemático profissional que ajuda o processador a resolver os problemas mais complexos, que ele demoraria muito para resolver sozinho.

Até o 386, o coprocessador era apenas um acessório que podia ser comprado à parte e instalado num encaixe apropriado da placa mãe, sendo que cada modelo de processador possuía um modelo equivalente de coprocessador. O 8088 utilizava o 8087, o 286 o 287, o 386SX e 386DX utilizavam respectivamente o 387SX e o 387DX e o 486SX utilizava 487DX.

O problema nesta estratégia é que como poucos usuários equipavam seus micros com coprocessadores aritméticos, a produção destes chips era baixa, e consequentemente os preços eram altíssimos, chegando ao ponto de em alguns casos o coprocessador custar mais caro que o processador principal. Com o aumento do número de aplicativos que necessitavam do coprocessador, sua incorporação ao processador principal apartir do 486DX foi um passo natural. Com isso, resolveu-se também o problema do custo de produção dos coprocessadores, barateando o conjunto.

Atualmente, o desempenho do coprocessador determina o desempenho do micro em jogos e aplicativos gráficos em geral, justamente as aplicações onde os processadores atuais são mais exigidos. Infelizmente, o desempenho do coprocessador é uma característica que varia muito entre os processadores atuais.

Encaixe para o coprocessador aritmético

Memória Cache

Enquanto os processadores tornaram-se quase 10 mil vezes mais rápidos desde o 8088 (o processador usado no XT), a memória RAM, sua principal ferramenta de trabalho, pouco evoluiu em performance.

Quando foram lançados os processadores 386, percebeu-se que as memórias não eram mais capazes de acompanhar o processador em velocidade, fazendo com que muitas vezes ele tivesse que ficar “esperando” os dados serem liberados pela memória RAM para poder concluir suas tarefas, perdendo muito em desempenho.

Se na época do 386 a velocidade das memórias já era um fator limitante, imagine o quanto este problema não atrapalharia o desempenho dos processadores que temos atualmente. Para solucionar este problema, começou a ser usada a memória cache, um tipo ultra-rápido de memória que serve para armazenar os dados mais frequentemente usados pelo processador, evitando na maioria das vezes que ele tenha que recorrer à comparativamente lenta memória RAM.

Sem ela, o desempenho do sistema ficará limitado à velocidade da memória, podendo cair em até 95%!. São usados dois tipos de cache, chamados de cache primário, ou cache L1 (level 1), e cache secundário, ou cache L2 (level 2).

O cache primário é embutido no próprio processador e é rápido o bastante para acompanhá-lo em velocidade. Sempre que um novo processador é desenvolvido, é preciso desenvolver também um tipo mais rápido de memória cache para acompanhá-lo.

Como este tipo de memória é extremamente caro (chega a ser algumas centenas de vezes mais cara que a memória RAM convencional) usamos apenas uma pequena quantidade dela. O 486 traz apenas 8 KB, o Pentium traz 16 KB, enquanto o Pentium II e o Pentium III trazem 32 KB, enquanto o Athlon e o Duron da AMD trazem 128 KB.

Para complementar, usamos também um tipo um pouco mais lento de memória cache na forma do cache secundário, que por ser muito mais barato, permite que seja usada uma quantidade muito maior. Nos micros 486 o mais comum é o uso de 128 ou 256 KB de cache L2, enquanto nos micros mais modernos o mais comum é o uso de 512 KB. Dependendo do processador usado, o cache L2 pode vir embutido no próprio processador ou fazer parte da placa mãe.

Sempre que o processador precisar ler dados, os procurará primeiro no cache L1. Caso o dado seja encontrado, o processador não perderá tempo, já que o cache primário funciona na mesma frequência que ele. Caso o dado não esteja no cache L1, então o próximo a ser indagado será o cache L2. Encontrando o que procura no cache secundário, o processador já perderá algum tempo, mas não tanto quanto perderia caso precisasse acessar diretamente a memória RAM.

Por outro lado, caso os dados não estejam em nenhum dos dois caches, não restará outra saída senão perder vários ciclos de processamento esperando que eles sejam entregues pela lenta memória RAM. Para exemplificar, imagine que você estivesse escrevendo um e-mail e derrepente precisasse de uma informação que você havia anotado em um papel.

Se o papel estivesse sobre sua mesa, você poderia lê-lo sem perder tempo. Se estivesse dentro de uma gaveta da sua mesa, já seria necessário algum tempo para encontrá-lo enquanto se ele estivesse perdido em algum lugar de um enorme fichário do outro lado da sala, seria preciso um tempo enorme.

Antigamente, era comum as placas mães virem com soquetes apropriados, que permitiam ao usuário adicionar mais memória cache caso quisesse. Os módulos adicionais, chamados de módulos COAST (cache on a stick) eram relativamente acessíveis, levando muita gente a fazer o upgrade.

Entretanto, atualmente esta possibilidade não existe mais, pois a grande maioria dos processadores já trazem o cache L2 integrado, não permitindo qualquer modificação, já que não dá para abrir o processador e soldar mais cache. Mesmo no caso de processadores que ainda usam cache embutido na placa mãe, como o K6-2, não existe mais o encaixe para adicionar mais cache.

Ou seja, atualmente a quantidade de cache que você deseja no processador ou placa mãe deve ser decidida antes da compra, baseado nas opções disponíveis. Uma vez adquiridos o processador e a placa mãe não será possível fazer qualquer alteração.

Processadores RISC X Processadores CISC

Sempre houve uma grande polêmica em torno de qual dessas plataformas é melhor. Talvez você ache inútil eu estar falando sobre isto aqui, mas é interessante que você compreenda a diferença entre estas duas plataformas, para entender vários aspectos dos processadores modernos.

Um processador CISC (Complex Instruction Set Computer, ou "computador com um conjunto complexo de instruções"), é capaz de executar várias centenas de instruções complexas diferentes, sendo extremamente versátil. Exemplos de processadores CISC são o 386 e o 486.

No começo da década de 80, a tendência era construir chips com conjuntos de instruções cada vez mais complexos. Alguns fabricantes porém, resolveram seguir o caminho oposto, criando o padrão RISC (Reduced Instruction Set Computer, ou "computador com um conjunto reduzido de instruções").

Ao contrário dos complexos CISC, os processadores RISC são capazes de executar apenas algumas poucas instruções simples. Justamente por isso, os chips baseados nesta arquitetura são mais simples e muito mais baratos. Outra vantagem dos processadores RISC, é que, por terem um menor número de circuitos internos, podem trabalhar a frequências mais altas. Um exemplo são os processadores Alpha, que em 97 já operavam a 600 MHz.

Pode parecer estranho que um chip que é capaz de executar algumas poucas instruções, possa ser considerado por muitos, mais rápido do que outro que executa centenas delas, seria como comparar um professor de matemática com alguém que sabe apenas as quatro operações. Mas, um processador RISC é capaz de executar tais instruções muito mais rapidamente.

A idéia principal, é que apesar de um processador CISC ser capaz de executar centenas de instruções diferentes, apenas algumas são usadas frequentemente. Poderíamos então criar um processador otimizado para executar apenas estas instruções simples que são mais usadas. Como de qualquer forma, pouca gente programa diretamente em Assembly, bastaria alterar os compiladores, para que os programas fossem compatíveis com os novos processadores.

É indiscutível, porém, que em muitas tarefas os processadores CISC saem-se melhor, principalmente pelo seu grande número de recursos. Por isso, ao invés da vitória de uma das duas tecnologias, atualmente vemos processadores híbridos, que são essencialmente processadores CISC, mas incorporam muitos recursos encontrados nos processadores RISC (ou vice-versa).

Apesar de por questões de Marketing, muitos fabricantes ainda venderem seus chips, como sendo "Processadores RISC", não existe praticamente nenhum processador atualmente que siga estritamente uma das duas filosofias. Tanto processadores da família x86, como o Pentium II, Pentium III e AMD Athlon, quanto processadores supostamente RISC, como o MIPS R10000 e o HP PA-8000 misturam características das duas arquiteturas, por simples questão de performance. Por que ficar de um lado ou de outro, se é possível juntar o melhor dos dois mundos? A última coisa que os fabricantes de processadores são é teimosos, sempre que aparece uma solução melhor, a antiga e abandonada.

Examinando de um ponto de vista um pouco mais prático, a vantagem de uma arquitetura CISC é que já temos muitas das instruções guardadas no próprio processador, o que facilita o trabalho dos programadores, que já dispõe de praticamente todas as instruções que serão usadas em seus programas. No caso de um chip estritamente RISC, o programador já teria um pouco mais de trabalho, pois como disporia apenas de instruções simples, teria sempre que combinar várias instruções sempre que precisasse executar alguma tarefa mais complexa.

Seria mais ou menos como se você tivesse duas pessoas, uma utilizando uma calculadora comum, e outra utilizando uma calculadora cientifica. Enquanto estivessem sendo resolvidos apenas cálculos simples, de soma, subtração, etc. quem estivesse com a calculadora simples poderia até se sair melhor, mas ao executar cálculos mais complicados, a pessoa com a calculadora científica disporia de mais recursos.

Nos chips atuais, que são na verdade misturas das duas arquiteturas, juntamos as duas coisas. Internamente, o processador processa apenas instruções simples. Estas instruções internas, variam de processador para processador, são como uma luva, que se adapta ao projeto do chip. As instruções internas de um K6 são diferentes das de um Pentium por exemplo.

Sobre estas instruções internas, temos um circuito decodificador, que converte as instruções complexas utilizadas pelos programas em várias instruções simples que podem ser entendidas pelo processador. Estas instruções complexas sim, são iguais em todos os processadores usados em micros PC. É isso que permite que um K6 e um Pentium sejam compatíveis entre sí.

