

Universidade de Passo Fundo

Instituto de Ciências Exatas e Geociências

Curso de Sistemas de Informação - CST

Organização de Computadores

Notas de Aula

Autor : Marcelo Trindade Rebonatto

E-mail : rebonatto@upf.tche.br

Página : <http://vitoria.upf.tche.br/~rebonatto>

SUMÁRIO

Lista de Figuras	4
Introdução.....	5
1 Conceitos sobre organização de computadores.....	6
1.1 Funcionamento básico dos sistemas de computação	6
1.2 Bits, bytes & Cia.	7
1.3 Registradores e memória	8
1.4 Barramentos	8
1.5 Interface CPU/Memória Principal	9
1.6 Clock.....	10
1.7 Taxa de transferência e multiplicador de frequência	11
1.8 Barramentos de expansão	12
1.8.1 Barramentos em microcomputadores	13
2 CPU	16
2.1 Componentes	17
2.1.1 Função Processamento	18
2.1.2 Função de Controle	19
2.1.3 Barramentos Internos	20
2.2 Ciclo de instrução	20
2.3 Medidas de Desempenho	22
2.4 Mais de uma instrução por ciclo	23
2.4.1 Pipeline	23
2.4.2 Unidades de execução especializadas	25
3 Memória	26
3.1 Organização e hierarquia da memória	27
3.2 Tempo e ciclo de acesso	28
3.3 Tecnologias de memória RAM	28
3.4 Módulos de memória	30
3.5 Memórias do tipo ROM	31
3.6 Memória cache	32
4 Unidades de discos e discos removíveis.....	36
4.1 Cabos e configurações físicas	36
4.2 Unidade de disquetes	37
4.3 H.D.s	37
4.3.1 Anatomia do H.D.	39
4.3.2 Desempenho	40
4.3.3 Padrões de conexão	41
4.4 Limites em HDs	42
4.4.1 Limite de 504Mb	42
4.4.2 Limite de 2Gb	43
4.4.3 Limite de 8Gb	43
4.5 CD-ROM	44
5 Entrada e Saída.....	45
5.1 Conexões de periféricos de entrada e saída padrões	46
5.1.1 Monitor	47
5.1.2 Teclado	47
5.1.3 Porta paralela	48
5.1.4 Portas seriais	49
5.2 Formas de comunicação entre CPU/MP e interface de I/O	50
5.3 Endereços de I/O para as portas de comunicação padrão	50

5.4 Formas de realização de entrada/saída	51
5.4.1 Entrada/saída por programa	51
5.4.2 Interrupções	52
5.4.3 Acesso direto à memória	54
6 Arquiteturas de computadores	56
6.1 CISC	56
6.2 RISC	57
6.3 ILP – Instruction Level Parallelism	58
6.3.1 Superscalares	58
6.3.2 VLIW - EPIC	59
6.4 CISC versus RISC versus VLIW	59
6.5 Paralelas	60
Bibliografia	62

Lista de Figuras

Figura 1 – Componentes básicos de um sistema de computação.....	6
Figura 2 – Forma geral das instruções	6
Figura 3 – Barramentos	9
Figura 4 – Interface CPU/MP.....	10
Figura 5 – Barramentos em um microcomputador.....	14
Figura 6 – Ciclo básico de instrução	16
Figura 7 – Componentes da CPU.....	17
Figura 8 – Ciclo detalhado de instrução.....	21
Figura 9 – Algoritmo para execução de instruções	21
Figura 10 – Linha de montagem	23
Figura 11 – Processadores superescalares.....	25
Figura 12 – Hierarquia de acesso à memória	27
Figura 13 – Hierarquia de memória	28
Figura 14 – EEPROM	32
Figura 15 – Transferência de dados entre CPU/Cache/MP.....	33
Figura 16 – Controladoras de unidades de disco.....	36
Figura 17 – Esquema de um HD	37
Figura 18 – Posição dos jumpers de configuração.....	38
Figura 19 – Visão interna de um HD	39
Figura 20 – MBR e partições	40
Figura 21 – Transferência de dados em HDs	41
Figura 22 – Características dos principais periféricos de I/O	45
Figura 23 – Conectores de cabos paralelos	48
Figura 24 – Conectores de cabos seriais	49
Figura 25 – Conectores min-DIN.....	49
Figura 26 – Formas de organização de memória para comunicação com interface de I/O	50
Figura 27 – Organização de computadores	61

Introdução

Somente os trabalhadores despreparados culpam seus instrumentos de trabalho pelo seu mau desempenho. Que tipo de trabalhadores não conhece seus instrumentos de trabalho, não sabem como funcionam e não distinguem um bom instrumento de trabalho de um ruim?

Muita gente ainda desconhece a composição e organização dos componentes que integram um instrumento vital para o seu trabalho: o computador. Hoje em dia, um bom profissional da área de informática, deve possuir sólidos conceitos sobre os componentes e a organização dos computadores. Mesmo o profissional que irá se dedicar 100% de seu tempo a trabalhar projetando, construindo, testando, validando, instalando, treinando ou qualquer outra tarefa em nível de software, deve, pelo menos, informar-se sobre o hardware. O trabalho em nível de software pode ser profundamente influenciado, pelo hardware, quer seja na produtividade, quer seja em seu planejamento; desta forma, os componentes físicos de um sistema computacional não podem ser desconhecidos a um profissional da área de informática.

Saber usar um computador, reconhecer e saber usar seus principais periféricos de entrada e saída pode ser suficiente para futuros profissionais, mas você estará subestimando seu potencial e a si próprio, além de tornar-se limitado. Sem saber sobre a organização e o funcionamento de seus principais componentes não saberá se adquiriu ou indicou o computador mais adequado à determinada função, ou, um equipamento caro incapaz de realizar tarefas nas quais os modelos mais simples se sobressaem.

“Os computadores são como automóveis. Você não precisa saber detalhes do seu funcionamento para utilizá-los, mas algum conhecimento ajuda. As pessoas inteligentes sabem selecionar o carro certo – ou componentes de hardware certo – para suas necessidades”.(Meyer, 2000, p. 53)

O texto de Meyer acima citado faz referência a usuários comuns de computadores. Uma comparação pode ser estabelecida apenas tendo o cuidado de ressaltar que profissionais da área de informática devem saber sobre o funcionamento interno e composição dos computadores. Este texto destina-se aos alunos da disciplina de organização de computadores para que conheçam os principais componentes dos computadores, como funcionam e se organizam.

1 Conceitos sobre organização de computadores

Para estudar organização de computadores alguns conceitos básicos devem ser assimilados/revisados. Estes conceitos abrangem os componentes funcionais básicos e as formas de representação/armazenamento de informações, além do funcionamento básico dos sistemas de computação.

1.1 Funcionamento básico dos sistemas de computação

Os computadores executam quatro funções distintas sendo elas: (a) Entrada; (b) Processamento; (c) Armazenamento/recuperação de dados; (d) Saída. A Figura 1 ilustra a organização dos sistemas de computação.

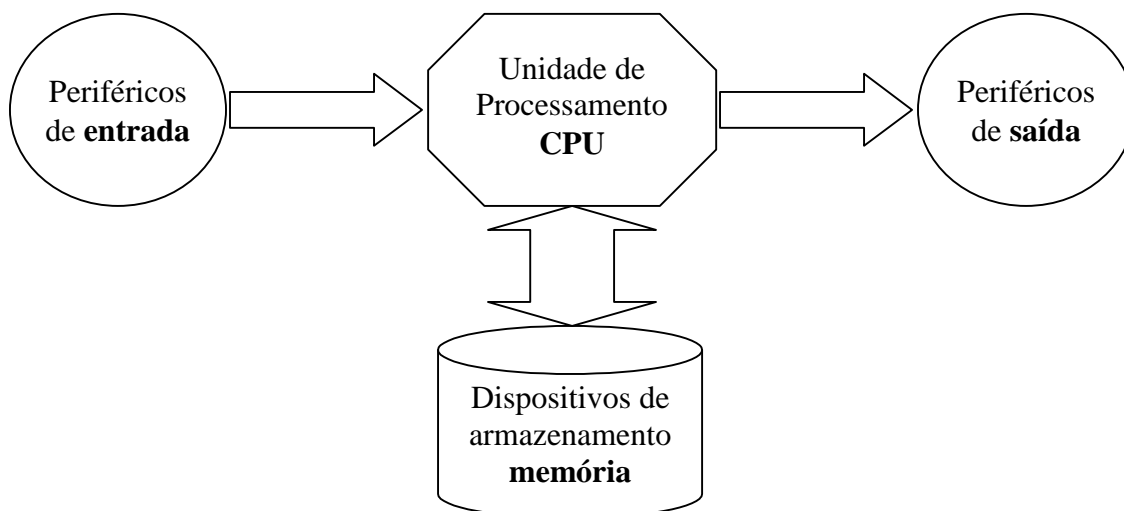


Figura 1 – Componentes básicos de um sistema de computação

Para que um computador trabalhe é necessária à inserção de informações (entrada). Seguindo as instruções fornecidas pelos programas, o computador processa os dados oriundos da entrada (processamento) armazenando-os logo em seguida para posterior utilização. As informações produzidas ficam disponíveis para utilização (saída) e a menos que se deseje as informar e produzir novamente, elas devem ser armazenadas em um dispositivo de armazenamento estável.

O esquema geral da figura 1 é seguido por praticamente todos os computadores, sendo que os dados são produzidos através de instruções durante a etapa de processamento, realizada pela CPU (Unidade Central de Processamento - processador). Cada processador tem um conjunto único de instruções para processar os dados, porém geralmente utilizam a mesma forma de composição das instruções. A Figura 2 mostra a forma das instruções comumente utilizada pelos processadores.

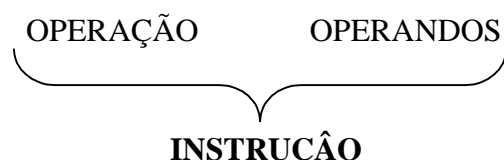


Figura 2 – Forma geral das instruções

A “operação” especifica a função a ser desempenhada, por exemplo, soma, armazene ou desvie, entre outras. Os “operandos” fornecem os dados a serem utilizados na operação ou ainda a forma de alcançar a posição destes dados na memória.

1.2 Bits, bytes & Cia.

Todas as informações manipuladas pelos computadores devem ser entendidas pela máquina. Como o computador é um dispositivo eletrônico, ele armazena e movimenta as informações de forma eletrônica, podendo utilizar um valor de corrente. Para que a máquina representasse todos os símbolos da linguagem humana eletricamente seriam necessárias mais de 100 diferentes voltagens (corrente). Uma máquina deste tipo além de ser de custo elevado, seria difícil de construir e de baixa confiabilidade. Desta forma, optou-se por construir máquinas binárias, capazes de entender apenas dois valores diferentes.

Os computadores digitais são totalmente binários, isto é, trabalham apenas com dois valores, tornando assim simples o emprego da lógica booleana (Sim/Não, Verdadeiro/ Falso, Aberto/Fechado,...) tanto na construção de componentes quanto como base para a escrita de programas (programação).

Convencionou-se chamar os dois níveis elétricos de 0 e 1 sendo que cada algarismo da representação numérica binária é denominado de **bit**, correspondente a abreviatura de *binary digit* (dígito binário).

Obviamente com apenas 1 bit isolado pode-se representar muito pouca coisa (apenas 2 valores), desta forma, usam-se agrupamentos ordenados de bits para a representação de informações úteis. A menor informação inteligível aos seres humanos é o **caractere**, como por exemplo, o número “5” ou a letra “a”. Existem diversos agrupamentos de bits para representar caracteres, sendo que o mais popularmente utilizado é chamado de **byte**. Um byte é uma sequência ordenada de 8 bits, sendo cada bit tratado de forma independente dos demais e com um valor fixo de acordo com sua posição. Qualquer sequência binária pode ser convertida para um número na base decimal, sendo utilizado este valor para encontrar o caractere correspondente, utilizando uma tabela de caracteres. As memórias geralmente armazenam e recuperam informações byte a byte, ou ainda em múltiplos de bytes.

A representação binária de valores também é utilizada para a representação de números dentro dos computadores. A metodologia é a mesma, sendo convertido o valor em base decimal para o correspondente em base binária. Por exemplo, o número 23_{10} pode ser armazenado no seguinte byte 00010111.

Os dispositivos de memória atuais utilizam agrupamentos de bytes para representar sua capacidade de armazenamento. Uma vez que tais agrupamentos são oriundos de uma base binária o fator de multiplicação utilizado é 1024 (2^{10}). Cada faixa possui também uma letra para abreviar a categoria. A Tabela 1 demonstra alguns agrupamentos de bits e bytes utilizados.

Tabela 1 – Agrupamento de bits e bytes

Agrupamento	Símbolo	Representa
Byte	B	8 bits
Kilo	K	1024 Bbytes
Mega	M	1024 KBytes
Giga	G	1024 MBytes
Tera	T	1024 GBytes
Peta	P	1024 TBytes

Estes agrupamentos são utilizados na descrição das capacidades de armazenamento dos computadores hoje em dia, sendo que esta capacidade é referenciada por um número e um símbolo correspondente. Por exemplo, 256MB de memória.

Outro conceito importante muito utilizado na descrição do mecanismo de transferência de informações entre a CPU e a memória principal é o conceito de **palavra**. A palavra é utilizada para

indicar a unidade de transferência e processamento de um computador. As palavras são múltiplos de 1 byte, sendo que os microprocessadores geralmente utilizam 32bits – 4 bytes como tamanho da palavra (já existem projetos e microprocessadores que utilizam palavras de 64 bits, porém estes microprocessadores ainda não se popularizaram). Máquinas de maior porte como estações SUN já utilizam palavra de 64 bits há algum tempo.

1.3 Registradores e memória

Os registradores são dispositivos que armazenam valores temporários principalmente dentro dos processadores. São utilizados para a realização das instruções que fazem uso de operações lógicas e aritméticas tanto para armazenar o resultado final quanto para obter os valores de entrada da operação. Os registradores também são utilizados para a recuperação/armazenamento dos valores na memória. Em outras palavras, são os locais onde se pode armazenar temporariamente um valor dentro da CPU. Encontram-se em número variado dependendo do modelo do processador utilizado.

A CPU não consegue manter todos os valores manipulados por um programa apenas em registradores, por isso necessita de uma memória para o armazenamento das informações. Somente ficam na memória os dados não manipulados atualmente pela CPU, existindo uma intensa troca de valores entre a memória e a CPU (Interface Memória – CPU). Além dos valores não utilizados em um determinado momento, a memória armazena também TODAS as instruções a serem executadas pela CPU.

A memória é um conjunto de células, todas com o mesmo número de bits, sendo cada célula identificada por um número único que é seu endereço. Os acessos à memória são realizados através de palavras de memória, sendo que a velocidade da memória, indicada pelo seu tempo de acesso, é significativamente inferior à velocidade com que o processador trabalha. Pode-se ler ou escrever dados em uma posição da memória.

1.4 Barramentos

Os processadores são circuitos integrados passíveis de serem programados para executar uma tarefa predefinida, basicamente manipulando e processando dados. A CPU manipula dados de acordo com programas, que deverão para isso, estar na memória. Um programa pode ordenar que dados sejam armazenados de volta na memória ou recuperar programas/dados armazenados em sistemas de memória de massa (disquetes, H.D.,...). Os caminhos por onde estas informações circulam em um computador é genericamente conhecido como barramento.

Em outras palavras, os barramentos nada mais são do que um conjunto de condutores (fios, trilhas) por onde passam os bits. Possuem duas principais características:

- A largura do barramento: número de bits transportados numa operação;
- A frequência de operação: velocidade com que os dados são transmitidos.

A Figura 2.3 ilustra a funcionalidade dos barramentos.

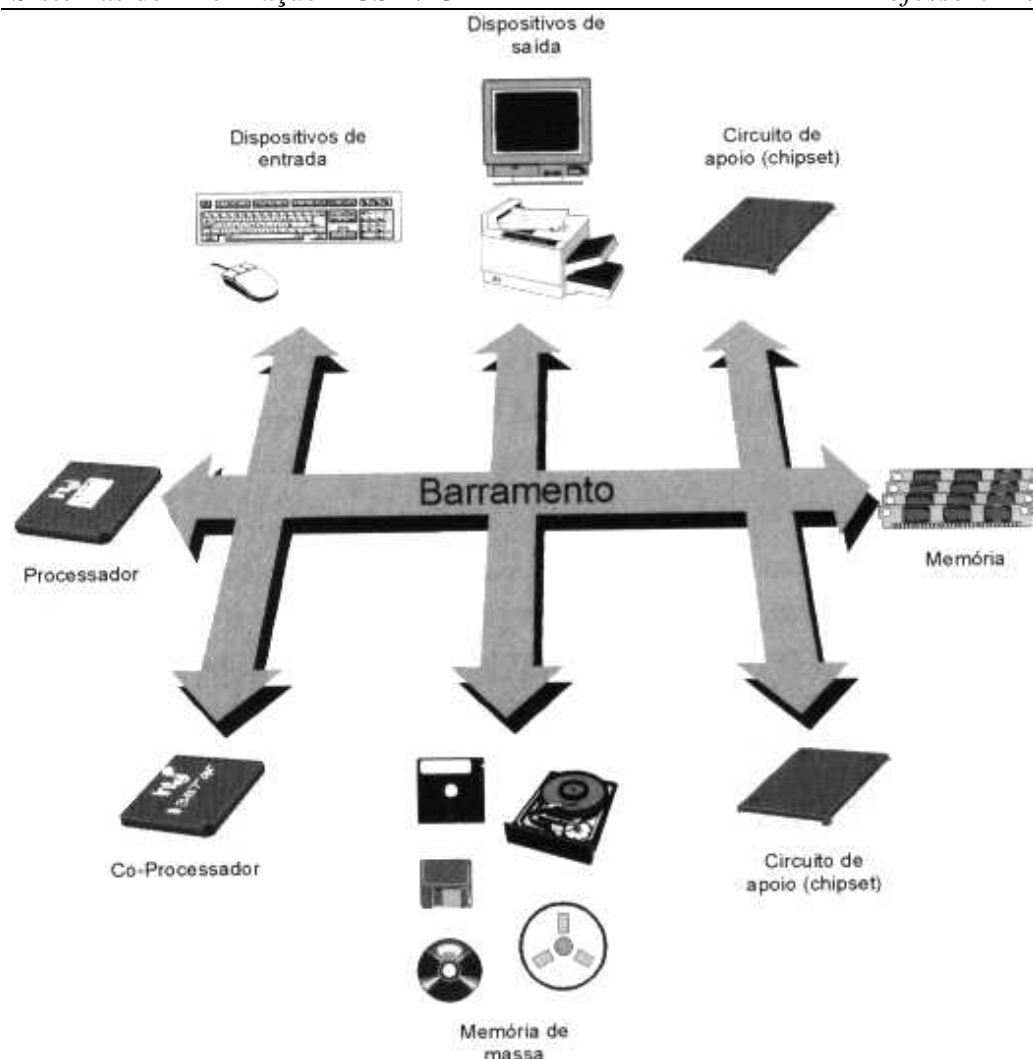


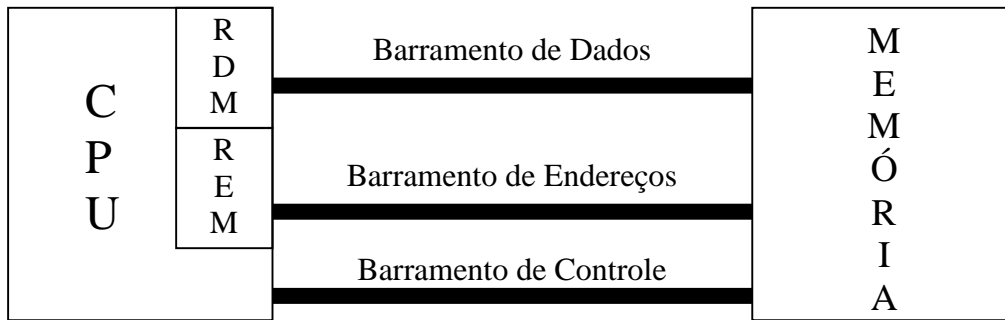
Figura 3 – Barramentos

Existem diversos barramentos nos computadores atuais, sendo os principais o barramento local e de expansão. O barramento local é vital para o funcionamento do computador, pois interliga o processador à memória. Por outro lado, o barramento de expansão interliga os demais componentes, tais como, periféricos de entrada, de saída e memória de armazenamento estável (secundária).

1.5 Interface CPU/Memória Principal

Conhecida genericamente como barramento local, a interface CPU/Memória principal é de vital importância para o funcionamento do computador, uma vez que os blocos que formam a CPU não podem manipular diretamente os dados armazenados na memória, pois somente operam sobre valores mantidos em registradores. Desta forma, como as instruções e os dados a serem manipulados estão na memória, para que uma instrução seja executada, a memória deve ser acessada no mínimo uma vez.

O barramento local é dividido em outros três: barramento local para dados, para endereços e de controle. A interface CPU/MP se completa com os registradores auxiliares no acesso à memória. A Figura 4, ilustra o esquema da interface CPU/MP.

**Figura 4** – Interface CPU/MP

Segue abaixo uma breve descrição dos componentes da interface CPU/Memória principal.

- RDM - Registrador de Dados da Memória: Armazena o endereço da célula onde deve ser feita a próxima operação de leitura ou escrita na memória;
- Barramento de Endereços: Liga o REM à memória para transferência do endereço da célula a ser lida ou escrita. Sua largura em bits deve ser igual ao REM;
- RDM - Registrador de Dados da Memória: Armazena os dados que estão sendo transferidos de/para a memória;
- Barramento de Dados: Liga o RDM à memória, possuindo a mesma largura deste. É o caminho por onde é feita a transferência do conteúdo;
- Barramento de Controle: Interliga a CPU à memória para enviar os comandos de *READ* e *WRITE* e receber *WAIT*.

1.6 Clock

De todos os sinais de informação que circulam pelo barramento, o mais importante é o *clock* que é um sinal de sincronismo, realizando a sincronização de todos os circuitos que constituem o micro. Todos os circuitos trocarão informações no momento em que o *clock* permitir, enviando um aviso aos componentes do micro como que dizendo “JÁ”.

Como os circuitos eletrônicos são rápidos, a frequência com que o *clock* fica ativo é alta. Normalmente esta frequência, também chamada frequência de operação, está na casa do MegaHertz (MHz), ou seja, milhões de vezes por segundo.

Todos os componentes do micro utilizam uma frequência de operação como os barramentos, a memória, o processador, entre outros, sendo que cada componente do micro trabalha a sua própria frequência. Os processadores atualmente utilizam técnicas de multiplicação de frequência aumentando assim sua frequência interna de processamento. Geralmente é divulgado apenas o *clock* interno do processador, ficando subentendido que é essa sua frequência de operação. Resta salientar que a frequência de operação **NÃO** é sinônima de desempenho do micro. Ela é apenas um fator que contribui no desempenho.

