Terceira aula de FSO

José A. Cardoso e Cunha
DI-FCT/UNL

Este texto resume o conteúdo da aula teórica.

1 Objectivo

O objectivo da aula foi a continuação da apresentação de alguns conceitos dos sistemas de operação, ligados à evolução histórica dos sistemas de computadores.

2 Parâmetros de eficiência do sistema

Nos sistemas de processamento em *batch*, sugem três parâmetros principais a considerar, importantes do ponto de vista da 'eficiência' do sistema:

Utilização do processador: é a relação entre o *Tempo de utilização do CPU* e o tempo total decorrido, num certo intervalo de observação

$$\frac{\texttt{TempoCPU}}{\texttt{TempoTotal}}$$

Débito dos trabalhos: é o número de trabalhos (veja a segunda aula teórica) executados por unidade de tempo.

Tempo de resposta ao utilizador: é o tempo que medeia entre a submissão de um trabalho por um utilizador e a obtenção dos resultados da execução desse pedido.

A figura 1 ilustra, esquematicamente, as relações de grandeza típicas entre as acções do operador humano e a execução de instruções.

Os intervalos 'úteis' são, como se ilustra, muito reduzidos, em termos relativos, em comparação com as tarefas de inicialização e operação manual

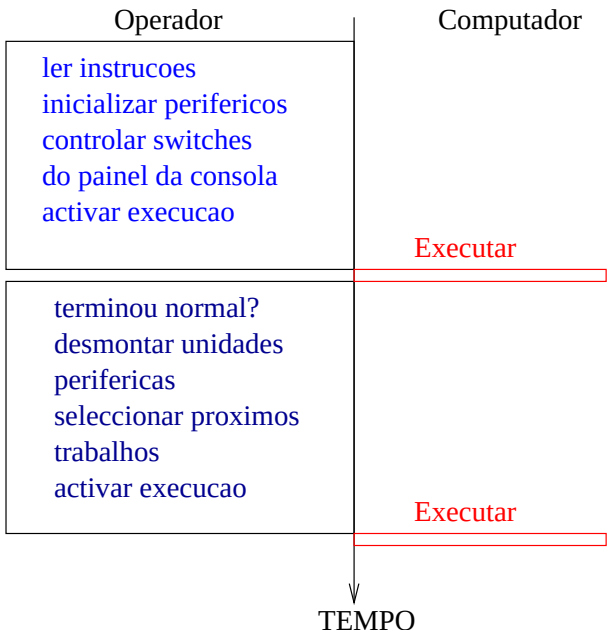

Figura 1: Relação de tempos

do sistema. Em consequência, os sistemas de processamento em *batch* tinham valores baixos para a utilização do processador.

Isto era agravado pelo facto de a execução ser estritamente sequencial. Ainda que este aspecto não seja mostrado na figura, durante o intervalo total de execução de um programa, nem todo o tempo é útil, isto é, de processamento efectivo, pois o programa efectua operações de entrada e saída, que obrigam o processador a esperar pelos periféricos lentos.

Assim, tem-se, em geral:

```
Tempo total de execução = Tempo de CPU + Tempo de Espera
```

A segunda parcela pode ser, em geral, muito superior à primeira, originando tempos de desocupação do CPU. Se este é o único programa no sistema, nada há a fazer, durante esses tempos e a utilização do CPU é baixa.

Para conseguirem um débito de trabalhos mais razoável (quando comparado com os anteriores sistemas de máquina dedicada, sem operador) estes sistemas tinham de agregar múltiplos trabalhos num lote, para assim beneficiarem de alguma automatização de tarefas suportada pelo monitor de controlo de trabalhos.

O tempo de resposta ao utilizador era da ordem de dias ou horas, no melhor dos casos. O utilizador não beneficiava deste regime pois, por exemplo, o mais pequeno erro num passo de compilação, produzia a terminação do trabalho de um utilizador, que só seria informado no dia seguinte... para ter possivelmente de fazer uma pequena correcção e voltar a submeter o trabalho. A produtividade no desenvolvimento de programas era, assim, muito baixa, aspecto este que viria a ser modificado radicalmente com o aparecimento dos sistemas interactivos (adiante).

