Você Sabe o que é Pipeline???

Microprocessadores é um assunto interessante a ser tratado dentro do mundo da informática sendo, com certeza, um dos assuntos que geram mais espanto devido à rápida evolução obtida nos últimos anos. É incrível a velocidade com que eles evoluem e como as empresas fabricantes de processadores acham formas de torná-los cada vez mais rápidos. Segundo Steve Jobs, um dos grandes nomes que a IBM já teve, os processadores tendem a duplicar sua capacidade a cada 18 meses, e esse crescimento geométrico vale até 2020, isto é, temos 20 anos de evoluções pela frente, e 20 anos de espanto! Eu me lembro bem que em 1988 ganhei o primeiro computador de meu pai; um 8088, 5 Mhz, com uma capacidade de endereçamento direto de 1Mb enquanto os processadores concorrentes tinha um acesso de 64Kb numa velocidade de 2Mhz até 4Mhz (o 8088 já existia em 1980 e nessa época o Brasil era mais fechado para produtos estrangeiros). Meu pai pagou a bagatela de 1500 dólares e, lógico, minha mãe ficou mais de 1 mês sem falar com ele. Hoje lembramos disto dando boas risadas. De 1988 até 2000 são 12 anos. Tendo em vista que processadores para PC hoje já estão na casa do GHz, é uma evolução de 1000Mhz, sendo que a cada ano os processadores evoluíram 84Mhz (em nível nacional). Claro, isto não diz o que aconteceu de fato, porque o que realmente impulsionou essa evolução foi a introdução do conceito de Pipeline nos processadores.

ENTÃO O QUE VEM A SER O TAL DO PIPELINE???

Esta dúvida existe na cabeça de muitos 'informáticos', pois, sempre que ligamos o PC aparece na janelinha do setup da máquina o nome Pipeline, mas muitos não sabem sequer do que se trata...

Na verdade é muito simples. De um modo grosseiro, é a capacidade que o processador tem de fazer o processamento através de fases, tornando-se, assim, muito mais otimizado e rápido. Explico: Imagine uma linha de montagem de carros, onde o carro passa por diversas fases, de funilaria, peças, etc. Essa produção funciona em paralelo para diferentes tipos de carro. Essa é a idéia básica colocada no pipeline.

Vamos supor que temos um grupo de técnicos montando, um único carro, e temos um segundo grupo de técnicos montando carros em uma linha de produção. Podemos até ter os dois grupos terminando a produção do primeiro carro ao mesmo tempo. Já para o segundo carro, enquanto o primeiro grupo de técnicos, iria começar, na linha de produção, já estaria em fase final, e assim sucessivamente. É nisso que consiste o pipeline, colocar as instruções em uma linha de produção de modo que torne o processamento mais rápido.

Para se medir a velocidade da linha de produção, usamos carros feitos por hora, e no pipeline é a mesma coisa, só que a medida é um pouco diferente. Para medirmos usamos a seguinte fórmula:

Tempo por instrução sem pipeline

Número de estágios do pipeline

Antes de continuar devo abrir parênteses aqui para explicar os tipos de arquiteturas de processadores para que não haja engano. Existem dois tipos de arquiteturas básicas, o RISC e o CISC. O RISC quer dizer "Reduced Instruction Set Computer" e o CISC quer dizer "Complex Instruction Set Computer". Dentro disso, a arquitetura que apareceu primeiro foi o CISC, em 1964 com a IBM, e o conceito do CISC era: "Tudo que é realizado em Hardware é rápido", porém, em 1980, elaboraram o RISC, que tinha um conceito um pouco diferente: "Faça o caso comum (98% das instruções executadas) ser rápido". A idéia era a seguinte: O que é mais rápido? Ir de Fortaleza ao porto de Santos de navio ou ir de avião até São Paulo e lá pegar uma bicicleta e pedalar até Santos. É nisso que se apoiavam, não importa que 2% das instruções não seja processada rapidamente, se tratarmos o caso mais genérico (98% das instruções) com certeza deve haver um ganho de performance. Hoje as duas arquiteturas brigam no mercado tendo como exemplo de processadores: RISC com o PowerPC, HP PA, Alpha, MIPS e CISC com a maioria dos processadores Intel. Como a arquitetura RISC é mais simples de se trabalhar, vou usar o RISC-DLX como exemplo, pois é um processador de fácil entendimento e muito mais didático. Isso evita diagramas muito grandes e complexos.

