Antes de mais nada, pedimos a bênção de Oberlan Romão(Eternamente lembrado).

Ó grande Oberlan, dai-nos força para vencer mais este obstáculo. xD

OBJETIVO:

E aí galera, tudo bem?

*A ideia desse Docs é tentar **aprender/ensinar/discutir** as marretas para a prova final do Ricardo (uma vez que não vai dar pra gente aprender toda teoria em pouco tempo). Cada um colocaria aqui o que sabe e o que tem dúvida pra gente entrar num consenso. Depois a gente marca de reunir no DPI pra estudarmos mais.*

Pelo que fiquei sabendo muita gente foi mal na prova substitutiva achando que tinha ido bem, então para aqueles que ficaram na pendura e conhecem outros que ficaram também e possuem o e-mail deles, por favor, os convide. Afinal, não possuo o e-mail de todos.

A ideia aqui é fazer as listas e as provas anteriores do Ricardo. Qualquer sugestão é bem vinda. Abraço a todos! Espero que possa ser útil a todos nós.

Obs.: A intenção não é fazer dessa lista o único meio de estudo, mas sim criar mais uma documentação mais acessível do que os slides que o Ricardo nos passou.

PESSOAL, VOU DIVIDIR AS "MARRETAS" EM TRÊS NÍVEIS (DEPENDENDO DO NÍVEL, ACHO QUE VOCÊ DEVE FOCAR UM POUCO MAIS, LEMBRANDO QUE NA PROVA VAI CAIR QUASE SEMPRE A ÚLTIMA OPÇÃO - AFINAL, O RICARDO NÃO VAI PEGAR MAIS LEVE NA FINAL):

BÁSICA	FAZ QUE APRENDE	FROM HELL

Vários simuladores de execução:

http://www.ecs.umass.edu/ece/koren/architecture/

MATÉRIA PROVA 1:

LISTA 1: Resolvendo Conflitos e Dependências de Dados

Exemplo 1: Sem encaminhamento. Conflitos de RAW (Read after Write)

BÁSICA

(Por favor, se tiver qualquer coisa errada, me corrijam... =D)

A ideia aqui é, para cada instrução:

- Buscar - (F)etch
- Decodificar - (D)ecode
- Executar - (E)xecute
- Acessar memória - (M)emory - (Acessa memória, escreve dados ou pega dados dela)
- Escreve - (W)rite Back - (Grava o valor calculado em um registrador)

De forma que cada uma bh executada de forma independente (claro) e repeitem as dependencias de registradores necessários. Note que na figura cada dependencia está em negrito e colorida.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Add r1,r2,r3	F	D	E	M	W												
Subi **r2**,r2,5		F	D	E	M	W											
*Add r4,r3,**r2**			F	D	D	D	E	M	W								

Ld r1,8(r2)				F	F	F	D	E	M	W							
**Ld r3,16(r2)							F	D	E	M	W						
Add r5,r1,r3								F	D	D	D	E	M	W			
Sd r5,8(r2)									F	F	F	D	D	D	E	M	W

(*) A instrução necessita do registrador r2, e por isso deve esperar até ele ser escrito pela instrução anterior no registrador, propriamente dito (W), para que assim possa continuar sendo executada.

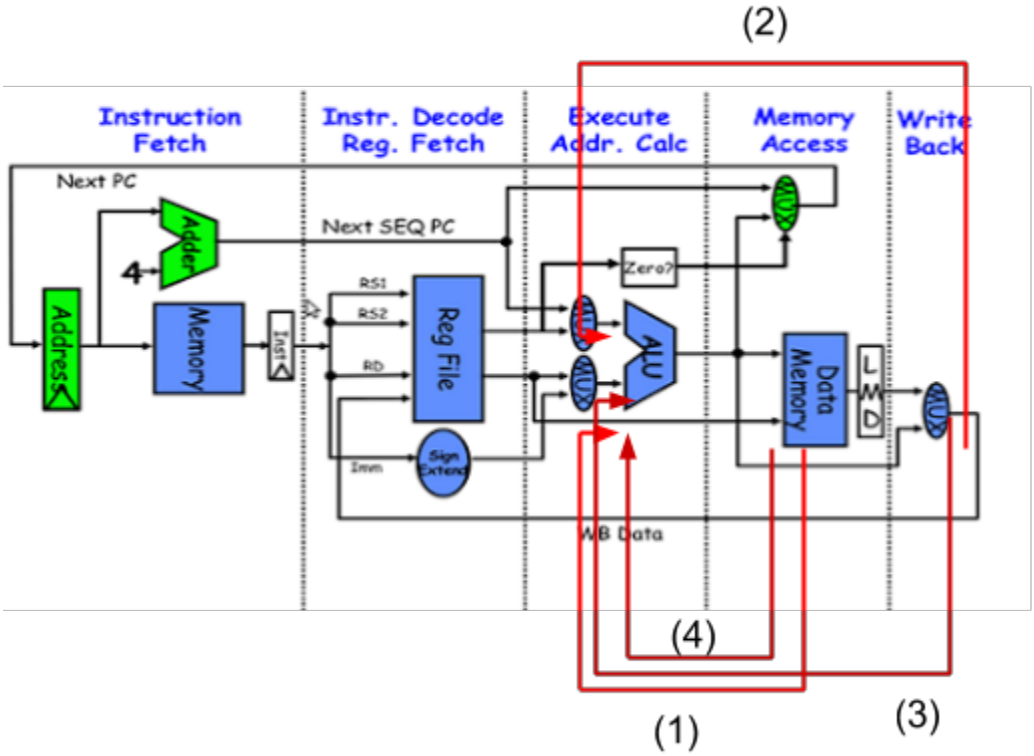
As instruções seguintes seguem o mesmo padrão, observando que nenhum passo pode ser feitos ao mesmo tempo, por esse motivo...

(**) A instrução anterior ficou buscando por 3 ciclos, pois a instrução anterior não havia acabado de ser decodificada. Como as duas não poderiam ser executadas ao mesmo tempo, a instrução de baixo (**Ld r3,16(r2)) ficou aguardando a instrução anterior ser buscada.

Exemplo 1: Com encaminhamento

FAZ QUE APRENDE

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Add r1,r2,r3	F	D	E	M	W							
Subi r2,r2,5		F	D	E	M(1)	W(2)						
Add r4,r3,r2			F	D	E(1)	M	W					
Ld r1,8(r2)				F	D	E(2)	M	W				
Ld r3,16(r2)					F	D	E	M	W(3)			
Add r5,r1,r3						F	D	D	E(3)	M(4)	W	
Sd r5,8(r2)							F	F	D	E(4)	M	W



Obs.: Link da imagem original (sem os fios) pra quem quiser editá-la com Drawing: https://lh3.googleusercontent.com/-xZmzucrgXlc/ThCWrea1csI/AAAAAAAAAB8/-N9T6nJgrOc/pipeline.png

MATÉRIA PROVA 2:

CACHE - CÁLCULO DE CUSTOS

	NumLinhas	TAG*	Comp	DecoLinha	DecoBloco
Map. Direto	Cache	Memoria / NumLinhas	1 de TAG bits	1 de n : 2^n ; 2^n = NumLinhas	-
Map. Direto c/ bloco	Cache / Bloco	Memoria / (NumLinhas * Bloco)	1 de TAG bits	1 de n : 2^n ; 2^n = NumLinhas	1 de m : 2^m ; 2^m = Bloco
Tot. Associativa	Cache	Memoria	NumLinhas de TAG bits	-	-
Tot. Associativa c/ bloco	Cache / Bloco	Memoria / Bloco	NumLinhas de TAG bits	-	1 de m : 2^m ; 2^m = Bloco
N-way	Cache / N	Memoria / NumLinhas	N de TAG bits	1 de n : 2^n ; 2^n = NumLinhas	-
N-way c/ bloco	Cache / (N * Bloco)	Memoria / (NumLinhas * Bloco)	N de TAG bits	1 de n : 2^n ; 2^n = NumLinhas	1 de m : 2^m ; 2^m = Bloco

***** A TAG é o Log2 do resultado.Ou seja, se *Memoria / NumLinhas = 2^13*,
então TAG = 13

Map. Direto: Linhas * (Tag + Dados) + Decodificador + Comparador

Map. Direto c/ bloco: Linhas * (Tag + Bloco) + Decodificadores + Comparador

Tot. Associativa: Dados + Linhas * (Tag + Comparador)

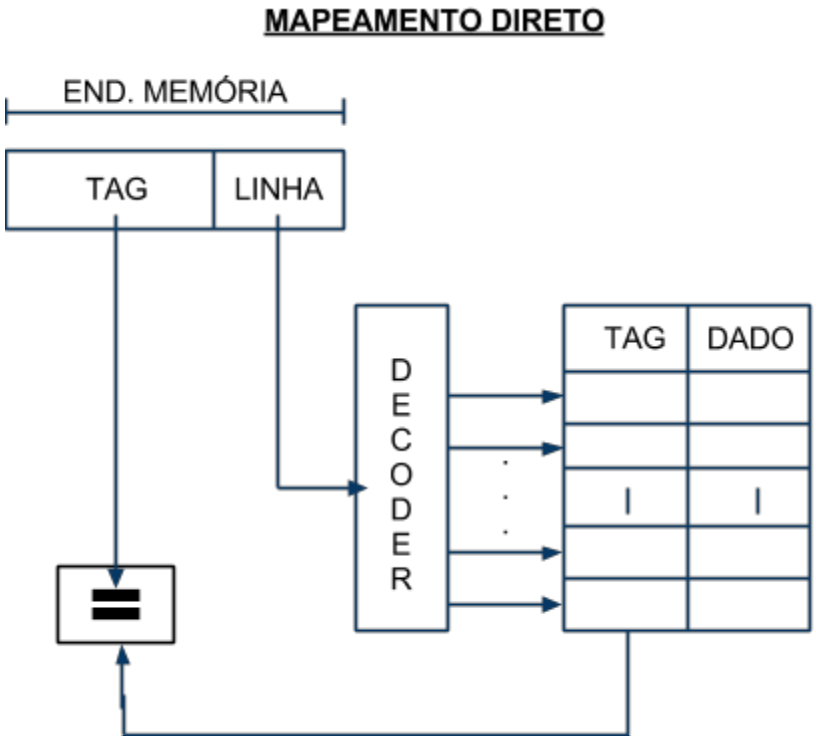
Tot. Associativa c/ bloco: Dados + Linhas * (Tag + Comparador) + Decodificador