A evolução dos primeiros SO, nas décadas de 1050 e 60 foi profundamente marcada pelas tentativas de melhorar os três parâmetros indicados. Repare-se que estes parâmetros reflectem duas perspectivas distintas, mas ambas importantes: (i) por um lado, a tentativa de melhorar o tempo de resposta garantido ao utilizador, para melhorarar a sua produtividade e, por outro lado, (ii) as tentativas de aumentar o débito de trabalhos e a utilização do processador, assim melhorando o rendimento de utilização dos recursos do computador. Nos primeiros sistemas de computadores, o tempo de utilização do sistema era pago pelo utilizador, portanto um maior débito de trabalhos significava mais dinheiro a entrar...

3 Uma tentativa para melhorar a eficiência

A figura 2 ilustra a arquitectura de um sistema com dois computadores: um deles é dedicado exclusivamente às operações de entrada e de saída, enquanto o outro fica apenas ocupado com a execução de lotes de trabalhos. Os dois computadores operam concorrentemente, ou melhor dizendo, mesmo em paralelo. O computador 1 vai lendo, do leitor de cartões, as informações de controlo e de dados, dos trabalhos submetidos pelos utilizadores, e vai-as escrevendo numa banda magnética B1. Uma vez completada, esta banda é desmontada da unidade do computador 1 e montada na unidade de banda do computador 2. Este computador pode, então, começar o processamento dos trabalhos descritos no lote agora contido na banda B1, num funcionamento idêntico ao de um sistema de processamento em *batch*. Os resultados do processamento deste lote são escritos, pelo computador 2, numa banda B2, que, uma vez cheia, vai ser montada no computador 1, para este produzir as listagens dos trabalhos.

Este sistema, designado por 'tratamento de trabalhos em lote, com entradas e saídas desligadas' (*input/output offline*), tinha vantagens em relação ao sistema anterior, por diminuir os tempos de leitura dos trabalhos e seus dados, bem como os tempos de escrita dos resultados de cada trabalho.

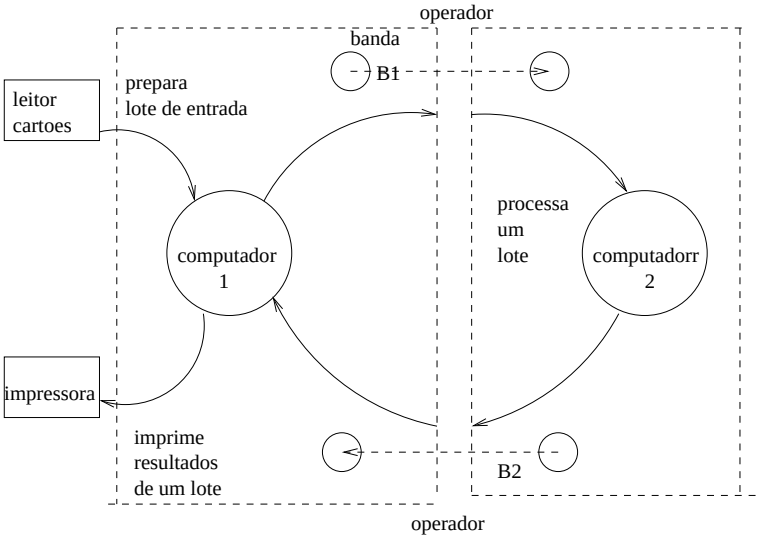

Figura 2: Concorrência na preparação dos lotes

Tratava-se de uma melhoria quantitativa: o programa, em execução no computador 2, acede a uma unidade de banda magnética, em vez de a um leitor de cartões e a uma impressora. Esta rapidez foi obtida à custa de um computador adicional, o computador 1, que, no entanto, apenas precisava de ser eficiente nas operações de entrada e saída, não lhe sendo exigida grande rapidez de processamento quanto à execução das instruções. O computador 1 podia, assim, ser mais barato, do que o computador 2.

