RISC X CISC

O projeto do Conjunto de Instruções inicia com a escolha de uma entre duas abordagens, a abordagem RISC e a CISC. O termo RISC - Reduced Instruction Set Computer, ou Computador de Conjunto de Instruções Reduzido e CISC - Complex Instruction Set Computer, ou Computador de Conjunto de Instruções Complexo.

Um computador RISC parte do pressuposto de que um conjunto simples de instruções vai resultar numa Unidade de Controle simples, barata e rápida. Já os computadores CISC visam criar arquiteturas complexas a ponto de facilitar a vida dos compiladores, assim programas complexos são compilados em programas de máquina mais curtos. Com programas mais curtos, os computadores CISC precisariam acessar menos a memória para buscar instruções e seriam mais rápidos.

Os processadores RISC geralmente adotam arquiteturas mais simples e que acessam menos a memória, em favor do acesso aos registradores. A arquitetura Registrador-Registrador é mais adotada, enquanto que os computadores CISC utilizam arquiteturas Registrador-Memória.

A tabela a seguir sumariza as características principais dos computadores RISC em comparação com os CISC.

Características	**CISC**	**RISC**
Arquitetura	Registrador-Memória	Registrador-Registrador
Tipos de Dados	Muito variada	Pouca variedade
Formato das Instruções	Instruções com muitos endereços	Instruções poucos endereços
Modo de Endereçamento	Muita variedade	Pouca variedade
Estágios de Pipeline	Entre 20 e 30	Entre 4 e 10
Acesso aos dados	Via memória	Via registradores

Como as arquiteturas RISC visam Unidades de Controle mais simples, rápidas e baratas, elas geralmente optam por instruções mais simples possível, com pouca variedade e com poucos endereços. A pouca variedade dos tipos de instrução e dos modos de endereçamento, além de demandar uma Unidade de Controle mais simples, também traz outro importante benefício, que é a previsibilidade. Como as intrusões variam pouco de uma para outra, é mais fácil para a Unidade de Controle prever quantos ciclos serão necessários para executá-las. Esta previsibilidade traz benefícios diretos para o ganho de desempenho com o Pipeline. Ao saber quantos ciclos serão necessários para executar um estágio de uma instrução, a Unidade de Controle saberá exatamente quando será possível iniciar o estágio de uma próxima instrução.

Já as arquiteturas CISC investem em Unidades de Controle poderosas e capazes de executar tarefas complexas como a Execução Fora de Ordem e a Execução Superescalar. Na execução Fora de Ordem, a Unidade de Controle analisa uma sequência de instruções ao mesmo tempo. Muitas vezes há dependências entre uma instrução e a seguinte, impossibilitando que elas sejam executadas em Pipeline. Assim, a Unidade de Controle busca outras instruções para serem executadas que não são as próximas da sequência e que não sejam dependentes das instruções atualmente executadas. Isso faz com que um programa não seja executado na mesma ordem em que foi compilado.

A Execução Superescalar é a organização do processador em diversas unidades de execução, como Unidades de Pontos Flutuante e Unidades de Inteiros. Essas unidades trabalham simultaneamente. Enquanto uma instrução é executada por uma das unidades de inteiros, outra pode ser executada por uma das unidades de Pontos Flutuantes. Com a execução Fora de Ordem junto com a Superescalar, instruções que não estão na sequência definida podem ser executadas para evitar que as unidades de execução fiquem ociosas.

Estas características de complexidade tornam os estágios de Pipeline dos processadores CISC mais longos, em torno de 20 a 30 estágios. Isto porque estas abordagens de aceleração de execução devem ser adicionadas no processo de execução. Já os processadores RISC trabalham com estágios mais curtos, em torno de 4 a 10 estágios.

Os processadores CISC também utilizam mais memória principal e Cache, enquanto que os processadores RISC utilizam mais registradores. Isso porque os processadores CISC trabalham com um maior volume de instruções e dados simultaneamente. Esses dados não poderiam ser armazenados em registradores, devido à sua elevada quantidade e são, geralmente, armazenados em memória Cache. Enquanto que os processadores RISC trabalham com menos instruções e dados por vez, o que possibilita a utilização predominante de registradores.

