ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

CAPÍTULO 1 - POR ONDE COMEÇAR PARA ENTENDER COMO O COMPUTADOR FUNCIONA?

Fernando Cortez Sica

INICIAR

Introdução

Neste capítulo, vamos estudar os conceitos mais fundamentais, relacionados ao funcionamento do computador, e compreender como ocorreu a evolução técnica até chegar os computadores que conhecemos atualmente. Estes conceitos são importantes para que se conheça, posteriormente, elementos mais aprofundados a respeito do computador como um todo e a sua interoperabilidade com o Sistema Operacional.

Para começar esse percurso, você vai se deparar com a diferenciação entre dois níveis de abstrações, que podemos associar aos sistemas computacionais: a arquitetura e a organização. Para entender essa diferenciação vamos nos perguntar: o que é família de processadores?

Também veremos um histórico dos principais computadores projetados até os dias atuais, correlacionando-os com a própria evolução da eletrônica. E, para concluir, apresentaremos os conceitos inerentes ao *pipeline* e à superescalaridade, para que você possa refletir sobre as seguintes questões: o que influencia na performance de processamento de um computador? Como poderemos aumentar o poder de processamento de um sistema computacional?

A partir dos conceitos aqui apresentados, você terá condições de se situar na trajetória dos sistemas computacionais e aplicar seu conhecimento no seu cotidiano, seja na tomada de decisões, ou na aplicação no desenvolvimento de aplicações.

Vamos acompanhar com atenção? Bons estudos!

1.1 Visão geral da arquitetura e organização de computadores

Muito se fala em família de processadores, principalmente quando estamos escolhendo um Sistema Operacional ou um aplicativo a ser instalado no computador. Mas, o que vem a ser uma família, dentro do mundo dos computadores? Para que se possa responder a essa pergunta, devemos, antes, responder a outra pergunta: o que vem a ser arquitetura e organização de computadores?

Esse tópico ajudará você a encontrar a resposta, ao compreender os detalhes mais evidentes para um programador, em relação à construção do processador, e aqueles que não são muito importantes, pelo menos, a princípio. Apesar de menos importantes, esses detalhes devem ser de conhecimento do programador, em algumas situações, para que se possa alcançar um nível maior de eficiência computacional, diminuindo-se o tempo de processamento pela otimização de seu código para um processador específico.

Então, a partir de agora, vamos entender como se organizam os computadores, por meio de sua arquitetura.

1.1.1 Arquitetura e organização de computadores

Vamos fazer uma analogia da noção de arquitetura e organização de computadores com a implementação de uma função no âmbito de programação. Uma função tem o seu protótipo (tipo de retorno e a lista de parâmetros) e a parte da implementação com o seu respectivo código (instruções). Quando uma biblioteca de funções é utilizada, o programador precisa apenas se preocupar em como passar os parâmetros e como coletar o seu resultado. Dessa forma, não é necessário saber como a função foi implementada.

Assim ocorre também nos computadores: há uma separação das características e estruturas importantes, para que se possa construir aplicativos, compiladores, sistemas operacionais e todos os demais *softwares* que rodarão sobre um processador específico. Sabendo disso, podemos exemplificar como elementos da arquitetura de computadores (STALLINGS, 2010):

- **conjunto de instruções**: conhecer quais instruções exportadas pelo processador são indispensáveis para, por exemplo, os projetistas de compiladores, para que seja possível realizar a tradução das instruções escritas em uma linguagem de alto ou médio nível (por exemplo, C/C++) para a linguagem de máquina (*assembly*);
- **mecanismos de Entrada e Saída (E/S) (I/O – *Input*/ *Output*)**: o programador deve saber como proceder para que o seu programa possa interagir com dispositivos externos, como escrever um *driver* que leia e envie informações via USB (*Universal Serial Bus*);
- **técnicas de endereçamento**: neste caso, podemos citar como deve ser implementada a parte do Sistema Operacional que manipula o gerenciamento da memória;
- **tamanho dos tipos de dados**: refere-se ao tamanho do espaço de memória (quantidade de *bytes*) necessário para armazenar uma variável de um certo tipo (*char*, *int*, *float*, por exemplo). Neste caso, o programador deverá ter conhecimento do tamanho e, consequentemente, dos tipos de dados disponíveis para escolher o mais apropriado para o seu programa.

Por outro lado, o termo ‘organização de computadores’ refere-se aos aspectos internos de projeto e implementação de um processador, que não são fundamentais ao programador. Dentre os itens relacionados à organização, podemos citar:

- **sinais internos de controle**: podem ser entendidos como pulsos elétricos, ativados pela execução de uma instrução, responsáveis pela ativação e sincronização dos módulos de *hardware*;
- **forma de implementação das instruções**: por exemplo, uma subtração é realizada com um circuito específico de subtração, ou é implementada com o circuito de adição, somando-se o inverso da seguinte forma: a = b – c → a = b + (-b)?;
- **hierarquia dos módulos funcionais**: como os sub-módulos de *hardware* estão organizados dentro do processador e como são as suas ligações elétricas?

Agora, você pode estar se perguntando: qual a relação entre arquitetura e organização de computadores com o termo ‘família de processadores’? Então, vamos entender isso melhor.

Uma família relaciona processadores que possuem a mesma arquitetura. Então, pode-se afirmar, por exemplo, que todos os processadores da família x86 possuem o mesmo conjunto de instruções, mesma forma de endereçamento e mesmo espaço de memória para os tipos disponíveis de dados. É por esse motivo que você consegue executar o mesmo programa em um computador com um processador da Intel e também em um outro com um processador da AMD, sem a necessidade de se realizar qualquer alteração na programação ou recompilar o código. Isso porque, neste exemplo, ambos os processadores pertencem à família x86.

Vamos, então, entender como o computador é estruturado e suas funcionalidades.

1.1.2 Estrutura e função

Ao nos aprofundar na organização de computadores, podemos pensar o computador como sendo a interconexão entre os módulos internos de *hardware*. Esses módulos poderão, ainda, ser subdivididos. A cada aprofundamento em suas camadas, teremos módulos mais simples e especializados. O nome dessa estruturação física em módulos funcionais é **estrutura**. Uma possível abstração de estrutura computacional é exibida na figura a seguir.