N-way: Dados + Linhas * N (Tag) + N Comparadores + Decodificador

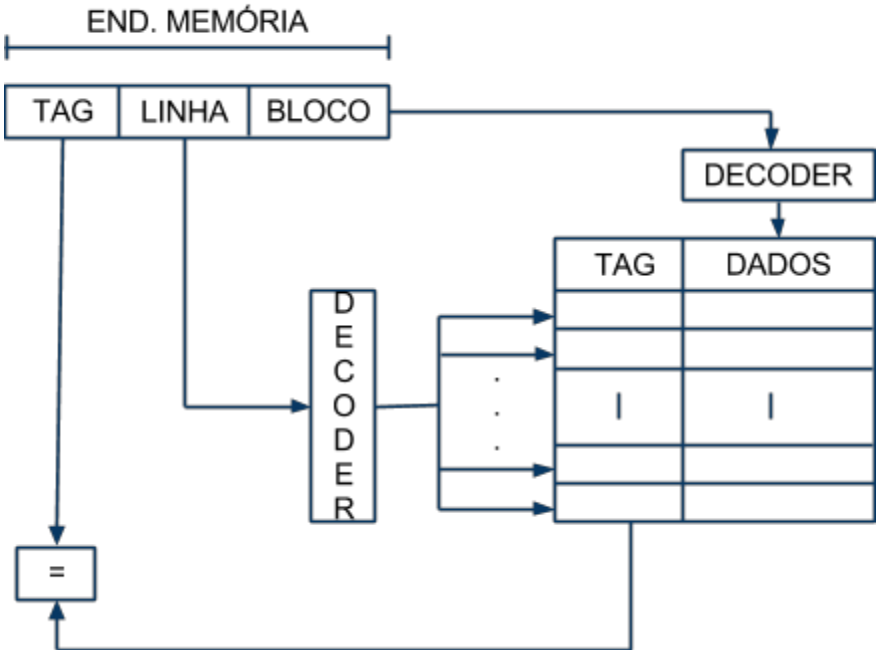
N-way c/ bloco: Dados + Linhas * N (Tag) + N Comparadores + Decodificadores

Dados = Cache (pelo menos é o que parece)

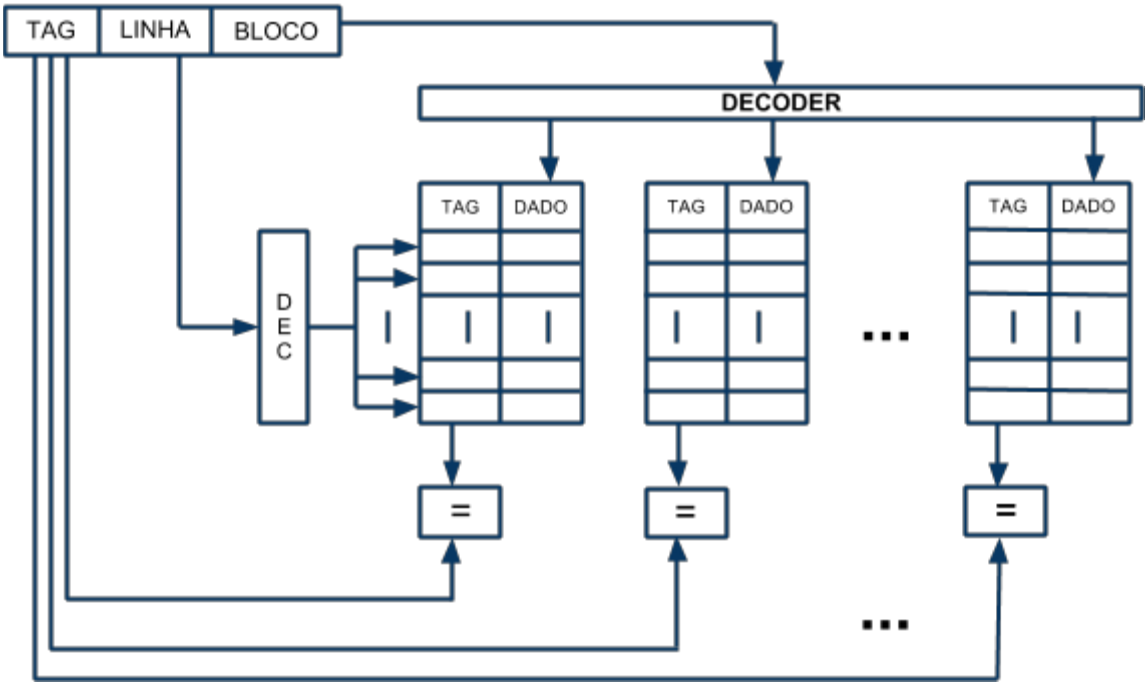
Apenas para ilustrar e tornar a tabela mais mnemônica:



MAPEAMENTO DIRETO C/ BLOCO



ASSOCIATIVA POR CONJUNÇÃO (N-WAY) COM BLOCO



Exercício 1 da prova substitutiva.

Suponha 4Gb de memória. Calcule o custo da cache 4-way de 128K com bloco de 64.

Começamos transformando essa coisa toda pra potências de 2.

Memória: 4Gb = 2^32
Cache: 128K = 2^17
Bloco: 64 = 2^6
N: 4 = 2^2 (opcional)
^^
Agora é só aplicar a marreta!

NumLinhas = Cache / (N * Bloco) = 2^9
TAG = log2 (Memória / (NumLinhas * Bloco)) = log2 (2^17) = 17

Comparadores: N de TAG bits = 4 de 17 bits
DecoLinha*: 1 de n:2^n, sendo que 2^n = NumLinhas. Ou seja, 9:2^9
DecoBloco: 1 de m:2^m, sendo que 2^m = Bloco. Ou seja, 6:2^6

**Cuidado pra não confundir n com N!*

Resposta
TAG = 17
Linhas = 2^9
Decodificadores = 9:2^9 e 6:2^6
Comparadores = 4 de 17 bits

FAZ QUE APRENDE

Exercício 1 da prova de 2010.

Calcule o espaço ocupado pela TAG, numero de comparadores, decodificadores em um sistema com

as seguintes caches, considerando uma memória principal de 8Gb.

(a) Mapeamento Direto de 2Mb e bloco de 128 bytes.

Memória: 8Gb = 2^33
Cache: 2Mb = 2^21
Bloco: 128bytes = 2^7

NumLinhas = 2^14
Tag = 12
1 Comparador de 12 bits
1 DecoLinha de 14:2^14
1 DecoBloco de 7:2^7

(b) 4-way com 64Kb e 16bytes por bloco.

Memória: 8Gb = 2^33
Cache: 64Kb = 2^16
Bloco: 16bytes = 2^4
N: 4 = 2^2

NumLinhas = 2^10
Tag = 19
4 Comparadores de 19 bits
1 DecoLinha de 10:2^10
1 DecoBloco de 4:2^4

(c) Totalmente associativa com 16Kb e blocos de 8 bytes

Memória: 8Gb = 2^33
Cache: 16Kb = 2^14
Bloco: 8bytes = 2^3

NumLinhas = 2^11
Tag = 30
2^11 Comparadores de 30 bits

1 DecoBloco de 3:2^3

CACHE - CÁLCULO DE TEMPO

Existe uma fórmula para calcular o tempo médio gasto?

Tmedio = Tcache + Falhas * Penalidade

Tcache: é o tempo gasto para verificar se o dado requisitado está na cache. Depende do tipo de cache (mapeamento direto, associativa, n-way).

Falhas: é a probabilidade do dado requisitado não estar na cache. Depende do tipo de cache, da localidade (?) e do tamanho do bloco.

Penalidade: é o tempo gasto para buscar o dado em outro lugar. Depende do tempo de acesso à memória, da quantidade de níveis (L2, L3), da política de escrita (write-back ou write-through), da polítca de substituição e do tamanho do bloco.

E se tiver cache L2?

Tmedio = Tcache + Falhas * Penalidade

Mas penalidade é o tempo gasto para buscar o dado em outro lugar, ou seja, na L2, que também é uma cache. Logo,

Tmedio = Tcache + Falhas * (TcacheL2 + FalhasL2 * Penalidade)

Em último caso, a penalidade será o tempo gasto para procurar o dado na memória. É aí que começa a complicar...

Como calcular a penalidade de acesso à memória?

Depende de 3 fatores básicos:

TPrim: Tempo gasto para acessar a primeira posição da memória

TMem: Tempo gasto para transferir as informações

Bloco: Tamanho do bloco da cache anterior à memória

Penalidade = TPrim + TMem * Bloco

Tmedio = Tcache + Falhas * Penalidade

[EXEMPLO]

Pra complicar ainda mais, a penalidade depende também da política de escrita

Write-back

Entra mais uma variável...

Sujo: Porcentagem de vezes que vamos encontrar um bloco sujo na memória. Bloco sujo é aquele cujo conteúdo da cache é diferente do conteúdo na memória.

Para facilitar, vamos fazer...

Mem = TPrim + TMem * Bloco

Penalidade = Sujo * 2 * Mem + (1-Sujo) * Mem.

