



**CONCEITOS BÁSICOS DE O.S.
(PARA O SISTEMA IBM/360-370)**

Roberto Barsotti

**CONCEITOS BÁSICOS DE O.S.
(PARA O SISTEMA IBM/360-370)**

Roberto Barsotti

**CENTRO DE PROCESSAMENTO DE DADOS
(CPD)**

**INSTITUTO DE ENERGIA ATÔMICA
SÃO PAULO – BRASIL**

CONSELHO DELIBERATIVO

MEMBROS

Klaus Reinach — Presidente
Roberto D'Utra Vaz
Helcio Modesto da Costa
Ivano Humbert Marchesi
Admar Cervellini

PARTICIPANTES

Regina Elisabete Azevedo Beretta
Flávio Gori

SUPERINTENDENTE

Rômulo Ribeiro Pieroni

INSTITUTO DE ENERGIA ATÔMICA
Caixa Postal 11.049 (Pinheiros)
Cidade Universitária "Armando de Salles Oliveira"
SÃO PAULO — BRASIL

ÍNDICE

	Página
Prefácio	01
1 Introdução ao O.S.	01
2 Organização do O.S.	05
2.1 Programas de controle	05
2.1.1 JOB MANAGEMENT	06
2.1.2 TASK MANAGEMENT	09
2.1.3 DATA MANAGEMENT	11
2.2 Programas de processamento	12
2.2.1 Compiladores, tradutores ou montadores	12
2.2.2 Programas de serviço	13
2.2.2.1 LINKAGE EDITOR	13
2.2.2.2 LOADER	17
2.2.2.3 SORT / MERGE	18
2.2.2.4 UTILITÁRIOS	19
2.2.3 Programas do usuário	21
3 Opções do programa de controle	23
3.1 P.C.P. — Primary Control Program	23
3.2 Multiprogramação	23
3.2.1 M.F.T. — Multiprogramming with a fixed number of Tasks	26
3.2.2 M.V.T. — Multiprogramming with a variable number of Tasks	29
4 Características do O.S.	32
5 Conceitos de DATA MANAGEMENT	35
5.1 Formatos de registros	35
5.1.1 Registros de formato fixo	37

5.1.2 Registos de formato variável	38
5.1.2.1 Registos de formato variável spanned	39
5.1.3 Registos de formato indefinido	39
5.2 Organização de data sets	40
5.2.1 Organização seqüencial	41
5.2.2 Organização seqüencial indexada	41
5.2.2.1 Área primária	42
5.2.2.2 Área de índices	42
5.2.2.3 Área de overflow	43
5.2.3 Organização direta	48
5.2.3.1 Endereçamento direto	48
5.2.3.2 Endereçamento indireto	49
5.2.4 Organização particionada	51
5.3 Técnicas de acesso	52
5.3.1 Técnica QUEUED	53
5.3.2 Técnica BASIC	53
5.4 Métodos de acesso	54
5.5 Labels	55
5.5.1 Tape labels	55
5.5.2 Dasd labels	57
5.6 Data sets do sistema	57
5.6.1 Data sets obrigatórios	58
5.6.2 Data sets opcionais	58
6 Controle do sistema	59
6.1 Introdução	59
6.2 JOB CONTROL LANGUAGE	61
6.2.1 Cartões de J.C.L.	61
6.2.2 Campos no cartão de J.C.L.	62
6.2.3 Regras para continuar um cartão de J.C.L.	63
6.2.4 Tipos de parâmetros no J.C.L.	63
6.2.5 Uso de parênteses e apóstrofes em J.C.L.	63
6.2.6 Procedimentos catalogados	64
6.2.7 Bibliotecas particulares	64
6.2.8 Data sets concatenados	65
7 Noções de teleprocessamento	66
7.1 Elementos básicos de um sistema de T.P.	66
7.2 Principais aplicações de teleprocessamento	67

8 Algumas considerações sobre o /370	68
9 Anexos	70
9.1 Tabela de aplicações usuais dos utilitários	70
9.2 Comparação entre linguagens quanto a facilidades de DATA MANAGEMENT	72
9.3 Tipos de sistemas operacionais pra computadores IBM/360	73
9.4 Repertório de algumas siglas e acrônimos mais usados em computação	74
ABSTRACT	81
REFERÊNCIAS BIBLIOGRÁFICAS	81

CONCEITOS BÁSICOS DE O.S. (PARA O SISTEMA IBM/360-370)

Roberto Barsotti*

RESUMO

São apresentados conceitos básicos do Sistema Operacional O.S. com intuito de fornecer aos leitores uma visão geral do potencial e das características do referido Sistema Operacional.

Estas características são apresentadas de forma sucinta mas suficiente para compreender-lhe a eficiência e o uso.

PREFÁCIO

Esta publicação é o resultado de um curso interno que tivemos oportunidade de desenvolver, no período de setembro-outubro de 75, para alguns programadores do nosso C.P.D.

É extremamente difícil, senão impossível, escrever sobre O.S., de forma introdutória, sem repetir o que já foi escrito em outras publicações. Querer escrever algo diferente implicaria em descobrir ou inventar algo novo no O.S. que ninguém ainda tivesse descoberto ou inventado. Ora, isso é impossível. O que varia entre as inúmeras introduções ao O.S. existentes, é a abordagem do assunto e o maior ou menor detalhamento de uma ou de outra parte, dependendo do interesse específico da publicação ou de seu autor. Examinada sob este prisma a presente publicação se justifica porque, mesmo sem acrescentar nada de novo ao O.S., ela é resultado de um curso estruturado de forma a, pelo menos tentar, apresentar o O.S. conforme as necessidades dos programadores que frequentaram o curso, devendo considerar as limitações do autor que foi quem ministrou o citado curso.

Outro ponto que justifica a existência destas notas é a necessidade de publicações introdutórias ao O.S., para os novos integrantes do nosso C.P.D., muitos dos quais entram para este Centro sem conhecimento algum do assunto, recebendo aqui seu treinamento.

Sempre visando os novos integrantes com pouca ou nenhuma experiência em computação, foi anexada no final da publicação uma série de anexos que nós consideramos úteis para esta fase de treinamento e, em alguns casos, mesmo para fases mais adiantadas.

Para finalizar agradeceríamos toda e qualquer sugestão que viesse melhorar ou completar estas notas tornando-as mais úteis.

1 – INTRODUÇÃO

A evolução da tecnologia de computadores faz com que o hardware se torne cada vez mais completo e complexo para conseguir resultados cada vez mais satisfatórios. Isto torna extremamente difícil e demorada a operação do computador acarretando um uso ineficiente da U.C.P. (Unidade Central de Processamento) e dos periféricos.

* Analista de Sistemas — Centro de Processamento de Dados; Instituto de Energia Atômica, São Paulo, SP.

O sistema operacional visa eliminar estes problemas conferindo ao computador a possibilidade de se auto controlar dependendo o menos possível da intervenção humana.

O O.S. — OPERATING SYSTEM — é um sistema operacional constituído por um conjunto de programas fornecidos e mantidos pelo fabricante — no caso a IBM — e que tem por finalidade básica obter o melhor rendimento possível da máquina. Em outras palavras: são recursos de Software destinados a obter uma utilização máxima dos recursos de Hardware.

Do conjunto de programas que constituem o Sistema operacional completo, o usuário seleciona aqueles de que necessita baseando o critério de seleção nas necessidades reais do seu C.P.D. e na configuração disponível.

A qualquer momento é sempre possível ao usuário ampliar o número de programas que constituem o seu O.S. e esta ampliação tanto pode advir de necessidades de trabalho a ser desenvolvido pelo C.P.D., como do aumento da configuração. É interessante notar que, devido à filosofia modular com que são concebidos o hardware e o software, a instalação de um aumento da configuração ou de um enriquecimento do Sistema Operacional não chegam a acarretar transtornos mais sérios.

Independentemente de fabricante e da configuração, os componentes básicos do hardware de um computador são:

- Memória Principal
- Unidade de cálculo aritmético e lógico
- Unidade de controle de processamento
- Canais de entrada e saída
- Unidade de controle dos dispositivos de entrada e saída
- Dispositivos de entrada e saída

Além desses, podemos acrescentar mais um componente que podemos chamar genericamente de Memórias Auxiliares.

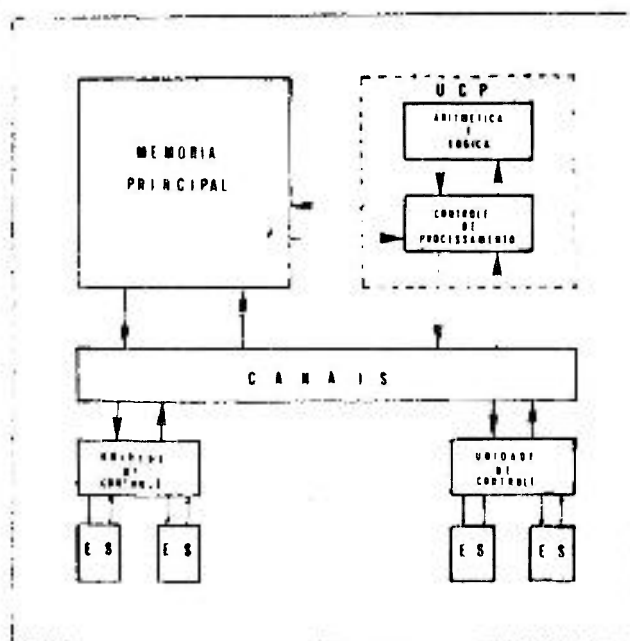


Figura 1 — Componentes Básicos do Hardware

No que se refere ao sistema operacional, também podemos distribuí-lo em três grupos cada um dos quais composto de vários módulos:

- Grupo de módulos requeridos
- Grupo de módulos alternativos
- Grupo de módulos opcionais

onde os módulos requeridos são aqueles de que o sistema operacional sempre necessita. Os alternativos são módulos também necessários porém com alternativa de escolha. Finalmente os opcionais podem ou não fazer parte do sistema operacional.

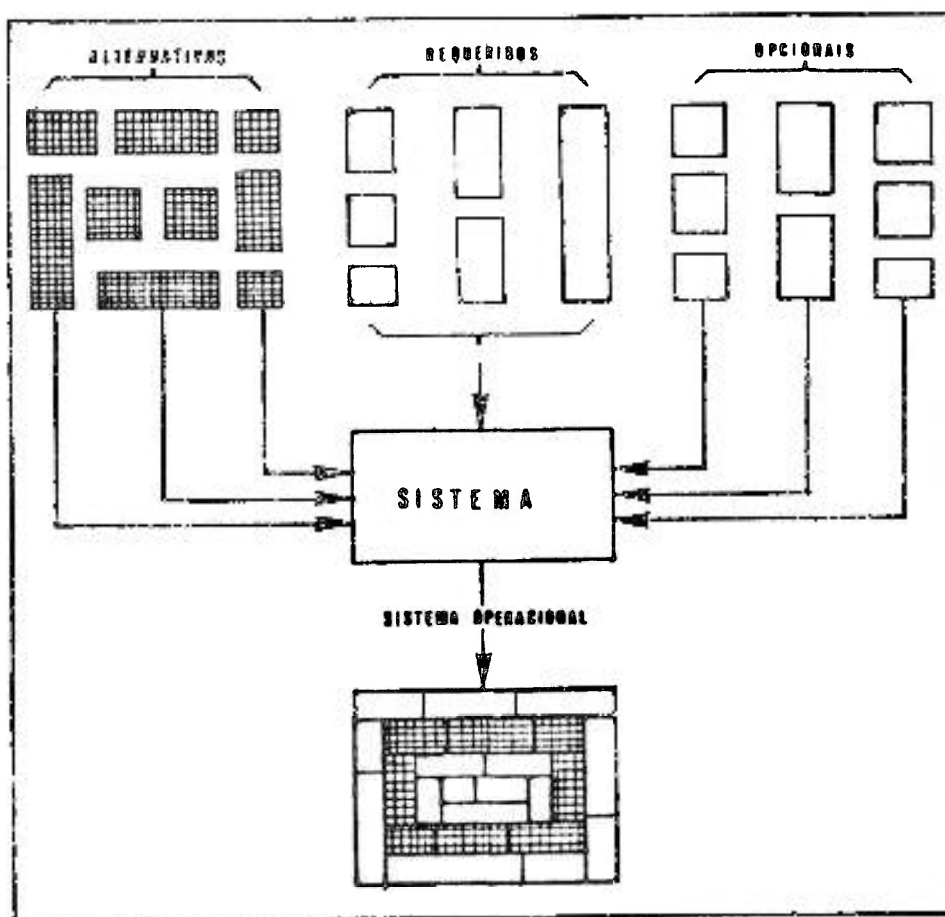


Figura 2 — Criação do Sistema Operacional do Usuário

A criação do sistema operacional do usuário se dá em três etapas:

I — STARTER SYSTEM a biblioteca completa de programas de O.S. (STARTER SYSTEM) enviada ao usuário pelo Program Information Department (P.I.D.)

II — SYSTEM GENERATION dessa biblioteca completa são selecionados os programas e bibliotecas necessárias sendo a seleção efetuada a partir de um procedimento chamado SYSTEM GENERATION.

III – O.S. DO USUÁRIO o O.S. do usuário está criado, correspondendo às suas necessidades.

Uma vez instalado, o O.S. propicia ao Sistema uma série de facilidades dentre as quais destacam-se:

- **Determinação da prioridade de Job** – o O.S. permite a execução dos Jobs segundo a prioridade estabelecida pelo usuário, independentemente da ordem de entrada no sistema. Esta competição prioritária existe tanto em Jobs alimentados por terminais remotos, como em Jobs que entram via equipamentos da própria instalação.

- **Aproveitamento do tempo de Set-up** – Tempo de Set-up é o tempo que o operador gasta para preparar um dispositivo de entrada e saída. O O.S. permite aproveitar este tempo, processando programas em outras partes da memória.

- **Alocação automática de recursos** – Por recursos entendemos tudo aquilo que é passível de ser disputado ex: memória principal, áreas em discos, etc. Estes recursos são alocados de forma automática e otimizada pelo O.S. que cria as chamadas RESOURCE QUEUE (Filas de Recursos) onde especifica qual recurso está sendo requisitado e qual o programa que está requisitando. Estas RESOURCE QUEUE são pesquisadas sempre que ocorrer uma interrupção qualquer.

- **Designação simbólica de unidades de E/S** – O O.S. permite que o usuário associe uma ou mais unidades físicas de entrada e saída, com características similares, a um certo símbolo (FITA, DISCO, SYSDA, 3420) e a partir desse instante usar o símbolo do grupo de unidades desejado, deixando ao sistema a escolha final. Com isto, o usuário não precisa se preocupar com o endereço físico e outros detalhes específicos de cada unidade, assim como, fica eliminado o problema do usuário escolher uma unidade ocupada ou não operativa naquele momento.

O principal objetivo do O.S. é conseguir um uso eficiente do sistema considerando a configuração – hardware/software – e os tipos de aplicações do usuário. Este uso eficiente de um sistema é denominado "PERFORMANCE" e não é nenhum valor numérico mas sim a combinação de três fatores:

- **THROUGHPUT** – volume total de trabalho executado pelo computador num dado intervalo de tempo.

- **TURNAROUND TIME** – ou tempo de resposta – tempo médio transcorrido entre a submissão de um item de trabalho ao computador e a obtenção do seu resultado.

- **DISPONIBILIDADE** – grau de disponibilidade dos recursos necessários a serem usados para o processamento.

Com o O.S. conseguimos incrementar o throughput, reduzir o turnaround time e otimizar automaticamente o uso de recursos. Em outras palavras, o uso do O.S. acarreta uma considerável melhora na Performance de um sistema.

Outro objetivo do O.S. é proporcionar, aos usuários em geral, assistência e informações necessárias ao desempenho de suas atribuições, além de permitir expansão do sistema – aumento de recursos de Hardware e Software – sem necessidade de reprogramar o que já havia sido desenvolvido.

Embora voltemos a este assunto em detalhes, vamos adiantar que existem três opções para instalar um sistema operacional:

- P. C. P. — Primary Control Program
- M. F. T. — Multiprogramming with a Fixed Number of Tasks
- M. V. T. — Multiprogramming with a Variable Number of Tasks

e a escolha de uma delas depende basicamente das necessidades do CPD e da configuração disponível.

2 – ORGANIZAÇÃO DO O.S.

O sistema operacional do /360 é constituído de uma coleção organizada de programas que podem ser agrupados em duas classes fundamentais: **programas de controle** e **programas de processamento**. Os primeiros são aqueles cuja execução somente é possível quando a U.C.P. estiver em **estado de supervisor**; os outros tem sua execução permitida quando a U.C.P. estiver em **estado de problema**.

2.1 -- Programas de Controle

Os programas de controle dispõem de rotinas que coordenam as ações do O.S. respondendo por todas as funções oferecidas ao usuário. Compõem-se de:

- JOB MANAGEMENT (programas controladores de JOB)
- TASK MANAGEMENT (programas controladores de TASKS)
- DATA MANAGEMENT (programas controladores de DADOS)

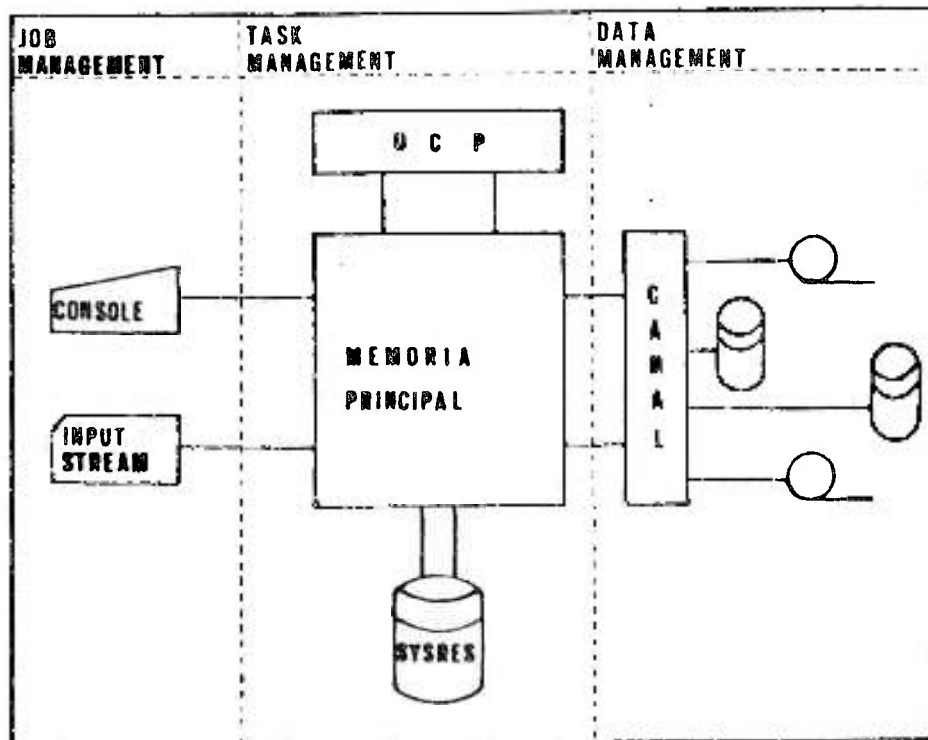


Figura 3 — Componentes do Programa de controle

2.1.1. – JOB MANAGEMENT

As rotinas do JOB MANAGEMENT ocupam uma posição importantíssima no O.S. pois cabe-lhes dirigir e controlar o fluxo de Jobs no sistema além de possibilitar a comunicação entre o operador e o sistema e vice-versa. Basicamente suas funções são:

- Analisar o input stream
- Obter unidades de entrada/saída
- conseguir espaço em DASD
- seleccionar jobs para a execução
- transcrever dados (spooling)
- comunicar-se com o operador

Para cumprir suas funções o JOB MANAGEMENT é dividido em:

```

*JOB SCHEDULER < { READER / INTERPRETER
                   { INITIATOR / TERMINATOR
                   { OUTPUT WRITER
  
```

*MASTER SCHEDULER

O MASTER SCHEDULER é a parte do JOB MANAGEMENT encarregada da comunicação entre o operador e o sistema operacional. Esta comunicação se dá, via de regra, através da console de operação. Nos casos em que a operação do sistema possa ser planejada com antecipação, certos comandos de operação podem ser colocados, via JOB CONTROL CARDS, no INPUT STREAM onde são enviados ao Master Scheduler à medida que são identificados pelo Job Scheduler (Reader/Interpreter).

