**LISTA 02**

*Superescalaridade*

1. **O QUE SIGNIFICA O TERMO SUPERESCALAR? QUAL A SUA ORIGEM?**

O termo tem haver com um computador que poder escalonar instruções de forma a aproveitar o paralelismo a nível de instrução que um determinado código pode oferecer.

1. **O QUE É PARALELISMO DE INSTRUÇÃO (ILP)?**

É um tipo de paralelismo que pode ser explorado e que se baseia na possibilidade de executar várias instruções ao mesmo tempo, por isso o nome de Paralelismo a Nível de instrução.

1. **EXPLIQUE O DESPACHO DE INSTRUÇÕES.**

Despacho de instruções é o fornecimento de instruções para unidades funcionais. Esse dispacho pode ser em ordem ou fora de ordem.

1. **COMENTE LIMITAÇÕES DA TÉCNICA DE SUPERESCALARIDADE.**

Computadores que empregam as técnicas de superescalaridade estão fadados a ficarem presos com o mesmo hardware. Isso significa que uma nova versão de um processador indica uma nova versão do hardware de escalonamento, ou seja, o processador se tornará dependente do hardware. Por isso os computadores superescalares são geralmente usados para aplicações embarcadas.

1. **QUAL FOI O PRIMEIRO COMPUTADOR SUPERESCALAR?**

CDC 6600

1. **DENTRE OS PROCESSADORES INTEL UTILIZADOS NOS PCS, QUAL O PRIMEIRO A APRESENTAR RECURSO DE SUPERESCALARIDADE?**

O Pentium 5.

1. **O QUE É VLIW/EPIC?**

VLIW é uma abordagem do despacho múltiplo de instruções dito estático. Várias instruções são executadas em paralelos em diferentes unidades funcionais, mas a decisão sobre quais instruções executarão em que ciclo de clock é feita em tempo de compilação. O EPIC é a implementação VLIW desenvolvida pela intel.

*Limites do Paralelismo*

**1. SE O FRAGMENTO DE CÓDIGO ABAIXO FOSSE EXECUTADO EM UM PROCESSADOR SUPERESCALAR COM UM NÚMERO INFINITO DE UNIDADES DE EXECUÇÃO (COM LATÊNCIA DE UM CICLO PARA TODAS OPERAÇÕES) QUANTOS CICLOS SERIAM NECESSÁRIOS PARA EXECUTÁ-LO? EM OUTRAS PALAVRAS, QUE LIMITAÇÕES SOBRE O TEMPO DE EXECUÇÃO SÃO IMPOSTAS PELAS DEPENDÊNCIAS EXISTENTES NO FRAGMENTO DE CÓDIGO?**

1 - LD r7, (r8)

2 - SUB r10, r11, r12

3 - MUL r13,r7,r11

4 - ST (r9), r13

5 - ADD r13, r2, r1

6 - LD r5, (r6)

7 - SUB r3, r3, r5

As dependências são:

* 1 e 3 em R7 (read-write)
* 3 e 4 em R13 (read-write)
* 3 e 5 em r13 (write-write)
* 4 e 5 em r13 (write-read)
* 6 e 7 em r5 (verdadeira)

|  |  |  |
| --- | --- | --- |
| LD r7, (r8) | SUB r10, r11, r12 | LD r5, (r6) |
| MUL r13,r7,r11 |  | SUB r3, r3, r5 |
| ST (r9), r13 | ADD r13, r2, r1 |  |

3 ciclos, isso se deve as 3 instruções que não podem se executadas em paralelo. 1, 3 e 4, pois cada uma depende do resultado da anterior.