Mas de onde saiu esse 2? Quando o bloco está sujo, temos que ir na memória duas vezes. Uma para gravar lá o bloco mais atualizado (que está na cache), e outra para pegar o bloco que estamos querendo.

Fórmula final:
Tmedio = Tcache + Falhas * Penalidade =>
Tmedio = Tcache + Falhas * (Sujo * 2 * Mem + (1 - Sujo) * Mem) =>
Tmedio = Tcache + Falhas * (Sujo * 2 * (TPrim + TMem * Bloco) + (1-Sujo) *
(TPrim + TMem * Bloco))

[EXEMPLO]

Write-through

Esqueça aquela fórmula básica do Tmedio...

TaxaLeitura: Portentagem de operações feitas que são de leitura

Penalidade = TPrim + TMem * Bloco

Tmedio = TaxaLeitura * (Tcache + Falhas * Penalidade) + (1 - TaxaLeitura) * (TPrim + Falhas * Penalidade)

FROM HELL :(

Continuação do exercício 1:

Se tem política write-back com 20% de blocos sujos. Qual o tempo médio de acesso para um sistema de cache com 3% de falhas? Escreva a equação em função dos outros parâmetros que influenciam o tempo da cache.

Relembrando as fórmulas...

Tmedio = Tcache + Falhas * Penalidade

Penalidade = Sujo * 2 * Mem + (1-Sujo) * Mem

Mem = TPrim + TMem * Bloco

Do exercício, temos:

Sujo = 0,2

Falhas = 0,03

Então...

Tmedio = Tcache + 0,03 * (0,4 * Mem + 0,8 * Mem)

Escreva também a equação para um sistema com 2 níveis de cache usando write-back nos dois níveis.

Nem o Ricardo deve saber fazer essa...

FROM HELL :(

Exerício 2 da prova de 2010.

Calcule o tempo médio de acesso para as seguintes caches

(a) Cache com Tcache = 1ns, 3% de falhas, Write-back, 30% de blocos sujos, 10ns + 2ns por byte para acesso à memória, bloco de 32 bytes.

Traduzindo...

Tcache = 1ns (nanosegundo)
Falhas = 0,03
Sujo = 0,30
TPrim = 10ns
TMem = 2ns
Bloco = 32

Relembrando as fórmulas...

Tmedio = Tcache + Falhas * Penalidade
Penalidade = Sujo * 2 * Mem + (1-Sujo) * Mem
Mem = TPrim + TMem * Bloco

Aplicando a marreta...

Mem = 10ns + 2ns * 32
Penalidade = 0,6 * Mem + 0,7 * Mem
Tmedio = 1ns + 0,03 * Penalidade

Abrindo a resposta...

Tmedio = 1ns + 0,03 * (0,6 * (10ns + 2ns * 32) + 0,7 * (10ns + 2ns * 32))

(b) Acrescente uma cache L2 com write-back, bloco de 128bytes, 50% de blocos sujos, TL2 = 3ns e 35% de falhas.

Traduzindo...

TCacheL2 = 3ns
Falhas = 0,35
Sujo = 0,50
Bloco = 128bytes

E agora??? Até então, quando ocorria uma falha, procurávamos o dado na memória e pagávamos Mem por isso:

Tmedio = 1ns + 0,03 * (0,6 * Mem + 0,7 * Mem)

Agora, no lugar da memória, vamos procurar antes na cache L2...

TmedioL2 = 3ns * 32 + 0,35 * (1,00 * Mem' + 0,5 * Mem')
Mem' = 10ns + 2ns * 128

Note que multiplicamos o TcacheL2 pelo tamanho dos blocos da cache anterior. Agora é só trocar o Mem do cálculo anterior pelo TmedioL2! :D

Tmedio = 1ns + 0,03 * (0,6 * TmedioL2 + 0,7 * TmedioL2)
TmedioL2 = 3ns + 0,35 * (1,00 * Mem' + 0,5 * Mem')
Mem' = 10ns + 2ns * 128

MEMÓRIA VIRTUAL

Começe por aqui:
http://regulus.pcs.usp.br/~jean/so/AULA%2014%20-%20Mem%F3ria%20Virtual.pdf

Como calcular a quantidade de molduras, páginas, etc?

Temos as seguintes variáveis, que são informadas no problema:

Tamanho da página [TPag]

Tamanho da memória virtual [TVirtual]

Tamanho da memória real (ou física) [TReal]

Usamos isso para descobrir o resto...

Quantidade de molduras [QMold]: TReal / TPag

Quantidade de páginas [QPag]: TVirtual / TPag

Tamanho da tabela de páginas [TTabPag]: TVirtual / TPag

Quantidade de molduras que a Tabela de páginas ocupa [QMoldTabPag]: TTabPag / TPag

BÁSICA

Exemplo com 1 nível de paginação retirado dos slides no PVANet:

Tamanho da página: 2K

Tamanho da memória virtual: 16M

Tamanho da memória física: 128K

Começamos transformando essa coisa toda pra potências de 2

Tamanho da página: 2^11

Tamanho da memória virtual: 2^24

Tamanho da memória real: 2^17

Aplicando a marreta...

Quantidade de molduras: TReal / TPag = 2^6 = 64

Quantidade de páginas: TVirtual / TPag = 2^13 = 8K

Tamanho da Tabela de Páginas: TVirtual / TPag = 2^13 = 8K
Quantidade de molduras que a Tabela de Páginas ocupa: TTabPag / TPag = 2^2 = 4

[DESENHOS]

Paginação em 2 níveis

Usamos 2 níveis de paginação quando a memória real não tem tamanho suficiente nem pra armazenar a Tabela de Páginas. Por exemplo:

Tamanho da página: 10
Tamanho da memória virtual: 1000
Tamanho da memória física: 100

Faça as contas e verá que só a Tabela de Páginas já ocupa toda a memória física.

Como faço as contas usando 2 níveis da paginação? Não sei. Espero que não caia na prova, já que não tem nenhum exemplo assim no PVANet

FAZ QUE APRENDE

Exerício 2 da prova substitutiva:

Suponha um sistema com 1 Terabyte de memória virtual, 1Mbytes de Memória real e páginas de 4Kbytes.
Quantos níveis de paginação são necessários?

Transformando pra potências de 2...

Tamanho da página: 2^12
Tamanho da memória virtual: 2^40
Tamanho da memória real: 2^20

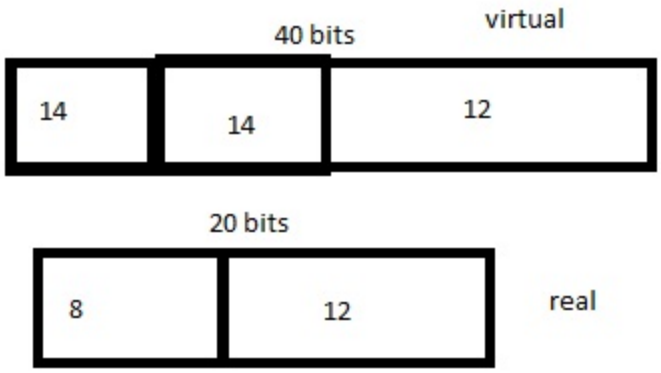
Aplicando a marreta...

Quantidade de molduras: TReal / TPag = 2^8
Quantidade de páginas: TVirtual / TPag = 2^28
Tamanho da tabela de páginas: TVirtual / TPag = 2^28
Quantidade de molduras que a Tabela de páginas ocupa: TTabPag / TPag = 2^16

Note que o tamanho da tabela de páginas é maior que o tamanho da memória real. Logo, serão necessários 2 níveis de paginação.

FROM HELL :(

Como fica dividido o endereço virtual no real?(Acho que é isso by Charles)



Quantos acessos à memória virtual são feitos para mapear o endereço virtual no real no pior caso?
Pega os 28bits iniciais do endereço virtual, procura na tlb-> falha!
Pega os 14bits iniciais, e procura oq tem gravado na posição->Numero da moldura!
Procura nessa moldura e na linha dos 14 bits do meio->Vazio, falha!
Procura uma moldura que esta vazia para trazer dados da virtual.
Todas molduras cheias.
Substitui a menos recente, leva essa pra virtual(1º acesso).
Busca a pagina que contem o endereço requisitado e coloca na pagina agora vazia(2º acesso)
Faz as devidas indexações.

2 acessos!

Se tiver TLB com 128 entradas, qual o custo da cache totalmente associativa?

TAG()	**DADOS(indica moldura)**
28 bits	**10 bits**

Tag: 28 bits

Linhas: 128

Dados: 8 bits;

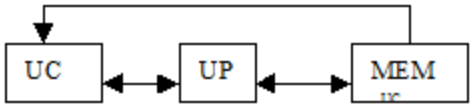
128 comparadores de 28 bits;

MATÉRIA PROVA 3:

1. Arquiteturas Paralelas

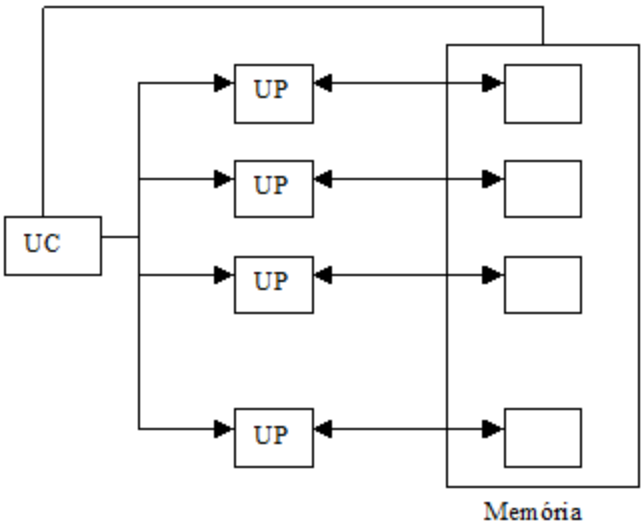
1.1 SISD (Single Instruction Single Data)

Arquitetura em que um único processador, executa um único fluxo de dados, para operar em dados armazenados em uma única memória. De acordo com Michael J. Flynn, SISD pode ter características de processamento concorrente. Pré-busca de instruções e o uso de pipelines na execução de instruções, são os exemplos mais comuns achados na maioria dos Computadores SISD modernos.