O JOB SCHEDULER por sua vez se compõe de três programas:

I – READER/INTERPRETER – Este programa do JOB MANAGEMENT é carregado na memória via comando START READER. Em MVT a Reader entra na posição mais alta da memória e seu tamanho varia entre 40 e 60 K de memória. Em MFT ela tem o tamanho do Initiator.

As funções principais do READER/INTERPRETER são:

- a – ler, interpretar e analisar o Input Stream
- b – criticar a consistência dos cartões de controle – J.C.C.
- c – assumir os defaults
- d – carregar os cartões de dados num data set temporário e intermediário chamado Input Data Set, assinalando seu endereço nas tabelas correspondentes da SYS1.SYSJOBQE
- e – gravar os J.C.C. nas SYS1.SYSJOBQE segundo classe e prioridade em forma de tabelas (blocos) de controle que são utilizados para descrever o JOB ao sistema.

As tabelas são:

- J.C.T. – JOB CONTROL TABLE – construída a partir de informações do cartão JOB
- S.C.T. – STEP CONTROL TABLE – construída a partir de informações do cartão EXEC
- S.I.O.T. – STEP I/O TABLE – construída a partir de informações dos cartões DD para o STEP.
- J.F.C.B. – JOB FILE CONTROL BLOCK – construída a partir de informações dos cartões DD referentes aos arquivos do JOB.

A partir das tabelas JCT, SCT e SIOT é construída uma outra tabela chamada TIOT – TASK INPUT OUTPUT TABLE – a qual possui uma entrada para cada data set usado durante o job step.

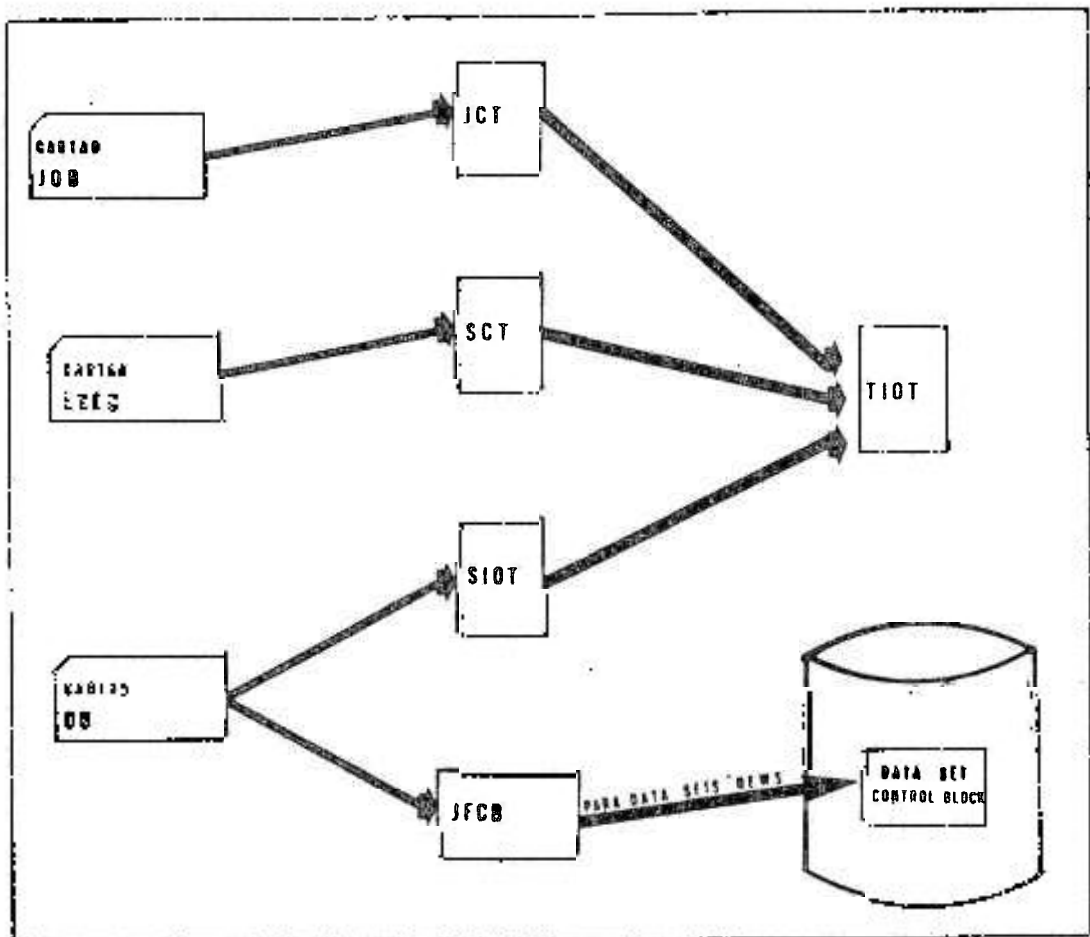


Figura 4 – Formação de Tabelas (BLOCOS) de Controle

II – INITIATOR/TERMINATOR – é carregado na memória via comando START INIT,, A...

Em MVT podemos ter até oito classes e em MFT até três.

As funções principais do INITIATOR/TERMINATOR são:

- a — seleccionar os JOBS a partir das classes correspondentes na SYS1.SYSJOBQE: sequencialmente se for P.C.P. e por prioridades se for M.F.T. ou M.V.T.
- b — alocar memória (em MVT)
- c — alocar dispositivos de entrada e saída
- d — ficar totalmente dedicado ao JOB até o último Step.
- e — fechar os arquivos que não foram fechados pelo Job.
- f — carregar o registrador 15 com código de retorno.

III — OUTPUT WRITER — realiza o 'spool' de saída, isto é: transcreve a saída intermediária do usuário (OUTPUT DATA SET) criada temporariamente em disco, para os respectivos dispositivos de saída tais como: impressora, perfuradora, etc (ver Figura 5). Por meio do cartão DD do JCL o usuário indica em que classe de writer deverá ser armazenado seu relatório. Cada classe, em tempo de geração, é associada a um dispositivo de saída ex: // DD SYSOUT = A. Isto significa que o relatório do usuário será armazenado em uma classe de writer A.

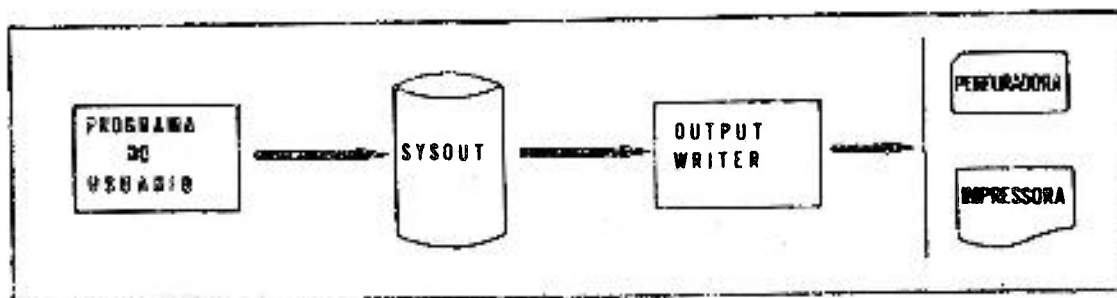


Figura 5 — Esquema do "spool" de Saída

Estes três programas que compõem o JOB SCHEDULER manipulam dados contidos na SYS1.SYSJOBQE. Para isto eles se utilizam de rotinas do Sistema Operacional chamadas QUEUE MANAGER às quais cabe a manutenção da SYS1.SYSJOBQE que reside em unidade de acesso direto.

Vistas, de forma esquemática, as principais funções da READING TASK, INITIATING TASK e WRITING TASK, vejamos o funcionamento do JOB SCHEDULER como um todo considerando as opções MFT e MVT.

O Input Stream é lido, interpretado e analisado, separando os cartões de dados dos cartões de controle — J.C.C. — . Os primeiros são gravados num data set temporário chamado Input Data Set e os J.C.C. são gravados num data set do sistema chamado SYS1.SYSJOBQE, segundo classe e prioridade, indo constituir as tabelas, ou blocos, de controle que descrevem o JOB para o sistema. Os JOBS assim gravados ficam a espera de serem selecionados para a execução. Uma vez selecionado o JOB na SYS1.SYSJOBQE, o INITIATOR/TERMINATOR descobre, através da Step Control Table, qual o JOB STEP a ser carregado, carrega-o cuidando da alocação dos recursos necessários à sua execução. Feito isto o JOB STEP é inicializado, após o que o controle é transferido até o final da execução quando o controle é retomado pelo INITIATOR/TERMINATOR para finalizar o JOB STEP liberando ou não os recursos alocados (dependendo do que for determinado pelos J.C.C.). Depois disto outro JOB STEP é buscado e o ciclo recomeça até que todos os JOB STEPs daquele JOB estejam terminados.

O esquema representado na Figura 6 mostra o fluxo de operações realizadas pelos programas do JOB MANAGEMENT.

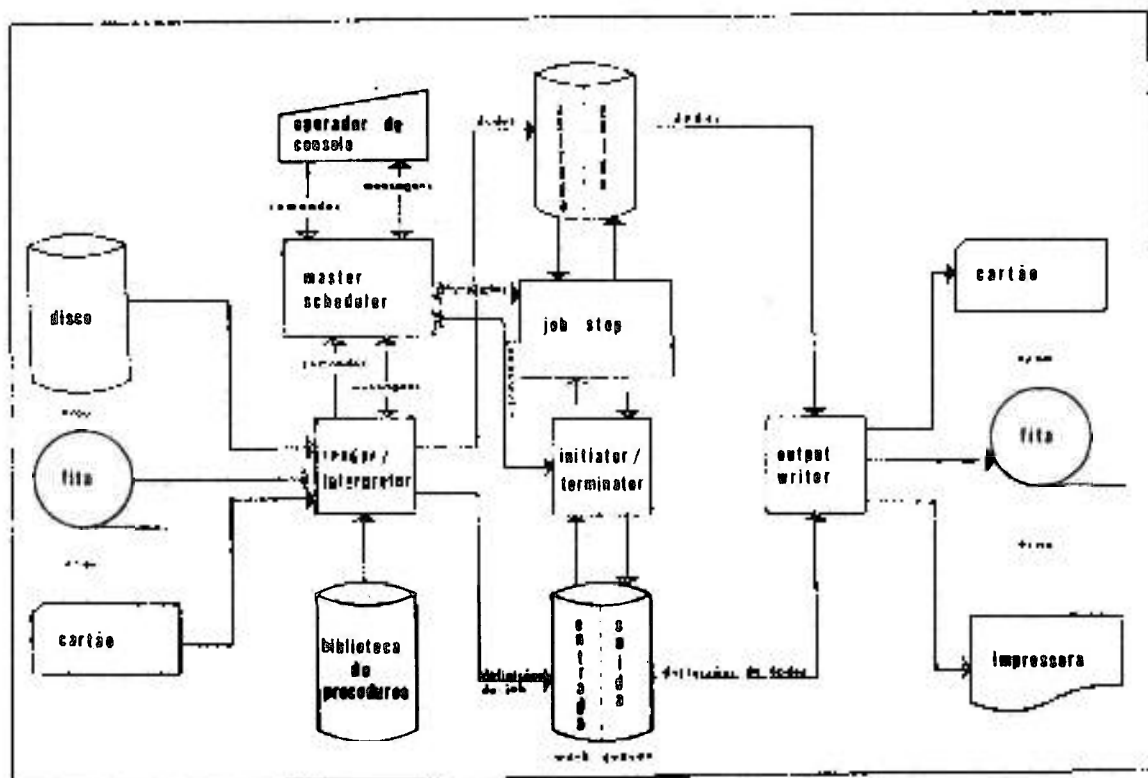


Figura 6 — Job Management

2.1.2 — TASK MANAGEMENT

De modo geral Task é a menor unidade de trabalho que luta independentemente, através da sua Dispatching Priority, pela U.C.P. ou, em outras palavras, é um "programa" passível de execução independente.

Em O.S. as Tasks são representadas pelo Task Control Block-T.C.B. — que é um bloco de controle onde estão contidas todas as informações necessárias ao processamento de uma Task.

É através da TCB que o O.S. "fica sabendo" que uma tarefa precisa ser executada e existirão tantas TCBs quantas forem as Tasks em execução no sistema.

O TASK MANAGEMENT é um conjunto de rotinas, que operam todas em estado de supervisor, cujas funções básicas são:

A) – Supervisionar a execução de todo trabalho que o sistema realiza.

B) – Controlar o uso da U.C.P. e outros recursos.

Em vista do tipo de função que lhe cabe o Task Management é chamado também de Supervisor.

O Supervisor desenvolve suas funções através de:

I – Supervisão de Interrupções

O supervisor recebe o controle da U.C.P. imediatamente após uma interrupção a qual é analisada para determinar qual rotina do sistema deve ser ativada.

Uma interrupção ocorre sempre que um programa necessite dos serviços do programa ou então se algum evento requer processamento do Supervisor.

Existem 5 tipos de interrupções:

A) – Interrupção por chamada do supervisor (SVC INTERRUPTION). Esta interrupção é causada por instruções do programa que pedem a transferência do controle para o Supervisor.

B) – Interrupção de entrada e saída (I/O INTERRUPTION) – causada sempre pelo fim de uma operação de entrada e saída.

C) – Interrupção externa (EXTERNAL INTERRUPTION) – causada por algum dispositivo externo que necessita do Supervisor.

D) – Interrupção por erro de programa (PROGRAM INTERRUPTION) – este tipo de interrupção ocorre sempre que um programa tentar executar operações inválidas.

E) – Interrupção por erro de máquina (MACHINE INTERRUPTION) – ocorre quando a U.C.P. detecta erro no Hardware do Sistema.

II – Supervisão de TASKs

Exerce o controle de todas as Tasks do sistema no que diz respeito a seu uso, execução, status, sincronização e principalmente a ordem em que estas tasks são executadas (Dispatching Priority). Esta supervisão é efetuada ao nível das TCBs. (TASK CONTROL BLOCK).

III – Supervisão de Memória Principal

É a alocação, controle e liberação da memória principal na região dinâmica e na System Queue Área.

IV – Supervisão de Conteúdo

As rotinas de supervisão de conteúdo da memória tem a atribuição de buscar e carregar programas e rotinas, não residentes, na memória principal e passar-lhes o controle.

V – Supervisão de TIMER

Trata-se de exercer um controle interno do tempo do sistema e processar suas interrupções.

Uma interrupção de tempo ocorre quando um valor, no intervalo de tempo, torna-se negativo mostrando com isto que o tempo determinado expirou.

Uma vez vistas as funções das rotinas do Supervisor ou Task Management – resta chamar a atenção para a interação existente entre o supervisor e o usuário. Através dos parâmetros que o usuário fornece ao sistema, quer diretamente, quer por default, ele está interagindo direta ou indiretamente com as rotinas do TASK MANAGEMENT. Como exemplo de parâmetros que participam da interação temos:

- Alocação da memória
- Criação de Tasks
- Determinação de prioridades
- Time slicing
- Etc.

2.1.3 – DATA MANAGEMENT

Com o advento de instalações de processamento de dados cada vez maiores e mais possantes, cresce consideravelmente o volume de programas e arquivos a serem manipulados diariamente, o que acarreta problemas na coordenação da utilização dos mesmos. Os volumes de memória auxiliar aumentam sua capacidade de armazenamento e os dispositivos suportam cada vez mais volumes. Isto torna o acesso mais complexo e o tempo de set-up maior.

Para isso o O.S. conta com um conjunto de programas controladores de dados – DATA MANAGEMENT – os quais são os responsáveis pela organização dos registros de informações, pelo controle de todo o meio externo de armazenamento e pela transferência de dados entre a memória principal e a externa.

Estes programas são compostos de rotinas tais como:

I – DADSM – DIRECT ACCESS DEVICE SPACE MANAGEMENT – são rotinas que procuram e alocam espaço em DASD de forma automática e otimizada.

II – Métodos de Acesso – rotinas para manusear tipos de arquivos particulares considerando sua organização e a técnica de pesquisa dos mesmos.

III – Supervisor de Entrada e Saída – afeta o controle de transferência de dados da memória para um dispositivo de entrada e saída e vice-versa.

IV – Suporte de Entrada e Saída – consiste de rotinas transientes que realizam funções específicas relacionadas com arquivos. Entre elas temos OPEN, CLOSE, EOVS, etc.

Mais adiante, num capítulo a parte, serão apresentados alguns conceitos relativos ao DATA MANAGEMENT, tais como: formatos de registros, organização de data sets, técnica de acesso, etc.

2.2 – Programas de Processamento

Existem três tipos de programas de processamento:

- COMPILADORES, TRADUTORES OU MONTADORES
- PROGRAMAS DE SERVIÇO
- PROGRAMAS DO USUÁRIO

2.2.1 – Compiladores, Tradutores ou Montadores

São programas que convertem (traduzem) uma linguagem fonte compreensível pelo usuário, numa outra linguagem compreensível pela máquina chamada linguagem de máquina.

Os compiladores são programas com características similares a outros programas em estado problema e para o O.S. não há distinção entre estes e aqueles.

Evidentemente existe um compilador para cada linguagem fonte ao qual cabe a tradução daquela linguagem para a linguagem de máquina.