*Execução em ordem*

**2. EM QUANTOS CICLOS O SEGUINTE FRAGMENTO DE CÓDIGO SERIA EXECUTADO EM UM PROCESSADOR SUPERESCALAR (COM EXECUÇÃO EM ORDEM) COM DUAS UNIDADES DE EXECUÇÃO, ONDE TODAS INSTRUÇÕES POSSUEM LATÊNCIA DE UM CICLO E AMBAS AS UNIDADES DE EXECUÇÃO PODEM EXECUTAR QUALQUER INSTRUÇÃO?**

1 - LD r1, (r2)

2 - SUB r4, r5, r6

3 - ADD r3, r1, r7

4 - MUL r8, r3, r3

5 - ST (r11), r4

6 - ST (r12), r8

7 - ADD r15, r14, r13

8 - SUB r10, r15, r10

9 - ST (r9), r10

|  |  |
| --- | --- |
| 1 - LD r1, (r2) | 2 - SUB r4, r5, r6 |
| 3 - ADD r3, r1, r7 |  |
| 4 - MUL r8, r3, r3 | 5 - ST (r11), r4 |
| 6 - ST (r12), r8 | 7 - ADD r15, r14, r13 |
| 8 - SUB r10, r15, r10 |  |
| 9 - ST (r9), r10 |  |

6 ciclos

**3. QUANTOS CICLOS SERIAM NECESSÁRIOS PARA EXECUTAR A SEQÜÊNCIA DE INSTRUÇÕES A SEGUIR EM UM PROCESSADOR SUPERESCALAR (COM EXECUÇÃO EM ORDEM) COM 4 UNIDADES DE EXECUÇÃO, ONDE QUALQUER UNIDADE PODE EXECUTAR QUALQUER OPERAÇÃO, SENDO QUE OPERAÇÕES DO TIPO *LOAD* POSSUEM UMA LATÊNCIA DE 2 CICLOS E TODAS AS DEMAIS POSSUEM LATÊNCIA DE APENAS 1 CICLO?**

1 - ADD r1, r2, r3

2 - SUB r5, r4, r5

3 - LD r4, (r7)

4 - MUL r4, r4, r4

5 - ST (r7), r4

|  |  |  |  |
| --- | --- | --- | --- |
| ADD r1, r2, r3 | SUB r5, r4, r5 | LD r4, (r7) |  |
|  |  | LD r4, (r7) |  |
| MUL r4, r4, r4 |  |  |  |
| ST (r7), r4 |  |  |  |

4 ciclos

**Execução fora de ordem**

**4. EM QUANTOS CICLOS O FRAGMENTO DE CÓDIGO DO PROBLEMA 3 SERIA EXECUTADO EM UM PROCESSADOR SUPERESCALAR (AGORA COM EXECUÇÃO FORA DE ORDEM) COM DUAS UNIDADES DE EXECUÇÃO, ONDE TODAS INSTRUÇÕES POSSUEM LATÊNCIA DE UM CICLO E AMBAS AS UNIDADES DE EXECUÇÃO PODEM EXECUTAR QUALQUER INSTRUÇÃO? ASSUMA QUE A JANELA DE INSTRUÇÕES É GRANDE O SUFICIENTE PARA CONTER TODA A SEQÜÊNCIA DE INSTRUÇÕES E QUE O PROCESSADOR UTILIZA UM MÉTODO "GULOSO"\* PARA A EXECUÇÃO DAS INSTRUÇÕES.**

*\* Método guloso:* as instruções vão sendo escalonadas de acordo com as possibilidades(unidades de execução disponíveis e dependências), sendo que, quando há mais de uma candidata para o escalonamento, a opção é pela primeira instrução que aparece no código. Não existe preocupação em encontrar um escalonamento ótimo de instruções. Entende-se por "ótimo" o escalonamento que possibilitaria a execução correta de todas as instruções no menor número de ciclos.

1 - ADD r1, r2, r3

2 - SUB r5, r4, r5

3 - LD r4, (r7)

4 - MUL r4, r4, r4

5 - ST (r7), r4

|  |  |
| --- | --- |
| SUB r5, r4, r5 | LD r4, (r7) |
| ADD r1, r2, r3 | MUL r4, r4, r4 |
| ST (r7), r4 |  |

