As páginas abaixo contêm questões utilizadas pela professora na disciplina de Arquitetura de Computadores (não sei dizer de qual ano são). Sintam-se livres para respondê-las e discuti-las, mas procurem manter o padrão de formatação para facilitar a leitura por parte de todos os interessados. Utilize outras cores para destacar as respostas.

Caso você tenha acesso a questões que não constam deste documento colaborativo, sinta-se livre para adicioná-las também.

Turmas unidas graduam-se mais rápido.

Se você não pretende ajudar nem concorda com estudo colaborativo, apenas feche este documento.

Não seja babaca!

Diretrizes de estudo para a P2 passadas pela professora em aula:

Stallings

- ***Capítulo 14 - Instruction-Level Parallelism and Superscalar Processors***
    - **Execução de programas com problema**
        - **Explicação do porquê**
        - **Possível solução**
        - **Dizer se tem forward, se tem bolha, etc**
    - **Programação**

- ***Capítulo 17 - Parallel Processing***
    - ***Não vai se cobrar programação neste capítulo***
    - **O que esse processador faz?**
    - **Qual a capacidade dele?**
    - **Abordagem conceitual**

- ***Capítulo 18 - Multicore Computers***
    - ***Não vai se cobrar programação neste capítulo***
    - **Abordagem conceitual**

Bibliografia utilizada na disciplina:

Básica

- **Organização e Projeto de Computadores: A interface Hardware/Software. Patterson, D.A.; Hennessy, J.L. 3ª edição, Ed.Campus (https://libgen.pw/view.php?id=78653), (https://libgen.pw/view.php?id=1174086 (5ª ed))**

- **Arquitetura e Organização de Computadores. Stallings W. 8ª edição, Ed. Prentice Hall (https://libgen.pw/view.php?id=578352)**

Complementar

- **Computer Architecture: A Quantitative Approach. J.L. Hennessy & D. A. Patterson (https://libgen.pw/view.php?id=863308)**

P2 Antiga 01

1. (2,0) Explique no que consiste a metodologia de avaliação de desempenho utilizada na confecção da lista TOP 500.

Resposta e discussão:

2. (2,0) Considere o seguinte programa para responder à questão:

```
I1 MOVE R3,R7          /R3 <- (R7)/
I2 LOAD R8,(R3)        /R8 <- MEM(R3)/
I3 ADD R3,R3,4         /R3 <- (R3) + 4/
I4 LOAD R9,(R3)        /R9 <- MEM(R3)/
I5 BLE R8,R9,L3        /Desvia se (R9) < (R8)/
```

Este programa apresenta que tipos de dependência? Justifique sua resposta mostrando as instruções.

OBS: Branch on Less or Equal desvia se R9 <= R8(só pra constar)

Resposta e discussão:

Write-Write: I1, I3 (dependência de saída)
Read-Write: I2, I3 (antidependência)
Write-Read: I1, I2 (dependência de dados)
Dependência procedural: I3, I5

Mas se fossem todas:
Write-Read: I3, I4(dados)
Write-Read: I4, I5(dados)
Write-Read: I2, I5(dados)
Write-Read: I1, I4(dados)
(tem que fazer o pipeline??? Acho que não)
Cuíca:

I2 - Dependência em R3 que não está pronto do MOVE ainda
I3 - Antidependência
I4 - Dependência em R3 que o ADD não escreveu ainda nele
I5 - Dependência em R9 que não está na memória do LOAD ainda

3. (2,0) Atualmente as principais máquinas de alto poder de processamento são clusters de computadores (conforme visto nas últimas edições da Top500). O que é um cluster de computadores? Quais são seus principais benefícios? Quais são os desafios?

Resposta e discussão:

Clusters são um conjunto de computadores fracamente acoplados que são estruturados de forma que trabalham como um único, grande, computador. Se dividem em nós, sendo cada nó, responsável por uma atividade ou tarefa delegada que por sua vez são controlados por softwares. Devido ao seu grande porte, geralmente também fazem utilização de barramentos como redes de alta performance, como por exemplo a topologia Dragonfly utilizada nos computadores Cray.