Por outro lado, em decorrência de limitações de HARDWARE, a maioria dos compiladores para linguagens de alto nível não se apresenta constituída de todas as possibilidades, mas sim de um subconjunto das mesmas que varia em quantidade dependendo da potencialidade da máquina. Desta forma, embora teoricamente seja possível um programa numa certa linguagem de alto nível ser convertido para uma linguagem de máquina, utilizando qualquer compilador desta linguagem, na prática isto não acontece. É preciso conhecer as restrições específicas do compilador utilizado, para efetuar certas modificações no programa fonte de modo a permitir sua compilação.

Uma observação a ser feita diz respeito ao uso do termo MONTADOR. A princípio este termo era usado quando se tratava de conversão de uma linguagem de baixo nível para linguagem de máquina. Entretanto a distinção não se faz mais necessária na medida em que muitas linguagens de baixo nível se apresentam com características similares às de alto nível, principalmente no que se refere à utilização de macro instruções. De qualquer forma o termo mais utilizado entre todos é o termo COMPILADOR, independentemente do nível de linguagem considerada.

São exemplos de compiladores:

- ASSEMBLER
- COBOL
- FORTRAN
- R P G
- Etc.

A Figura 7 mostra os compiladores fornecidos pela IBM e o mínimo de memória necessária para cada um deles.


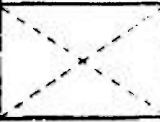
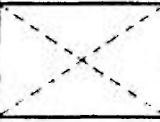






EXISTE  NAO EXISTE 	COMPILADORES			
	E 32.768 bytes	F 65.536 bytes	G 131.072 bytes	H 262.144 bytes
ASSEMBLER				
FORTRAN				
COBOL				
PL/I				
RPG				
ALGOL				

Figura 7 — Compiladores IBM

2.2.2 — Programas de Serviço

Estes programas, via de regra fornecidos pelo fabricante, tem a finalidade de auxiliar em funções frequentemente usadas na programação no que diz respeito à manipulação de programas e dados.

Apresentamos a seguir os quatro grandes grupos que constituem os chamados programas de serviço abordando-os apenas em seus aspectos gerais e em nível informativo, como aliás todas estas notas o são.

Os programas de serviço dividem-se em:

2.2.2.1 — Linkage Editor

É um programa encarregado de preparar os módulos de carga para execução. Módulo de carga é uma unidade lógica de codificação capaz de realizar uma função ou várias relacionadas, que está em formato relocável, já compilado, pronto para execução e carregado a partir de qualquer endereço de memória. Portanto a função do Linkage Editor é preparar a saída dos compiladores para execução. Esta saída é trazida posteriormente para a memória pelo programa FETCH. .

As funções básicas do Linkage Editor são:

— Ligação de Módulos

Devido à facilidade de trabalho e economia de tempo de programação um programa é, freqüentemente, dividido em módulos cada um dos quais podendo ser codificado em linguagens diferentes. Assim que cada módulo for compilado o Linkage Editor intervem e faz a ligação dos mesmos originando um módulo único.

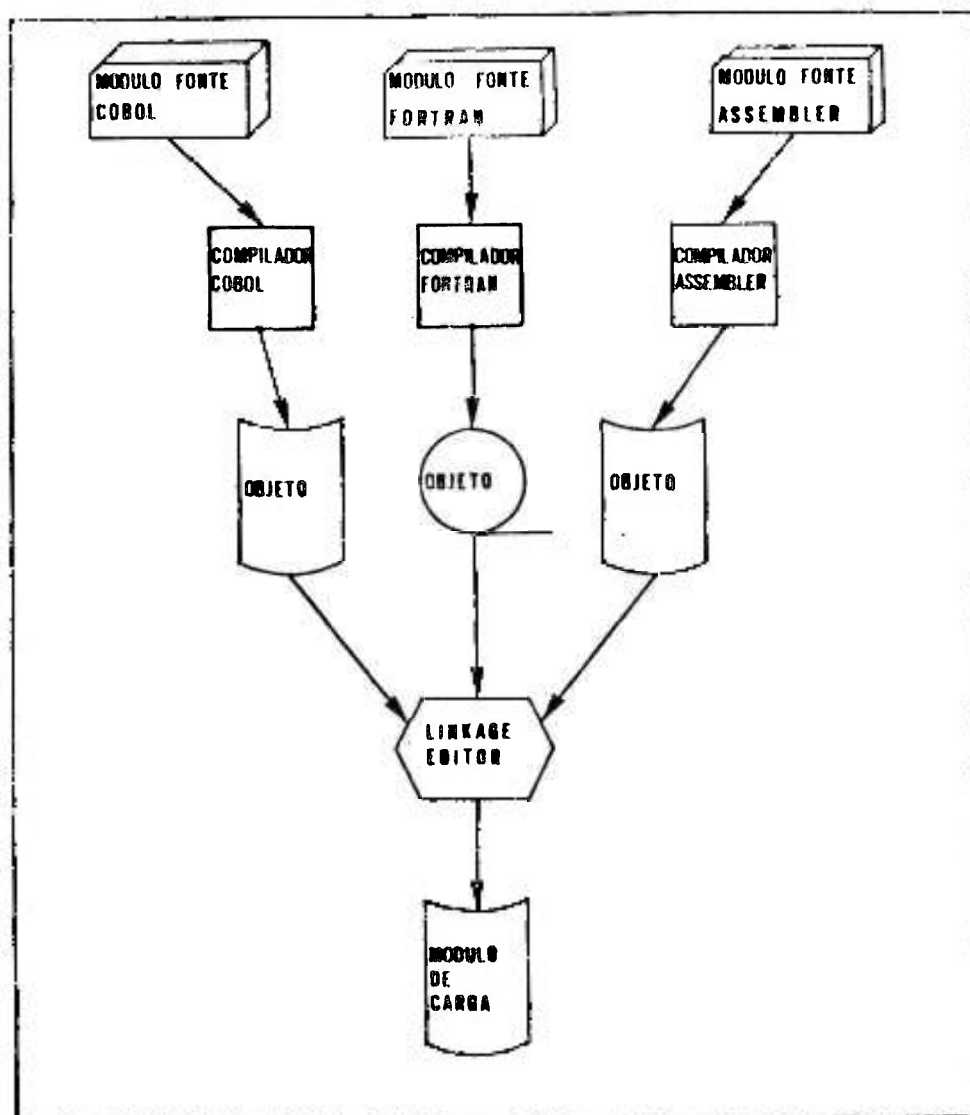


Figura 8 — Ligação de Módulos pelo Linkage Editor

— Edição de Módulos

Nos casos em que certas funções de um programa são substituídas é possível Linkeditar somente as seções de controle afetadas e não o módulo fonte inteiro.

A partir de informações vindas dos cartões de controle — J.C.C. — o Linkage Editor pode substituir, renomear, deletar ou mover seções de controle.

Entendemos por seção de controle a menor unidade que pode ser carregada separadamente.

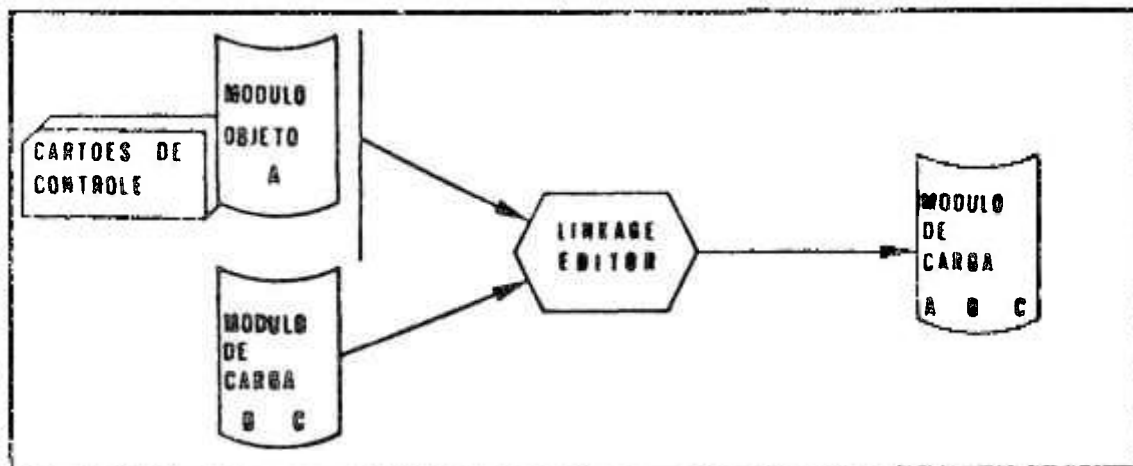


Figura 9 — Edição de Módulos

— Aceitação de Fontes Adicionais de Entrada

O Linkage Editor permite que subrotinas padrões possam ser incluídas reduzindo portanto o trabalho de codificação do programa. Esta inclusão é efetuada sob comando de cartões de controle. Quando o Linkage Editor processa um programa deste tipo, o módulo contendo a subrotina é recuperado na fonte de entrada modificada e feito a parte um módulo de saída.

Os símbolos não resolvidos, mesmo após o processamento de toda a entrada, acionam o mecanismo de pesquisa automática da biblioteca.

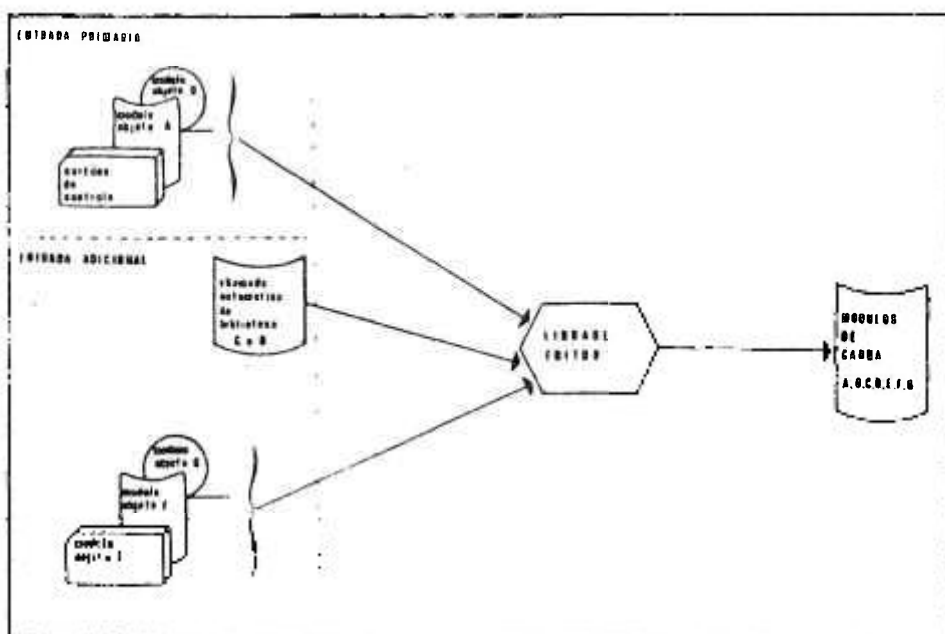


Figura 10 — Fontes Adicionais de Entrada

— Reserva de Memória

O Linkage Editor processa seções comuns geradas pelos compiladores FORTRAN e ASSEMBLER. Da mesma forma podem ser processadas áreas estáticas externas reservadas pelo compilador PL/I.

As áreas comuns são coletadas pelo Linkage Editor e a reserva de memória principal é providenciada dentro do módulo de saída.

— Processa Pseudo Registros (Registros Dummy)

O Linkage Editor processa pseudo registros acumulando o tamanho total da memória requerida para os mesmos e registrando o deslocamento de cada um deles. Durante a execução, a memória necessária é adquirida dinamicamente pelo programa.

— Cria Estruturas de Overlay

Em caso de programas que usam subrotinas externas que não devem ser incorporadas aos mesmos para minimizar a quantidade de memória necessária à execução do programa. Neste caso é reservada uma área no programa principal, para a qual são trazidas as subrotinas e executadas uma a uma sob o controle do programa. O Linkage Editor resolve os endereços e a relocação destas subrotinas e a isto dá-se o nome de criação de overlay.

— Criação de Módulos de Carga Múltiplos

O Linkage Editor tem a faculdade de criar vários módulos de carga dentro de um mesmo Step. Cada módulo de carga é colocado na biblioteca com um nome de membro único, especificado por cartão de controle.

— Processamentos Especiais e Opções no Diagnóstico de Saída

Por meio de cartões de controle é possível influir no processamento bem como requisitar maior número de informações no relatório de saída.

— Designação de Atributos ao Módulo de Carga

Quando o Linkage Editor gera um módulo de carga, ele coloca uma entrada para o módulo no diretório da biblioteca. Esta entrada contém atributos que descrevem a estrutura, o conteúdo e o formato lógico do módulo. O programa de controle usa estes atributos para determinar: de que maneira o módulo deve ser carregado, qual seu conteúdo, se é executável, se é executável mais de uma vez sem recarga, se pode ser executado concorrentemente por mais de uma TASK.

— Designação de Memória com Hierarquia

Esta característica é possível quando for utilizada a memória de núcleos IBM — 2361 (L.C.S. — Large Capacity Storage)

— Reserva de Áreas na Memória Principal

Permite que seja especificada a quantidade de memória principal disponível para o Linkage Editor e para os Buffers. Estas opções somente podem ser especificadas quando for utilizado o nível F do Linkage Editor (44K bytes de memória principal).

– Relacionamento com o Sistema Operacional

O Linkage Editor pode ser executado de três formas:

- * JOB STEP — quando o Linkage Editor é especificado num cartão de controle EXEC no Input Stream.
- * SUBPROGRAMA — com a execução das macro instruções: CALL, LINK, XCTL.
- * SUBTASK — com a execução da macro instrução ATTACH.

2.2.2.2 – LOADER

Possui, basicamente, as mesmas funções do Linkage Editor porém não permite a catalogação dos módulos de carga. O LOADER prepara um programa executável e passa o controle diretamente ao mesmo. É mais rápido que o Linkage Editor e por isso mesmo é indicado para testes de programas.

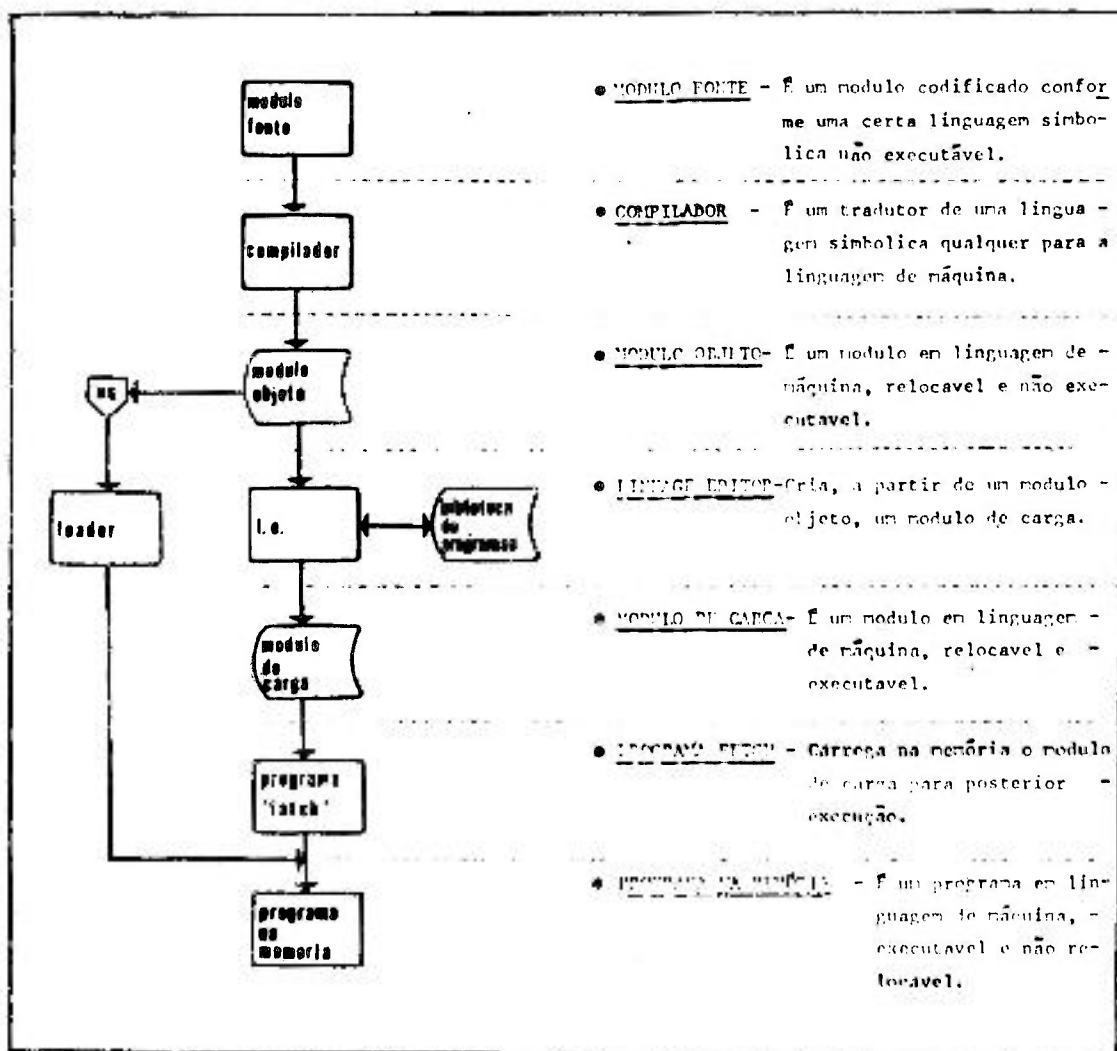


Figura 11 — Módulos

2.2.2.3 – SORT/MERGE

É um programa de serviço utilizado para classificar (SORT) e/ou intercalar (MERGE) registros de tamanho fixo ou variável, em várias modalidades, utilizando disco ou fita como entrada, saída e/ou área de trabalho.

Para poder utilizar o programa SORT/MERGE são necessários os seguintes requisitos:

I – Memória Principal:

15.500 bytes além da memória ocupada pelo Sistema Operacional. Destes, o SORT/MERGE utiliza 12.000 e certas funções do sistema utilizam os 3.500 restantes.

II – Memória Auxiliar (somente em caso de SORT pois o MERGE não precisa)

A ser usada como áreas de trabalho devendo estar constituída no mínimo de:

1 Unidade de Acesso Direto (2311, 2314, 2301 ou 3330) ou

3 Unidades de Fita Magnética

Ainda com relação à memória auxiliar é de se observar que a quantidade necessária depende do tamanho do INPUT e os dispositivos utilizados devem respeitar às seguintes restrições:

1ª – Residir no mesmo TIPO DE DISPOSITIVO

2ª – Tratando-se de fita magnética o programa SORT/MERGE pode operar com uma mistura de fitas de 7 a 9 trilhas. Se o data set de entrada do SORT está numa fita de 7 trilhas, pode ser usada qualquer combinação de fitas de 7 e 9 trilhas, para memória auxiliar e para saída, podendo ainda gravar a saída em discos (2311, 2314, 3330), ou tambor 2301.