O conjunto básico de instruções usadas em micros PC é chamado de conjunto x86. Este conjunto é composto por um total de 187 instruções, que são as utilizadas por todos os programas. Além deste conjunto principal, alguns processadores trazem também instruções alternativas, que permitem aos programas executar algumas tarefas mais rapidamente do que seria possível usando as instruções x86 padrão. Alguns exemplos de conjuntos alternativos de instruções são o MMX (usado apartir do Pentium MMX), o 3D-NOW! (usado pelos processadores da AMD, apartir do K6-2), e o SSE (suportado pelo Pentium III).

PCs x Macs

Continuando na discussão de processadores RISC e CISC, vamos estudar um pouco sobre a arquitetura de dois processadores atuais, o G4, utilizado nos micros Macintosh e o AMD Athlon, usado em micros PC.

Existe uma idéia geral de que o G4, usado nos Macs é um processador RISC, enquanto os processadores usados em micros PC, incluindo o Pentium III e o Athlon são todos CISC. Ambas as afirmações estão erradas. Na verdade, tanto o G4, quanto o Athlon e o Pentium III são considerados processadores Post-RISC, processadores que possuem um conjunto de instruções gigantesco, maior do que o conjunto de instruções de um processador CISC típico.

A diferença é que toda essa gigantesca gama de instruções diferentes, podem ser decodificadas em instruções RISC simples, estas sim que serão processadas. A “conversão” das instruções é feita por um componente especial do processador, chamado de Hardware Decoder, encontrado tanto no G4 quanto no Athlon.

O G4 possui um enorme conjunto de instruções, assim como os processadores x86, mas todas instruções que podem ser convertidas pelo Hardware decoder e em seguida processadas. O Hardware Decoder é extremamente rápido, por isso não compromete o desempenho do processador.

De fato, a perda de desempenho por usar este grande conjunto de instruções que precisam ser quebradas em instruções menores é de menos de 1%. É por isso que os processadores atuais abandonaram a idéia RISC original: a perda de desempenho é ínfima perto do ganho de flexibilidade.

O Athlon por sua vez, tem que ser compatível com o conjunto de instruções x86, caso contrário não poderia ser usado em micros PC. As instruções x86 consistem em basicamente dois tipos de instruções, as instruções simples, que podem ser diretamente processadas pelo Hardware decoder, sem perda de tempo, e as instruções complexas, que são quebradas em instruções simples por outro componente, chamado Microcode decoder.

As instruções simples, que podem ser diretamente processadas, são as mais frequentemente usadas nos programas. De fato, num programa atual típico, é composto de entre 95 e 97% destas instruções simples.

O restante são as instruções complexas, que apesar de raramente usadas são as que dão mais trabalho, pois precisam passar por um processo de decodificação muito mais lento, feito pelo Microcode Decoder.

Para amenizar Este problema, a AMD incluiu um buffer de pré extração no Athlon, que funciona como uma espécie de fila por onde as instruções já decodificadas passam antes de ser processadas. Graças a isto, o processador pode processar outras instruções enquanto aguarda o Microcode Decoder decodificar cada instrução complexa, sem perder muito tempo.

Com isto, mesmo mantendo compatibilidade com o conjunto de instruções x86, o Athlon perde muito pouco em desempenho em relação ao G4, isto naturalmente comparando dois processadores de mesma frequência. O IPC, ou seja, o número de instruções processadas por ciclo de ambos é muito próximo, o que garante que um Athlon de 500 MHz apresente um desempenho muito parecido com um G4 também de 500 MHz.

Front End e Back End

Qualquer processador atual pode ser dividido em dois blocos básicos, o Front End e o Back End.

O Front End corresponde aos circuitos que decodificam as instruções, no caso o Hardware decoder, Microcode decoder e buffer de pré extração que acabei de explicar, junto com mais alguns componentes, como os circuitos de Branch Prediction (que ordenam as instruções de forma que o processador possa processador o maior número possível de instruções por ciclo e o cache L1. Estes componentes são a “porta de entrada” do processador, tendo a função de preparar as instruções para serem processadas.

O Back End é a parte do processador que finalmente processa as instruções, sendo composto basicamente pelas unidades de execução.

Como vimos, o fato como ambos os processadores decodificam as instruções, contando com o Hardware decoder é bastante semelhante, mas o Athlon possui alguns componentes a mais para garantir compatibilidade com as instruções x86. Isto não atrapalha o desempenho do processador, mas o torna um projeto mais complexo.

Em termos de unidades de execução, ou seja, o Back End, é que os processadores mostram mais algumas diferenças na forma como processam as instruções já decodificadas.

O Athlon possui um total de 9 unidades de execução, enquanto o G4 possui apenas 6. A diferença parece grande, mas na prática o desempenho é quase o mesmo, veja por que:

O Athlon possui 3 unidades de execução para leitura/gravação de dados na memória, enquanto o G4 possui apenas uma. O ponto é que todas as instruções, tanto de inteiros, quanto de ponto flutuante no Athlon, vem com um espaço reservado para uma instrução de leitura/gravação, espaço que nem sempre é preenchido, fazendo com que as 3 unidades fiquem ociosas na maior parte do tempo, apesar de agilizarem algo de vez em quando.

No G4, só existe uma unidade de leitura/gravação, mas que em compensação fica ocupada na maior parte do tempo. Na prática, esta única unidade acaba fazendo o mesmo volume de trabalho das três do Athlon, que ficam boa parte do tempo ociosas. Sem dúvida, o G4 perde alguma coisa em termos de desempenho, mas muito pouco.

Em termos de unidades de execução de inteiros e de ponto flutuante, que são as mais importantes, temos especificações parecidas em ambos:

O Athlon possui três unidades de ponto flutuante (que formam o coprocessador aritmético), o mesmo número encontrado no G4. Apenas para efeito de comparação, o Pentium 3 possui apenas duas. Com o mesmo número de unidades, o desempenho dos dois processadores no quesito ponto flutuante é quase igual.

Já em termos de unidades de processamento de inteiros, o cenário muda um pouco de figura, pois o Athlon possui três unidades de execução contra apenas duas do G4.

Isto garante que o Athlon tenha um desempenho um pouco melhor que o G4 em aplicativos de escritórios, mas a diferença é pequena, pois o desempenho real também depende do cache, velocidade de acesso à memória, etc.

Em termos de instruções 3D, o Athlon conta com o 3D-Now, o famoso conjunto de instruções, embutido nos processadores AMD que permite melhorar o desempenho do processador em jogos e aplicativos 3D. O Athlon traz também o velho MMX, que garante algum ganho em aplicativos multimídia.

O G4 por sua vez traz um conjunto unificado, o Altivec, que inclui tanto instruções 3D (como no 3D-Now!), quanto instruções multimídia (como no MMX), isto garante que tanto o Athlon quanto o G4 possuam armas semelhantes neste quesito, o resto fica por conta dos programadores.

Do 8086 ao Pentium MMX

O primeiro microprocessador foi lançado pela Intel em 1971 e se chamava i4004. Este era um processador extremamente simples, formado por pouco mais de 2000 transístores, mas que foi o precursor dos processadores que temos atualmente. A chamada lei de Moore, que leva o nome do fundador da Intel, Gordon Moore, prega que a potência dos processadores dobra a cada 18 meses. Apesar desta previsão ter sido feita no final da década de 70, continuou mantendo-se verdadeira até os dias de hoje, com uma precisão notável.

De lá pra cá, foi um longo caminho. Enormes investimentos foram feitos e muitos dos maiores gênios do planeta trabalharam em busca de soluções para questões cada vez mais complexas. Vamos agora examinar os avanços feitos desde o 8088, usado no XT, até o Pentium, onde estudaremos quando e porque recursos como o modo protegido e a multiplicação de clock foram introduzidos, e no que eles afetam o funcionamento do processador. Entendendo estes conceitos, você poderá facilmente entender as diferenças entre os processadores Pentium III, Athlon, K6-3 etc. que temos atualmente e veremos com mais detalhes adiante, assim como dos processadores que vierem a ser lançados futuramente que, pode ter certeza, continuarão utilizando os mesmos conceitos básicos.

8088

O 8088 era na verdade uma versão económica do processador 8086, que havia sido lançado pela Intel em 78. Quando a IBM estava desenvolvendo seu computador pessoal, chegou a ser cogitado o uso do 8086, mas acabou sendo escolhido o 8088 devido ao seu baixo custo.

Tanto o 8086 quanto o 8088 são processadores de 16 bits e eram considerados avançadíssimos para a época, apesar de serem extremamente simples para os padrões atuais. A diferença entre eles é que o 8088, apesar de internamente trabalhar com palavras binárias de 16 bits, usava um barramento de apenas 8 bits, o que permitiu à IBM utilizar os mesmos componentes usados nos computadores de 8 bits da época, que eram muito mais baratos do que os periféricos de 16 bits.

Esta arquitetura permitiu ao primeiro PC competir na mesma faixa de preço dos computadores de 8 bits mais populares e, ao mesmo tempo, possuir um desempenho bem superior devido ao seu processador de 16 bits. O 8088 é capaz de acessar até 1 MB de memória RAM, e funciona a 4.77 MHz, recursos incríveis para a época, já que estamos falando de um processador lançado no final de 1979.