Seus principais benefícios estão no poder de processamento, sendo considerado atualmente o método de alcançar os melhores índices de poder computacional, ao mesmo tempo que também desfruta do paralelismo de thread e instrução para cada nó, que o permite executar múltiplos programas em diferentes nós, cada qual rodando diferentes thread de diferentes processos em diferentes etapas do programa.

Atualmente os maiores desafios de Clusters podem ser resumidos em Comunicação (o barramento continua sendo um gargalo para o sistema), Nível de paralelismo de thread e código (dificuldades para otimizar programas cujo nível de paralelismo seja baixo ou não exista paralelismo), Controle de memória (protocolos e padrões que definem coerência de cache ou estruturação do gerenciamento de memória de forma que o acesso e compartilhamento seja o mais ótimo possível – aumentando tempo de processamento dos nós).

Cluster é um grupo de computadores completos interconectados trabalhando juntos, criando a ilusão de ser uma única máquina. Cada computador em um cluster é chamado de nó e é capaz de operar independentemente. Ele é muito usado para aplicações de servidores. Seus principais benefícios são:

- ***Escalabilidade absoluta**: um cluster pode ter dezenas, centenas ou até milhares de máquinas, cada uma sendo um multiprocessador.*
- ***Escalabilidade incremental:** é possível adicionar novos nós ao cluster em incrementos pequenos.*
- ***Alta disponibilidade**: como cada nó no cluster é um computador independente, a falha de um nó não significa a perda do serviço.*
- ***Desempenho superior**: é possível montar um cluster com poder computacional igual ou maior do que uma única máquina de grande porte, com custo bem menor.*

<u>Arquitetura do cluster:</u>
*Os computadores individuais são conectados por uma rede de alta velocidade. Cada computador é capaz de operar independentemente. Além disso, uma camada intermediária (middleware) de software é instalada em cada computador para possibilitar a operação do cluster. **O middleware do cluster fornece uma imagem unificada do sistema para o***

***usuário, conhecida como imagem de sistema único**. O middleware é responsável também por fornecer alta disponibilidade, pelo balanceamento de carga e respostas a falhas em componentes individuais.*

4. (2,0) Caracterize uma máquina CC-NUMA e apresente suas vantagens e desvantagens.

Resposta e discussão:

*Uma máquina CC-NUMA é uma máquina NUMA – **Non Uniform Memory Access** (**máquina fortemente acoplada** – em que todos os processos acessam memória principal (mas divergem no tempo de acesso) –, que possui **Coerência de Cache**. Resumidamente, coerência de cache é o protocolo do sistema que força uma atualização sempre que todas as caches que compartilham variáveis estão inválidas com relação a uma a outra. Atualizando todas baseada na cache mais atual (que sofreu a modificação).*

CC-NUMA é uma máquina NUMA no qual é mantida a coerência de cache entre as memórias de cache dos vários processadores. Uma máquina NUMA é uma máquina de acesso não uniforme à memória, com memória compartilhada (fortemente acoplada). Coerência de cache é um problema dos sistemas multiprocessadores modernos que utilizam um ou dois níveis de cache associados a cada processador.

Prós e contras de NUMA

A principal vantagem de um sistema CC-NUMA é que ele pode permitir desempenho eficiente em níveis mais altos de paralelismo do que SMP, sem requerer maiores mudanças no software. Com vários nós NUMA, o tráfego do barramento de qualquer nó individual está limitado a uma demanda com qual o barramento pode lidar. No entanto, se muitos dos acessos à memória forem para nós remotos, o desempenho começa a falhar. Há uma razão para acreditar que essa falha de desempenho pode ser evitada. Primeiro, o uso de caches L1 e L2 é projetado para minimizar todos os acessos à memória, incluindo os remotos. Se uma boa parte do software tiver uma boa localidade temporal, então acessos remotos à memória não devem ser excessivos. Segundo, se o software tiver uma boa localidade espacial e se a memória virtual está em uso, então os dados necessários para uma aplicação residirão em um número limitado de páginas frequentemente usadas que podem ser carregadas inicialmente na memória local da aplicação em execução. Os projetistas do Sequent reportaram que tal localidade espacial aparece, sim, em aplicações representativas. Finalmente, o esquema de memória virtual pode ser aprimorado ao incluir no sistema operacional um mecanismo de migração de página que move uma página da memória virtual para um nó que a está usando frequentemente; os projetistas da Silicon Graphics reportaram sucesso com essa abordagem.