1.7 Taxa de transferência e multiplicador de frequência

A frequência de operação (clock) juntamente com a largura são os fatores determinantes do desempenho (taxa de transferência) dos barramentos. A largura do barramento local pode ser dividida em dados e instruções, porém para efeitos de cálculo da taxa de transferência é considerado um único valor, que a partir dos processadores Pentium, é de pelo menos 64 bits. A frequência de operação não pode ser confundida com o *clock* da máquina (normalmente associado ao *clock* do processador). A frequência de operação do barramento é normalmente divulgada através da frequência de operação da placa-mãe. A fórmula abaixo pode ser utilizada para calcular a taxa de transferência dos barramentos:

$$\text{Taxa de transferência} = \text{frequência de operação} \times \text{número de bits} / 8$$

A Tabela 2 mostra a taxa de transferência para diversas frequências de operação do barramento local, considerando processadores que acessam a memória local a 64 bits.

Tabela 2 – Barramentos e taxas de transferência

Frequência do barramento local	Taxa máxima de transferência
66 MHz	528 MB/s
100 MHz	800 MB/s
133 MHz	1064 MB/s
200 MHz	1600 MB/s
266 MHz	2128 MB/s
400 MHz	3200 MB/s

O barramento local é o mais rápido, pois os circuitos se comunicarão com o processador em seu desempenho máximo. Entretanto, o processador trabalha atualmente a uma frequência de operação superior a do barramento local, utilizando um recurso denominado de “multiplicação de frequência”. Para conhecer o fator de multiplicação de frequência utilizado por um micro, pode-se utilizar a fórmula abaixo:

$$\text{Multiplicador} = \text{frequência do processador} / \text{frequência do barramento local}$$

O valor divulgado como sendo a velocidade (frequência de operação) da máquina é o do processador, o que não está totalmente errado, pois é neste valor que as informações são processadas, porém ao avaliar o desempenho de um micro, deve-se levar em conta a frequência do barramento local. O barramento local não é padronizado: cada processador utiliza seu próprio modelo, e é por este motivo que cada processador necessita de um modelo de placa-mãe. A tabela 3 mostra os *clocks* de alguns processadores, juntamente com seus fatores de multiplicação.

Tabela 3 – Clock de processadores e fator de multiplicação

Processador	Clock		Fator de multiplicação
	Interno	Externo	
Pentium	75/125 MHz	50 MHz	1,5 / 2,5
	90/120/150/180 MHz	60 MHz	1,5 / 2 / 2,5 / 3
	100/133/166/200 MHz	66 MHz	1,5 / 2 / 2,5 / 3
AMD K6	166/200/233 MHz	66 MHz	2,5 / 3 / 3,5
Pentium II	233/266/300/333 MHz	66 MHz	3,5 / 4 / 4,5 / 5
	350/400/450 MHz	100 MHz	3,5 / 4 / 4,5
Celeron	366/400/500/566/600/700 MHz	66 MHz	5,5 / 6 / 7,5 / 8,5 / 9 / 10,6
	800/850/900 MHz	100 MHz	8 / 8,5 / 9
AMD K6-II	266/300/333 MHz	66 MHz	4 / 4,5 / 5
	300/350/400/450/500/533/550 MHz	100 MHz	3 / 3,5 / 4 / 4,5 / 5
AMD K6-III	350/400/450/500/550/600 MHz	100 MHz	3,5 / 4 / 4,5 / 5 / 5,5 / 6
Pentium III	450/500/550/600 MHz	100 MHz	4,5 / 5 / 5,5 / 6
	533/600/666/733/866/933 MHz / 1GHz	100/133 MHz	4 / 4,5 / 5 / 5,5 / 6,5 / 7 / 7,5
Duron	800/850/900/950 MHz	200 MHz	4 / 4,25 / 4,5 / 4,75
Athlon	1 / 1,1 / 1,2 / 1,3 / 1,4 GHz	200 MHz	5 / 5,5 / 6 / 6,5 / 7
	1,0 / 1,13 / 1,2 / 1,33 / 1,4 GHz	266 MHz	3,75 / 4,25 / 4,5 / 5 / 5,25
Pentium 4	1,4 / 1,5 / 1,6 / 1,7 / 1,8 GHz	400 MHz	3,5 / 3,75 / 4 / 4,25 / 4,5

Como as informações a serem manipuladas pelo processador são oriundas da memória, de nada adianta melhorar apenas a frequência de operação do barramento local. Deve-se verificar se a memória acompanha a velocidade do barramento local, a fim de evitar o estado de espera do processador pela memória (*wait state*), que diminui o desempenho.

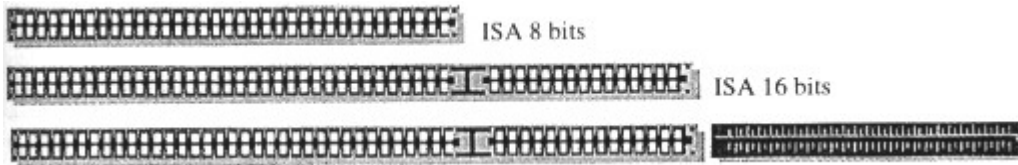
1.8 Barramentos de expansão

Para que uma simples placa de vídeo ou um H.D. possa ser utilizada em qualquer computador, independentemente do processador instalado, utiliza-se diversos modelos de barramentos de expansão. Os barramentos de expansão são disponibilizados na placa-mãe dos micros através de slots (exceção do USB, Firewire e IrDA, que são diretamente conectados a placa-mãe).

Os slots nada mais são do que encaixes para que as conexões entre placas presentes no sistema computacional utilizem determinados padrões de barramento. Na parte superior dos slots, encontram-se ranhuras para a conexão de placas de circuito que funcionam com a placa-mãe. Sem as ranhuras, os micros ficariam limitados aos circuitos que estivessem permanentemente montados na placa-mãe.

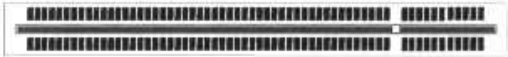
Os principais modelos são:

- **ISA (*Industry Standard Architecture*)**: origina-se dos primeiros computadores IBM PC-AT. Os slots ISA são ligados em paralelo, possibilitando que uma placa seja ligada em qualquer slot. Utiliza conectores de 62 e 96 pinos. Trabalha com uma largura de barramento de 8 ou 16 bits a uma frequência máxima de 8,33MHz;
- **VLB (*VESA Local Bus*)**: desenvolvido pelo grupo *Video Eletronics Standard Association*, para ser ligado ao barramento local dos 486. Trabalha na velocidade do processador, porém limitado a 3 slots. Compatível com o ISA adaptou-se bem a vídeo e disco rígido, entretanto como era muito direcionado ao 486, entrou em desuso com o declínio do 486. Utiliza conectores de 168 pinos, largura de barramento de 32bits a uma frequência máxima de 50MHz (limitado pelo barramento local);

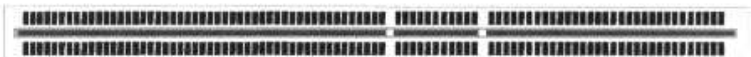



- **PCI (Peripheral Component Interconnect):** desenvolvido pela Intel para ser seu próprio padrão de barramento, “matou” o VLB. Diferente do VLB, não é ligado diretamente ao barramento local, mas através de uma ponte (*bridge* – parte do chipset). Não é compatível com o ISA, porém é totalmente independente de processador. Possui conectores menores e ranhuras mais densamente acondicionadas que as ISA. Em geral, trabalha com slots de 32bits (existindo versões de slots para 64bits), a uma frequência de 33MHz (usual) ou 66MHz;

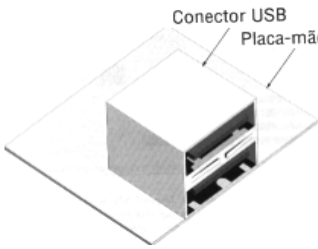
Slot PCI 32 bits




Slot PCI 64 bits




- **AGP (Accelerated Graphics Port):** conector projetado especialmente para vídeo, pela Intel, que permite a uma interface de vídeo a comunicar-se diretamente com a memória RAM do computador. Fisicamente o barramento AGP é conectado ao mesmo circuito que contém a ponte PCI-barramento local (norte). Utiliza largura de barramento de 32bits, operando a uma frequência máxima de 66MHz, porém possuindo 4 modos de operação: x1, x2, x4 e x8;
- **USB (Universal Serial Bus):** um barramento onde através de um plugue na placa-mãe pode-se ligar todos os periféricos externos (até 127). Passível de cascadeamento ou instalação de HUBs utiliza duas taxas de transferência: 12Mbps e 1,5Mbps. Para muitos, não é exatamente um barramento, mas uma porta serial de alta velocidade;




- **Firewire (IEEE 1394):** a idéia é semelhante a do USB, porém com diferente foco: pretende substituir o padrão SCSI (*Small Computer System Interface*). Com taxa de transferência maior que o USB (200 a 400Mbps), poderá num futuro próximo ser utilizado na conexão de discos rígidos;
- **IrDA (Infrared Developers Association):** é um barramento sem fios, onde a comunicação é feita através de luz infravermelha. Pode-se ter até 126 periféricos “conversando” em uma mesma porta. Comum em notebooks pode estar diretamente na placa-mãe ou conectado a porta serial. Existem dois padrões, com taxas de transferência de 115200bps e 4Mbps;

Junto com a evolução dos computadores, o desempenho dos barramentos também evolui. Novos barramentos e/ou melhorias nos atuais estão sempre surgindo. Convém lembrar que no desempenho de um computador o que conta não é apenas o desempenho do processador ou o barramento local (interligação com a memória). Esses são fatores de extrema importância, porém todos os componentes “auxiliares” contribuem para o desempenho geral.

1.8.1 Barramentos em microcomputadores

Para ilustrar o conhecimento sobre a organização dos barramentos locais e de expansão, vamos tomar como exemplo um microcomputador. A Figura 5 mostra o esquema de barramentos de um microcomputador atual.

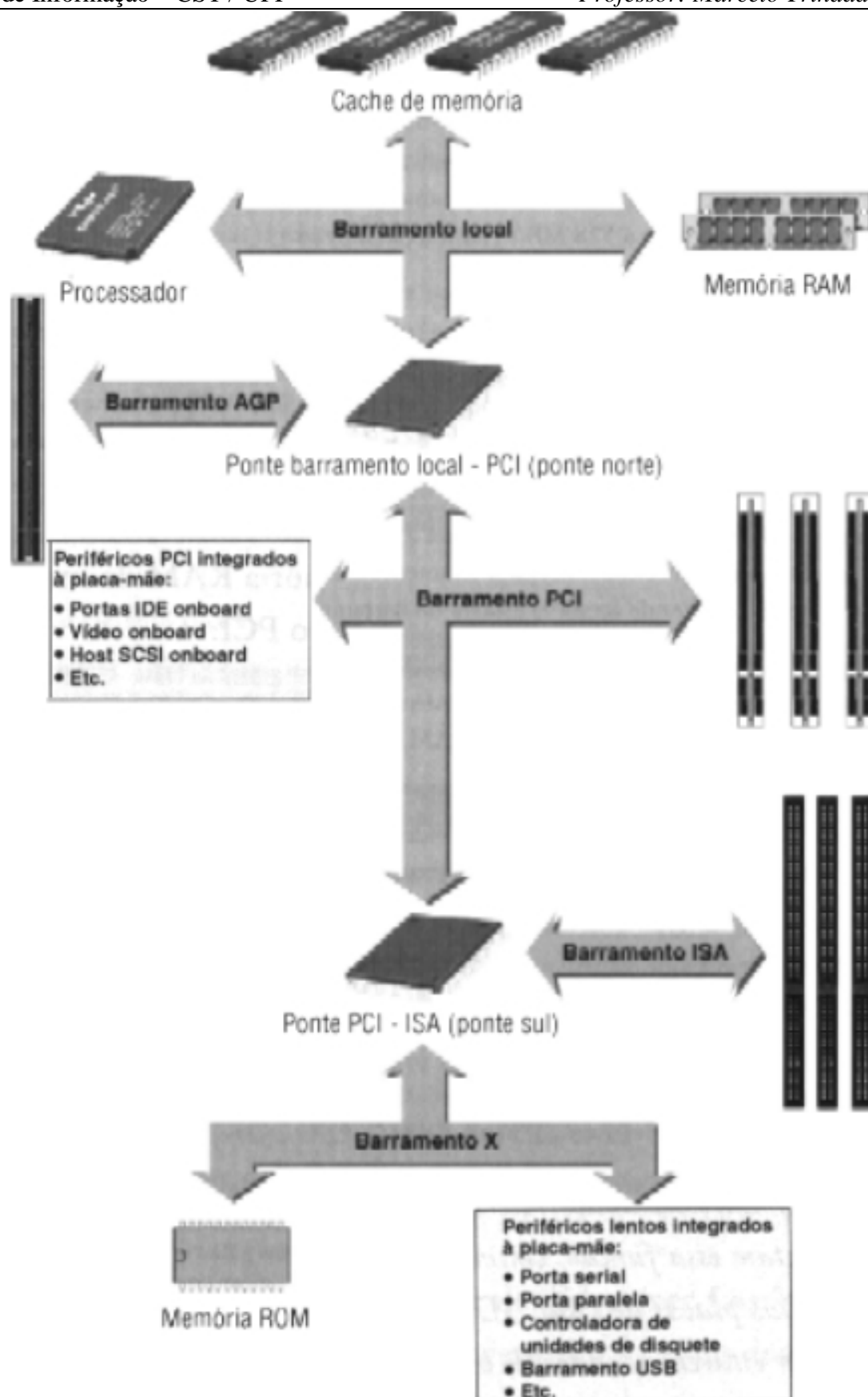


Figura 5 – Barramentos em um microcomputador

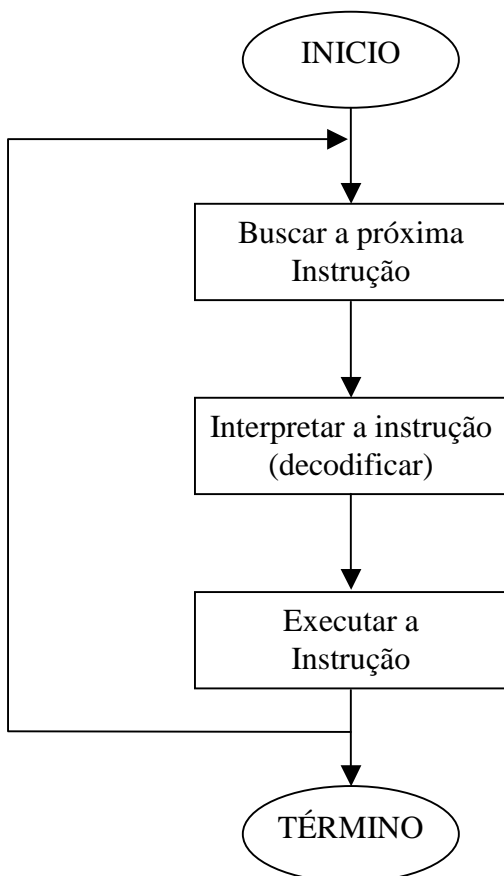
A comunicação dos barramentos de expansão com o local é realizada por um circuito chamado ponte (não necessariamente apenas uma, mas podendo conter também *buffers*), que faz parte dos circuitos de apoio da placa-mãe (*chipset*).

Os componentes *on-board*, normalmente são ligados ao barramento local, por uma extensão do barramento de expansão, chamado barramento X (*eXtension bus*). Assim, mesmo que o periférico esteja integrado a placa-mãe, ele é tratado como se estivesse conectado a um dos slots de expansão.

2 CPU

O processador (CPU) é o componente vital de um sistema de computação, responsável pela realização das operações de processamento (cálculos matemáticos, entre outros) e de controle durante a execução de um programa.

Um programa, para ser efetivamente executado por um processador, deve ser constituído de uma série de instruções (em linguagem de máquina). Estas instruções devem estar armazenadas em posições sucessivas da memória principal. A execução é sequencial, ou seja, se a instrução executada está na posição x , a próxima instrução a ser executada deverá estar na posição $x+1$. A sequência de funcionamento de uma CPU é conhecida como ciclo “Busca – Decodificação – Execução” de Instruções. A Figura 6 ilustra este ciclo.



- Um elemento dentro do processador, denominado contador de instruções (PC – *Program Counter*), contém a posição da próxima instrução a ser executada. Quando uma sequência de execução de instruções têm início, a instrução cujo endereço está no contador de instruções é trazida da memória para uma área chamada registrador de instruções (RI). Este processo é conhecido como **busca da instrução**.
- A instrução é interpretada por circuitos de decodificação que fazem com que sinais eletrônicos sejam gerados no processador como resultado do valor do campo de operação, isto é, **decodificam** a informação correspondente à operação a ser realizada.
- Esses sinais resultam na **execução da instrução**, isto é, aplicação da função contida pela operação sobre os operandos. Quando a execução de uma instrução é terminada, o contador de instruções é atualizado para o endereço da memória da próxima instrução ($x + 1$).

Figura 6 – Ciclo básico de instrução

A sequência de instruções pode mudar como resultado de uma instrução que direciona um desvio (também chamado de salto, *jump*). Instruções deste tipo contêm no campo operandos o endereço da próxima instrução a ser executada. Elas causam mudanças no fluxo do programa como resultado das condições dos dados. O desvio condicional representado por uma instrução de alto nível IF traduz-se em algum tipo de instrução de desvio.

As atividades realizadas pela CPU podem ser divididas em duas grandes categorias funcionais:

- Funções de processamento; e
- Funções de controle.

A função de processamento se encarrega de realizar as atividades relacionadas com a efetiva execução de uma operação, ou seja, processar (executar a instrução) de instruções. O principal componente da CPU que realiza a função de processamento é a ULA (unidade lógica e aritmética), sendo que ação dela é complementada pelo uso de registradores de processamento.

A função de controle é exercida pelos componentes da CPU que se encarregam de atividades de busca, interpretação e controle da execução das instruções, bem como do controle da ação dos demais componentes do sistema de computação (memória, entrada/saída). O principal componente da CPU responsável pela função de controle é a UC (unidade de controle).

2.1 Componentes

As funções de controle e processamento necessitam de componentes, compostos de circuitos digitais, para sua realização. Estes componentes são interligados interna e externamente através de barramentos. A figura abaixo ilustra os principais componentes de um processador atual.

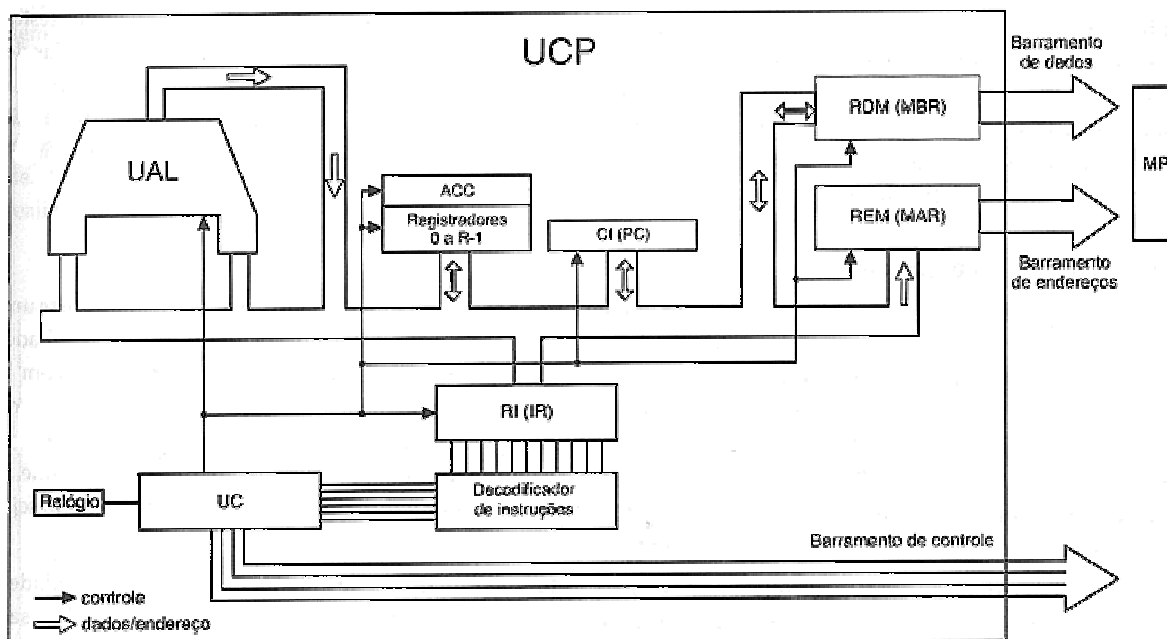


Figura 7 – Componentes da CPU

A figura ilustra uma CPU de um microprocessador simples, justamente para facilitar as primeiras explicações sobre o assunto, porém os seguintes componentes, pelo menos, devem fazer-se presentes em um microprocessador:

- UAL: Unidade Lógica e Aritmética (ULA) responsável por efetuar operações matemáticas com os dados. Estas operações podem ser, por exemplo, soma, subtração, multiplicação, divisão, operações lógicas AND, OR, XOR, NOT, deslocamento de bits à direita e esquerda, incremento e decremento, comparações;
- ACC: registrador(es) auxiliares onde sobre os quais a ULA realiza suas operações. Quando é único, todas as instruções o utilizam, como fonte e destino;
- CI (PC): Contador de instruções (*Program Counter*). Registrador cuja função específica é armazenar o endereço da **próxima** instrução a ser executada;
- RDM: Registrador de dados da memória;
- REM: Registrador de endereços da memória;
- MP: Memória principal. Local onde ficam dados e instruções já/a serem executadas;

- RI: Registrador de instruções: tem a função específica de armazenar a instrução a ser executada;
- Relógio: É o dispositivo gerador de pulsos. A quantidade de vezes em que este pulso básico se repete define a unidade de medida do relógio chamado frequência, usada também para definir velocidade na CPU;
- UC: Unidade de controle. É o dispositivo mais complexo da CPU, pois além de controlar a ação da ULA, possui a lógica necessária para realizar a movimentação de dados e instruções de e para a CPU, através de sinais de controle emitidos em instantes de tempo programados. Os sinais de controle ocorrem em vários instantes durante o período de realização do ciclo de instruções;
- Decodificador de instruções: É um dispositivo utilizado para identificar as operações a serem realizadas relacionadas à instrução a ser executada;
- Barramentos: utilizados para a comunicação entre os dispositivos, possuem três tipos específicos: dados, endereços e controle.

