Unidade 6: Memórias e Barramento de Sistema

Prof. Daniel Caetano

Objetivo: Compreender os tipos de memória, como funcionam e a estrutura de comunicação dos sistemas computacionais modernos.

INTRODUÇÃO

Nas aulas anteriores foram apresentadas diversas maneiras de interpretar os bits na memória; essa compreensão é de extrema importância, mas não responde às perguntas: o que é, como funciona e como é acessada a memória?

O objetivo desta aula é apresentar uma introdução sobre os diferentes tipos de memórias existentes no computador, além de apresentar a forma com que a memória - e outros dispositivos - são acessados, através do **barramento de sistema**.

1. O QUE É A MEMÓRIA?

Em palavras simples, a memória é um dispositivo físico capaz de armazenar e recuperar uma configuração elétrica em um "conjunto de fios". Uma vez que essa configuração elétrica estabelece um padrão de bits, ligados ou desligados, é possível dizer que a memória armazena e recupera **dados**.

Olhando como uma caixa preta, a memória é bastante simples. Observe a Figura 1.

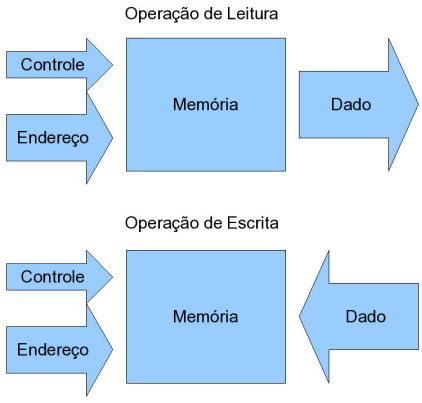

Figura 1: Operações de leitura e escrita na memória

Simplificadamente, o dispositivo memória recebe sinais de controle - que indicam se a operação é de leitura ou escrita na memória - e um endereço. Se a operação for de leitura, o

dispositivo memória responde emitindo o dado armazenado no endereço em questão; se a operação for de escrita, o dispositivo recebe o dado e o armazena na posição de memória indicada pelo endereço.

Apesar de ser um dispositivo de funcionamento aparentemente simples, as memórias são possivelmente os dispositivos com maior diversidade de implementações. Isso ocorre porque há diversas características que buscamos nas memórias como velocidade e capacidade, que não são atendidas plenamente por nenhum dos tipos de memória existente.

Há memórias que são rápidas, mas se forem desligadas perdem os dados armazenados e são muito caras; há memórias que são de velocidade média e possuem um preço razoável, mas se forem desligadas também perdem os dados. Há memórias muito baratas e que mantém a informação quando são desligadas, mas que são muito lentas...

Além disso, nem todas as memórias fornecem dados do "tamanho" que o processador "quer". O tamanho dos dados lidos pelo processador é denominado **palavra** e pode ter diferentes tamanhos, como 8, 16, 32, 64, 128bits... dentre outros. Isso é o que determina, em geral, a expressão "processador de 64 bits": o tamanho do dado que ele manipula diretamente. Algumas memórias simplesmente fornecem os dados em blocos maiores do que uma palavra, exigindo algum *"malabarismo*" para permitir seu uso direto com um dado processador.

Adicionalmente, para que se possa tirar máximo proveito de um sistema computacional, a velocidade da memória deve ser compatível com a velocidade do processador, de maneira que esse último não precise ficar esperando por respostas da memória por muito tempo. Em tese, considerando os processadores atuais, isso exigiria que toda a memória fosse muito rápida e, como consequência, os equipamentos seriam muito caros e praticamente não poderiam ser desligados.

Certamente esse não era um caminho viável e, por essa razão, cirou-se uma outra alternativa: usar diversos tipos de memória para obter o melhor desempenho ao menor custo.

2. HIERARQUIA DE MEMÓRIA

A quantidade de dados que um usuário medio armazena é gigantesca. Se considerarmos um servidor de uma grande empresa, essa quantidade de dados é ainda maior. Entretanto, a grande maioria desses dados **raramente** é usada pelo computador. Isso ocorre porque apenas um pequeno conjunto de programas e dados é usada rotineiramente.