3 ciclos

**5. QUANTOS CICLOS SERIAM NECESSÁRIOS PARA EXECUTAR A SEQÜÊNCIA DE INSTRUÇÕES DO EXERCÍCIO 3 EM UM PROCESSADOR SUPERESCALAR (AGORA COM EXECUÇÃO FORA DE ORDEM) COM 4 UNIDADES DE EXECUÇÃO, ONDE QUALQUER UNIDADE PODE EXECUTAR QUALQUER OPERAÇÃO, SENDO QUE OPERAÇÕES DO TIPO *LOAD* POSSUEM UMA LATÊNCIA DE 2 CICLOS E TODAS AS DEMAIS POSSUEM LATÊNCIA DE APENAS 1 CICLO? ASSUMA ESCALONAMENTO GULOSO E QUE A JANELA DE INSTRUÇÕES É GRANDE O BASTANTE PARA CONTER TODO FRAGMENTO DE CÓDIGO.**

|  |  |  |  |
| --- | --- | --- | --- |
| SUB r5, r4, r5 | LD r4, (r7) | ADD r1, r2, r3 |  |
|  | LD r4, (r7) |  |  |
|  | MUL r4, r4, r4 |  |  |
|  | ST (r7), r4 |  |  |

4 ciclos

*Renomeação de Registradores*

**6. QUANTOS REGISTRADORES SÃO NECESSÁRIOS PARA PERMITIR QUE O MECANISMO DE RENOMEAÇÃO DE REGISTRADORES ELIMINE TODAS AS DEPENDÊNCIAS DO TIPO WAR (*WRITE-AFTER-READ*) E WAW (*WRITE-AFTER-WRITE*) DO SEGUINTE CONJUNTO DE INSTRUÇÕES?**

1 - LD r1, (r2)

2 - ADD r3, r4, r1

3 - SUB r4, r5, r6

4 – MUL r7, r4, r8

5 – ASH r8, r9, r10

6 - SUB r11, r8, r12

7 - DIV r12, r13, r14

8 – ST (r15), r12

Dependências WAR:

* 2 e 3 no r4
* 4 e 5 em r8
* 6 e 7 no r12

Dependências WAW:

* Nenhuma

3 registradores.

1. **MOSTRE UMA MANEIRA DE COMO A RENOMEAÇÃO DE REGISTRADORES TRANSFORMARIA O FRAGMENTO DE CÓDIGO DO EXERCÍCIO ANTERIOR. ASSUMA QUE O PROCESSADOR POSSUI REGISTRADORES SUFICIENTES PARA EFETUAR A REQUERIDA RENOMEAÇÃO.**

1 - LD r1, (r2)

2 - ADD r3, r4, r1

3 - SUB RX, r5, r6

4 – MUL r7, r4, r8

5 – ASH RY, r9, r10

6 - SUB r11, r8, r12

7 - DIV RZ, r13, r14

8 – ST (r15), r12

Utilizando os registradores RX, RY e RZ

1. **EM QUANTOS CICLOS A SEQÜÊNCIA ORIGINAL DO PROBLEMA 6 E A REESCRITA NO EXERCÍCIO 7 SERIAM EXECUTADAS EM UM PROCESSADOR SUPERESCALAR (COM EXECUÇÃO FORA DE ORDEM) COM 4 UNIDADES DE EXECUÇÃO, CADA UMA PODENDO EXECUTAR QUALQUER INSTRUÇÃO? ASSUMA QUE A LATÊNCIA DE TODAS INSTRUÇÕES É DE 1 CICLO, UTILIZE ESCALONAMENTO "GULOSO" E QUE A JANELA DE INSTRUÇÕES É CAPAZ DE CONTER TODO CÓDIGO.**

1 - LD r1, (r2)

2 - ADD r3, r4, r1

3 - SUB r4, r5, r6

4 – MUL r7, r4, r8

5 – ASH r8, r9, r10

6 - SUB r11, r8, r12

7 - DIV r12, r13, r14

8 – ST (r15), r12

|  |  |  |  |
| --- | --- | --- | --- |
| 1 - LD r1, (r2) |  |  |  |
| 2 - ADD r3, r4, r1 | 3 - SUB r4, r5, r6 |  |  |
| 4 – MUL r7, r4, r8 |  |  |  |
| 5 – ASH r8, r9, r10 |  |  |  |
| 6 - SUB r11, r8, r12 | 7 - DIV r12, r13, r14 |  |  |
|  | 8 – ST (r15), r12 |  |  |