Qual a melhor: RISC ou **CISC**

Processadores RISC geralmente resultam em projetos menores, mais baratos e que consumem menos energia. Isso os torna muito interessante para dispositivos móveis e computadores portáteis mais simples. Já os processadores CISC trabalham com *clock* muito elevado, são mais caros e mais poderosos no que diz respeito a desempenho. Entretanto, eles são maiores e consomem mais energia, o que os torna mais indicados para computadores de mesa e notebooks mais poderosos, além de servidores e computadores profissionais.

Os processadores CISC iniciaram com processadores mais simples e depois foram incorporando mais funcionalidades a esses processadores. Os fabricantes, como a Intel e a AMD precisavam sempre criar novos projetos, mas mantendo a compatibilidade com as gerações anteriores. Ou seja, o Conjunto de Instruções executado pelo 486 precisa também ser executado pelo Pentium para os programas continuassem compatíveis. O Pentium IV precisou se manter compatível ao Pentium e o Duo Core é compatível com o Pentium IV, mantido ainda compatibilidade nas recentes gerações i3, i5 e i7.

Isso tornou o projeto dos processadores da Intel e AMD muito complexos, mas não pouco eficientes. Os computadores líderes mundiais em competições de desempenho computacional utilizam processadores CISC.

Já o foco dos processadores RISC está na simplicidade e previsibilidade. Além do benefício da previsibilidade do tempo de execução ao Pipeline, ele também é muito interessante para aplicações industriais. Algumas dessas aplicações são chamadas de Aplicações de Tempo Real. Essas aplicações possuem como seu requisito principal o tempo para realizar as tarefas.

Assim, o Sistema Operacional precisa saber com quantos milissegundos um programa será executado. Isso só é possível com processadores RISC, com poucos estágios de Pipeline, poucos tipos de instrução, execução em ordem etc.

Mesmo que os processadores RISC sejam mais lentos do que os CISC, eles são mais utilizados nessas aplicações críticas e de tempo real, como aplicações industriais, de automação e robótica.

RISC	**CISC**
Múltiplos conjuntos de registradores, muitas vezes superando 256	Único conjunto de registradores, tipicamente entre 6 e 16 registradores
Três operandos de registradores permitidos por instrução (por ex., add R1, R2, R3)	Um ou dois operandos de registradores permitidos por instrução (por ex., add R1, R2)
Passagem eficiente de parâmetros por registradores no chip (processador)	Passagem de parâmetros ineficiente através da memória
Instruções de um único ciclo (ex. load e store)	Instruções de múltiplos ciclos
Controle hardwired (embutido no hardware)	Controle microprogramado
Altamente paralelizado (pipelined)	Fracamente paralelizado
Instruções simples e em número reduzido	Muitas instruções complexas
Instruções de tamanho fixo	Instruções de tamanho variável
Complexidade no compilador	Complexidade no código
Só instruções load e store podem acessar a memória	Muitas instruções podem acessar a memória
Poucos modos de endereçamento	Muitos modos de endereçamento

- **Menor quantidade de instruções**: talvez a característica mais marcante das arquiteturas RISC, seja a de possuir um conjunto de instruções menor (todas também com largura fixa), que as máquinas que possuíam a arquitetura CISC, porém com a mesma capacidade. O nome da arquitetura RISC ou computadores com um conjunto reduzido de instruções.

Com o conjunto de instruções reduzido e cada uma delas tendo suas funções otimizadas, os sistemas possuíam um resultado melhor em questão de desempenho. Em virtude do conjunto reduzido das instruções, acarretavam em programas um pouco mais longos.

- **Execução otimizada de chamadas de função**: outra evolução da arquitetura RISC para a arquitetura CISC tem relação com a chamada de retinas e passagem de parâmetros. Estudos indicam que as chamadas de funções consomem um tempo significativo de processador. Elas requerem poucos dados, mas demoram muito tempo nos acessos a memória.