Ainda assim, mesmo considerando os programas e dados que são processados com frequência, se medirmos a quantidade de tempo que o processador gasta com cada um destes bytes, veremos que a maior parte do tempo o computador está executando pequenos blocos de instruções e dados, realizando **tarefas repetidas**.

Observando este comportamento, os projetistas de *hardware* concluíram que poderiam equilibrar o custo de um equipamento se usassem um tipo de diferente de memória para cada tarefa:

a) Registradores e Memória Cache (Armazenamento Interno): Para armazenamento de curto prazo, de dados usados intensivamente pelo computador, adotam-se dispositivos de armazenamento volátil extremamente rápidos, mas de pequena capacidade devido ao **custo por bit ser muito alto**.

b) Memória Principal (Armazenamento Interno): Para armazenamento de médio prazo, de dados medianamente usados, adotam-se dispositivos de armazenamento volátil, cujo **custo por bit é médio**, proporcionando média capacidade com uma velocidade de acesso também média, já que estes dados são usados com alguma frequência.

c) Memória Secundária (Armazenamento Externo): Para armazenamento de longo prazo, de dados pouco usados, adotam-se dispositivos de armazenamento não volátil e cujo **custo por bit é baixo**, proporcionando grande capacidade, ainda que sejam lentos. A lentidão não é um problema, pois os dados aí contidos são pouco acessados.

d) Memória de Segurança (Armazenamento de Segurança): Para armazenamento de longuíssimo prazo, de dados que talvez nunca sejam necessários, adotam-se dispositivos de armazenamento não volátivo de **custo por bit extremamente baixo**, com enrome capacidade, ainda que extremamente lentos.

Estes quatro níveis formam a hierarquia de memória, lembrando que todos os dados úteis de um computador precisam estar armazenados na memória secundária, sendo transferidos para a memória principal na medida em que são necessários. Da mesma forma, os dados da memória principal são transferidos para o cache e para os registradores também na medida em que são necessários. A comunicação ocorre obedecendo a hierarquia, como pode ser visto na Figura 2.

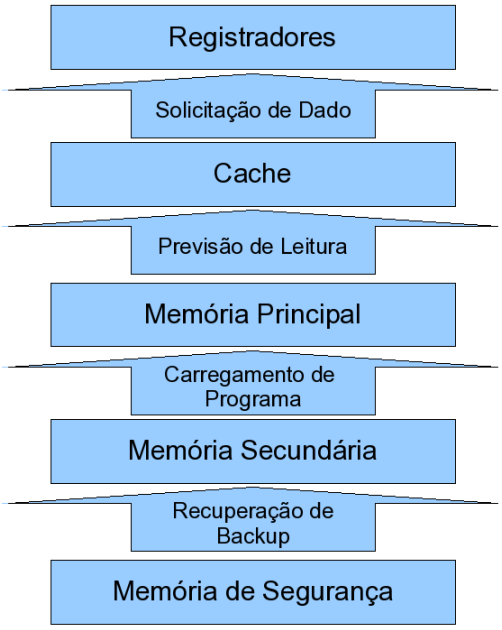

Figura 2: Situações de transferência de dados entre memórias (leitura)

3. TIPOS DE MEMÓRIA

As memórias utilizadas em cada uma destas camadas da hierarquia podem ser construídas com diferentes tecnologias. A diferenciação mais básica está entre os tipos RAM e ROM, mas existem diversos subtipos. Algumas delas estão descritas no quadro a seguir.