6 ciclos

1 - LD r1, (r2)

2 - ADD r3, r4, r1

3 - SUB RX, r5, r6

4 – MUL r7, r4, r8

5 – ASH RY, r9, r10

6 - SUB r11, r8, r12

7 - DIV RZ, r13, r14

8 – ST (r15), r12

|  |  |  |  |
| --- | --- | --- | --- |
| 1 - LD r1, (r2) | 3 - SUB RX, r5, r6 | 4 – MUL r7, r4, r8 | 5 – ASH RY, r9,r10 |
| 2 - ADD r3, r4, r1 | 6 - SUB r11, r8, r12 | 7 - DIV RZ, r13, r14 | 8 – ST (r15), r12 |

2 ciclos

*Escalonamento VLIW*

1. **MOSTRE COMO UM COMPILADOR PODERIA ESCALONAR O CÓDIGO DO PROBLEMA 3 PARA A EXECUÇÃO EM UM PROCESSADOR VLIW COM O MESMO NUMERO DE UNIDADES DE EXECUÇÃO E LATÊNCIAS DE INSTRUÇÃO DO EXERCÍCIO ORIGINAL. AO CONTRÁRIO DOS PROBLEMAS DE EXECUÇÃO FORA DE ORDEM, ASSUMA QUE O COMPILADOR EXAMINA TODAS AS POSSIBILIDADES DE ORDENAMENTO DE INSTRUÇÕES PARA ENCONTRAR O MELHOR ESCALONAMENTO POSSÍVEL (ISSO ADVÉM DO FATO DO COMPILADOR PODER UTILIZAR MUITO MAIS RECURSOS E TEMPO PARA ENCONTRAR O MELHOR ESCALONAMENTO DO QUE O USUALMENTE DISPONÍVEL NO HARDWARE DE PROCESSADORES SUPERESCALARES). CERTIFIQUE-SE DE INCLUIR AS INSTRUÇÕES NOP PARA OPERAÇÕES NÃO UTILIZADAS.**

|  |  |  |  |
| --- | --- | --- | --- |
| ADD r1, r2, r3 | SUB r5, r4, r5 | LD r4, (r7) | NOP |
| NOP | NOP | LD r4, (r7) | NOP |
| MUL r4, r4, r4 | NOP | NOP | NOP |
| ST (r7), r4 | NOP | NOP | NOP |

1. **MOSTRE COMO UM COMPILADOR PODERIA ESCALONAR O SEGUINTE PROGRAMA PARA EXECUÇÃO EM UM PROCESSADOR VLIW COM 4 UNIDADES DE EXECUÇÃO, SENDO QUE CADA UMA PODE EXECUTAR QUALQUER TIPO INSTRUÇÃO. INSTRUÇÕES DE *LOAD* POSSUEM UMA LATÊNCIA DE 3 CICLOS E TODAS AS OUTRAS INSTRUÇÕES TÊM LATÊNCIA DE APENAS 1 CICLO.**

Tenha em mente que em um VLIW, o valor antigo de um registrador de destino de uma instrução mantém-se disponível para leitura até que a referida instrução seja completada.

SUB r4, r7, r8

MUL r10, r11, r12

DIV r14, r13, r15

ADD r9, r3, r2

LD r7, (r20)

LD r8, (r21)

LD r11, (r22)

LD r12, (r23)

LD r13, (r24)

LD r15, (r25)

LD r3, (r30)

LD r2, (r31)

ST (r26), r4

ST (r27), r10

ST (r28), r14

ST (r29), r9

|  |  |  |  |
| --- | --- | --- | --- |
| SUB r4, r7, r8 | MUL r10, r11, r12 | LD r8, (r21) | LD r7, (r20) |
| ADD r9, r3, r2 | DIV r14, r13, r15 | LD r8, (r21) | LD r7, (r20) |
| LD r11, (r22) | LD r12, (r23) | LD r8, (r21) | LD r7, (r20) |
| LD r11, (r22) | LD r12, (r23) | LD r13, (r24) | LD r15, (r25) |
| LD r11, (r22) | LD r12, (r23) | LD r13, (r24) | LD r15, (r25) |
| LD r3, (r30) | LD r2, (r31) | LD r13, (r24) | LD r15, (r25) |
| LD r3, (r30) | LD r2, (r31) | ST (r26), r4 | ST (r27), r10 |
| LD r3, (r30) | LD r2, (r31) | ST (r28), r14 | ST (r29), r9 |