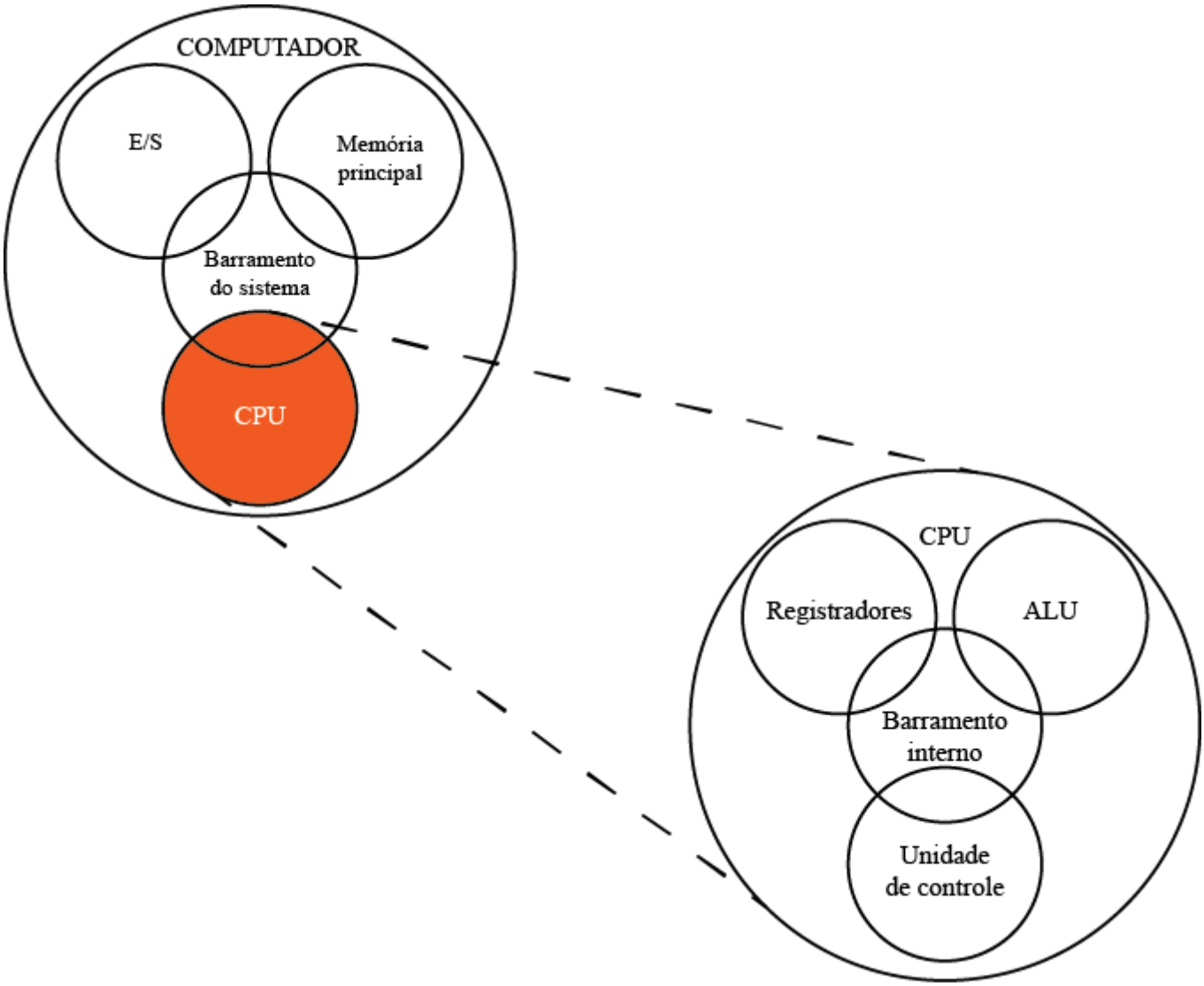


Figura 1 - Estrutura de um computador em sua abstração de alto nível e a decomposição do processador em submódulos pela aplicação da metodologia top-down. Fonte: STALLINGS, 2010, p. 10.

A figura acima utiliza uma abordagem denominada como "*top-down*" para descrever os módulos internos. Ao contrário da "*bottom-up*", na metodologia "top-down", parte-se dos módulos mais gerais e superiores para que estes sejam decompostos em submódulos, até que se alcancem módulos que não precisem ser subdivididos, devido à sua simplicidade e funcionalidade bem determinada.

Observe novamente a figura anterior e perceba que, no primeiro nível de abstração, o computador é formado pelos seguintes módulos primários (STALLINGS, 2010):

- **CPU**: módulo responsável pelo processamento propriamente dito. Cada instrução do programa é carregada para a CPU, a fim de ser decodificada e executada;
- **memória principal**: na memória principal são alocados espaços para que possam ser carregados os processos (programas em execução). Sendo assim, na memória principal, encontramos instruções, dados e informações de controle de processamento;
- **E/S (Entrada/ Saída)**: entidades responsáveis por realizar a interação com o "mundo externo" para coletar ou externar informações. São exemplos de módulos de E/S: controladores de teclado, de vídeo, HD (*Hard Disk*), etc.;
- **Barramento do Sistema**: elemento responsável por realizar a interconexão dos demais módulos. Os módulos são distribuídos seguindo uma organização física denominada como topologia.

Por sua vez, a CPU é decomposta pelos submódulos:

- **registradores**: representam a memória interna do processador, para que sejam carregadas as instruções e as informações a serem manipuladas;
- **ALU** (*Arithmetic Logic Unit*, em inglês, traduzido como Unidade Lógica e Aritmética, ULA): tem a responsabilidade de executar operações aritméticas (como uma adição) e operações lógicas (como comparação de magnitude);
- **Barramento Interno**: módulo cujo objetivo é permitir a interconexão entre os demais módulos, permitindo-se um fluxo de informações entre eles;
- **Unidade de Controle**: gerencia o fluxo de informações internamente ao processador, sendo, também, responsável pela gerenciamento do processamento em si.

Um outro conceito que podemos mencionar é denominado ‘função’. Como o próprio nome indica, **função** representa as funcionalidades associadas a cada módulo que integra o computador. Essencialmente, as funções do computador podem ser agrupadas em quatro grupos básicos, conforme indica a figura a seguir.

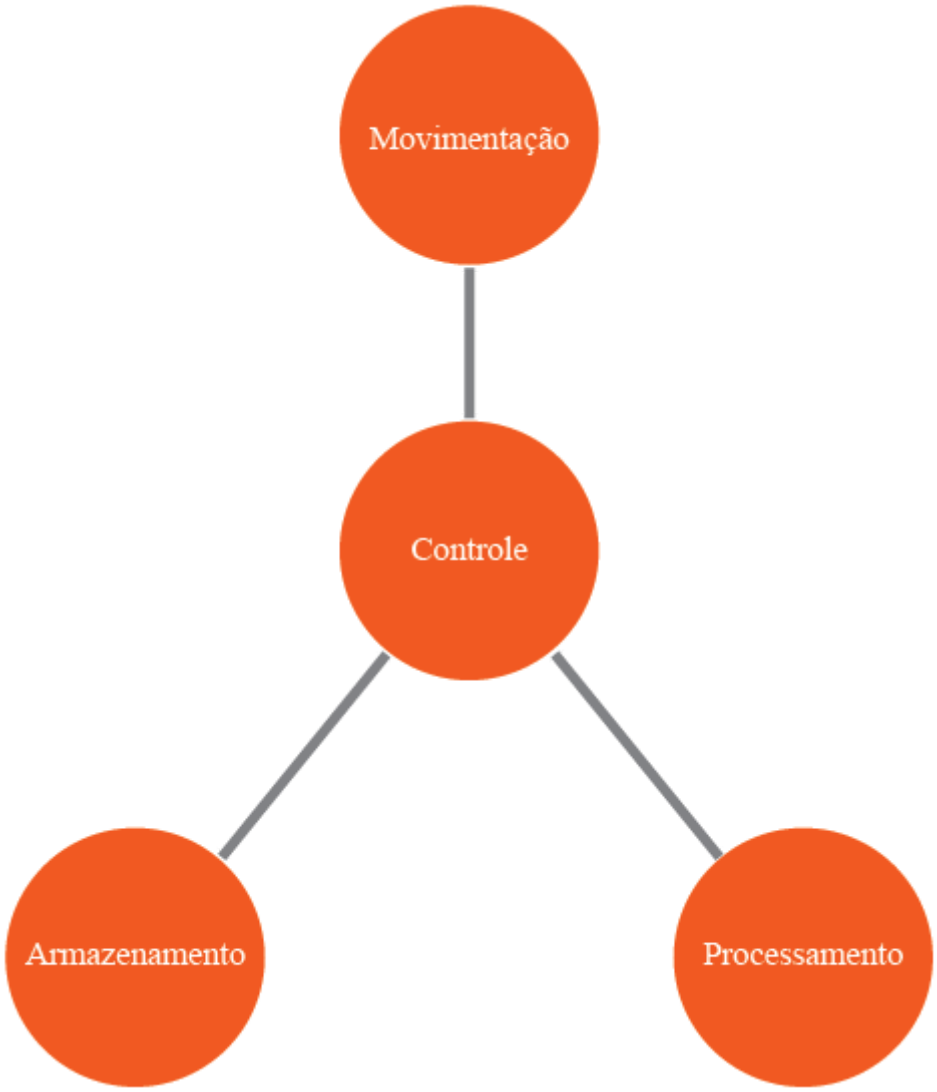


Figura 2 - Ao verificar as funções básicas de um computador, percebemos que o módulo responsável pela funcionalidade ‘Controle’ centraliza todas as operações. Fonte: STALLINGS, 2010, p. 4.

Na figura acima, notamos que a funcionalidade básica de controle gerencia as demais funcionalidades, estabelecendo a interface entre os módulos ativados para a execução das atividades correspondentes. Além da funcionalidade de controle, encontramos a **funcionalidade de processamento**, cuja missão é executar as instruções que manipularão as informações coletadas do mundo externo (por meio dos módulos de E/S como, por exemplo, o ‘disco rígido’ ou, ainda, por uma rede de comunicação de dados).

O processo de coleta ou de envio das informações para o mundo externo representa a **funcionalidade de movimentação**. Por fim, temos a **funcionalidade de armazenamento**, cujo objetivo consiste em guardar, em módulos de memória, as informações oriundas do mundo externo ou derivadas do processamento.

Agora, você pode estar se perguntando: mas, como o computador, em sua funcionalidade de controle, sabe quais módulos ativar?

O módulo de controle realiza sua tarefa em função das instruções obtidas do programa sob execução. Cada instrução é decodificada para que o módulo de controle saiba qual funcionalidade deve ser atingida naquele momento e, consequentemente, quais módulos deverão ser ativados, alocando os recursos necessários.

Agora, que você sabe o que é arquitetura e organização, estrutura e função, vamos dar uma olhada na evolução dos computadores? Podemos observar que a arquitetura e organização dos computadores estão associadas com a própria evolução técnica que os computadores tiveram. Você pode aproveitar a deixa e refletir sobre estrutura e função, correlacionando-as com a evolução dos computadores.