Tipo	Categoria	Apagamento	Escrita	Volatilidade	Palavra / Bloco	Interna / Externa	Velocidade	Usos	Custo por Bit
Memória de Acesso Aleatório Estática (SRAM)	Escrita e Leitura	Eletrica-mente	Eletrica-mente	Volátil	Bytes	Interna	Variada (pode ser tão rápida quanto o processador)	Registradores, cache, memória principal	De muito alto a alto
RAM Dinâmica (DRAM)	Escrita e Leitura	Eletrica-mente	Eletrica-mente	Volátil	Bytes	Interna	Média	Memória principal	Médio
Memória apenas de Leitura (ROM)	Apenas de leitura	Impossível	Máscaras	Não volátil	Bytes	Interna / Externa	Média para Rápida	Memória principal ou secundária	Baixo
ROM Programável (PROM)	Apenas de leitura	Impossível	Eletrica-mente	Não volátil	Bytes	Interna / Externa	Média para Rápida	Memória principal ou secundária	Baixo
PROM Apagável (EPROM)	Principal-mente de leitura	Luz UV	Eletrica-mente	Não volátil	Bytes	Interna / Externa	Rápida para leitura	Memória principal ou secundária	Baixo
EPROM Eletricamente apagável (EEPROM)	Principal-mente de leitura	Eletrica-mente	Eletrica-mente	Não volátil	Bytes ou Blocos	Interna / Externa	Rápida para leitura	Memória principal ou secundária	Médio
Memória Flash	Principal-mente de leitura	Eletrica-mente	Eletrica-mente	Não volátil	Blocos	Externa	Média para Lenta	Memória secundária	Médio
Disco Magnético	Escrita e Leitura	Magneti-camente	Magneti-camente	Não volátil	Blocos	Externa	Lenta	Memória secundária	Baixo
Disco Óptico	Leitura (e, opcio-nalmente, Escrita)	Óptica	Óptica	Não volátil	Blocos	Externa	Muito lenta	Memória secundária e de segurança	Muito baixo
Fita Magnética	Escrita e Leitura	Magneti-camente	Magneti-camente	Não volátil	Bytes	Externa (no passado, interna)	Extremamente Lenta	Memória secundária e de segurança (no passado, principal)	Extrema-mente baixo

Das memórias voláteis, qual a diferença entre a memória SRAM e DRAM? Bem, a SRAM é um dispositivo que basta estar ligado para preservar seus dados; a DRAM, por outro lado, exige que de tempos em tempos seja feita uma "simulação de leitura" em cada região da memória, para garantir que ela não seja perdida, em um processo chamado ***refresh***.

Como a maioria das memórias permite apenas **um acesso** por vez, isto é, ela só permite que um endereço de memória seja acessado de cada vez, durante o momento em que o ***refresh*** está sendo executado a memória DRAM fica **indisponível** para o processamento e, por essa razão, ela acaba sendo, no geral, mais lenta que a SRAM.

Projetar circuitos com SRAM é muito mais simples do que com DRAM; entretanto, a diferença de preço entre ambas faz com que, em geral, se a velocidade de uma SRAM não é necessária, os projetistas adotem o uso de DRAMs.

4. ACESSO A MEMÓRIA

Desde o início do curso é comentado que as partes do computador se comunicam por "fios". De fato, é isso que ocorre, embora esses "fios" muitas vezes sejam trilhas minúsculas em uma placa de circuito impresso.

Por exemplo, no caso da memória, apresentamos na Figura 1 "setas" que indicavam o fluxo de informações para a memória. Na prática, essas "setas" são fios ou trilhas, pelos quais trafegam sinais elétricos: a presença de sinal é interpretada como "1" e a ausência de sinal é interpretad como "0", formando os padrões desejados. Veja o exemplo da Figura 3.

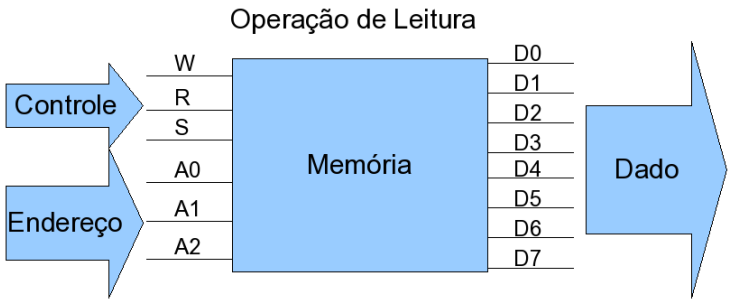

Figura 3: Nomenclatura dos "fios" que ligam a memória ao sistema

Nesta figura, os "fios" do controle foram nomeados de W (Write), R (Read) e S (Select). Estes fios controlam o funcionamento da memória. Sempre que a memória estiver sendo usada, S (ou MREQ) deverá estar com sinal em 1, indicando que a memória está selecionada. Quando S (ou MREQ) estiver em 1, os valores de W e R indicarão qual é a operação solicitada.