Mesmo que a diminuição do desempenho por causa do acesso remoto seja tratada, existem duas outras desvantagens para a abordagem CC-NUMA. Primeiro,o CC-NUMA não se parece transparentemente como um SMP; alterações de software serão necessárias para mover um sistema operacional e aplicações de um sistema SMP para um CC-NUMA.

Isso inclui alocação de página, alocação de processos e balanceamento de carga pelo sistema operacional. Uma segunda preocupação é em relação à disponibilidade. Esta é uma questão complexa e depende da implementação exata do sistema CC-NUMA.

5. (2,0) Discorra sobre as três principais mudanças na organização do processador vistas durante a disciplina que permitiram o oferecimento de execução paralela (de várias granularidades) e, consequentemente, o aumento de desempenho.

Resposta e discussão:

1. ***As mudanças organizacionais no projeto dos processadores se concentraram, em primeiro lugar, no aumento do paralelismo em nível de instruções, para que mais trabalho pudesse ser feito em cada ciclo de clock. Estas mudanças incluem, em ordem cronológica (Figura 18.1):***

2. ***pipeline: instruções individuais são executadas por um pipeline de estágios de tal forma que, durante a execução de uma instrução em um estágio do pipeline, outra instrução é executada em outro estágio do pipeline.***

3. ***superescalar: vários pipelines são construídos pela replicação de recursos da execução. Isto possibilita execução paralela de instruções em pipelines paralelos, assim que os hazards são evitados.***

4. ***multithreading simultâneo (smT) : bancos de registradores são replicados para que várias threads possam compartilhar o uso dos recursos do pipeline.***

A partir do avanço da organização dos processadores, dentre SISD, SIMD,MISD e o MIMD, múltiplas instruções e múltiplos dados, obteve-se uma organização de múltiplos processadores acarretando em 3 arquiteturas: Cluster como um conjunto de computadores completos interconectados trabalhando como um só e com memória distribuída; SMP sendo um conjunto de processadores conectados por barramento com memória compartilhada e mesmo tempo de acesso e

P2 Antiga 02

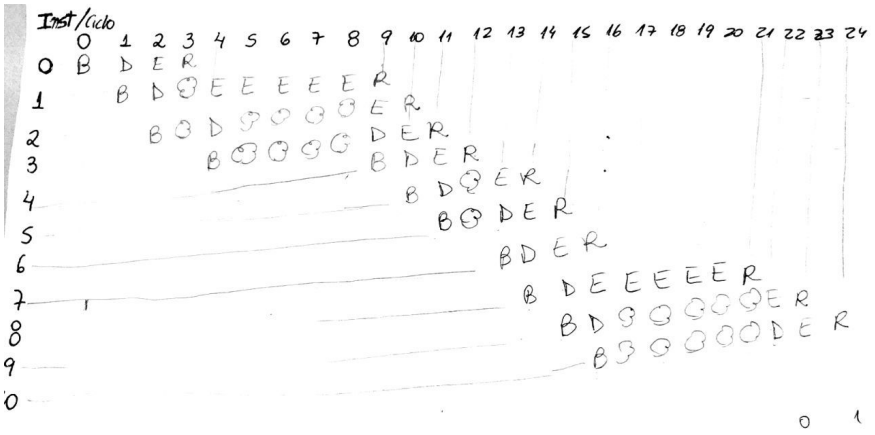
1. (2,0) Suponha uma pipeline de 4 estágios: Busca, Decodificação/iniciação, Execução e Resposta. Suponha que cada estágio consome um ciclo, exceto o de execução. O estágio de execução gasta um ciclo para operações lógicas e para operações aritméticas simples, mas consome cinco ciclos para uma instrução de carga de dado na memória. Uma pipeline escalar simples, com execução fora de ordem, teria a seguinte execução para as primeiras instruções:

Instrução	Busca	Decodificação	Execução	Resposta
0 ADD r3, r1, r2	0	1	2	3
1 LOAD r6, [r3]	1	2	4	9
2 AND r7, r5, 3	2	3(a bolha ta no 3º ciclo, nao tem como ser 3. Acho que seria 4) (Porque seria necessário a bolha se ele pede “3”e não “r3”, seria desnecessário esperar 1 ação e fazer no 4)(entendi)	5	6

a) (0,5) Utilizando o raciocínio apresentado, complete a tabela abaixo supondo que instruções não podem ser emitidas para execução fora de ordem (preencher a caneta):

Instrução	Busca	Decodificação	Execução	Resposta
0 ADD r3, r1, r2	**0**	**1**	**2**	**3**
1 LOAD r6, [r3]	**1**	**2**	**4**	**9**
2 AND r7, r5, 3	**2**	**4 (pq é em ordem)**	**9**	**10**
3 ADD r1, r6, r0	**4**	**9**	**10**	**11**
4 ADD r7, r0, 8	**9**	**10**	**11**	**12**
5 OR r2, r4, r7	**10**	**11**	**13**	**14**
6 SUB r5, r4, r7	**11**	**13**	**14**	**15**
7 ADD r0, r1, 10	**13**	**14**	**15**	**16**
8 LOAD r6, [r5]	**14**	**15**	**16**	**21**
9 SUB r2, r1, r6	**15**	**16**	**22**	**23**
10 AND r3, r7, 15	**16**	**22**	**23**	**24**

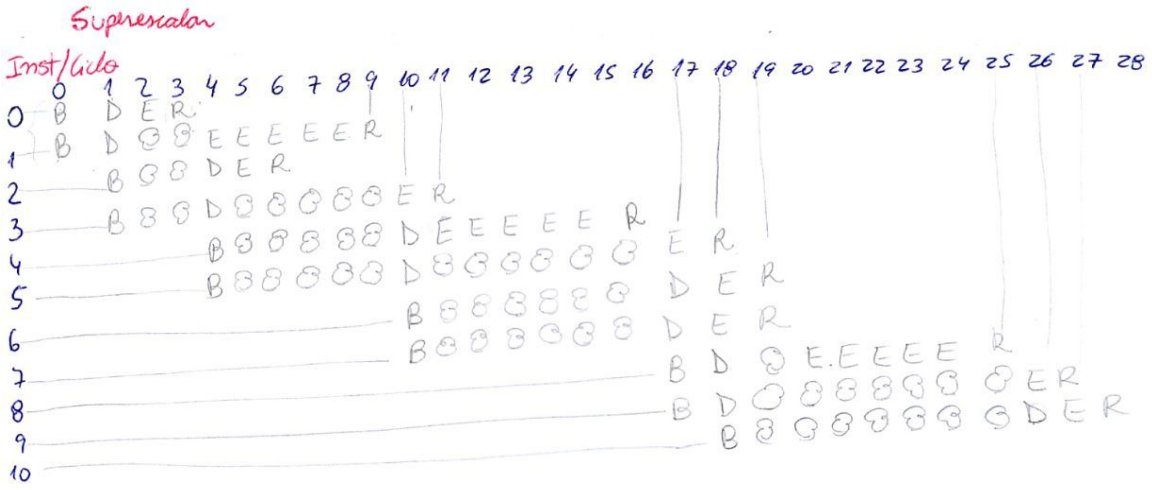
Meu pipeline:



b) (1,5) Complete a tabela abaixo supondo uma implementação superescalar que pode manipular duas instruções de cada estágio (preencher a caneta):
Não confiem na minha resposta, não ta totalmente certa

Instrução	Busca	Decodificação	Execução	Resposta
0 ADD r3, r1, r2	0	1	2	3
1 LOAD r6, [r3]	0	1	4	9
2 AND r7, r5, 3	1	4	5	6
3 ADD r1, r6, r0	1	4	10	11
4 SRL r7, r0, 8	4	10	11	16
5 OR r2, r4, r7	4	10	17	18
6 SUB r5, r3, r4	10	17	18	19
7 ADD r0, r1, 10	10	17	18	19
8 LOAD r6, [r5]	17	18	20	25
9 SUB r2, r1, r6	17	18	26	27
10 AND r3, r7, 15	20	26	27	28

Meu pipeline:



3. (2,0) Explique como é realizada a coerência de cache em máquinas que possuem mais de um nível de cache (L1 e L2), podendo ser exclusivo ou compartilhado.