1.2 SIMD (Single Instruction Multiple Data)

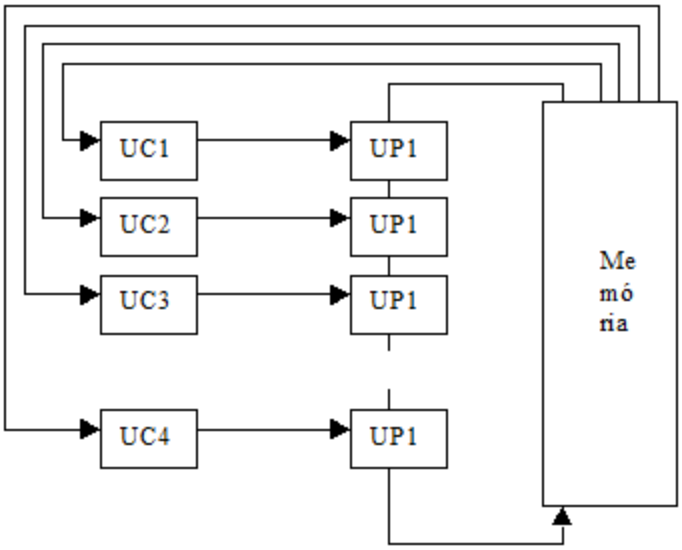
Neste modo, a mesma instrução é aplicada simultaneamente a diversos dados para produzir mais resultados. O modelo SIMD é adequado para o tratamento de conjuntos regulares de dados, como as matrizes e vetores. Esse tipo de máquina opera aplicando uma única instrução a um conjunto de elementos de um vetor.



Existem duas categorias de máquinas SIMD: processadores matriciais (array processors) e processadores vetoriais (vector processors). Os processadores matriciais não implementam funções escalares e, geralmente, são configurados como periféricos de outras arquiteturas (mainframes e minicomputadores) para a execução da porção vetorial dos programas. Já os processadores vetoriais são mais bem sucedidos comercialmente que os matriciais. Parte do seu sucesso deve-se ao fato de que um processador vetorial é uma arquitetura de uso geral, capaz de executar todas as operações escalares normais.

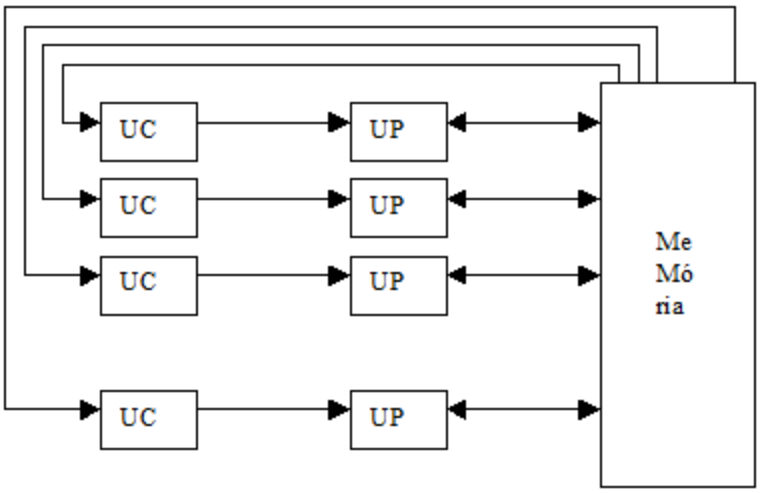
1.3 MISD (Multiple Instruction Single Data)

Tipo de arquitetura onde muitas unidades funcionais executam operações diferentes sobre os mesmos dados. Arquiteturas pipeline pertencem a este tipo. Computadores executando as mesmas instruções redundantemente, a fim de detectar erros e máscara, de uma forma conhecida como replicação de dados, pode-se considerar que pertencem a este tipo. A proposta de implementação que mais se aproxima desta categoria é a da máquina de fluxo de dados.



1.4 MIMD (Multiple Instruction Multiple Data)

É o caso dos multiprocessadores, onde várias instruções podem ser executadas ao mesmo tempo em unidades de processamento diferentes controladas por unidades de controle independentes (uma para cada unidade de processamento).



2. Processador Vetorial (SIMD)

São máquinas que possuem um conjunto de processadores que operam de forma paralela e síncrona, normalmente executando a mesma função. Os processadores de uma máquina vetorial são chamados de Unidades de Processamento (UP) e trabalham sob a supervisão de uma única Unidade de Controle (UC).

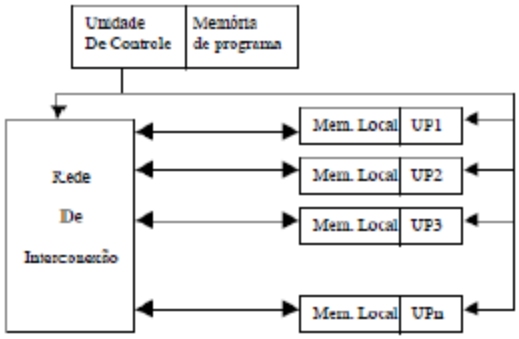


Figura 1 – UPs com Memória Local

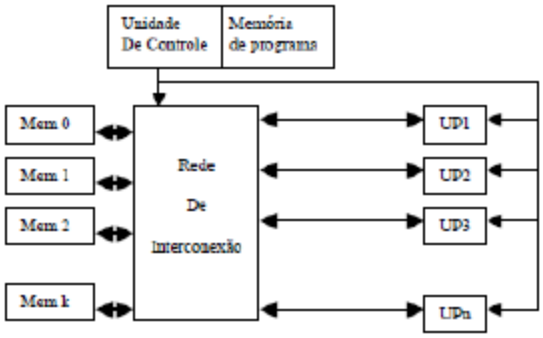


Figura 2 – UPs sem Memória local

Existem dois tipos básicos de arquiteturas vetoriais. A principal diferença entre eles é a presença ou ausência de memória local nas unidades de processamento. Em um caso, a UP possui uma unidade lógico-aritmética(ULA), registradores e uma memória local. A Unidade de Controle decodifica as instruções do programa e determina onde elas serão executadas. Se for uma instrução vetorial, esta será executada nas unidades de processamento. Se a instrução for escalar, estas serão executadas em um processador de propósito geral.

Portanto, as arquiteturas vetoriais funcionam como aceleradores de cálculo para operações vetoriais e normalmente funcionam conectadas a um computador de propósito geral.

Principais características:

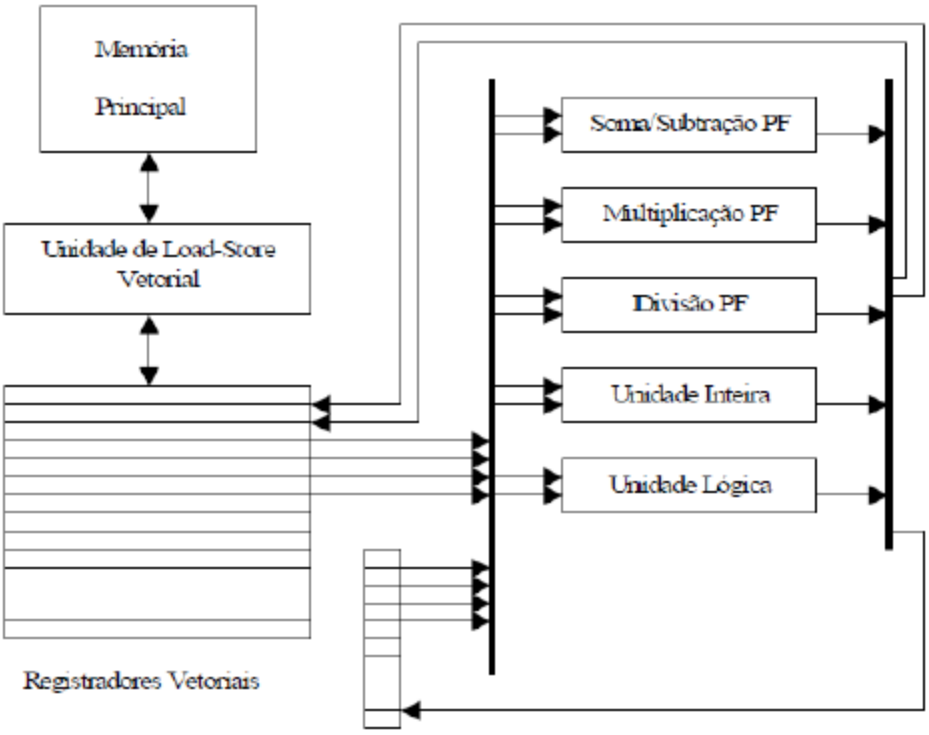
- As unidades de processamento funcionam de forma síncrona executando uma mesma operação definida pela Unidade de Controle sobre um conjunto de dados. (SIMD);
- A Unidade de Controle tem capacidade de desabilitar as Unidades de processamento que não devem participar de uma dada operação;
- As unidades de processamento são construídas usando processadores dedicados que funcionam como aceleradores de computadores de uso geral.

3. Arquitetura vetorial com pipeline

Basicamente, consiste em uma máquina com unidade aritméticas implementadas usando técnicas de pipeline. O objetivo deste tipo de arquitetura é acelerar o processamento de instruções aritméticas realizadas sobre estruturas de dados do tipo vetor.

