Fundamentos de Sistemas Operacionais

Parte I – Introdução e Histórico dos Sistemas Operacionais

Introdução

O computador é uma máquina composta por diversos elementos de hardware e software que devem funcionar de forma harmônica, eficiente e segura. Além disso, é fundamental que ele seja uma ferramenta simples e acessível a qualquer pessoa.

Para garantir as características acima, é necessário que haja "alguém" ou "algo" responsável por gerenciar os recursos do computador (hardware e software) e, ao mesmo tempo, tornar amigável a interação homem/máquina.

O sistema operacional é um conjunto de rotinas/programas que têm como funções principais o gerenciamento dos recursos do computador e realizar a interface entre este e o usuário.

A execução das rotinas que compõem o Sistema Operacional é realizada pelo processador de forma semelhante à execução dos programas do usuário. Porém, no caso do sistema operacional, essas rotinas são acionadas por eventos que ocorrem de forma assíncrona, ou seja, sem que haja uma previsão do momento em que vão ocorrer.

Quanto à função de interface, é importante perceber que quando um usuário deseja, por exemplo, imprimir um texto ou gravar um arquivo no disco rígido, ele não precisa conhecer os detalhes de funcionamento da impressora ou do disco. Basta um comando simples e a operação é realizada sem maiores dificuldades, já que o sistema operacional se encarrega da comunicação com o dispositivo em uso. Isto não era possível nos primórdios da computação, quando os sistemas operacionais não existiam ou eram ainda muito precários. Somente os usuários que conheciam profundamente o funcionamento da máquina podiam operá-la.

A evolução dos computadores foi acompanhada do aumento da demanda e exigência por parte dos usuários. Cada vez mais funções eram requeridas e, portanto, mais funções foram sendo acrescentadas aos sistemas operacionais. Um dos aspectos mais interessantes e que dão uma ideia da importância do sistema operacional para a eficência e até mesmo a existência dos computadores é a multiprogramação. Conforme veremos adiante com mais detalhes, essa técnica permite que diversos programas sejam carregados na memória e concorram pelos recuros do computador, especialmente o processador e a própria memória. Assim, é necessário um sistema que gerencie o compartilhamento desses recursos para que o computador funcione corretamente e com a rapidez e eficiência desejadas.

Neste curso estudaremos as funções genéricas[1] dos sistemas operacionais e os conceitos necessários para a compreensão do seu funcionamento.

A partir de agora vamos ver um pequeno histórico da computação, que permitirá entendermos como os sistemas operacionais foram se tornando componentes essenciais ao funcionamento, eficiência e disseminação dos computadores.

Breve Histórico

Como já foi mencionado anteriormente, a operação dos primeiros computadores exigia um profundo conhecimento por parte daqueles que o utilizavam. Assim, era necessário que se soubesse como os dispositivos funcionavam e o trabalho envolvia a manipulação de fios em paineis, o que tornava a atividade complexa e sujeita a muitas falhas. O surgimento e evolução dos sistemas operacionais contribuiram para simplificar as tarefas e minimizar as falhas, tornando as funções

[1] As funções, conceitos e técnicas que abordaremos são comuns a todos os sistemas operacionais. Quando oportuno, poderemos citar alguns detalhes específicos de algum sistema operacional.

executadas pelo hardware cada vez mais transparentes do ponto de vista dos usuários. Vamos acompanhar a partir de agora essa evolução.

Entre todas as importantes contribuições ao longo dos séculos para o surgimento e desenvolvimento dos computadores como os conhecemos hoje, vale destacar a lógica *booleana*, desenvolvida pelo matemático inglês George Boole. Este conceito de lógica digital baseada em dois valores (0 e 1) foi fundamental para a criação de dispositivos como relés e válvulas, base dos primeiros computadores construídos.

Na década de 1940, a Segunda Guerra Mundial criou a necessidade de desenvolver equipamentos que tornassem os cálculos para procedimentos militares mais rápidos.