*Loop Unrolling*

1. **POR QUE DESENROLAR UM LAÇO FORNECE UM INCREMENTO DE DESEMPENHO?**

Ao copiar o corpo do laço diversas vezes no programa, evita-se o overhead causado por instruções de controle do loop além de diminuir o número de verificações e instruções de desvios associadas aos loops diminuindo assim a possiblidade de hazards de controle. Além disso, ao copiar o loop várias vezes estamos expondo uma quantidade maior de instruções que podem ser exploradas de forma paralela.

1. Mostre como um compilador desenrolaria o seguinte laço infinito 4 vezes. Inclua o código inicial (o código que calcula todos os ponteiros necessários para as operações de cada iteração do laço desenrolado). Assuma que o processador possuem tantos registradores quanto o necessário.

loop:

LD r1, (r2)

LD r3, (r4)

LD r5, (r6)

ADD r1, r1, r3

ADD r1, r1, r5

DIV r1, r1, r7

ST (r0), r1

ADD r2, #4, r2

ADD r4, #4, r4

ADD r6, #4, r6

ADD r0, #4, r0

BR loop

1. Mostre como um compilador poderia escalonar as versões original e modificada ("desenrolada") do laço do exercício anterior para a execução em um processador VLIW com 4 unidades de execução que podem executar qualquer tipo de instrução. Assuma latência de 3 ciclos para instruções LD e 2 ciclos para DIVs e ADDs. Assuma que o atraso do *branch* é grande o suficiente para que todas as operações de uma iteração se completem antes do início da próxima iteração. Como nos outros exercícios de VLIW, assuma que o compilador examine todas as possibilidades de ordenamento para encontrar aquela que permite a execução no menor número de instruções possível. Para o laço modificado, escalone apenas o laço e não o código inicial.
2. **QUAIS AS DIFERENÇAS ENTRE MÁQUINAS SUPERESCALARES E MÁQUINAS VLIW?**

Superescalares são máquinas de despacho mútiplo dinâmico, isso significa que o escalonamento de instruções ocorre em tempo de instruções e é executado por um hardware especializado. Isso significa também que as dependências são resolvidas em tempo de execução. Muito dependente do hardware, próprio para aplicações embarcadas.

As máquinas VLIW implementam o despacho múltiplo estático, assim o escalonamento das instruções ocorre em tempo de compilação e é o próprio compilador o responsável por fazer a tarefa de escalonamento. As dependências são resolvidas durante a compilação. Nessa maquinas várias instruções independentes são codificadas em uma única palavra longa, a posição de cada instrução na palavra indica qual unidade funcional executará aquela instrução. Mais independente do hardware no sentido de que uma alteração no número de unidades funcionais não altera a funcionalidade.

**Hierarquia de memória**

1. **DIFERENCIE MEMÓRIA ROM E RAM QUANTO A FORMA DE ACESSO E VOLATILIDADE DOS DADOS.**

Memórias ROM são memórias não voláteis, ou seja os dados podem ser guardados e relidos após cortar o fornecimento de energia. As RAM são memórias voláteis, ou seja, mantém os dados somente enquanto há fornecimento de energia.

As memórias ROM eram antigamente memórias apenas de leitura, enquanto que as memórias RAM podem ser lidas e escritas de forma “aleatória”.

1. **DIFERENCIE MEMÓRIA DRAM E SRAM QUANTO À TECNOLOGIA E APLICAÇÕES. CLASSIFIQUE QUANTO A: FORMA DE ACESSO, VOLATILIDADE DOS DADOS, VELOCIDADE, CUSTO.**

DRAM, utiliza capacitores e são geralmente aplicadas para formar a memória principal do computador. Tem acesso randômico, com dados voláteis e velocidade reduzida de busca de dados além de um custo menor por bit se comparada a SRAM.