Falando em recursos, só para matar sua curiosidade, o PC original da IBM, lançado em Agosto de 1981 possuía apenas 64 KB de memória RAM (a versão mais simples vinha com apenas 16 KB), monitor MDA mono de 12 polegadas, usava uma unidade de disquetes de 5 1/4 de apenas 160 KB e vinha sem disco rígido. O sistema operacional usado era o MS-DOS 1.0 (na época ainda chamado de PC-DOS), que foi desenvolvido pela Microsoft com base num sistema operacional mais simples, chamado QDOS, comprado da Seattle Computers, uma pequena empresa desenvolvedora de sistemas. Na verdade, a Microsoft foi a segunda opção da IBM, depois de ter sua proposta de licença recusada pela Digital Research, que na época desenvolvia versões do seu CP/M para várias arquiteturas diferentes.

Dois anos depois, foi lançado o PC XT, que apesar de continuar usando o 8088 de 4.77 MHz, vinha bem mais incrementado, com 256 KB de RAM, disco rígido de 10 MB, monitor CGA e o MS-DOS 2.0.

Mesmo com o surgimento dos micros 286, o XT ainda continuou sendo bastante vendido, pois era mais barato. Fabricantes de clones criaram projetos de micros XTs mais avançados, equipados com processadores 8088 de 8 MHz, discos rígidos maiores e até 640 KB de memória RAM.

Segmentação de Endereços

Um recurso bem interessante, usado no 8088, é a segmentação de endereços, que permitiu aumentar a quantidade de memória RAM suportada pelo processador.

Para que o processador possa acessar a memória RAM, é preciso que a memória seja dividida em endereços. Cada byte depositado na memória recebe um endereço único, assim como cada rua do Brasil tem um CEP diferente. Como o 8088 pode lidar apenas com palavras binárias de 16 bits, a princípio não seria possível para ele acessar mais do que 64 Kbytes de memória RAM, já que 16 bits permitem apenas 65,536 combinações diferentes (2 elevado à 16º potência).

Se o 8088 pudesse acessar apenas 64 KB de memória RAM, os micros baseados nele seriam muito limitados e poderiam apenas rodar programas muito simples. Para você ter uma idéia, 64 KB não dariam nem mesmo para carregar o DOS 3.0.

Para solucionar este problema, foi adotada uma solução bastante engenhosa: apesar do processador continuar podendo acessar apenas 64 KB de memória de cada vez, foram criados mais 4 bits de endereçamento, que permitem o acesso a 16 blocos de memória. Como cada bloco possui 64 KB, chegamos a 1 MB inteiro de capacidade total. Basicamente criamos 16 áreas diferentes de memória, cada uma com 64 KB, que é o máximo que o 8088 pode endereçar.

O processador pode acessar uma única área de cada vez. Se por exemplo, está sendo usado o bloco 1, e de repente é preciso ler um dado gravado no bloco 2, é preciso limpar todos os endereços relativos ao bloco 1 e carregar os endereços do bloco 2. Neste momento, o processador perde o acesso ao bloco 1 e passa a enxergar apenas o segundo bloco.

Quando novamente for preciso ler ou gravar dados no bloco 1 (ou qualquer outro bloco), novamente são carregados os endereços relativos a ele, e o acesso ao bloco 2 será perdido. É mais ou menos como se você precisasse fazer anotações em várias páginas de um caderno. Como só é possível ler ou escrever em uma página de cada vez, você precisaria ficar continuamente virando as páginas.

286

O processador 286 foi lançado em Fevereiro de 1982, apenas 6 meses após a IBM ter lançado o seu primeiro PC. Porém, o 286 passou a ser utilizado apenas em 1984, quando a IBM lançou o seu PC AT. Esta demora é justificável, pois, para lançar um computador usando o novo processador da Intel, foi preciso desenvolver toda uma nova arquitetura. Da placa de vídeo ao gabinete, praticamente tudo foi mudado, o que somado à burocracia e a longos períodos de testes antes do lançamento, demandou um certo tempo.

Atualmente, o período de desenvolvimentos dos periféricos é muito mais curto. Quase sempre quando um novo processador é lançado, já temos placas mãe para ele disponíveis quase que imediatamente, pois o desenvolvimento é feito de forma simultânea.

O 286 trouxe vários avanços sobre o 8088. Ele utilizava palavras binárias de 16 bits tanto interna quanto externamente, o que permitia o uso de periféricos de 16 bits, muito mais avançados do que os usados no PC original e no XT. O custo destes periféricos desta vez não chegou a ser um grande obstáculo, pois enquanto o PC AT estava sendo desenvolvido, eles já podiam ser encontrados com preços mais acessíveis.

O principal avanço trazido pelo 286 são seus dois modos de operação, batizados de “Modo Real” e “Modo Protegido”. No modo real, o 286 se comporta exatamente como um 8086 (apesar de mais rápido), oferecendo total compatibilidade com os programas já existentes. Já no modo protegido, ele manifesta todo o seu potencial, incorporando funções mais avançadas, como a capacidade de acessar até 16 Megabytes de memória RAM (usando os 24 bits de endereçamento do 286), multitarefa, memória virtual em disco e proteção de memória.

Assim que ligado, o processador opera em modo real, e com uma certa instrução, passa para o modo protegido. O problema é que trabalhando em modo protegido, o 286 deixava de ser compatível com os programas escritos para o modo real, inclusive com o próprio MS-DOS. Para piorar, o 286 não possuía nenhuma instrução que fizesse o processador voltar ao modo real, isto era possível apenas resetando o micro.

Isso significa que um programa escrito para rodar em modo protegido, não poderia usar nenhuma das rotinas de acesso a dispositivos do MS-DOS, tornando inacessíveis o disco rígido, placa de vídeo, drive de disquetes memória, etc., a menos que fossem desenvolvidas e incorporadas ao programa todas as rotinas de acesso a dispositivos necessárias. Isso era completamente inviável para os

desenvolvedores, pois para projetar um simples jogo, seria praticamente preciso desenvolver todo um novo sistema operacional. Além disso, o programa desenvolvido rodaria apenas em micros equipados com processadores 286, que ainda eram minoria na época, tendo um público alvo muito menor. De fato, apenas algumas versões do UNIX e uma versão do OS/2 foram desenvolvidas para utilizar o modo protegido do 286.

Basicamente, os micros baseados no 286 eram usados para rodar aplicativos de modo real, que também podiam ser executados em um XT, aproveitando apenas a maior velocidade do 286. Falando em velocidade, a primeira versão do 286 funcionava a apenas 6 MHz, sendo lançada logo depois uma nova versão de 8 MHz, que foi usada no PC AT. Posteriormente, foram desenvolvidas versões de até 20 MHz.

Devido às várias mudanças na arquitetura, destacando o acesso mais rápido à memória e alterações no conjunto de instruções do processador, que permitiam realizar muitas operações de maneira mais rápida e eficiente, um 286 consegue ser quase 4 vezes mais rápido que um 8088 do mesmo clock.

386

O 386 foi lançado apenas em Outubro de 85, três anos e meio depois do 286. Desta vez, a diretoria da IBM demorou muito para chegar à um acordo e desenvolver um sistema baseado no 386, dando tempo para a Compaq sair na frente. Este foi um verdadeiro marco pois, de repente, as companhias perceberam que não eram mais obrigadas a seguir a IBM.

Qualquer um que tivesse tecnologia suficiente poderia sair na frente, como fez a Compaq. A partir daí, a IBM começou a gradualmente perder a liderança do mercado, tornando-se apenas mais um entre inúmeros fabricantes de PCs.

O 386 trouxe vários recursos novos. Para começar, o 386 trabalha tanto interna quanto externamente com palavras de 32 bits e é capaz de acessar a memória usando um barramento de 32 bits, permitindo uma transferência de dados duas vezes maior. Como o 386 pode trabalhar com palavras binárias de 32 bits, é possível acessar até 4 GB de memória (2 elevado à 32º potência), mesmo sem usar a segmentação de endereços, como no 8088 e no 286.

Assim como o 286, o 386 continua possuindo os dois modos de operação. A diferença é que no 386 já é possível alternar entre o modo real e o modo protegido livremente. Um programa que rode sobre DOS, pode chavear o processador para o modo protegido, para beneficiar-se de suas vantagens, e voltar ao modo real sempre que precisar usar alguma sub-rotina do DOS, de maneira transparente ao usuário. Neste caso, é usado um programa de DPMI (“DOS Protected Mode Interface”, ou “interface DOS de modo protegido”) para fazer o chaveamento entre os dois modos.

Toda vez que o programa precisa usar alguma sub-rotina do DOS, ele passa o comando ao chaveador e fica esperando. O chaveador por sua vez, passa o processador para o modo real, executa o comando, chaveia o processador para o modo protegido e entrega o resultado ao aplicativo, que continua trabalhando como se nada tivesse acontecido. Um bom exemplo de programa de DPMI é o DOS4GW, que é usado por muitos jogos que rodam sobre o MS-DOS, como o Doom, Sim City 2000 e vários emuladores de vídeo-games.

O esquema de chaveamento também é utilizado pelo Windows 3.x, que já inclui todas as rotinas necessárias, dispensando qualquer programa de DPMI. O Windows 95/98 também pode chavear para o modo real caso precise carregar algum driver de dispositivo de modo real. Porém, devido ao modo virtual 8086, que veremos logo a seguir, não é preciso colocar o processador em modo real para executar aplicativos MS-DOS dentro do Windows 95/98

Ter um processador 386 é o requisito mínimo para rodar qualquer sistema operacional ou aplicativo de modo protegido moderno. Com um 386, um mínimo de memória RAM e espaço em disco suficiente, você pode rodar o Windows 95 e a maioria dos aplicativos para ele, embora bem lentamente devido à pouca potência do processador.