Em virtude disso, na arquitetura RISC foram utilizados mais registradores. As chamadas de função que na arquitetura CISC ocorriam com acessos à memória, mas na RISC isso era feito dentro do processador mesmo, utilizando os registradores que foram colocados a mais.

- **Modo de execução com Pipelining**: uma das características mais relevantes da arquitetura RISC é o uso de pipelining, mesmo sabendo que ela tem um funcionamento mais efetivo quando as instruções são todas bastante parecidas.

Estágios de uma linha de montagem, não são interessantes que um estágio termine antes do outro, pois nesse caso perde-se a vantagem da linha de montagem. O objetivo de cada instrução é completar um estágio de pipeline em um ciclo de clock, mas esse objetivo nem sempre é alcançado.

O processamento de uma instrução é composto pelo menos por cinco fases:

- Instruction fetch;
- Instruction decode;
- Operand fetch;
- Execution;
- Write back.

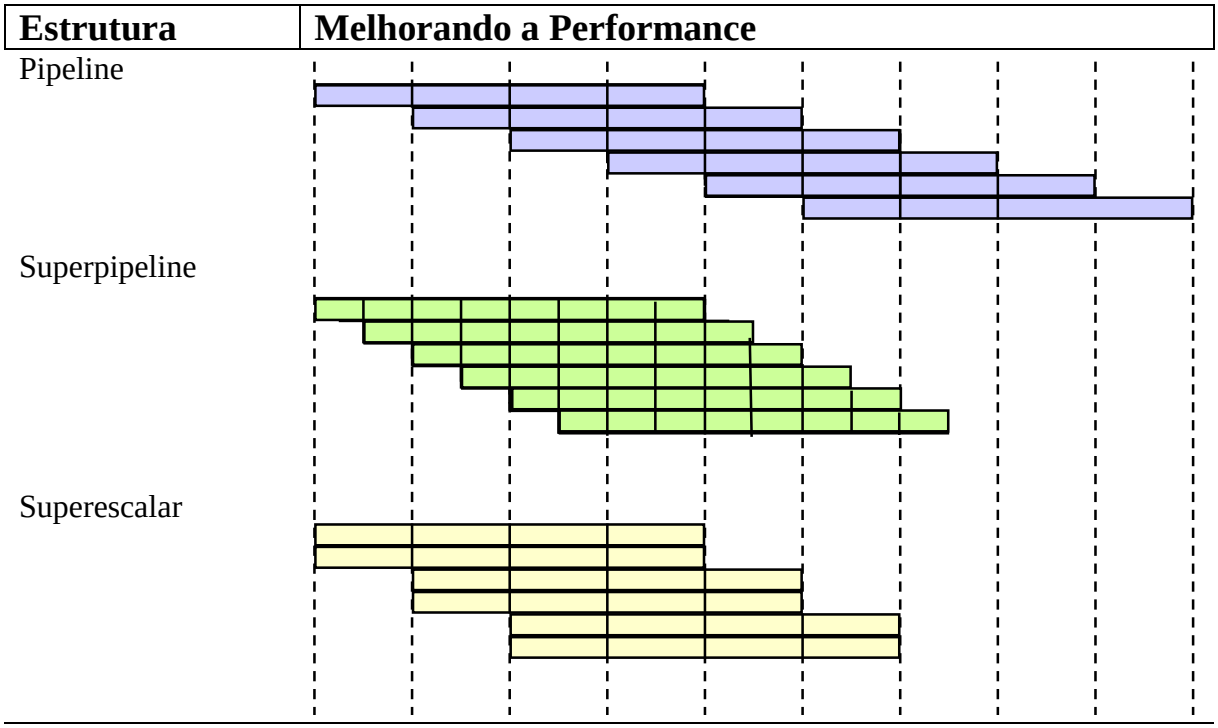
Atualmente o pipeline não se limita a apenas 5 estágios, mas pode chegar a 20 ou 30 estágios (Intel Pentium 4). No entanto, para que todo o processo funcione é necessário que determinadas restrições se verifiquem. A prioridade é que todas as instruções permaneçam em cada estágio o mesmo tempo, para que:

- O sinal de relógio seja usado como cadência de processamento;
- Não sejam necessários buffers.