2.1.1 Função Processamento

Processar o dado é executar com ele uma ação que produza algum tipo de resultado. Esta é, pois, a atividade-fim do sistema computacional: ele existe para processar dados. Entre as tarefas comuns a esta função podem ser citadas as que realizam:

- Operações aritméticas: somar, subtrair, multiplicar, dividir, ...;
- Operações lógicas: and, or, xor, not,... ;
- Movimentação de dados: memória-CPU, CPU-memória, registrador-registrador, ...;
- Desvios: alteração da sequência de execução das instruções;
- Operações de entrada ou saída;

O principal dispositivo da CPU para a realização das atividades de processamento é a ULA. Ela utiliza os registradores de propósito gerais (ACC) como auxiliares à função de processamento. A ULA (UAL) é um aglomerado de circuitos lógicos e componentes eletrônicos simples que, integrados, realizam as funções acima citadas. Ela pode ser uma pequena parte da pastilha do processador, usada em pequenos sistemas, ou pode compreender um considerável conjunto de componentes lógicos, ocupando mais espaço na pastilha do processador.

A capacidade de processamento de uma CPU é em grande parte determinada pelas facilidades embutidas no hardware da ULA¹ para realizar as operações matemáticas projetadas. Um dos elementos fundamentais é a definição do tamanho da palavra. O valor escolhido no projeto de fabricação da CPU irá determinar o tamanho dos elementos ligados a área de processamento (ULA, barramentos internos e registradores).

Um tamanho maior ou menor de palavra acarreta, sem dúvida, diferenças fundamentais no desempenho da CPU, e, por conseguinte, do sistema como um todo. Os seguintes componentes possuem influência direta do tamanho da palavra:

- Influência no desempenho devido ao maior ou menor tempo de execução com operações matemáticas na ULA;
- Influência no desempenho devido ao tamanho escolhido para o barramento interno e externo da CPU. Em geral, Obtém-se o máximo de desempenho quando a largura (tamanho em bits) do barramento de dados é, no mínimo, igual ao tamanho da palavra (barramento interno);

¹ A ULA é composta exclusivamente de hardware.

- Influência também na implementação física do acesso à memória. Embora atualmente a capacidade das memórias seja medida bytes (ou conjunto de), o movimento de dados entre CPU e memória é normalmente medido em palavras (barramento externo).

2.1.2 Função de Controle

A área de controle de uma CPU é a parte funcional que realiza as atividades de: (uma de cada vez, geralmente)

- a) Busca da instrução que será executada, armazenando-a em um registrador especialmente projetado para esta atividade (RI);
- b) Interpretação das ações a serem desencadeadas com a execução da instrução (se é uma soma ou subtração, por exemplo, como realizá-las); e
- c) Geração de sinais de controle apropriados para realização das atividades requeridas para a execução propriamente dita da instrução identificada. Esses sinais de controle são enviados aos diversos componentes do sistema, sejam internos da CPU (como ULA, por exemplo) ou externos (como a memória);

A área destinada à função de controle é projetada para entender o que fazer, como fazer e comandar quem vai fazer no momento adequado. Uma analogia pode ser feita com os seres humanos, imaginando que a área de controle é o cérebro que comanda o ato de andar, enquanto a área de processamento são os músculos e ossos das pessoas que realizam efetivamente o ato. Os nervos podem ser relacionados com os barramentos entre os diversos elementos envolvidos.

Os dispositivos básicos que fazem parte da área de controle são:

- Unidade de controle (UC);
- Decodificador de instruções;
- Registrador de instruções (RI);
- Contador de instruções (program counter – PC);
- Relógio ou “dock”;
- Registrador de endereços da memória (REM);
- Registrador de dados da memória (RDM).

O componente vital para as funções de controle é a Unidade de Controle (UC). Ela recebe como entrada o valor do registrador de instruções e decodifica-o (através do decodificador de instruções). Para cada código de instruções ele gera uma seqüência de sinais diferentes, ativando os circuitos correspondentes para cada uma das tarefas necessárias para a busca e execução da instrução a ser executada.

Cada sinal de controle comanda uma **microinstrução** (que denota uma tarefa a ser executada para a execução de uma operação). Uma microinstrução pode ser responsável pela realização de uma carga em um registrador, uma seleção de um dado para entrada em um determinado componente, uma ativação da memória, a seleção de uma operação da ULA ou a habilitação de um circuito lógico, para citar alguns exemplos.

Existem várias formas de implementações (organizações) das unidades de controle. As duas usuais são:

- Organização convencional: a unidade de controle é composta por componentes digitais como flip-flops, contadores e decodificadores, que geram, seqüencialmente e nos instantes de tempo adequados, todos os sinais de controle necessários à ativação da unidade operacional;
- Organização microprogramada: em uma unidade de controle microprogramada, os sinais de controle estão armazenados numa memória especial chamada **memória de controle**. Vários sinais de controle são buscados a cada acesso a memória de controle;

2.1.3 Barramentos Internos

Os barramentos internos interligam os componentes do processador para troca de sinais e valores. Assim como no caso da comunicação processador/memória, são três os barramentos internos:

- Barramento Interno de Dados: Usado na transferência de dados entre os registradores e entre os registradores e a ULA. A sua largura define o tamanho da palavra da máquina;
- Barramento Interno de Endereços: Permite a transferência de endereços entre os registradores;
- Barramento Interno de Controle: Transmite os sinais do bloco de controle que comandam o funcionamento de cada circuito do processador.

2.2 Ciclo de instrução

Busca, decodificação e execução de instruções são tarefas básicas realizadas por um processador. Caracterizam um ciclo, pois as tarefas são executadas repetidamente, sempre e sempre, até que seja decodificada uma instrução que indique parada ao computador.

O fluxograma do ciclo de instrução anteriormente mostrado ilustra o mesmo de forma simplificada. Um dos pontos omitidos é o incremento do PC, função indispensável ao funcionamento de qualquer sistema de computação.

Outro ponto também apresentado de forma resumida são os acessos à memória. Tanto as instruções como os dados ficam armazenados na memória e portanto existem buscas de operandos na memória e cálculo do endereço da próxima instrução a ser executada. A Figura 8 ilustra mais alguns detalhes do ciclo de instrução.

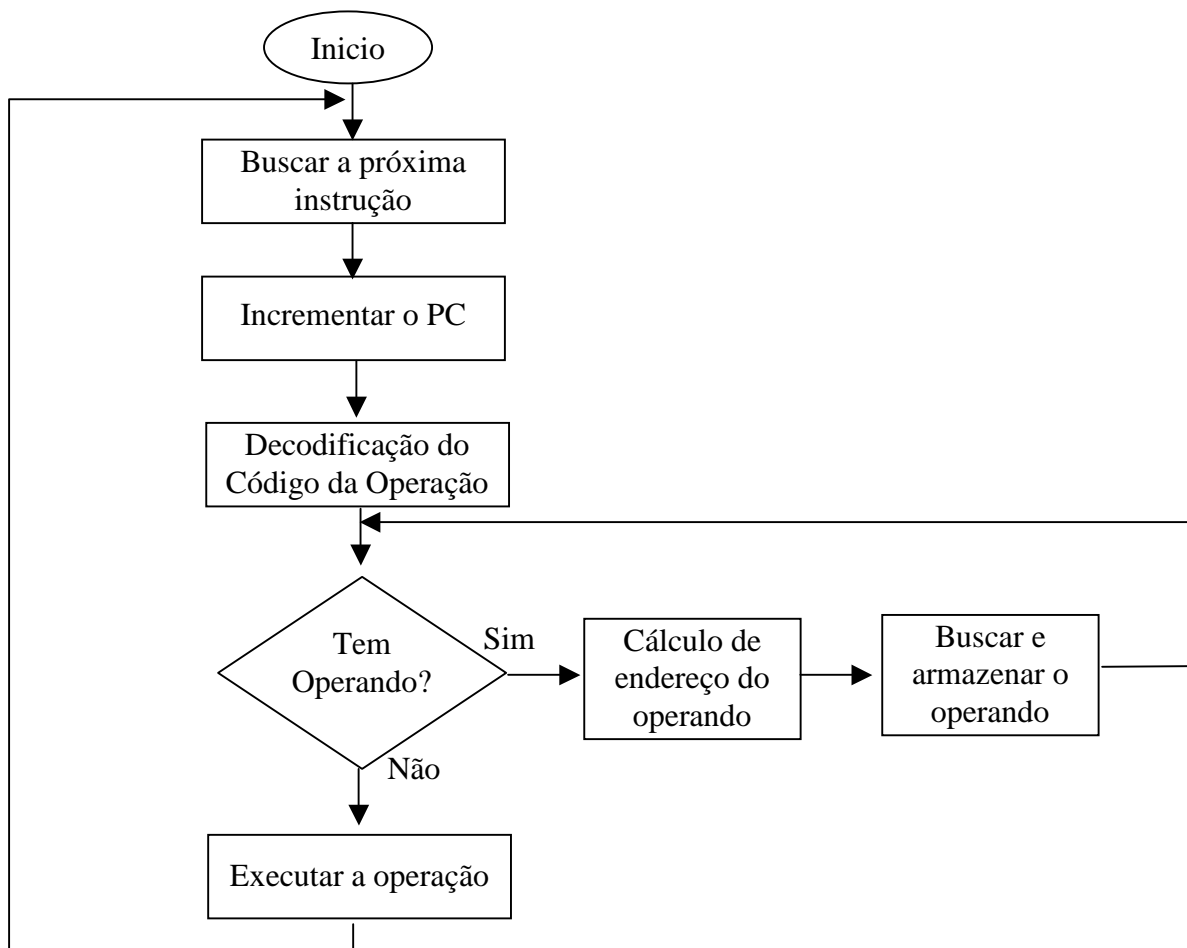


Figura 8 – Ciclo detalhado de instrução

Este ciclo de instrução pode ser descrito em LTR (Linguagem de Transferência entre Registradores), de modo que possamos acompanhar sua realização com a movimentação de informações entre os componentes da CPU. A Figura 9 ilustra este “Algoritmo” de funcionamento.

```

Iniciar

CICLO  RI ← MEM[PC]
       PC ← PC + 1

       Interpretar o Código da Operação

       Enquanto houver operandos não carregados
           Buscar Operandos
           PC ← PC + 1

       Executar a instrução

       Ir para CICLO

Fim
  
```

Figura 9 – Algoritmo para execução de instruções

Inicialmente, o conteúdo de memória no endereço da próxima instrução a ser executada (PC), tem seu valor transferido para RI. Logo após, o valor de PC é incrementado para o endereço da próxima instrução a ser executada. O decodificador de instruções irá receber os bits referentes ao Código da Operação e decodificá-lo, dando entrada na UC este valor. A UC gera os sinais necessários para a execução da Instrução.

Detalhando ainda mais, pode-se elencar uma série de passos a serem realizados em cada parte do ciclo. São eles:

a) Busca

- Copiar o PC para o registrador de endereços da memória (REM);
- Ler uma instrução da memória;
- Copiar o registrador de dados da memória (RDM) para o registrador de instruções (RI);
- Atualizar o contador de programa (PC);

b) Decodificação

- Nesta fase é determinada qual instrução deve ser executada;

c) Execução

- Cálculo de endereço dos operandos (se houver);
- Busca dos operandos na memória (se houverem);
- Seleção da operação a ser realizada pela ULA;
- Carga de registradores;
- Escrita de operandos na memória;
- Atualização do PC (somente no caso das instruções serem desvios)

2.3 Medidas de Desempenho

A medida geral de desempenho de um sistema de computação depende fundamentalmente da capacidade e velocidade de seus diferentes componentes, da velocidade com que estes componentes se comunicam entre si e do grau de compatibilidade entre eles (p. ex., se a velocidade da CPU de um sistema é muito maior que a da memória).

Considerando a existência de tantos fatores que influenciam o desempenho de um sistema de computação, desenvolveram-se diversos meios de medir seu desempenho, entre os principais relacionados exclusivamente com a CPU destacam-se:

- MIPS; e
- FLOPS.

O desempenho dos processadores, em geral, é medido em termos de sua velocidade de trabalho; como o trabalho da CPU é executar instruções, criou-se a unidade MIPS – Milhões de instruções por segundo. O MIPS é muito questionado, pois mede apenas a execução de instruções, sendo que existem diferentes instruções, com tempos de execução distintos, como por exemplo, multiplicação de números inteiros e multiplicação de números reais (ponto flutuante).

Em contraste com o MIPS, FLOPS – Operações de ponto flutuante por segundo, mede basicamente o desempenho da ULA, analisando apenas as instruções mais complexas (as que envolvem ponto flutuante). Hoje em dia, os microprocessadores estão na faixa de MFLOS (milhões de flops) sendo encontradas máquinas com GFLOPS (supercomputadores). O FLOPS foi inicialmente criado para ser a

medida de estações de trabalho e de supercomputadores, mas atualmente também é utilizado em microcomputadores.

Quando se trata da recuperação ou escrita de dados na memória, o **tempo de acesso** é a unidade de medida apropriada, estando relacionada à velocidade de cada componente e a do canal (barramento(s)) de interligação entre CPU e memória.

Tempo de resposta é uma medida ligada ao desempenho global do sistema e não de um ou outro componente. Trata-se do período de tempo gasto entre o instante em que o usuário iniciou uma solicitação e o instante em que o sistema apresentou ao usuário a resposta.

2.4 Mais de uma instrução por ciclo

Descrevendo o funcionamento da CPU na realização de seus ciclos de instrução foi observado que, embora o ciclo de instrução seja composto de várias etapas, ele é realizado de forma sequencial, isto é, uma etapa se inicia após a conclusão da etapa anterior. Desta forma enquanto a fase de decodificação da instrução estava sendo executada, a busca (RDM e REM) e a execução (ULA) estavam ociosas.

A situação acima leva a crer que a CPU não pode executar mais de uma instrução por ciclo, mas isso não é verdade. Graças aos avanços da tecnologia existem CPUs que executam mais de uma instrução por ciclo, sendo chamadas de superescalares.

2.4.1 Pipeline

O pipeline usa a metodologia de uma linha de montagem, onde a fabricação de um produto qualquer (carro, bicicleta, ...) é subdividida em partes (implementadas em setores), nas quais tarefas independentes estão sendo desenvolvidas ao mesmo tempo. A Figura 10 exemplifica um pipeline na produção em serie de bicicletas.

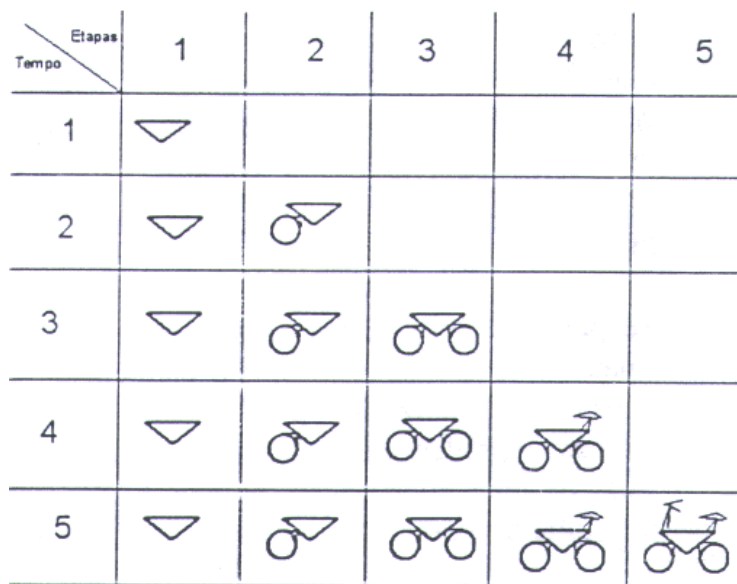


Figura 10 – Linha de montagem

A divisão das tarefas resultou em 5 etapas. Na primeira é feita a estrutura da bicicleta com pedais e correa, na segunda é instalada a roda dianteira, na terceira é colocada a roda traseira, na quarta é colocado o assento e, por fim, na quinta e última etapa é colocado o guidom. É conveniente lembrar que o tempo para a produção de uma bicicleta não diminui, mas o tempo entre bicicletas sim: a primeira

bicicleta demora cinco unidades de tempo para ser produzida, mas logo a seguir, em cada unidade de tempo uma nova bicicleta é produzida (se não houverem problemas).

O *pipeline* é uma técnica de implementação de CPU onde múltiplas instruções estão em execução ao mesmo tempo. O processador é construído com diversos estágios distintos, cada um responsável pela execução de uma parte da instrução e possuindo o seu próprio bloco de controle. Assim que um estágio completa sua tarefa com uma instrução passa esta para o estágio seguinte e começa a tratar da próxima instrução. Uma vez que diversas instruções são executadas ao mesmo tempo, obtêm-se um acréscimo no desempenho do processador.

No caso dos processadores, pode-se, por exemplo, dividir a execução de uma instrução em 5 estágios básicos:

1. Busca da Instrução (*fetch*)
2. Decodificação (*decode*)
3. Busca do Operando (*operand fetch*)
4. Execução (*execute*)
5. Armazenamento do Resultado (*store*)

A tabela 4 mostra uma simulação da execução de instruções em uma CPU com 5 estágios.

Tabela 4 – Exemplo de instruções em um pipeline de 5 estágios

Tempo	Busca Instrução	Decodificação	Busca Operando	Execução	Armazenamento
1	I ₁				
2	I ₂	I ₁			
3	I ₃	I ₂	I ₁		
4	I ₄	I ₃	I ₂	I ₁	
5	I ₅	I ₄	I ₃	I ₂	I ₁
6	I ₆	I ₅	I ₄	I ₃	I ₂
7	I ₇	I ₆	I ₅	I ₄	I ₃
N

Numa situação ideal, a aceleração seria igual ao número de estágios do *pipeline*. Porém como alguns estágios são mais lentos que outros e devido ao tempo necessário para encher o *pipeline* e esvaziá-lo, isto não acontece. Dentre os principais problemas com o desempenho dos pipelines relaciona-se:

- **Conflitos Estruturais:** ocorrem quando dois ou mais estágios do pipeline tentam acessar o mesmo dispositivo de hardware simultaneamente. O principal exemplo deste tipo é o conflito pela memória, quando os estágios de busca da instrução, busca do operando e armazenamento de resultados podem tentar acessar a memória no mesmo ciclo de clock. Este caso pode ser minimizado com o uso de memória separada para dados e instruções. Para outros casos de conflitos uma solução poderia ser a duplicação do hardware envolvido, se possível;
- **Conflito por Dados:** Acontecem quando instruções consecutivas fazem acesso aos mesmos operandos e a execução de uma instrução depende da execução da outra. Um destes problemas, comum em pipelines, é chamado de *Dependência Verdadeira* de dados;
- **Conflitos de Controle:** Ocorrem com instruções de desvio, fazendo com que instruções já buscadas sejam descartadas e *bolhas* sejam inseridas no pipeline. Os incondicionais podem ser trabalhados através da execução especulativa, enquanto os condicionais devem ser trabalhados através de predição (estática ou dinâmica).]

2.4.2 Unidades de execução especializadas

Processadores superescalares possuem várias unidades de execução permitindo que diversas instruções sejam executadas ao mesmo tempo. Um aspecto importante neste tipo de processador é que como mais de uma instrução é executada simultaneamente e devido a tempos de execução diferente entre elas ou a conflitos ocorridos, pode se dar à execução de instruções fora da ordem original do programa. A Figura 11 ilustra este tipo de organização de CPUs.

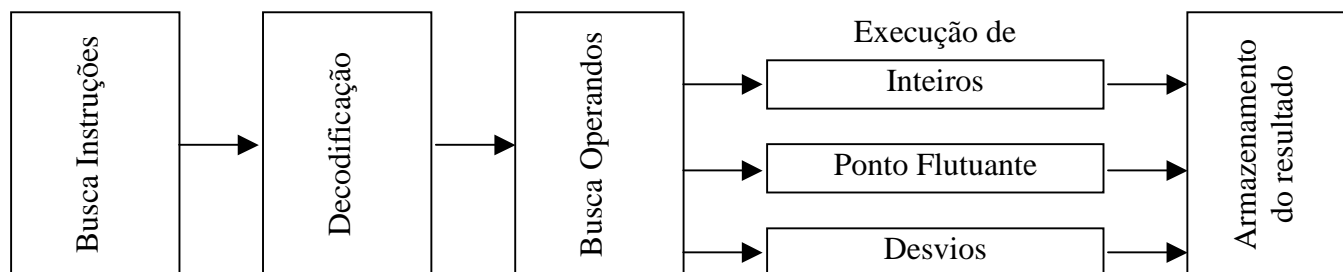


Figura 11 – Processadores superescalares

Estes processadores possuem unidades funcionais específicas para cada tipo de instrução. As unidades funcionais podem internamente ser divididas em várias etapas cada uma. Desta forma, têm-se pipelines distintos: para execução de instruções envolvendo números inteiros, reais e desvios.

3 Memória

Uma das partes mais importante do computador é a memória. O processador apenas recebe dados e os processa segundo alguma pré-programação, logo após devolvendo-os, não importando de onde vem e para onde vão. Os programas a serem executados e os dados a serem processados (inclusive os que já o foram) ficam na memória, visto que a área para armazenamento de dados do processador é pequena. Há basicamente dois tipos de memória:

- ROM (*Read-Only Memory*): Só permitem a leitura de dados e são lentas; em compensação não perdem seu conteúdo quando desligadas;
- RAM (*Random Access Memory*): São rápidas, permitem leitura e escrita, porém, seu conteúdo é perdido quando são desligadas.

Dentro da ROM existem basicamente 3 programas: BIOS, POST e SETUP. É comum acontecer confusão em relação aos nomes, sendo usado atualmente o termo BIOS como algo genérico. Para acessar o programa de configuração, basta acessar um conjunto de teclas durante o POST (geralmente na contagem da memória). Na maioria das máquinas, basta apertar a tecla ‘DEL’ ou ‘Delete’, porém, esse procedimento pode variar de acordo com o fabricante.

Quando o computador é ligado, o POST (*Power On Self Test*) entra em ação, identificando a configuração instalada, inicializando os circuitos da placa-mãe (chipset) e vídeo, e executando teste da memória e teclado. Após, ele carrega o S.O. de algum disco para a memória RAM, entregando o controle da máquina para o S.O.

Na RAM, ficam armazenados o S.O., programas e dados que estejam em processamento. O processador acessa a RAM praticamente o tempo todo. Atualmente a memória RAM, formada por circuitos de memória dinâmica (DRAM - *Dynamic RAM*), é mais lenta que o processador, ocasionando *wait states*, até que a memória possa entregar ou receber dados, diminuindo assim o desempenho do micro. Memórias mais rápidas amenizam este problema, assim como a utilização de cache de memória.

A cache é uma memória estática (SRAM - *Static RAM*) de alto desempenho, utilizada para intermediar a comunicação com o processador. Na maioria das vezes o processador não acessa o conteúdo da RAM, mas sim uma cópia que fica na cache. A cache é utilizada desde o 386DX, e a partir dos 486, todos os processadores passaram a conter uma quantidade de memória estática, conhecida como L1 ou interna. A cache fora do processador é conhecida como L2 ou externa. Hoje, existem processadores com mais níveis de cache. Uma ressalva é que os processadores a partir do Pentium II possuem a L2 dentro da caixa que envolve o processador, não fazendo mais sentido as denominações interna e externa.