Durante este período destacaram-se alguns equipamentos desenvolvidos tais como o Colossus (Alan Turing), o Mark I (utilizado pela Marinha norte-americana) e o ENIAC (primeiro computador digital eletrônico).

O mais interessante nesta década foi o conceito de "programa armazenado" desenvolvido por John Von Neumann em que a ideia principal era ter o programa e os dados armazenados na mesma memória, tornando o processo de programação mais simples e flexível. Este conceito é utilizado até hoje em nossos computadores, que podem ser vistos como "Arquiteturas de Von Neumann".

Também é importante ressaltar que as máquinas dessa época não possuíam o que entendemos hoje como sistema operacional e não existiam também dispositivos de E/S tais como monitores e teclados.

A década seguinte marcou o surgimento dos primeiros computadores a utilizar transistores e memórias magnéticas[2], o que permitia máquinas mais confiáveis, com mais velocidade de acesso aos dados e maior capacidade de armazenamento. O computador UNIVAC I foi o primeiro computador fabricado para fins comerciais.

O surgimento do primeiro sistema operacional, no Centro de Pesquisas da General Motors, foi uma tentativa de automatizar determinadas tarefas que eram simples porém muito repetitivas e, portanto, passíveis de erros. Surgiu o processamento *batch*, através do qual diversas tarefas eram submetidas ao computador de uma só vez. Este então as executava sequencialmente na ordem em que foram submetidas.

Com o aparecimento das primeiras linguagens de programação de alto nível, foi necessário que os sistemas operacionais se aperfeiçoassem para tornar as taferas de manipulação do hardware transparentes para os programadores. Também foi nesta década que começou a se desenvolver o conceito de *memória virtual*, que será um dos nossos tópicos mais adiante.

A década seguinte foi marcada pelo aparecimento do conceito de multiprogramação, que permitia mais de um programa na memória simultaneamente, concorrendo pelos recursos do computador. Esta tecnologia trouxe como grande vantagem permitir uma utilização mais intensa dos recursos, uma vez que, enquanto um determinado processo estivesse aguardando alguma informação ou evento, outro estaria usando o processador. Durante nosso curso, estudaremos os diversos aspectos do sistema operacional necessários para suportar a multiprogramação.

Também neste período surgiram os sistemas *time-sharing*, que abordaremos mais adiante e os primeiros dispositivos de E/S, teclados e monitores.

Diversas tecnologias foram tornando os computadores mais eficientes, baratos e menores, tais como as tecnologias LSI (Large Scale Integration) e VLSI (Very Large Scale Integration). Essas tecnologias eram acompanhadas pela evolução dos sistemas operacionais, que passaram a suportar mais de um usuário (sistemas multiusuário). Nesta época, também surgiram o primeiro *Personal Computer* (PC), lançado pela IBM, que começou a popularizar o uso de computadores domésticos e os primeiros sistemas operacionais de rede.

[2] O Whirlwind, desenvolvido pelo MIT (Massachusets Institute of Techology), foi o primeiro computador a utilizar memória magnética.

É importante ressaltar que os desenvolvimentos de harware e software caminham lado a lado se influenciando mutuamente. Se o hardware evolui, o software tem que evoluir para acompanhar as novas tecnologias. Da mesma forma, o desenvolvimento do software, em especial o sistema operacional, torna mais fácil o uso do computador, o que aumenta a demanda por parte do mercado, fomentando assim mais pesquisas e desenvolvimento de hardware, num processo dinâmico e contínuo.

A década de 1990 teve como um dos seus marcos principais o *boom* da internet e a consequente consolidação do protocolo TCP/IP. Assim, os sistemas operacionais passaram a ter suporte para esse protocolo para que pudessem ser compatíveis com as demandas do mercado. Além disso, as interfaces gráficas passaram a ser um componente básico para todos os sistemas operacionais.