Resposta e discussão:

Quando há mais de um nível de cache, geralmente L1 fica dentro do núcleo e L2 pode ser compartilhado ou exclusivo.
Se for **exclusivo**, significa que L2 pertence só aquele núcleo;
Se for **compartilhado**, significa que L2 pertence a todos os núcleos do processador.
*Dado de L1 podem estar em outro L1 em outro processador, portanto, é necessário fazer um desses esquemas: Version Control(estático) ou One-time identifier(dinâmico).

Em organizações que possuem dois níveis de cache, a cache L1 não está conectada diretamente ao barramento, desta forma, políticas para manutenção da coerência de cache, como o protocolo MESI, devem ser empregadas em L1. Uma forma de manter a coerência é aplicando uma política write-through, onde as alterações são feitas em L1 são copiadas para a cache L2, tornando-se assim visíveis á todas as outras caches. Para que essa política funcione, é necessário que L1 seja um subconjunto de L2.

5. (2,0) Quais são as abordagens que possibilitam a execução paralela e simultânea de várias threads? Descreva as organizações dos processadores que a suportam.
(não sei se é isso, foi a única coisa que achei da matéria)
Aborgadem: TLP - Paralelismo em nível de thread;
Alguns exemplos de organizações:múltiplas instruções e múltiplos dados(MIMD). que se enquadram nessa categoria são SMPs, clusters e sistemas NUMA, que compartilham uma memória única e acessos de dispositivos de I/O.

Em termos gerais, existem quatro abordagens principais para multithreading:

Multithreading intercalado: isto é conhecido também como multithreading de granularidade fina.O processador lida com dois ou mais contextos de thread ao mesmo tempo, trocando de uma thread para outra a cada ciclo de clock. Se uma thread é bloqueada por causa das dependências de dados ou latências de memória, ela é pulada e uma thread pronta é executada.

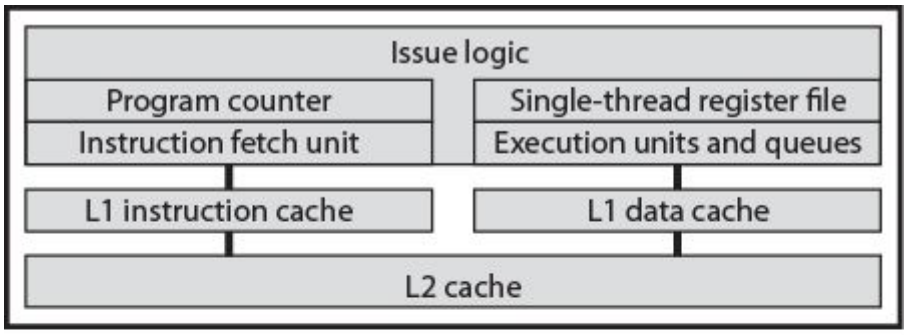
Multithreading bloqueado: isto é conhecido também como multithreading de granularidade grossa.As instruções de uma thread são executadas sucessivamente até que ocorra um evento que possa causar atraso, como uma falha de cache. Este evento induz uma troca para outra thread. Esta abordagem é eficiente em um processador em-ordem que iria parar o pipeline num evento de atraso como uma falha de cache.

Multithreading simultâneo (SMT): instruções são enviadas simultaneamente a partir de múltiplas threads para unidades de execução de um processador superescalar. Isto combina a capacidade de envio de instruções superescalares com o uso de múltiplos contextos de threads.