A DRAM é formada por capacitores, que são fáceis de construir, baratos e pode-se aglomerar muitas células de memória em pequeno espaço físico. O problema é que após algum tempo, eles descarregam, dessa forma deverá haver um período de recarga, chamado *refresh*. Durante este período, a memória geralmente não pode ser acessada, limitando assim com uma imposição física sua velocidade. Por outro lado, a SRAM é formada por caros circuitos digitais chamados flip-flops, que armazenam dados sem a necessidade de ciclos para *refresh*. Um flip-flop, por ser um circuito completo, é maior que um capacitor, conseqüentemente, onde cabem muitos capacitores têm-se somente alguns flip-flops. Devido ao preço, tamanho e consumo, não é possível que um micro tenha toda sua RAM de memória estática, então a partir dos 386, utiliza-se um modelo híbrido com SRAM como cache e DRAM como RAM propriamente dita.

A Tabela 5 mostra as principais características das memórias DRAM e SRAM.

Tabela 5 – Comparativo entre DRAM e SRAM

	DRAM	SRAM
Preço	Barata	Cara
Integração	Fácil (muita capacidade em pouco espaço)	Difícil (pouca capacidade em muito espaço)
Consumo	Baixo	Alto
Velocidade	Lenta, pois necessita de refresh	Rápida

Dentro da memória, os dados são organizados em uma matriz, composta de linhas e colunas, tendo seus endereços crescendo da esquerda para direita (coluna) e de cima para baixo (linha). Quando um endereço de memória é necessário, o circuito que controla acessos à memória recebe o endereço a ser buscado, dividindo-o em 2, gerando assim o valor da linha e da coluna (geralmente pela utilização dos bits mais significativos e menos significativos).

3.1 Organização e hierarquia da memória

A memória deve ser antes de tudo organizada, para que o processador possa saber onde buscar um dado e onde colocar outro já processado. Para isso, ela é organizada em pequenas áreas, chamadas “endereços”. Da mesma forma que um grande armário, repleto de gavetas, sendo cada uma delas diferenciada através de um número, onde dentro de cada gaveta podemos colocar uma informação. Cada gaveta assemelha-se a um endereço de memória. Por motivos históricos e de retrocompatibilidade a unidade de referência à memória continua sendo o byte, mesmo com os processadores atuais acessando a memória a 32 ou 64 bits por vez.

Caso a RAM se esgote, o processador transfere o conteúdo atual da RAM para um arquivo em disco, chamado arquivo de troca (*swapping*), liberando espaço na RAM. O conteúdo deste arquivo é colocado de volta na RAM quando algum dado nele contido for solicitado sendo o processo conhecido como memória virtual. Quanto maior a quantidade de memória RAM, menor a probabilidade de arquivos de trocas, aumentando assim o desempenho do micro uma vez que o acesso ao arquivo de troca (H.D.) é bem mais lento que a memória RAM.

Dividindo as memórias em uma hierarquia, o objetivo que se deseja alcançar é um sistema de memória com desempenho próximo ao da memória mais rápida e custo por bit próximo ao da memória mais barata.

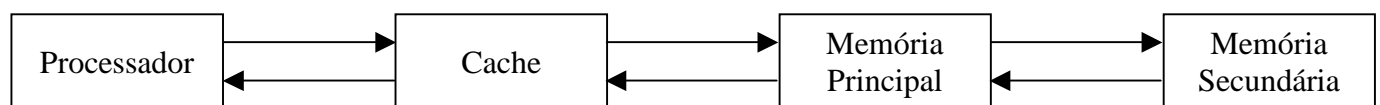


Figura 12 – Hierarquia de acesso à memória

A memória secundária é geralmente memória em disco magnético, a memória cache é tradicionalmente uma memória estática (SRAM) enquanto a principal, ou simplesmente RAM, é uma memória dinâmica (DRAM).

Ao executar uma instrução, a CPU não manipula dados da memória, nem mesmo da cache. Ao invés disso, ela processa os dados já carregados em registradores; desta forma, alguns autores colocam os registradores como integrantes da hierarquia de memória. A Figura 13 ilustra uma hierarquia de memória incluindo os registradores e colocando informações em relação a capacidade, custo e tempo de acesso.

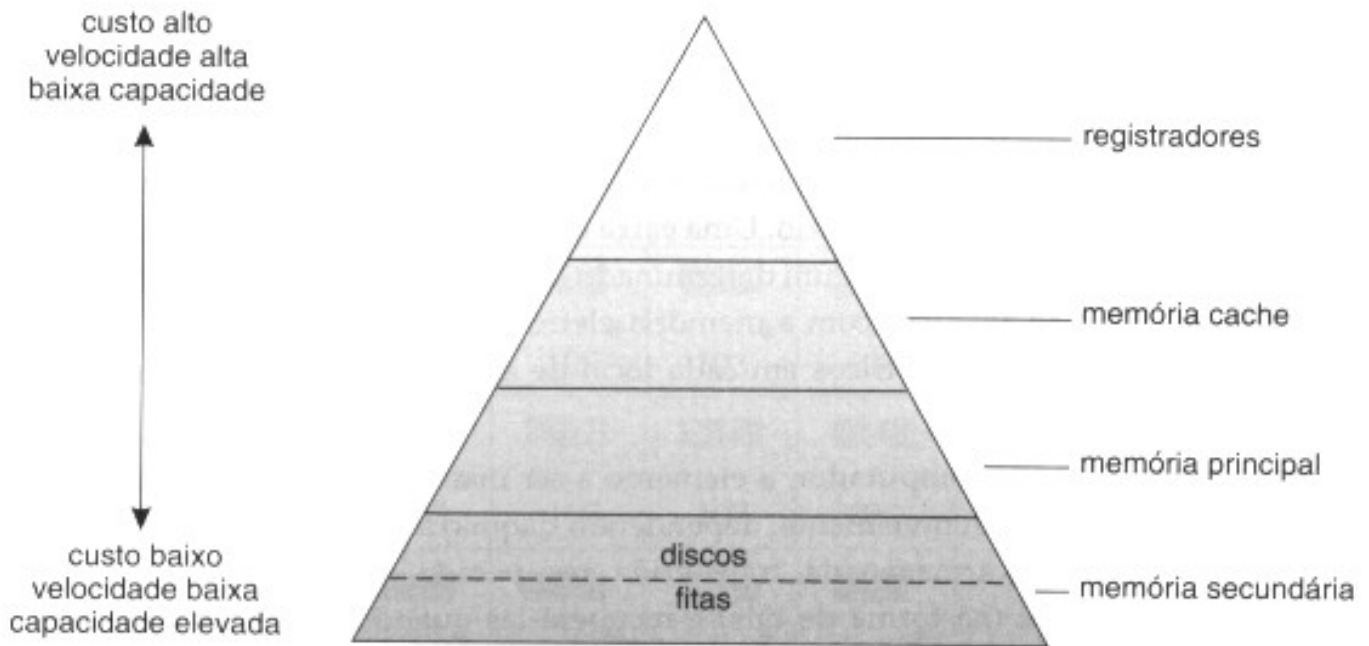


Figura 13 – Hierarquia de memória

A representação em forma de pirâmide facilita a compreensão pois no topo estão os registradores, de acesso quase imediato com capacidade reduzida, enquanto na base encontra-se a memória secundária, disponível em grandes quantidades porém de acesso mais lento.

3.2 Tempo e ciclo de acesso

Quando o processador ordena o armazenamento de um dado na memória ou solicita que a memória devolva um dado armazenado, isto não é realizado imediatamente. Essa demora é chamada de tempo de acesso, e é uma característica de todas as memórias. As DRAM possuem tempo de acesso entre 70 e 50ns, enquanto as SRAM apresentam tempo de acesso menor que 20ns (chegando atualmente a 5ns).

Em memórias FPM, EDO e BEDO, o tempo de acesso vem estampado na nomenclatura do circuito integrado, como sufixo. O “6” em uma memória dinâmica significa que esta tem 60 ns de tempo de acesso. Nas memórias SDRAM, por serem sincronizadas com o barramento local, o que há escrito NÃO é seu tempo de acesso, mas sim sua frequência máxima de operação, expressa em nanosegundos. Outro fator a ser levado em conta nas memórias SDRAM é o número de pulsos de latência.

O processador gasta, pelo menos, 2 pulsos de clock para acessar a memória RAM (clock externo, ou do barramento local). Um problema é que a velocidade do barramento local aumentou sem uma redução proporcional no tempo de acesso das memórias. Para resolver este problema são utilizados *wait states*, para realizar um ciclo de acesso. Um *wait state* é um pulso extra de clock adicionado ao ciclo de leitura ou escrita da memória. Como o ciclo de acesso à memória RAM é de 2 pulsos de clock, com a adição de 1 *wait state*, o ciclo passa a ter 3 pulsos. A adição de *wait states* reduz o desempenho do micro; desta forma a solução para este impacto é o uso da memória estática (cache), deixando o processador conversando com a cache sem *wait states*, em pelo menos 80% de seu tempo.

3.3 Tecnologias de memória RAM

Mesmo não podendo baixar o tempo de acesso da memória dinâmica, sobretudo por causa da necessidade do refresh, desenvolveram-se diversas tecnologias de construção de circuitos de memória

RAM. Embora tenham o mesmo tempo de acesso, circuitos com construção diferente podem apresentar desempenhos diferentes. As principais tecnologias são:

- **FPM (*Fast Page Mode*):** Ela retém o valor da última linha acessada, assim para os acessos realizados na mesma linha o controlador não necessita enviar o endereço da linha. O resultado é que o primeiro acesso demorará o tempo normal e os seguintes na mesma linha serão mais rápidos. É o mais velho e menos sofisticado tipo de RAM, usada em micros 486 e antigos Pentium, usando barramentos de até 66MHz. Os ciclos de acessos não podem ser menores do que 5-3-3-3 (5 pulsos para o primeiro elemento e 3 pulsos para cada um dos 3 dados seguintes), na prática 6-3-3-3;
- **EDO (*Extended Data Output*):** Retém os dados na saída da memória mesmo quando o sinal de leitura é desabilitado. Por consequência, o próximo endereço poderá começar a ser enviado enquanto os dados do endereço anterior ainda estão na saída. Possui ciclo típico de acessos não menores que 5-2-2-2 (na prática 6-2-2-2), sendo teoricamente 20% mais rápida que a FPM. Foram introduzidas com o Pentium;
- **BEDO (*Burst Extended Data Output*):** É semelhante a EDO, com a diferença de possuir integrado um contador de endereços. Na prática, só é necessário enviar os valores de linha e coluna iniciais que ela devolverá também os 3 dados seguintes automaticamente. Possui ciclo de acesso desta forma de 5-1-1-1, sendo teoricamente mais rápidas que as FPM (40%) e EDO (25%);
- **SDRAM (*Synchronous Dynamic RAM*):** Ao contrário das anteriores, é uma memória síncrona, utilizando o barramento local para comandar os circuitos internos, conseguindo trabalhar com valores acima de 66MHz. Trabalha em *pipeline*, disponibilizando um segundo acesso antes do final do primeiro, por este motivo o ciclo típico é x-1-1-1, onde “x” varia de acordo com o chipset. A característica mais importante é a sua capacidade de trabalhar com frequências mais elevadas mantendo o mesmo desempenho (nas FPM e EDO quando se aumenta a frequência, deve-se aumentar o número de *wait states*). Possui diversas modificações internas, como a presença de 2 matrizes, o que permite 2 que acessos diferentes sejam iniciados em paralelo. São encontradas com velocidade de 15, a 7 ns, teoricamente funcionando a 125 MHz, mas na prática, dificilmente passam de 83 MHz (conhecidas como PC-66). Não sendo adequadas para placas que usam barramento de 100 MHz;
- **PC-100:** São memórias SDRAM com vários aperfeiçoamentos, que permitiram o funcionamento estável com bus de 100MHz. As memórias de 100MHz foram criadas antes dos processadores com barramento externo de mesmo valor. Quando eles chegaram ao mercado, a Intel descobriu que as memórias para 100MHz não funcionavam corretamente a essa frequência, realizando assim as mudanças para seu correto funcionamento;
- **PC-133:** O limite teórico da memória SDRAM é 125MHz. Entretanto, com o avanço tecnológico foi possível desenvolver memórias SDRAM que conseguem trabalhar a 133MHz, chamadas PC-133;
- **DDR-SDRAM (*Double Data Rate SDRAM*):** Um avanço sobre a SDRAM que suporta transferências de dados duas vezes por pulso (na subida e na descida do pulso), dobrando o desempenho da SDRAM tradicional. Também chamada de SDRAM II;
- **ESDRAM (*Enhanced SDRAM*):** Consiste de uma SDRAM com uma pequena quantidade de memória estática dentro do circuito, criando uma pequena cache. Diminui a latência permitindo que trabalhe com frequências de até 200MHz;
- **RDRAM (*Rambus DRAM*):** Ao contrário das tecnologias anteriores de memória, a RDRAM é baseada em protocolo, utilizando um padrão de barramento proprietário. Necessita de um barramento curto entre a ponte-norte e os módulos de memória. Esse barramento trabalha com apenas 16bits, mas a uma frequência mínima de 400MHz, atingindo uma taxa de 800MB/s. Como a RDRAM transfere dados tanto na subida do clock como na descida, a taxa é dobrada, atingindo 1,6GB/s. O controlador de memória Rambus é um conversor de protocolos e a arquitetura interna é diferente das existentes,

principalmente por usar 16 matrizes de capacitores ao invés de 1 ou 2, porém ainda apresenta problemas de latência;

- **SLDRAM (SyncLink DRAM):** Assim como a RDRAM é baseada em protocolo, porém de arquitetura aberta. Pode operar a 200MHz, a uma taxa de transferência de 1,6GB/s, porém pode entregar dados tanto na subida como na descida do pulso, assim sua taxa pode chegar a 3,2GB/s. É mais barata que a RDRAM, possui melhor desempenho e menor latência.

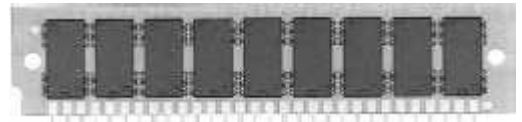
3.4 Módulos de memória

Também estarão presentes nas Placas-Mãe os slots para a conexão dos módulos de circuitos eletrônicos que correspondem à memória, indispensáveis para o funcionamento do sistema computacional.

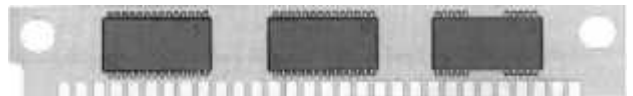
Inicialmente a memória RAM era composta de pequenos chips DIP (*Dual In Parallel*) encaixados na placa mãe. A instalação destes módulos era muito trabalhosa, e para facilitar a vida dos usuários (e aumentar as vendas) os fabricantes desenvolveram módulos de memória: placa de circuito impresso onde os circuitos integrados de memória se encontravam soldados. Basta encaixar a placa a placa-mãe do micro. Os principais módulos são:



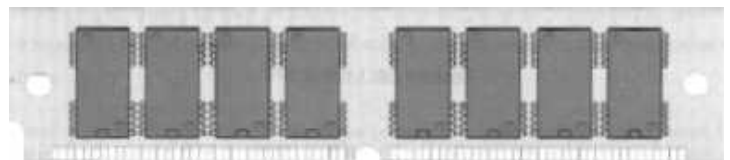
- **SIPP (Single in Line Pin Package):** foi o primeiro módulo a ser criado e sua aparência lembrava um pente (daí o apelido “pente de memória”). Os terminais eram similares aos usados nos DIP, causando mau contato e danificação. Eram encontrados em versões de 256KB, 1MB e 4MB, todos de 8bits;



- **SIMM30-(Single in Line Memory Module):** é basicamente um SIPP com novo encaixe, semelhante ao dos slots e não permite que os módulos sejam colocados invertidos. Eles têm 30 terminais, operando a 8bits em versões de 256KB, 1MB e 4MB; Possui módulos com e sem paridade;



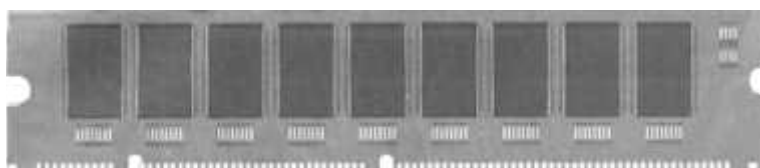
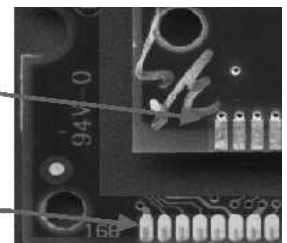
- **SIMM-72 (Single in Line Memory Module):** possuem 72 terminais e trabalham com 32 bits, tendo sido criados para uso com 486 e superiores. São encontrados com diversas capacidades, sendo as mais usuais de 4MB, 8MB, 16MB e 32MB, com e sem paridade. Para saber se o módulo tem ou não paridade, basta contar o número de circuitos: se for ímpar ele possui paridade (em módulos com dupla-face, contar somente os circuitos de uma face);



- **DIMM (Double in Line Memory Module):** possuem 168 terminais (84 DE CADA LADO) e trabalham com 64bits. São encontrados com diversas capacidades acima de 8MB, com e sem paridade. Os primeiros eram montados com FPM ou EDO e atualmente utilizam SDRAM ou superiores. Ao contrario dos anteriores, possui contatos independentes nas duas faces;

SIMM possuem as trilhas das duas faces conectadas

DIMM possuem as trilhas das duas faces independentes



- RIMM (*Rambus In Line Memory Module*): padronizado pela Rambus para uso da RDRAM no micro. São fisicamente semelhantes as DIMM, porém não é possível o encaixe de módulos RIMM em soquetes DIMM e vice-versa;



Para conhecer a capacidade de armazenamento de cada módulo assim como suas demais características deve-se consultar o *databook* (livro com dados técnicos) do fabricante. Na prática, o monitor mostra o tamanho através da contagem de memória, no POST. É importante salientar que módulo de memória (SIMM, DIMM, RIMM) não tem relação com a tecnologia usada nos circuitos (FPM, EDO, SDRAM). É nos módulos que os circuitos são soldados. Por exemplo, podemos ter um módulo SIMM-72 que utiliza circuitos FPM ou EDO.

3.5 Memórias do tipo ROM

Há muito tempo, os sistemas de computação utilizam uma parte do espaço de endereçamento da memória principal com memórias do tipo ROM. Um exemplo típico é o dos microcomputadores do tipo PC, que vêm de fábrica com um conjunto de rotinas básicas armazenadas em ROM, denominadas em conjunto de BIOS (Basic Input/Output System, sistema básico de entrada/saída).

Existem basicamente quatro tipos de memória ROM:

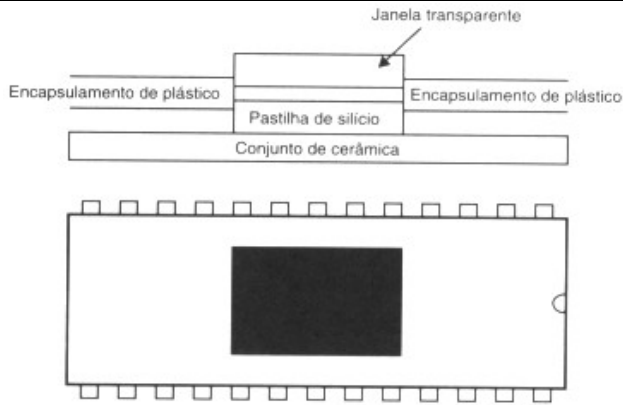
- ROM (pura);
- PROM;
- EPROM;
- EEPROM;

A memória ROM pura (o termo pura serve apenas para diferenciar das demais) é conhecida tecnicamente como “programável por máscara” (mask programmed), devido ao seu processo de fabricação e escrita dos bits na memória. Na ROM o conjunto de bits é inserido durante o processo de fabricação, sendo que após o término da fabricação, a pastilha ROM está completa, com o programa armazenado e nada poderá alterar o valor de qualquer de seus bits.

Uma variação deste tipo de memória chama-se PROM (programed read only memory). Na realidade, não se trata propriamente de ser programável, porque não é possível sua reutilização. Nela, como nas ROM puras, somente é possível gravar os bits desejados uma única vez, porém com a diferença de que a gravação dos bits é posterior a fase de fabricação da pastilha, embora deva ser realizada por um dispositivo especial.

Há outros dois tipos de ROM, desenvolvidos posteriormente, que têm uma particularidade interessante: ainda que se mantenham somente para leitura de programas aplicativos durante uma execução normal, elas podem ser apagadas e regravadas, sendo portanto reutilizáveis. A forma para eliminação dos dados atuais e reprogramação é o que diferencia os dois tipos, sendo elas a EPROM (erasable PROM) e EEPROM (electrically ou electronically EPROM), também chamada EAROM (electrically alterable ROM).

A EPROM pode ser utilizada diversas vezes porque os dados nela armazenados podem ser apagados ao se iluminar a pastilha com luz ultravioleta. A luz deve incidir sobre uma janela de vidro, montada na parte superior da pastilha.

**Figura 14 – EEPROM**

Uma vez apagada, a pastilha pode ser reutilizada através de um novo processo de “queima” dos novos bits. Neste tipo de memória, a janela de vidro costuma ser coberta para evitar a eliminação de informações acidental.

A EEPROM ou EAROM, introduz uma característica mais versátil e prática no processo de reutilização das ROM: a programação (escrita dos bits), a eliminação de conteúdo e a reprogramação são efetuadas por controle da CPU. Com a EEPROM programada, as instruções nela armazenadas são retidas indefinidamente até que um sinal para eliminação de conteúdo seja recebido por um sensor especial.

3.6 Memória cache

Parte do problema de limitação de velocidade do processador refere-se à diferença de velocidade entre o ciclo de tempo da CPU e o ciclo de tempo da memória principal, ou seja, a MP transfere bits para a CPU em velocidades sempre inferiores às que a CPU pode receber e processar os dados, o que acarreta, muitas vezes, a necessidade de acrescentar-se um tempo de espera para a CPU (wait state).

O problema de diferença de velocidade se torna difícil de solucionar apenas com melhorias no desempenho das MP, devido a fatores de custo e tecnologia. Enquanto o desempenho dos microprocessadores, por exemplo, vem dobrando a cada 18/24 meses, o mesmo não acontece com a taxa de transferência e o tempo de acesso das memórias DRAM, que vêm aumentando pouco de ano para ano.