A viabilidade desta solução foi comprovada na prática, em sistemas construídos e comercializados pela IBM.

Esta solução mantém, contudo, diversas limitações:

1. tempos de desocupação do CPU do computador 2, durante os tempos de espera de dados da banda B1 ou de possibilidade de escrita na banda B2;

2. operações manuais de montagem e desmontagem das bandas nas respectivas unidades;

3. execução dos trabalhos de um lote (isto é, gravados na banda B1), numa ordem estritamente sequencial, conforme a ordem segundo a qual tenham sido gravados.

A primeira limitação resulta de este ser um regime de *monoprogramação*. O sistema só activa a execução de um programa, quando tiver completado a execução do programa anterior do lote. Sendo assim, se um programa esperar por dados ou pela impressão dos resultados, é como se todo o lote ficasse bloqueado.

A segunda limitação aconselhava a que os lotes fossem de dimensão razoável, isto é, 'quanto mais trabalhos num lote, melhor', pois assim reduzia-se a sobrecarga percentual das tarefas manuais de montagem / desmontagem de bandas pelo operador humano. Por exemplo, se este tempo fosse, de 10 minutos para instalar B1 e desinstalar B2 (incluindo rebobinar as fitas magnéticas) e um lote tivesse 10 trabalhos, obtinha-se uma sobrecarga média por trabalho, de 1 minuto. Aumentando o número de trabalhos por lote, para 100, e assumindo que o tempo de instalação era pouco alterado, a sobrecarga média por trabalho descia para 6 segundos!

No entanto, aumentar muito o número de trabalhos de um lote tinha um efeito indesejável, relacionado com a terceira limitação acima apontada. Como os resultados de um lote só ficavam disponíveis depois de a banda B2 ter sido completada pelo computador 1, instalada no computador 1 e impressa na impressora, todos os trabalhos de um mesmo lote tinham aproximadamente o mesmo tempo de resposta e, quanto mais trabalhos num lote, maior seria esse tempo. Uma consequência indesejável disto era que um trabalho de curta duração, num dado lote, seria prejudicado pela presença de outros trabalhos, de grande duração, no mesmo lote.

Pode perguntar-se: mas por que motivo não eram os trabalhos executados por uma outra ordem, mais justa, por exemplo, para os trabalhos mais curtos? Na figura 3, ilustra-se, do ponto de vista lógico, a estrutura de uma banda magnética.

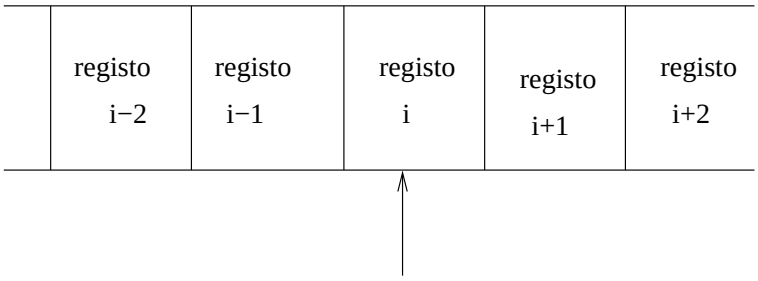

Figura 3: Banda magnética

O acesso é sequencial, só havendo acesso, num dado momento, ao registo

da banda i que esteja sob a cabeça de leitura/escrita da unidade de controlo da banda. Para aceder ao registo k da banda, estando a cabeça posicionada sobre o registo i, há que rebobinar (*rewind*) a fita através das posições intermédias de i até k. O tempo desta operação é proporcional àquela distância e pode ser apreciável, para bandas de grande capacidade.

Uma solução provisória para este problema foi a de permitir que o operador da instalação, ordenasse previamente os trabalhos submetidos em cartões, pelos utilizadores, submetendo-os ao computador 2, de modo a agrupar num mesmo lote, todos os trabalhos mais curtos, e num outro lote, todos os mais longos. Isto correspondia a delegar no operador a decisão de qual a ordem pela qual os trabalhos eram escalonados (agendados para execução), possivelmente com base em informação que o utilizador lhe dava, através de um cartão de controlo (onde indicava a duração típica esperada (!) dos seus trabalhos).