As arquiteturas de computadores baseadas no modelo de Von Neumann vêm apresentando algumas limitações com a demanda crescente por desempenho computacional. O que é comumente chamado de *Gargalo de Von Neumann* é uma limitação causada pela comunicação entre o processador e o sistema de memória. Resumidamente, o que acontece é que o processador perde algum tempo aguardando as operações de leitura/escrita na memória. Algumas soluções vêm sendo propostas para eliminar ou minimizar esse problema, tais como a introdução das memórias *cache* e o processamento paralelo. Desta forma, os sistemas operacionais têm que suportar essas tecnologias.

Atualmente, os sistemas operacionais devem garantir as funções que permitam a implantação dos sistemas distribuídos, onde dados procedimentos e funções se encontram em máquinas fisicamente distantes umas das outras.

Parte II – Classificação dos Sistemas Operacionais

Para classificarmos qualquer objeto é necessário definir os parâmetros para tal classificação. Por exemplo, se vamos classificar pessoas, podemos fazer isso por idade, sexo, escolaridade e nacionalidade, entre outros.

No caso dos sistemas operacionais não é diferente. Devemos escolher os parâmetros a serm usados para diferenciá-los. Entre estes parâmetros identificamos a quantidade de processos na memória, a quantidade de usuários que podem usar o sistema simultaneamente, a interatividade com os usuários e a quantidade de unidades de processamento envolvidas. A partir de agora, vamos ver como classificar os sistemas operacionais quanto a estes parâmetros.

1. Sistema monoprogramáveis x Sistemas multiprogramáveis

Uma das maneiras de diferenciarmos os sistemas operacionais é quanto à quantidade de programas presentes na memória.

Os sistemas monoprogramáveis são aqueles em que só há um processo sendo executado e para o qual todos os recursos da máquina estão disponíveis. Vamos entender porque estes sistemas não são mais usados atualmente. O problema de termos apenas um processo[3] é que há um desperdício dos recursos do sistema. Por exemplo, se o processo está em execução (utilizando o processador), os demais recursos, tais como os dispositivos de E/S ficam ociosos. Se, em algum momento ele precisar de uma operação de E/S, como acessar o disco rígido ou aguardar um comando do usuário, o processador ficará ocioso. Além disso, por termos apenas um processo carregado, a ocupação da memória pode ser apenas parcial, o que também é um desperdício de recurso.

Já os sistemas multiprogramáveis não apresentam esse tipo de limitação. Por permitirem mais de um processo na memória, concorrendo pelos recursos da máquina, eles conseguem que as potencialidades do processador e do computador como um todo sejam mais exploradas. Assim, enquanto um processo aguarda uma operação de E/S, outro processo pode ser atendido pelo processador, maximizando a utilização dos recursos.

Atualmente, todos os sistemas operacionais são multiprogramáveis e as funcionalidades decorrentes dessa característica aumentam a quantidade de tarefas que cabem ao sistema operacional. Como são vários processos na memória é necessário organizar o acesso ao processador (escalonamento de processos), organizar e proteger a memória para que os processos não misturem seus dados e instruções, e administrar e resolver os conflitos que eventualmente ocorrem em razão de vários processos precisarem utilizar os mesmos recursos. Podemos dizer que a maior parte do que estudaremos de agora em diante é consequência da implementação da multiprogramação.

2. Sistemas monousuários x Sistemas multiusuários

Quanto à quantidade de usuários que podem utilizar e/ou compartilhar os recursos dos computadores, os sistemas operacionais podem ser classificados em *monousuários*, que permitem apenas um usuário de cada vez utilizando o sistema e *multiusuários*, que, obviamente, permitem que mais de um

[3] Para este momento, vamos enteder "processo" como "programa em execução". Mais adiante, estudaremos com mais detalhes este conceito.

usuário utilize os recursos. Os principais exemplos de sistemas multiusuários são os sistemas operacionais de redes.

3. Monoprocessamento x Multiprocessamento

Quanto à quantidade de processadores envolvidos no processamento os sistemas se caracterizam em monoprocessados (um único processador) e multiprocessados (dois ou mais processadores).