SRAM utiliza flip flops e são geralmente aplicadas para formar as memórias cache. Tem acesso randômico com dados voláteis, mas alta velocidade de busca e preço por bit mais elevado.

1. **DESCREVA AS CARACTERÍSTICAS GERAIS DE UM PROGRAMA QUE APRESENTARIA BAIXA LOCALIDADE TEMPORAL E ESPACIAL COM RELAÇÃO AOS DADOS. FAÇA UM EXEMPLO EM PSEUDOCÓDIGO.**

Programa que apresenta poucos acessos a locais de memórias acessados anteriormente (com poucos loops) e que não acessa locais de memórias contíguos (com muitos jumps e chamadas de funções).

**4. APRESENTE O PSEUDOCÓDIGO DE UM PROGRAMA COM BOA LOCALIDADE TEMPORAL E BAIXA LOCALIDADE ESPACIAL PARA O ACESSO AOS DADOS.**

Um programa com vários loops (acesso de locais repetidos), programa com muitos jumps e chamadas de funções, pois não acessará locais continguos de memória.

**5. É DADA UMA CACHE INICIALMENTE VAZIA. OS SEGUINTES ENDEREÇOS SÃO ACESSADOS EM SEQÜÊNCIA: 2, 3, 11,16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, 11. CONSIDERANDO QUE A CACHE USE MAPEAMENTO DIRETO E 16 BLOCOS DE UM BYTE CADA, INDIQUE CADA REFERÊNCIA NA LISTA COMO UM ACERTO OU FALHA E MOSTRE O CONTEÚDO FINAL DA CACHE.**

16 linhas = 4 bits de índice

1 byte por linha = 0 bits de byte select

A partir do endereço maior 64, supomos que cada endereço seja de 8 bits.

4 bits de tag.

|  |  |  |
| --- | --- | --- |
| IND | TAG | DADO |
| 0000 | 1011 | MEM(02)miss MEM(03)miss MEM(11)miss MEM(13)miss MEM(11)miss MEM(03)miss MEM(04)miss MEM(06)miss MEM(11)miss |
| 0001 | 1011 | MEM(16)miss MEM(21)miss MEM(19)miss MEM(22)miss MEM(27)miss |
| 0010 |  |  |
| 0011 | 0000 | MEM(48) |
| 0100 | 0000 | MEM(64)miss |
| 0101 |  |  |
| 0110 |  |  |
| 0111 |  |  |
| 1000 |  |  |
| 1001 |  |  |
| 1010 |  |  |
| 1011 |  |  |
| 1100 |  |  |
| 1101 |  |  |
| 1110 |  |  |
| 1111 |  |  |

6. Repita o exercício anterior para blocos de 16 bytes.

4 bits de BS

4 bits de indice

Tag ?

**7. EXPLIQUE O FUNCIONAMENTO DO MAPEAMENTO DE CACHE TIPO ASSOCIATIVO POR CONJUNTO (CONJUNTO ASSOCIATIVO).**

Existe um número fixo de locais onde cada bloco pode ser colocado. Cada bloco da memória é mapeado para um único conjunto, determinado pelo índice. Um bloco pode ser colocado em qualquer elemento desse conjunto

**8. O QUE SIGNIFICA DIZER QUE UM PROJETO DE CACHE É MULTINÍVEL? É CORRETO DIZER QUE AS CACHES DE PRIMEIRO NÍVEL PROCURAM DIMINUIR A TAXA DE FALHAS, ENQUANTO AS DE SEGUNDO VISAM A DIMINUIR O TEMPO DE ACERTO?**

São projetos que apresentam níveis hierárquicos de cache de modo que apresentará mais de uma cache disposta em de maneira hierárquica.

**9. QUAL DAS TRÊS POSSÍVEIS FALHAS DE CACHE PODE DIMINUIR SE O TAMANHO DO PROGRAMA FOR REDUZIDO?**

Diretamente a Falha de capacidade. Indiretamente as falhas compulsórias, pois haverá menos blocos para ser tocados e a falha de conflito, pois haverá menos blocos para disputar um mesmo conjunto.