Na busca de uma solução para este problema foi desenvolvida uma técnica que consiste da inclusão de um dispositivo entre a CPU e a MP, denominado de memória CACHE, cuja função é acelerar a transferência de informações entre CPU e MP e, com isso, aumentar o desempenho do sistema de computação.

A *cache* é um nível na hierarquia de memória entre a CPU e a memória principal (MP). É construída com memória SRAM, que é muito mais rápida do que a DRAM, normalmente empregada na construção de MP. A cache é fabricada com tecnologia semelhante à da CPU e, em consequência possui tempos de acesso compatíveis, resultando numa considerável redução da espera da CPU para receber dados e instruções. Como o custo da memória estática é muito alto, a *cache* normalmente é muito menor do que a memória principal (geralmente ≤ 1 Mb).

Com a inclusão da cache pode-se enumerar, de modo simplista, o funcionamento do sistema:

1. Sempre que a CPU vai buscar uma nova informação (instruções ou dados), ela acessa a memória cache;
2. Se a informação estiver na cache (chama-se “cache hit” – acerto), ela é transferida em alta velocidade (compatível com a da CPU);
3. Se a informação não estiver na cache (chama-se “cache miss” – falta), então o sistema está programado para transferir a informação desejada da MP para a cache. Só que esta informação não é somente da instrução ou dado desejado, mas dele e de um grupo subsequente, na pressuposição de que as instruções/dados do grupo serão requeridas pela CPU em seguida e, portanto, já estarão na cache quando necessário.

A Figura 15 ilustra a troca de informações entre a CPU, Cache e MP.

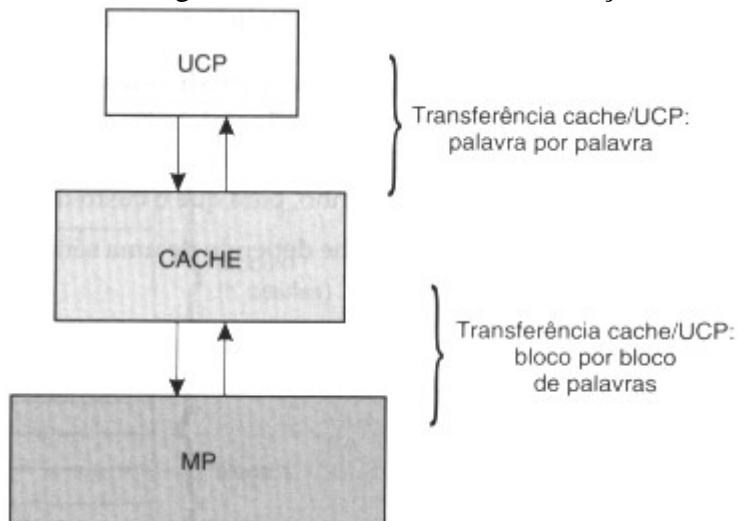


Figura 15 – Transferência de dados entre CPU/Cache/MP

Para haver algum aumento no desempenho de um sistema de computação com a inclusão da memória cache é necessário que haja mais acertos (hits) do que faltas (miss). Isto é, a memória cache somente é produtiva se a CPU puder encontrar uma quantidade apreciável de palavras na cache, suficientemente grande para sobrepujar as eventuais perdas de tempo com as faltas, que redundam em transferência de um bloco de palavras da MP para a cache, além da efetiva transferência da informação desejada.

A implementação de memórias foi referendada pela existência muito forte de localidades na execução dos programas. Existem dois tipos de localidades: a espacial e a temporal.

- **Localidade Temporal:** As posições da memória, uma vez acessadas, tendem a ser acessadas novamente num futuro próximo. Normalmente ocorrem devido ao uso de laços de instruções, acessos a pilhas de dados e variáveis como índices, contadores e acumuladores;
- **Localidade Espacial:** Se um programa acessa uma palavra de memória, há uma boa probabilidade de que o programa acesse num futuro próximo, uma palavra subsequente ou um endereço adjacente àquela palavra que ele acabou de acessar. Em outras palavras, os endereços em acessos futuros tendem a ser próximos de endereços de acessos anteriores. Ocorre devido ao uso da organização sequencial de programas.

O tamanho da cache influencia diretamente no desempenho do sistema, uma vez que para existir um aumento real de desempenho o número de acertos deve superar e muito o número de faltas. Com uma cache maior, há maior probabilidade de a informação desejada estar contida nela, porém outros fatores também devem ser levados em conta, como o tempo de acesso da MP, da memória cache e a natureza dos programas em execução (princípio da localidade).

Um fator que influencia diretamente o desempenho da cache é a forma de mapeamento dos dados da MP nela. Como o tamanho da cache é bem menor que MP, apenas uma parte dos dados da MP pode ser copiada na cache. Existem basicamente três formas de mapeamento: o associativo, o direto e o set-associativo.

- **Mapeamento completamente associativo:** A palavra pode ser colocada em qualquer lugar da cache. Neste caso deve ser armazenado na cache não somente o dado mas também o endereço. Para descobrir se a posição procurada está armazenada na cache, é feita a comparação simultânea de todos os endereços, caso seja localizado (cache hit) o dado é devolvido ao processador. Caso o endereço pesquisado não se encontre na cache (cache miss) a memória principal é acessada.
- **Mapeamento direto:** Cada palavra deve ser armazenada em um lugar específico na cache, o qual que depende do seu endereço na memória principal. O endereço é dividido em duas partes: Tag e Índice. O índice é usado como endereço na cache. Indica a posição da cache onde pode estar armazenada a palavra. O tag é usado para conferir se a palavra que está na cache é a que está sendo procurada, uma vez que endereços diferentes, com o mesmo índice serão mapeados sempre para a mesma posição da cache.
- **Mapeamento set-associativo:** O mapeamento set-associativo é um projeto intermediário entre os dois anteriores. Neste caso, existem um número fixo de posições onde a palavra pode ser armazenada (pelo menos duas) que é chamado um conjunto. Uma cache com um conjunto de duas posições é chamada 2-way set-associativa. Como na cache com mapeamento direto, o conjunto é definido pela parte do endereço chamada índice. Cada um dos tags do conjunto são comparados simultaneamente como tag do endereço. Se nenhum deles coincidir ocorre um cache miss.

Outro problema a ser considerado na implementação de uma memória cache é a política de substituição das palavras. Deve-se responder a seguinte pergunta: ‘Em que local da cache será colocada a nova linha?’ A política de substituição define qual linha será tirada da cache para dar lugar a uma nova. No caso do mapeamento direto, cada palavra tem um lugar predefinido, então não existe escolha. Para o mapeamento completamente associativo, pode-se escolher qualquer posição da cache e no set-associativo, qualquer posição dentro do conjunto definido pelo índice. As principais políticas de substituição são:

- **Substituição Aleatória:** Neste caso é escolhida uma posição qualquer da cache aleatoriamente para ser substituída. É mais simples de implementar mas não leva em conta o princípio da localidade temporal.
- **First-in First-out:** Remove a linha que está a mais tempo na cache. Exige a implementação de uma fila em hardware.
- **LRU - Least Recently Used:** Menos recentemente utilizado. Remove-se a linha que a mais tempo não é referenciada. Exige implementação de um contador para cada linha. Quando um hit ocorre na linha seu contador é zerado enquanto todos os demais são incrementados. Quando for necessário substituir uma linha, será retirada aquela cujo contador tiver o valor mais alto.

Uma vez que todas as solicitações feitas a memória são realizadas através da cache, caso a CPU fizer a escrita de uma posição que estiver armazenada na cache, se esta alteração não for repassada para a MP, pode-se perder a atualização quando a linha da cache for substituída. Para evitar este problema pode-se adotar duas estratégias:

- **Escrita em ambas (Write-through):** Cada escrita na cache é imediatamente repetida na memória principal;
- **Escrita no retorno (Write-back):** As escritas são feitas apenas na cache, mas ela será escrita na MP quando for substituída. pode-se escrevê-la mesmo se não foi alterada ou somente se tiver sido modificada.

Com a política “write-through” pode haver uma grande quantidade de escritas desnecessárias na MP, com natural redução do desempenho do sistema. Já a política “write-back” minimiza esta desvantagem, porém a MP fica potencialmente desatualizada para utilização por outros dispositivos a ela ligados, como módulos de E/S, o que os obriga a acessar dados na cache.

Uma vez que tanto os dados como as instruções são mantidos na memória principal, ambos tiram proveito da memória cache. Pode-se adotar cache unificada ou separada para cada tipo de conteúdo. A cache separada possui como vantagem possuir uma Cache de instruções somente de leitura, o que simplifica o circuito e barramento independente ligando cada uma das caches e o processador, o que permite transferências simultâneas. Além disso, pode-se adotar estratégias diferentes para cada cache (tamanho, organização, etc.). Por outro lado, a abordagem de cache separada apresenta como desvantagem o fato de que instruções e dados colocados em posições próximas da memória poderão estar nas duas caches.

4 Unidades de discos e discos removíveis

Como a memória RAM se apaga quando desligamos o micro, devemos ter outros meios de armazenar dados e programas, de forma que eles possam ser utilizados no futuro. Dados e programas devem estar armazenados em memória secundária, geralmente discos magnéticos (disquetes e H.D.) ou ainda meios óticos, disponíveis em CD ou DVD. Somente assim, com as informações armazenadas em um meio não-volátil poderão ser recuperadas.

Dessa forma, temos praticamente 2 tipos de memória no micro: a principal RAM e a secundária, formada por unidades de disco. Por vezes ainda existe uma referência a memória terciária, composta de mídias de cópias de segurança (discos óticos, fitas ...).

A quantidade de unidades de discos presente num micro comum é limitada pelas suas conexões nas controladoras na placa-mãe (Figura 16), pelo número de extensões de fonte de alimentação e pelo endereçamento de cada unidade. Usualmente, podemos ter até 2 unidades de disco flexível (disquete, *floppy*, ...) e até 4 unidades de H.D. (contando junto às unidades de CD-ROM e DVD).

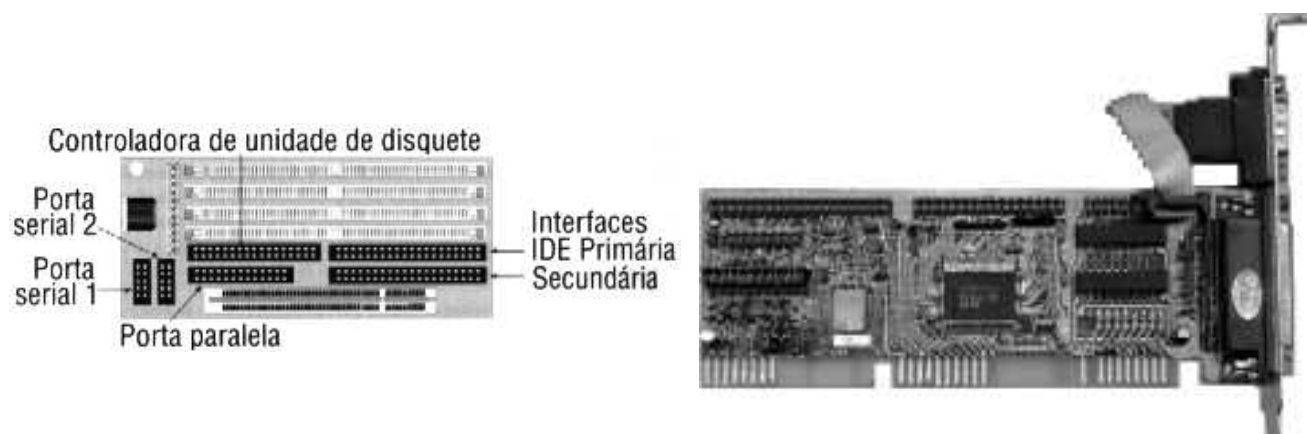
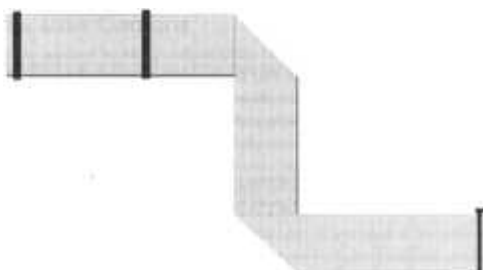


Figura 16 – Controladoras de unidades de disco

Atualmente, as controladoras de disco encontram-se *on-board* (integradas a placa-mãe), com exceção de micros antigos que usam placas controladoras de unidades de disco IDE plus ou Super-IDE ou controladoras de melhor desempenho como SCSI (algumas placas-mãe também já a disponibilizam on-board).

4.1 Cabos e configurações físicas



Para a ligação entre as unidades de disco e sua controladora utiliza-se um cabo plano e flexível, chamado *flat cable*. Conhecido também como cabo lógico, possui uma marcação do pino 1 que deverá coincidir com o pino 1 no periférico. A marcação é feita através de um fio de cor diferenciada, geralmente vermelho, sendo que o flat cable geralmente é cinza, com os conectores em preto.

Determinados periféricos assim como a maior parte das placas, incluindo a placa-mãe, necessitam de configurações. As configurações são realizadas na maioria dos casos através de *jumpers* ou ainda por DIP-Switches (microchaves). Tanto a microchave como o jumper servem apenas para fazer um contato, habilitando ou desabilitando alguma função no periférico ou placa.

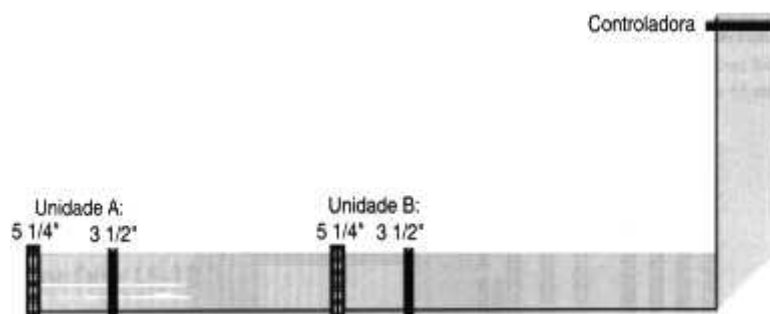
4.2 Unidade de disquetes

A unidade de disquetes é o elemento responsável pela leitura/gravação em um disquete. Como existem diversos tipos de disquete com capacidades de formatação diferentes, haverá unidades de disquete diferentes para cada tipo de disquete.

Existem disquetes com diferentes tamanhos físicos e capacidades. Existem disquetes de 5¼" de 360KB e 1,2MB e de 3½" de 720KB, 1,44MB e 2,88MB. É comum encontrar unidades de 5¼" que aceitem 360KB e 1,2MB, assim como unidades de 3½" que aceitem 720KB e 1,44KB, sendo estas últimas as mais utilizadas atualmente. Os disquetes de 2,88MB e suas unidades são difíceis de encontrar.

A unidade de disquete é conectada a controladora de unidades de disquete, que pode estar *on-board* ou não. É fácil de localizá-la, pois é o único conector com 34 pinos existente no micro. Pode-se teoricamente ligar 4 unidades de disquete no micro, mas o mais comum é encontrar controladoras que controlam no máximo 2 unidades. Elas são referenciadas em S.O. baseados no DOS, como A: e B:, sendo que a segunda só irá existir com a presença da primeira.

Para diferenciar a unidade A da B, o recurso utilizado é o próprio *flat cable*, que recebe o nome de cabo de inversão, por ter na extremidade uma inversão de alguns pinos, indicando que a unidade de disquete conectada a ponta do cabo será a unidade A. Os conectores das unidades de 5¼" e 3½" são diferentes, sendo o primeiro maior e possuindo uma placa plástica que impede a conexão invertida. Já os conectores de unidades 3½" permitem a inversão, devendo-se ficar atento ao lado em que se encontra o pino 1 do cabo e da unidade de drive.



4.3 H.D.s

O disco rígido (H.D.) é uma das melhores formas de armazenamento de grandes quantidades de dados para uso posterior. A idéia é simples, são vários discos magnéticos (semelhantes a disquetes, porém mais confiáveis e com maior precisão) empilhados um sobre o outro, com várias unidades de leitura (geralmente 2 para cada disco). A caixa preta que os envolve compõe o H.D. A Figura 17 ilustra um H.D.

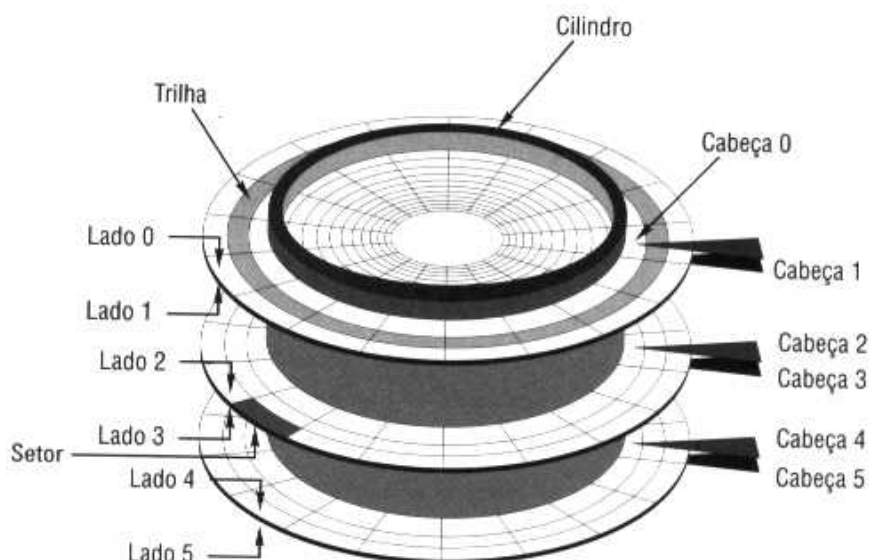


Figura 17 – Esquema de um HD

Diferente dos disquetes, que possuem um tamanho pré-definido, os H.D. utilizam alguns parâmetros de configuração. Estes parâmetros também estão presentes nos disquetes, porém sempre pré-definidos (2 *heads*/80 *cyl*/18 *sect*). São eles:

- Número de cabeças (*heads*): é a quantia de faces que o disco possui. Por exemplo, se forem 3 discos magnéticos, serão 6 cabeças (2 para cada face dos discos);
- Número de trilhas: são círculos concêntricos que dividem a área de armazenamento, em alguns casos referenciados por número de cilindros;
- Número de setores: são as divisões das trilhas. Dentro de cada setor cabem sempre 512 bytes (mesmo em H.D.s este valor é fixo).

Para calcular a capacidade de um disco rígido, utiliza-se a fórmula abaixo:

$$\text{Capacidade de armazenamento} = \text{trilhas} \times \text{setores} \times \text{cabeças} \times 512$$

A grande maioria dos micros hoje utiliza o processo de detecção automática, porém todos os H.D.s têm (ou deveriam) estes dados estampados em sua capa metálica, de modo que se o micro não possuir detecção automática ou a mesma não esteja funcionando corretamente, o H.D. possa ser configurado.

A ligação do H.D. com sua controladora ocorre por meio de um *flat cable* de 40 posições, sendo necessário observar a correta posição do pino 1, pois o conector deste cabo pode ser ligado invertido. Alguns cabos mais recentes possuem conectores com uma saliência, evitando que se conecte cabos virados.

Um micro pode conter 2 controladoras de H.D. sendo ligadas em cada uma até 2 unidades. As controladoras utilizam a interface IDE (*Integrated Drive Eletronics*) e podem ser referenciadas como IDE1 e IDE2 ou IDE primária e IDE secundária. Cada controladora deve conhecer qual é o H.D. principal, referenciado por *master*, e qual o auxiliar, conhecido por *slave*. A Figura 18 mostra a posição onde geralmente os HDs possuem jumpers para configuração.

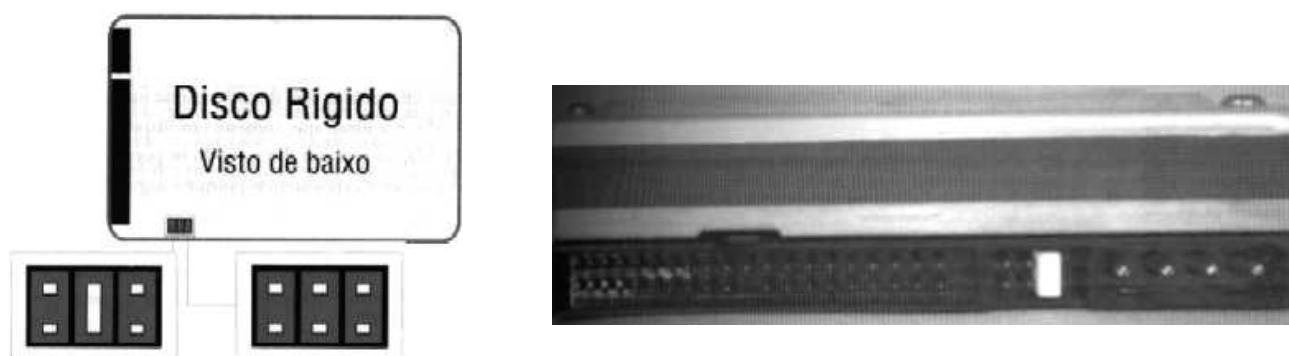


Figura 18 – Posição dos jumpers de configuração

Ao contrário dos disquetes onde a configuração é pela posição do cabo, os H.D.s são configurados com o uso de *jumpers*. Os jumpers de configuração localizam-se geralmente na parte posterior, entre o conector de dados e fonte de alimentação, ou junto à placa lógica do disco. O esquema de *jumpers* da configuração dos H.D.s também deve vir estampado na capa.

4.3.1 Anatomia do H.D.

Uma unidade de disco rígido é o componente que mais trabalha em um micro. Os discos magnéticos que os compõem, onde os dados são armazenados, giram a altas velocidades durante todo o tempo em que o micro está ligado. Cada acesso ao disco rígido, faz com que as cabeças de leitura/gravação se movimentem, sempre com precisão microscópica, sendo que a distância entre uma cabeça de leitura/gravação e o meio magnético é menor do que um fio de cabelo humano. Tão severas restrições e condições para o correto funcionamento dão a impressão de um ambiente sujeito constantemente a defeitos; mas ao invés disto, o que ocorre são unidades de disco rígido desempenhando com afinco a função de fiéis depositários de informação. A Figura 20 ilustra o interior de um HD.