Nas primeiras implementações deste tipo de arquitetura todas as operações eram feitas buscando os operandos diretamente da memória. Este tipo de arquitetura era chamada de memória-memória. Este tipo de arquitetura tem o grande inconveniente dos conflitos que podem ocorrer com os bancos de memória, mesmo quando se utiliza memória intercalada com um número grande de bancos.

Um segundo tipo de arquitetura, chamada registrador-vetor (figura abaixo), utiliza um tipo especial de registrador que é capaz de armazenar vários elementos de um vetor.



- **Registradores vetoriais:** cada registrador vetorial é um conjunto de tamanho fixo de n registradores, podendo armazenar um vetor de até n posições.
- **Registradores escalares:** utilizados para o armazenamento de dados escalares a serem usados nas operações escalares ou cálculos de endereços dos vetores.
- **Unidades funcionais vetoriais:** são pipelines aritméticos que podem iniciar uma nova operação a cada ciclo de clock.
- **Unidade "Load-Store" vetorial:** responsável pela busca (e armazenamento) de um vetor da (na) memória principal.

3.1 Requisitos de processamento vetorial

Em geral, as arquiteturas que usam o processamento vetorial com pipeline devem ter as seguintes propriedades:

- Processos idênticos são repetidamente invocados, cada qual podendo ser subdividido em sub processos;
- Operandos sucessivos são fornecidos ao pipeline e necessitam de um mínimo de controle local e buffers;
- Operações executadas por unidades de pipeline distintas devem ser capazes de compartilhar recursos como memória e barramento.

Estas características explicam porque grande parte dos processadores vetoriais possuem estruturas de pipeline.

As instruções vetoriais são normalmente especificadas pelos seguintes campos:

- **Código de operação:** seleciona a unidade funcional a ser usada (ou define a configuração a ser adotada por uma unidade multifuncional);
- **Endereço de base:** usado para especificar o endereço inicial (primeiro elemento do vetor) de operandos e resultados;
- **Incremento de endereço:** especifica a distância entre os elementos do vetor que serão utilizados na operação (nem sempre a distância é igual a 1).
- **Off-set do endereço:** especifica um deslocamento a partir do endereço de base. Nem todas as operações precisam pegar o primeiro elemento do vetor.
- **Tamanho do vetor:** usado para verificação do limite do vetor para impedir que um elemento que não pertença ao vetor seja utilizado.

3.2 Problemas comuns

Dois problemas ocorrem muito frequentemente e podem prejudicar o desempenho dos pipelines aritméticos. O primeiro problema diz respeito ao tamanho do vetor. Nem sempre caberá totalmente dentro de um registrado vetorial. O segundo problema acontece quando a operação não utiliza elementos adjacentes do vetor. Como resolver estes problemas?

3.2.1 Controle de tamanho do vetor

Muitas vezes o tamanho do vetor não é conhecido em tempo de compilação. A solução para este problema consiste na utilização de um registrador de tamanho de vetor (VLR – Vector Length Register). O uso do VLR resolve o problema quando n <= MVL(tamanho máximo do vetor – Maximum Vector Length) definido pela arquitetura da máquina.

E quando N > MVL? Para estes casos uma técnica chamada “strip minning” é utilizada. Esta técnica consiste em quebrar o loop de n iterações em 2 partes:

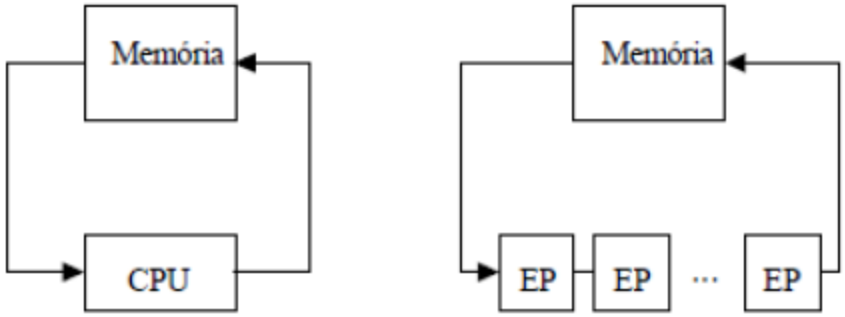
1. Cria um loop que manipula qualquer número de iterações que seja múltiplo de MVL;
2. Cria outro loop que manipula as iterações restantes.

3.2.2 Elementos não adjacentes

A solução para este tipo de problema é projetar as unidades de load-store com capacidade de lidar com incrementos (strides) não unitários. O incremento é guardado em um registrador escalar e instruções específicas para lidar com este tipo de situação são projetadas.

4. Processadores Sistólicos

O princípio de funcionamento dos processadores sistólicos se baseia na subdivisão de tarefas, como em um pipeline. Assim, um único processador é substituído por um conjuntos de Elementos de Processamento (EPs) de forma a aumentar a taxa de computação. Não existe Unidade de Controle (UC).



a) arquitetura convencional

b) arquitetura sistólica

Cada EP é uma unidade simples com uma capacidade de processamento limitada e dedicada a uma tarefa específica. As arquiteturas sistólicas são voltadas para a resolução de problemas específicos.

São suas principais características:

- Arquitetura composta por um conjunto de células (EPs) conectadas diretamente a suas células vizinhas;
- Cada célula é capaz de executar apenas algumas operações simples;
- A informação entre as células flui de maneira semelhante a um pipeline;
- Somente as células da borda se comunicam com o mundo exterior;

4.1 Arquitetura Sistólica x Pipeline

- Na arquitetura sistólica o fluxo pode ser bidimensional e em diferentes velocidades;
- Na arquitetura sistólica tanto as entradas primárias quanto os resultados parciais podem fluir entre as células, enquanto que no pipeline apenas os resultados parciais são passados entre os estágios;
- Os pipelines, normalmente possuem apenas um estágio de entrada e um de saída, nas arquiteturas sistólicas é possível existir mais de um célula de entrada e mais de uma célula de saída.

4.2 Aspectos de projeto de sistemas dedicados

- **Definição da tarefa:** Este passo procura identificar primeiramente se o algoritmo a ser implementado em hardware é realmente do tipo “compute-bound”. Após este estudo será possível responder à pergunta: vale a pena implementar um circuito dedicado para este algoritmo? Se a resposta for sim, passa-se para o passo de projeto.

- **Projeto:** Definição da geometria do circuito a ser integrado. O projeto deve procurar minimizar a área total, já que o custo é proporcional a área do chip. Outras preocupações podem ser a minimização da potência, regularidade do projeto, etc.
- **Validação:** É necessário validar a solução. A validação é feita através de técnicas de simulação. A simulação pode ser lógica e/ou elétrica.
- **Implementação:** Após simulado e validado, o projeto pode ser produzido em escala comercial;
- **Testes:** Os chips produzidos podem possuir algum defeito inerentes ao processo de integração. Os testes são feitos para sanar esses defeitos.

Com isso, espera-se alcançar algumas metas no projeto:

- **Projeto simples e regular:** O custo do projeto deve ser baixo o suficiente para justifica-lo. A reutilização de estruturas otimizadas e validadas reduzem o custo na fase de projeto. Uma das metas principais é conseguir que o projeto seja modular (expansível). Neste caso, o custo pode ser ajustado ao desempenho desejado.
- **Concorrência e comunicação:** Não basta apenas ter Elementos de processamento eficientes, mas é necessário também que exista paralelismo na execução do algoritmo. Quando vários EPs trabalham concorrentemente, a coordenação da comunicação pode ser outro fator a influir no desempenho do algoritmo.
- **Balanceamento entre computação e entrada e saída:** Considerações de E/S influenciam todo o desempenho da máquina. O objetivo é ter uma taxa de computação balanceada com a taxa de operações de E/S.

5. Multiprocessadores

São máquinas que possuem mais de um processador. Cada processador é de propósito geral capaz de trabalhar sozinho ou em cooperação com os demais processadores para aumentar o desempenho da máquina. Um multiprocessador pode ser caracterizado por dois atributos básicos:

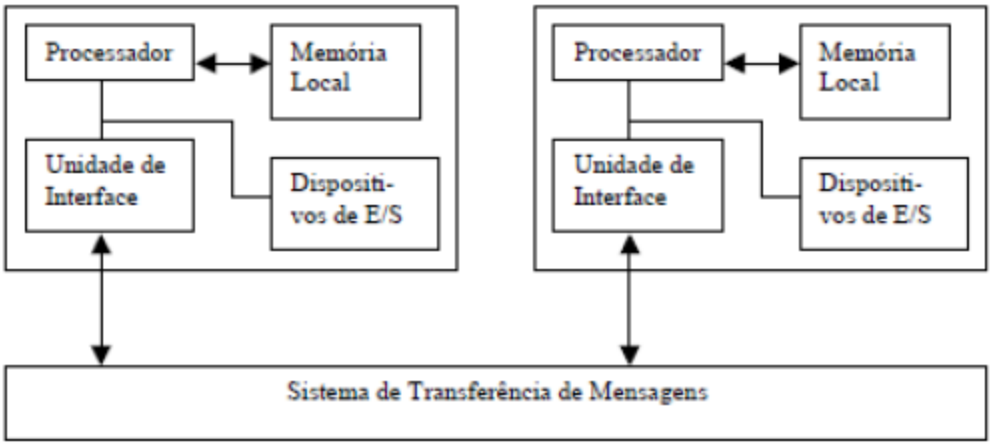
- É um único computador que possui vários processadores;
- Seus processadores podem comunicar-se e cooperar para resolver um dado problema.

5.1 Tipos de comunicação

A comunicação pode ser feita através de troca de mensagens (fracamente acoplado) ou através de memória compartilhada (fortemente acoplado).

