**CAPÍTULO 4 – PATTERSON**

4.1 – Introdução

* A medição do desempenho é difícil para os computadores atuais.
* Existem limitações associadas a medição.

Definindo desempenho

* Podemos definir desempenho de várias maneiras (de acordo, dependente de algum fator)
* Tempo de resposta é o tempo que o computador leva para cumprir uma tarefa.
* Preocupação no aumento da vazão.
* Mudar o tempo de execução muda a vazão em muitos casos da vida real.
* Desempenho = 1/Tempo de execução
* N = Desempenho de X/Desempenho de Y
* Tempo de execução de Y/Tempo de execução de X = n

Medindo o desempenho

* O tempo é a medida de desempenho dos computadores.
* Tempo de execução é diferente de tempo de CPU. Tempo de CPU é o tempo de execução menos as tarefas de E/S e a execução de outros programas.
* Tempo de CPU do usuário é o tempo de CPU gasto efetivamente em um programa.
* Tempo de CPU do sistema é o tempo que o sistema operacional gasta realizando tarefas a pedido do programa.
* Tempo de CPU = Ciclos de clock para um programa X Duração do Ciclo de Clock
* Tempo de CPU = Ciclo de clocks para um programa / Frequência da CPU
* **Ciclos de clock = Instruções para um programa X Tempo médio por instrução.**
* **Tempo de CPU = (Contagem de instruções x CPI)/Velocidade de CLock.**
* **Contagem de instruções x CPI = Ciclos de clock.**
* Tempo de execução mede-se executando o programa.
* Velocidade de clock é distribuído com o computador.
* Ciclos de clock da CPU = o somatório de todos os CPI’s vezes o número de vezes que uma instrução x aparece.

Afetam a equação de desempenho

|  |  |  |
| --- | --- | --- |
| Algoritmo | Contagem de instruções, possivelmente o CPI. | Pode favorecer as instruções mais lentas de algum modo. |
| Linguagem de programação | Contagem de instruções, CPI |  |
| Compilador | Contagem de instruções, CPI | Pois o compilador faz a tradução da linguagem de alto nível para a de nível mais baixo. Uma má escolha nas instruções pode aumentar o CPI. |
| Arquitetura | Tudo. | Afeta as instruções necessárias à uma função, o custo em ciclo para cada função e a velocidade do processador. |

4.3 – Avaliando o desempenho

* Melhor benchmark são as aplicações reais.
* Diferentes classes de computadores exigem diferentes benchmarks.
* Relatório de benchmark deve ser reprodutível.

Comparando e resumindo o desempenho

* Tempo de execução total é consistente para mais de um programa.

4.4 – Vida real

* Para determinado conjunto de instruções o aumento do desempenho pode vir: Aumento da velocidade de clock, Melhorias na organização do computador. Aprimoramento do compilador.

4.5 – Falácias e armadilhas

* Armadilha: Esperar a melhoria de um aspecto de um computador para aumentar o desempenho por uma quantidade proporcional ao tamanho da melhoria.
* Lei de Amdahl: Tempo de execução após melhoria = (tempo afetado pela melhoria/quantidade de melhroia + tempo não afetado)
* Armadilha: Usar um subconjunto de equação de desemepnho como métrica.
* MIPS = Contador de instrução/Tempo de Execuçãox10^6
* MIPS não leva em conta a capacidade das instruções.
* MIPS varia por programa num mesmo computador.
* O MIPS pode variar inversamente com o desempenho.

4.6 – Comentários finais

* Projetar apenas pelo desempenho é irrealista.
* Alguns Servidores = Alto desempenho
* Embarcados = Baixo custo
* Desktops = Custo/Desempenho
* Tempo de execução é a única medida válida e infalível de desempenho.

**CAPÍTULO 5 – PATTERSON**

5.1 – INTRODUÇÃO

* Compilador e conjunto de instruções determinam a contagem de instruções.
* O tempo de clock e o CPI são determinados pela implementação do processador.

Uma implementação básica do MIPS

* O conjunto de instruções afeta a implementação.

Uma sinopse da implementação.

* Todas as instruções começam de modo parecido: Buscam instrução da memória pelo PC e acessa registradores.
* Todas as classes exceto o jump usam a ULA.
* Aspectos importantes: Cada unidade pode receber dados de diferentes pontos. As unidades precisam ser controladas de acordo com o tipo de instrução.
* Unidade de controle recebe a instrução como entrada e ativa os sinais necessários para ativas as unidades operacionais necessárias.