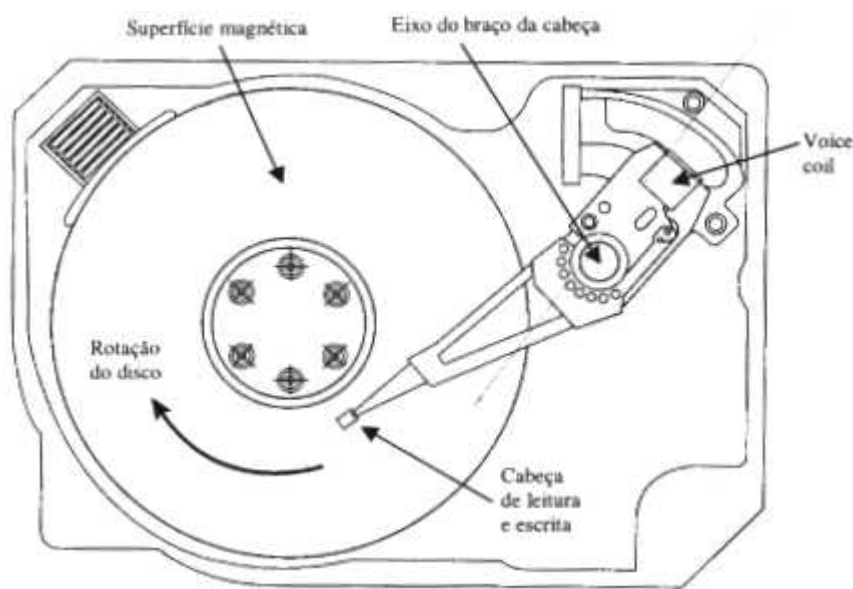


Figura 19 – Visão interna de um HD

Por ser lacrado, o H.D. trabalha com uma precisão muito maior em relação aos disquetes. Por ser fixo, o tamanho das cabeças de leitura/gravação pode ser reduzido, diminuindo também o campo magnético para armazenamento de dados, possibilitando a gravação de dados mais próximos, e, também, maior quantidade de dados em reduzido espaço físico.

Todos os discos rígidos possuem um motor, que faz com que o conjunto de discos gire a uma velocidade de pelo menos 3.600rpm (em geral 4.500 9.000rpm). Desde o primeiro instante de alimentação elétrica no H.D. os discos começam a girar e não param, sendo que as cabeças de leitura/gravação nunca encostando neles. Nos primeiros H.D.s, quando era desligada a energia elétrica as cabeças de leitura/gravação permaneciam no lugar do último acesso, possibilitando assim que informações fossem perdidas na ocorrência de choques mecânicos. Como solução, devia-se antes de desligar, estacionar as cabeças em um lugar seguro (*park*). Atualmente, não existe esta preocupação, pois os H.D.s usam um motor baseado num atuador eletromagnético “*voice coil*” que automaticamente recolhe as cabeças.

Todos os setores do H.D. são numerados, porém a numeração não segue a ordem sequencial em uma face, mas sim a ordem dos cilindros, com setores “vizinhos” localizando -se na mesma trilha, porém em outro disco (ou face). Juntamente com os 512bytes armazenados em cada setor, guardam-se outras informações:

- *gap*: espaço entre setores;
- Cabeçalho do setor: contém informações físicas como seu cilindro, lado e número. Usado para comparações;
- CRC (*Cyclical Redundance Check*): armazena valor para verificação do cabeçalho do setor;
- Dados: 512bytes de informação;
- ECC (*Error Correction Code*): armazena o código de correção dos dados armazenados.

O primeiro setor de uma unidade de disco, o setor 1, é conhecido como MBR (*Master Boot Record*), que identifica como e onde o S.O. está localizado e se ele pode ser carregado para a memória RAM. Ele também é conhecido como “*Master partition table*”, tabela mestre de partições, por incluir informações relativas as partições do disco como localização, tamanho, tipo e flag de ativação. Além destes dados, o MBR também inclui um programa que lê o *boot sector* da partição que contém o S.O. ativo, que se encarrega de disparar o programa que carrega o restante do S.O. na máquina. A Figura 20 ilustra um HD com duas partições indicando a área de MBR.

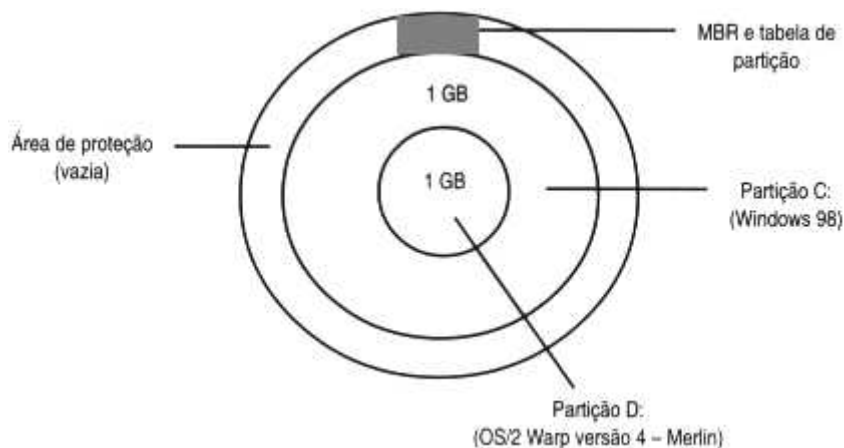


Figura 20 – MBR e partições

Uma partição é uma divisão lógica do disco criada para se possa ter diferentes S.O.s num mesmo disco, ou criar a aparência de se possuir diversos discos no gerenciador de arquivos, sendo criada na formatação do disco. Existem dois tipos de formatação: a física (baixo nível) que divide a superfície magnética em trilhas e setores e a lógica (alto nível) que prepara os setores para uso do S.O. Antes da formatação lógica, um disco necessita da definição da tabela de partições (MBR). Assim, o processo de formatação de um H.D. seria:

- Formatação física (os discos atuais não podem ser formatados em baixo nível);
- Particionamento;
- Formatação lógica.

Atualmente, os programas de formatação em baixo nível, não realizam uma formatação do disco, mas sim, procuram setores defeituosos no disco e atualizam o mapa de setores defeituosos, ou ainda, realizam a substituição por setores reserva. Caso utilize estes programas (deve-se optar pelos fornecidos pelo fabricante) os setores defeituosos irão sumir de sua visualização, porém continuarão a existir.

4.3.2 Desempenho

Apesar do giro rápido e constante de um disco rígido, ele não é capaz de enviar instantaneamente as informações solicitadas, sempre há um atraso chamado de latência ou tempo de acesso, medidos em milissegundos. Esse é o tempo gasto para a localização de um setor específico, determinado pelo tempo

necessário para mover a cabeça de leitura/gravação até o cilindro desejado, ativar a cabeça e esperar que o cilindro seja lido. Como o posicionamento da cabeça em uma trilha é fixo, deve-se esperar que o disco gire e que o setor desejado passe pela cabeça. Atualmente, os H.D.s possuem tempo de acesso de até 15ms.

Para agilizar a transferência de dados, os discos IDE possuem uma pequena memória volátil. Quando o sistema operacional lê um setor, o disco rígido lê a trilha inteira e armazena nesta memória, para num futuro acesso poder entregar os dados sem consultar a superfície magnética (*cache* ou *buffer*). O cache de disco pode também ser feito em software (smartdrv, Windows 9x: propriedades do sistema de arquivo), com a mesma idéia, porém utilizando a memória RAM para esta finalidade. A Figura 21 demonstra esta organização.

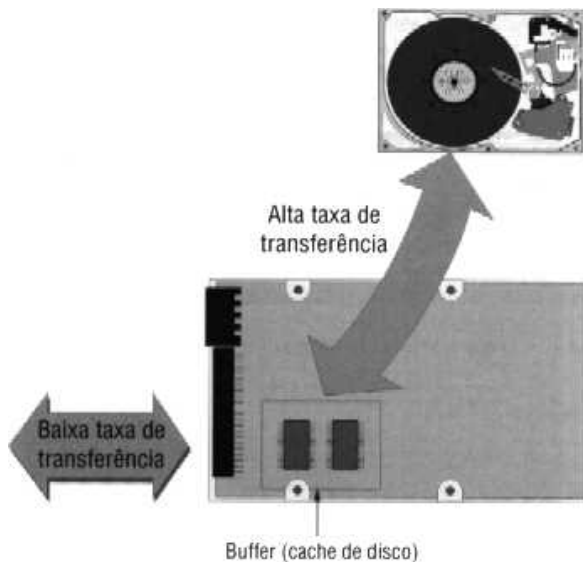


Figura 21 – Transferência de dados em HDs

Além do cache, os discos rígidos modernos conseguem transportar mais de um setor por vez, o que faz com que o desempenho do micro aumente. Essa característica é conhecida como *blockmode* (tamanho do bloco), e pode ser habilitada no SETUP do micro.

4.3.3 Padrões de conexão

A comunicação de um H.D. IDE com o micro é extremamente simples de ser implementada, uma vez que a controladora do disco rígido está integrada diretamente no próprio disco. Essa tecnologia, criada pela Western Digital, eliminou o problema de ruído na comunicação entre a controladora e o H.D.

A conexão de discos rígidos IDE ao micro usam a especificação ATA (*AT Attachment*) provida através de um conector de 40 pinos sobre um *flat cable*. Sua taxa de transferência original vai até 4MB/s. A norma ATA não é específica de discos rígidos, mas pode ser usada também para discos removíveis, leitores de CD, fitas e outros dispositivos de armazenamento, através da interface ATAPI (*ATA Packet Interface*), lançada junto com o padrão IDE Avançado (EIDE - *Enhanced IDE*). A interface ATAPI usa o mesmo conector e cabo da ATA, com mudança no protocolo de transferência de dados introduzindo taxas de transferência mais elevadas e cuidados para evitar a interferência entre dispositivos ATAPI e IDE. Com o passar do tempo, padrões como ATA-3, ATA-4 e ATA-5 também foram desenvolvidos.

Para se comunicarem com o micro, os H.D.s IDE utilizam um circuito chamado PIO (*Programmed I/O*), controlado pelo processador, por isso muitas vezes chamado de *Processor I/O*. O PIO trabalha em 6 modos de operação, atingindo taxas de até 22MB/s. A Tabela 6 reúne algumas tecnologias e taxas de transferência.

Pode-se também deslocar o controle das transferências de dados do processador para o *chipset* da placa-mãe, habilitando o recurso chamado *Bus Mastering*. Dessa forma o desempenho do micro aumenta, pois o processador pode realizar outras tarefas ao mesmo tempo em que os dados estão sendo transferidos do H.D. para a memória RAM. O *Bus Mastering* é executado pela ponte-sul do *chipset*, sendo também conhecido como modo DMA *Bus Master*. Este recurso possui diversos modos de operação e pode atingir taxas de transferência de até 33MB/s.

Tabela 6 – Taxas de transferência de HDs

Tecnologia	Modo	Taxa de Transferência	Padrão
PIO	0	3,3 MB/s	ATA
	1	5,2 MB/s	ATA
	2	8,3 MB/s	ATA
	3	11,1 MB/s	ATA-2
	4	16,6 MB/s	ATA-3
	5	22 MB/s	ATA-3
DMA	0	4,2 MB/s	ATA
	1	13,3 MB/s	ATA-2
	2 (UDMA 0)	16,6 MB/s	ATA-3
	3 (UDMA 2)	33,3 MB/s	ATA-4
UDMA	0	16,6 MB/s	ATA-3
	1	25 MB/s	ATA-4
	2	33,3 MB/s	ATA-4
	3	44,4 MB/s	ATA-5
	4	66,6 MB/s	ATA-5

Atualmente, através do protocolo UltraDMA, discos rígidos IDE podem atingir taxas de transferência de até 66MB/s. Conhecido também como UDMA, possibilita taxas de transferência que só eram atingidas no passado por discos SCSI. A tabela ao lado ilustra as taxas de transferências máximas teóricas dos modos de ligações de H.D. Todavia, não é só o modo PIO, DMA ou UDMA que influi no desempenho do H.D.: o cache do disco e o *blockmode* também interferem.

Para utilizar todo o poder do UDMA, além do disco suportar esta tecnologia, a placa-mãe aceitar este padrão e estar configurada, os modos 3 e 4, que utilizam o ATA-5, necessitam de um cabo especial com 80 fios, porém utilizando o mesmo conector de 40 pinos.

4.4 Limites em HDs

Além das limitações impostas pela capacidade dos H.D.s, existem também limitações impostas pelo gerenciamento dos H.D.s (controladoras de disco), pelo BIOS da máquina e também pelo sistema de arquivos. Todos estes limites dizem respeito a capacidade com que o H.D. pode ser formatado: mesmo possuindo uma capacidade maior, ela não pode ser utilizada.

4.4.1 Limite de 504Mb

A definição do BIOS do IBM PC prevê 10 bits para a indicação do cilindro, 8 bits para o número de cabeças e 6 bits para setores. Isto estabelece um limite de 1024cilindros, 256cabeças e 63 setores por trilha (0 não é válido para número de setor). Com isso, a capacidade máxima da BIOS é de 7,8GB. As controladoras IDE, por outro lado, utilizam 16bits de cilindro, 4bits de cabeças e 6bits para o setor,

resultando em uma capacidade de 31,5GB. Observando-se as restrições de tamanho impostas pela BIOS e pelas controladoras, têm-se 10bits para cilindros, 4bits para cabeças e 6bits para setores, resultando em capacidade máxima de 504MB (1024 x 16 x 63 x 512).

Atualmente, há um modo chamado LBA (*Logical Block Addressing*), que foi criado para transpor esse limite. Nas máquinas que não possuem este modo, pode-se utilizar um driver fornecido pelo fabricante. O modo LBA surgiu junto com o padrão IDE avançado (EIDE).

A idéia do LBA é numerar os setores de um H.D. seqüencialmente ao invés da tríade cilindro/cabeça/setor. Assim, determinado setor seria conhecido pelo seu número de ordem dentro do H.D. O BIOS passaria a pedir o setor 1000 em vez da posição física cilindro/cabeça/setor. A formatação e a criação das partições podem ser realizadas como em qualquer H.D.

Um ponto a ser levado em conta é que uma vez formatado o disco pelo modo LBA, utilizando uma determinada geometria (cilindro/cabeça/setor), esse disco somente será acessado com a mesma geometria. Não podemos acessá-lo no modo normal, a não ser após nova formatação.

4.4.2 Limite de 2Gb

O sistema de arquivos FAT16, amplamente utilizado pelos sistemas operacionais do mercado, possui um limite de 2GB por partição. Esse sistema de arquivos, poderia acessar 65536 diferentes setores (2^{16}), acessando assim míseros 32MB. Uma idéia utilizada antes de aumentar o número de bits (FAT32) foi à criação do cluster. Como o nome diz, é um agrupamento de setores, dessa forma ao invés de apontar um setor, seria apontado um conjunto de setores. A definição do tamanho do cluster, utilizado ocorre na formatação, utilizando 16KB para discos de até 1GB e 32KB para discos de até 2GB em discos FAT16.

Um problema da organização em clusters é o desperdício de espaço, uma vez que um cluster pode conter somente 1 arquivo. Por exemplo, um arquivo de 100KB em um sistema com cluster de 32KB, ocupará 4 clusters (128KB). Dessa forma, quanto maior o tamanho do cluster, maior o desperdício.

Um problema de diminuir o tamanho do cluster é a capacidade máxima do disco. Então a Microsoft lançou o sistema FAT32, utilizando 32bits para o endereçamento de setores, podendo assim diminuir o tamanho do cluster (e também o desperdício) e acessar partições maiores que 2GB. A tabela ilustra o tamanho dos clusters e a capacidade máxima de acesso em discos FAT32.

Tamanho do Cluster	Capacidade máxima
512bytes	256MB
4KB	8GB
8KB	16GB
16KB	32GB
32KB	2TB

Fonte: Torres, Gabriel. Hardware Curso Completo

4.4.3 Limite de 8Gb

Com os limites da BIOS (1024 cilindros, 256 cabeças e 63 setores), mesmo utilizando o modo LBA, a capacidade máxima de um disco é de 7,8GB. Isso é limitado pelo fato de se usar 24bits para definir um setor e agravado por não existir o setor 0.

Os acessos por software a unidades de discos são controlados por uma interrupção, a INT 13H, que repassa os pedidos a BIOS. Para permitir acessos a discos maiores foi criada uma série de expansões na INT 13H. Estas alterações, denominadas EDD (*Extended Disk Drive*) estão disponíveis nos micros que utilizam um *chipset* recente, permitindo assim um limite fixado pelo padrão ATA: 127GB.

4.5 CD-ROM

As unidades de CD-ROM quando inicialmente lançadas utilizavam um cabo específico que era ligado à placa de som. Recentemente a ligação de dados das unidades de CD-ROM foi trocada, passando a ser utilizada a interface IDE. Dessa forma, uma unidade de CD-ROM é ligada ao micro como se fosse uma unidade de disco rígido, devendo ser configurada para *master* ou *slave*, através de *jumpers*, e tomando cuidado com a correta posição do pino 1 do *flat cable*.

Apenas os dados trafegam pelo cabo de 40 vias, uma vez que as unidades de CD-ROM possuem outro cabo, com geralmente 3 vias, para a transmissão do áudio, ligado diretamente na placa de som.

Quando só existia CD de áudio, desempenho não era problema, uma vez que o áudio é escutado a uma velocidade fixa, com a taxa de transferência do aparelho de CD de áudio sendo 176.400 bytes/s. A taxa de transferência padrão das unidades de CD-ROM no modo 1 (1X) é de 153.600 bytes/s, ou 150KB/s. Uma taxa muito baixa se comparadas às taxas de transferências dos H.D.s, mesmo no padrão ATA inicialmente especificado.

O aumento da taxa de transferência do CD-ROM foi executado basicamente aumentando-se a velocidade de rotação do CD na unidade. Uma unidade 2X apresenta uma taxa de transferência que equivalente ao dobro da usada na unidade padrão (300KB/s). A tabela ao lado ilustra a taxa de transferência de algumas unidades de CD-ROM.

Velocidade	Taxa de transferência Nominal
1 x	150 KB/s
2 x	300 KB/s
8 x	1.200 KB/s
24 x	3.600 KB/s
36 x	5.400 KB/s
44 x	6.600 KB/s
52 x	7.800 KB/s
56 x	8.400 KB/s
60 x	9.000 KB/s

A taxa de transferência informada é a máxima, que nem sempre é atingida, principalmente em unidades com alta taxa de transferência. Isso ocorre, pois a velocidade é constante e somente as informações dos setores mais externos conseguem obter a taxa nominal. Efetivamente a taxa de transmissão varia da metade do valor nominal até ele.

Outro fator que mede o desempenho da unidade de CD-ROM é o tempo de acesso, que não é baixo, pois o mecanismo da cabeça é lento para se mover de uma trilha para outra. Este tempo de acesso que nos H.D.s era próximo a 1 dezena de milissegundos, em unidades de CD-ROM ficam em algumas dezenas de ms.

5 Entrada e Saída

Para que se possa desfrutar da rapidez e flexibilidade de um computador, não basta saber que ele pode armazenar na memória os programas e dados que desejamos processar e nem que ele pode executar mais de um milhão de instruções por segundo. É preciso que o programa que temos escrito, por exemplo, em uma folha de papel e os dados a serem manipulados por ele sejam inseridos no sistema, caractere a caractere, incluindo os espaços em branco, sinais de pontuação e símbolos de operações matemáticas. Para realizar estas tarefas é necessário um meio qualquer que faça esta comunicação homem-máquina.

Da mesma forma que se necessita de comunicação com a máquina, também é preciso comunicação no sentido contrário (máquina-homem), de modo que o usuário possa entender os resultados de um processamento. Os dispositivos responsáveis pela entrada/saída são também denominados de periféricos (instalados fora do núcleo principal CPU/MP ficam na maioria das vezes próximos, na periferia).

O funcionamento dos dispositivos de entrada/saída em um computador é geralmente caracterizado pela existência de diversos elementos que, embora realizem o mesmo tipo de função, possuem características diversas. Desta forma, costuma-se integrar os diversos elementos que cooperam no processo de entrada e saída num subsistema, parte do sistema de computação. Um subsistema de I/O deve ser capaz de realizar duas funções:

- receber ou enviar informações ao meio exterior;
- converter as informações (de entrada ou de saída) em uma forma inteligível para a máquina (se estiver recebendo) ou para o usuário (se estiver enviando).

Os periféricos que compõem o subsistema de I/O possuem diferentes características, como, por exemplo, velocidade de operação e taxa de transferência das informações. Na Figura 22 estão ilustradas algumas características dos principais periféricos de I/O.

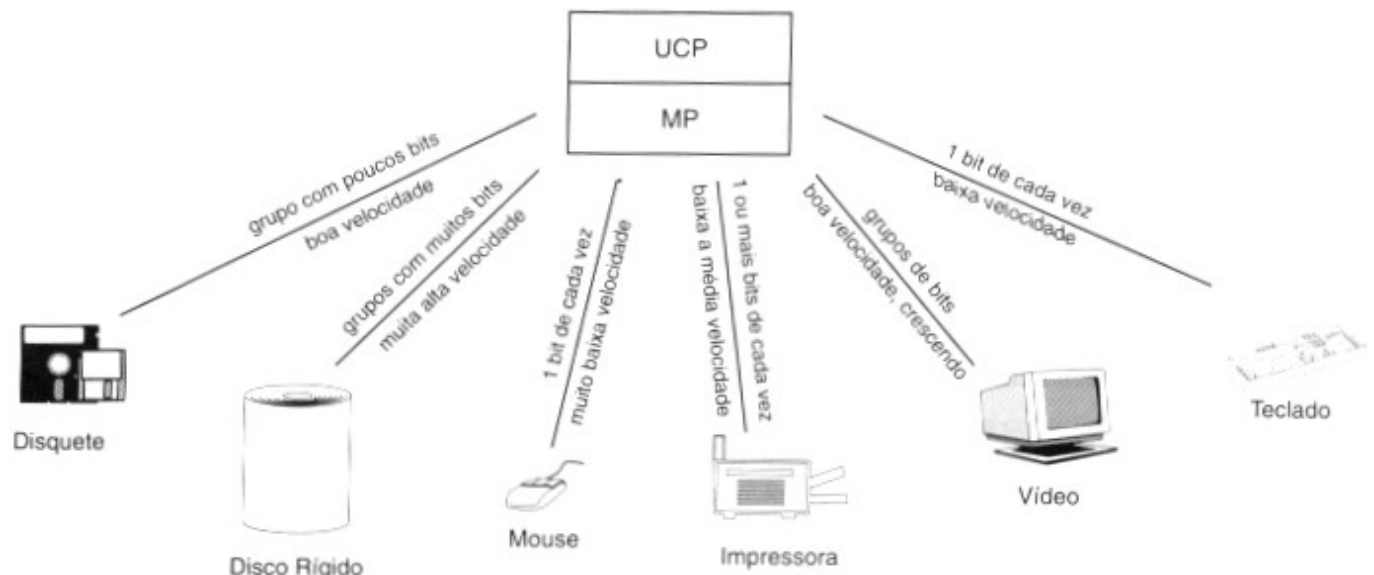


Figura 22 – Características dos principais periféricos de I/O

Devido às diferenças, a ligação entre os dispositivos de I/O e a CPU não é realizada direta e individualmente. Uma ligação direta aconteceria se houvesse uma ligação direta entre CPU e teclado, por exemplo. Na prática, a CPU se conecta a dispositivos que realizam a tradução e a compatibilização das características de um (periféricos de I/O) para outro (CPU), além de realizar outras tarefas de controle.