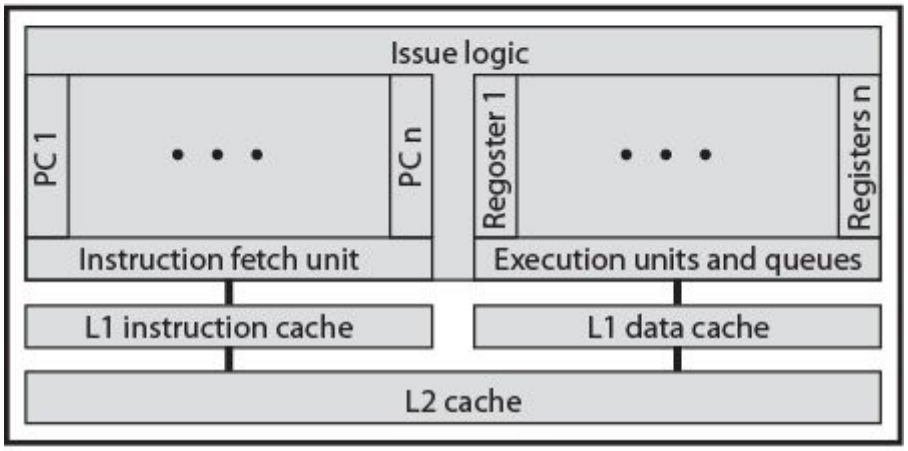
Chip multiprocessadores (multicore): neste caso, o processador inteiro é replicado em um único chip e cada processador lida com threads separadas. A vantagem desta abordagem é que a área de lógica disponível em um chip é usada eficientemente sem depender da sempre crescente complexidade no projeto do pipeline. Isto é conhecido como multicore; analisamos este tópico separadamente no Capítulo 18.

(será que ela não quer que fale sobre TLP também?)

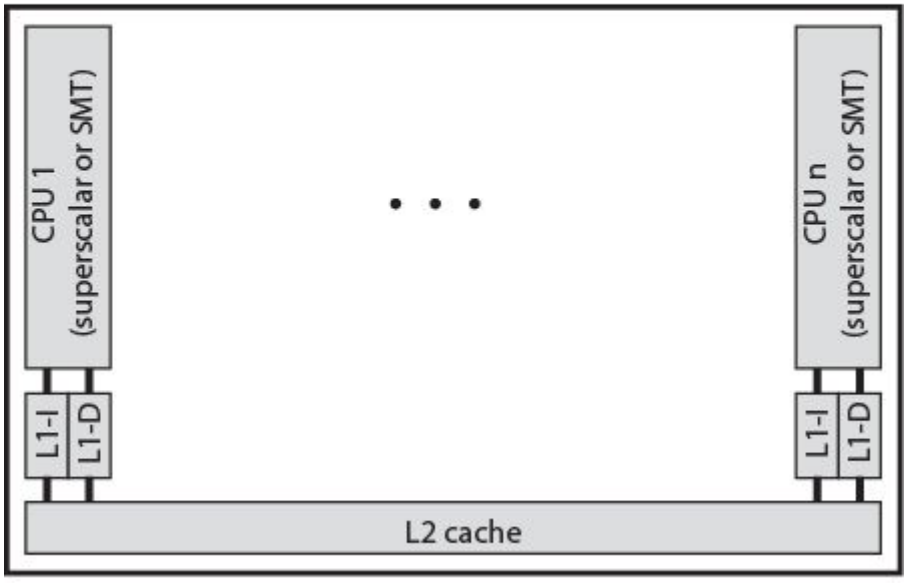
b e c da figura abaixo



(a) Superscalar



(b) Simultaneous multithreading



(c) Multicore

Prova X

4) Descreva dois tipos de organizações classificadas como MIMD e as compare apresentando vantagens e desvantagens entre elas.

- SMP(Symmetric Multiprocessor): Todos os processadores desempenham as mesmas funções (daí o termo simétrico). Compartilham uma memória única ou conjunto; compartilham barramento de acesso à memória; o tempo de acesso à memória para uma dada área é aproximadamente o mesmo para qualquer processador.
  Vantagens:
  - Crescimento incremental: o usuário pode aumentar o desempenho do sistema adicionando novos processadores.
  - Escalabilidade: o fabricantes podem oferecer variedade de produtos com características de desempenho e custo diferentes (número de processadores).
  - Disponibilidade: em um multiprocessador simétrico, como todos os processadores podem efetuar as mesmas funções, a falha de um único processador não trava a máquina.
  - Transparência: a existência de vários processadores é transparente para usuário. O sistema operacional toma conta do escalonamento de threads o processos em processadores individuais e da sincronização entre processadores.