Ultimamente, os sistemas computacionais têm sofrido um contínuo e crescente aumento de demanda por velocidade e desempenho. Assim, existem situações em que um único processador, por mais moderno e eficiente que seja, não consegue dar conta de determinadas aplicações por estas envolverem uma grande quantidade de cálculos, exigindo muito do processador.

Uma das soluções para aumentar o desempenho dos computadores e permitir a execução em tempo satisfatório de aplicações extremamente complexas é a utlização de diversos processadores trabalhando em conjunto.

Os sistemas com multiprocessamento são subdivididos em dois tipos: os sistemas *fortemente acoplados* e os sistemas *fracamente acoplados.*

Sistemas fortemente acoplados - A configuração desses sistemas se caracteriza por diversos processadores que se comunicam através de uma memória compartilhada (*shared memory*), gerenciados por um único sistema operacional, conforme a Figura 1.

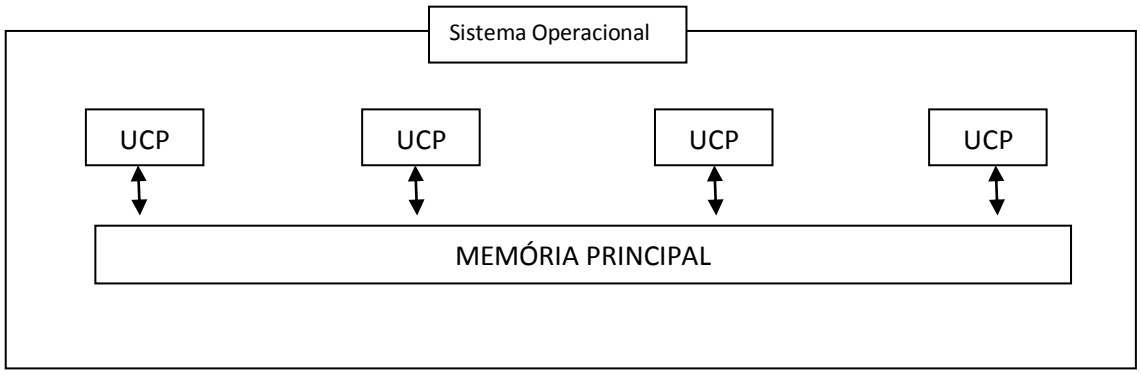

Figura 1 – Sistema fortemente acoplado

Sistemas fracamente acoplados – Esses sistemas se caracterizam pela existência de sistemas computacionais independentes quanto à memória. A comunicação entre as unidades de processamento é eita por linhas de comunicação (*message-passing*) e cada nó do sistema possui seu próprio sistema operacional. A Figura 2 mostra a configuração de um sistema fracamente acoplado.

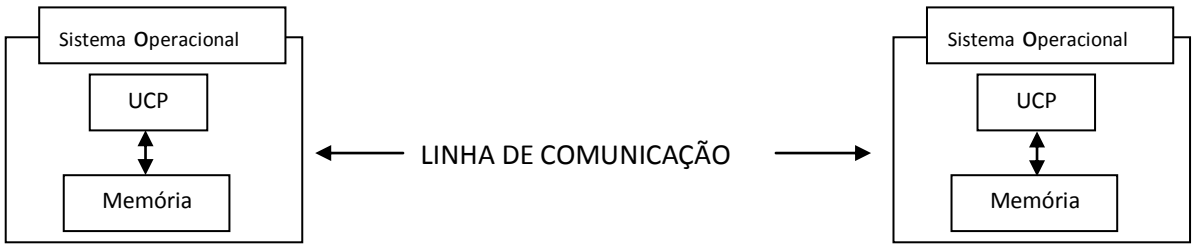

Figura 2 – Sistema Fracamente acoplado

4. Sistemas Interativos x Sistemas Batch

Os primeiros sistemas operacionais a figurar nos computadores se caracterizavam por um processamento sequencial com pouca ou nenhuma participação dos usuários. Nesta configuração, as tarefas eram passadas ao computador que as executava em lotes e na ordem em que eram submetidas. Este tipo de processamento é conhecido como *batch*.