Estes dispositivos costumam ser chamados de interface de I/O, porém, há outros nomes como controlador, canal e adaptador. Independente do nome a ser empregado, a função é a mesma: compatibilizar as diferentes características de cada periférico e da CPU/MP, permitindo um fluxo correto de dados em uma velocidade adequada a ambos os elementos sendo conectados.

Há duas maneiras básicas de se realizar a transmissão/recepção de dados entre periféricos/interfaces e interfaces/CPU:

- Transmissão serial: a informação é transmitida/recebida bit a bit;
- Transmissão paralela: a informação é transmitida/recebida em grupos de bits de cada vez.

A velocidade de transferência de dados de um periférico é geralmente muito menor que a da CPU. Não é eficaz conectar os periféricos ao barramento do sistema (local), pois isto reduziria a velocidade da comunicação entre CPU e MP. Além disso, cada periférico tem uma velocidade diferente, por exemplo, o teclado é muito mais lento que o disco. Desta forma, não é utilizado um único caminho, mas vários, ligados ao barramento de expansão.

Os periféricos de I/O utilizam diferentes tamanhos de unidades de transferência de dados (alguns transferem 1 bit, outros 1 byte enquanto outros podem enviar/receber centenas ou mesmo milhares de bits num único bloco de transferência). Da mesma forma, os fabricantes adotam diferentes tamanhos de palavras de dados.

Todos estes aspectos garantem ser necessário o emprego de um dispositivo intermediário na ligação da CPU/MP a um periférico de I/O. Estas interfaces podem servir a ligação de um único dispositivo (vídeo) ou vários (USB). Uma interface ou um controlador é, em geral, responsável pelas seguintes tarefas:

- a) Controlar e sincronizar o fluxo de dados entre CPU/MP e o periférico;
- b) Realizar a comunicação com a CPU, inclusive interpretando suas instruções (sinais de controle) para acesso físico ao periférico;
- c) Servir de memória auxiliar para o trânsito das informações entre os componentes (buffer de dados);
- d) Realizar algum tipo de detecção e correção de erros durante as transmissões.

A utilização de um buffer interno pelas interfaces é um fator fundamental para a compatibilização de velocidades diferentes entre o barramento do sistema e suas linhas externas.

5.1 Conexões de periféricos de entrada e saída padrões

Grande parte das placas-mãe dos microcomputadores atuais incorporam os circuitos de controle dos principais periféricos de entrada e saída, porém ainda encontram-se casos que os mesmos são ligados a placas auxiliares (conectadas em slots do barramento de expansão). A conexão com o teclado, por exemplo, é diretamente na placa-mãe há muito tempo, enquanto que o controle de vídeo passou há pouco tempo ser incorporado, porém, pode-se ainda instalar uma placa auxiliar de vídeo em slots.

As principais conexões de entrada e saída do micro são o monitor, teclado, porta paralela e portas seriais. Cada um destes possui uma especificação diferente de pinos e conectores específicos, sendo neles ligados os mais diversos periféricos. O mouse, que não é considerado periférico padrão de

entrada, há muito tempo vem sendo utilizado, portanto, deve-se saber como e onde podemos conectá-lo. As portas de comunicação são utilizadas para que o micro possa se comunicar com periféricos externos e até mesmo outros micros.

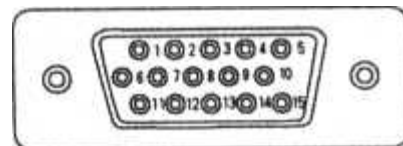
5.1.1 Monitor

Todo o micro necessita de um sistema de vídeo para que se possa monitorar sua operação, ver o que se digita ou carrega, acompanhar o andamento de cálculos e verificar os resultados. Os monitores usados com micros hoje em dia são geralmente coloridos e possuem características como a resolução máxima, frequência de varredura e o *dot pitch*.

Os monitores são ligados a placas de vídeo, que têm algumas características básicas: quantidade de memória, modelo do *chipset* (controlador) e tipo de barramento utilizado (ISA, VESA, PCI, AGP, ...). Algumas placas-mãe possuem controlador de vídeo integrado (*on-board*), porém mesmo nestes micros pode-se instalar outra placa de vídeo.



Independente de onde está localizado o circuito controlador de vídeo, os computadores atuais utilizam o padrão super-VGA (*Video Graphics Array*). A conexão entre o monitor e o adaptador de vídeo ocorre com um conector de 15 pinos (DB-15), sendo que o cabo que é ligado ao monitor



possui uma interface “macho” e o conector da placa (ou adaptador no caso de *on-board*) uma interface fêmea.

Com o aumento da utilização de aplicações gráficas, sentiu-se a necessidade de recursos para a aceleração de vídeo. Praticamente todas as placas de vídeo, independentes ou não (placas 2D), possuem estes recursos. Um problema ocorre com aplicações em 3D, utilizando objetos que demandam muitos cálculos matemáticos e consumindo grandes áreas de memória. Para solucionar este problema, pode-se instalar juntamente com uma placa 2D uma (ou mais) placa aceleradora 3D.

5.1.2 Teclado

O teclado ainda é o periférico de entrada mais utilizado. Desde o início de sua utilização em micros, o número de teclas vem aumentando, começando pelo teclado do PC original com 84 teclas até os dias de hoje com 101 teclas ou mais. Utilizam em sua maioria o padrão QWERTY US-*Internacional*, porém a ABNT já padronizou um *layout* de teclas (baseado no QWERTY, porém com caracteres a mais como a cedilha e uma melhor posição dos acentos gráficos) para ser usado com palavras da língua brasileira.

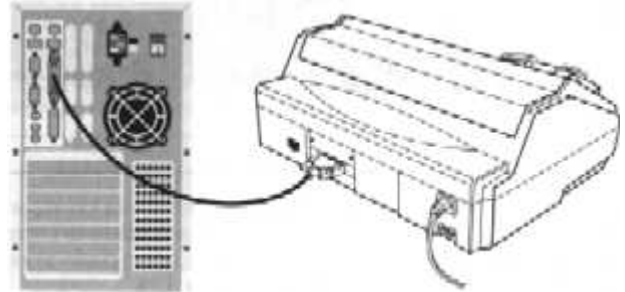
Para sua ligação a placa-mãe do micro, utilizam basicamente dois padrões: o DIN de 5 pinos e o mini-DIN com até 6 pinos, introduzido pelo PS/2. Pode-se ainda utilizar adaptadores, caso a placa-mãe não possua conector para um tipo específico de teclado.



Com o objetivo de diminuir o risco de L.E.R. (Lesão por Esforço Repetitivo) em usuários, foram criados teclados ergonômicos. Outra maneira de diminuir o risco de lesões é a utilização de um apoio de punho de material emborrachado na base do teclado.

5.1.3 Porta paralela

A porta paralela permite que dados saiam do micro diretamente para um dispositivo externo byte a byte. Ela utiliza a comunicação paralela, que apesar de ser extremamente rápida e segura, está sujeita a ruídos; deste modo não pode usada para comunicar dispositivos muito longe um do outro. A porta paralela tradicional é unidirecional, permitindo apenas que dados sejam enviados pelo micro ao periférico.



A interface da porta paralela é constituída de um conector de 25 pinos (DB-25) fêmea, enquanto as impressoras que utilizam interface paralela possuem um conector Centronics de 36 pinos fêmea. Para a ligação deve ser utilizado um cabo com um conector DB-25 macho em uma ponta e um conector Centronics macho na outra extremidade. A Figura 23 mostra os conectores de cabos paralelos.



Figura 23 – Conectores de cabos paralelos

Existem basicamente 3 modos de operação das portas paralelas:

- *SPP (Standard Parallel Port)*: este é o modo de operação padrão, unidirecional com transferência máxima de 150 KB/s;
- *EPP (Enhanced Parallel Port)*: introduziu o modo bidirecional e aumentou a taxa de transferência, podendo alcançar taxas de transferência de 2MB/s, porém na prática alcança 800KB/s. Necessita de cabo com blindagem especial, conhecido como “bidirecional”;
- *ECP (Extended Capabilities Port)*: possibilita na prática taxa de 2MB/s, pois utiliza compactação automática de dados e utiliza DMA. De maneira semelhante a EPP deve utilizar cabo com blindagem especial.

Os circuitos de controle tanto da porta paralela, quanto das portas seriais vêm atualmente na placa-mãe. Dessa forma, pode-se alterar o modo de operação da porta paralela através do SETUP. Os conectores chegam até a parte traseira do micro para conexão através de adaptadores, com exceção da porta paralela em placas super-IDE, que a conexão é diretamente na placa.

O uso típico da porta paralela é a conexão do micro com a impressora. Porém, atualmente, está surgindo um número maior de periféricos que podem ser ligados a porta paralela como Zip-drive e scanners. Estes periféricos foram beneficiados pelos avanços na tecnologia da porta paralela, permitindo a comunicação bidirecional e modos avançados, com uma taxa de transmissão mais elevada.

5.1.4 Portas seriais

As portas seriais realizam a comunicação serial, enviando os dados bit a bit ao invés de palavra por palavra. Isso faz com que a comunicação seja mais lenta e propensa a erros, por outro lado, menos fios são necessários para ligar transmissor e receptor e a distância entre eles pode ser grande.

Existem dois tipos de comunicação serial: síncrona e assíncrona. No primeiro tipo há um canal para transmissão de dados e um para o sincronismo, que mostra ao receptor onde começa e termina cada conjunto de dados que está sendo transmitido no canal de dados. Na comunicação assíncrona, utilizada nos computadores, o mesmo canal que transmite os dados é responsável pelo sincronismo, enviando sinais de início e fim do conjunto de dados.

Na porta serial do micro geralmente conecta-se o mouse, muito embora cada vez surjam mais aplicações em que a porta serial é utilizada, como conexões micro à micro, de agendas eletrônicas, de câmeras digitais e modem externo. A porta serial é mais lenta que a paralela, pois além de enviar dados bit a bit, ela necessita de bits extras para o início e fim do conjunto de dados e ainda bits para os mecanismos de detecção e correção de erros.

Fisicamente, as interfaces seriais podem apresentar 2 conectores: 9 pinos (DB-9) ou 25 pinos (DB-25), ambos com conectores macho e fêmea. Em geral, o micro tem 2 portas seriais, uma com conector DB-9 macho, onde se liga geralmente o mouse, que possui um conector DB-9 fêmea, e outra com conector geralmente DB-25 fêmea, que fica na maioria dos casos disponível. A Figura 24 mostra conectores utilizados em cabos seriais.



Figura 24 – Conectores de cabos seriais

Com relação à conexão de mouse, existe um tipo chamado mouse de barramento que não é conectado a porta serial do micro e sim a uma porta própria. Chamado também de mouse PS/2, utiliza o mesmo conector mini-DIN de 6 pinos utilizado também nos teclados. Este conector pode vir diretamente na placa-mãe, ou sendo nela ligado através de um adaptador. A Figura 25 mostra conectores min-DIN.

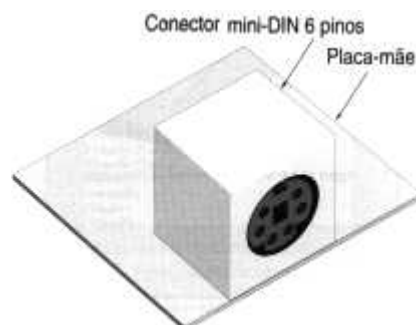


Figura 25 – Conectores min-DIN



5.2 Formas de comunicação entre CPU/MP e interface de I/O

Há duas formas de se organizar a comunicação entre CPU, memória principal e interface de entrada/saída: por memória compartilhada (“memory-mapped”) e E/S (I/O) isolada. No primeiro caso, a memória principal é compartilhada tanto por instruções e dados comuns de um programa quanto por instruções/dados de operações de I/O. O processo de entrada/saída com memória isolada consiste em criar um espaço de memória próprio de I/O e diferente, portanto, da memória principal. A figura 26 ilustra ambas as formas de organização da memória para comunicação com a interface de I/O.



Figura 26 – Formas de organização de memória para comunicação com interface de I/O

Com memória compartilhada, basta haver no barramento de controle uma única linha de leitura (“read”) de escrita (“write”), visto que a memória é a mesma e, portanto, somente poderá ocorrer em cada instante uma operação de leitura/escrita para CPU/MP ou uma para CPU/E-S. No caso da memória isolada é necessário que haja um sinal de identificação no topo de operação, sempre que um endereço for colocado no barramento, para saber se o mesmo é de I/O ou não. Os microprocessadores da família Intel e similares, por exemplo, usam mais comumente a técnica de memória isolada do que a de memória compartilhada.

Os endereços de I/O isolados são chamados portas (“ports”). No caso dos microprocessadores há em geral uma convenção que os endereços de I/O podem ter 8 ou 16 bits. Com 8 bits são endereçados dispositivos localizados na própria placa principal (mãe), como controlador de tempo (“timer”) e o interface de teclado. Com 16 bits são endereçados os dispositivos externos, como vídeo e controladores de disco.

O método de I/O isolada tem a vantagem de não utilizar espaço da memória principal, deixando-a toda para outras aplicações, mas tem a desvantagem de só ser utilizado com instruções especiais de I/O (no caso dos microprocessadores Intel, são usadas apenas as instruções IN, INS, OUT e OUTS). Além disso, também foram desenvolvidos sinais especiais de controle para o espaço de I/O. Por outro lado, o método de memória compartilhada tem a vantagem de não necessitar de qualquer instrução especial, ou seja, qualquer instrução que utilize a memória principal se aplica também a I/O. Mas tem a desvantagem de ocupar parte do espaço de memória principal para o uso de entrada/saída.

5.3 Endereços de I/O para as portas de comunicação padrão

Para a comunicação com os periféricos de entrada/saída, o processador precisa programar os circuitos de apoio periféricos da placa mãe (chipset). Essa comunicação é realizada através de uma área distinta e independente, chamada área de I/O. A área de I/O nos processadores compatíveis com o padrão x86 é de 64KB, ou seja, há 65535 endereços (0000h a FFFFh) que são utilizados pelo processador para se comunicar com algum circuito periférico (ligado ao barramento).

Para exemplificar os endereços de I/O Base, vamos utilizar a comunicação do micro com a impressora a fim de que alguma informação seja impressa. A comunicação é realizada pela porta paralela, que normalmente usa o endereço 378h. Dessa forma, quando o micro quer enviar algum dado a impressora, ele simplesmente “joga” esse dado para o endereço de I/O 378h, sendo posteriormente lido e interpretado pela impressora. Todos os periféricos de entrada ou saída utilizam um endereço de I/O para a comunicação com o processador.

As portas de comunicação utilizam endereços de entrada e saída geralmente padrão, sendo mostrados na tabela abaixo.

Tabela 7 – Endereços das portas de comunicação do padrão PC

Porta	Tipo de comunicação	Endereço de I/O (Hex)
COM 1	Serial (assíncrona)	3F8
COM 2	Serial (assíncrona)	2F8
COM 3	Serial (assíncrona)	3E8
COM 4	Serial (assíncrona)	2E8
LPT 1	Paralela	378
LPT 2	Paralela	278

Tanto as portas de comunicação serial quanto as paralelas utilizam um buffer para o envio das informações. Inicialmente, as informações a serem enviadas devem ser copiadas para o buffer para após serem transmitidas no canal (abstração do cabo de transmissão). Quando ocorre o recebimento das informações, elas chegam diretamente ao buffer, sendo após repassadas a CPU.

5.4 Formas de realização de entrada/saída

Durante a execução de um programa, por diversas vezes a CPU se depara com a necessidade de enviar ou receber dados de algum dispositivo periférico. Para tanto, da mesma forma que acontece com a comunicação CPU/MP é necessário que a CPU indique o endereço correspondente ao periférico desejado, pois há sempre mais de um periférico ligado a um sistema de computação.

Cada periférico conectado ao sistema de computação possui um endereço, denominado de endereço de I/O (port number). Conforme a quantidade de bits que tenha sido estabelecida no projeto do sistema para definir os endereços de I/O teremos o limite de periféricos que podem ser conectados ao sistema. O acesso da CPU a um periférico é obtido através do barramento do sistema e da respectiva interface do periférico.

Há 3 métodos distintos de efetuar operações de entrada e saída:

- Entrada/saída por programa;
- Entrada/saída com emprego de interrupção;
- Acesso direto a memória (DMA – Direct Memory Access).

5.4.1 Entrada/saída por programa

Por esse método, a CPU executa diretamente instruções de I/O, enviando e recebendo dados da interface de I/O. Cada instrução serve para uma ação típica, como: examinar o estado de uma interface, realizar a operação de I/O com a interface (enviar ou receber dados).

Na entrada/saída por programa toda a transferência de dados entre a CPU e o dispositivo de E/S é controlada pelo programa que o processador executa. A CPU solicita o início da transferência de dados à controladora e depois a CPU fica testando o estado do dispositivo para verificar se a operação de E/S terminou. Assim, a CPU fica em um loop de espera e teste ('polling').

A grande desvantagem deste método consiste no uso intenso de CPU. Como tem que manter um loop de execução para diversas atividades (por exemplo, verificar o estado de uma interface, se este completar uma operação e estiver com um dado disponível), ocorre um desperdício de uso em detrimento de atividades mais importantes. Neste método de I/O, uma solução para compatibilizar as diferentes velocidades entre o processador e o periférico é justamente o loop de interrogação do estado do periférico, não se devendo enviar o dado enquanto ele não estiver pronto para recebê-lo.

5.4.2 Interrupções

Na utilização de I/O por interrupção a CPU não fica continuamente interrogando sobre o estado de um dispositivo; ao invés disso, funciona da seguinte forma:

- a) A CPU emite uma instrução de I/O para a interface e, como não deverá receber uma resposta imediata, em vez de ficar continuamente verificando o estado do periférico, a CPU desvia-se para realizar outra atividade (provavelmente executar um outro programa), suspendendo a execução do programa que requer I/O;
- b) Quando a interface estiver pronta para enviar os dados do periférico a CPU, ela "avisa" a CPU através de um sinal de interrupção. Chama-se interrupção porque realmente o sinal interrompe a atividade corrente da CPU para que esta dê atenção ao dispositivo que a interrompeu;
- c) A CPU inicia então as rotinas de I/O requisitadas.

Uma interrupção consiste em uma série de procedimentos que suspendem o funcionamento corrente da CPU, desviando sua atenção para outra atividade. Quando esta outra atividade é concluída, a CPU retorna à execução do ponto que estava antes de ser interrompida.

Existem basicamente dois tipos de interrupção:

- **Internas:** Geradas pela execução de uma instrução, como por exemplo uma divisão por zero, overflow, código de operação inválido, tentativa de acesso a uma área de memória protegida ou inexistente, etc.
- **Externas:** Causadas por um sinal externo à CPU. Geralmente ativada devido a uma interface de E/S que está avisando que um determinado periférico deseja a atenção para transferir dados.

A detecção de uma interrupção faz com que o processador transfira o controle para uma rotina de tratamento de interrupção ("*interrupt handler*"). A rotina de tratamento de interrupções faz o processador executar as seguintes ações:

1. Identificar o tipo de interrupção;
2. Detectar qual o dispositivo que sinalizou (interrompeu);
3. Executar as ações apropriadas (que dependem do dispositivo),
4. Retornar ao ponto do programa em que estava quando iniciou o atendimento à interrupção.

Para que o processador possa executar esta rotina, ele deve conhecer o endereço de memória onde ela está armazenada. Alguns computadores mais simples utilizam uma única rotina para o tratamento de todas as interrupções sendo que esta rotina está armazenada em um endereço definido da memória. Para aumentar a versatilidade, a maioria dos sistemas utiliza vetores de interrupções, que armazenam o endereço da rotina específica para o tratamento de cada interrupção.

Após a CPU receber o sinal de interrupção, ela deve terminar a execução da instrução corrente e salvar todo o contexto de execução (valores de registradores como PC e ACs, entre outros). A operação de armazenamento do contexto é necessária para que quando a CPU terminar a rotina de tratamento de I/O, ela possa retornar ao ponto correto, com os valores de execução dos registradores corretos.

Desta forma, quando pressionamos uma tecla no teclado, o próprio teclado gera um pedido de interrupção, fazendo com que o processador pare o que esteja fazendo, receba os dados, prosseguindo após com o processamento normal das atividades.

Os microprocessadores da família Intel possuem apenas uma entrada para interrupção externa, o que é muito pouco, pois permite que se conecte apenas um periférico de E/S ao micro. A solução encontrada foi à utilização de um circuito controlador de interrupções, capaz de gerenciar até 8 pedidos de interrupção (IRQ0 – IRQ7). Cada linha de interrupção está conectada a um periférico distinto, não podendo dois periféricos compartilhar a mesma IRQ. Com apenas 8 pedidos de interrupção, as linhas disponíveis foram rapidamente preenchidas e no projeto do AT o número de IRQs foi aumentado, adicionando-se um segundo controlador de interrupções conectado em cascata ao primeiro controlador. Essa conexão foi realizada utilizando a IRQ2, disponibilizando 15 IRQs para uso, sendo este esquema utilizado até hoje. Os periféricos configurados para a IRQ2 foram automaticamente reconfigurados pelo chipset para a IRQ9. A tabela 8 ilustra o mapa de IRQs externas geralmente utilizadas em um PC. Convém ressaltar que os periféricos podem ser configurados para IRQs diferentes das constantes na tabela.

Tabela 8 – Mapa de interrupções da plataforma de um PC

IRQ	Ligação
IRQ0	Temporizador da placa –mãe (conectado ao chipset)
IRQ1	Teclado (conectado ao chipset)
IRQ2	Conexão em cascata (conectado ao chipset)
IRQ3	COM2 e COM4 (comunicação serial)
IRQ4	COM1 e COM3 (comunicação serial)
IRQ5	Placa de Som
IRQ6	Unidade de Disquete
IRQ7	LPT1 - Porta paralela
IRQ8	Relógio de tempo real (conectado ao chipset)
IRQ9	Interface de vídeo
IRQ10	Normalmente disponível
IRQ11	Normalmente disponível
IRQ12	Mouse de barramento (bus mouse, mouse PS/2)
IRQ13	Co-processador matemático (conectado ao chipset)
IRQ14	Porta IDE-primária
IRQ15	Porta IDE-secundária

Pode-se observar que esta modalidade de realizar operações de I/O melhorou o desempenho dos sistemas, mas ainda apresenta algumas desvantagens. Embora a CPU não necessite mais interrogar o estado de disponibilidade de um periférico, ela continua gastando tempo para executar o programa de I/O e para efetivar a transferência de dados (salvar o contexto).