Quando a memória for usada para escrita, o sinal de W deve estar em 1 e R em 0; quando a memória for usada para leitura, R deverá estar em 1 e W em 0. O comportamento se S (MREQ), R e W estiverem todos simultaneamente em 1 é indefinido.

Os "fios" de endereço foram nomeados de A0 a A2 (A vem de Address). Isso significa que essa memória tem 3 bits de endereçamento, permitindo acesso a 8 dados (2^3). Em outras palavras, **temos 8 posições de memória**.

Os "fios" para representar os dados foram nomeados de D0 a D7 (D vem de Data). Isso significa que cada uma das posições de memória tem 8 bits (1 byte), proporcionando o armazenamento de 256 valores (2^8) diferentes em cada posição de memória.

Assim, como temos 8 posições de 1 byte cada, esta é uma memória de 8 bytes.

Vamos analisar agora como funcionaria a memória no momento de uma leitura. Num primeiro momento, o circuito do computador irá indicar nos "fios" do controle que ele deseja ler a memória e, nos "fios" de endereço, vai indicar que deseja ler um determinado endereço.

Suponhamos que o endereço a ser lido seja o endereço 6, ou seja, 110b. Observe a configuração na Figura 4.

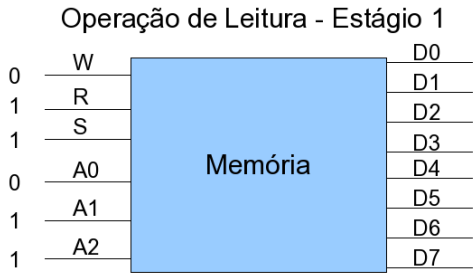

Figura 4: Configuração para a leitura do endereço de memória 6 (110b)

O circuito do computador coloca essa configuração elétrica nos "fios" de entrada da memória e, alguns instantes depois, a memória configura eletricamente os "fios" de dados (D0 a D7) com a informação que nela está armazenada, conforme indicado na Figura 5.

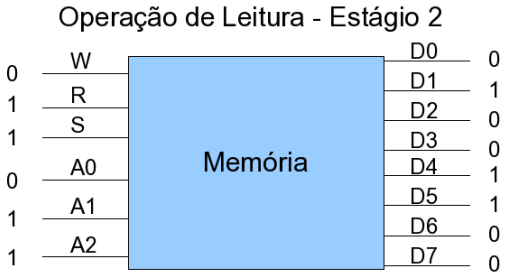

Figura 5: Resposta da memória à leitura do endereço de memória 6 (110b)

Ou seja: o dado armazenado na posição 6 da memória é: 00110010b, ou, interpretando como um decimal inteiro, 50.

Por terem funções **muito** distintas, cada um destes conjuntos de fios recebem nomes específicos. Os "fios" que controlam os dispositivos ligados ao computador são chamados de **barramento de controle**. Os "fios" que configuram endereços de memória e outros dispositivos são chamados de **barramento de endereços** e, finalmente, os "fios" que servem para a troca de dados entre os vários dispositivos são chamados de **barramento de dados**.

O exemplo desta aula foi feito com uma unidade de memória mas, de maneira geral, o procedimento é parecido para qualquer dispositivo que seja ligado no sistema computacional moderno.

Isso ocorre porque os computadores são projetados segundo um paradigma de arquitetura denominado "barramento de sistema", que será visto com mais detalhes a seguir.

5. BARRAMENTO DE SISTEMA

O Modelo de Barramento de Sistema é composto de três componentes:

1. Unidade de Entrada e Saída - usada pela CPU para receber e fornecer dados (e instruções) ao usuário.
2. CPU - responsável por coordenar todo o funcionamento do computador.
3. Unidade de Memória - responsável por armazenar dados e instruções a serem utilizadas pela CPU.

Observe pela Figura 6 que agora todas as unidades estão interconectadas, permitindo assim algumas caraterísticas interessantes como uma unidade de entrada ler ou escrever na memória sem a necessidade de intervenção da CPU (característica chamada DMA - Direct Memory Access).