Mas, se não for usada uma fita de 7 trilhas como suporte do data set de entrada do SORT, também não pode ser usada uma fita de 7 trilhas para memória auxiliar ou para saída.

3ª – Quando a memória auxiliar estiver em DASD devem ser usadas no mínimo três áreas do mesmo tamanho.

III – No mínimo um canal seletor ou um canal multiplexor.

A lógica do SORT/MERGE pode ser esquematizada conforme mostra a Figura 12.

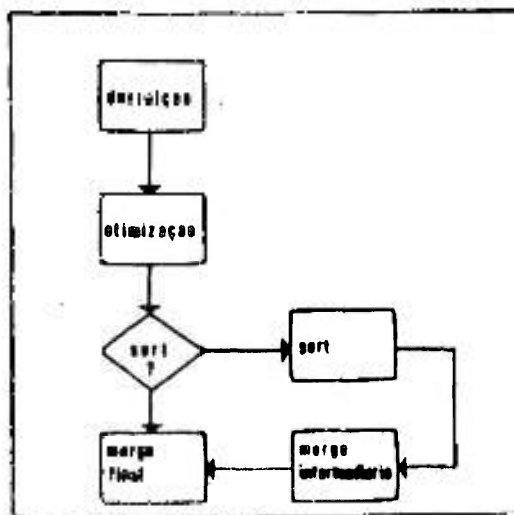


Figura 12 – Lógica do Sort / Merge

2.2.2.4 – UTILITÁRIOS

São programas de serviço utilizados para manipulações de data set como: mudança de meio de armazenamento, atualização, deleção, inicialização de volumes, etc. São programas bastante poderosos e normalmente cada um deles pode executar mais de uma função, assim como, cada função pode, via de regra, ser executada por mais de um utilitário.

Estes programas são fornecidos pela IBM e são agrupados em três classes de acordo com suas características:

Divisão dos Utilitários no O.S. Release 19			Divisão dos Utilitários no O.S. Release 21.7		
SYSTEM UTILITIES	DATA SET UTILITIES	INDEPENDENT UTILITIES	SYSTEM UTILITIES	DATA SET UTILITIES	INDEPENDENT UTILITIES
IEHPROGM	IEBCOPY	IBCDASDI	IEHPROGM	IEBCOPY	IBCDASDI
IEHMOVE	IEBGENER	IBCDMPRS	IEHMOVE	IEBGENER	IBCDMPRS
IEHLIST	IEBCOMPR	IBRCVPR	IEHLIST	IEBCOMPR	IBRCVPR
IEHINITT	IEBTPCH		IEHINITT	IEBTPCH	ICAPRTBL
IEHIOSUP	IEBTCRIN		IEHIOSUP	IEBTCRIN	
IFCEREPO	IEBUPDTE		IEHDASDR	IEBUPDTE	
IFCDIPOO	IEB.SAM		IEHATLAS	IEB.SAM	
IEHDASDR	IEBEDIT		IEHSTARTR	IEBEDIT	
IEHATLAS	IEBUPDAT			IEBUPDAT	
IFHSTARTR	IEBDG			IEBDG	

I – SYSTEM UTILITIES

Executam funções do sistema tais como: criar estruturas, deletar e catalogar data sets, etc. Sua operação é controlada pelo usuário por meio de cartões de controle J.C.L. e por cartões de controle do próprio utilitário. As funções gerais de cada utilitário desta classe são:

- IEHPROGM – construção e manutenção dos dados de controle do sistema.
- IEHMOVE – cópia ou movimentação de conjunto de dados.
- IEHLIST – listagem dos dados de controle do sistema.
- IEHINITT – gravação de "Labels Standards" em volumes de fita magnética.
- IEHIOSUP – atualização das entradas na biblioteca de rotinas não residentes – SYS1.SVCLIB.
- IEHDASDR – inicialização de volumes DASD e emissão de "DUMPs" ou "RESTORE" de dados em DASD.
- IEHATLAS – assinalação de trilhas alternadas, para trilhas defeituosas.
- IFHSTARTR – seleção, formatação e gravação de informações relativas a erros na fita que coleta informações quanto à utilização dos recursos para cada programa processado – IFASMFDP ou SYS1.MAN.

II – DATA SET UTILITIES

Executam funções de reorganização, mudança ou comparação de dados ao nível de arquivo e/ou registro. São controlados também por cartões de controle J.C.L. e do próprio utilitário, com exceção do IEBISAM que não tem cartões de controle próprios e usando portanto apenas os cartões de controle do J.C.L.

As funções gerais de cada um são:

- IEBCOPY – copia, condensa ou intercala data sets particionados (P.D.S.); ou exclue determinados membros numa operação de cópia; rebatiza e/ou substitue membros de um P.D.S.
- IEBGENER – copia registros de arquivos seqüenciais, ou converte um arquivo seqüencial em P.D.S.
- IEBCOMPR – compara registros de arquivos seqüenciais ou P.D.S.
- IEBPTPCHr – imprime ou perfura registros de arquivos seqüenciais ou P.D.S.
- IEBTCRIN – constroe registros a partir de entrada lida em 2495 Tape Cartridge Reader.
- IEBUPDTE – atualiza uma biblioteca simbólica.
- IEBISAM – grava dados de um arquivo indexado-seqüencial, num arquivo seqüencial; reconstruindo o arquivo indexado-seqüencial a partir do seqüencial criado.
- IEBEDIT – cria um Input Stream.
- IEBUPDAT – atualiza uma biblioteca simbólica.
- IEBDG – cria um arquivo amostra para ser usado em teste de programas.

III – INDEPENDENT UTILITIES

São usados para preparar volumes DASD para uso posterior. São chamados independentes porque são processados sem o sistema operacional O.S. e, por isso mesmo, sua operação é controlada sem o uso de cartões de controle J.C.L., utilizando apenas os cartões de controle do utilitário.

As funções gerais são:

- IBCDASDI – inicializa e assinala trilha alternativa para volume DASD.
- IBCDMPRS – permite fazer “DUMP” e “RESTORE” de dados contidos em volume DASD.
- IBCRCVRP – recupera registros aproveitáveis de trilhas defeituosas; assinala trilha alternada e intercala registros de “substituição” com os registros recuperados, gravando-os na trilha alternada.
- ICAPRTBL – permite carregar o buffer do Universal Character Set (UCS) e o buffer de controle do formulário (FCB) da impressora IBM 3211.

No anexo I apresentamos uma tabela mostrando algumas aplicações mais usuais de utilitários, em ordem crescente, e ao lado de cada aplicação o utilitário ou utilitários que as executam.

O formato geral dos cartões de controle dos utilitários é o seguinte:

NOME	OPERAÇÃO	OPERANDO	COMENTÁRIO
Nome simbólico e opcional, exceto para o IEHINITT. Começa na col. 1.	Identifica o tipo de operação devendo ser precedido e seguido de, pelo menos, um branco.	Parâmetros "KEYWORDS" separados por vírgula.	Opcional. Se codificado deve ser precedido de, pelo menos, um branco.

2.2.3. – Programas do Usuário

São aqueles programas escritos pelos usuários e que tem por finalidade executar determinadas tarefas específicas. Por ex: folha de pagamento, controle de estoque, cálculo de uma matriz, etc.

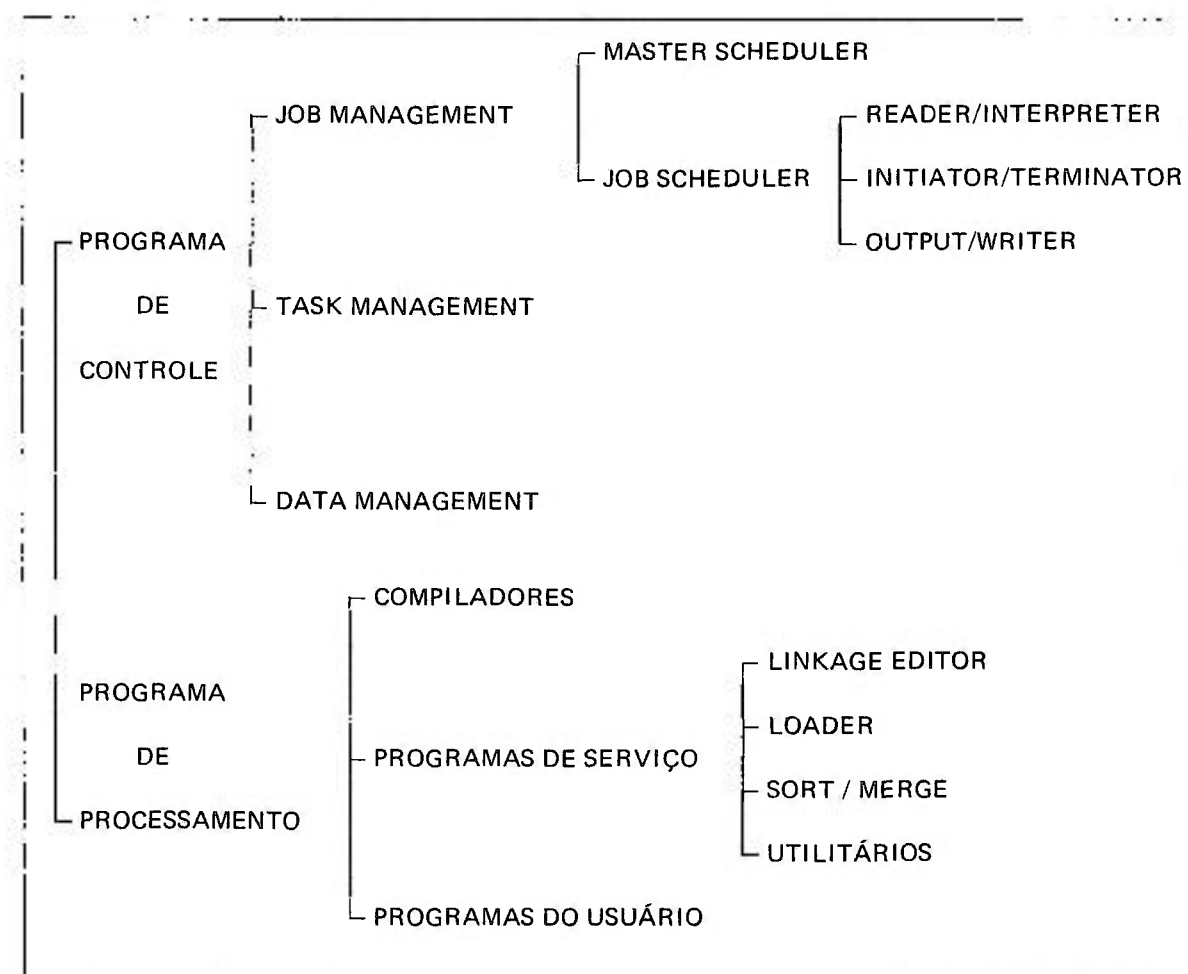


Figura 13 – Organização do O.S.

Cada vez que o programa de processamento necessita dos serviços de uma rotina do programa de controle é gerada uma interrupção por meio de uma instrução SVC e a U.C.P. entra em estado de Supervisor determinando qual a rotina necessária para processar o serviço requisitado pelo programa de processamento.

Para devolver o controle ao programa de processamento, o sistema usa a instrução LPSW que muda o estado da UCP, levando-a a estado problema.

A Figura 14 mostra um esquema onde são representados os dois tipos de programas do O.S., suas subdivisões e a relação entre ambos.

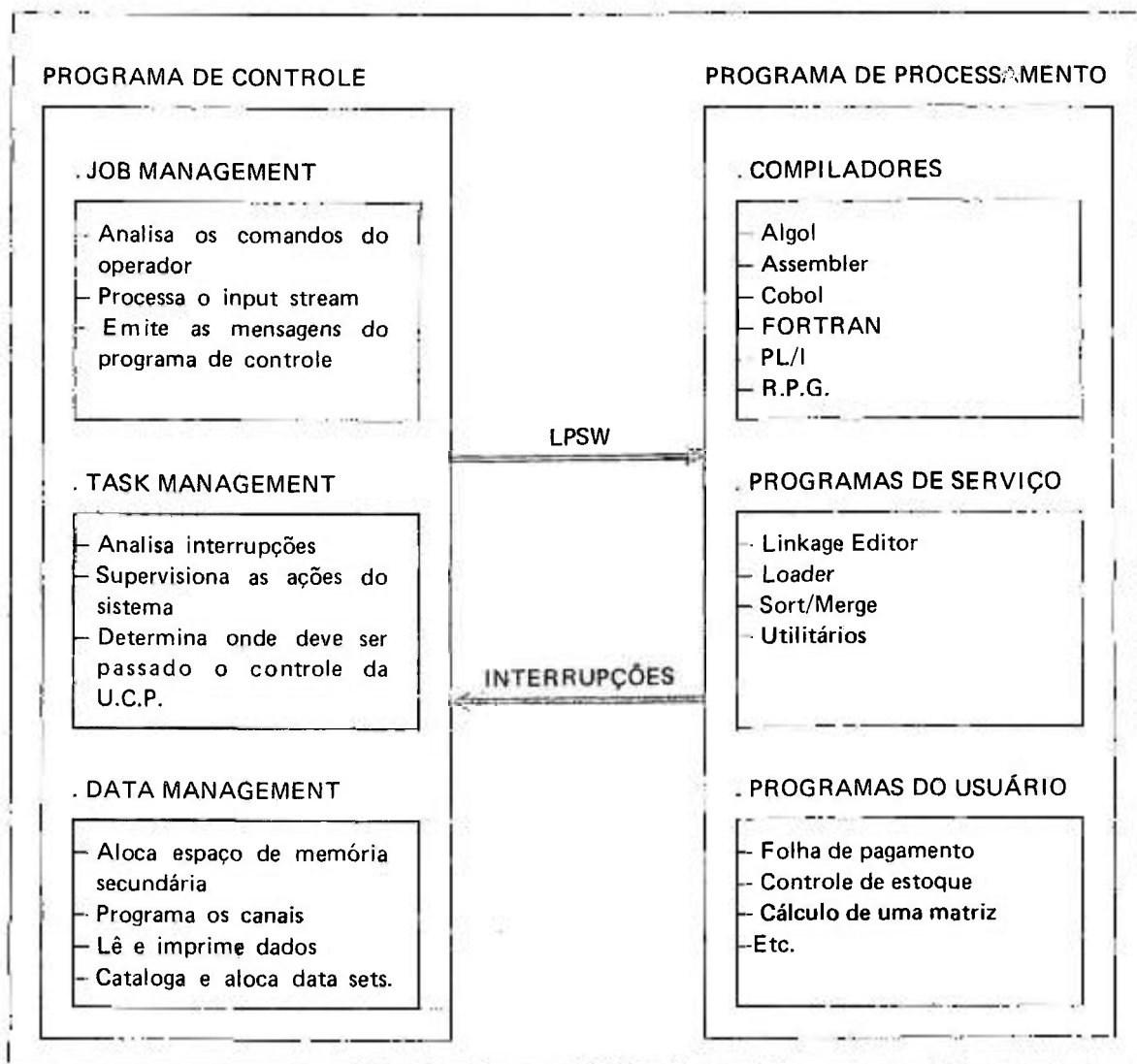


Figura 14 — Sistema Operacional / 360

3 – OPÇÕES DO PROGRAMA DE CÔNTROLE

Quando o usuário decide instalar um sistema em O.S., ele pode optar por dois aspectos fundamentais de programa de controle. O primeiro deles, o P.C.P. – PRIMARY CONTROL PROGRAM –, tem por característica a monoprogramação. O segundo se caracteriza por proporcionar multiprogramação e por sua vez se constitui de:

- M. F. T. – MULTIPROGRAMMING WITH A FIXED NUMBER OF TASKS
- M. V. T. – MULTIPROGRAMMING WITH A VARIABLE NUMBER OF TASKS

3.1. – P. C. P. (Primary Control Program)

É uma opção que oferece uma performance muito baixa e por isso utilizada quase que exclusivamente para orientar o usuário ao uso do O.S.. É, portanto, recomendado para sistemas não muito possantes no momento da opção, mas que virão a se-lo posteriormente. Desta forma o usuário já estará familiarizado com a terminologia e com a lógica do O.S. e os programas também estarão dentro do âmbito do O.S. e assim, no momento da expansão, as modificações a serem introduzidas não chegarão a afetar sobremodo o andamento do trabalho.

As características básicas do P.C.P. são:

- Monoprogramação
- Processamento seqüencial obedecendo a ordem do input stream
- Uso da lógica do O.S.

A área do sistema é dinâmica sendo utilizada para uma só task (ver Figura 15).

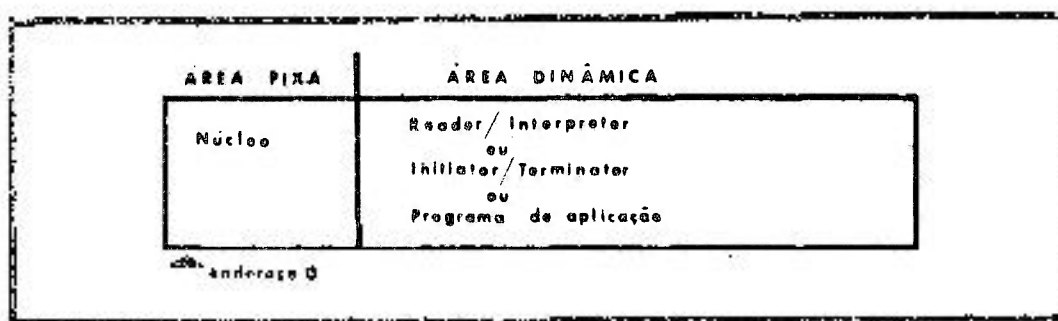


Figura 15 – SISTEMA P. C. P.

3.2 – Multiprogramação

Chamamos multiprogramação à execução concorrente de dois ou mais programas residindo simultaneamente na memória.

STORAGE PROTECTION — é um elemento de Hardware vital na multiprogramação pois impede que um programa de uma região ou partição invada outra região ou partição. Cada 2 K (2048) bytes de memória apresentam um byte no qual os 4 bytes de mais baixa ordem indicam a chave da região ou partição. Sempre que houver uma gravação na memória, a U.C.P. compara a chave da P.S.W. com a chave da região ou partição. Se forem iguais a gravação será executada, caso contrário haverá uma interrupção (Protection Exception).

Com 4 bits podemos representar os números de 0 a 15. As chaves de proteção de 1 a 15 são usadas pelo JOB. A chave 0 é usada pelo programa de controle. Isto explica porque apenas 15 programas podem residir na memória simultaneamente.

Vimos acima que podemos ter dois tipos de multiprogramação: um com número fixo de tarefas — M.F.T. — e outro com número variável — M.V.T. A diferença básica entre ambos diz respeito à alocação da memória.