1.2 Histórico e evolução dos computadores

Ao observar os computadores atuais, podemos dizer que eles evoluíram junto com a eletrônica, desde o tempo das válvulas termiônicas até os atuais circuitos integrados (*chips*).

Com a miniaturização do componentes, foi possível construir circuitos mais complexos em espaços muito menores do que se usava anteriormente. Com os componentes menores, a construção de memórias com mais capacidade e mais rápidas, também se tornou possível, o que garante informações processadas de forma mais eficiente. As novas tecnologias também permitem o emprego de frequências maiores de operação, dissipando menos potência.

Além dos fatores físicos, novas técnicas de processamento, como o emprego de *pipeline*, superescalaridade e execução fora de ordem, vem permitindo o aumento da performance de processamento.

Então, vamos ver como se sucederam as gerações dos computadores, com um breve histórico da evolução dos computadores, para que você possa tirar suas próprias conclusões sobre arquitetura e organização, e do quanto evoluímos nos aspectos tecnológicos.

VOCÊ O CONHECE?

A matemática e escritora inglesa Ada Lovelace foi a primeira pessoa a escrever um algoritmo para computador. Em sua homenagem, foi atribuído o seu nome à uma linguagem de programação. A linguagem *Ada* foi criada em 1982, teve como base o *Cobol* e o *Basic* e foi referência para a criação da linguagem de programação *Ruby* (PORTAL EBC). Saiba mais na matéria: <http://www.ebc.com.br/tecnologia/2015/03/conheca-historia-da-ada-lovelace-primeira-programadora-do-mundo (http://www.ebc.com.br/tecnologia/2015/03/conheca-historia-da-ada-lovelace-primeira-programadora-do-mundo)>.

A partir de agora, vamos observar que a evolução das gerações dos computadores terá inúmeras consequências, não somente no impacto positivo do poderio de processamento, quanto também nas funcionalidades exportadas pelos processadores. Sob o ponto de vista da programação, a evolução pode ser percebida pela forma de interação e programação, passando da programação pela atuação direta de chaves e fios, até as linguagens atuais de quarta geração.

1.2.1 Primeira geração

Essa geração foi marcada pelo emprego das válvulas termiônicas em sua construção. O pioneiro dos computadores foi o ENIAC (*Electronic Numerical Integrator and Computer*), construído entre os anos de 1943 e 1946 pela Universidade da Pensilvânia pelo físico John Mauchly e pelo engenheiro John Eckert.

O ENIAC foi construído com o intuito de realizar cálculos balísticos para o BRL (*Ballistic Research Laboratory*, laboratório do exército dos Estados Unidos) para que fosse utilizado durante a Segunda Guerra Mundial. Apesar de não ter sido finalizado a tempo de entrar em operação durante a guerra, teve, como primeira missão, cálculos para a construção da bomba de hidrogênio.

O ENIAC operava segundo um sistema decimal de numeração (não era binário) e tinha a capacidade de processar 5000 adições por segundo. Porém, para que fossem usadas as suas 30 toneladas, em suas 18 mil válvulas, era necessária a configuração contínua de chaves e de cabos (conectando-os e desconectando-os) para denotar as operações e as informações a serem processadas. Tudo isso era feito manualmente por seis programadoras do exército, especialistas em cálculos balísticos.

O ENIAC tem um merecido destaque, pois foi o primeiro computador eletrônico construído para uso genérico. Porém, em função da dificuldade de sua manipulação, pode-se dizer que, em termos de estruturação, o destaque da primeira geração fica a cargo do EDVAC (*Electronic Discrete Variable Automatic Computer*) e, um pouco depois, do IAS (*Institute for Advanced Studies*, da Universidade de Princeton), projetado pelo matemático John von Neumann, nos anos de 1946 e 1952, respectivamente.

A ideia de von Neumann consistia no emprego de memória para o armazenamento do programa. Isso tornava mais fácil a interação com o computador para a programação. A figura a seguir, ilustra a concepção proposta por von Neumann no IAS.

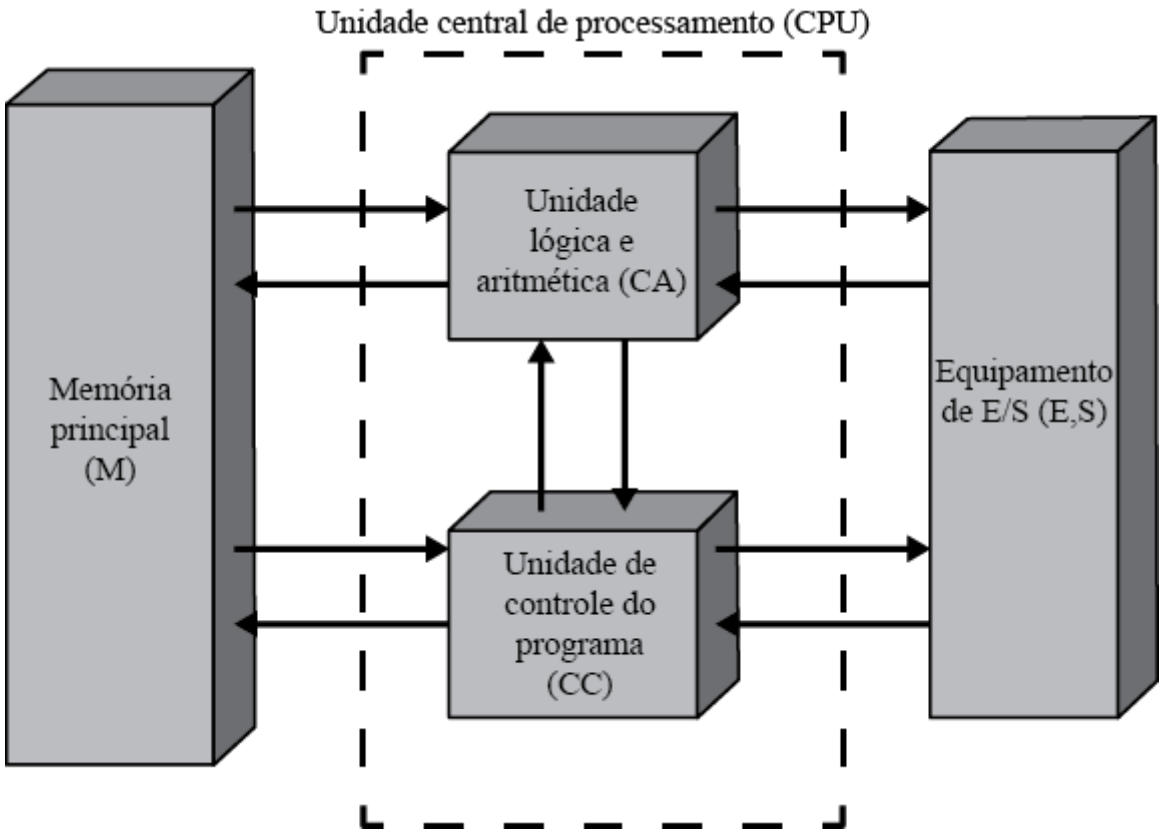


Figura 3 - Estruturação do IAS com os elementos encontrados nos computadores atuais: programa armazenado em memória, ULA, unidade de controle e E/S. Fonte: STALLINGS, 2010, p. 14.

Na figura acima podemos encontrar, como módulos básicos:

- **Memória Principal**: memória utilizada para armazenar, instruções e dados a serem processados, ou resultantes do processamento. As instruções são formadas por palavras binárias, interpretadas diretamente pelo *hardware*;
- **Unidade Lógica-Aritmética (ULA)**: módulo responsável por realizar as operações aritméticas e lógicas do computador;
- **Unidade de Controle (UC)**: a partir das instruções coletadas da memória, a UC tem a missão de decodificá-las e providenciar a sua execução;
- **equipamento de E/S**: interface entre o computador e o mundo externo.

Podemos observar que alguns conceitos e estruturas desenvolvidos no IAS ainda tem significado para os computadores atuais. Inicialmente, foi usado o termo ‘palavra’ para designar a unidade básica de armazenamento e transferência (no

caso, a palavra era de 40 *bits*). Também foi utilizado o termo 'registrador' para denotar estruturas de armazenamento internamente à ULA ou à unidade de controle (STALLINGS, 2010):

- **Registrador de *Buffer* de Memória** (MBR, *Memory Buffer Register*): usado para armazenar ou receber uma palavra da memória ou da unidade de E/S;
- **Registrador de Endereço de Memória** (MAR, *Memory Address Register*): armazena o endereço de memória que contém a informação a ser enviada ao MBR ou o endereço de memória que receberá a informação presente no MBR;
- **Registrador de Instrução** (IR, *Instruction Register*): contém a palavra referente à instrução em execução;
- **Registrador de *Buffer* de Instrução** (IBR, *Instruction Buffer Register*): mantém a próxima instrução a ser executada;
- **Contador de Programa** (PC, *Program Counter*): indica o endereço de memória da próxima instrução a ser executada;
- **Acumulador e Quociente-Multiplicador** (AC e MQ, respectivamente): armazena operandos e resultados provenientes da ULA.