Com um simples 286, no máximo você poderá rodar o DOS e aplicativos mais simples, que trabalhem somente com o modo real. Também é possível rodar o Windows 3.0, porém em modo “Standard”, onde é possível acessar todos os 16 MB de memória permitidos pelo 286, mas sem memória virtual nem multitarefa.

A Introdução do Cache

Os processadores 386 acima de 20 MHz eram muito rápidos para as memórias RAM existentes na época. Por isso, a cada acesso, o processador tinha que ficar “esperando” os dados serem liberados pela memória RAM para poder concluir suas tarefas, perdendo muito em desempenho. Para solucionar esse problema, passaram a ser usadas pequenas quantidades de memória cache na grande maioria das placas mãe para micros 386 e superiores.

A memória cache é um tipo de memória ultra-rápida, que armazena os dados mais usados pelo processador, evitando na grande maioria dos casos, que ele precise perder tempo buscando dados diretamente na lenta memória RAM. Mesmo uma pequena quantidade de memória cache é capaz de melhorar bastante a velocidade da troca de dados entre o processador e a RAM.

Apesar de já ser bem mais rápido que a memória RAM, o 386 ainda não era um processador muito rápido, justamente por isso, ainda não era tão dependente do desempenho da memória cache quanto os processadores atuais. Um 386 equipado com memória cache é de 20 a 30% mais rápido que um 386 da mesma frequência, mas sem memória cache, enquanto um processador moderno pode ficar até 20 vezes mais lento caso sejam desabilitados tanto o cache L1 quanto o cache L2.

386SX

Como o 386 era um processador de 32 bits, foi preciso desenvolver toda uma nova categoria de chipsets e circuitos de apoio para trabalhar com ele, o que acabou encarecendo bastante os sistemas baseados no 386 e afastando muitos compradores em potencial.

Para contornar este problema, a Intel optou por lançar uma versão de baixo custo do 386, batizada de 386SX, que apesar de continuar funcionando internamente com palavras de 32 bits, comunicava-se com a memória RAM e os demais periféricos usando palavras de 16 bits (como o 286). Apenas para diferenciar os dois processadores, a Intel passou a chamar o 386 original de 386DX.

Esta arquitetura permitiu que fossem aproveitados os mesmos periféricos usados em placas de micros 286, tornando as máquinas baseadas no 386SX muito mais acessíveis. Pra você uma idéia, um PC básico equipado com um 386SX, chegava a custar menos de 1,000 dólares, quase metade de um equipamento com uma configuração parecida baseado no 386DX.

Apesar de, devido ao preço, o 386SX ter tornado-se uma boa opção em termos de custo-beneficio, em termos de performance ele fica bem atrás de um 386DX da mesma frequência, pois apesar de internamente os processadores serem idênticos, o SX usa praticamente os mesmos componentes usados nos micros 286, acessa a memória usando palavras de 16 bits e, para completar, as placas mãe para ele não possuem memória cache.

Modo Real x Modo Protegido

Operando em modo real, o processador funciona exatamente como um 8086, apenas trabalhando com uma velocidade maior. Não somente o 386, mas todos os processadores atuais podem alternar entre o modo real e o modo protegido livremente, sempre que necessário. No modo real, rodamos o MS-DOS e outros aplicativos de modo real mais antigos, enquanto no modo protegido rodamos o Windows e seus programas.

Com certeza, alguma vez ao tentar rodar um programa antigo, você já se deparou com uma enigmática mensagem de falta de memória, apesar dos manuais do programa dizerem que ele precisa apenas de 500 ou 600 KB de memória e você ter instalado bem mais do que isso. Estas mensagens surgem por que estes programas rodam com o processador operando em modo real onde, como o 8086, ele é capaz de reconhecer apenas o primeiro Megabyte da memória RAM. Este primeiro Megabyte por sua vez, é subdividido em dois blocos, chamados de memória convencional e memória estendida.

A memória convencional corresponde aos primeiros 640 Kbytes da memória, e é a área de memória usada pelos programas que operam em modo real. Os 384 Kbytes restantes são chamados de memória superior, e são reservados para armazenar uma cópia do BIOS, que passa a ser executado mais rapidamente, já que a memória RAM é muito mais rápida do que o chip de memória ROM ou

Flash onde ele é originalmente armazenado. Esta cópia do BIOS é chamada de “Shadow”, ou sombra, e serve para aumentar o desempenho geral do sistema. A memória superior também é usada para armazenar sombras dos BIOS de outros dispositivos, como placas de vídeo, aumentando também a velocidade de operação destes periféricos.

Apesar de existirem 640 Kbytes de memória convencional, protos para ser usada por qualquer programa que opere em modo real, nem toda esta memória fica disponível, já que parte dela é usada pelo MS-DOS e drivers de dispositivos de modo real. É possível liberar mais memória convencional, editando os arquivos de inicialização do DOS, conseguindo assim rodar estes programas.

Quando o computador é ligado, o processador está operando em modo real. Quem dá o comando para que ele mude para o modo protegido é o sistema operacional. No caso do Windows, este comando é dado durante o carregamento do sistema.

Em modo protegido, o processador é capaz de reconhecer toda a RAM instalada no sistema, além de incorporar recursos como a multitarefa e a memória virtual em disco. É neste modo que usamos a interface gráfica do Windows e rodamos seus aplicativos.

Recursos do Modo Protegido

Apesar de, em nome da compatibilidade retroativa com programas desenvolvidos para micros PC XT e 286, tanto o 386 como todos os processadores atuais poderem operar em modo real, apenas no modo protegido eles incorporam os recursos mais avançados, que permitem a existência dos softwares que temos atualmente.

A partir do 386, poucas funções novas foram incorporadas aos novos processadores. Basicamente, evoluímos apenas em termos de velocidade. Tanto que, com um simples 386, é possível rodar praticamente qualquer aplicativo mais atual, apenas com uma velocidade menor.

O modo protegido traz basicamente quatro novos recursos: memória virtual, multitarefa, proteção de memória e o modo virtual 8086.

Memória Virtual

A capacidade do 386 de trabalhar com vários aplicativos ao mesmo tempo (multitarefa) é realmente muito útil, mas esta característica traz um pequeno problema: abrindo vários aplicativos sucessivamente, logo a memória RAM do sistema se esgota. Para corrigir este problema, o modo protegido traz também a memória virtual, que permite criar um arquivo temporário no disco rígido, chamado de Swap File, ou arquivo de troca, que funciona como uma extensão da memória RAM, permitindo abrir quantos aplicativos forem necessários, até que o espaço do disco rígido se esgote.

Por exemplo, só o Windows 2000 Professional, junto com os serviços básicos ocupa cerca de 40 MB de memória. Se você abrir o Word 97, serão necessários mais 10 Megabytes, um total de quase 50 MB. Caso o micro em questão possua apenas 32 MB de memória, seria criado um arquivo temporário de 18 MB no disco rígido, que armazenaria os dados que não couberam na memória RAM.

O problema em usar memória virtual é que o disco rígido é centenas de vezes mais lento do que a memória RAM. Um disco rígido razoável possui um tempo de acesso em torno de 10 milessegundos (milésimos de segundo) enquanto um módulo de memória PC-100 possui um tempo de acesso inferior a 10 nanossegundos (bilionésimos de segundo) ou seja, um tempo de acesso um milhão de vezes menor!Em termos de taxa de transferência, novamente temos um contraste marcante: 800 MB para o módulo de memória e de 5 a 20 MB (dependendo do modelo) para o disco rígido.

Graças a este abismo, apesar dos programas funcionarem normalmente usando memória virtual, o sistema vai ficando cada vez mais lento. Experimente, por exemplo, tentar trabalhar em um PC com apenas 4 MB de RAM (seja qual for o processador) rodando o Windows 95. A lentidão é insuportável.

No Windows 3.x, era necessário reservar uma quantidade espaço do disco rígido para a memória virtual, quantidade que podia ser configurada livremente através do Painel de Controle. O problema é que este espaço ficava indisponível. Se você possuísse um disco de 800 MB, e reservasse 200 para a memória virtual, ficaria com apenas 600 MB para instalar programas e guardar arquivos. Se por

outro lado, você reservasse pouco espaço para a memória virtual, ficaria com pouca memória para abrir vários programas e trabalhar com arquivos grandes.

Apartir do Windows 95 este problema foi resolvido com a adoção de um arquivo de troca dinâmico, que vai aumentando ou diminuindo de tamanho conforme a necessidade de memória, evitando o desperdício de espaço em disco que tínhamos no Windows 3.x.

Apartir do Windows 95, existe também uma administração mais racional dos recursos do sistema, movendo os arquivos mais importantes, acessados com mais frequência para memória RAM (ou memória cache, dependendo da importância do arquivo), e deixando apenas arquivos usados mais raramente no arquivo de troca. Esta simples medida diminui bastante a perda de performance causada pelo uso da memória virtual.

No Windows 2000 é possível determinar um valor inicial e um valor máximo para um arquivo de troca. No caso do Linux, a fim de melhorar o desempenho, os desenvolvedores optaram por criar um sistema de arquivos próprio para a memória virtual.

Multitarefa

Multitarefa significa executar mais de uma tarefa de cada vez, como assobiar e chupar cana ao mesmo tempo. Apesar de na vida real não ser muito fácil fazer duas coisas ao mesmo tempo, do ponto de vista de um computador este processo é relativamente simples.

Todos os aplicativos são carregados na memória e o processador passa a executar algumas instruções de cada aplicativo por vez. Como o processador é capaz de executar vários milhões de instruções por segundo, esta troca é feita de maneira transparente, como se os aplicativos estivessem realmente sendo executados ao mesmo tempo. Enquanto o processador dá atenção para um aplicativo, todos os demais ficam paralisados, esperando sua vez.