- **Execução de cada instrução em um ciclo de clock**: se o uso do pipelining se considera uma característica importante da arquitetura RISC, a execução de uma instrução por ciclo de clock é mais importante, segundo os que estabeleceram suas bases. Um dos pontos mais negativos das arquiteturas RISC é o longo tempo de execução de cada instrução. Com o surgimento dessa nova arquitetura, cada instrução passou a ser executada a cada ciclo de clock.

Como melhorar a performance de máquinas implementadas como pipeline?

- Aumentar o número de estágios do pipeline: Superpipeline
- Replicar recursos para executar instruções em paralelo: Superescalar

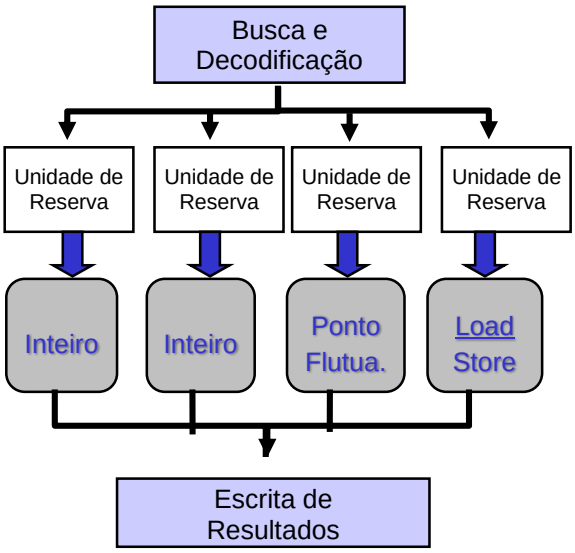


Melhorando a Performance – Superpipeline

- Limitações no tamanho do pipeline:
  1) Hazards de dados: pipeline maior ➔ mais paradas
  2) Hazards de controle: pipeline maior ➔ saltos mais lentos
  3) Tempo dos registradores do pipeline: Limita o tempo mínimo por estágio (clock)
- Superescalares:
  1) Execução simultânea de instruções: aritméticas, loads, stores, etc
  2) Aplicável a máquinas RISC e CISC:

- RISC: melhor uso efetivo
- CISC: implementação mais difícil

Processador Superescalar



Modos

Despacho:
Em ordem

Execução:
Fora de ordem

Escrita:
1) Em ordem
2) Fora de ordem.

Very Long Instruction Word

O compilador descobre as instruções que podem ser executadas em paralelo e agrupa-las formando uma longa instrução que será despachada para a máquina.

Comparação entre Arquiteturas

	Superescalar	VLIW
Detecção de paralelismo	Hardware	Compilador
Tempo disponível para realizar a detecção	Pouco	Muito
Relógio (clock)	Mais Lento	Mais Rápido
Arquitetura	CISC	RISC

Barramento

Bus é um conjunto de condutores elétricos em um computador que permite a comunicação entre vários componentes do computador, tais como; CPU, memória, dispositivos de I/O.
Que sinais trafegam no barramento?

- Dados;
- Relógio;
- Endereços;
- Sinais de controle.

Bus Standard (protocolo) é um conjunto de regras que governam **como** as comunicações no barramento serão efetuadas.

Vantagens

- Baixo custo na comunicação entre componentes, desde que um simples conjunto de fios é compartilhado em múltiplo sentidos;
- Versatilidade, que permite a fácil adição de novos dispositivos no computador.