Por fim, o conjunto do IAS tinha 21 instruções, agrupadas de acordo com as suas funcionalidades: transferência de dados (entre memória e registrador ou registrador-registrador), desvio incondicional, desvio condicional, aritméticas e modificação de endereço (para permitir um endereçamento flexível).

VOCÊ O CONHECE?

O matemático Alan Turing é considerado como o pai da computação. Desenvolveu várias teorias, que culminaram na criação da Inteligência Artificial e que contribuíram na criação do computador como conhecemos atualmente. Quanto à criação do computador, ele teve, simultaneamente a von Neumann, a mesma ideia do programa armazenado em memória para que pudesse ser processado (NUNES, 2012).

Após conhecer essas informações sobre o IAS, podemos comparar quais os elementos presentes no IAS que são encontrados nos computadores atuais. Certamente, é uma reflexão interessante de se fazer.

Finalizando a primeira geração, podemos destacar os momentos importantes (STALLINGS, 2010):

- 1947 – fundação da Eckert-Mauchly Computer: UNIVAC;
- 1952 – IBM: implementação do computador modelo 700;
- 1953 – IBM: implementação do computador modelo 701;
- 1955 – IBM: implementação do computador modelo 702.

A seguir, vamos acompanhar um breve histórico da segunda geração de computadores.

1.2.2 Segunda geração

A segunda geração teve seu início a partir da substituição de válvulas por transistores, que, além de serem menores, consumiam menos energia, o que ampliou a eficiência computacional e possibilitou desenvolver memórias com maior capacidade de armazenamento (STALLINGS, 2010). Essa substituição foi possível em um contexto no qual vários pesquisadores buscavam inovações no campo das telecomunicações, até que, em 1948, o físico William Shockley, desenvolveu o transistor de junção, no laboratório Bell Labs.

O transistor é feito com materiais denominados semicondutores, formados pela adição de impurezas ao silício, por exemplo. Essa adição permite que sejam criados semicondutores do tipo P (positivo) e do tipo N (negativo). Essa diferença de carga permite o controle do fluxo de elétrons – base dos circuitos eletrônicos.

Como unidades de memória, mencionamos o aparecimento dos núcleos de ferrite, fitas e tambores magnéticos como dispositivos de memória.

Além das alterações físicas (*hardware*), essa geração foi marcada com o aparecimento de linguagens de programação de alto nível, como Fortran e Cobol, e pelas aplicações administrativas, gerenciais e comerciais.

Em relação ao gerenciamento dos computadores, essa geração também foi marcada pelo aparecimento dos primeiros sistemas operacionais com a função de sequenciar automaticamente as tarefas, apenas, por esse motivo, eles eram chamados de “monitor”. Assim, desta segunda geração, podemos destacar os seguintes momentos (STALLINGS 2010):

- 1957 – fundação da companhia Digital Equipment Corporation (DEC): PDP-1 (início da geração de minicomputadores);
- 1960 – IBM: modelo 7090;
- 1962 – IBM: modelo 7094 I;
- 1964 – IBM: modelo 7094 II.

Dos modelos implementados pela IBM, o 7094 merece destaque, pois introduziu um processador dedicado às operações de E/S (Entrada/ Saída), como veremos na figura a seguir. Nelas, o processador principal envia um sinal de controle ao processador de E/S (canal de dados), que fica responsável por realizar a transferência das informações. Após o término da operação, o canal de dados emite um sinal ao processador principal para que ele processe os dados recém transferidos. Dessa forma, ocorre uma liberação de processamento do processador principal. Esta técnica era conhecida como “*buffering*”.

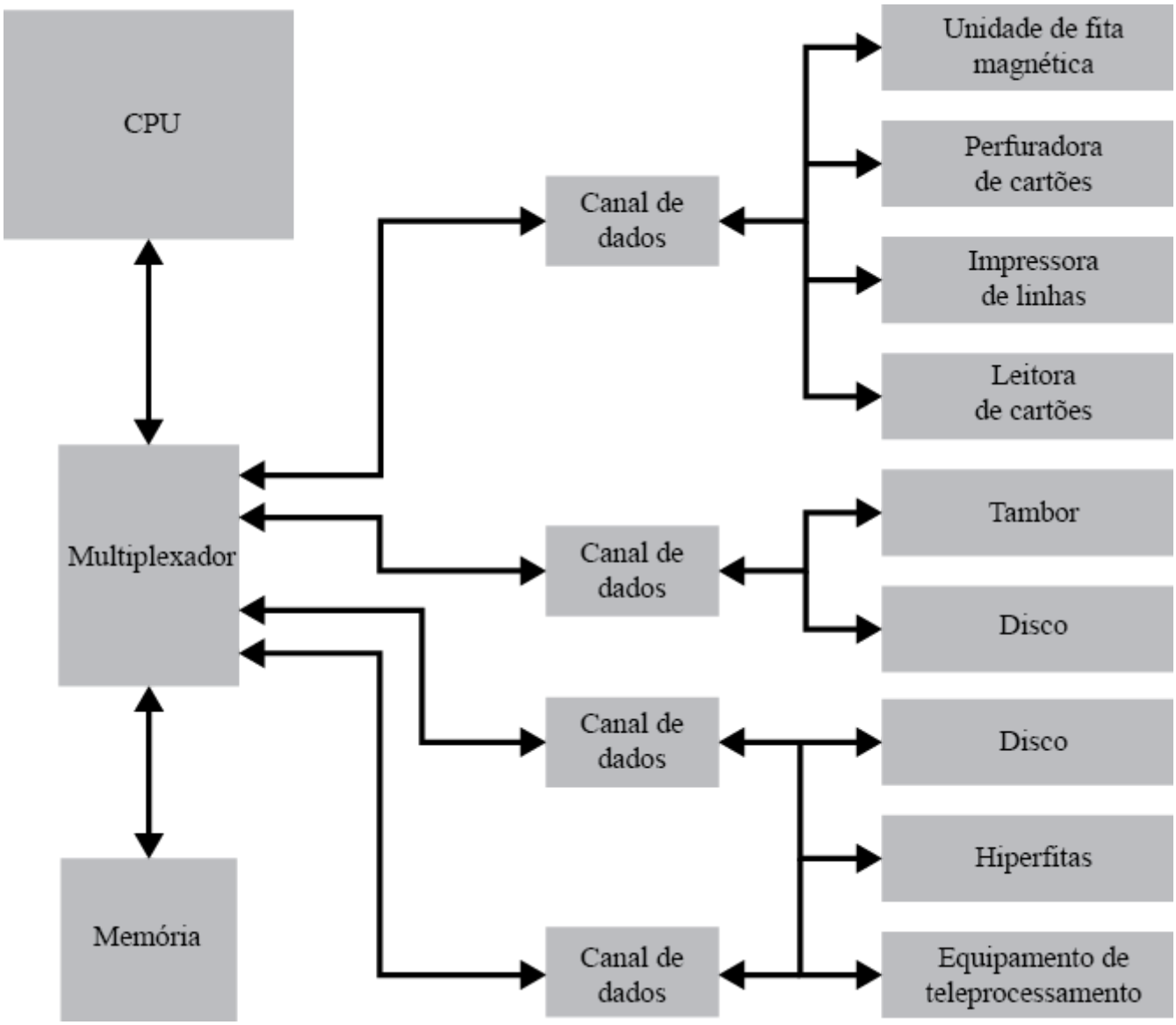


Figura 4 - Estruturação do IBM, modelo 7094, no qual se pode notar a presença de vários módulos de E/S compartilhando o acesso à CPU e à memória. Fonte: STALLINGS, 2010, p. 21.

Observe que na figura acima, há um módulo “multiplexador”. Multiplexador é um componente que permite selecionar um canal para que somente um dispositivo possa, em um determinado tempo, enviar ou receber informações do processador.