4 Periféricos mais rápidos e de acesso directo

O aparecimento das unidades de discos magnéticos viria a ajudar a resolver, de modo significativo, as limitações anteriormente apontadas, nos sistemas de tratamento de lotes de trabalhos.

Para além de uma maior rapidez no acesso (os tempos envolvidos passavam da ordem dos minutos, no caso das bandas, para a ordem das fracções de segundo - milisegundos, no caso dos discos), o disco permitia acesso directo aos registos nele gravados. Na figura 4, ilustra-se, do ponto de vista lógico, a estrutura de um disco. O disco está organizado num certo número de pistas concêntricas, cada uma das quais dividida num mesmo número de sectores. Os sectores têm um tamanho de, por exemplo, 1K ou 2K ou 4K bytes.

O acesso faz-se em três tempos:

1. tempo de pesquisa: especifica-se um número de pista e espera-se que o braço do disco se mova e seleccione essa pista;

2. tempo de rotação: espera-se que o disco rode até que o sector desejado, de número i, da pista seleccionada, passe sob a cabeça de leitura/escrita do braço; este tempo depende da velocidade de rotação do disco;

3. tempo de transferência: desencadeia-se a transferência, habitualmente por DMA, do sector indicado, para/de a memória central do computador; este tempo depende da rapidez de acesso ao *bus* do computador.

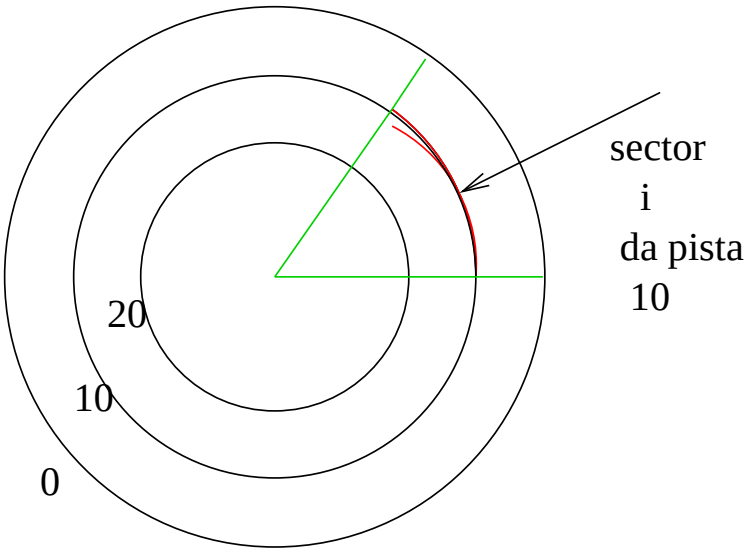

Figura 4: Disco

Ao permitir o acesso directo, através da indicação de um par (pista, sector), o disco possibilitou disciplinas de escalonamento dos trabalhos mais flexíveis, por exemplo, baseadas em prioridades dos trabalhos, ajudando assim a resolver a limitação apontada na execução dos trabalhos mais curtos.

5 Controlo das entradas e saídas com maior autonomia

Com o aparecimento gradual de diversos tipos de periféricos (leitores de fita e de cartões perfurados, impressoras, unidades de banda magnética, unidades de disco), a quantidade de tarefas de controlo, da responsabilidade do SO, foi aumentando. Note-se, no entanto, que o mecanismo de interrupções de programa possibilitou, desde muito cedo, o processamento concorrente de múltiplos periféricos. À medida que os sistemas de computadores, na década de 1960, iam encontrando maior desenvolvimento e aplicação em domínios que exigiam grande diversidade de periféricos e grande capacidade de armazenamento de dados, convinha libertar o CPU do maior número de tarefas possível.