Em M.F.T. a memória é alocada em partições de tamanho fixo estabelecido na Geração do Sistema, podendo ser alterado pelo operador no decorrer do processamento.

Em M.V.T. a memória é alocada em regiões de tamanho determinado pelo próprio JOB.

Tanto em M.F.T. como em M.V.T. a seleção dos JOBS é prioritária por classes. Os JOBS compondo o input stream são lidos, a partir dos equipamentos de entrada, por meio da rotina Reader/Interpreter do JOB MANAGEMENT e são selecionados em classes permanecendo em cada uma delas ordenadas segundo sua prioridade.

Existem 15 classes ou filas de entrada que residem numa biblioteca chamada SYS1.SYSJOBQE, geralmente no SYSRES, as quais são representadas pelas letras A até O. Nas filas de entrada são enviados apenas os cartões de controle e os dados correspondentes vão para um data set intermediário, também em disco, chamado INPUT DATA SET. (Ver Figura 16).

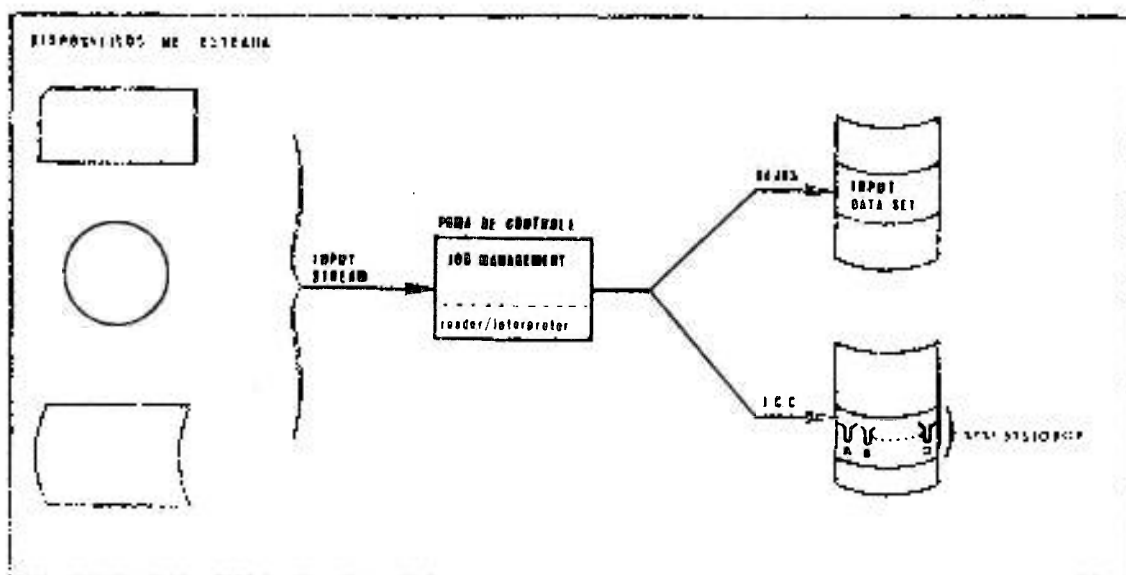


Figura 16 — Classes de Entrada

Assim como existem as classes ou filas de entrada existem também as classes ou filas de saída representadas pelas letras A a Z e pelos dígitos 0 a 9.

A finalidade das classes de saída é substituir equipamentos de baixa velocidade por equipamentos de alta velocidade com o intuito de evitar "ESTRANGULAMENTOS" na saída.

Imaginemos uma situação em que várias Tasks estejam sendo processadas concorrentemente e muitas delas necessitam de uma impressora como equipamento de saída. Por ser a impressora um dispositivo de saída lento (considerando a velocidade de processamento), as Tasks, à medida que vão necessitar utilizar-se da impressora, terão que aguardar que a mesma seja liberada por Tasks que a requisitaram anteriormente. Isto cria um verdadeiro estrangulamento na saída acarretando uma diminuição considerável na eficiência do sistema.

Para evitar isto são criadas as classes de saída e a cada uma delas, em tempo de geração, é feita a associação com um determinado equipamento de saída. Então, a medida que as Tasks vão sendo processadas, automaticamente vão sendo descarregadas nestas classes e daí, segundo a prioridade que tinham na entrada, são enviadas para equipamentos de saída requisitados.

Quem retira os dados das classes de saída e envia para os respectivos dispositivos de saída é outra rotina do JOB MANAGEMENT chamada Output Writer (ver Figura 17)

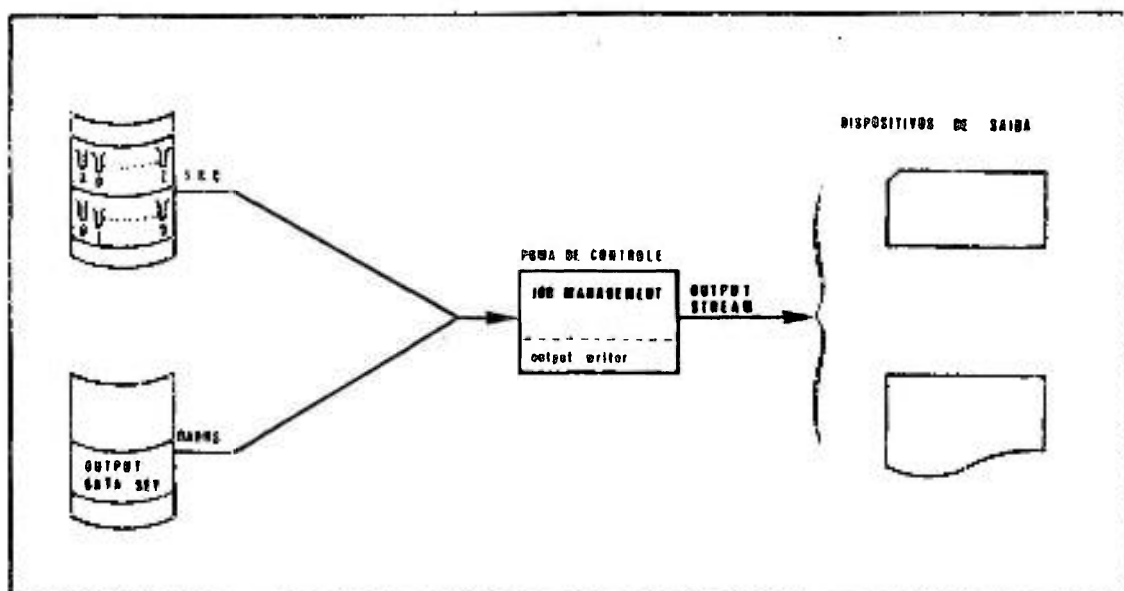


Figura 17 -- Classes de Saída

Embora existam as classes de SYSOUT, é possível alocar um determinado equipamento de saída exclusivamente para uma Task, através do uso de comandos do J.C.L.

3.2.1 – Multiprogramming With a Fixed Number of Tasks

Conforme vimos, a característica central do M.F.T. é a alocação da memória por partições ou alocação por bloco fixo.

O número máximo de partições em que pode ser dividida a memória é 52, representadas respectivamente por P_0, P_1, \dots, P_{51} , onde a partição P_0 é a de maior prioridade e se situa no mais alto endereço de memória (Figura 18). O número e o tamanho das partições é estabelecido na Geração do Sistema podendo ser redefinido a qualquer momento pelo operador.

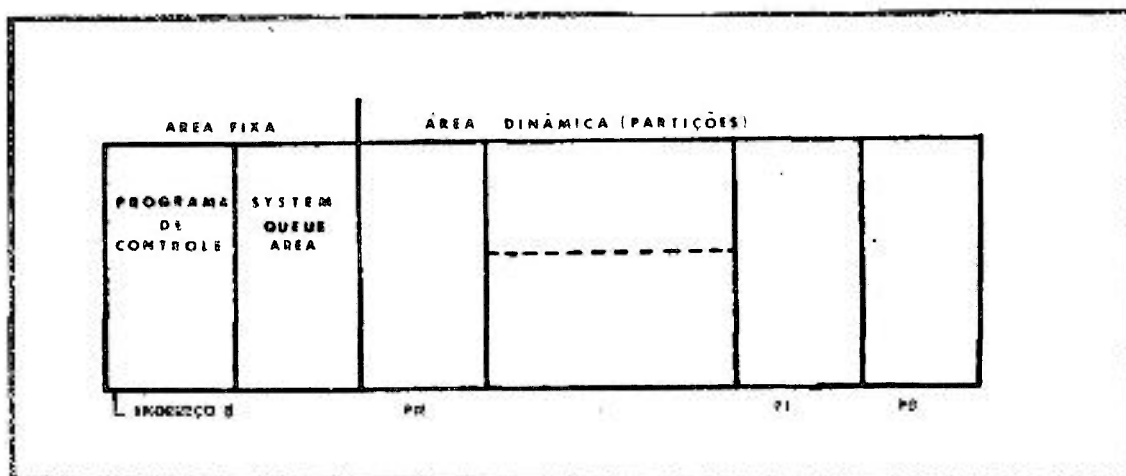


Figura 18 — Disposição de Memória em M. F. T.

As partições são independentes entre si e cada uma delas tem a possibilidade de processar até 3 classes do usuário ou uma Task do sistema.

No M.F.T. podemos ter até 15 Tasks do usuário e 39 Tasks do sistema assim distribuídos: 3 READERS e 36 OUTPUT WRITERS.

Cada Job tem sua prioridade especificada pelo usuário através do parâmetro PRTY do J.C.L. ou assumida por "default". Esta prioridade somente é considerada para a seleção do Job porque para execução a prioridade é dada pela partição em que o Job vai ser executado.

No caso em que dois Jobs tenham a mesma classe e o usuário tenha atribuído a ambos a mesma prioridade, o atendimento obedecerá à ordem cronológica de leitura.

Vimos portanto que em M.F.T. podemos ter 52 partições, cada uma das quais pode processar até 3 classes do usuário ou uma Task do sistema. Vimos também que dentro de cada classe a prioridade é atribuída pelo usuário através do parâmetro PRTY que pode variar entre 0 e 13 sendo 13 a maior delas.

Vejamos, por meio de um exemplo, como se dá o processamento de Jobs em M.F.T.

Na Figura 19 temos um esquema representando a memória dinâmica dividida em partições cada uma das quais com 3 classes atribuídas na geração ou pelo operador. Em baixo as classes da SYS1.SYSJOBQE contendo os Jobs a serem processados.

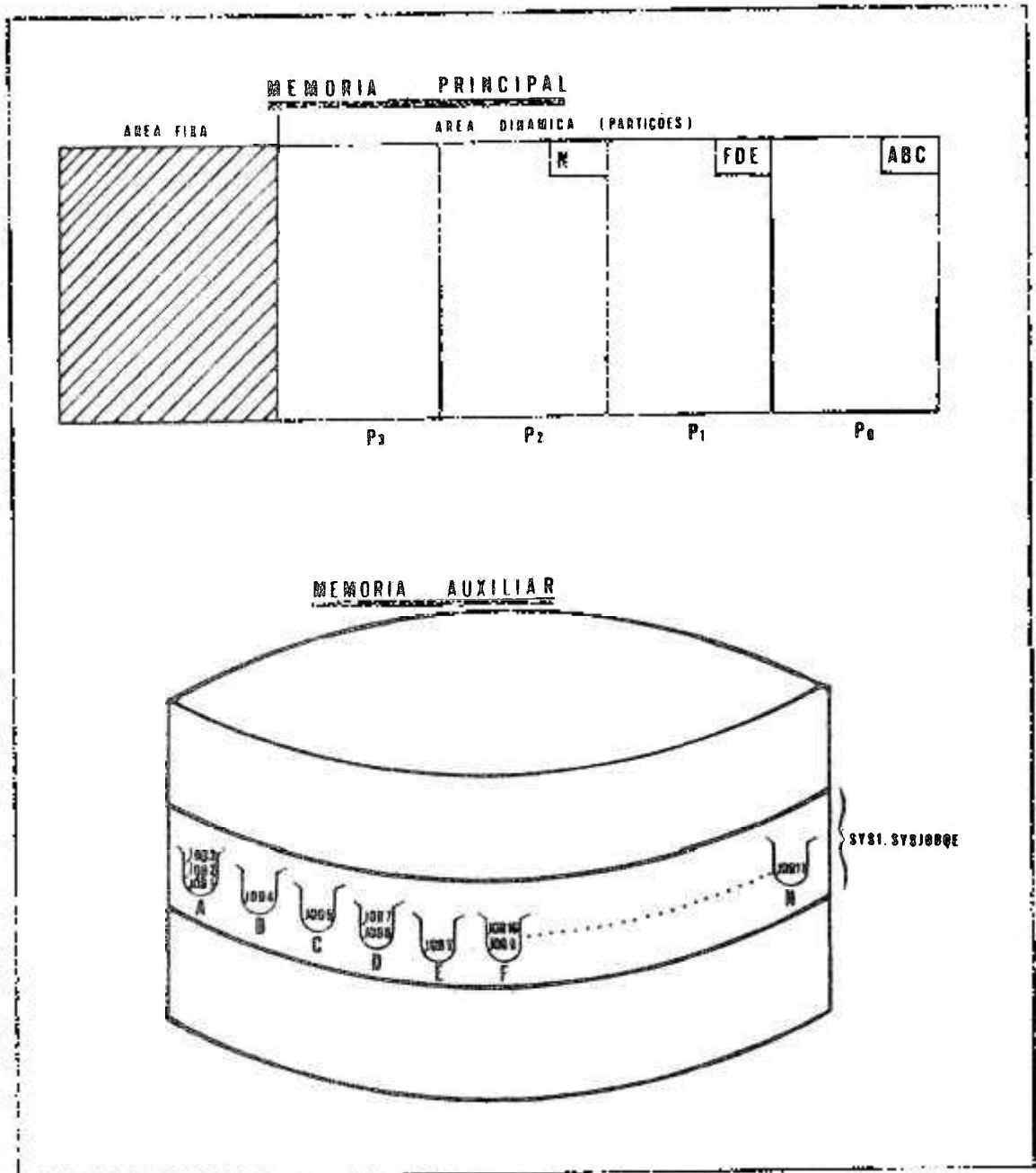


Figura 19 — Distribuição das Classes nas Partições

Cada partição tem, associado a ela, um INITIATOR, ao qual cabe selecionar os Jobs das classes onde se encontram, alocando os recursos necessários à sua execução e após isto passar o controle ao programa.

No nosso exemplo o INITIATOR da partição P0 “verifica” que A é a primeira classe da partição a ser processada. Feita esta verificação, procura dentro da classe o JOB de maior prioridade (prioridade atribuída via parâmetro PRTY ou prioridade cronológica), que é, digamos, o JOB 1, para ser executado. Quando o JOB 1 terminar será selecionado o segundo JOB (segundo em prioridade) da classe A, no caso JOB 2. Quando este também terminar será a vez do JOB 3 e com isso a Classe A ficará vazia. Nesse momento o INITIATOR da partição P0 passará a retirar os Jobs da classe B levando-os para a partição P0, para que sejam executados usando o mesmo critério para os Jobs de classe A.

Este processo será repetido por todos os Jobs de todas as classes, partição por partição.

Em M.F.T., normalmente, os programas do tipo I/O BOUND (programas limitados por entrada e saída) são colocados nas partições de maior prioridade e os do tipo COMPUTER BOUND (programas limitados por processamento) são colocados nas de menor prioridade. Isto para permitir às partições de menor prioridade aproveitarem os tempos de espera dos programas limitados por entrada e saída.

Uma característica importante do M.F.T. é a possibilidade de criação de pequenas partições para executar pequenos Jobs. Estas partições são conhecidas como SMALL PARTITIONS e o seu tamanho varia entre 8K e o tamanho do INITIATOR, devendo ser sempre múltiplas de 2. (O INITIATOR em M.F.T. pode apresentar dois tamanhos: 30 K ou 44 K). Isto quer dizer que o INITIATOR não cabe nestas partições devendo, os Jobs, serem inicializados em partições “NO SMALL”, por INITIATORS destas, e após a inicialização serem devolvidos às respectivas SMALL PARTITIONS.

Outra característica do M.F.T. diz respeito à possibilidade de especificar, na geração do sistema, a existência de “SUBTASKING”.

SUBTASKING é o fenômeno pelo qual uma TASK (chamada TASK mãe) cria outra TASK ou TASKs (chamada TASK filha) que concorrerá com a primeira pelos recursos da U.C.P.

A SUBTASK é criada via ATTACH e removida via macro DETACH e o número máximo de subtasks que podem ser criadas em M.F.T. varia entre 196 e 243.

RESUMO DO M. F. T.

- * Multiprogramação com número fixo de tarefas
- * Programação por prioridades
- * Número fixo de partições (até um máximo de 52)
- * Até 15 Tasks do usuário – chave de 1 a 15 –
- * Até 39 Tasks do sistema (3 Readers e 36 Writers) – chave 0
- * Partições programadas independentemente
- * Pequenas partições (SMALL PARTITIONS)

- * Possibilidade de redefinição das partições durante a operação
- * Até 3 classes por partição
- * Prioridades dentro de classes
- * Subtask por partição

3.2.2. — M. V. T. — Multiprogramming With a Variable Number of Tasks

A característica central do M.V.T. é a alocação da memória de forma dinâmica, em regiões, cujo tamanho é especificado pelo próprio JOB. A memória é portanto um recurso disputado dinamicamente pelas Tasks em execução.

A condição para que um Job seja alocado, é que o sistema disponha de memória adjacente suficientemente grande para a necessidade do Job. Se isto não acontecer o Job permanece em estado de espera até conseguir a área adjacente necessária.

A alocação das regiões se faz do mais alto endereço de memória para o mais baixo e a disposição da memória é aquela mostrada na Figura 20.

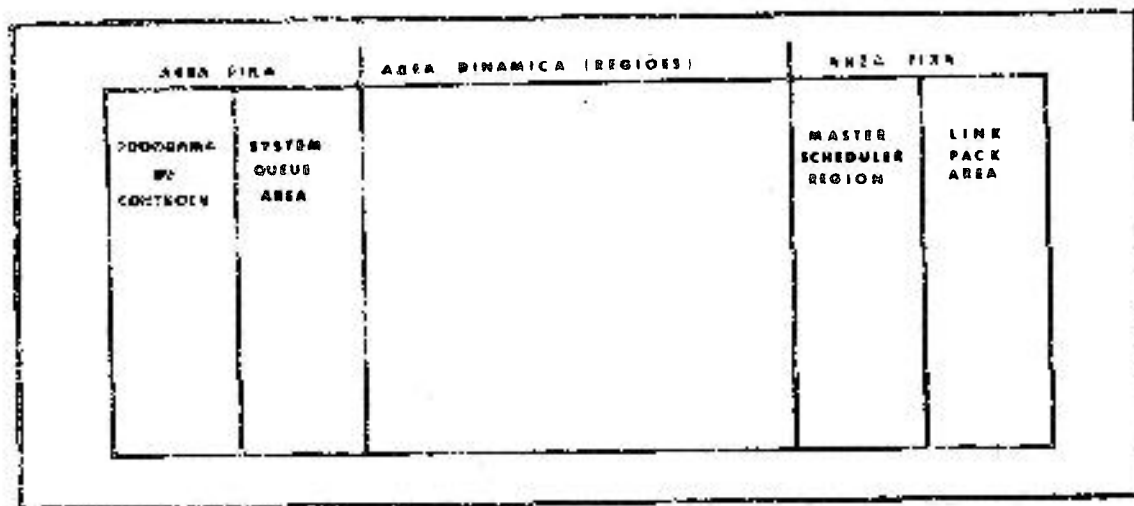


Figura 20 — Disposição da Memória em M. V. T.