1.2.3 Terceira geração

Os computadores iniciais da segunda geração tinham na ordem de 10 mil transistores em sua implementação. Mas, a partir da implementação de circuitos integrados (1958), esse número teve um aumento muito expressivo chegando a centenas de milhares na terceira geração. A partir disso, podemos destacar nesta geração (SILBERSCHATZ; GALVIN; GAGNE, 2015):

- **o aparecimento das memórias baseadas em semicondutores e discos magnéticos**: a partir desta geração, passou-se a utilizar componentes baseados em semicondutores (silício, por exemplo) para a fabricação de memória. Além dos semicondutores, o sistema de memória também foi contemplado com os discos magnéticos (memória secundária);
- **a criação da técnica de *spool*** (*Simultaneous Peripheral Operation On-Line*): ao invés dos canais de dados serem manipulados de forma multiplexada, como na segunda geração, a manipulação dos dispositivos de E/S passou a ser simultânea;
- **evolução dos sistemas operacionais com o aparecimento da multiprogramação**: essa técnica consiste em compartilhar o processamento entre vários programas abertos. A execução dos programas se dá de forma alternada. Para que o processador não seja monopolizado, um certo programa é executado durante uma fatia de tempo (chamada como *time slice*) ou até ser provocado um evento de E/S. Quando um desses fatos ocorrer, o programa em questão deixa de alocar o processador para que um outro programa o aloque, repetindo-se o processo. Essa alternância é gerenciada pelo Sistema Operacional por meio do escalonador de processos.

Devido ao escalonamento pelo tempo, a multiprogramação é também denominada como “tempo compartilhado” (figura a seguir). Com essa ideia, surgiram alguns sistemas operacionais como o CTSS/MIT, que proporcionou o aparecimento do MULTICS, que evoluiu para o UNIX. Na figura a seguir, temos a execução de três processos (P0, P1 e P2) concorrendo ao uso do processador.

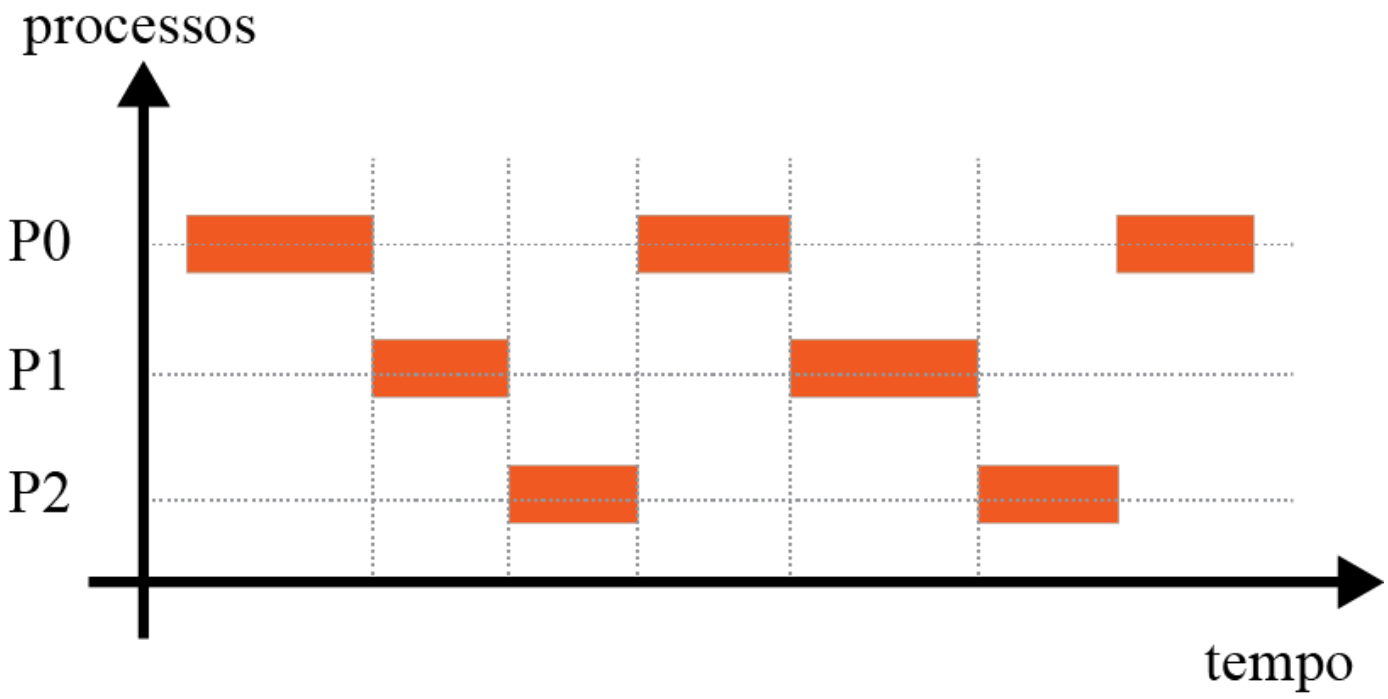


Figura 5 - Neste escalonamento, os três processos concorrem ao acesso à CPU, porém, apenas um, de cada vez, é detentor do recurso. Fonte: elaborada pelo autor, 2018.

Voltando ao histórico, um marco que se destacou nesta geração foi o lançamento da linha IBM System 360, em 1964.

Para este lançamento, a estratégia da IBM foi lançar modelos distintos de computadores dentro do linha 360, criando, então, a família 360. Os computadores desta família eram compatíveis em relação ao código das aplicações desenvolvidas e rodavam o mesmo Sistema Operacional (OS 360). Os computadores da família 360 diferenciavam apenas em relação aos custos, tamanho da memória, número de canais de dados e performance de processamento (SILBERSCHATZ; GALVIN; GAGNE, 2015).

Você poderia, agora, traçar um paralelo entre a ideia de família 360 e a família x86 que temos atualmente? O que se mantém nesses computadores da mesma família?

Por fim, não poderíamos deixar de citar, dentro desta geração, o lançamento do DEC PDP-8 no mesmo ano de lançamento do IBM 360. O diferencial entre os dois modelos é que a DEC continuou estimulada a produzir minicomputadores para ganhar espaço em um segmento de mercado não coberto pela IBM: o de baixo custo (US$ 16 mil do PDP-8 contra centenas de milhares de dólares do IBM 360).

VOCÊ SABIA?

O termo OEM (*Original Equipment Manufacturers*) é utilizado quando um módulo (por exemplo, um circuito ou dispositivo) é criado por uma empresa para que possa fazer parte do produto de final de outra empresa. Você sabia que essa ideia foi concebida pela DEC em seu computador PDP-8 na década de 1960?

Para finalizar, também não poderíamos deixar de mencionar que a DEC lançou, em seu PDP-8, a ideia de barramento para interconectar os módulos do computador. A ideia deste barramento, o "*Omnibus*", deu origem aos barramentos que conhecemos atualmente equipando os microcomputadores.

1.2.4 Quarta geração

A quarta geração está vinculada à alta escala de integração dos transistores dentro dos circuitos integrados. Na ULSI (*Ultra Large Scale Integration*) temos mais de um milhão de transistores, dentro de um mesmo *chip*. Essa redução drástica do espaço ocupado pelos componentes eletrônicos propiciou o aparecimento dos microprocessadores e multiprocessadores.

VOCÊ QUER LER?

Em 1965, o co-fundador da Intel, Gordon Moore, anunciou a profecia de que a capacidade (e o número de transistores) de um processador dobraria, mantendo-se o custo. Essa profecia se tornou verdade e, atualmente, é considerada como a "Lei de Moore". Ainda não se sabe até quando essa lei será mantida como verdade (MARKOFF, 2016). Leia mais na matéria do New York Times, que mostra que a lei de Moore ainda é um desafio: <http://www1.folha.uol.com.br/tec/2016/05/1768028-lei-de-moore-esta-se-esgotando-e-tecnologia-busca-sucessores.shtml (http://www1.folha.uol.com.br/tec/2016/05/1768028-lei-de-moore-esta-se-esgotando-e-tecnologia-busca-sucessores.shtml)>.

O alto fator de integração possibilitou a introdução de técnicas mais agressivas de processamento no que diz respeito à performance computacional e a gama de funcionalidades. As técnicas presentes nos computadores atuais têm como foco garantir a alta disponibilidade das informações e do processamento. Para exemplificar esse foco, podemos citar o *pipeline* e a superescalaridade.

Em relação aos *softwares*, podemos mencionar o surgimento de linguagens de programação como C++, SQL, JAVA, assim como os sistemas operacionais de rede e distribuídos.

1.3 Projetando e visando ao desempenho

Do ponto de vista da computação podemos perguntar: o *hardware*, em relação ao seu desempenho, tem que acompanhar as novas aplicações que exigem cada vez mais recursos, ou as aplicações cada vez mais complexas aproveitam dos recursos oferecidos pelos novos equipamentos?

Independentemente da resposta, o que se sabe é que o poderio dos processadores e dos computadores, como um todo, sofre uma contínua evolução frente ao seu desempenho. Um outro ponto merece ser instigado: como se ganha tanta performance se os conceitos usados ainda hoje, provém do modelo proposto por von Neumann em sua máquina IAS?