  Desvantagens:
  - Todas as referências à memória passam pelo barramento comum. – Gargalo no acesso à memória. É desejável equipar cada processador com uma memória cache.
  - Cache por CPU x Problemas de coerência de Cache (várias cópias dos mesmos dados podem existir em caches diferentes simultaneamente)

- NUMA(Non Uniform Memory Access): (descrição la em cima)
  Vantagens:
  - O espaço de endereçamento global permite um modelo de programação mais amigável
  - O compartilhamento de dados entre as tarefas é rápido
  - Menor latência de acesso a memória (menos gente tentando usar o barramento)

  Desvantagens:
  - Falta de escalabilidade. Aumentando o número de CPUs, aumenta o tráfego entre memória e CPU
  - Custo: alto para máquinas com o número elevado de processadores

REC

5.(2,0) Caracterize cada uma das alternativas de multithreading em uma máquina.
(Caiu na REC, confie que não vai cair amanhã) São 11 tipos!!!! Misericórdia

§ Escalar único:
Este é o pipeline simples encontrado em máquinas RISC e CISC tradicionais, sem multithreading.
§ Multithread escalar intercalado:
Esta é a abordagem de multithreading mais fácil de ser implementada. Ao trocar de uma thread para outra em cada ciclo de clock, os estágios do pipeline podem ser mantidos totalmente ocupados, ou quase totalmente ocupados. O hardware deve ser capaz de trocar de um contexto de thread para outro entre os ciclos.
§ Multithread escalar bloqueado:
Neste caso, uma única thread é executada até que ocorra um evento de atraso que pararia o pipeline, momento em que o processador troca para outra thread.
§ Superescalar:
Esta é a abordagem superescalar básica sem nenhum Multithread . Até há relativamente pouco tempo, esta era a abordagem mais poderosa para permitir paralelismo dentro de um processador. Observe que, durante alguns ciclos, nem todos os slots de envio são usados. Durante esses ciclos, menos que o número máximo de instruções é usado; chamamos isso de perda horizontal. Durante outros ciclos de instrução, nenhum slot de envio é usado; estes são os ciclos quando nenhuma instrução pode ser enviada; chamamos isso de perda vertical.
§ Multithread superescalar intercalado:
Durante cada ciclo são emitidas tantas instruções quantas forem possíveis a partir de um único thread. Com esta técnica, atrasos potenciais por causa das trocas de threads são eliminados, conforme discutido anteriormente. No entanto, o número de instruções enviado em qualquer ciclo ainda é limitado pelas dependências que existem dentro de qualquer thread.
§ Multithread superescalar bloqueado:
Novamente, as instruções de apenas uma thread podem ser emitidas durante qualquer ciclo e a multithread bloqueado é usado.
§ slot de envio:
Uma arquitetura VLIW, como IA-64, coloca várias instruções em uma única palavra. Normalmente, uma VLIW é construída pelo compilador, o qual coloca operações que podem ser executadas em paralelo na mesma palavra. Em uma máquina VLIW simples se não for possível preencher a palavra completamente com instruções a serem emitidas em paralelo, no-ops são usados.
§ Multithread intercalado:
Esta abordagem deveria fornecer eficácia semelhante àquela provida por multithreading intercalada em uma arquitetura superescalar.
§ Multithread VLIW bloqueado:
Esta abordagem deveria fornecer eficácia semelhante àquela provida por multithread bloqueado em uma arquitetura superescalar.
§ Multithreading simultâneo:

Se um thread possui um alto grau de paralelismo em nível de instruções, ela pode, em alguns ciclos, ser capaz de preencher todos os slots horizontais. Em outros ciclos, as instruções de duas ou mais threads podem ser enviados. Se threads suficientes estão ativos, normalmente seria possível enviar o número máximo de instruções em cada ciclo, fornecendo um nível alto de eficiência.

§ chip multiprocessador (multicore):

A cada processador é atribuído um thread a partir do qual ele pode enviar até duas instruções por ciclo.