Aqui está o desenho de um processador sem pipeline baseado no DLX.

Podemos analisar os principais itens de um processador, como o PC (Program Counter) e o ULA (Unidade Lógica Aritmética), e todo caminho que a instrução percorre dentro do processador. Mas este exemplo apresenta muitos problemas, pois sem o pipeline ele é lento já que trata uma instrução de cada vez. Então esse processador não serviria para um computador pessoal, mas serviria para uma máquina de lavar ou outro eletrodoméstico que não faz nada científico. Porém, não vou entrar nesses detalhes, pois teríamos uma outra matéria.

Desde a entrada da linha de Comando até o final, temos um Clock! Quando a instrução sai, outra instrução entra no PC e assim sucessivamente. Sendo assim, temos sempre uma situação estável. Na entrada temos uma situação estável e no final temos outra situação estável. Portanto, se tentarmos pegar um dado que passa no ULA, não teremos uma situação estável. É aí que temos um problema. E se nós dividíssemos o processador como numa linha de produção???

Agora o desenho do mesmo processador com Pipeline.

A divisão ficaria assim, com os Registradores Especiais posicionados dessa forma. Assim, teremos um Clock em cada espaço do processador, só que esse Clock seria muito mais rápido por ter um caminho muito menor e muito menos ação deveria ser tomada pelo processador e em cada Registrador teríamos uma situação estável. Ele processa muito mais informações justamente pelos Registradores, pois quando PC lê uma instrução e joga no processador, e essa informação chega no primeiro Registrador, o processador já pode estar lendo a segunda informação pronta para jogar de novo no Pipeline, e esse trabalho é totalmente sincronizado, sempre na velocidade do passo mais lento. Entretanto, em estudos feitos, 30% dos códigos de programação são desvios, e isso causa perda de performance ao Pipeline, e em vista disso criaram-se soluções para esses problemas. A primeira solução encontrada foi interpretar o que o código quer fazer. Por exemplo: num loop, onde os processadores teoricamente iriam inserir "bolhas" (é quando o processador insere espaço em branco em um registrador e essa ação se denomina "bubble", ou seja, "bolha") em cada retorno ao topo do loop, ele deveria saber quando existe o loop e o compilador deveria colocar em código simples o loop um embaixo do outro. Por exemplo:

```
for(x=0;x<=3;x++)
{
a=b+c;
d=c-e;
}
```

O compilador se encarregaria de transformar esse código, ao ler o "*for*", e colocar o código desta forma.

```
1  a=b+c;
 d=c-e;
2  a=b+c;
 d=c-e;
3  a=b+c;
 d=c-e;
```

Como o "*for*" pede para fazer isso 3 vezes, o compilador coloca o código a ser executado um embaixo do outro, eliminando, assim, a leitura do "*for*", evitando que, quando se chegasse ao final do loop, o processador tenha que inserir uma bolha, pois teria que voltar ao topo do loop.

Outro exemplo de problema seria o desvio incondicional, no caso de um retorno de programa. A solução básica é pressupor que o desvio vai ser falso e continuar com o código corrente mas, se for verdadeiro, ele terá que limpar os registradores e retornar o valor original antes de terminar a execução. Esse trabalho do processador de limpar os registradores denomina-se squashing. Isso requer alguma peça que armazene esses dados originais.

Ainda temos o desvio condicional, onde ele trabalha da mesma forma que o desvio incondicional só que o processador pressupõe que o desvio vai ser verdadeiro. Caso não seja, executa o squashing e continua o código.