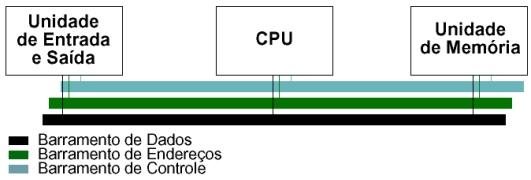

Figura 6: Modelo de Barramento de Sistema

Cada barramento consiste de um determinado conjunto de fios interligando os componentes do sistema. Quando dizemos, por exemplo, que um sistema tem 32 bits no barramento de dados, isso quer dizer que existem 32 fios para transmitir dados, e eles estão interligando todas as unidades do computador.

Um computador moderno tem, normalmente, três barramentos:

1. **Barramento de Dados** - para transmitir dados.
2. **Barramento de Endereços** - para identificar onde o dado deve ser lido/escrito.
3. **Barramento de Controle** - para coordenar o acesso aos barramentos de dados e endereços, para que não ocorra conflitos (do tipo vários periféricos querendo escrever na memória ao mesmo tempo).

Neste modelo, para a CPU ler a memória, ocorre (simplificadamente) o seguinte mecanismo:

1. CPU verifica se há uso da memória por algum dispositivo. Se houver, espera.
2. CPU indica no Barramento de Controle que vai ler a memória (impedindo que outros dispositivos tentem fazer o mesmo). Isso faz com que a memória se prepare para receber um endereço pelo barramento de endereços.
3. CPU coloca no barramento de endereços qual é o endereço de memória a ler.
4. Memória lê barramento de endereços e devolve o dado no barramento de dados.
5. CPU lê o dado solicitado no barramento de dados.

Repare que tudo isso existe uma coordenação temporal enorme, já que os barramentos não são "canos" onde você joga a informação e seguramente ela chega do outro lado. Na verdade, a atuação é mais como um sinal luminoso: quem liga uma lâmpada precisa esperar um tempo para que a outra pessoa (que recebe o sinal) veja que a lâmpada acendeu. Entretanto, a pessoa que liga a lâmpada não pode ficar com ela acessa indefinidamente, esperando que o recebedor da mensagem veja a lâmpada. Se isso ocorresse, a comunicação seria muito lenta.

Assim, tudo em um computador é sincronizado em intervalos de tempo muito bem definidos. Estes intervalos são os **ciclos de clock**. Assim, quando a memória coloca um dado no barramento de endereços, por exemplo, ela o faz por, digamos, 3 ciclos de clock. A CPU precisa ler este dado nos próximos 3 ciclos de clock ou, caso contrário, a informação nunca será recebida pela CPU e temos um computador que não estará funcionando corretamente (um computador que não respeite essa sincronia perfeita não costuma sequer ser capaz de passar pelo processo de inicialização; quando a sincronia falha em algumas situações apenas, o resultado pode ser travamentos aparentemente sem explicação).

Como é possível notar, se a sincronia tem que ser perfeita e existe um controle, quando há um único barramento para comunicação (entrada e saída - relativo à CPU) de dados, há uma limitação: ou a CPU recebe dados ou a CPU envia dados, nunca os dois ao mesmo tempo. Em algumas arquiteturas específicas são feitos barramentos distintos - um para entrada e um para saída, de forma que entrada e saída possam ocorrer simultaneamente.

Além disso, o acesso a dispositivos pode ser de duas maneiras. Algumas arquiteturas exigem que os dispositivos sejam **mapeados em memória**, ou seja, para enviar uma informação a um dispositivo deste tipo, a CPU deve escrever em um (ou mais) endereço(s) de memória específico(s). Para receber informações do dispositivo, a CPU deve ler um (ou mais) endereço(s) de memória específico(s). Outras arquiteturas, mais flexíveis, possuem dois tipos de endereçamento: um endereçamento de memória e outro de entrada e saída (I/O). Neste caso, os dispositivos podem tanto ser mapeados em memória como **mapeados em portas** de I/O. O uso de mapeamento de dispositivos em portas de I/O permite que todo o endereçamento de memória esteja disponível, de fato, para o acesso à memória.

5.1. Arquitetura de Barramento Simples

Como já foi apresentado, em determinado momento os arquitetos de computador perceberam que seria interessante se todos os periféricos pudessem conversar entre si, sem intermediários. A primeira forma com que imaginaram isso ocorrendo seria através da interligação direta de todos os dispositivos entre si.