5.2 – Convenções lógicas do projeto

* Unidades funcionais do MIPS: Elementos que operam nos valores dos dados, elementos que contém estado.
* Elementos que operam nos valores de dados são combinacionais, suas saídas dependem apenas das entradas. A ULA é um exemplo.
* Elementos que contém um estado, tem um armazenamento interno. Memórias e registradores. Entrada para dados e clock. Sequenciais, pois as saídas dependem da entrada e do estado anterior.

Metodologia de clocking

* Método usado para determinar quando os dados são válidos e estáveis em relação ao clock.
* Sincronização acionada por transição: Um esquema de clocking em que todas as transições de estado ocorrem numa transição de clock.

5.3 – Construindo um caminho de dados

* Elementos do caminho de dados: Unidade funcional usada para operar sobre dados ou contê-los. No MIPS são memórias (de dados e instrução), banco de registradores a ULA e somadores.
* Memória de instruções
* Contador de Programa
* Somador, para incrementar o PC.
* Banco de registradores, estrutura onde qualquer registrador pode ser lido ou escrito especificando-se o número do registrador. 4 entradas e 2 saídas.
* Instruções R tem 3 operandos, ler 2 registradores e escreve 1.
* Para cada leitura do BR precisamos de uma entrada que especifique o registrador e uma saída para os dados. Para a escrita também.
* Unidade para estender o sinal, pois só temos 16 bits para endereços nas instruções I.
* Instruções BEQ são relativas ao PC, o offset é somado ao PC.
* Offset precisa ser deslocado 2 bits para representar o offset de uma word.
* Instrução jump substitui os 28 bits menos significativos do PC pelos 26 bits da instrução deslocado 2 bits.

Cirando um caminho de dados simples

* O caminho de dados mais simples executa todas as instruções em um ciclo de clock.
* Isso significa que nenhum recurso do caminho de dados pode ser executado mais de uma vez por instrução.
* Por isso memória de dados é separada de memória de instruções.

5.4 – Um esquema de implementação simples

* ULA possui 4 entradas de controle das quais apenas 6 combinações são usadas. AND, OR, add, sub, slt e nor.

Projetando a unidade de controle

* Movimentação no caminho de dados das instruções.

Finalizando o controle

* Ciclo de clock único se torna atrativo para um número pequeno de instruções.

5.6 – Uma implementação multiciclo

* Cada etapa leva um ciclo.
* Permite o uso de uma unidade funcional mais de uma vez.
* Cada instrução tem um número diferente de ciclos de clock.
* Uma única memória é usada.
* Existe uma única ULA.
* Um ou mais registradores são adicionados depois de cada unidade funcional para conter o valor que será usado no clock subsequente.
* Posição dos registradores adicionais é determinada: Que unidades combinacionais cabem em um clico e que dados serão necessários posteriormente.
* Cada ciclo de clock pode acomodar no máximo uma: Acesso a memória, acesso ao banco de dados, operação de ULA.
* Adicionar os registradores de Instrução e Dados. Incluir os registradores A e B para conter operandos. Registrador SaidaUla.
* Todos os registradores exceto o IR não precisam de controle de escrita, pois contém dados entre pares de ciclos. IR precisa manter o dado até o fim da instrução.
* Barramento compartilhado. Conjunto de linhas que conectam várias unidades funcionais.

Dividindo a execução das instruções em ciclos de clock.

* Tentar manter a quantidade de trabalho por ciclo igual.
* Etapas em um ciclo ocorrem em paralelo.
* 1 Etapa – Busca de Instrução

Busca instrução da memória – Envia o PC e escreve a instrução no RI.

Calcula o endereço da próxima instrução – Incrementa o PC

* 2 Etapa – Decodificação e busca de registradores

Ler os registradores rs e rt – Armazena-os em A e B

Calcula o endereço de destino na ULA

* 3 Etapa – Execução, cálculo de endereço ou conclusão do desvio.

Instrução I - Soma A + O offset para calcular o endereço de memória.

Instrução R - Realiza uma operação lógica/aritmética sobre A e B

Instrução I de desvio - Testa o desvio e salva o valor do PC

Instrução J - Atualiza o PC com o endereço do JUMP

* 4 Etapa – Etapa de acesso a memória ou conclusão de instrução R

Load ou store acessa a memória. Endereço foi calculado anteriormente e esta no registrador saída ULA.

Instrução R escreve o resultado. SaídaUla vai para o registrador de destino.