Em M.V.T. quando é dado o START para o INITIATOR, são especificadas diretamente quais classes serão atendidas pelo mesmo, sendo que, cada INITIATOR, pode ter associadas a ele até 8 classes. Daí em diante o processo é similar ao M.F.T. isto é: o INITIATOR liga-se ao Job mais prioritário de cada classe, alocando-lhe recursos, inicializando-o e esperando o mesmo terminar. Então verifica se aquela classe tem outros Jobs para serem processados, caso não tenha, serão retirados Job da segunda classe associada ao mesmo INITIATOR. E assim sucessivamente até esgotar todas as classes.

Diferentemente do M.F.T., em M.V.T. a prioridade de execução é a mesma daquela de seleção. Em outras palavras: a prioridade estabelecida pelo usuário, através dos parâmetros PRTY E DPRTY, é utilizada para seleção do Job e execução da TASK respectivamente.

Existe a possibilidade de alterar a prioridade de execução de um Job Step através da macro CHAP – Change Dispatching Priority – ou através do parâmetro DPRTY no cartão EXEC do J.C.L., podendo desta forma, atribuir a certos Job Steps prioridade diferente daquela atribuída inicialmente ao JOB. A prioridade de execução varia de 0 a 15 sendo 15 a maior e geralmente reservada para Tasks do sistema.

A DISPATCHING PRIORITY é um valor através do qual um programa luta pela U.C.P.. Em M.F.T. está associado à partição. Em M.V.T. está associado ao número fornecido por cartão de controle.

Em M.V.T. temos também a possibilidade de criar SUBTASKS, não havendo limitação definida previamente quanto ao número das mesmas, dependendo apenas dos recursos existentes.

Existem muitas opções do M.V.T., opções estas que conferem ao M.V.T. uma performance otimizada além de prestarem ajuda ao usuário que delas se utilizam.

Dentre todas existem três que nos parecem mais importantes e é destas três que apresentaremos algumas informações suficientes para caracterizá-las e conhecê-las, mas, insuficientes para delas nos utilizarmos. Para isso terão que ser consultados manuais apropriados.

Trata-se das opções:

I – TIME SLICING

II – ROLLOUT / ROLLIN

III – T. S. O.

I – TIME SLICING e TIME-SLICE GROUP

Time Slicing é o tempo máximo que um programa pode ficar consecutivamente sob controle da U.C.P. Este tempo é determinado na geração por meio da macro-instrução CTRLPROG, mas pode ser modificado ou mesmo cancelado em tempo de inicialização do sistema.

Vejamos através de um exemplo como funciona esta opção:

```
CTRLPROG    TYPE = MVT, TMSLICE = (9, SLC - 210, 5, SLC - 480)
```

Isto quer dizer que: 2 "TIME-SLICE GROUPs" foram definidos. Todos os Jobs de prioridade 9 sofrerão "TIME SLICE" cada um deles tendo direito a 210 milissegundos de U.C.P. Após este grupo de TIME-SLICE, sofrerão "TIME SLICE" os Jobs de prioridade 5, cada um deles com direito a 480 milissegundos de U.C.P.

Quem limita o tempo máximo disponível para cada Task do grupo de TIME-SLICE é o SUPERVISOR, que volta a devolver-lhe o controle somente após passar, seqüencialmente, por todas as outras Tasks do grupo.

II – ROLLOUT / ROLLIN

É a capacidade que o sistema possui de, sob certas condições, poder conseguir memória adicional para um certos Job Step que dela necessite, rolando para fora da memória um outro Job Step e utilizando a memória deste. O Job Step rolando vai para um "data set" chamado SYS1.ROLLOUT.

As condições são:

- a) – O JOB STEP a ser rolado deve ocupar uma área contínua à do JOB STEP que vai rolá-lo.
- b) – O JOB STEP a ser rolado deve ter prioridade menor ou igual à do JOB STEP que vai rolá-lo.
- c) – O JOB STEP que necessita de memória adicional deve ter a capacidade de poder rolar o outro.
- d) – O JOB STEP que vai ceder sua área de memória deve ter a capacidade de ser rolado.

As capacidades de rolar e/ou ser rolado são fornecidas pelo parâmetro ROLL do J.C.L., o qual possui dois subparâmetros que indicam se o Job Step pode ser rolado fora e se o Job Step pode rolar outros Job Steps, nesta ordem.

O formato do parâmetro em questão é:

ROLL = (x,y) onde:

x indica a possibilidade do Job Step ser rolado fora. Se x for substituído por YES significa que o Job Step pode ser rolado fora. Se x for substituído por NO o Job Step não pode ser rolado.

y indica a possibilidade do Job Step rolar outro Job Step. Se y for substituído por YES, o Job Step pode rolar outro. Se y for substituído por NO, o Job Step não pode rolar outro.

O parâmetro ROLL pode ser codificado no cartão JOB valendo para todos os Jobs Steps daquele JOB ou pode ser codificado apenas no cartão EXEC valendo portanto apenas para aquele Job Step.

III – T. O. S. – TIME SHARING OPTION

Esta opção acrescenta ao O.S. a possibilidade de compartilhar tempo conversacional no uso de terminais remotos. Desta forma, em breves períodos de tempo, cada terminal terá sua comunicação com o sistema. Considerando que o terminal é lento em relação ao sistema e considerando também que o tempo de espera de cada terminal é curto, a impressão que o usuário tem é que seu terminal está sempre em comunicação com o sistema.

A principal vantagem do T.S.O. é sua simplicidade de utilização e esta simplicidade decorre de fatores tais como:

- meio de comunicação fácil de usar por tratar-se de um teclado semelhante ao das máquinas de escrever convencionais. Via teclado o usuário tem acesso a todas as possibilidades que teria no processamento por lotes (batch).
- o trabalho é definido no terminal por meio de linguagem simples e natural. Os comandos de Time Sharing são palavras em inglês que descrevem a função desejada.
- se a sintaxe de algum comando ou subcomando for esquecida, basta teclar o comando HELP que a descrição é fornecida.

- teclando um “?” o usuário recebe maiores detalhes sobre mensagens do sistema que não tenham sido devidamente entendidas.

Outras vantagens são:

- permite o uso simultâneo por número bem maior de usuários do que usando multiprogramação (batch)
- o usuário tem controle sobre o seu JOB constantemente, Step por Step, recebendo resposta para cada ação e sendo notificado dos erros que o sistema detetou.
- o “turnaround – time” é reduzido graças às informações imediatas.

RESUMO DO M. V. T.

- * Multiprogramação com número variável de tarefas.
- * Programação por prioridades.
- * Número de regiões limitado apenas pelo tamanho da memória.
- * Até 15 programas do usuário.
- * READERS e WRITERS limitadas apenas pela capacidade da memória e/ou dispositivos de E/S.
- * Memória alocada dinamicamente para cada etapa ou serviço.
- * Mudança de prioridade dinâmica.
- * Até 8 classes por INITIATOR.
- * ROLLOUT / ROLLIN.
- * SUBTASK por regiões.
- * Time Slicing.
- * T.S.O.

4 – CARACTERÍSTICAS DO O. S.

Neste ítem serão apresentadas, de forma conceitual apenas, as principais características do O.S., mesmo aquelas que já foram abordadas ou que o serão mais adiante. Isto porque a finalidade deste ítem é reunir as características do O.S. de modo a dar uma visão de conjunto do mesmo bem como de sua potencialidade.

* **ALIAS** – são apelidos que podem ser atribuídos, até um máximo de 16, para cada data set ou um membro de um data set.

* **ALOCAÇÃO DE “BUFFERS”** – o Buffer não está preso ao arquivo, podendo ser usado por vários data sets, desde que dele se utilizem um de cada vez.

* **ARQUIVOS "DUMMY"** — são arquivos que permitem testar rotinas sem que haja leitura ou gravação física, ou melhor, a leitura e a gravação são simuladas.

* **BIBLIOTECAS EM NÚMERO ILIMITADO** — a criação de um número ilimitado de bibliotecas se prende ao fato de as mesmas serem consideradas pelo sistema como simples data sets (arquivos).

* **BPAM — BASIC PARTITIONAL ACCESS METHOD** — é um método de acesso de suporte para trabalhar com data sets particionados.

* **CATALOGAÇÃO DE ARQUIVOS** — o sistema, mantém no seu catálogo — **SYSCTLG** —, o nome dos data sets associados ao volume e tipos de dispositivos. Para recuperar data sets catalogados é suficiente fornecer o nome dos mesmos (**DSNAME**).

* **CHECK POINT RESTART** — é a capacidade de, através da Macro "Check Point", gravar de tempo em tempo um arquivo em imagem da situação atual do programa, de modo que, em caso de acidente, seja possível recomençar o programa a partir do último Check Point e não desde o início.

* **COMPILADORES PODEROSOS** — compiladores que oferecem maior número de facilidades e opções que resultam em maior rapidez.

* **CONTROLE DE ÁREAS DE MEMÓRIA E ROTINAS CARREGADAS** — é o controle, efetuado pelo sistema, das rotinas já carregadas para que possam ser usadas por qualquer programa na memória, quando necessário.

* **DASD — DIRECT ACCESS DEVICE SPACE MANAGEMENT** — são rotinas do **DATA MANAGEMENT** cuja função é procurar e alocar espaço em **DASD**, para arquivos, automaticamente.

* **D.C.B. — DATA CONTROL BLOCK** — é uma tabela dentro do programa do usuário que contém informações a respeito do arquivo. O **OPEN** ao perceber que uma **DCB** não está preenchida, preenche-a retirando dados do parâmetro **DCB** do cartão **DD** e, quando necessário, do próprio **LABEL** (nesta ordem de prioridade).

* **DESIGNAÇÃO AUTOMÁTICA DE UNIDADES FÍSICAS** — a designação das unidades de entrada e saída é feita automaticamente pelo sistema.

* **DISP = SHR, DISP = OLD** — **DISP** é um parâmetro do cartão **DD** que descreve para o sistema o 'status' do data set e indica o que deve ser feito com ele após o término do Job Step que o processa ou após o término do Job. Se atribuímos o valor **SHR** ao parâmetro **DISP** significa que o data set pode ser compartilhado simultaneamente por outros Jobs. Atribuindo o valor **OLD**, o data set será usado somente por este Job.

* **ENQUE E DEQUE DE PROGRAMAS** — são blocos de controle na **SYSTEM QUEUE** ÁREA para associação (**ENQUE**) de recursos à **TASK** e liberação (**DEQUE**) posterior.

* **FILAS DE RECURSOS (RESOURCE QUEUE)** — são os recursos alocados automaticamente e otimizados pelo **O.S.** em forma de filas ou classes e onde é especificado qual recurso está sendo requisitado bem como qual programa o está requisitando. Estas filas são pesquisadas sempre que houver uma interrupção qualquer.

* **G.D.G. — GENERATION DATA GROUP** — é uma característica especial para designar data sets catalogados que são gravados periodicamente. Cada criação do data set é considerada uma geração, tendo um número associado à mesma.

***JOB E STEP LIMIT** — é uma característica utilizada para forçar o término de um JOB ou de um JOB STEP ao final de um certo tempo. Com isto é possível prevenir eventuais 'Loops' de programas

***LINK PACK AREA** — é uma porção de memória que pode ser utilizada para carregar as rotinas mais freqüentemente usadas durante o dia, economizando tempo de processamento.

***M.C.S. — { MULTIPLE CONSOLE SUPPORT
MASTER CONSOLE SYSTEM** — conjunto de rotinas que suportam a existência de várias consoles ON-LINE, havendo inclusive a separação de mensagens e comandos para cada console.

***MÉTODOS DE ACESSO** — rotinas que realizam a parte de entrada e saída de responsabilidade do usuário. No O.S. existem duas técnicas de acesso: QUEUED e BASIC, cada uma das quais combinada com a organização do arquivo forma o método de acesso.

***MSGCLASS** — é um parâmetro que permite separar as listagens dos cartões de controle das demais, dirigindo-as para determinada classe.

***MULTIPROCESSAMENTO** — é a característica pela qual, através da utilização de um dispositivo de Hardware, dois computadores, distintos entre si, compartilham suas memórias.

***MULTIPROGRAMAÇÃO** — é a característica pela qual é possível executar concorrentemente dois ou mais programas residindo simultaneamente na memória. Em O.S., com exceção da opção P.C.P., é possível processar até 15 programas do usuário.

***MULTITASK** — é a característica que uma TASK tem de, através de um MACRO ATTACH, ser subdividida em TASKS menores e independentes, executadas em uma mesma região ou partição

***PASSWORD** — são palavras de segurança — senhas — usadas para proteger arquivos confidenciais. Para funcionar esta proteção é preciso que haja um arquivo seqüencial chamado PASSWORD, residente no SYSRES, contendo todas as senhas de arquivos confidenciais da instalação.

***PRIORIDADE** — é a característica que permite processar Jobs em ordem diferente daquela que os mesmos tinham no Input Stream.

***PROCEDIMENTOS CATALOGADOS** — são conjuntos de cartões de controle, necessários na realização de alguma função, gravados em forma de membros de um data set particionado denominado SYS1.PROCLIB.

***PROGRAMAÇÃO MODULAR** — é uma filosofia de programação que defende a idéia que um programa deve ser cindido em várias subrotinas fechadas, cada uma das quais podendo ser codificada na linguagem mais própria, que serão reunidas posteriormente num só programa pelo Linkage Editor.

***PROGRAMAS RELOCÁVEIS** — são programas que podem ser carregados a partir de qualquer endereço na memória principal. Uma vez carregado porém, o programa ficará com endereços fixos, devendo ser processado até o final nos mesmos.

***QUALIFICAÇÃO** — é a característica que possibilita a atribuição de dois ou mais nomes simples ligados por ponto, formando um novo nome denominado nome qualificado. Ex: SYS1.PROCLIB, DEPT.SEC.GRUPO.

***REGION** — é um parâmetro que determina a quantidade de memória necessária para a alocação do JOB. O espaço é dado em K bytes.

***REGISTROS SPANNED** – são registros em que o fator de bloco é menor que 1 ou, em outras palavras, um registro lógico é constituído de mais de 1 bloco ou ainda, os registros lógicos são maiores que os registros físicos.

***REMOTE JOB ENTRY** – é uma característica que permite formar Input Streams em terminais remotos e executar Jobs.

***ROLLOUT / ROLLIN** – é a característica que o sistema possui de, em certas condições, poder conseguir memória adicional para um Job que dela necessita, retirando provisoriamente um outro Job da memória e cedendo esta para o Job que a necessita.

***S.M.F. – SYSTEM MANAGEMENT FACILITIES** – é um sistema automático de coleta de informações referentes à utilização dos recursos (dispositivos) para cada um dos programas processados. Ex: utilização de U.C.P., de memória, de canal, de área em disco, tempo de espera, etc. As informações são guardadas em dois arquivos:

. SYS1.MANX – primário

SYS1.MANY – alternativo

***“SPOOL” AUTOMÁTICO COM “READERS” E “WRITERS”** – permite eliminar os pontos de “estrangulamento” do sistema, quais sejam: a leitora de cartões (Reader) e a impressora (Writer).

***TIME SLICING** – é o tempo máximo permitido a um programa para ficar consecutivamente sob o controle da U.C.P.

***TIMER** – controle interno do tempo do sistema ao qual todas as TASKS tem acesso.

***T.S.O. – TIME SHARING OPTION** – é a característica que possibilita um atendimento seqüencial de terminais de telecomunicações, com tempo médio para cada um.

***TYPRUN = HOLD** – é um parâmetro do J.C.L. que permite, ao JOB que o possui, ficar preso na fila até ser liberado pelo operador ou por um “release” automático.

5 – CONCEITOS DE DATA MANAGEMENT

5.1 – Formatos de Registros

O O.S. pode suportar registros de formato: FIXO, VARIÁVEL e INDEFINIDO. Com exceção do indefinido ou outros dois podem ser bloqueados ou não bloqueados. Além disso o formato variável apresenta uma variação que é a spanned, também bloqueada ou não. Considerando isto tudo podemos esquematizar os tipos de formatos de registros como:

– F – FIXO

– FB – FIXO BLOCADO

– V – VARIÁVEL

– VB – VARIÁVEL BLOCADO

– U – INDEFINIDO

– VS – VARIÁVEL SPANNED

– VBS – VARIÁVEL BLOCADO SPANNED

Com relação ao bloqueamento ou não de registros devemos considerar o seguinte:

- 1º) – Bloqueamento é a reunião de dois ou mais registros lógicos num só bloco ou registro físico, contrariamente ao não bloqueamento onde cada registro lógico é o próprio registro físico.
- 2º) – Os registros físicos são separados entre si por intervalos denominados IBG (INTER BLOCK GAP), exceção feita ao equipamento UNIT RECORD.
- 3º) – O registro físico é levado para a memória (Buffer) como um todo pelo PIOCS (Physical Input Output Control System) e lá é desbloqueado pelo LIOCS (Logical Input Output Control System) para que cada registro lógico seja processado.

A partir destes três itens podemos concluir que o bloqueamento proporciona maior velocidade de processamento e maior aproveitamento de memória auxiliar. Se tivermos, por exemplo, 5 registros lógicos desbloqueados, em disco, e quisermos processá-los, o PIOCS terá que transportá-los, um de cada vez, para a memória. Com isso, por 5 vezes, o disco tem que desacelerar e depois retomar a velocidade normal. Se, por outro lado, os 5 registros estiverem num só bloco, haverá apenas uma desacelerada na qual o PIOCS levará o bloco todo para a memória deixando ao LIOCS a tarefa de desbloqueagem.

Desta forma verificamos que o bloqueamento é mais eficiente quanto à rapidez.

Quanto ao aproveitamento de memória auxiliar, basta lembrar que o bloqueamento diminui os IBGs e que cada IBG ocupa um certo número de bytes – cerca de 480 bytes, variando bastante de acordo com um conjunto de fatores – cada IBG eliminado corresponde a um certo número de bytes economizados.

Como vimos, o bloqueamento de registros oferece duas vantagens que, em processamento de dados, são fundamentais: tempo de processamento e espaço de memória auxiliar menores.

Claro está que na criação de blocos devem ser considerados também outros fatores importantes entre os quais, a título de lembrete, podemos incluir o tamanho da memória principal. Isto porque, conforme vimos, o bloco é carregado inteiro para a memória e um bloco demasiadamente grande pode criar certos problemas com relação a Buffers, etc.