Memória Protegida

Usando a multitarefa, quase sempre teremos vários aplicativos carregados na memória, seja na memória RAM ou no arquivo de troca. Se não houvesse nenhum controle por parte do processador, um aplicativo poderia expandir sua área de memória, invadindo áreas de outros aplicativos e causando travamentos no micro.

Um editor de imagens, por exemplo, precisa ocupar mais memória conforme as imagens vão sendo abertas ou criadas. Sem nenhuma orientação por parte do processador, simplesmente seriam ocupadas as áreas adjacentes, que poderiam tanto estar vazias, quanto estar ocupadas pelo processador de textos, por exemplo.

Para colocar ordem na casa, foi desenvolvido o recurso de proteção de memória, que consiste no processador isolar a área de memória ocupada por cada aplicativo, impedindo que ele ocupe outras áreas ao seu bel prazer. Se, por acaso, o programa precisar de mais memória, o próprio processador irá procurar uma área vazia de memória e ordenar ao aplicativo que ocupe a área reservada.

Existem basicamente dois tipos de multitarefa, denominadas multitarefa preemptiva e multitarefa cooperativa, que diferem justamente pelo uso ou não da proteção de memória.

O Windows 3.x, apesar de ser considerado um sistema operacional multitarefa, não é capaz de usar o recurso de proteção de memória, nele é usada a multitarefa cooperativa, que consiste em cada aplicativo usar os recursos do processador por um certo tempo, passar para outro programa e esperar novamente chegar sua vez para continuar executando suas tarefas.

A alternância entre os programas neste caso não é comandada pelo sistema e sim pelos próprios aplicativos. Neste cenário, um aplicativo mal comportado poderia facilmente monopolizar o sistema, consumindo todos os recursos do processador por um longo período, ou mesmo invadir áreas de memória ocupadas por outros aplicativos, causando em qualquer um dos casos o famoso GPF, (“General Protection Falt”, ou “falha geral de proteção”) que tanto atormentava os usuários do Windows 3.x.

Experimente tentar fazer dois irmãos dividirem os mesmo brinquedo; pode funcionar durante um certo tempo, mas uma hora um não vai querer deixar o outro brincar e vai sair briga, exatamente como acontece com os aplicativos dentro da multitarefa cooperative.

O Windows 95/98 por sua vez, usa a multitarefa preemptiva, isolando as áreas de memória ocupadas pelos aplicativos. Isto garante uma estabilidade bem maior do que a que temos no Windows 3.11. Porém, o modo como a multitarefa preemptiva é implementada no Windows 95 assim como do Windows 98 e do Windows Millennium, que são baseados no mesmo kernel (núcleo) do Windows 95, ainda possui dois problemas graves:

O primeiro é que, quando é executado um programa de 16 bits, o Windows 95 cai em multitarefa cooperativa para poder rodar o programa, deixando de proteger as áreas de memória e tornando-se tão vulnerável quanto o Windows 3.11.

Porém, mesmo usando apenas aplicativos de 32 bits os travamentos ainda são comuns, pois o Windows 95 os serviços do sistema não tem prioridade sobre os aplicativos. Isto significa que caso um aplicativo qualquer entre em loop, poderá consumir todos os recursos do processador, neste caso o sistema operacional ficará paralisado, simplesmente sem ter como fechar o aplicativo e restaurar o sistema, obrigando o usuário a resetar o micro e perder qualquer trabalho que não tenha sido salvo. Na verdade costuma-se dizer que o Windows 95/98 utiliza multitarefa semi-preemptiva, pois não utiliza todos os recursos de uma verdadeira multitarefa.

A solução para este problema veio com o Windows NT. Desde suas primeiras versões, o Windows NT é bem estável neste aspecto, pois implementa a multitarefa preemptiva de forma completa. As tarefas executadas pelo sistema operacional, são priorizadas sobre as de qualquer outro aplicativo. Isto significa que em nenhuma situação, um aplicativo terá como passar por cima do sistema operacional e consumir todos os recursos do processador como acontece no Windows 95/98.

Na prática, significa que o sistema até pode travar devido a algum bug, mas se algum aplicativo travar ou tentar invadir uma área de memória não designada para ele, simplesmente será fechado, permitindo que todos os demais aplicativos continuem trabalhando sem problemas. Você logo notará quais aplicativos costumam dar problemas, bastando substituí-los por versões mais recentes que corrijam seus bugs ou mesmo passar a usar um programa concorrente.

Tanto o Windows 2000, quanto o XP são baseados no kernel do Windows NT e mantém o mesmo sistema de funcionamento. Por ter sido inspirado no Unix, o Linux utiliza multitarefa preemptiva desde suas primeiras versões, é por isso que o Linux é considerado um dos sistemas mais estáveis, a ponto de ser usado em vários dos mais importantes servidores do planeta.

O MacOS por sua vez, utilizou a multitarefa cooperativa durante muito mais tempo, até a versão 9.x. Os usuários dos Mac só passaram a ter disponível um sistema com multitarefa preemptiva apartir do MacOS X, que é baseado no FreeBSD, um sistema Unix de código aberto, semelhante ao Linux em vários aspectos. A Apple usou o FreeBSD para construir o Darwin, que é a base do sistema e completou a obra com a interface Aqua, que mantém a idéia de facilidade de uso das versões anteriores do MacOS.

Modo Virtual 8086

Apesar de, operando em modo real, o processador ser totalmente compatível com qualquer programa antigo, seria impossível executar um aplicativo de modo real dentro do Windows 95 ou qualquer outro sistema operacional que utilize o modo protegido. Seria preciso fechar o Windows e fazer o processador voltar para o modo real para poder executar o aplicativo.

Pensando nesta possível limitação, os projetistas da Intel desenvolveram o modo virtual 8086 onde o processador, operando em modo protegido, é capaz de simular vários ambientes de modo real, cada um com 1 MB de memória e total acesso ao hardware do micro, chamados de máquinas virtuais. É como se dentro do 386 fossem abertos vários XTs completos, um para cada programa de modo real a ser executado. É justamente o modo virtual 8086 que permite abrir janelas DOS dentro do Windows 95/98.

Como o processador continua em modo protegido, cada máquina virtual tem sua área isolada na memória. O programa roda sem prejudicar a estabilidade do sistema.

486

O 386 foi o grande marco dos processadores para micros PC, pois foi o primeiro processador a trazer o conjunto de instruções x86, que são suportadas por todos os processadores modernos. Apartir dele, surgiram vários melhoramentos, mas apenas em termos de desempenho.

Apesar de não trazer instruções novas, o 486 conquistou seu lugar na história, por trazer vários recursos que continuam sendo usados até os processadores atuais. Em primeiro lugar, o 486 foi o primeiro processador a trazer cache integrado. Eram 8 Kbytes, mas que eram capazes de entregar dados a cada ciclo do processador. Como os fabricantes continuaram incluindo cache na placa mãe, um pouco mais lentos, mas em maior quantidade, surgiu também a distinção entre o cache L1 e o L2.

Outra evolução foi o coprocessador aritmético. Ao invés do caríssimo componente que deveria ser adquirido separadamente, o coprocessador passou a ser um item de série. Este foi o impulso que faltava para a popularização de vários programas e o surgimento de jogos bem mais elaborados.

Com tudo isso, um 486 é quase duas vezes mais rápido do que um 386 da mesma frequência. Em alguns aplicativos, que dependem do coprocessador aritmético, um 486 chega a ser 10 vezes mais rápido.

Como fez anteriormente com o 386, a Intel criou um 486 de baixo custo chamado de 486SX. A diferença entre o SX e o 486 original, que passou a ser chamado de 486DX. Os dois compartilhavam a mesma arquitetura, mas o SX vinha sem o coprocessador aritmético, o que o tornava muito mais lento em aplicativos gráficos e científicos.

Para os proprietários, existia a opção de posteriormente comprar um 80487SX, um coprocessador aritmético que era vendido separadamente. O problema era que comprado separadamente, o coprocessador custava quase tanto quanto um processador 486DX que já vinha com o coprocessador embutido, definitivamente um péssimo negócio. Para evitar confusão, o 486 original passou a ser chamado de 486DX.

Foram lançadas versões do 486 rodando à 25 MHz, 33 MHz e 40 MHz, porém, criou-se uma barreira, pois não haviam na época circuitos de apoio capazes de trabalhar a mais de 40 MHz. Para solucionar esse problema, foi criado o recurso de Multiplicação de Clock, através do qual o processador trabalha internamente à uma velocidade maior do que a da placa mãe. Foram lançados então os processadores 486DX2 (que trabalhavam ao dobro da frequência da placa mãe) e logo depois os 486DX4 (que trabalhavam ao triplo da frequência da placa mãe):

Processador	Placa mãe	Multiplicador
486DX-2 50 MHz	25 MHz	2x
486DX-2 66 MHz	33 MHz	2x
486DX-2 80 MHz	40 MHz	2x
486DX-4 75 MHz	25 MHz	3x
486DX-4 100 MHz	33 MHz	3x
486DX-4 120 MHz	40 MHz	3x

Com isso, surgiram também as placas mãe upgradable, que permitem atualizar o processador, apenas configurando alguns jumpers da placa.