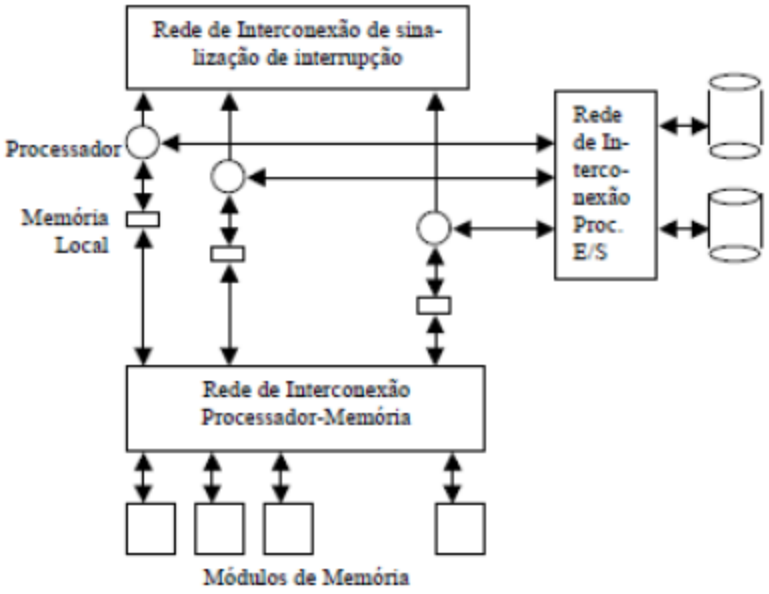
5.1.1 Arquiteturas fracamente acopladas

A comunicação entre os processadores é feita através de troca de mensagens. O desempenho da máquina pode ficar limitado pelo desempenho do sistema de transferência de mensagens.



5.1.2 Arquiteturas fortemente acopladas

A comunicação entre os processadores é feita através de memória compartilhada. É mais utilizada quando o número de processadores é pequeno. Quando o número de processadores aumenta o desempenho da máquina pode ser reduzido quando o número de comunicações aumenta, já que cada troca de informações envolve acessos à memória.



O que define a complexidade da computação em multiprocessadores é a comunicação entre os processadores.

Características desejáveis de processadores para multiprocessamento:

- **Recuperabilidade do processo:** deve ser possível para um outro processador, em caso de falha, recuperar o estado do processo interrompido e reiniciar o processo.
- **Troca eficiente de contexto:** a troca de contexto consiste em salvar o estado de um processo ativo em um processador, interrompê-lo e carregar um outro processo no processador.
- **Espaço de endereçamento virtual e físico grandes:** quanto maior for a capacidade de memória da máquina, melhor. O esquema de memória virtual deve oferecer mecanismos de proteção de memória para evitar acessos indevidos.
- **Primitivas de sincronização:** o paralelismo de um multiprocessador é tipicamente assíncrono, ou seja, um processador pode trabalhar independentemente dos demais. Entretanto a solução do problema muitas vezes exige sincronismo entre os processadores. As primitivas de sincronização são utilizadas para prover a comunicação entre os processadores e implementar exclusão mútua de recursos.
- **Mecanismo de comunicação entre os processadores:** devem existir instruções específicas para prover a comunicação entre os processadores, seja usando mensagens ou usando memória compartilhada.

5.2 Coerência em cache

Um sistema de cache é coerente se, e somente se, uma operação de leitura feita por um processador sempre retorna o valor mais recente do dado.

5.2.1 Verificação estática de coerência

Divide as informações em duas classes distintas:

- **Cacheáveis:** para dados que não mudam (read-only, constantes)
- **Não cacheáveis:** para dados que não podem ser colocados em caches privadas.

5.2.2 Verificação dinâmica de coerência

Permite que multiplas cópias de um dado existam em diferentes caches privadas. Sempre que um processador alterar o valor de um dado na sua cache, este deve sinalizar aos demais para que o dado seja invalidado. Para isso, são usadas as seguintes políticas:

- **Write-update:** grava e atualiza o valor nas outras caches
- **Write-invalidade:** grava e sinaliza que o valor das outras caches é inválido.

A diferença entre as políticas ocorre quando um valor é alterado constantemente. Por exemplo,

“for (x=0; x<100; x++)”.

As cópias dos dados nas caches podem estar nos seguintes estados:

- **Read-only (RO):** existem 2 ou mais caches com cópia do bloco e estas cópias não foram alteradas.
- **Exclusive Read-Write (RW):** existe uma única cópia do bloco e ela foi alterada.
- **Exclusive Read-Only (EX):** existe uma única cópia do bloco e ela não foi alterada.

Resolver os problemas de coerência causa tráfego entre as caches, diminuindo o desempenho do sistema.

6. Redes de interconexão

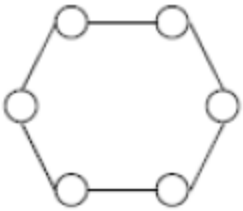
Uma rede de interconexão define a forma de ligação entre as diversas Unidades de processamento da máquina vetorial.

O desempenho dessas redes pode ser dado por três variáveis:

- **Largura de bisseção:** indica quantas mensagens simultâneas podem ser trocadas entre duas metades da rede de interconexão; é definida como o número mínimo de “links” de comunicação que necessitam ser removidos para particionar a rede em duas metades iguais.
- **Diâmetro:** indica qual o menor número de nós intermediários que precisam ser envolvidos, para que dois processadores, o mais distantes possível, se comuniquem; é a distância máxima entre quaisquer dois nós da rede.
- **Grau:** indica o número máximo de mensagens que podem ser manipuladas simultaneamente por cada um dos processadores. Número de canais que incidem em um nó da rede.

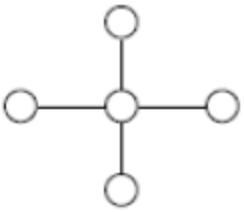
6.1 Redes Estáticas

Possuem uma única forma de interconexão durante toda a sua vida útil.



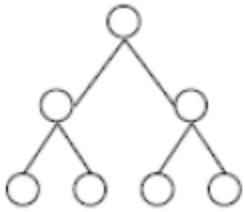
Anel

- A mensagem é repetidamente passada para o próximo nó até chegar ao seu destino;
- Quando o primeiro e últimos nós da topologia em linha estão interconectados.



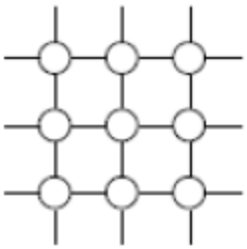
Estrela

- Um nó atua como nó de controle ao qual todos os demais nós estão conectados;
- Por manipular toda a comunicação entre os nós, o nó central é o gargalo do sistema.



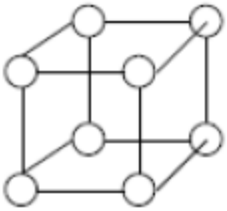
Árvore

- Uma árvore binária de profunidade d tem 2^d-1 nós;
- As redes em árvores sofrem um gargalo de comunicação nos níveis mais altos da árvore binária;
- Este problema pode ser resolvido aumentando a capacidade de comunicação dos "links" que estão mais perto da raiz. Esta rede é chamada de Árvore Gorda.



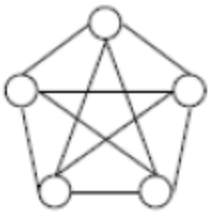
Malha

- Cada processador tem quatro vizinhos aos quais está conectado por um “link”;
- A malha bidimensional é uma extensão do vetor linear;
- Se as duas dimensões da malha não forem iguais, temos uma malha retangular.



Cubo

- Todos os nós podem ser identificados por um número binário;
- Escalabilidade restrita a potências de 2;
- Os tamanhos do hipercubo são definidos por potências de 2.



Malha Completamente Conectada

- Todos os nós estão conectados entre si por um “link” direto;
- O número de arestas do grafo totalmente conectado é dado por: $d = n(n-1)/2$;
- Este tipo de rede é pouco utilizado devido aos altos custos de comunicação.

Propriedades das Redes de Interconexão Estáticas

Topologia	Grau Nó	Diâmetro	Largura Bisseção	Custo (Links)
Linha	1 ou 2	N-1	1	N-1
Anel	2	N/ 2	2	N
Estrela	1 ou N-1	2	1	N-1
Árvore Binária	1, 2 ou 3	$2 \log_2 * ((N+1)/2)$	1	N–1
Malha 2-D	2, 3, ou 4	$2 (N^{1/2}-1)$	$N^{1/2}$	$2(N-N^{1/2})$

Topologia	Grau Nó	Diâmetro	Largura Bisseção	Custo (Links)
Toro	4	$(N^{1/2}-1)$	$2*(N^{1/2})$	2* N
Cubo 3-D	3, 4, 5 ou 6	$3(N^{1/3}-1)$	$N^{2/3}$	$2 (N-N^{2/3})$
Hipercubo	$\log_2 N$	$\log_2 N$	N/ 2	$(N\log_2 N) / 2$
Comp. Conectada	N - 1	1	$(N^2)/ 4$	N(N-1) / 2

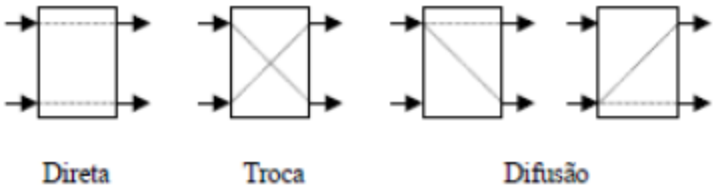
6.2 Redes dinâmicas

São capazes de definir a configuração de conexão dinamicamente no tempo.