A IBM propôs, então, a seguinte arquitectura, ilustrada na figura 5

Um *canal*, na terminologia então proposta pela IBM, era um processador dedicado às entradas e saídas, que podemos considerar uma generalização

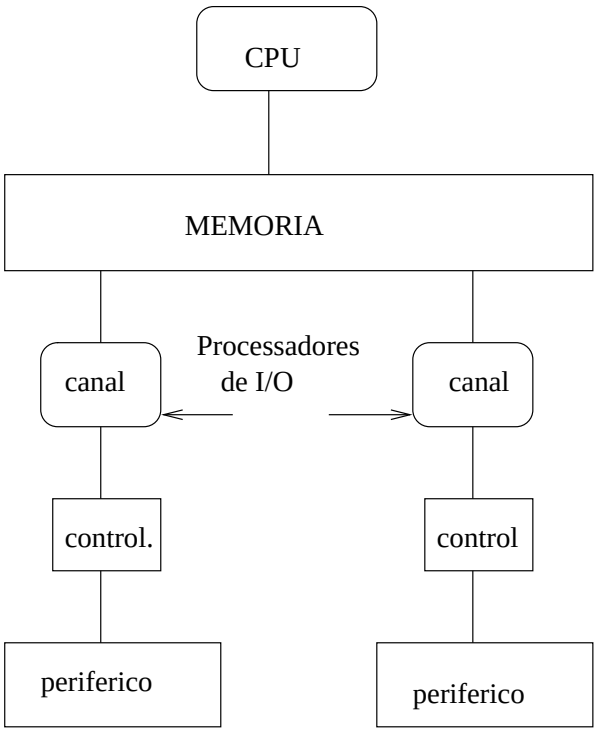

Figura 5: Processadores dedicados às entradas/saídas

do conceito de 'acesso directo à memória'. Tal processador dispunha de uma unidade de controlo, capaz de interpretar um programa, dito *programa de input/output*, previamente carregado em memória. Como se vê na figura, os processadores de canal e o CPU do computador, partilhavam o acesso à memória central.

O processador de canal executava sequências de comandos, cada um doss quais correspondia a uma instrução de um programa de input/output. Por exemplo, podia executar o comando 'posicionar cabeça de disco numa pista e sector indicados', seguido de um comando 'transferir bloco de bytes do disco para um endereço indicado de memória', e assim sucessivamente, até atingir um comando indicando fim de programa. Cada processador de canal podia assinalar o seu estado ao CPU, bem como a ocorrência de erros, através de *flags* de estado.

A vantagem deste conceito era libertar o processador central de uma boa parte das tarefas de controlo dos periféricos, em casos de computadores com

muitos periféricos, como os que começam a aparecer em grandes empresas com muitos arquivos de dados.

6 Sistemas de SPOOL

Com o aparecimento dos discos, a arquitectura de um sistema de tratamento de lotes de trabalhos passou a ser a ilustrada na figura 6