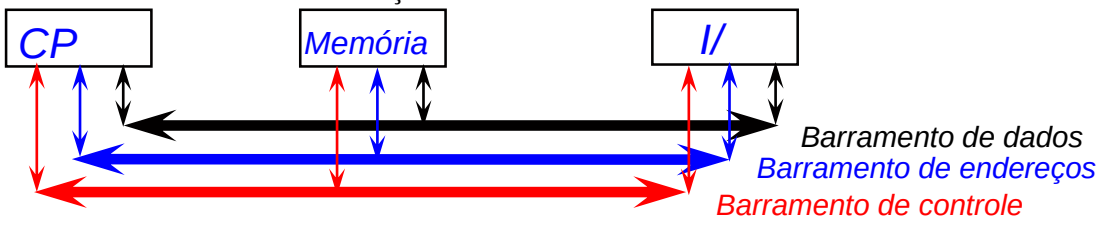
Desvantagens

- Criação de engarrafamento (**bottleneck**) na comunicação, limitando a máxima vazão de dados (**throughput**) para dispositivos de I/O.

Quanto à Funcionalidade

- Linhas de dados (barramento de dados) - fornecem o meio de transmissão de dados entre os módulos do sistema.

- Linhas de endereço (barramento de endereços) - usadas para designar fonte e destino dos dados do barramento de dados.
- Linhas de controle (barramento de controle) - usadas para controlar o acesso e o uso de linhas de dados e endereços.



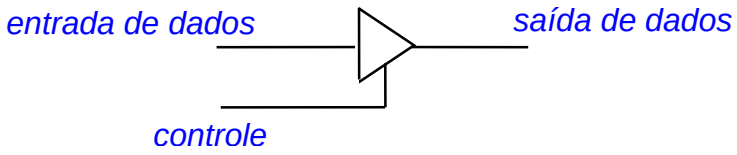
Sinais de controle típicos

- **Memory Write** - Causa a escrita de dados do barramento de dados no endereço especificado no barramento de endereços.
- **Memory Read** - Causa dados de um dado endereço especificado pelo barramento de endereço ser posto no barramento de dados.
- **I/O Write** - Causa dados no barramento de dados serem enviados para uma porta de saída (dispositivo de I/O).
- **I/O Read** - Causa a leitura de dados de um dispositivo de I/O, os quais serão colocados no barramento de dados.
- **Bus request** - Indica que um módulo pede controle do barramento do sistema.

  Reset - Inicializa todos os módulos

Características de acesso

- Todo dispositivo de memória ou I/O deve ser exclusivo no acesso ao barramento.
- A seleção é feita através de sinais especiais de controle como:
  - Memory Read
  - Memory Write
  - I/O Read
  - I/O Write
- Todo dispositivo deve escrever no barramento através de “buffers Tristate”.



Dispositivos

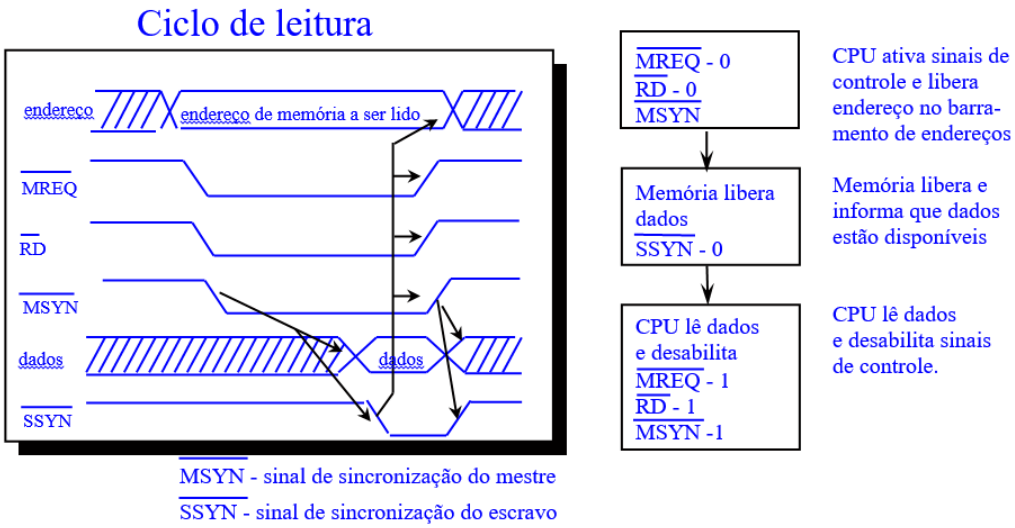
- **Ativos ou Mestres** - dispositivos que controlam o protocolo de acesso ao barramento para leitura ou escrita de dados;
- **Passivos ou Escravos** - dispositivos que simplesmente obedecem à requisição do mestre.