Mesmo encontrando essas soluções, estudos ainda mostram que os processadores têm perda de 40% da performance em várias operações efetuadas, portanto, foram criadas diversas arquiteturas para que se resolva esse problema do Pipeline.

Os Pipelines, hoje, levam muitas estruturas diferentes, do tipo Superescalar, Dinâmico, Out-of-Order, entre outros, que vêm solucionar problemas nos processadores com Pipeline, tentando minimizar o uso de bolhas nos processadores, assim maximizando o uso do processador. Ao mesmo tempo em que os processadores ficam mais sofisticados eles têm diminuído de tamanho através do uso de tecnologias de fabricação mais precisas. Isto leva a um problema de dissipação de calor. Não vou entrar em detalhes de como funciona ou como se resolvem problemas de Engenharia. Vou apenas tentar dar a idéia básica do que existe hoje no mercado e como essas tecnologias funcionam.

O SUPERPIPELINING:

Consiste em se colocar um grande número de estágios, no caso, sendo mais que 6 estágios.

Vantagens: Maior número de instruções sendo processadas ao mesmo tempo e maior freqüência de Clock.

Desvantagens: Aumenta a complexidade, dependências e desvios.

O PIPELINE SUPERESCALAR:

Consiste em se aumentar o número de pipelines, ao invés de 1, teríamos 2 ou 3 pipelines em paralelo.

Vantagens: Paralelismo real, com 2 ou mais instruções sendo processadas em paralelo, com melhora significativa de performance.

Desvantagens: Necessidade do código ser preparado, aumento de complexidade e problemas de dependências e desvios.

PREVISÃO DE DESVIOS:

Esse pipeline procura "adivinhar" qual será o comportamento de um desvio condicional.

Vantagens: Diminuição do número de vezes em que é necessário inserir bolhas no pipeline devido a um desvio.

Desvantagens: Complexidade.

PIPELINE ESPECULATIVO:

Ele inicia o processamento dos 2 caminhos possíveis após um desvio.

Vantagens: Elimina a perda de tempo por seguir a seqüência da instrução errada.

Desvantagens: Altamente complexo pois necessita dois PCs.

PIPELINE DINÂMICO:

Consiste em várias unidades de processamento em paralelo. O processador executa instruções de tipos diferentes em paralelo e em ordem invertida. Seria mais ou menos o seguinte, o processador teria um paralelo especializado em instruções de Ponto Fixo (Soma e Subtração), outro paralelo para Ponto Flutuante (senos e cosenos) e, às vezes, um terceiro paralelo para Multimídia (Som, Vídeo).

Vantagens: Instruções mais lentas não atrasam o processamento das instruções seguintes.

Desvantagens: Complexidade e dificuldade de testar o processador.

OUT-OF-ORDER:

A Execução Fora-De-Ordem faz o processador reordenar as instruções para melhor aproveitar o pipeline e evitar a inserção de bolhas. Ele necessita uma unidade de reordenação de instruções após executá-las. Pode existir em todos os tipos de pipeline mas é obrigatório em um pipeline dinâmico.

Vantagens: Performance otimizada e menos bolhas.

Desvantagens: Complexidade.

Esses são os pipelines mais utilizados e a maioria dos processadores utilizam várias destas técnicas ao mesmo tempo. Existem outros tipos de Pipeline, mas raramente são utilizados. Existem ainda processadores muito rápidos que não utilizam pipeline, mas isso já é outra história.

CURIOSIDADES

O Primeiro processador com pipeline foi o Stretch IBM 7030. Ele é uma evolução do IBM 704, porém 100 vezes mais rápido. O processador RISC foi pensado pela primeira vez com pipeline, em 1980.

O primeiro processador com SuperPipelining foi o MIPS R4000 e foi também o primeiro com arquitetura 64Bits. Os processadores Alpha (21064 foi o primeiro) são totalmente baseados nessa arquitetura e chegam a ser incrivelmente mais velozes que muitos processadores concorrentes.