Os processadores 486, apartir do DX-33 foram os primeiros a utilizar cooler, que naquela época eram dissipadores com menos de um centímetro de altura, com exaustores minúsculos. Conforme os processadores passaram a dissipar cada vez mais calor, os coolers foram crescendo na mesma proporção, até chegar nos exageros que vemos atualmente ☺

Multiplicação de Clock

Dentro de qualquer computador, os dados são transmitidos e processados na forma de sinais elétricos. O processador é muito pequeno, não mede mais do que 1, ou 1,2 centímetros quadrados. A placa mãe por sua vez é muito maior que isso.

Graças a esta diferença de proporções, acaba sendo muito mais fácil desenvolver um processador capaz de operar a, digamos, 2 gigahertz, do que uma placa mãe capaz de acompanha-lo. Apesar dos sinais elétricos percorrerem os circuitos a uma velocidade próxima da da luz, estamos falando de bilhões de transmissões por segundo.

O recuso de multiplicação de clock surgiu para evitar que os processadores ficassem limitados à frequência da placa mãe. Num Pentium III de 800 MHz por exemplo, a placa mãe opera a apenas 100 MHz. O multiplicador é de 8x.

Hoje em dia os processadores trazem tanto cache L1, quanto cache L2 integrados, operando na mesma frequência do restante do processador, o que diminui muito a dependência da velocidade da memória RAM, que sempre opera na mesma frequência de a placa mãe, meros 100 ou 133 MHz. Mesmo assim, quanto maior for o multiplicador, maior será a perda de desempenho. Um bom exemplo disso, é uma comparação entre o Celeron 766 (que usa bus de 66 MHz) e o Celeron 800 (que já usa bus de 100 MHz). Apesar da frequência de operação ser quase a mesma, o Celeron 800 chega a ser 20% mais rápido, graças ao acesso mais rápido à memória.

Apesar das limitações, o recurso de multiplicação de clock é indispensável atualmente, pois sem ele seria impossível desenvolver processadores muito rápidos, já que não é possível aumentar a frequência das placas mãe e dos demais periféricos na mesma proporção do aumento do clock nos processadores. Se o Pentium III, por exemplo, tivesse que trabalhar na mesma frequência da placa mãe, não passaríamos de 100 ou 133 MHz.

Nos PCs 486, Pentium, MMX e K6 é necessário configurar o multiplicador manualmente, através de alguns jumpers da placa mãe. É uma maravilha, principalmente quando você não têm o manual da placa em mãos. Mas, apartir do Pentium II, a placa é capaz de detectar automaticamente o multiplicador. Na verdade, apartir do Pentium II, todos os processadores Intel têm o seu multiplicador travado ainda na fábrica. Não é possível alterá-lo mesmo que queira.

Pipeline

Até o 386, os processadores da família x86 eram capazes de processar apenas uma instrução de cada vez. Uma instrução simples podia ser executada em apenas um ciclo de clock, enquanto instruções mais complexas demoravam vários ciclos de clock para serem concluídas. Seria mais ou menos como montar um carro de maneira artesanal, peça por peça.

Para melhorar o desempenho do 486, a Intel resolveu usar o pipeline, uma técnica inicialmente usada em processadores RISC, que consiste em dividir o processador em vários estágios distintos. O 486, possui um pipeline de 5 níveis, ou seja, é dividido em 5 estágios.

Quando é carregada uma nova instrução, ela primeiramente passa pelo primeiro estágio, que trabalha nela durante apenas um ciclo de clock, passando-a adiante para o segundo estágio. A instrução continua então sendo processada sucessivamente pelo segundo, terceiro, quarto e quinto estágios do processador. A vantagem desta técnica, é que o primeiro estágio não precisa ficar esperando a instrução passar por todos os demais para carregar a próxima, e sim carregar uma nova instrução assim que se livra da primeira, ou seja, depois do primeiro pulso de clock.

As instruções trafegam dentro do processador na ordem em que são processadas. Mesmo que a instrução já tenha sido processada ao passar pelo primeiro ou segundo estágio, terá que continuar seu caminho e passar por todos os demais. Se por acaso a instrução não tenha sido completada mesmo após passar pelos 5, voltará para o primeiro e será novamente processada, até que tenha sido concluída.

Desta maneira, conseguimos que o processador seja capaz de processar simultaneamente, em um único ciclo de clock, várias instruções que normalmente demorariam vários ciclos para serem processadas. Voltando ao exemplo do carro, seria como se trocássemos a produção artesanal por uma linha de produção, onde cada departamento cuida de uma parte da montagem, permitindo montar vários carros simultaneamente.

O uso dos 5 estágios de pipeline no 486 não chega a multiplicar por cinco a performance do processador, na verdade a performance não chega nem mesmo a dobrar, mas o ganho é bem significativo.

Pentium

Assim como o 486, o Pentium é um processador de 32 bits, capaz de acessar até 4 GB de memória RAM. Mas, novamente o processador trouxe várias melhorias que o tornaram muito mais rápido que a geração anterior. Não é à toa que o primeiro Pentium operava a apenas 60 MHz, e era, na época do lançamento, muito mais caro que um 486DX4-100. O Pentium é de 65 a 100% mais rápido que um 486 do mesmo clock. Como o processador aritmético também foi completamente remodelado, o Pentium acaba sendo ainda mais rápido em aplicativos que demandam um grande número de cálculos.

Os processadores Pentium existiram em versões de 60 a 200 MHz, sempre utilizando multiplicação de clock (com exceção apenas para as versões de 60 e 66 MHz):

Processador	Placa Mãe	Multiplicador
Pentium 60, 90, 120, 150 e 180 MHz	60 MHz	Respectivamente 1x, 1.5x, 2x, 2.5x e 3x
Pentium 66, 100, 133, 166 e 200 MHz	66 MHz	Respectivamente 1x, 1.5x, 2x, 2.5x e 3x
Pentium 75 MHz	50 MHz	1.5 x
Pentium 200 MHz	66 MHz	3 x

Como na época dos micros 486, as placas mãe para processadores Pentium (com excessão de placas muito antigas) suportam várias frequências de barramento e vários multiplicadores distintos. Na maioria dos casos é possível configurar a placa mãe para utilizar qualquer processador da família.

Melhorias no Cache L1

A primeira mudança trazida pelo Pentium foi o aumento da quantidade de cache L1, que passou a ser de 16 KB, o dobro do 486. Além do aumento da capacidade, foram implementados três novos recursos, que tornaram o cache ainda mais eficiente:

A primeira medida foi a divisão do cache em dois blocos de 8 KB, um dedicado a armazenar dados e outro dedicado a armazenar instruções. Esta divisão permite que tanto as instruções a serem executadas pelo processador (comparação, multiplicação, soma, decisão, etc.) quanto os dados a serem processados, possam ser acessados simultaneamente no cache, aumentando sua eficiência.

Se, por exemplo, um programa qualquer ordena que o processador leia um número gravado na memória e verifique se ele é maior ou menor que 10, temos duas instruções (ler o número e compará-lo com o número 10) e duas variáveis (o número 10 e o número a ser lido).

Com um cache unificado, como no 486, primeiro seriam lidas as instruções e em seguida as variáveis. No cache dividido do Pentium, ambos podem ser lidos no mesmo ciclo de clock, poupando tempo.

Outra modificação foi a ampliação do barramento de dados entre o processador e o cache. Enquanto no 486 podem ser lidos ou gravados até 128 bits de dados por ciclo de clock, no Pentium podem ser lidos ou gravados até 256 bits no cache de instruções e mais 256 no cache de dados. Como ambos os caches podem ser acessados simultaneamente, temos um barramento total de 512 bits, o quádruplo do que tínhamos no 486! Este barramento mais largo permite que quantidades maiores de dados possam ser lidos a partir do cache no mesmo espaço de tempo, permitindo ao cache acompanhar a maior velocidade de processamento do Pentium.

A última medida foi a adoção de um cache Write Back, que é capaz de cachear tanto as operações de leitura de dados na memória RAM, quanto as operações de escrita.

O cache usado no 486, cacheia apenas as operações de leitura, o que permite ao processador ganhar tempo ao ler dados, mas não ajuda na hora de gravar dados, quando são perdidos vários ciclos até que a memória RAM torne-se disponível.

Previsão de Desvio Dinâmico

Após concluída uma instrução do programa, para que o processador não perca um único ciclo de clock aguardando que o cache ou a memória RAM enviem a próxima instrução a ser processada, foi incluído no Pentium um buffer de pré extração. Este pequeno circuito armazena as próximas instruções a serem processadas, formando uma espécie de fila. Na verdade, o buffer de pré extração funciona como uma espécie de cache L0, ficando entre o processador e o cache L1.

A maior utilidade deste buffer é prever o resultado de operações de tomada de decisão. Se chega ao processador uma instrução como “Se X > Y então Z = K, senão Z = Q” o buffer irá carregar tanto a instrução seguinte para X < Y quanto para X > Y, fazendo com que seja qual for o resultado da operação anterior, a próxima instrução já esteja carregada no buffer.

O buffer de pré extração também ajuda a memória cache a carregar antecipadamente os dados que o processador possa precisar. No caso do exemplo anterior, seriam carregados na memória cache tanto o valor de K quanto de Q.

Coprocessador Aritmético Mais Rápido

O coprocessador aritmético do Pentium foi completamente remodelado. Foram alteradas algumas das instruções, que passaram a ser executadas muito mais rapidamente e, como o processador principal, o coprocessador do Pentium também passou a utilizar um pipeline para aumentar a velocidade de execução das instruções.

Somadas com as brutais melhorias na arquitetura, o coprocessador do Pentium tornou-se cerca de 5 vezes mais rápido do que o utilizado no 486, tornando o processador muito mais rápido em aplicativos que demandem um grande número de cálculos.