Exemplo:

- CPU ordena que o controlador de disco leia ou escreva um bloco de dados.
- A CPU é o mestre e o controlador de disco é o escravo.

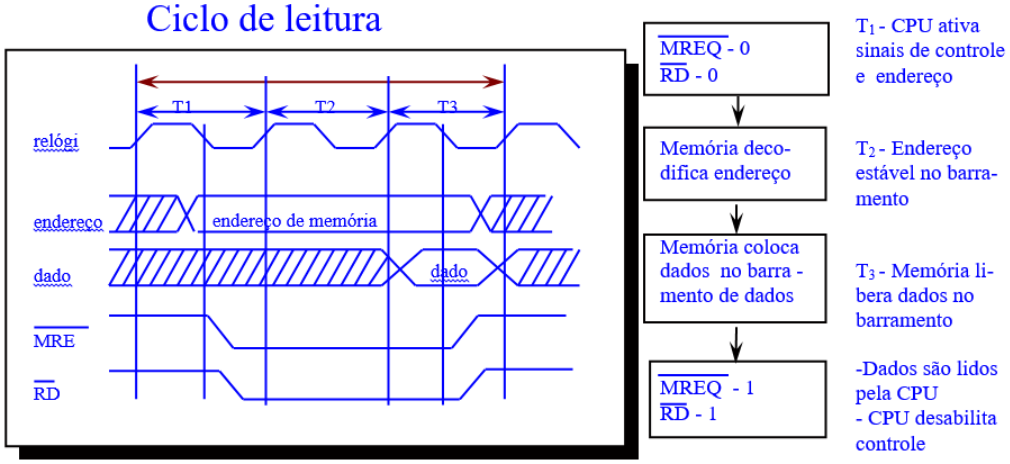
Quanto à temporização

- Barramento assíncrono
  - O controle ocorre exclusivamente por meio de sinais trocados entre os dispositivos. Os ciclos de barramentos podem ter qualquer duração e não precisam ser iguais para todas as situações.
  - São barramentos mais rápidos que os síncronos.



Barramento Assíncrono

- Barramento síncrono
  - Este tipo de barramento exige que todo o tráfego de dados e controle seja sincronizado sob uma mesma base de tempo chamado de relógio (clock)



Barramento Síncrono

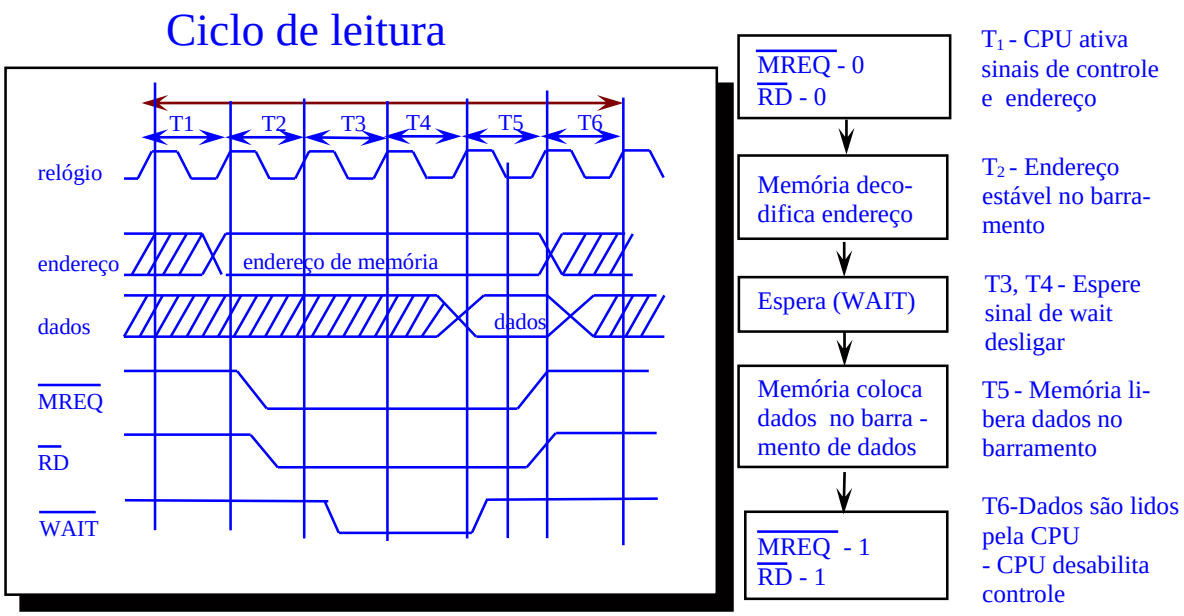
Barramento síncrono:
Incluindo wait-states