Como resposta inicial, podemos dizer que a evolução tecnológica provida pela miniaturização dos componentes (estamos na era da nanoeletrônica, pois os componentes estão na casa dos nanômetros) permite que circuitos mais complexos sejam implementados, ocupando o mesmo espaço físico. Uma outra questão é que os sinais, em circuitos muito menores, apresentam um tempo de propagação menor, pois as distâncias a serem percorridas são igualmente menores.

Porém, além do avanço da eletrônica, a melhoria de desempenho é alcançado por intermédio da introdução de novas técnicas e metodologias de processamento.

Como técnicas adicionais, além das representadas pelo paralelismo encontrado nos processadores, podemos citar (STALLINGS, 2010):

- **previsão de desvio**: caso nenhuma técnica fosse utilizada, ao se deparar com uma instrução que contenha um desvio (do tipo ‘if...else’ ou um laço de repetição), o processador ficaria a esperar pelo resultado da expressão associada para que tomasse a decisão de qual linha a ser executada. Na previsão de desvio, é criada uma tabela que contém o resultado do desvio (desvio tomado ou não tomado). Cada linha da tabela correspondente aos *bits* menos significativos dos endereço de memória (localização da instrução). Com base nesta tabela de histórico, o processador tenta antecipar o início da próxima instrução. Caso tenha tomado a decisão errada, nenhum impacto de atraso será inserido no processamento pois será como o processador ficasse esperando pelo resultado da condição do desvio. Caso tenha acertado na previsão, então economizou-se o tempo de espera pelo resultado da condição do desvio;
- **análise do fluxo de dados**: na análise de fluxo, observamos as instruções em relação a suas dependências quanto à disponibilidade de recursos de *hardware* e de dados. Neste caso, pode haver uma execução fora de ordem quando a instrução apta a ser executada não corresponde à próxima imediata na sequência;
- **execução especulativa**: representa a fusão da previsão de desvio com a análise de fluxo, de forma a tentar antecipar ao máximo o início da próxima instrução para manter o *pipeline* cheio.

1.3.1 Balanço ao desempenho

Caso sejam colocados dois processadores com frequências de *clock* diferentes, qual o mais eficiente? Aquele que apresenta a maior frequência de *clock*? A resposta não é bem essa. Para se decidir qual o melhor, deve-se realizar uma análise não somente do processador mas, também, de todo o conjunto.

Balanço computacional refere-se a atenuar os gargalos que afetam a performance na execução das instruções. Por exemplo, não basta o processador atuar em uma frequência extremamente grande, se a memória não for capaz de fornecer-lhe as informações com a eficiência necessária.

Quanto à memória, algumas técnicas poderão ser utilizadas para diminuir a diferença de desempenho entre a memória e o processador. Vamos entender a seguir (STALLINGS, 2010).

- **Aumentar a largura de transferência entre a memória e o processador**: um dos grandes vilões para a performance de processamento é o barramento – canal por onde passam as informações vinculadas aos dispositivos de E/S e à própria memória. Aumentando-se a largura de transferência, com uma única requisição, é possível a transferência de um conjunto maior de *bits*. Com isso evitamos que o tráfego de sinais de sincronização de barramento e de endereços impacte negativamente na taxa de eficiência de transmissão do barramento. Uma outra técnica para evitar gargalos de transferência entre memória e processador consiste em criar um barramento exclusivo entre essas duas entidades.
- **Introdução de *caches* mais eficientes ou mais próximas ao nó de processamento**: a memória *cache* evita o acesso frequente à memória principal (MP). O acesso à MP tem dois pontos negativos em relação ao acesso à *cache*: a tecnologia empregada na MP é mais lenta em relação à *cache* e quanto mais longe o módulo de memória estiver do núcleo de processamento, maior será a latência de tempo, devido ao maior percurso que o sinal deverá percorrer.
- **Aumentar o número de portos de leitura e escrita da memória**: quando se opera com paralelismo no processamento das instruções, a probabilidade de se ter duas requisições para a utilização da memória é alta. Nesta situação, caso a memória tenha apenas um porto de leitura/escrita, umas das instruções ficaria bloqueada até o término da outra. Para se evitar esse problema, é necessário dotar a memória de dois portos (canais) de leitura/escrita para que ambas as instruções possam acessá-la simultaneamente.
- **Melhoria do desempenho dos dispositivos de E/S**: de nada adianta aumentar o poderio do processador, da memória e dos barramentos, se os dispositivos de E/S continuarem a ser um gargalo para a performance computacional. Diante disso, além da preocupação em se construir dispositivos mais eficientes, técnicas de otimização do tratamento das

requisições e de *cache* (em nível das unidades de controle de cada dispositivo de E/S) vêm sendo implantadas.

VOCÊ SABIA?

Você sabia que, para reduzir o tráfego de endereços e de sinais de controle, a memória trabalha no modo “rajada” (*burst*)? Neste modo de operação, a memória poderá enviar informações cujas localizações (endereços) sejam consecutivas ao endereço de uma solicitação anterior. O artigo no portal “Clube do Hardware” (TORRES, 2016), menciona esse aspecto: <https://www.clubedohardware.com.br/artigos/memoria/tudo-o-que-voc%C3%AA-precisa-saber-sobre-as-temporiza%C3%A7%C3%B5es-das-mem%C3%B3rias-ram-r34433/?nbcpage=3 (https://www.clubedohardware.com.br/artigos/memoria/tudo-o-que-voc%C3%AA-precisa-saber-sobre-as-temporiza%C3%A7%C3%B5es-das-mem%C3%B3rias-ram-r34433/?nbcpage=3)>.

Além dos investimentos dos sistemas de memória, barramentos e dispositivos de E/S, muito se tem investido para se ter processadores cada vez mais poderosos e eficientes, principalmente para aumentar o fator de integração de seus componentes (em nível de transistores) ou seja, na adoção de técnicas de processamento mais agressivas. Como duas principais técnicas, temos o *pipeline* e a superescalaridade.

1.3.2 Pipeline e superescalaridade

Como dito anteriormente, *pipeline* e superescalaridade são técnicas, baseadas na execução paralela das instruções, que visam o aumento da performance dos sistemas computacionais.

O ponto chave do *pipeline* é tentar antecipar o início da próxima instrução. Mas, como fazer isso? Para começar, vamos falar que, para se alcançar um melhor resultado, uma instrução pode ser decomposta em várias microinstruções (STALLINGS, 2010). Para uma melhor abstração, vamos imaginar uma sequência para calcular a ação de um jogador no futebol de robôs, composta de quatro etapas:

1) obter uma imagem que representa a situação atual do campo da partida;

2) decodificar a imagem para que se possa gerar as posições de todos os jogares e da bola (por exemplo, no formato de coordenadas cartesianas);

3) diante do posicionamento dos robôs do próprio time e do adversário, gerar uma estratégia de ação;

4) colocar em prática a estratégia gerada com a ativação da movimentação dos robôs.

Para tanto, vamos imaginar, inicialmente, que as etapas citadas sejam executadas sequencialmente, e para se iniciar o processamento de uma imagem temos que finalizar, totalmente, o processamento da imagem anterior (desde a coleta da imagem até a movimentação dos robôs). Essa abstração sequencial é ilustrada na figura abaixo, item (a).

Em um segundo momento, imagina-se que cada etapa seja executada por um *hardware* específico. Dessa forma, o resultado de uma etapa, assim que for finalizada, será encaminhado para a etapa seguinte. O *hardware* que passou a informação para o módulo seguinte, ao invés de ficar ocioso, começa a atuar no processamento da próxima informação. O item (b) da figura a seguir, mostra exatamente essa antecipação do processamento da imagem seguinte.

Etapas \ Tempos	1	2	3	4	5	6	7	8	9	10	11	12
Obter Imagem	Img 1				Img 2				Img 3			
Decodificar Imagem		Img 1				Img 2				Img 3		
Gerar Estratégia			Img 1				Img 2				Img 3	
Movimentar robôs				Img 1				Img 2				Img 3

(a)

Etapas \ Tempos	1	2	3	4	5	6	7	8	9	10	11	12
Obter Imagem	Img 1	Img 2	Img 3									
Decodificar Imagem		Img 1	Img 2	Img 3								
Gerar Estratégia			Img 1	Img 2	Img 3							
Movimentar robôs				Img 1	Img 2	Img 3						

(b)

Figura 6 - Diferenças entre o tempo de processamento sequencial (a) e com pipeline (b). Fonte: Elaborada pelo autor, 2018.

Na figura acima, em (b), percebemos que a antecipação do início da próxima etapa, conseguida pela divisão da ação em etapas, fez com que, mesmo não diminuindo o tempo para processar uma imagem (definida na figura com o termo “img”), conseguiu-se uma redução no tempo de processamento de um conjunto de imagens. No caso exemplificado, para se processar três imagens, passou-se de 12 para apenas seis unidades de tempo.