5.4.3 Acesso direto à memória

Para o acesso a dados da memória, o meio a ser utilizado deve ser o processador. Porém esta regra teve de ser revista, pois imagine a carga na memória de um arquivo de 100KB; teríamos várias instruções do tipo: ‘Leia do disco e armazene no endereço X’ a serem realizadas pelo processador, o que demandaria muito tempo além atrasar o processamento de outras informações.

Para esses casos, o periférico pode usufruir um circuito de apoio chamado controlador de DMA (*Direct Memory Access* – Acesso Direto à Memória). Este circuito já existia no PC original, onde até 4 periféricos (ligados aos canais DMA0 a DMA3) que poderiam ser conectados ao controlador de DMA, melhorando o desempenho, pois enquanto uma transferência à memória via DMA estivesse sendo realizada, o processador poderia estar executando outra tarefa.

De modo geral, a técnica DMA consiste na realização da transferência de dados entre uma determinada interface e a memória principal, praticamente sem a intervenção da CPU. Esta se limita a solicitar a transferência para o dispositivo denominado controlador de acesso direto a memória (‘DMA controller’), o qual realiza por si só a transferência. A CPU fica liberada para realizar outras atividades, quando o controlador termina a transferência, ele sinaliza para a CPU através de uma interrupção.

Da mesma forma que as IRQs, o número de canais de DMA do PC original era insuficiente. Assim no projeto do AT, o controlador de DMA foi substituído por outro de 16bits (o original era de 8 bits), permanecendo o controlador antigo, por questões de compatibilidade, ligado em cascata com o novo controlado pelo DMA4. A tabela 9 ilustra os endereços para acesso direto à memória.

Tabela 9 – Mapa de DMA no PC

DMA	Ligação	Tamanho
DMA0	Normalmente disponível	8 bits
DMA1	Placa de Som	8 bits
DMA2	Unidade de disquete	8 bits
DMA3	Normalmente disponível	8 bits
DMA4	Conexão em cascata (conectada ao chipset)	16 bits
DMA5	Placa de Som	16 bits
DMA6	Normalmente disponível	16 bits
DMA7	Normalmente disponível	16 bits

Hoje em dia o controlador do DMA encontra-se no chipset, podendo manipular toda a quantidade de memória da máquina a mesma frequência de operação da placa-mãe (barramento local), liberando quase que totalmente a CPU das tarefas de transferência de dados entre periféricos de I/O e a memória.

6 Arquiteturas de computadores

Desde o desenvolvimento do primeiro computador com programa armazenado na memória, por volta de 1950, houve poucas inovações significativas nas áreas de arquitetura e organização de computadores. Alguns dos maiores avanços desde o nascimento do computador foram:

- **O conceito de família de computadores:** introduzido pela IBM com o Sistema/360, em 1964, e seguido de perto pela DEC, com o PDP-8. O conceito de família de computadores desvincula uma arquitetura de máquina de suas implementações. São fabricados diversos computadores com características de preço e desempenho diferentes, que apresentam, para o usuário, a mesma arquitetura. As diferenças de preço e desempenho são devidas às diferentes implementações da mesma arquitetura.
- **Unidade de controle microprogramada:** sugerida por Wilkes, em 1951, e introduzida pela IBM na linha S/360 em 1964. A microprogramação facilita a tarefa de projetar e implementar a unidade de controle e oferece suporte para o conceito da família de computadores.
- **Memória cache:** introduzida comercialmente no IBM S/360 modelo 85, em 1968. A adição desse componente na hierarquia de memória melhorou sensivelmente o desempenho.
- **Pipeline:** mecanismo para introduzir o paralelismo na natureza essencialmente sequencial de programas em linguagem de máquina. Alguns exemplos são os pipelines de instruções e processamento vetorial.
- **Múltiplos processadores:** essa categoria abrange grande número de organizações diferentes, com distintos objetivos.

A essa lista deve ser ainda adicionada uma das inovações mais interessantes e, potencialmente, mais importantes: a arquitetura com um conjunto reduzido de instruções (RISC).

6.1 CISC

Os processadores construídos segundo a arquitetura CISC (Complex Instruction Set Computer) possuem uma grande quantidade de complexas instruções, vários modos de endereçamento (a memória), poucos registradores de dados (propósito geral) e processamento controlado por microprograma. A filosofia é que o hardware é mais rápido que o software, então um conjunto poderoso de instruções produziria programas executáveis pequenos, ou seja, com poucas instruções.

Foram propostas e implementadas unidades lógica e aritmética complexas, unidades de multiplicação, unidades de divisão, unidades de processamento numérico com suporte à representação numérica em ponto flutuante e cálculo de funções matemáticas como trigonometria e logaritmos. Quanto à movimentação de dados, verificou-se o surgimento de inúmeros tipos de endereçamento de memória, instruções que permitiram a movimentação de blocos de dados e instruções para tratamento de vetores e matrizes.

Todo este desenvolvimento acarretou em um problema: a dificuldade em aumentar a frequência de operação. Devido à complexidade envolvida, os circuitos se tornaram muito grandes, ocupando uma área considerável em silício. Este fato dificulta a redução do ciclo de operação.

Dessa forma, os projetistas então voltaram sua atenção ao desenvolvimento de suporte a operações pelo hardware, ou seja, operações complexas de transferência de memória ou operações aritméticas passaram a serem resolvidas por circuitos eletrônicos através de uma única instrução. Vem daí a denominação de processadores CISC (Complex Instruction Set Computer).

Porém, existe um problema: muitas instruções significam muitos códigos de operação e, portanto, muitos bits para o código. Este fato acarreta duas desvantagens: instrução com maior comprimento (mais de uma palavra) e mais tempo de interpretação (decodificação) da instrução. Além disso, instruções muito complexas demoravam vários ciclos de máquina para serem executadas, enquanto que instruções mais simples (que continuavam existindo) eram resolvidas mais rapidamente, o que dificultava a implementação otimizada de pipelines.

6.2 RISC

Estudos do comportamento de execução de programas de linguagens de alto nível foram a base para o surgimento de uma nova arquitetura de processadores: o Reduced Instruction Set Computer (RISC). A predominância de atribuições sugeria que a movimentação de dados simples deveria ser otimizada. Existiam também muitas instruções condicionais e loops, sugerindo que o mecanismo de controle sequencial deveria ser otimizado. E, finalmente, o padrão de referências de operandos sugeria a possibilidade de melhorar o desempenho mantendo um moderado número de operandos em registradores.

As máquinas RISC apareceram no início dos anos 80 em laboratórios de empresas (p.e. IBM), como o IBM 801, e em instituições de pesquisa: RISC I e RISC II na Universidade da Califórnia/Berkeley e MIPS na Universidade de Stanford. Em 1985, todas as três estavam operacionais. Estes projetos não tinham a intenção de revolucionar o projeto de processadores. Principalmente os projetos de Berkeley e Stanford, tinham como foco o projeto de máquinas eficientes, mas simples o suficiente para poderem ser implementados em ambiente universitário.

As características principais de uma máquina RISC são:

- Conjunto de instruções limitado, com um formato fixo e simples (não ultrapassavam o tamanho da palavra);
- Poucos modos de endereçamento;
- As operações eram registrador-para-registrador (e não memória-para-memória) com somente instruções LOAD e STORE para acesso à memória;
- Grande número de registradores ou a utilização de compiladores que otimizam a utilização de registradores;
- Ênfase na otimização da pipeline de instruções.

A arquitetura RISC permitiu a implementação de circuitos operando a uma frequência elevada, devido ao fato de necessitar de circuitos muito mais simples comparados aos processadores CISC. Tudo isso, devido ao fato de implementar apenas um conjunto limitado e otimizado de instruções. Os circuitos de cada estágio de um pipeline RISC são muito mais simples, o que para o mundo da microeletrônica significa menos atrasos e mais velocidade. É claro que alta frequência de operação não é sinônimo de processamento veloz (é um grande apelo de mercado), mas a arquitetura RISC abriu novas fronteiras para o projeto de microprocessadores.

O objetivo das máquinas RISC era o de aumentar a eficiência da interface entre o compilador e o hardware e não, diminuir o "gap" entre o usuário e o hardware. Este último tem sido o objetivo das máquinas consideradas CISC, como os microprocessadores de desktops e alguns servidores.

A compilação em processadores RISC é simples e o código gerado é eficiente, até mesmo em função da própria filosofia de se ter um conjunto reduzido de instruções. Ao lado disso, com o avanço tecnológico da integração VLSI ("Very Large Scale Integration"), também tornou-se possível realizar o máximo de operações no interior do circuito integrado, aumentando a velocidade das operações.

Pelo fato das instruções serem simples e pouco numerosas, várias transferências CPU/Memória e vice-versa são necessárias. Este problema vem sendo resolvido com a técnica de "cache", ou seja, ler um grupo de instruções de uma única vez, enfileirá-las e executá-las em seguida.

Um outro aspecto diz respeito aos numerosos registradores internos a processadores RISC. Pode-se armazenar nesses registradores uma grande quantidade de dados temporários, no interior da pastilha, sem que sejam necessários vários acessos à memória principal.

A técnica de "pipeline" em vários níveis também é utilizada para aumentar ainda mais a taxa de execução de instruções. Algumas máquinas RISC executam uma instrução por ciclo de relógio, isto para implementar um pipeline eficiente. As máquinas clássicas utilizam vários ciclos de máquina, contendo vários ciclos de relógio para a execução de uma única instrução. Técnicas para tentar manter o processador ocupado o máximo do tempo foram desenvolvidas, como por exemplo o desvio atrasado (delayed branch) e os circuitos de predição de desvios.

As instruções RISC, em geral, são de mesmo comprimento para aproveitar de maneira eficiente a estrutura pipeline, uma vez que comprimentos diferentes levariam ao aumento do número de acessos à memória para busca de instruções. Outro problema, é que o formato da instrução somente seria "descoberto" após a identificação da mesma, podendo causar problemas como "bolhas" no pipeline.

6.3 ILP – Instruction Level Parallelism

Os processadores haviam evoluído de processadores sequenciais para processadores "pipelined". E com o desenvolvimento da arquitetura RISC, surgiram técnicas para o projeto de "pipelines" balanceados e eficientes. O próximo passo foi, então, disponibilizar um maior número de recursos, ou seja, implementar mais de uma unidade de execução. O paradigma ILP tenta executar mais de uma instrução por ciclo de relógio, extraíndo da sequência de instruções aquelas que são independentes, e que possam ser enviadas para as unidades de execução ao mesmo tempo.

O processador examina a sequência de instruções, verifica a existência de dependência de dados ou de controle, e pode vir até a executar as instruções fora de ordem.

6.3.1 Superescalares

O projeto superescalar apareceu logo depois dos projetos de arquitetura RISC. Embora as técnicas de arquitetura superescalar possam ser aplicadas mais diretamente a uma arquitetura RISC com conjunto simplificado de instruções, a abordagem superescalar também pode ser usada em arquiteturas CISC.

Inicialmente surgiram as implementações superescalares, tendo em geral pares de pipelines, permitindo que duas instruções pudessem ser executadas simultaneamente. Em geral, em caso de

dependência de dados, estes processadores acabavam parando um dos pipelines, deixando fluir apenas uma instrução. Era comum também a implementação de um pipeline complexo, que poderia executar qualquer instrução, e um outro mais simples, que executava apenas instruções simples. Como representantes podemos citar o MC68060 desenvolvido pela Motorola e o Pentium desenvolvido pela Intel.

Logo em seguida surgiram as implementações de processadores com execução dinâmica fora de ordem. Nestas implementações os processadores ganharam mais de cinco unidades de execução, e passou a ser comum o despacho de mais de três instruções por ciclo de relógio.

6.3.2 VLIW - EPIC

Surgiram então os processadores que codificam mais de uma operação por instrução (VLIW – Very Long Instruction Word ou EPIC – Explicit Parallel Instruction Computer (Intel + HP)). Neste caso, a verificação de dependência de dados e/ou de controle fica a cargo do compilador.

Um exemplo é a arquitetura IA-64 (Itanium). A arquitetura IA-64 usa instruções de tamanho fixo (40 bits) e elas são executadas em um único período de relógio, ou seja, são semelhantes às instruções RISC: comprometidas com o desempenho. As instruções IA-64 são arrumadas em pacotes de 128 bits, cada pacote com 3 instruções. Isso é o que se chama VLIW, mas a Intel prefere usar o termo EPIC ("Explicitly Parallel Instruction Computing"), pois nesse pacote, além das três instruções, existem bits para indicar quais podem ser executadas em paralelo. A CPU não mais precisa ficar analisando (durante a execução) o fluxo de instruções tentando descobrir o paralelismo intrínseco do problema, pois ele é explicitado durante a compilação. Cada instrução tem o tamanho fixo de 40 bits. A sequência das instruções é fornecida pelo compilador e não necessariamente coincide com a ordem original do programa. O campo de 8 bits, denominado "template", indica quais instruções desse pacote, e dos próximos, podem ser executadas em paralelo.

Nesta fase foram implementadas as mais brilhantes idéias no campo dos processadores. Eles passaram a ser capazes de executar algumas instruções simultaneamente, resolveram os problemas de dependência de dados através da renomeação de registradores e de dependência de controle através da predição de saltos.

Começou também neste momento, desaparecer a distinção entre processadores CISC e processadores RISC. Atualmente, as técnicas utilizadas para o projeto são comuns tanto para os ditos puramente CISC como para os puramente RISC. O conjunto de instruções CISC, de tamanho variado, é codificado internamente para um formato fixo, e então é utilizadas técnicas de projeto RISC. Por outro lado, os processadores RISC ganharam um maior número de instruções, diminuindo a distância entre as duas arquiteturas.

6.4 CISC versus RISC versus VLIW

Conta a história que as máquinas RISC ("Reduced Instruction Set Computer") surgiram quando os projetistas substituíram as instruções complexas por um conjunto mais simples, mais veloz e de controle mais fácil. Com isso, o chip ficou menos sobrecarregado, sobrando mais área para ser usada em outras unidades funcionais.

A história real é um pouco diferente! As máquinas RISC nasceram da necessidade de simplificar-se a CPU. A idéia foi trocar complexidade da CPU por otimização durante a compilação. Usou-se então um conjunto de instruções pequeno mas veloz, que pudesse ser executado em paralelo e

ainda que fosse de fácil controle. Um grande conjunto de registradores substituiu os complexos modos de endereçamento. Com a simplificação da unidade de controle, uma maior área do CI pode ser empenhada diretamente no processamento. O trabalho de arrumar as instruções, de forma a espremer o máximo desempenho da CPU, foi transferido para o compilador. Em outras palavras, boa parte da complexidade da CPU foi transferida para o compilador. Assim o termo RISC não é muito feliz pois ele faz referência apenas ao conjunto de instruções e ignora o importante papel do compilador em gerar programas otimizados para essas máquinas.

A família x86 começou com a CPU 8086, que é uma máquina CISC. Com a demanda por desempenho, os projetistas da Intel espremeram todo o desempenho passível de ser obtida com essa arquitetura. A evolução natural seria para uma arquitetura RISC, mas isso comprometeria seriamente a compatibilidade com o passado. A solução encontrada foi a de projetar uma máquina híbrida: uma casca CISC com um coração RISC. Assim são os Pentium Pro e Pentium II que traduzem as complexas instruções x86 em uma ou mais micro-operações que são executadas por uma máquina RISC. Essas micro-operações são de mais fácil controle e podem ser executadas em paralelo.

Porém uma grande parte da CPU ainda é dedicada ao controle dessas instruções, à predição dos desvios e à tradução. As atuais CPUs possuem milhões de transistores, mas, desse total, apenas uma pequena parte é usada diretamente no processamento, pois a grande maioria é consumida no controle das instruções para possibilitar a execução fora de ordem e na predição dos desvios, ou seja, as atuais arquiteturas RISC/CISC estão complexas novamente e, por isso, é hora de aplicar o método: vamos simplificar a CPU e passar o trabalho de otimização para o compilador. Essa é a filosofia das máquinas com arquitetura VLIW – EPIC.

6.5 Paralelas

Tradicionalmente, o computador tem sido visto como uma máquina seqüencial. A maioria das linguagens de programação requer que o programador especifique um algoritmo como uma seqüência de instruções. Os processadores executam programas por meio da execução seqüencial de instruções de máquina. Cada instrução é executada como uma seqüência de operações (busca de instrução, busca de operandos, execução da operação, armazenamento dos resultados).

Essa visão do computador nunca foi totalmente verdadeira. No nível de microoperações, vários sinais de controle são gerados ao mesmo tempo. A técnica de pipeline de instruções tem sido usada a muito tempo, estendendo esta sobreposição pelo menos para as operações de busca e execução de instruções. Esses são dois exemplos de execução de funções em paralelo. Esta abordagem é levada mais adiante em uma organização superescalar, que explora o paralelismo em nível de instrução. Nas máquinas superescalares, existem diversas unidades de execução em um mesmo processador, que podem assim executar várias instruções de um mesmo programa em paralelo.

A medida que a tecnologia evoluiu e o custo do hardware do computador tornou-se mais baixo, os projetistas de computadores têm buscado outras oportunidades de exploração do paralelismo, usualmente para melhorar o desempenho e, em alguns casos, para aumentar a disponibilidade do sistema.

Podem ser encontradas diversas formas de organização paralela. A figura 27 ilustra os principais tipos.

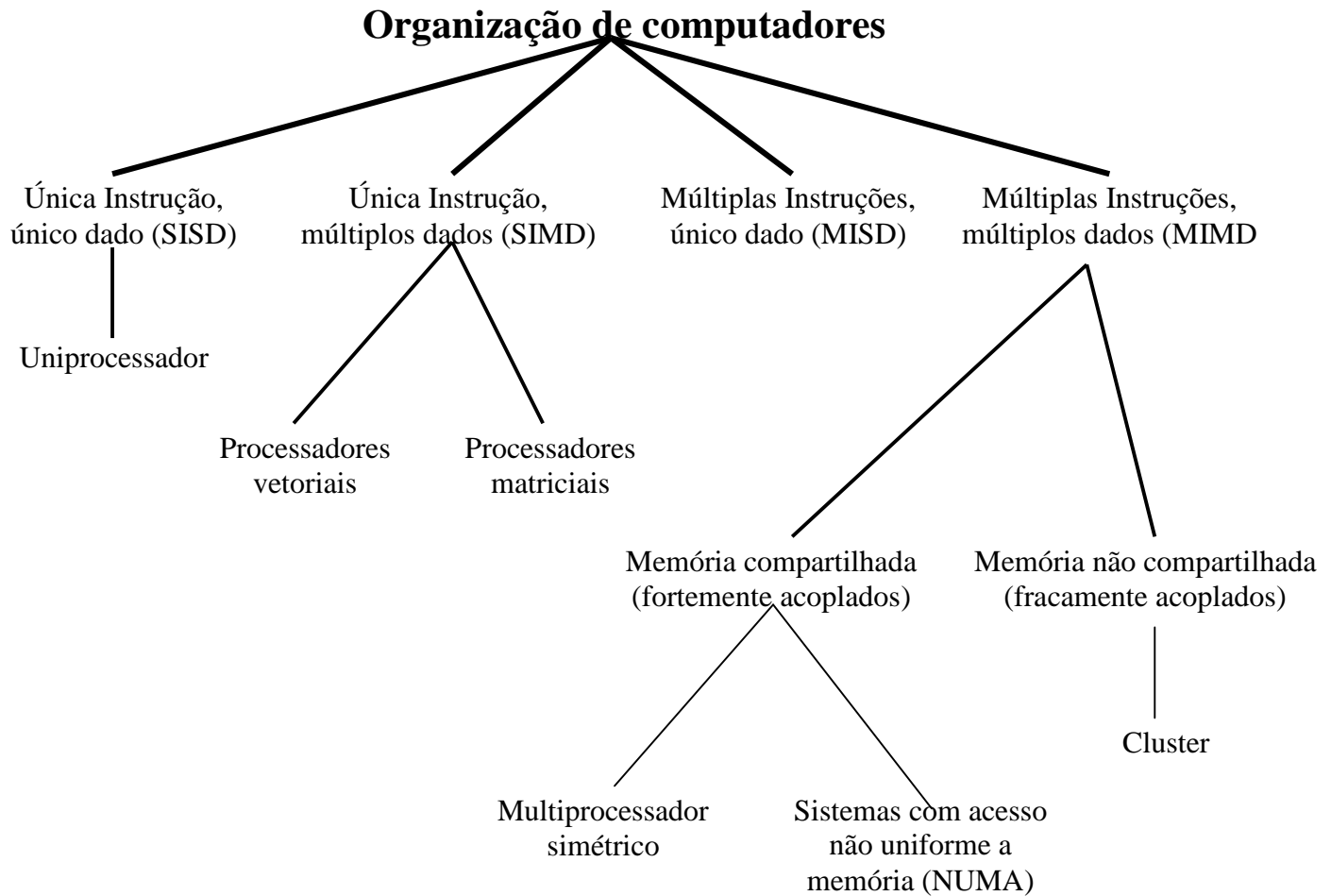


Figura 27 – Organização de computadores

A categoria SISD, representa as máquinas com apenas um processador. A categoria MISD, não possui, pelo menos por enquanto, representante devido ao seu princípio de funcionamento. As máquinas paralelas situam-se nas nas categorias SIMD e MIMD, sendo que 90% concentram-se nas máquinas MIMD. As máquinas SIMD são conhecidas como máquinas vetoriais ou paralelas.

As máquinas MIMD possuem uma divisão em relação ao espaço de endereçamento de memória. Multiprocessadores possuem um único espaço para endereçamento da memória, desta forma possibilitando regiões de dados compartilhados. São máquinas geralmente de custo elevado e limitado número de processadores devido a contenção de acesso ao barramento de memória (SMP). Os multicomputadores são máquinas MIMD com vários espaços de endereçamento de memória, desta forma não possibilitando memória compartilhada. Estas máquinas possuem um custo reduzido, uma vez que podem ser empregados componentes de computadores comerciais na sua construção e não possuem um limite rígido quanto ao número de computadores.

Bibliografia

- MONTEIRO, Mário A. **Introdução à organização de computadores**. Rio de Janeiro : LTC , 2001. 397p.
- BRUSSO, Marcos José. **Arquitetura e Organização de Computadores I: notas de aula**. Disponível por WWW em <<http://vitoria.upf.tc.br/~brusso/arq1/download.html>>
- Weber, Raul Fernando. **Arquitetura de computadores pessoais**. Porto Alegre: Sagra Luzzatto , 2001. 271 p.
- Weber, Raul Fernando. **Fundamentos de arquitetura de computadores**. Porto Alegre : Sagra Luzzatto , 2000. 262 p.
- MEYER, Marilyn *et. al.* **Nosso futuro e o computador**. Porto Alegre : Bookman , 2000. 599 p.
- HENNESSY, John L., PATTERSON, David A.. **Computer organization and design: the hardware, software interface**. California : Morgan Kaufmann , 1998. 758 p.