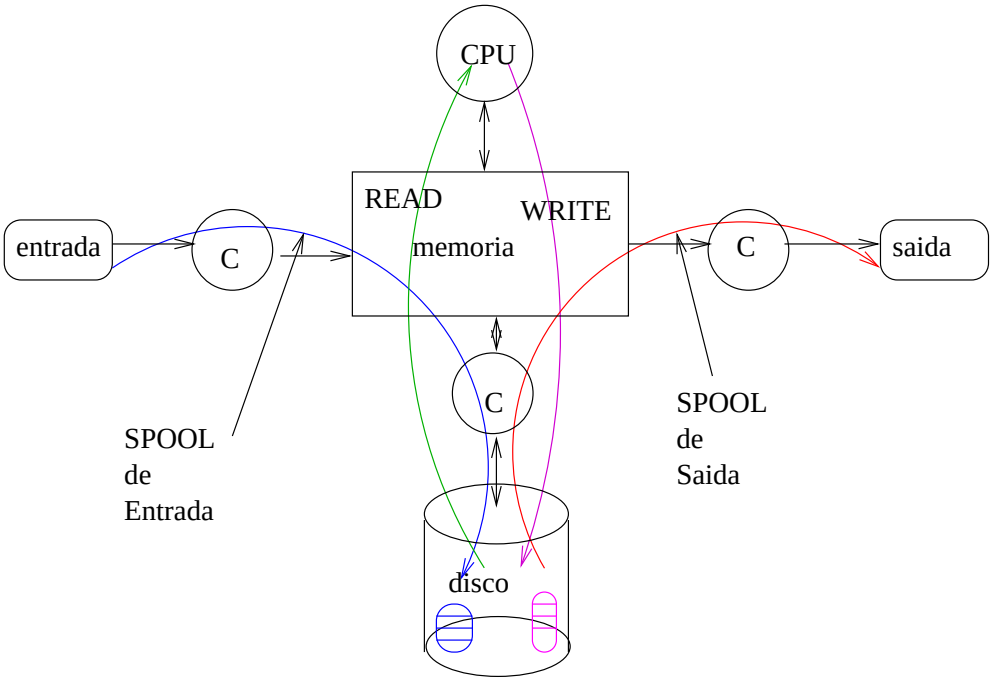

Figura 6: SPOOL

Na figura, ilustram-se três periféricos, um de entrada (por exemplo, leitor de cartões ou de banda magnética), um de saída (por exemplo, uma impressora) e um de entrada/saída (um disco). Cada periférico é controlado por um dispositivo de controlo (C, na figura) que pode ser, no caso do disco, um processador de canal, ou simplesmente um controlador de DMA.

Este sistema processa as seguintes tarefas, de forma concorrente:

- *Monitor do SO*: selecciona o próximo trabalho para execução, a partir de uma fila de entrada, que contém os lotes de trabalhos submetidos;

esta fila encontra-se em disco e contém, para além do programa de cada trabalho, também os respectivos dados (o sistema não é interactivo);

- *Leitor ou SPOOL de entrada*: controla a leitura e a transferência de programas e dados dos trabalhos submetidos, no leitor de cartões, armazenando-os numa fila de entrada em disco;

- *Escritor ou SPOOL de saída*: controla a escrita e a transferência dos resultados dos trabalhos executados, do disco para a impressora;

- *Programa utilizador*: programa em execução, sujeito a interrupções originadas pelos periféricos mencionados, para processamento das tarefas acima indicadas.

A designação destes sistemas - **S***imultaneous* **P***eripheral* **O***peration* **O***nline* - resulta do facto dos periféricos, nomeadamente a unidade de disco, estarem permanentemente ligados, 'em linha' (*online*, durante a operação do sistema, que suporta multiplas actividades de entrada e saída em simultâneo.

A actividade de SPOOL de entrada permite armazenar os trabalhos submetidos, num disco, com a vantagem de mais rápido acesso e de maior flexibilidade no escalonamento de trabalhos para execução, devido ao modo de acesso directo ao disco (quando comparado com dispositivos de acesso exclusivamente sequencial, como as bandas magnéticas). Quando o programa utilizador. uma vez carregado em memória, executa uma chamada ao SO, para uma operação READ, como se fosse ler um cartão de dados. o sistema encaminha o pedido para os ficheiros em disco, nos quais foram previamente carregados os dados do cartão pedido. Assim, o programa não se apercebe que o periférico ao qual acede é afinal, um 'leitor de cartões' virtual, suportado pelo disco.