Atualmente, os sistemas operacionais oferecem a possibilidade de interação entre o computador e os usuários. Nos sistemas interativos um dos principais objetivos é um bom tempo de resposta, já que o usuário depende do resultado de suas solicitações para continuar trabalhando.

Parte III – Processos e Threads

Processo - conceituação

Vimos anteriormente que, de forma mais simples, um processo é um programa em execução. Essa definição, embora correta, é apenas superficial. Sempre que ativamos algum programa, o sistema operacional cria uma estrutura na memória para que este programa possa ser executado. A essa estrutura damos o nome de *processo*.

Assim, um processo não é simplesmente um programa em execução. Ele compreende as informações a respeito dessa execução, bem como os recursos que estão alocados ou reservados para essa isso. A partir de agora vamos ver que informações o sistema operacional precisa manter sobre os processos (contexto do processo) e as diferentes situações em que o processo pode se encontrar na memória (estados do processo). Em seguida, abordaremos os threads que, resumidamente, são diferentes fluxos de instruções dentro de um mesmo processo.

Contexto de processos

Como vimos, um processo é a estrutura criada pelo sistema operacional para que um programa possa ser executado corretamente. Esta estrutura é composta por informações a respeito da execução do programa e dos recursos reservados para isso. Ao conjunto de todas essas informações damos o nome de *contexto* do processo.

Esse contexto contém, entre outras, as seguintes informações;

- O estado do processo (pronto, bloqueado ou em execução);
- O conteúdo dos registradores (contexto de hardware), em especial o do *contador de instruções*, que aponta para o endereço da próxima instrução a ser executada;
- Informações a respeito das quotas e privilégios do processo (por exemplo, sua prioridade, a quantidade de arquivos que ele pode manipular, etc);
- O espaço de endereçamento, que é a área de memória que este processo pode utilizar;
- Informações a respeito dos recursos que estão alocados ou reservados para a execução do processo.

Todas as informações acima se localizam no Bloco de Controle do Processo (PCB – *Process Control Block*). Cada processo possui seu PCB, que é sua referência para o sistema operacional.

É importante ressaltar que essas informações são fundamentais para o sistema operacional para que a multiprogramação possa ser implementada com sucesso, já que, com diversos processos na memória concorrendo pelos recursos da máquina, é necessário que o sistema operacional faça um escalonamento dos recursos para atender a todos os processos, segundo os critérios que estudaremos mais adiante.

Estados de um processo

Como vimos, a multiprogramação permite que vários processos estejam na memória simultaneamente, concorrendo pelos recursos da máquina. Por isso, tais processos podem se encontrar em situações distintas. Essas situações são chamadas *estados do processo* e são as seguintes:

1) Execução – um processo se encontra em execução quando suas instruções estão sendo tratadas pelo processador. Essa situação pode demorar durante algum tempo pré-definido, chamado *time-slice (fatia de tempo)* ou *quantum,* ou indefinidamente, até que o processo encerre sua execução. O time-slice é o tempo que o processo tem para usar o processador. Ao final deste, ele retorna para uma fila onde aguarda uma nova sessão de processamento. Como veremos adiante, o sistema operacional pode dar uma fatia de tempo fixa para que cada processo utilize o processador, utilizando o conceito de

preempção (algoritmos de escalonamento preemptivos) ou pode permitir que um processo utilize o processador por tempo indefinido até que termine todas as suas operações (algoritmos de escalonamento não preemptivos)

2) Pronto – um processo está neste estado quando reúne todas as condições de entrar em execução e apenas aguarda sua vez de fazer uso do processador. Como podemos ter vários processos nessa situação, o sistema operacional organiza a *fila de prontos*. Esta fila pode ser ordenada de acordo com a ordem de chegada dos processos ou de acordo com suas prioridades.
3) Bloqueado ou Em Espera – Um processo se encontra bloqueado quando aguarda algum evento ou recurso do qual depende para entrar em execução. Por exemplo, se ele aguarda a busca de informações no disco rígido ou a liberação de um recurso que está em poder de outro processo (como uma variável compartilhada) ele se encontra bloqueado até que tal pendência seja resolvida.