- Usando um sinal extra (*wait*) este barramento pode se comportar como um misto de síncrono e assíncrono.
- Sempre que o dispositivo escravo não puder responder no tempo padrão do barramento, este liga o sinal de wait para fazer com que o mestre pare o protocolo. Quando o escravo puder prosseguir, desliga o wait.

Arbitragem

Quando dois ou mais dispositivos querem se tornar mestres do barramento ao mesmo tempo?

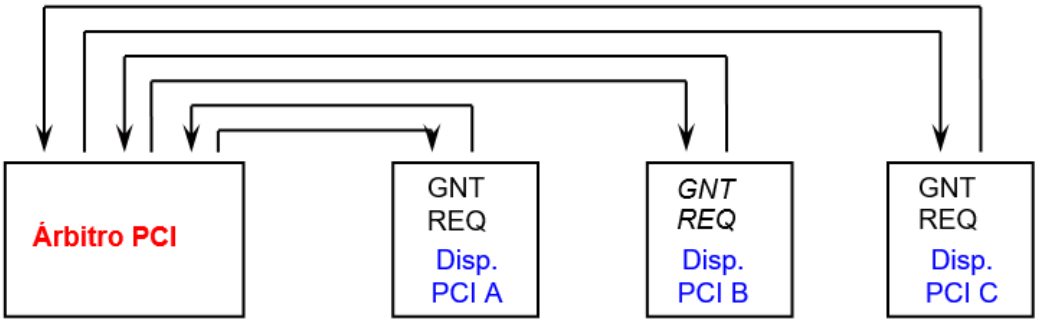
- Pode existir uma inviabilidade de operações (caos) do sistema se não houver um mecanismo adequado de arbitragem do barramento.
- A arbitragem decide qual mestre terá o controle do barramento num dado instante: **Arbitragem centralizada** ou **Arbitragem descentralizada**.



Arbitragem Centralizada

Arbitragem no barramento PCI (centralizado)

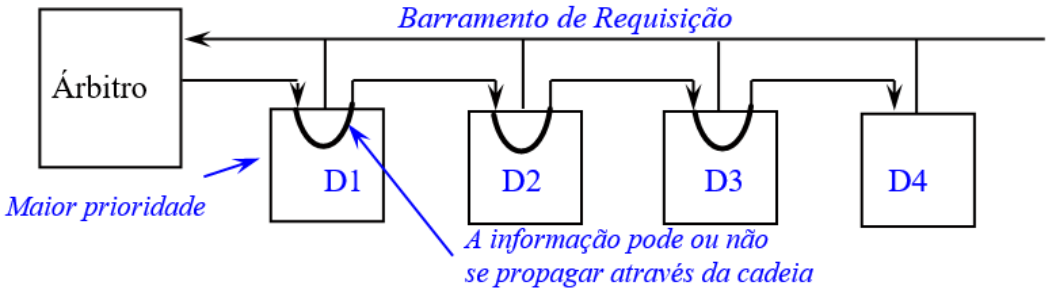
- Exemplo para três dispositivos
- O árbitro decide qual mestre controla o barramento



Arbitragem Híbrida

Este tipo de arbitragem é gerenciado por um árbitro que, juntamente com um daisy chain, estabelece a ordem de acesso ao barramento.