Com relação a desvantagens do bloqueamento de registros, podemos afirmar que, dentro de situações normais, elas não existem, porém, nos casos em que o tempo do processamento do registro lógico for muito pequeno – menor do que o tempo que o LIOCS leva para efetuar a desbloqueagem – o bloqueamento é desvantajoso. Neste caso é preferível deixar os registros desbloqueados e economizar o tempo de desbloqueagem que no caso seria maior do que o tempo de processamento do registro.

Outra consideração a ser feita diz respeito à escolha de um ou de outro formato de registros. Esta escolha deve ser feita levando em conta fatores tais como:

– A natureza do Data Set

– O método de acesso a ser utilizado

- O tipo de entrada e saída do programa
- Limitações do equipamento
- Facilidade de programação
- Velocidade de transferência
- Etc.

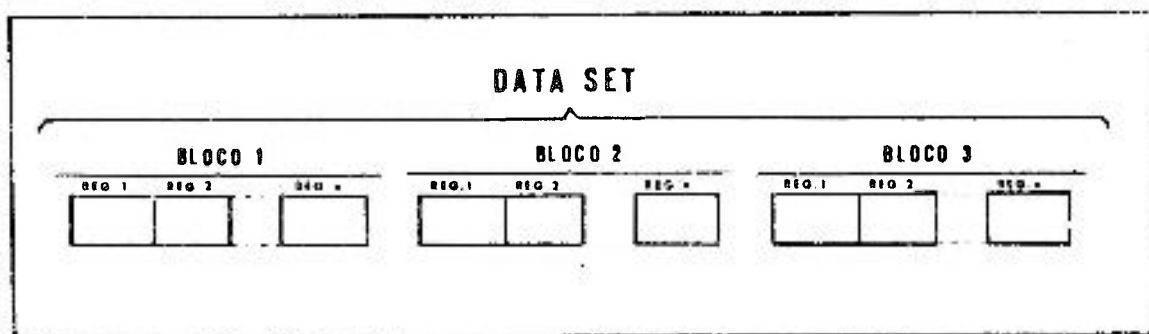


Figura 21 — Organização de Dados

5.1.1 — Registros de Formato Fixo

São chamados registros de formato fixo aqueles registros em que o tamanho não varia dentro do data set.

Podem ser bloqueados (FB) ou não bloqueados (F).

Os não bloqueados são aqueles em que o registro físico contém apenas um registro lógico.

Os bloqueados são aqueles em que o registro físico contém mais de um registro lógico e o número de registros lógicos contidos no registro físico é constante em todo o data set, podendo diferir no último devido a um eventual truncamento resultante da insuficiência de registros lógicos para preencher o registro físico.

Cada registro lógico pode conter, como caracter inicial, um caracter de controle para seleção de escaninho ou controle de carro.

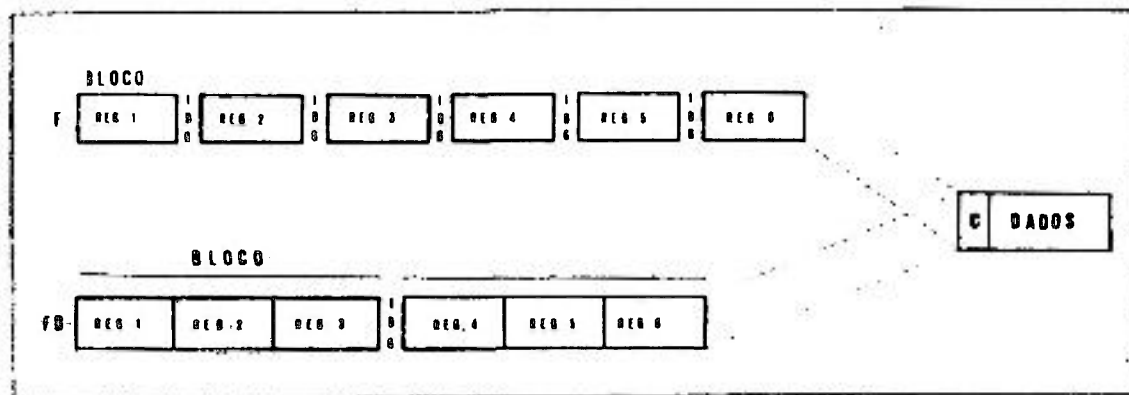


Figura 22 — Registros de Formato Fixo

5.1.2 — Registros de Formato Variável

Neste caso tanto o registro lógico como o registro físico podem apresentar tamanho variável.

Antes de cada registro lógico existe uma área de 4 bytes chamada RDW — RECORD DESCRIPTOR WORD — na qual os 2 primeiros bytes fornecem o tamanho de registro lógico, mais o comprimento do RDW. Os restantes 2 bytes são utilizados apenas para os registros variáveis SPANNED.

Temos também uma área de 4 bytes para cada registro físico, chamada BDW — BLOCK DESCRIPTOR WORD —, onde os 2 primeiros bytes fornecem o tamanho do registro físico, mais o tamanho da BDW e os dois últimos são reservados para o sistema.

Podemos ter registros de formato variável bloqueado e não bloqueado conforme um registro físico contenha mais de um registro lógico ou apenas um, respectivamente.

Quando forem usados os registros de formato variável, é preciso informar ao sistema os limites mínimo e máximo do tamanho do registro. Além disso as áreas de entrada e saída bem como os Buffers devem ter tamanho suficiente para conter também os RDW e os BDW.

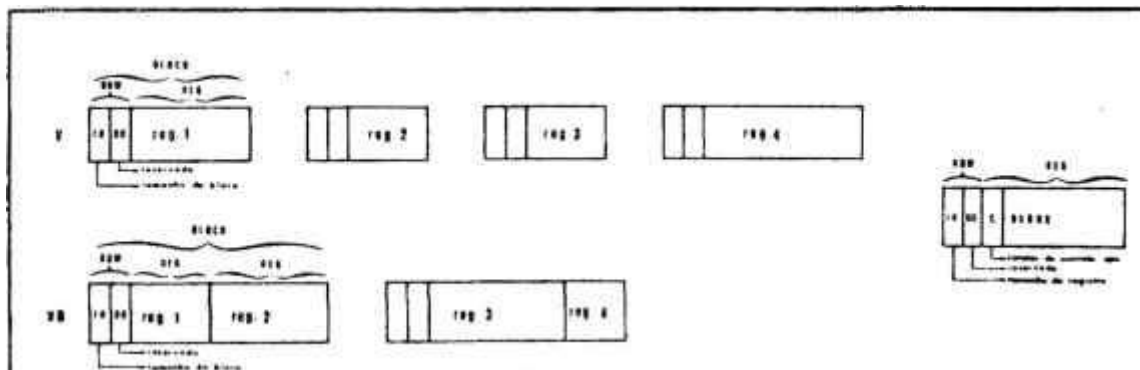


Figura 23 — Registros de Formato Variável

5.1.2.1 – Registros de Formato Variável Spanned

Trata-se de uma variação do formato variável que permite ignorar certas restrições físicas de dispositivo ao se projetar o registro lógico. Tais registros só podem ser usados em acesso seqüencial e sua característica é a de permitir que o registro lógico seja maior que um bloco físico. Isto é conseguido fracionando o registro lógico em partes chamadas SEGMENTOS que ocupam mais de um bloco.

O registro físico contém o campo RDW ocupando seus 4 primeiros bytes. Cada segmento, por sua vez, contém uma área de 4 bytes chamada SDW – SEGMENT DESCRIPTOR WORD – em cujos primeiros 2 bytes é dado o tamanho do segmento. O terceiro byte contém o SEGMENT CONTROL CODE que especifica a posição relativa do SEGMENTO e o quarto byte é de uso do sistema.

Os registros variáveis SPANNED também podem ser bloqueados ou não bloqueados. Se não for bloqueado – VS – o registro físico é composto de um segmento mais os RDW e SDW. Se for bloqueado – VBS – o primeiro segmento de um registro lógico poderá estar contido no mesmo registro físico que o último segmento do registro lógico precedente.

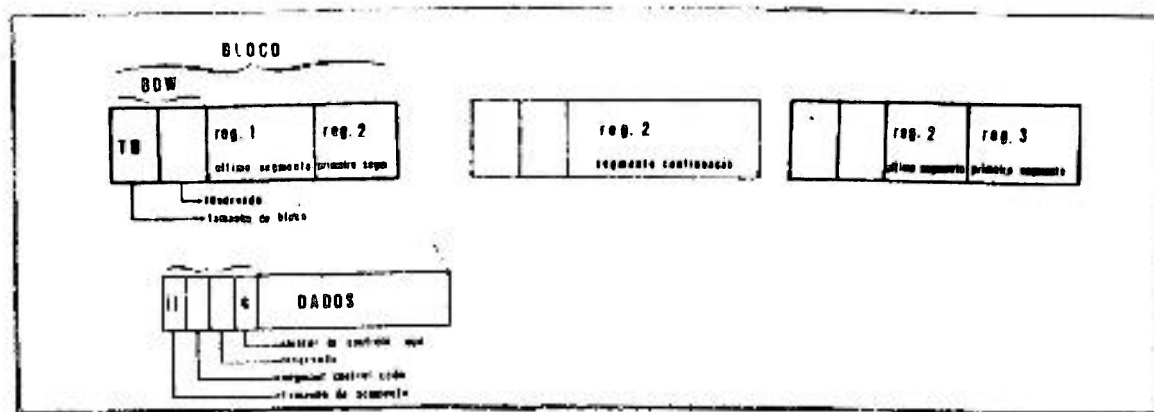


Figura 24 — Registros de Formato Variável "SPANNED"

5.1.3 – Registros de Formato Indefinido

São registros que não se enquadram no formato Fixo nem no formato Variável. Cada bloco é tratado como registro e, conseqüentemente, o bloqueamento e o desbloqueamento são efetuados pelo programa do usuário não havendo verificação de tamanho por parte do sistema.

Um registro desse tipo pode conter, como primeiro byte, um caracter de controle, mas não conterá nem RDW nem BDW a serem controlados pelo sistema.

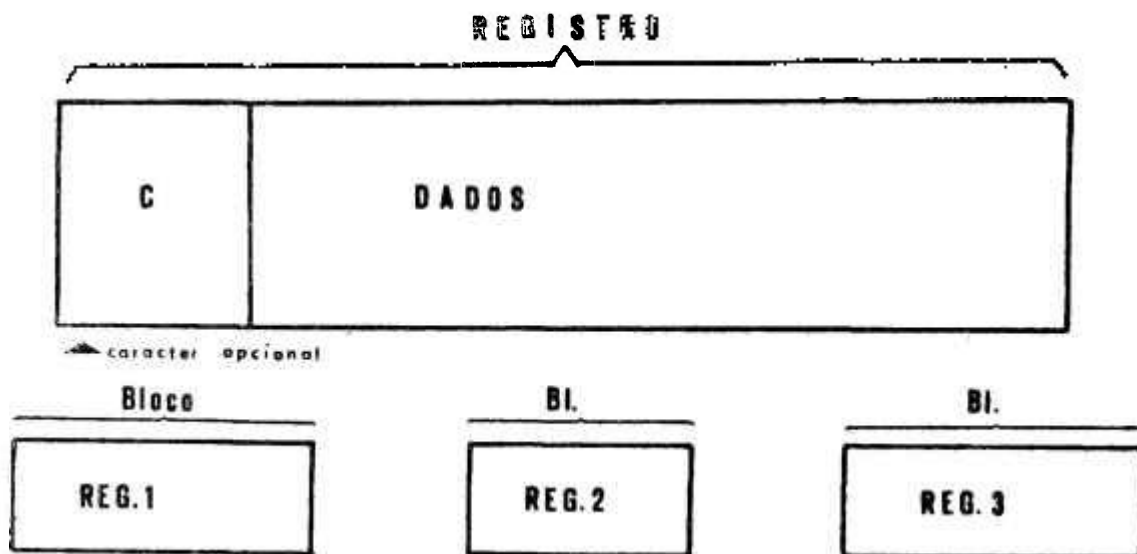


Figura 25 — Registros de Formato Indefinido

5.2 — Organização de Data Sets

Um data set deve ser constituído de registros organizados logicamente para que seja possível sua recuperação eficiente para processamento.

Por organização de um data set entendemos a maneira como os registros estão agrupados dentro deste data set.

Existem quatro maneiras de organizar os registros dentro de um data set, conforme veremos mais adiante, e a escolha de uma ou outra deve ser efetuada considerando as seguintes características de um data set:

* **VOLATILIDADE** — é a característica de um data set que se refere à adição e subtração de registros no data set. Um data set que apresenta baixo índice de adições e subtrações é chamado de estático. Contrariamente, o data set que apresenta um alto índice de adições e subtrações é chamado de volátil.

* **ATIVIDADE** — Refere-se à percentagem de registros do data set a serem processados. Um data set que apresenta uma percentagem de registros a serem processados baixo é um data set pouco ativo e neste caso a sua organização deveria ser tal que os registros fossem localizados diretamente. Já no caso de um data set muito ativo, a quase totalidade de seus registros é processada e portanto uma organização que permitisse localizá-los seqüencialmente seria satisfatória. É preciso considerar também como é distribuída esta atividade de forma a permitir que os registros mais freqüentemente processados sejam localizados mais rapidamente que os demais.

* **TAMANHO** — É outro fator a ser considerado na organização de um data set. Se o data set for extremamente pequeno, o tipo de organização do mesmo pouco importará. Por outro lado o data set pode ser tão grande que os registros que o compõe não possam estar disponíveis para o sistema ao mesmo tempo. Além disso, ao organizar um data set, é preciso ter em mente também o crescimento potencial do mesmo para evitar de, após um certo tempo, ter de reorganizá-lo.

* **SUPORTE** – É importante conhecer as características dos dispositivos que vão servir de suporte ao data set. Isto para evitar de organizar o data set de uma certa forma que esteja fora das possibilidades de trabalho do dispositivo. Exemplificando, de nada adiantaria organizar um data set de forma direta ou seqüencial indexada se o suporte do mesmo for um dispositivo de fita magnética.

Vejamos agora quais são os quatro tipos de organização de data sets:

5.2.1 – Organização Seqüencial

É a organização mais simples de todas onde os registros estão numa seqüência física, um após o outro, em vez de estarem numa seqüência lógica. Neste tipo de organização os registros são usualmente lidos ou atualizados na mesma ordem em que aparecem e a localização de registros individuais é relativamente lenta, porque, normalmente, nenhum registro é localizado sem antes passar por todos os registros precedentes no data set.

Esta organização é usada geralmente quando a maioria dos registros é processada cada vez que o data set é utilizado (data set muito ativo).

Para deletar ou adicionar registros o data set inteiro deve ser recriado. No caso de update – atualização de um registro em cima de sua própria localização – se o data set reside em disco, é possível, devido à forma como os registros são armazenados, efetuar este tipo de atualização sem necessidade de recriar o data set.

A seqüencial é uma organização que suporta registros em qualquer formato – F, V, U – podendo estar contida em fita magnética, DASD, fita de papel e dispositivos UNIT RECORD como por exemplo cartão. Quanto aos métodos de acesso, ela utiliza QSAM OU BSAM.

RESUMO – A organização seqüencial:

- I – Tem seus registros colocados um após o outro numa seqüência física.
- II – Faz com que deleções ou adições de registros impliquem numa recriação do data set.
- III – Suporta formatos de registros: F, FB, V, VB, VS, UBS, U.
- IV – Usa métodos de acesso: QSAM ou BSAM.
- V – Pode estar contida em qualquer tipo de dispositivo.

5.2.2 – Organização Seqüencial Indexada

Neste tipo de organização o data set é seqüencial, possuindo índices que permitem acesso direto, e portanto rápido, a registros individuais. Além disso pode ser processado seqüencialmente, desde o início ou então a partir de um certo registro.

Devido a esta característica que permite acesso direto a registros desejados, a organização seqüencial indexada somente é possível em DASD.

Para criar um data set com esta organização, os registros devem ser previamente ordenados em seqüência ascendente de chave. Chave é uma parte do próprio registro que serve como elemento identificador do mesmo, diferenciando-o dos demais registros do data set.

Um data set seqüencial indexado possui três áreas: área primária, área de índices e área de overflow.

5.2.2.1 – Área Primária

Esta área contém as informações de interesse do usuário ou seja, ela contém os registros de criação ou reorganização do data set.

Embora seja possível, graças ao C.S., à área primária ocupar várias áreas não contínuas e em mais de um volume, todos os registros estarão em ordem ascendente de chave.

Os registros podem ser bloqueados ou não bloqueados. Se forem bloqueados o sistema grava precedente ao bloco, a maior chave de registros contido no bloco.

Não é permitido trabalhar com registros indefinidos

5.2.2.2 – Área de Índices

Existem três diferentes níveis de índices criados pelo sistema operacional quando o data set é criado ou reorganizado.

I – ÍNDICE DE TRILHA – é o índice de mais baixo nível e necessariamente presente no data set. Suas entradas apontam para os registros de dados. Sempre que uma trilha da área primária for completada, uma entrada é criada no índice de trilha, e esta entrada aponta para o registro de chave mais alta, portanto o último, daquela trilha. Então, para cada cilindro utilizado na área primária temos um índice de trilha contendo entradas para cada trilha do cilindro a que se refere

Este índice de trilha ocupa sempre a primeira trilha do cilindro.

Cada índice de trilha pode conter um registro inicial especial conhecido como COCR – CYLINDER OVERFLOW CONTROL RECORD – que o sistema utiliza para guardar trilha de endereço do último registro de overflow no cilindro. O restante do índice de trilha é constituído por entradas normais e de overflow alternadas.

A entrada normal contém o endereço da trilha primária e a chave do registro mais alto da trilha.

A entrada de overflow tem o mesmo mecanismo da normal referindo-se a área de overflow. Esta entrada sofre modificações toda vez que registros forem acrescentados ao arquivo. Quando a área de overflow estiver vazia a entrada de overflow terá o mesmo conteúdo da área normal.

A última entrada de cada índice de trilhas é uma entrada "DUMMY" que representa o fim do índice de trilha. Se após o último índice (que é DUMMY) houver espaço, este será ocupado por registros da área primária.

O formato das entradas de índice, que seja entrada normal, de overflow ou "DUMMY", é o mesmo e se trata de um registro de comprimento fixo, não bloqueado e que contém área de contagem, área de chave e área de dados.

A área de chave contém a chave do registro para o qual a entrada aponta, exceto entradas "DUMMY". O comprimento desta área é especificada pelo usuário.

A área de dados é sempre composta de dez bytes e contém o endereço completo da trilha ou registro para o qual o índice aponta, além de outras informações.