A alteração do estado de um processo depende de eventos específicos. Vamos agora verificar como isso acontece.

Mudança de Estados de um processo

Vamos ver agora como acontecem as mudanças de estado de um processo. Observe a Figura 3 abaixo.

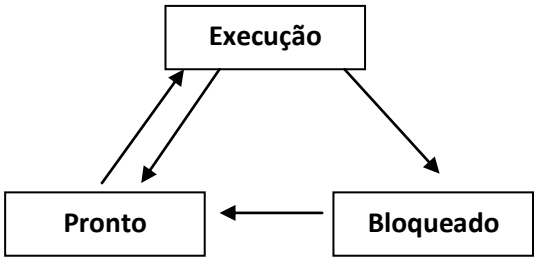

Figura 3 – estados de processo

Podemos identificar quatro mudanças de estado possíveis:

1) Execução → Pronto – Esta mudança ocorre quando o tempo reservado para que o processo utilize o processador termina. É necessário então que a execução seja interrompida e o processo seja encaminhado para a fila de prontos, onde aguardará sua próxima sessão de execução.
2) Pronto → Execução – Esta mudança ocorre quando o processo se encontra na fila de prontos e chega sua vez de utilizar o processador. Ocorre então a restauração do seu contexto e a retomada da execução do ponto em que foi interrompido.
3) Execução → Bloqueado – Quando o programa está em execução e detecta a necessidade de algum recurso que não está disponível, ele deve deixar o processador para que este possa atender a outros processos e aguardar até que a pendência que o impede de continuar executando seja resolvida.
4) Bloqueado → Pronto – Quando a pendência citada na mudança anterior é solucionada, o processo deve retornar à fila de prontos onde aguarda por uma chamada do sistema operacional para que entre em execução novamente.

É importante observar que a mudança de estados de bloqueado diretamente para execução não ocorre porque é necessário que sejam avaliadas as prioridades em relação aos processos que aguardam na fila de prontos. Também não ocorre a mudança de pronto para bloqueado porque não há como surgir uma pendência em um processo que não está executando.

Threads

Para entendermos o que são as threads, é necessário verificar a estrutura dos programas. De forma simples, um programa é uma sequência de instruções que são executadas em alguma ordem que depende dos valores de entrada. Os programas são divididos em tarefas que eventualmente são independentes e, portanto, podem ser executadas simultaneamente sem prejuízo do resultado final do programa.

Assim, os threads são fluxos independentes de instruções de um processo, que podem compartilhar os recursos alocados a este. Assim, um mesmo processo pode ter diversas atividades sendo executadas simultaneamente, melhorando sensivelmente o desempenho do sistema.

Parte IV – Interrupções e Exceções

1 – Interrupções

Os sistemas multiprogramáveis, como vimos, permitem que vários processos[4] estejam na memória, concorrendo pelos recursos do computador. Além disso, o próprio sistema operacional tem seus processos que também utilizam esses recursos. Desta forma, o processador pode ser requisitado para atender a diversas solicitações, tanto por parte dos processos que estão na memória quanto por parte de componentes do hardware, tais como as controladoras de dispositivos de entrada e saída.

Para que o processador atenda a esses inúmeros pedidos provenientes de diversos processos ou dispositivos, é necessário um mecanismo que permita a comunicação entre as controladoras e o processador. Isto é feito através das interrupções, que causam a suspensão temporária do fluxo do programa que estava em execução.

Tipos de interrupção

As interrupções podem ser causadas por controladoras de dispositivos de entrada e saída (hardware) ou pelo próprio sistema operacional (software). Em ambos os casos, é necessária uma sequência de operações que veremos mais adiante.