A actividade de SPOOL de saída permite ir transferindo dados, de forma regular (desde que haja resultados para imprimir), de disco para a impressora. Quando o programa utilizador invoca uma operação WRITE ou PRINT, afinal não vai aceder directamente à impressora, mas sim a uma fila em disco, onde os resultados produzidos pelo programa vão ficar temporariamente armazenados, até que o SPOOL de saída desencadeie a sua impressão. Do ponto de vista do programa, o disco suporta uma espécie de 'impressora virtual', com a vantagem de que o programa não tem de ficar à espera que a impressão se desencadeie, isto é, pode prosseguir a execução. Do ponto de vista do SO, o acesso à impressora fica sob a sua inteira responsabilidade, o que lhe permite escolher a ordem mais conveniente para a impressão dos

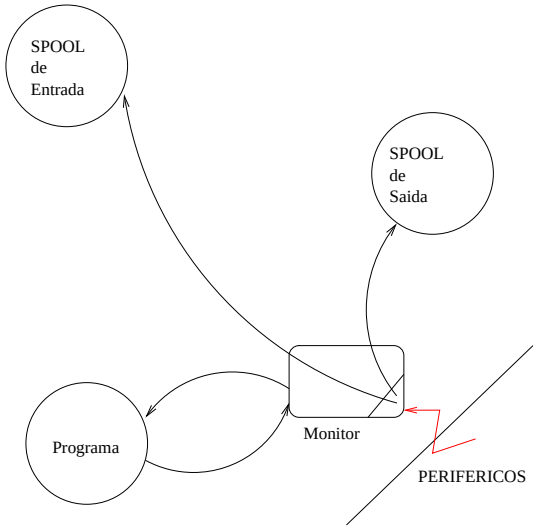

Figura 7: Coordenação das actividades de SPOOL

trabalhos (eg, imprimir em primeiro lugar os resultados dos trabalhos mais curtos).

A figura 7 ilustra de que modo o monitor coordena as actividades concorrentes acima descritas, com base no mecanismo de interrupções.

Tanto o SPOOL de entrada, como o de saída, controlam as transferência, respectivamente, do leitor para disco e do disco para a impressora, utilizando, evidentemente, *buffers* intermédios em memória central. O programa utilizador, uma vez activado, vê-se interrompido, sempre que ocorra um pedido de interrupção de um dos periféricos que, conforme o caso, irá reactivar um dos processos de SPOOL, o de entrada ou o de saída. Após tratado o pedido, a execução do programa utilizador será retomada.

Características dos sistemas de SPOOL. Os sistemas de SPOOL trouxeram melhorias significativas aos sistemas de tratamento de lotes de trabalhos, em diversos aspectos:

- maior eficiência, com melhorias nos parâmetros de utilização dos periféricos e do processador, e no débito de execução dos trabalhos; isto foi conseguido através da execução concorrente das diversas tarefas de entrada e de saída; para este aspecto, foi essencial dispor-se do mecanismo de interrupções de programa, bem como de controladores de periféricos dotados de uma

certa autonomia (abrangendo desde os mais simples controladores de DMA até aos mais sofisticados processadores de canal da IBM);

- maior flexibilidade nos algoritmos de escalonamento do SO, que escolhem o próximo trabalho para execução; isto foi possibilitado pelo modo de acesso directo a disco;

Contudo, estes sistemas apresentavam ainda algumas limitações:

1. sendo os ttabalhos executados em regime de *monoprogramação*, quando o programa utilizador invocava READ e não havia dados disponíveis em *buffers* em memória, o CPU ficava inactivo, pois não havia outros programas para executar, enquanto aquele programa aguardava os seus dados; eventualmente, a única coisa que o sistema poderia fazer, durante esse intervalo, seria tratar pedidos de interrupções que, entretanto, os periféricos gerassem;

2. sendo um regime de tratamento de trabalhos em lote (*batch*), não intaractivo, o tempo de resposta de cada trabalho continuava a intrinsecamente a depender do perfil de trabalhos de um lote, isto é, mesmo um trabalho mais curto teria de aguardar o fim completo do lote; no entanto, com a disposição das filas dos trabalhos em disco, tornava-se possível realizar outras estratégias, nomeadamente, dar prioridade à execução de trabalhos mais curtos, bem como à impressão dos seus resultados.

As duas limitações apontadas, uma devida ao regime de monoprogramação, outra devida ao regime de processamento não interactivo em lotes de trabalhos, iriam ser ultrapassadas, respectivamente, pelos sistemas de *multiprogramação* e pelos *sistemas interactivos*, como veremos na próxima aula.