Entretanto, não demorou para perceberem que isso traria problemas sérios, como por exemplo a quantidade de fios necessários nas placas para interligar todos os circuitos. Por esta razão, criou-se a idéia de **barramento**, que é um caminho comum de trilhas (fios) de circuito que interligam simultaneamente diversos dispositivos, em paralelo.

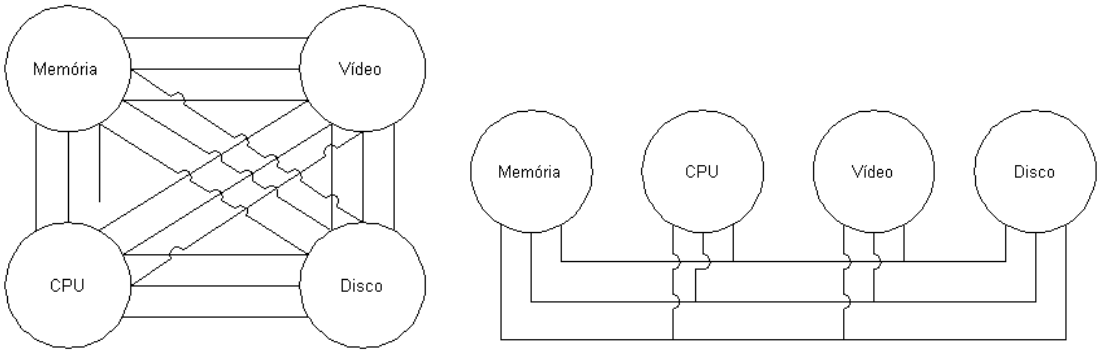

Figura 7: Elementos com ligação direta (esquerda) e através de barramento (direita)

O barramento é, fisicamente, um conjunto de fios que se encontra na *placa mãe* do computador, sendo um conjunto de fios que interliga a CPU (ou seu soquete) com todos os outros elementos. Em geral, o barramento (ou parte dele) pode ser visto como um grande número de trilhas de circuito (fios) paralelas impressas na placa mãe.

Quando se fala simplesmente em "barramento", na verdade se fala em um conjunto de três barramentos: o barramento de dados, o barramento de endereços e o barramento de controle. Alguns autores definem ainda o barramento de energia (cujas funções alguns incluem no barramento de controle).

Os barramentos de dados e endereços são usados para a troca de informações entre os diversos dispositivos e o barramento de controle é usado para a gerência do *protocolo de barramento*, sendo este o responsável por uma comunicação ordenada entre os dispositivos.

Este protocolo pode ser *síncrono* ou *assíncrono*. No caso do barramento síncrono, existe a necessidade de um circuito que forneça a cadência de operação, chamado de oscilador (ou relógio, ou *clock*, em inglês). Este dispositivo envia pulsos elétricos por um dos fios do barramento de controle, que podem ser entendidos como 0s e 1s, sendo estes pulsos utilizados por todos os outros dispositivos para *contar tempo* e executar suas tarefas nos momentos adequados. Observe na Figura 8 (MURDOCCA, 2000) o diagrama do sinal de *clock* em um barramento de 100MHz:

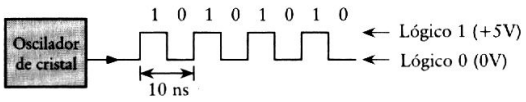

Figura 8: Sinal de *clock* (ao longo do tempo) de um barramento de 100 Mhz

Entretanto, a transição entre o sinal lógico 0 (0V) para 1 (5V) não é instantâneo e, na prática, é mais comum representar esta variação como uma linha inclinada, formando um trapézio.

O barramento freqüentemente opera a uma freqüência (*clock*) mais lenta que a CPU; além disso, uma operação completa (transação) no barramento usualmente demora vários ciclos deste *clock* de barramento. Ao conjunto dos ciclos de uma transação completa do

barramento chamamos de ***ciclo do barramento***, que tipicamente compreendem de 2 a 5 períodos de *clock*.