VOCÊ SABIA?

Você sabia que você pode usar essa abstração de *pipeline* também em nível de *software*? Para tanto, basta implementar o seu programa usando *threads*. As *threads* são funções do seu programa que se comportarão como se fossem programas independentes – porém usando o mesmo espaço de endereçamento (por exemplo, podendo acessar as mesmas variáveis globais). Para saber mais sobre programação usando *threads*, você poderá procurar pela biblioteca *pthread* (*POSIX threads*).

Para finalizar, devemos saber que, além de implementar a técnica de *pipeline*, um processador pode ser, também, superescalar. A superescalaridade consiste em se replicar unidades funcionais de processamento para que se possa processar duas ou mais instruções simultaneamente, ou seja, de forma paralela.

1.4. Evolução das arquiteturas

Atualmente, o cenário mundial dos processadores está polarizado entre dois mercados: o representado pelos computadores pessoais e o representado pelos sistemas embarcados (também chamados por sistemas embutidos).

No âmbito dos computadores pessoais, encontramos computadores baseados em processadores da família x86 e, em relação aos sistemas embarcados, processadores ARM (*Advanced RISC Machine*).

Vamos conhecer cada um deles a seguir.

1.4.1 Intel x86

A família x86 é muito importância para o mundo dos sistemas computacionais, não somente pela sua ampla utilização mas, também, por implantar vários aspectos tecnológicos nos projetos de seus processadores.

ão

to de seu lançamento, na década de 1980, o processador da Intel 8086 tinha, como principal concorrente, MC68000, que equipava os Apple Macintosh. Eles se diferenciavam completamente, tanto em relação à quanto à organização. Como exemplo, podemos citar a largura dos registradores: o processador da Intel radores de 16 *bits* e o modelo da Motorola manipulava registradores de 32 *bits*. Em linhas gerais, o 68000 sejável pelos programadores do que o 8086. Mas, se o modelo da Motorola é mais desejável e eficiente, nercado atualmente é dominado por processadores derivados do Intel 8086? A resposta baseia-se em le mercado. A Intel permitiu que qualquer fabricante de computadores pudesse usar o seu processador direitos (*royalties*), enquanto que a Motorola não tinha esse modelo de negócio. Diante dessa de das indústrias produzirem clones e placas-mãe baseadas no 8086 sem ter a necessidade de pagar os com o mercado apresentasse um crescimento muito rápido da utilização da plataforma da Intel. Com conseguiu fazer que o seu modelo dominasse o mercado, que ainda hoje é consequência desta estratégia gica.

Durante a evolução da família x86, podemos destacar os pontos principais (STALLINGS, 2010):

- 8080: primeiro microprocessador de uso geral; caminho de dados de 8 *bits*;
- 8086: contém 29 mil transistores; 16 *bits*; *cache* de instruções (pré-busca das instruções);
- 80386: suporte para multitarefa; 32 *bits*;
- 80486: *cache* evoluída para dados e instruções; presença de *pipeline*; coprocessador matemático embutido;
- Pentium: organização superescalar;
- Pentium Pro: superescalaridade aumentada; renomeamento de registradores; previsão de desvio; execução especulativa;
- Pentium II: incorporação de tecnologia MMX (para manipulação de informações multimídia);

- Core: primeiro x86 com dual core;
- Core 2: arquitetura de 64 *bits*;
- Core 2 Quad: 820 milhões de transistores; quatro processadores no chip.

VOCÊ QUER LER?

Nos processadores atuais, como o *Intel Core-i7*, comenta-se a existência de processadores (ou núcleos) lógicos (ou virtuais) e físicos. Mas, o que vem a ser isso? Recomendamos um artigo (TELES, 2009) que descreve a arquitetura do i7, desde o gerenciamento de memória até a descrição de núcleos virtuais e físicos: <http://www.ic.unicamp.br/~ducatte/mo401/1s2009/T2/042348-t2.pdf (http://www.ic.unicamp.br/~ducatte/mo401/1s2009/T2/042348-t2.pdf)>.

Atualmente, novas pesquisas vem sendo desenvolvidas para que ataque, de forma ainda mais agressiva, o paralelismo internamente ao processador. Processadores com mais núcleos (tanto físicos quanto virtuais) estão sendo lançados no mercado, possibilitando o paralelismo em vários níveis de abstração.

1.4.2 Sistemas embarcados e ARM

Um sistema embarcado é aquele que apresenta a combinação entre *software* e *hardware*, para executar um objetivo dedicado. Geralmente falamos que um sistema computacional é dedicado, quando ele desempenha apenas um tipo de funcionalidade, por exemplo, controlar um dispositivo tendo em vista os dados coletados por meio de sensoriamento. Aos sistemas embarcados, geralmente associamos condições limites como (STALLINGS, 2010):

- pequena quantidade de memória disponível para conter o programa e suas variáveis;
- limitação na dissipação de potência e, portanto, no consumo de energia;
- condições ambientais agressivas, como vibrações, altas amplitudes térmicas;
- geralmente associados com manipulação de sinais analógicos provenientes de sensores e interfaceados por conversores A/D

(analógico-digital) e enviados para atuadores por intermédio de conversores D/A (digital-analógico);

- não se limita a um processador por sistema embarcado: um único sistema embarcado pode conter, além do processador, outros elementos de processamento, tais como: DSP (*Digital Signal Processor*), FPGA (*Field Programmable Gate Array*) e circuitos do tipo ASIC (*Application Specific Integrated Circuit*).

A figura abaixo mostra as funcionalidades que podemos encontrar nos sistemas embarcados.

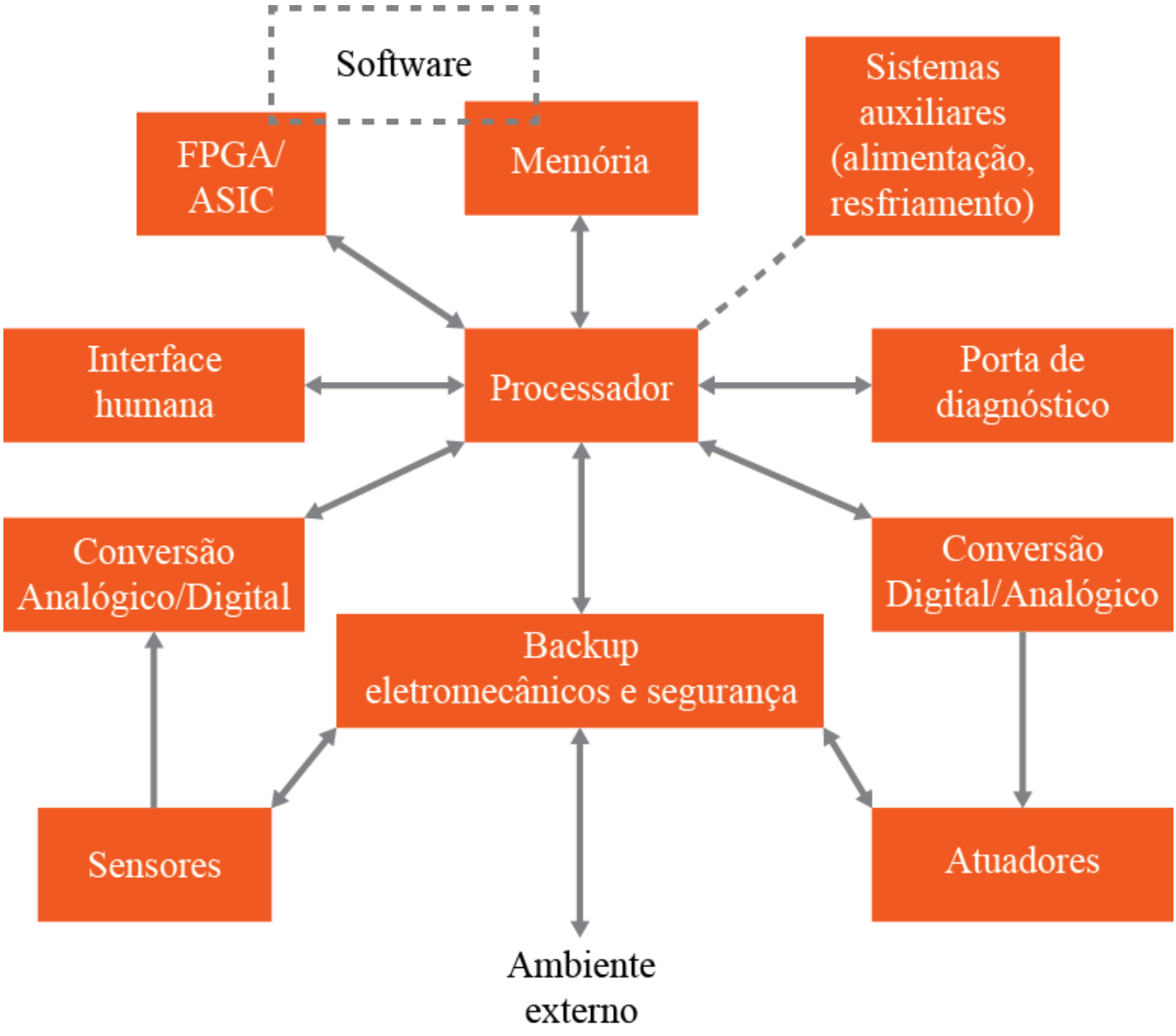


Figura 7 - Funcionalidades básicas que podemos encontrar em sistemas embarcados, que podem possuir vários nós de processamento. Fonte: Fonte: STALLINGS, 2010, p. 37.