Arquitetura Superescalar

Mais um aperfeiçoamento do Pentium e um dos principais motivos de seu maior desempenho, é a adoção de uma arquitetura superescalar.

Internamente, o Pentium trabalha como dois processadores de 32 bits distintos (chamados de canaleta U e canaleta V), sendo capaz de processar duas instruções por ciclo de clock (uma em cada processador). Cada processador possui acesso total ao cache, à memória RAM, e aos demais componentes do micro. Foi incluída também, uma unidade de controle, com a função de comandar o funcionamento dos dois processadores e dividir as tarefas entre eles.

Teoricamente, o uso de dois processadores distintos dobraria o desempenho do Pentium, já que ao invés de uma, poderiam ser executadas duas instruções por ciclo de clock. Mas, na prática existem algumas limitações.

Se por exemplo, um programa ordena que o processador some 4 números, X + Y + W + K, o processador poderia no primeiro ciclo de clock usar a canaleta U para somar X e Y e a canaleta V para somar W, mas no segundo ciclo, haveria apenas mais um cálculo para ser executado, o resultado das duas somas. Neste caso, apenas uma das canaletas poderia ser usada; a segunda ficaria ociosa. No final das contas, houve um ganho de 33%, já que ao invés do processador demorar 3 ciclos para executar o cálculo, demorou apenas 2.

Caso a continuação do cálculo dependesse da conclusão do cálculo anterior, como em “(X + Y) x 3”, a segunda canaleta novamente não poderia ser usada, pois o processador teria primeiro que somar X e Y para depois multiplicar o resultado por 3. Neste caso, não haveria ganho algum, pois o processador demoraria os mesmos dois ciclos que seriam necessários com apenas uma canalização.

Em média, a segunda canalização permite um desempenho 30 ou 40% superior ao desempenho obtido com apenas uma canalização. Caso o software seja alterado e otimizado para rodar em um processador com duas canalizações, ordenando as instruções de modo a deixar a segunda canaleta ocupada durante a maior parte do tempo, podemos conseguir mais 10 ou 15% de desempenho, chegando a algo entre 40 e 50%, mas, jamais será possível conseguir o dobro de desempenho.

Isto pode ser notado por exemplo ao recompilar o kernel do Linux. É possível orientar o utilitário a otimizar o código para qualquer processador, de um 486 a um Pentium III. Esta otimização permite que o Linux utilize todos os recursos do processador, obtendo um desempenho bastante superior.

Otimizar um programa para rodar em um processador Pentium, não o torna incompatível com processadores com apenas uma canalização (como o 486), nem torna seu desempenho menor nestes processadores, já que as mesmas instruções serão executadas, apenas em ordem diferente.

Execução Especulativa

Se o processador tivesse que processar uma instrução de tomada de decisão, como em “Se X > 0 então Y = 30, senão Y = 20”, enquanto a primeira canaleta do processador verifica se X é maior ou menor que 0, a segunda ao invés de ficar ociosa, pode executar uma das duas opções seguintes (atribuir o valor 30 ou atribuir o valor 20 a Y). No próximo ciclo, quando a primeira canaleta tivesse terminado de processar sua instrução, teríamos 50% de chance da segunda canaleta ter adivinhado qual seria a instrução seguinte e já tê-la executado. O nome “execução especulativa” foi dado por que a segunda canaleta escolhe aleatoriamente a instrução a executar, entre as duas possíveis.

Acesso mais rápido à Memória

O Pentium é capaz de acessar a memória usando palavras binárias de 64 bits, o dobro do 486, que a acessa a 32 bits. Este recurso permite que sejam lidos 8 bytes por ciclo, ao invés de apenas 4, dobrando a velocidade de acesso à memória. Isto diminuiu bastante o antigo problema de lentidão das memórias, mas apenas provisoriamente, pois logo surgiram processadores Pentium utilizando multiplicadores de clock cada vez mais altos.

Como a maioria das placas para processadores Pentium utiliza módulos de memória de 72 vias, que são módulos de 32 bits, é preciso usa-los em pares. O processador acessa cada dupla como se fosse um único módulo, chegando aos 64 bits necessários.

Mesmo com a capacidade de acessar a memória a 64 bits e sendo composto internamente por dois processadores de 32 bits, o Pentium continua sendo um processador de 32 bits. Estes novos recursos servem apenas para melhorar o desempenho do processador.

Multiprocessamento

Visando o mercado de Workstations (máquinas muito rápidas, destinadas a aplicações pesadas como processamento de imagens 3D ou vídeo) e servidores de rede, a Intel incluiu no Pentium o recurso de multiprocessamento simétrico, que permite o uso de dois processadores na mesma placa mãe. Neste caso, é preciso adquirir uma placa mãe especial, com encaixe para dois processadores e um chipset com suporte ao multiprocessamento.

Como a divisão das tarefas entre os dois processadores não é feita automaticamente pelo chipset, é preciso que o sistema operacional seja capaz de reconhecer os dois processadores e acessá-los individualmente, dividindo as tarefas entre eles da melhor maneira possível.

Caso o sistema operacional não ofereça suporte ao multiprocessamento, como é o caso do Windows 95 e do Windows 98, apenas um dos processadores será usado pelo sistema, ficando o outro inativo. Neste caso, será como se tivéssemos apenas um processador instalado. A maioria dos sistemas operacionais, incluindo o Windows NT, 2000 e XP, Linux e a maioria das versões do Unix suportam multiprocessamento, as excessões mais notáveis ficam por conta do Windows 95,98 e ME.

Apesar de, pela lógica, o desempenho dobrar com dois processadores trabalhando em paralelo, na prática o ganho dificilmente passa de 40 ou 60%, pois dificilmente será possível organizar a execução das tarefas de modo a deixar ambos os processadores ocupados todo o tempo, assim como é difícil manter as duas canaletas do Pentium cheias 100% do tempo.

Até pouco tempo atrás, o recurso de multiprocessamento foi exclusividade dos processadores Intel. Tanto o Pentium, quanto o MMX, o Pentium II e o Pentium III suportam o uso de até dois processadores simultaneamente, enquanto o Xeon suporta o uso de até quatro processadores (8 com a adição de um circuito especial na placa mãe). Com exceção do Athlon MP, todos os demais processadores da AMD e Cyrix não suportam multiprocessamento, o que não chega a ser uma grande desvantagem

para um usuário doméstico, já que fora do ramo dos servidores de alto desempenho, este recurso raramente é usado.

Clock e Overclock

Ao contrário do que muitos pensam, velocidade de operação dos processadores não é fixa, mas sim determinada pela placa mãe.

Na placa mãe temos um pequeno cristal de Quartzo, chamado gerador de clock, que vibra alguns milhões de vezes por segundo, com uma precisão quase absoluta. As vibrações deste cristal são usadas para sincronizar os ciclos da placa mãe, que sabe que a cada vibração do cristal deve gerar um certo número de ciclos de processamento.

É mais ou menos como um farol, que abre e fecha algumas vezes por minuto. Quando o farol está fechado, o trânsito fica parado, voltando a fluir quando a farol abre. Um pulso de clock é justamente a abertura do farol, um "já!" que faz todos os periféricos trabalharem simultaneamente e de forma sincronizada. O funcionamento de todos os periféricos, da placa de vídeo ao disco rígido, é coordenado por este relógio.

O processador não possui um gerador de clock, e por isso trabalha usando o sinal recebido da placa mãe. Num Pentium MMX de 200 MHz, por exemplo, a placa mãe funciona a 66 MHz, e o multiplicador é 3x, o que significa que para cada ciclo da placa mãe, o processador gerará 3 ciclos.

Justamente por estar limitada à frequência indicada pela placa mãe, a frequência do processador não é fixa; pode ser maior ou menor do que o especificado, dependendo de como a placa mãe estiver configurada.

Como as placas mãe atuais, para manter compatibilidade com vários processadores podem operar a várias frequências diferentes, é possível fazer o processador trabalhar mais rápido simplesmente configurando a placa mãe para trabalhar a uma frequência maior. Esta técnica é chamada de Overclock, uma gíria que significa "acima do clock" numa tradução livre.

Um Pentium 120 por exemplo, usa bus 60 MHz e multiplicador de 2x. Se configurássemos a placa mãe para trabalhar a 66 MHz, mantendo o multiplicador em 2x, o processador passaria a trabalhar a 133 MHz. Se a freqüência da placa mãe fosse aumentada para 75 MHz, o processador funcionaria a 150 MHz.

Em muitos casos, o processador também aceita um multiplicador maior. Um AMD K6 de 266 MHz por exemplo, trabalha com a placa mãe funcionando a 66 MHz e usando multiplicador de 4x. Se aumentássemos o multiplicador para 4.5x, mantendo a placa mãe funcionando a 66 MHz, faríamos o processador funcionar a 300 MHz.

A performance de um processador trabalhando em overclock é idêntica à de um processador "normal" funcionando a essa velocidade. Um Pentium 120 overclocado para 133 MHz por exemplo, apresenta exatamente a mesma performance de um Pentium 133 "de verdade".

Quando um fabricante desenvolve um projeto de processador, testa-o a várias frequências diferentes, a fim de determinar sua freqüência ideal de operação. Geralmente, os fabricantes adotam uma certa margem de segurança, vendendo o processador com uma freqüência ligeiramente inferior à freqüência máxima. É justamente esta margem de segurança que permite o overclock; estaríamos então simplesmente fazendo o processador funcionar na sua frequência máxima.

Esta margem muda de fabricante para fabricante e de processador para processador. Por isso, alguns processadores aceitam overclocks maiores que outros. Existem casos de processadores que aceitam trabalhar sem problemas a uma freqüência 50% maior que a original, assim como existem casos de processadores que apresentam instabilidade operando a uma freqüência apenas 10% maior que a original.