São caracterizadas pelos atributos:

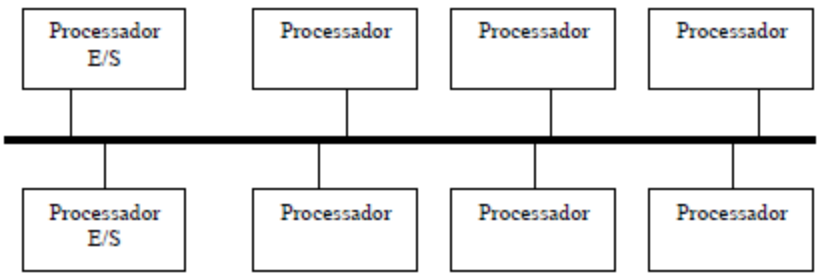
- **Topologia:** Define os possíveis padrões de conexão que podem ser selecionados;
- **Elementos chaveadores:** Definem a função que mapeia uma entrada(ou um conjunto de entradas) em uma saída(ou um conjunto de saídas);
- **Estrutura de controle:** Define o modo de operação dos elementos chaveadores;
- **Tipo de roteamento:** Define a forma através da qual a seleção do caminho é feita. Os dois mecanismos básicos são: transmissão de pacotes e transmissão por circuitos. No primeiro caso o pacote de informação de seleção e o roteamento vai sendo definido à medida que o pacote de informação chega nos elementos chaveadores. No segunda caso a rota é estabelecida completamente antes da transmissão do pacote de informação.

Exemplo de elemento chaveador e os modos de operação possíveis:



As redes dinâmicas podem ser de um ou mais estágios. Nas redes de apenas um estágio o tempo de chaveamento do circuito aumenta, mas por outro lado o tempo de propagação do sinal pode ser menor, uma vez que apenas um estágio precisa ser ultrapassado.

6.3 Barramento Comum de Tempo compartilhado



Vantagens:

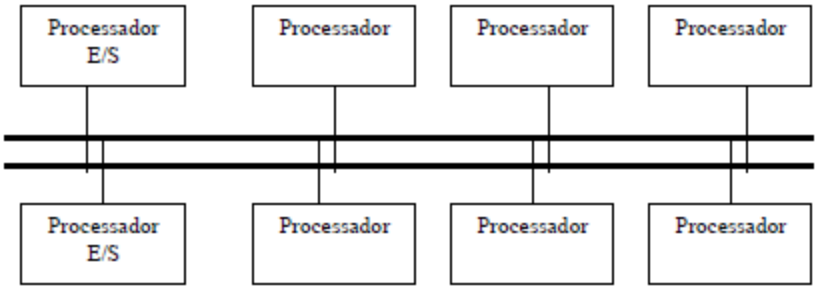
- Menos complexo e mais fácil de reconfigurar

- Facilita a expansão(aumento do número de processadores e/ou módulos de memória)

Desvantagem:

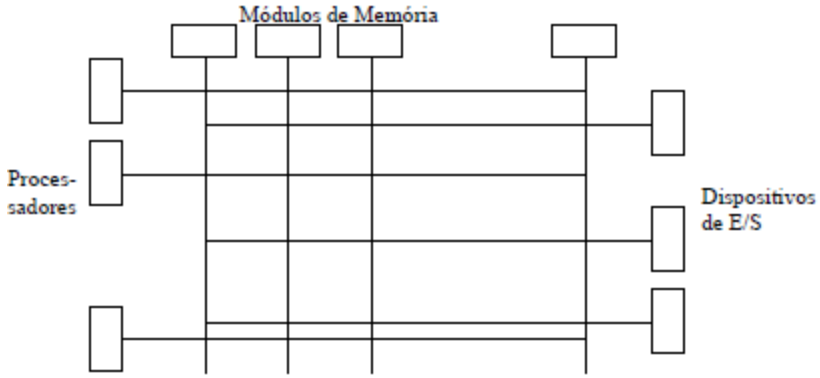
- Conflito, já que apenas uma comunicação pode acontecer de cada vez no barramento. Quando o número de processadores aumenta, essa restrição pode se tornar inaceitável.

6.4 Barramento Múltiplo(Multibus)



O esquema de barramento múltiplo consiste na replicação da estrutura do barramento. Sua vantagem é que duas comunicações simultâneas podem ser feitas. Em contrapartida, aumenta o custo do sistema pois, além de replicação do barramento, as interfaces devem ser alteradas para permitir a ligação de mais um barramento.

6.5 Rede de Barramento Cruzado



- Consiste de vários barramentos ligados através de pontos de cruzamento.
- Um processador pode se ligar em qualquer módulo de memória ou a qualquer dispositivo de E/S.
- Permite que diversas comunicações simultâneas aconteçam.

Podem ocorrer conflitos na rede, no caso de dois ou mais processadores tentarem acessar o

mesmo módulo de memória ou a um mesmo dispositivo de E/S. Assim cada ponto do cruzamento deve ter um circuito de arbitração baseada em pedidos. Assim, quando um processador X quiser acessar o módulo de memória controlado por este ponto de cruzamento ele deve ativar o seu sinal de requisição do barramento correspondente. Se dois ou mais processadores ativarem os seus sinais de requisição o módulo de arbitração decide quem vai usar.

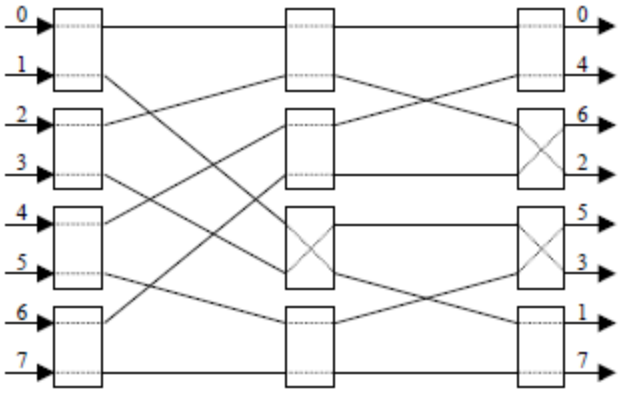
6.6 Rede Bayan

A rede Banyan tem a vantagem de prover uma interconectividade completa entre dois conjuntos de n dispositivos, com o custo do circuito de chaveamento crescendo a razão nlogn, ao contrário de uma rede crossbar onde o custo cresce na razão n^2.

Uma rede Banyan pode ser caracterizada pela tupla (f, s, 1), onde:

- f – Corresponde ao número de arcos de saída do nó;
- s – Corresponde ao número de arcos de chegada do nó;
- 1 – número de níveis da rede.

Implementação de uma rede Banyan (2,2,2) usando elementos chaveadores:



6.7 Pontos relevantes na análise de redes

- **Overhead**
- **Bandwidth:** taxa de transmissão permitida pelo barramento
- **Tempo de transporte**: tempo gasto pela informação na rede de interconexão
- **Custo:** custo necessário para criar a rede
- **Confiabilidade:** se a rede é confiável
- **Latência:** atraso total do transporte da informação pela rede

latência = overheadTX + tempo de transporte + (tamanho da mensagem / bandwidth) +

overheadRX

Onde:
overheadTX = transmissor
overheadRX = receptor

7. Sistemas Operacionais para Multiprocessadores

Conceitualmente existem poucas diferenças entre SO para multiprocessadores e SO para máquinas convencionais.

A complexidade criada pela existência de vários processadores é quanto ao grande número de tarefas assíncronas que podem ser executadas concorrentemente. Isto cria a necessidade de implementação de mecanismos de sincronização entre os processadores.

As principais capacidades funcionais de um SO para multiprocessadores são:

- **Alocação de recursos e esquemas de gerência:** os recursos do sistema vão ser disputados pelos processos ativos na máquina. O SO deve gerenciar a alocação destes recursos entre os processos procurando sempre uma utilização eficiente dos recursos;
- **Proteção de memória e dados:** Áreas de código do SO devem ser protegidas contra escrita por outros processos(exceto se a variável for compartilhada);
- **Prevenção de bloqueios:** O famoso deadlock. Quando um recurso R1 depende do recurso R2 e vice-versa. Nesse caso os dois processos ficaria eternamente esperando que o outro liberasse o recurso que ele está precisando;
- **Terminação anormal de processos ou manipulação de exceções:** a terminação de processos deve ser tratada com muito cuidado em SO de multiprocessadores pois um processos pode ser parte de um programa e, sem esta parte a computação não fica completa.
- **Esquemas de balanceamento de carga:** uma característica desejável dos SO de multiprocessadores é que eles possam distribuir de maneira mais uniforme possível a carga entre os processadores da máquina.

Obs.: Para aqueles que preferirem o PDF:

http://dl.dropbox.com/u/32527742/INF450-OC2-ExerciciosProva3.pdf

1) Definir multiprocessador (mínimo 6 linhas)

Resposta do Ricardo:

Dois ou mais processadores independentes, trabalhando de forma assíncrona ou síncrona. A comunicação pode ser feita por memória compartilhada (fortemente acoplado) ou por troca de mensagem (fracamente acoplado).

As redes de interconexão mais usadas são barramento (fortemente acoplado) com múltiplos caches e redes multiestágios, crossbar e hierárquicos para fracamente acoplados.

Outra resposta:

São máquinas que possuem mais de um processador. Cada processador é um processador de propósito geral capaz de trabalhar sozinho ou em cooperação com os demais processadores para aumentar o desempenho da máquina. Um multiprocessador pode ser caracterizado por dois atributos básicos:

1) É um único computador que possui vários processadores;
2) Seus processadores podem comunicar-se e cooperar para resolver um dado problema.

A comunicação entre os dois processadores pode ser feita através de troca de mensagem (arquitetura fracamente acoplado) ou através de memoria compartilhada (arquitetura fortemente acoplado)

2) Quais as diferenças e semelhanças entre:

a) Array processor e multiprocessador?