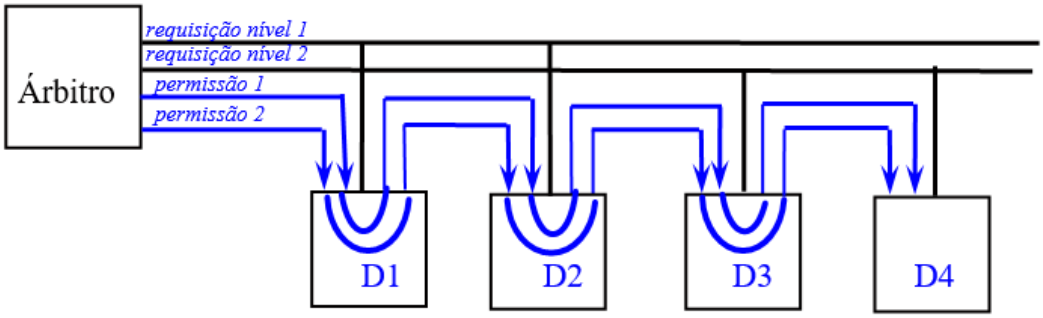
- **Barramento de um nível usando daisy chaining**



Características

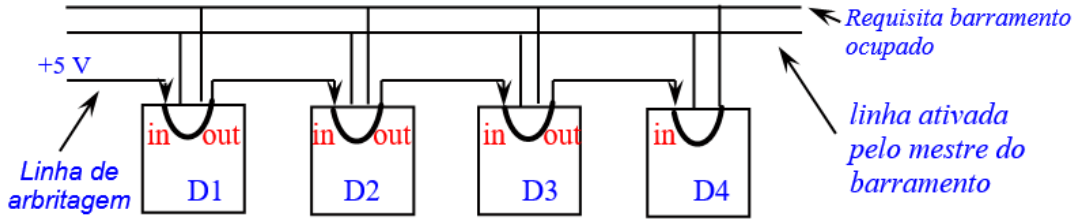
1. Todos os dispositivos são ligados em série, assim a permissão, dada pelo árbitro, pode ou não se propagar através da cadeia.
2. Cada dispositivo deve solicitar acesso ao barramento.
3. O dispositivo mais próximo do árbitro tem maior prioridade.

- **Arbitragem com dois níveis de prioridade**



Barramento descentralizado é o barramento Multibus - Daisy Chain sem árbitro
Características:

1. Quando nenhum dispositivo quer barramento, a linha de arbitragem ativada é propagada através de todos os dispositivos.
2. Para obter o barramento um dispositivo primeiro verifica se o barramento está disponível, e se a linha de arbitragem que está recebendo, **in**, está ativada.
3. Se **in** estiver desativada, ela não poderá tornar-se mestre do barramento.
4. Se **in** estiver ativada, o dispositivo requisita o barramento, desativa **out**, o que faz com que todos os dispositivos seguintes na cadeia desativem **in** e **out**.
5. O tempo de propagação do sinal **in** do primeiro dispositivo (D1) ao **out** do último dispositivo (D4) tem que ser menor que 1 período de clock.



Barramento Multibus

- O dispositivo mais próximo do início da cadeia que requer o barramento tem maior prioridade. Este esquema é similar ao sistema híbrido com *daisy chain*, exceto por:
  - Não existe mais a figura do árbitro
  - É mais rápido
  - Não vulnerável a falhas do árbitro
  - O barramento Multibus também oferece arbitragem centralizada, permitindo que os projetistas façam a escolha.

Quanto aos dispositivos a ele acoplados

Barramentos de Memória CPU-Memory Buses

- São pequenos
- Operam em alta velocidade
- São em geral conectados diretamente a CPU para maximizar a largura de banda entre memória e CPU (bandwidth)
- Tipos de dispositivos são conhecidos

Barramentos de Entrada e Saída CPU-I/O Buses

- Podem ser longos.
- Podem ter diferentes tipos de dispositivos conectados a ele.
- São, em geral, mais lentos que os barramentos de memória.