**10. POR QUE A TAXA DE FALHAS DA MEMÓRIA VIRTUAL PRECISA SER MENOR QUE A DA CACHE? COMO ESTE PROBLEMA É ABORDADO PELOS PROJETISTAS?**

Por que o custo da falha de memória virtual (falta de página) é bem maior do que o custo da falha de cache (cache miss). Por custosa significa que o tempo de busca da página faltante é muito maior do que o custo de busca de um bloco que não está na cache. Os projetistas abordam esse problema investindo alto em algoritmos que organizam de forma eficiente a substituição das páginas de forma a evitar descartar páginas que serão necessitadas logo.

**11. O QUE MOTIVA O DESENVOLVIMENTO DE UMA HIERARQUIA DE MEMÓRIAS?**

Com uma hierarquia de memória pode-se dar a ilusão de que o programa possui mais memória do que aquela disponível para ele utilizar. O principal motivo para isso ter acontecido é a discrepância no que diz respeito ao aumento de velocidade dos processadores (que aumentaram bastante) e das memórias (que aumentaram pouco). Implementando uma hierarquia com níveis de velocidade diferentes possibilita ao programador utilizar memórias tão ráidas quanto o processador que por conta da implementação da hierarquia podem parecer conter um tamanho muito maior do que realmente possuem.

**12. CONSIDERANDO UM SISTEMA DE MEMÓRIA VIRTUAL, DIFERENCIE ENDEREÇO FÍSICO DE ENDEREÇO LÓGICO.**

Endereço virtual é o endereço que corresponde a um local no espaço virtual e é traduzido pelo mapeamento de endereços para um endereço físico quando a memória é acessada.

Endereço físico é o endereço na memória principal. Possui um número menor de bits que o endereço virtual.

**13. INVESTIGUE O CONCEITO DE TLB (TRANSLATION LOOKASIDE BUFFER). POR QUE TLB SÃO UTILIZADAS EM HIERARQUIA DE MEMÓRIA? COMO FUNCIONA?**

A TLB é uma cache especial que monitora os mapeamentos de endereços recentemente usados para evitar um acesso direto a tabela de páginas. Desta forma o processador pode acessar a memória virtual à velocidade de cache.

**14. EXPLIQUE AS TÉCNICAS DE ESCRITA EM MEMÓRIA WRITE-BACK E WRITE-THROUGH.**

Em write back os dados são escritos na hierarquia inferior da memória somente quando um bloco do nível superior precisa ser substituído, dessa forma as alterações de um bloco são feitas apenas uma vez na memória de nível inferior da hierarquia. Ideal para níveis inferiores com tempo de acesso muito longo.

Em write through os dados são escritos tanto no nível superior quanto no nível inferior, usa-se um buffer de escrita para o envio de informações para o nível inferior.

**15. APRESENTE DUAS VANTAGENS DA TÉCNICA WRITE-BACK EM RELAÇÃO À WRITE-THROUGH.**

Os dados podem ser escritos e lidos à velocidade de cache, diversas escritas de um bloco só contarão como uma escrita no nível inferior.

16. São dadas a seguir duas funções em C que retornam a soma de todos os elementos da matriz. A única diferença entre as duas é que elas visitam os elementos da matriz em uma ordem diferente. Este fato não parece ser importante, mas com o uso da memória cache, a diferença pode ser substancial. a) Analise o código cuidadosamente de forma que você entenda em qual ordem os elementos da matriz são chamados. Explique como os elementos na matriz estão posicionados na memória principal e avalie o desempenho de cada uma das funções. Considere que um tipo int do C ocupa 32 bits. b) Considere uma cache de dados com mapeamento direto e 256 linhas de 64 bytes.

As linhas da matriz são linhas na cache. As colunas são bytes dentro de cada linha. Uma A matriz ocupa numa linha 320 bits, ou 40 bytes.

A diferença entre um programa e outro é que:

Num programa as linhas são lidas primeiro e depois cada coluna na linha.

Nesse programa só ocorrera um cache miss por linha, pois as outras colunas (que estavam no mesmo bloco) já terão sido trazidas.

No outro programa as colunas são lidas e depois as linhas em cada coluna.