Array processor: são máquinas que possuem um conjunto de 'processadores que operam de forma paralela e síncrona, executando normalmente a mesma função. Representam a classe de arquitetura SIMD.

Multiprocessadores: são maquinas que possuem mais de um processador independente, trabalhando de forma assíncrona ou síncrona. Cada processador é um processador de proposito geral capaz de trabalhar sozinho ou em cooperação com os demais processadores para aumentar o desempenho da maquina. Representam a classe de arquitetura MIMD.

Outra resposta:

Array Processors (SIMD): são representativos do paralelismo espacial síncrono, uma unidade de controle distribui para vários elementos de processsamento a mesma instrução que será executada em parelelo por estes sobre seus dados no mesmo instante de tempo.

Multiprocessadores (MIMD): apresentam paralelismo espacial assíncrono, diversos processadores trabalham em paralelo processando suas tarefas concorrentemente para num intervalo de tempo

concluirem a tarefa.
Semelhança: Multiprocessadores também apresentam paralelismo espacial síncrono.

b) Array processor e processador sistólico?

Resposta do Ricardo:

- Array processor:
  - regular ou não
  - programável
  - SIMD
  - uso geral
- Sistólico:
  - problema específico (fixo)
  - regular
  - fabricado nos circuitos
  - fluxo de dados

Outra resposta:

Sistolicos: sao formados substituindo o processador por um conjunto de elementos de processamento objetivando a subdivisão de tarefas, logo, um aumento computacional. Além disso, tem capacidade limitada, pois são criados para resolver um problema específico, o que gera custos altos.
Array Processors: …

c) Processador vetorial com pipeline e array processor?

Processador vetorial com pipeline:
consiste basicamente de uma máquina com unidades aritméticas implementadas usando técnicas de pipeline. O objetivo deste tipo de arquitetura é acelerar o processamento de instruções aritméticas realizadas sobre estruturas de dados do tipo vetor.
Array processor: 000000000000000000
são máquinas que possuem um conjunto de processadores que operam de forma paralela e síncrona, executando normalmente a mesma função. Representam a classe de arquitetura SIMD.

d) Processador vetorial com pipeline e processador superescalar?

Processador vetorial com pipeline:
consiste basicamente de uma máquina com unidades aritméticas implementadas usando técnicas de pipeline. O objetivo deste tipo de arquitetura é acelerar o processamento de instruções aritméticas

realizadas sobre estruturas de dados do tipo vetor.

Processadores Superescalares:

são os que possuem pipelines que permitem a execução de mais de uma instrução simultaneamente (no mesmo ciclo de clock). Isto é obtido através da implementação de múltiplas unidades funcionais, que são unidades onde as instruções são executadas.(http://pt.wikipedia.org/wiki/Superescalar)

3) Cite 3 exemplos de redes de interconexão:

->Crossbar (n nodos): ○ diâmetro = 1 ○ grau=n ○ roteamento direto ○ broadcast?

->Hipercubo (2n nodos): ○ diâmetro=n ○ grau=n ○ roteamento store-and-foward: origem XOR destino ○ broadcast em log(n) passos, nesse caso, n = no de nós

->Malha (n x n nodos): ○ diâmetro: O(n) (?) ○ grau=4 ○ roteamento XY ○ broadcast

4) Como funcionam os algoritmos de roteamento de:

a) Hipercubo?

Origem XOR Destino

b) Clos?

escolher sempre o switch do meio livre para evitar conflitos e soh depois de usar todos repetir.

c) Benes?

esse foi aquele do teste, onde a divisao das entradas e das saidas, ai vc faz a correspondencia para evitar conflitos.

d) Omega com estágios extras?

Origem B XOR B Destino, com B=0 ou B=1.

Exemplo: Origem = 1 (001), Destino = 7 (111) Tentar:

- 0010XOR0111
- 0011XOR1111

No resultado, 1 indica “cruzar” e 0 indica “seguir reto”.

5) Qual a diferença entre Multiprocessadores e Array Processors ?

(Questão igual a 2a...)

6) Quais as principais características de um multiprocessador ?

1. recuperabilidade do processo; 2. troca eficiente de contexto; 3. espaço de endereçamento virtual e físico grandes;

4. primitivas de sincronização; 5. mecanismo de comunicação entre processadores.

7) Explique para que serve um protocolo de coerência em cache ? Como o barramento é utilizado e como são implementadas as máquinas de estado.

Os esquemas de coerência de cache baseados em hardware são mais utilizados e são conhecidos como Protocolos de Coerência de Cache. São utilizados para manter a coêrencia entre os dados da cache. Um sistema é dito ser coerente se todas as leituras por qualquer processador retornam o valor produzido pela última operação de escrita, sem importar qual processador realizou esta operação.

8) Cite 3 redes de conexão, explique suas vantagens e desvantagens.

(não sei se é isso mesmo que foi pedido)

-> Barramento Comum:

+ Menos complexo e mais fácil de reconfigurar. + Facilita a expansão (aumento de processadores e/ou módulos de memória).

- Conflito: apenas uma comunicação pode acontecer de cada vez.

- Aumento de processadores pode se tornar inviável devido aos conflitos.

-> Barramento Múltiplo (Multibus)

+ Reduz o número de conflitos - Aumenta o custo do sistema

-> Barramento Cruzado (Crossbar)

+ Ponto de cruzamento configurado dinamicamente.

+Várias comunicações simultâneas

- Conflitos: quando 2 ou mais processadores tentam acessar a mesma memória ou o mesmo dispositivo de E/S

9) Quais as principais formas de paralelismo que podem existir em um sistema monoprocessador. Estas formas podem ser utilizadas pelos multiprocessadores ?

Microprogramação horizontal

Pré-busca de instruções Processamento de operações de E/S em paralelo com instruções de CPU

Execução em paralelo de instruções aritméticas

Construções de programas paralelos

Sim, pelos processadores Vetoriais

10) De exemplo de um código escalar e sua versão vetorial.

(COPIADO DA APOSTILA - PROCESSADORES VETORIAS)

ESCALAR: Algoritmo SISD:

FOR i = 1 TO N DO FOR j = 1 TO N DO

C(i,j) = 0; FOR k = 1 TO N DO

C(i,j) = C(i,j) + A(i,k)*B(k,j)
END
END Complexidade do algoritmo: O(n3) Número de UPs: 1
VERSÃO VETORIAL:
Algoritmo SIMD:
Pressupõe a existência de uma máquina com N unidades de processamento. FOR i = 1 TO N DO
PAR FOR k = 1 TO N DO C(i,k) = 0;
END FOR j = 1 TO N DO
END
END
PAR FOR k = 1 TO N DO C(i,k) = C(i,k) + A(i,j)*B(j,k) /* multiplicação de vetores */
END
END Complexidade do Algoritmo: O(n2) Número de UPs: n

12) Para que serve o deslocamento, o incremento, o endereço base, o tamanho em uma instrução vetorial ?

Deslocamento: deslocamento a partir do endereço base. Nem todas as operações precisam pegar a primeira posição do vetor.
Incremento: distância entre os elementos do vetor que serão utilizados na operação (nem sempre a distância é 1).
Endereço Base: especifica o endereço inicial (primeiro elemento do vetor) de operandos e resultado.
Tamanho: usado para verificar o limite do vetor e impedir que um elemento que não pertença ao vetor seja utilizado

13) Qual a diferença entre uma rede de conexão dinâmica e uma rede estática ?

As redes de interconexão se dividem em redes estáticas e redes dinâmicas. As redes dinâmicas são capazes de definir a configuração de conexão dinamicamente no tempo. Já as redes estáticas possuem uma única forma de interconexão durante toda a sua vida útil.

14) De um exemplo de código executando em um monoprocessador, em um array processor com barramento (n processadores), em um array processor com crossbar (n processadores). O código deve apresentar um ganho na versão array barramento em relação ao monoprocessador, e um ganho maior na versão crossbar. Considere a

comunicação entre os processadores com custo 1.

15) Para que servem as máscaras ? Explique, dê exemplos de código e de aplicações.

Nem todas as UPs precisam de participar do processamento.
Exemplo : Somar 2 vetores de tamanho n quando vc possui mais de n up's :

```
for i=1 to n
        do c(i)=a(i)+b(i);
end
```

Logo vai ter que ser usado mascaramento para não utilizar o resto das up's.
Complementando: Suponha que se tenha n U.P.'s. Cada bit do registrador de máscara (que possui n bits, cada bit se refere à uma U.P.) da U.C. define o status (ativo ou inativo) das U.P.

Prova do ano retrasado:

1) (5pts)é de rede omega sem estágio extra e com estagio extra só fazer o xor la No primeiro usa-se somente um B (teste com 0 ou 1, que nem no testinho) No segundo, vai tem que usar B1 e B2, aí voce testa 00, 01, 10 ou 11 o que der certo,deu! Ex: 2 -> 7 010 B1 B2 XOR B1 B2 111 …

2) (5pts)uma rede benes, completamente de graça

3) (5pts)explique a diferença entre um array sistólico e um array processor

4) (7pts)definir um multiprocessador (minimo 5 linhas)

5) (3pts)Explique como uma arquitetura dataflow captura o paralelismo.

6) (5pts)Dê um exemplo de uma rede de interconexão e comente sobre: broadcast, distancia mínima, media e máxima entre dois pontos, escalabilidade e paralelismo.

Uma rede de interconexao Bus possui broadcast e distancia minima, média e maxima entre dois pontos iguais a 1. Por sua regularidade é escalável e permite a execução de instruçoes paralelamente. Apesar da simplicidade possui muitas limitações. (A pessoa tirou 4 em 5)
paralelismo: microprogramação horizontal
pré-busca de instrução processamento de operações de E/S em paralelo com instruções de CPU
Execução em paralelo de instruções aritméticas Construção de programas paralelos