Existem algumas situações em que as interrupções, que são causadas por eventos externos ao processador, podem se acumular ou interromper a execução de um trecho de programa que não pode ser interrompido. Neste caso, para evitar erros no processamento, o controlador de interrupções pode inibir ou adiar o tratamento destas. Essas interrupções que podem ser adiadas ou inibidas são chamadas *interrupções mascaráveis*.

Tratamento de interrupções

O tratamento de interrupções diz respeito ao que deve ser feito pelo sistema operacional quando ocorre uma interrupção. Vamos agora entender como funciona esse tratamento.

Quando um processo está em execução, ou seja, tendo suas instruções tratadas pelo processador, ele altera constantemente os valores das variáveis que manipula. Desta forma, se ocorrer uma interrupção, esses valores precisam ser guardados em algum lugar para que não sejam perdidos. Além disso, também é importante registrar em que ponto do programa ocorreu a suspensão da execução. Ao final do tratamento da interrupção, o processo que foi interrompido deve ser retomado exatamente de onde parou com os mesmos valores que possuía naquele instante.

Assim, o tratamento de uma interrupção compreende os seguintes passos:

1) Salvar (guardar) os valores do processo interrompido que se encontram nos registradores do processador. O conjunto desses valores é parte do *contexto* do processo. Esse contexto será estudado mais adiante em detalhes;
2) Identificar o tipo de interrupção;
3) Buscar a rotina de tratamento da respectiva interrupção;
4) Executar tal rotina;
5) Restaurar os valores que foram guardados de forma que o processo que foi interrompido possa ter sua execução retomada sem perda de informações.

Os passos que envolvem a identificação e a busca das rotinas de tratamento de interrupções utilizam uma estrutura chamada *vetor de interrupções*, que contém os identificadores das interrupções juntamente com os endereços iniciais das suas respectivas rotinas de tratamento.

[4] Podemos entender o termo "processo", provisoriamente como um programa em execução. Porém, essa definição é bastante superficial já que um processo na verdade envolve diversos conceitos que iremos estudar mais adiante.

2 – Exceções

Vimos que as interrupções causam a parada temporária do programa que está em execução para que o processador possa tratar o motivo de tal evento.

Porém, existe outro evento que provoca a parada/suspensão do programa que está em execução, chamado *exceção*, que se caracteriza por ter sua origem devida à própria execução do programa como, por exemplo, um *overflow* ou erro de programação.

Por ser a combinação de algumas condições que devem ocorrer, a exceção é um evento *síncrono*, já que ocorrerá sempre que tais condições estiverem presentes.

DMA – Direct Memory Access

Até agora vimos que a comunicação entre o processador e os dispositivos de entrada/saída é feita pelo mecanismo de interrupções. Assim, sempre que a controladora de um destes dispositivos presisa entregar ou solicitar algo do processador, ela provoca uma interrupção, que suspende o programa em execução e faz com que o processador tenha que executar a respectiva rotina de tratamento. A ocorrência de várias interrupções provoca uma sobrecarga no processador, o que afeta o desempenho do sistema como um todo.

Para desafogar o processador das inúmeras tarefas que precisa executar foi criada a técnica de DMA (Direct Memory Access – Acesso Direto à Memória). Através dela, a transferência de dados entre a memória e as controladoras passou a ser feita sem a participação do processador, exceto no início e no fim desta transferência.

O funcionamento do DMA é conceitualmente simples. Quando há a necessidade de transferir dados entre a memória principal e os *buffers* das controladoras, o processador envia a solicitação para estas, informando as posições de origem e destino das informações. Em seguida, a própria controladora realiza a transferência (daí o acesso direto a memória) e, ao final, provoca uma interrupção no processador para informar a conclusão da tarefa. A grande vantagem desta tecnologia é que o processador não mais precisa se encarregar destas transferências, o que significava um desperdício de tempo.