Durante um ciclo do barramento, sempre existe **mestre do barramento**, que é o circuito que está responsável pelo barramento de controle. Todos os outros circuitos estarão em uma posição de **escravos**, isto é, todos os outros circuitos *observarão* os barramentos de controle, endereços e dados e atuarão quando solicitados.

5.2. Barramento Síncrono

Os barramentos síncronos possuem algumas limitações, mas são os mais utilizados pela facilidade de implementação e identificação de problemas e erros (*debug*), como veremos mais adiante. O barramento síncrono funciona com base em uma temporização, definida pelo *clock do barramento* (ou relógio do barramento). O funcionamento do barramento fica sempre mais claro com um exemplo e, por esta razão, será usado o exemplo de uma operação de leitura de memória por parte da CPU.

O primeiro passo é a CPU requisitar o uso do barramento, ou seja, a CPU deve se tornar o circuito *mestre* do barramento. Por hora, será feita a suposição que a CPU conseguiu isso. Dado que ela é a mestra do barramento, a primeira coisa que ela fará, no primeiro ciclo T_1 desta transação, é indicar o endereço desejado no barramento de endereços (através do registrador MAR).

Como a mudança de sinais nas trilhas do circuito envolve algumas peças como capacitores, que demoram um tempo a estabilizar sua saída, a CPU aguarda um pequeno intervalo (para o sinal estabilizar) e então, ainda dentro do primeiro ciclo T_1, indica dois sinais no barramento de controle: o de requisição de memória (MREQ, de *Memory REQuest*), que indicará **com quem** a CPU quer trocar dados (neste caso, a memória), e o sinal de leitura (RD, de *ReaD*), indicando qual a operação que deseja executar na memória, no caso a operação de leitura.

Neste ponto, a CPU precisa esperar que a memória perceba que é com ela que a CPU quer falar (através do sinal MREQ) e que é uma leitura (através do sinal RD). Recebendo estes dois sinais, a memória irá usar o sinal do barramento de endereços para escolher um dado e irá colocar este dado no barramento de dados. Ocorre que a memória demora um certo tempo para fazer isso; por esta razão, a CPU espera em torno de um ciclo inteiro do barramento (T_2) e apenas no terceiro ciclo (T_3) é que a CPU irá ler a informação no barramento de dados (através do registrador MBR).

Assim que a CPU termina de adquirir o dado, ela desliga os sinais de leitura (RD) e de requisição de memória (MREQ), liberando o barramento de controle em seguida. Todo esse processo ao longo do tempo pode ser visto no diagrama da Figura 9.

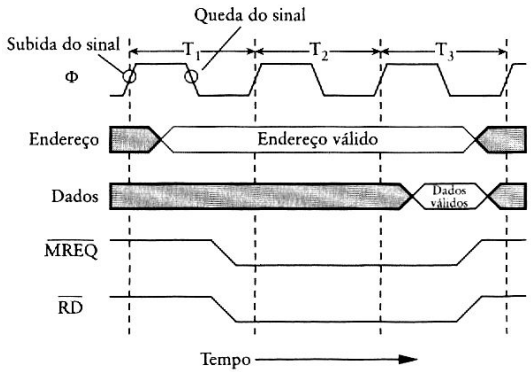

Figura 9: Um ciclo de barramento para leitura de memória
(fonte: Tanenbaum, 1999, apud Murdocca, 2000)

É importante ressaltar simbologia deste diagrama. A letra grega ϕ indica o sinal do *clock*. MREQ e RD possuem um traço em cima. Este traço indica que o acionamento ocorre quando a linha está em *zero*. Ou seja: se a linha MREQ estiver com sinal alto (1), a memória não responde. Quando MREQ estiver com o sinal baixo (0), a memória responderá. Ou seja: o traço em cima do nome indica que o acionamento do sinal é *invertido* com relação àquilo que se esperaria normalmente.

Anteriormente foi citado que este tipo de barramento tem uma limitação. Esta limitação é que seu protocolo é rigidamente ligado aos ciclos de *clock*. A CPU realiza sua operação em espaço de tempo pré-determinados e a memória precisa corresponder, em termos de velocidade. Se ela não corresponder, a CPU simplesmente lerá informações incorretas como se fossem corretas (qualquer "lixo" existente no barramento de dados no momento da leitura, em T_3, será lido como sendo um dado legítimo).