Obviamente, o overclock também traz algumas desvantagens. Fazendo o processador trabalhar a uma freqüência maior do que a ideal, podemos ter problemas de travamentos, superaquecimento e sempre haverá alguma diminuição de sua vida útil.

Outros Processadores

Além dos processadores “principais” que vimos até agora, existiram alguns modelos lançados como processadores de baixo custo, alternativa de upgrade para quem tinha um processador antigo e não queria gastar muito.

486DLC e 486SLC

Estes dois processadores foram a tentativa da Cyrix de entrar no mercado de processadores de baixo custo, oferecendo uma opção barata de upgrade para usuários de micros 386 e também uma opção de processador para micros de baixo custo, especialmente micros de menos de 1.000 dólares.

Estes processadores são basicamente processadores 386 (respectivamente o DX e o SX), que incorporam um pequeno cache L1 de apenas 1 KB. O cache não fazia milagres, mas já era o suficiente para aumentar um pouco o desempenho do processador, o que somado ao baixo preço de venda, foi suficiente para vários usuários investirem no upgrade, já que os 486DLC e SLC eram totalmente compatíveis com as placas para micros 386.

Vale lembrar que, como o 386 padrão, estes processadores não possuem coprocessador aritmético, podendo ser acoplados a eles o 387DCL ou o 387SLC, que deviam ser comprados separadamente.

Sob licença da Cyrix, a Texas Instruments desenvolveu versões próprias do 486DLC e SLC, preservando a mesma arquitetura, mas aumentando a quantidade de cache L1 para 8KB.

AMD 5×86

No início, a AMD produzia clones de processadores Intel, utilizando os projetos desenvolvidos pela Intel e pagando royalties em troca. Porém, devido a várias divergências, a aliança acabou sendo desfeita e a AMD passou a batalhar seus próprios projetos de processadores. Apesar de, durante muito tempo, a AMD ter tido que se contentar com um distante segundo lugar, produzindo basicamente processadores de baixo custo, atualmente ela vem competindo diretamente com a Intel também no ramo de processadores de alto desempenho com seu Athlon, conseguindo na maioria das vezes manter preços mais baixos que a concorrente.

Mas, voltando à nossa aula de história, depois que a Intel lançou o 486DX4-100, abandonou o desenvolvimento de processadores 486 para se dedicar somente ao desenvolvimento do Pentium. Com a intenção de apresentar um processador que possuísse um desempenho semelhante a um Pentium low end (os modelos mais lentos e baratos), mas que ao mesmo tempo tivesse um preço competitivo, a AMD continuou o desenvolvimento do seu processador 486, lançando uma versão de 120 MHz (que opera usando barramento de 40 MHz e multiplicador de 3x),e logo em seguida também uma versão de 133 MHz.

Por questões de Marketing, a AMD batizou este 486 de 133 MHz de “AMD 5×86” o que confundiu alguns usuários, que pensaram tratar-se de um processador semelhante ao Pentium. O AMD 5×86 utiliza placas mãe para 486, necessita apenas que a placa seja capaz de sinalizar o multiplicador de 4x. O clock fica em 33 MHz, totalizando seus 133 MHz.

Como o AMD 5×86 não passa de um 486 funcionando a 133 MHz, seu desempenho é pouco menos de 33% superior a um 486DX4-100, sendo mais ou menos equivalente ao de um Pentium de 75 MHz. Aliás, outra medida de marketing tomada pela AMD na época, foi criar um índice Pr, ou “Pentium Rating”, comparando o desempenho do 5×86 ao do Pentium. O 5×86 de 133 MHz recebeu o índice Pr 75, indicando possuir um desempenho semelhante ao apresentado por um Pentium de 75 MHz.

A AMD conseguiu fazer um razoável sucesso com este processador, já que além de ser sido muito usado em micros de baixo custo, o 5×86 passou a ser uma alternativa barata de upgrade para usuários de micros 486 com processadores mais lentos.

Cyrix Cx5x86

Além de desenvolver projetos de processadores 486, que foram fabricados pela Texas Instruments, a Cyrix lançou um processador que mistura recursos do 486 e do Pentium, oferecendo um desempenho bastante superior a um 486 padrão.

Este processador foi batizado como Cx5x86, e apresenta um cache L1 de 16 KB, além de algumas outras melhorias que tornam seu desempenho cerca de 35% superior ao de um 486 do mesmo clock. A versão de 100 MHz do Cx5x86 possui um desempenho equivalente ao 5×86 de 133 MHz da AMD e ao Pentium 75, enquanto a versão de 120 MHz rivaliza em desempenho com um Pentium 90.

Como o 5×86 da AMD, Cx5x86 é totalmente compatível com as placas mãe para 486, bastando configurar a placa com multiplicador de 3x e bus de 33 MHz para instalar a versão de 100 MHz e, 3x 40 MHz para utilizar a versão de 120 MHz.

AMD K5

Depois de muitos atrasos, a AMD finalmente conseguiu lançar um processador que pudesse concorrer diretamente com o Pentium. O K5, porém, não chegou a tornar-se muito popular devido ao seu lançamento atrasado. Quando finalmente saíram as versões Pr 120 e Pr 133 do K5, a Intel já havia lançado as versões de 166 e 200 MHz do Pentium, ficando difícil a concorrência. Ao invés de simplesmente tentar copiar o projeto da Intel, a AMD optou por desenvolver um processador completamente novo, tecnicamente superior ao Pentium.

O K5 também utiliza uma arquitetura superescalar, mas ao invés de duas, possui quatro canalizações. O cache L1 também foi ampliado, passando a ser de 24 KB, dividido em dois blocos, um de 16 KB para instruções e outro de 8 KB para dados.

O coprocessador aritmético porém não foi muito melhorado, apresentando um desempenho quase 50% inferior ao apresentado pelo coprocessador do Pentium, devido principalmente à ausência de Pipeline. Este acabou sendo o calcanhar de Aquiles do K5, que a AMD sempre fez o possível para tentar esconder. Mas, como na maioria das aplicações o K5 era bem mais rápido que o Pentium, a AMD optou novamente por vender seu processador segundo um índice Pr, que compara seu desempenho com o dos processadores Pentium:

Processador	Frequência Real de Operação
K5-Pr 120	90 MHz (1.5x 60 MHz)
K5-Pr 133	100 MHz (1.5x 66 MHz)
K5-Pr 166	116 MHz (1.75x 66 MHz)

Pentium Overdrive

Como fez com os antigos 386 SX, a Intel lançou (ou pelo menos tentou, pois este processador nunca chegou a ser muito vendido) também um Pentium “low cost”. Este processador, apesar de internamente ter um funcionamento idêntico a um Pentium, utiliza placas mãe para processadores 486, sendo por isso chamando de Overdrive.

A Intel lançou o Overdrive em versões de 63 MHz (25 MHz x 2.5) e 83 MHz (33 MHz x 2.5) mas, por utilizarem placas de 486, que operam a frequências muito mais baixas e acessam a memória a apenas 32 bits, estes processadores perdem feio em performance se comparados com um Pentium “de verdade”. O Overdrive de 63 MHz apresenta performance idêntica ao 486DX4-100, enquanto o de 83 MHz empata com o 5×86 de 133 MHz da AMD.

Além da baixa performance, o Overdrive era extremamente caro (por isso usei o low cost entre aspas no parágrafo anterior :-), e acabou sendo uma péssima opção de compra. Em termos de custo-benefício, o 5×86 da AMD foi uma opção muito melhor.

Mesmo após este primeiro fracasso, a Intel continuou tentando lançar sucessivamente vários processadores Overdrive, entre eles uma versão do MMX que funciona em placas soquete 7 antigas e uma versão do Pentium II que funciona em placas mãe para Pentium Pro. Apesar da propaganda feita por alguns “especialistas” nenhum destes modelos de Overdrive foi uma opção de compra que sequer

merecesse ser considerada, pois devido à baixa procura e consequentemente à baixa produção, sempre custaram muito mais caro do que processadores equivalentes.

A única vantagem de se utilizar um processador Overdrive seria continuar utilizando a mesma placa mãe, trocando apenas o processador. Porém, na maioria vezes, por causa da baixa produção o processador Overdrive custa mais caro que um processador normal mais uma placa mãe para ele, tornando-se uma opção lusitana de upgrade.

É bem possível que no futuro sejam lançados outros modelos de processadores Overdrive, mas não espere ver boas opções. Adianta muito pouco trocar apenas o processador, mantendo a mesma velha e ultrapassada placa mãe, a mesma quantidade de memória de RAM, o mesmo velho e lento disco rígido etc. e vale menos à pena ainda pagar caro por isso. Seria como comprar uma Brasília e trocar o motor, você vai continuar com o mesmo carro velho. Na maioria das vezes, é mais sensato gastar em mais memória RAM, num disco rígido maior e mais rápido, ou numa placa 3D, caso o micro seja utilizado para jogos.

Um tal de 186

O 8086 foi o pai do 8088, que equipou os primeiros PCs. Depois vieram os processadores 286, que também conhecemos bem. Mas, entre as duas gerações, existiu um modelo intermediário, pouco conhecido, mas igualmente importante. O 80186 é uma evolução do 8086, que trouxe algumas instruções novas e um sistema de tolerância à falhas. Apesar de não ter sido usado como processador em micros PC, o 80186 tornou-se um componente bastante popular, sendo usado em controladores de HDs, controladores de interrupção entre vários outros periféricos. Nestas aplicações, o 80186 continua em uso até hoje.