Sabemos que grande parte dos sistemas embarcados é equipada com processadores ARM, então, vamos conhecer algumas de suas características básicas (STALLINGS, 2010):

- projetado pela ARM Inc., Cambridge, Inglaterra. Projetos posteriores feitos em conjunto entre a Acorn, Apple, VLSI Technology. Atualmente, processadores ARM licenciados para que sejam construídos por diversas empresas, como Cirrus Logic, Texas Instruments, ATMEL, Samsung, Nvidia, Sharp, Qualcomm;
- alta velocidade, baixos requisitos de potência;
- encontrado em PDAs, jogos portáteis, telefones, freios ABS e aplicações que envolvem tempo real;
- suporta sistemas operacionais como Linux, Linux, Palm OS, Symbian OS, Windows Mobile.

Como você pode ter notado, desde o primeiro computador eletrônico até os modelos atuais, houve um grande salto, tanto da tecnologia eletrônica quanto dos métodos e algoritmos usados para acelerar ainda mais o poder computacional. Com o que aprendemos até aqui, podemos refletir sobre até quando será válida a Lei de Moore e quais serão as próximas inovações na informática, como alguns dos questionamentos atuais desse campo de estudo.

Síntese

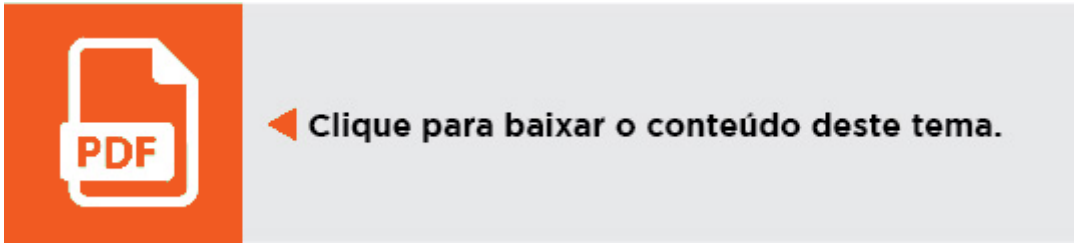
Chegamos ao final deste capítulo. Vimos aqui os conceitos iniciais importantes para começar a entender o funcionamento do computador. Nestes conceitos, estão presentes a definição de como o computador está organizado, em sua arquitetura e organização, assim como realizar a diferenciação entre estrutura e função.

Com os pontos aqui apresentados, esperamos que você prossiga os estudos relacionados ao funcionamento do computador e possa fazer comparações iniciais entre modelos de máquinas dentro de um processo de consultoria técnica,

por exemplo.

Neste capítulo, você teve a oportunidade de:

- compreender as diferenças entre arquitetura e organização, de modo a poder reconhecer e comparar famílias de processadores; listar as funcionalidades de um processador relacionando-as com as estruturas internas, representadas pelos submódulos de *hardware*;
- visualizar os computadores em função de seu contexto histórico, podendo aplicar os conceitos inerentes a cada geração de computadores às características dos computadores atuais;
- identificar os pontos que impactam no desempenho de processamento de um computador, assim como analisar e formular solução para a sua melhoria.



Bibliografia

NISENBAUM, M. A. **Condutores elétricos**. Canal VideosEducativos, YouTube, publicado em 16 de fevereiro de 2013. Tudo se transforma – Equipe Departamento de Química. Produção: Alexandre Sivolella; Alvaro Furloni; Breno Kuperman; Davi Kolb; Lígia Diogo. Disponível em: <https://www.youtube.com/watch?list=PLeH2yLt2MpadUzNZzRBWLePFDQDmlIb54&v=Szj09sb8uoU (https://www.youtube.com/watch?list=PLeH2yLt2MpadUzNZzRBWLePFDQDmlIb54&v=Szj09sb8uoU)>. Acesso em: 03/05/2018.

NUNES, M. Alan Turing: o pai da computação. **Jornal PETNews**. Edição: Jeymisson Oliveira; Revisão: Savyo Nóbrega e Joseana Fechine. Grupo PET Computação UFCG, 2012. Disponível em: <http://www.dsc.ufcg.edu.br/~pet/jornal/junho2012/materias/historia_da_compu

tacao.html (http://www.dsc.ufcg.edu.br/~pet/jornal/junho2012/materias/historia_da_computacao.html)>. Acesso em: 03/05/2018.

MARKOFF, J. Lei de Moore está se esgotando, e tecnologia busca sucessores. Portal **Folha de S. Paulo** [do New York Times, tradução de Paulo Migliacci], publicado em: 05/05/2016. Disponível em: <http://www1.folha.uol.com.br/tec/2016/05/1768028-lei-de-moore-esta-se-esgotando-e-tecnologia-busca-sucessores.shtml (http://www1.folha.uol.com.br/tec/2016/05/1768028-lei-de-moore-esta-se-esgotando-e-tecnologia-busca-sucessores.shtml)>. Acesso em: 03/05/2018.

PORTAL EBC. Conheça a história de Ada Lovelace, a primeira programadora do mundo. Portal **Empresa Brasil de Comunicação**, publicado em 02/03/2015. Disponível em: <http://www.ebc.com.br/tecnologia/2015/03/conheca-historia-da-ada-lovelace-primeira-programadora-do-mundo (http://www.ebc.com.br/tecnologia/2015/03/conheca-historia-da-ada-lovelace-primeira-programadora-do-mundo)>. Acesso em: 03/05/2018.

SILBERSCHATZ, A.; GALVIN, P. B; GAGNE, G. **Fundamentos de Sistemas Operacionais**. 9. ed. Rio de Janeiro: LTC, 2015. Disponível na Biblioteca Virtual-Minha Biblioteca: <https://integrada.minhabiblioteca.com.br/#/books/978-85-216-3001-2 (https://integrada.minhabiblioteca.com.br/#/books/978-85-216-3001-2)>. Acesso em: 03/05/2018.

STALLINGS, W. **Arquitetura e Organização de Computadores**. 8. ed. São Paulo: Pearson Pratice Hall, 2010. Disponível na Biblioteca Virtual: <https://Ânimabrasil.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_198689_1&content_id=_4122211_1&mode=reset (https://Ânimabrasil.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_198689_1&content_id=_4122211_1&mode=reset)>. Acesso em: 03/05/2018.

TELES, B. **O Processador Intel Core i7**. Arquitetura de Computadores I, Unicamp, publicado em 04/07/2009. Disponível em: <http://www.ic.unicamp.br/~ducatte/mo401/1s2009/T2/042348-t2.pdf (http://www.ic.unicamp.br/~ducatte/mo401/1s2009/T2/042348-t2.pdf)>. Acesso em: 03/05/2018.

TORRES, G. Tudo o que você precisa saber sobre as temporizações das memórias RAM. Portal **Clube do Hardware**, publicado em 18/01/2016. Disponível em: <https://www.clubedo (https://www.clubedohardware.com.br/artigos/memoria/tudo-o-que-voc%C3%AA-precisa-saber-sobre-as-temporiza%C3%A7%C3%B5es-das-mem%C3%B3rias-ram-r34433/?nbcpage=3)*hardware (https://www.clubedohardware.com.br/artigos/memoria/tudo-o-que-voc%C3%AA-precisa-saber-sobre-as-temporiza%C3%A7%C3%B5es-das-mem%C3%B3rias-ram-r34433/?nbcpage=3)*.com.br/artigos/memoria/tudo-o-que-voc%C3%AA-precisa-saber-sobre-as-temporiza%C3%A7%C3%B5es-das-mem%C3%B3rias-ram-r34433/?nbcpage=3 (https://www.clubedohardware.com.br/artigos/memoria/tudo-o-que-voc%C3%AA-precisa-saber-sobre-as-temporiza%C3%A7%C3%B5es-das-mem%C3%B3rias-ram-r34433/?nbcpage=3)>. Acesso em: 03/05/2018.