Por outro lado, não há ganho algum ao se substituir uma memória que atende aos requisitos de desempenho exigidos pela CPU por outra memória mais rápida: a CPU continuará demorando o mesmo número de ciclos de *clock* do barramento pera recuperar os dados: ela sempre irá esperar todo o ciclo T_2 sem fazer nada e irá ler a informação apenas no ciclo T_3 - mesmo que a memória seja rápida o suficiente para transmitir a informação já no ciclo T_2.

5.3. Barramentos Baseados em Pontes

Como dito anteriormente, uma forma de superar as limitações do barramento síncrono (que exige, por exemplo, que CPU e dispositivos operem sobre o mesmo *clock*), é através do uso de circuitos de compatibilização. Estes circuitos são chamados de *bridges* ou, em português, **pontes**.

Na arquitetura Intel atual, o circuito que faz essa intermediação primária é chamada *North Bridge*. O North Bridge é uma espécie de benjamim adaptador: é ligado na CPU através do Barramento Frontal (*Front Side Bus*), que trabalha na velocidade da CPU, e também é ligado ao Memory Bus, que trabalha na velocidade das memórias.

Em equipamentos com conector do tipo AGP (*Advanced Graphics Port*), o North Bridge também conecta a CPU e a Memória com o barramento de gráficos, que trabalha na velocidade admitida pela placa de vídeo (daí denominações do tipo 1x, 2x, 4x, 8x...).

Uma outra parte do North Bridge é ligado ao South Bridge, que faz a ponte com os barramentos dos conectores internos (slots), sejam eles PCI-X, PCI ou ISA, cada um deles trabalhando em uma freqüência (*clock*) específicos. Um esquema de uma arquitetura Intel recente pode ser visto na figura 11.

Na figura, o North Bridge está claro; o South Bridge, não (a figura sugere North e South Bridge integrados).

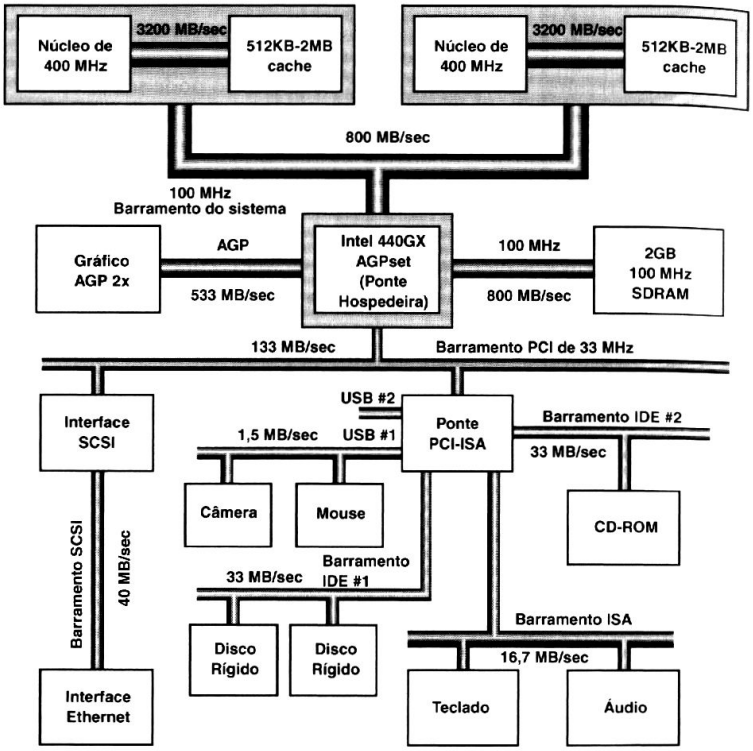

Figura 11: Arquitetura de Barramento em Pontes
(fonte: Intel apud Murdocca, 2000)

6. BIBLIOGRAFIA

MURDOCCA, M. J; HEURING, V.P. **Introdução à arquitetura de computadores**. S.I.: Ed. Campus, 2000.

STALLINGS, W. **Arquitetura e organização de computadores**. 5ed. São Paulo: Ed. Pearson Prentice Hall, 2003.