* 5 Etapa de Conclusão da leitura da memória

Load guarda o valor lido da memória.

Definindo o controle

* Máquinas de estado fintias, conjunto de estados e diretrizes sobre como mudar de estado. AS diretrizes funcionam em função do próximo estado e mapeiam o estado atual para definir o próximo estado.
* Microprogrmação, representação simbólica do controle através de microinstruções.

5. 6 - Exceções

* Exceção – Um evento não programado que interrompe a execução do programa.
* Interrupção – Uma exceção que vem de fora do processador.
* Exceções foram criadas para tratar de eventos inesperados e requisições de sérvios de IO.
* Registrador Cause indica o motivo de uma exceção.
* Manter a lógica do controle pequena e rápida.

5.7 - Microprogramação: simplificando o projeto.

* Controlador microprogramado – Um método de especificar controle que usa microcódigo em vez de uma representação de estados finitos.
* Controle hardwired – Uma implementação de controçe da máquina de estados finitos normalmente usa arrays lógicos programáveis ou grupos deles.
* Microcódigo – O conjunto de microinstruções que controlam um processador.
* Superescalar – Uma técnica de pipelining avançada que permite executar mais de uma instrução por ciclo de clock.
* Microinstrução – Uma representação do controle usando instruções de baixo nível, cada uma ativando um conjunto de sinais de controle em um determinado ciclo de clock, bem como especificando que microinstrução executar em seguida.
* Microoperação – As instruções RIS diretamente executadas pelo hardware nas recentes implementações do Pentium.
* Microcódigo armazenado em ROM.
* Microcódigos para a vida toda não são muito efetivos.

**CAPÍTULO 10 – WEBER**

Organização Do Neander

10.1 - Elementos Necessários

* Um registrador de 8 bits para ser o acumulador
* 1 registrador de 8 bits para ser o pc
* Dois flip flops para serem os códigos de condição n e z
* Uma memória de 256 posições com 8 bits cada.

10.2 – Fluxo de dados

* Incremento de PC pode ser feito a qualquer momento após sua transferência para REM.
* O incremento pode ser feito com outras operações. Registrador contador é o utilizado pelo NEander, mas pode haver somador próprio e até o uso da UAL.
* Cada registrador tem um sinal de carga indicando quando será atualizada a entrada.
* A UAL pode realizar 5 operações: Passar direto um valor, and, or, not e adição.
* As estradas da UAL foram escolhidas de forma a simplificar as transferências.
* REM é o único registrador que recebe dados de duas fontes distintas, por isso usa-se um multiplexador para determinar donde virá o endereço a ser armazenado nele.

10.3 – Sinais de Controle

* São necessários 8 tempos. Os 3 primeiros sempre compõe a fase de busca.
* Uma implementação simples do sistema de controle utiliza um registrador contador que gera os tempos e uma lógica combinacional para combinar os tempos(recebe-os juntamente com a decodificação da instrução, os sinais de estado) com os devidos sinais de controle.
* Circuito temporizador pode ser: 3 flip flops e um decodificador ou um registrador de deslocamento.

10.4 – O sistema de memória

* Existe uma hierarquia de memória atualmente.
* Registradores estão no nível mais alto: Rápidas, pequenas e caras.
* Cache estão no segundo nível: Caches interiores L1 e exteriores L2.
* Memória principal: A RAM
* Memórias secundárias: Magnéticas e mais lentas.
* Trazer as informações mais utilizadas para as memórias mais rápidas.
* O software também contribui para o uso de memórias mais rápidas.
* ROM é só de leitura e não volátil. Evoluiu de PROM e EPROM. Nos computadores atuais as EEPROMS sobrepõem uma parte da memória principal a fim de armazenar programas necessários na inicialização do computador.
* RAMS são voláteis. SRAM são flip flops e DRAM são cargas elétricasem um capacitor, refreshed continuamente.
* Tempo de ciclo de memória é o tempo de uma leitura ou escrita na memória a contar do envio do sinal pela unidade de controle.
* Na RAM dinâmica o tempo de ciclo é a soma dos tempos de acesso e refresh.
* Durante um refresh um dado recém lido já pode ser utilizado pelo processador.
* Seleção linear, são memórias que usam apenas uma linha de seleção para cada palavra de memória.
* Seleção matricial mais econômica.
* Quando usa-se bancos o endereço do REM é dividido para indicar H – o Banco e L-o endereço.

10.5 – O contador de programa

* Utilização de flip flops JK com sensitivade a borda alta.