II – ÍNDICE DE CILINDRO – este índice, que é de nível mais alto que o de trilha, será gerado automaticamente caso o data set ocupe mais de um cilindro. Para cada índice de trilha criado, o sistema gera uma entrada no índice de cilindro. Cada entrada no índice de cilindro aponta, através de seu campo de dados, para o endereço da trilha que contém o índice de trilha. Além disso a área de chave da entrada contém a chave do registro mais alto do cilindro.

Neste índice também a última entrada é uma entrada “DUMMY” que indica fim do índice.

III – ÍNDICE MESTRE – é o nível mais alto de índice e é opcional, isto é, pode ser requisitado pelo usuário através de comando de controle do J.C.L. É usado quando o data set for muito grande e conseqüentemente o índice de cilindro também, ocasionando uma pesquisa muito demorada. É conveniente criar um índice mestre sempre que o índice de cilindro ultrapassar quatro trilhas. O índice mestre aponta para a chave do registro mais alto de cada trilha no índice de cilindro.

O O.S. permite até três níveis de índice mestre, se assim for especificado no cartão DD do J.C.L. Consideremos o seguinte exemplo:

//MESTRE DD . . . ,DCB=(OPTCD=M,NTM=4, . . .)

OPTCD=M indica que a criação de índice mestre está sendo requisitada.

NTM=n é usado para especificar o número de trilhas de “índice de cilindro” para o qual será criada uma entrada no índice mestre e, ao mesmo tempo indica após quantas trilhas de índice mestre deve ser criado um outro nível de índice mestre.

Se substituirmos n por 4 queremos dizer que, cada 4 trilhas de índice de cilindro que forem criadas, será criada uma entrada no índice mestre. Além disso se o número de trilhas de cilindro for tão grande que acarrete um índice mestre superior a 4 trilhas, será criada uma entrada no índice mestre de segundo nível a cada 4 trilhas criadas de índice mestre de primeiro nível. E assim podemos criar até três níveis de índice mestre.

5.2.2.3 – Área de Overflow

A área de overflow é especificada, num data set seqüencial, para permitir a inclusão de registro sem necessidade de recriar imediatamente o data set.

É claro que após um certo número de inclusões, a área de overflow fica saturada e a localização de registros se torna extremamente complexa e demorada a ponto de justificar uma reorganização do data set. Por isso, dissemos que não é necessário recriar o data set imediatamente, porque a recriação não é devida à inclusão em si, como no caso de uma organização seqüencial, mas sim ao número de inclusões que podem diminuir a performance do sistema.

Na criação de um data set seqüencial indexado a área de overflow não é utilizada, começando a sê-lo à medida que no data set forem sendo incluídos registros novos.

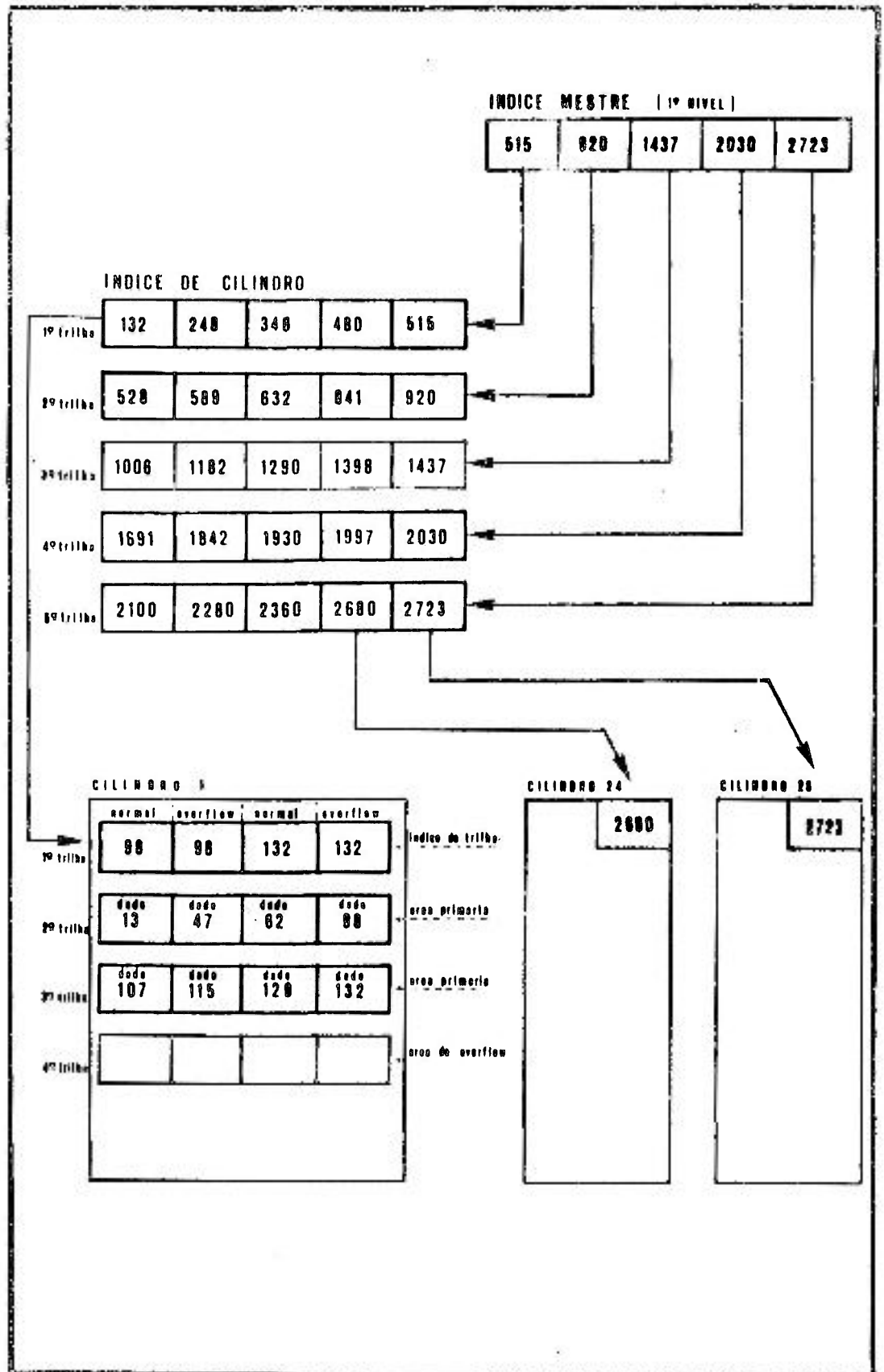


Figura 26 — Organização do Arquivo Seqüencial Indexado

Podemos ter dois tipos de área de overflow: de cilindro e independente.

I – ÁREA DE OVERFLOW DE CILINDRO — é a área de overflow situada no mesmo cilindro onde está situada a área primária. Neste caso um certo número de trilhas inteiras, especificado pelo usuário, é reservado em cada cilindro onde haja área primária.

A vantagem da área de overflow de cilindro está no fato de não haver necessidade de uma busca adicional no mecanismo de acesso para localizar um determinado registro de overflow.

A desvantagem é o espaço eventualmente não usado se as adições forem distribuídas desigualmente por todo o arquivo. Com isto haverá certas áreas de overflow de cilindro saturadas e outras quase vazias.

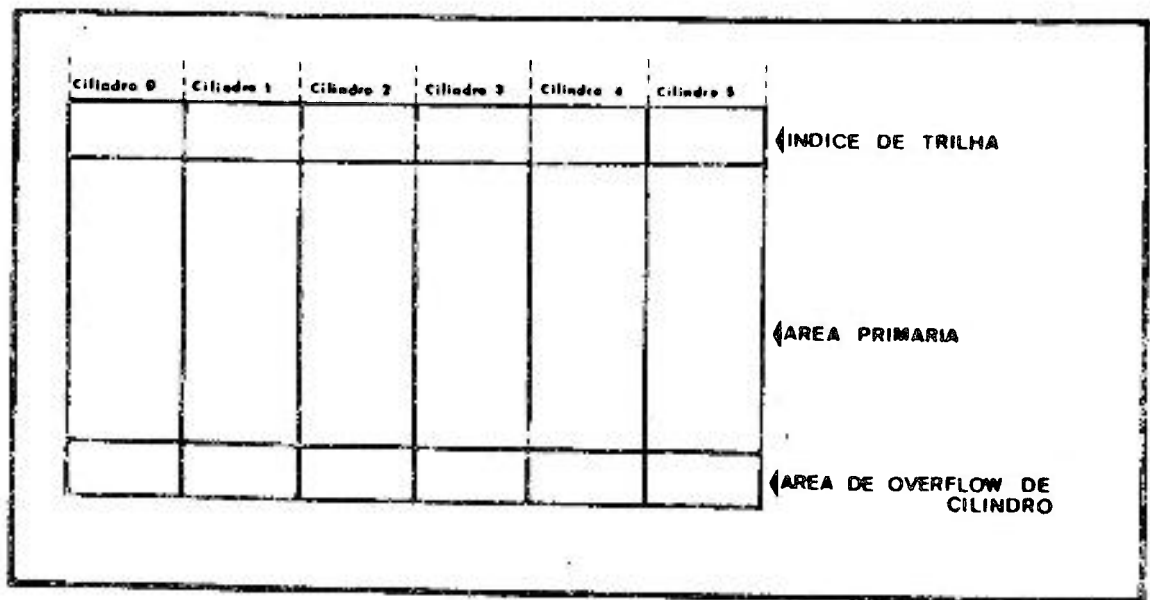


Figura 27 — Área de Overflow de Cilindro

II – ÁREA DE OVERFLOW INDEPENDENTE — é a área de overflow situada num certo número de cilindros reservados unicamente para isto independentemente da localização da área primária.

Esta área também tem seu tamanho e localização especificados pelo usuário devendo, porém, residir no mesmo tipo de DASD da área primária da qual faz parte.

A vantagem da área de overflow independente é que quantidade menor de espaço precisa ser reservada para overflow.

A desvantagem é a necessidade de busca adicional para a localização dos registros de overflow.

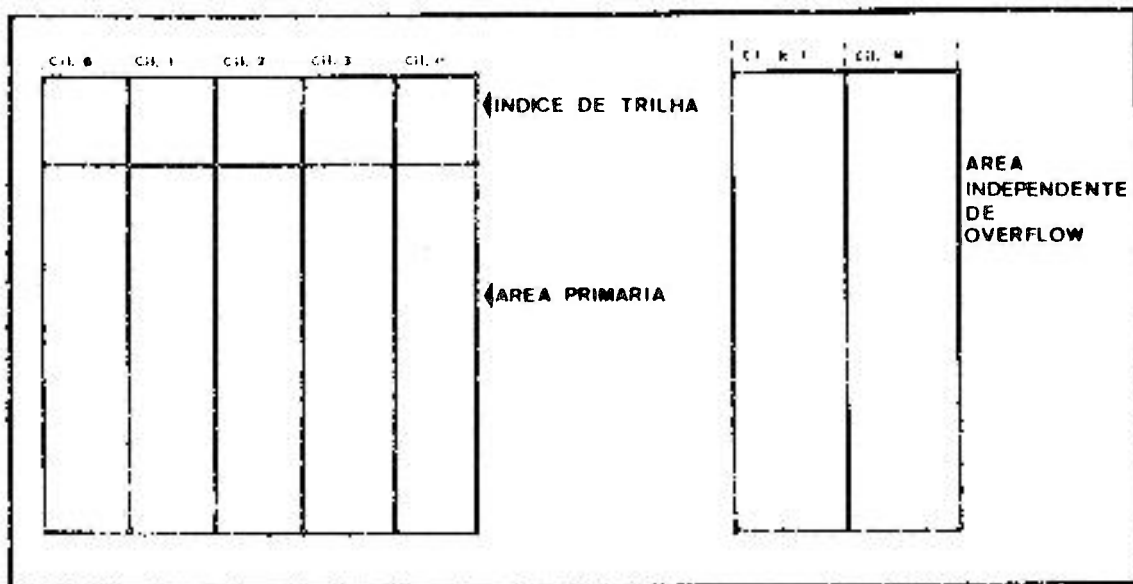


Figura 28 – Área de Overflow Independente

Normalmente, é conveniente que sejam reservados os dois tipos de áreas de overflow: a de cilindro e a independente. Desta forma podemos reservar áreas de overflow de cilindro considerando a média de registros de overflows, sem levar em conta a desigualdade de distribuição de cilindro para cilindro. A área de overflow independente seria utilizada por aqueles registros de overflow cujas áreas de overflow de cilindro já estivessem preenchidas.

A organização seqüencial indexada suporta registros nos formatos F e V e utiliza os métodos de acesso QISAM e BISAM.

O QISAM permite incluir registros no fim da área primária ainda não utilizada, mas não permite gravar registros na área de overflow. Estes registros devem ser classificados em ordem ascendente de chave e devem ter chave maior que o último registro gravado.

O BISAM por sua vez permite adicionar registros, de chave maior que a maior já existente, no final do data set, desde, porém, que seja na última trilha — parcialmente ocupada — ou na área de overflow.

Permite também intercalar registros na posição correta de chave, deslocando os demais registros na trilha. Caso o último registro da trilha não caiba na mesma, ele será transferido para a área de overflow.

Podemos então concluir dizendo que, o método QISAM é usado para criar um data set seqüencial indexado e o BISAM é usado para atualizar.

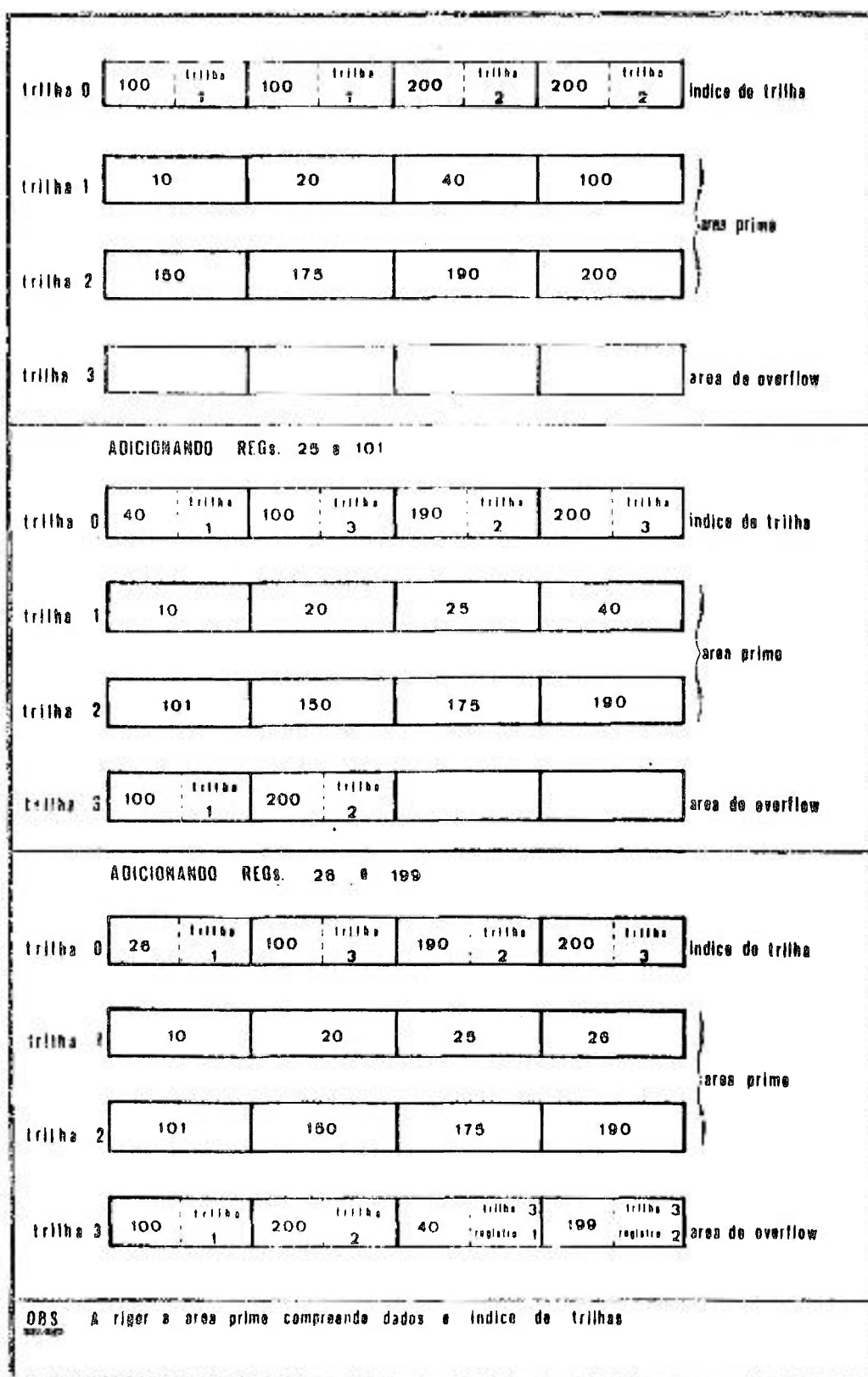


Figura 29 — Adição de Registros em um Data Set Sequencial Indexado

RESUMO: – A organização seqüencial indexada:

- I – Permite acesso direto a registros desejados e permite processar o data set seqüencialmente desde o início ou a partir de um determinado registro.
- II – Permite a inserção de registros sem necessidade de recriação do data set.
- III – Suporta formatos de registros F, FB, V, VB.
- IV – Usa métodos de acesso QISAM – para criação – e BISAM – para atualização randomica.
- V – Pode estar contido apenas em dispositivos de acesso direto

5.2.3 – Organização Direta

Esta organização é possível somente em dispositivos de acesso direto. Nela estão relacionados chave e endereço do registro. Esta relação é estabelecida pelo usuário.

A organização direta é usada para data sets cujas características não permitem o uso de outro tipo de organização ou então para data sets onde o tempo requerido para localizar registros individuais deve ser o menor possível. É uma organização flexível com a única desvantagem de requerer uma programação mais complexa.

Nesta organização existem dois tipos de endereçamento: o direto e o indireto.

5.2.3.1 – Endereçamento Direto

Este tipo de endereçamento permite que cada chave converta para um único endereço. Desta forma a localização de um registro é possível com um seek e uma leitura

Para que seja possível utilizar a chave de um registro como seu endereço, os registros devem ser de comprimento fixo e as chaves devem ser numeradas. O cálculo necessário para isto é o seguinte: divide-se a chave pelo número de registros por trilha; o quociente é igual ao endereço relativo da trilha e o resto mais 1 é igual ao número de registro. Vejamos um exemplo a partir da Figura 30 que representa uma parte de um data set com as características acima descritas.

Utilizando esta figura verifiquemos a localização das chaves: 6 - 19 - 32 e 44.

REGISTRO TRILHA	R0	R1	R2	R3	R4	R5	R6	R7
T0	0	1	2	3	4		6	7
T1	8	9		11	12	13	14	15
T2	16	17	18	19	20	21	22	
T3	24		26	27	28	29	30	31
T4	32	33	34		36	37	38	39
T5	40		42	43	44	45		47

Figura 30 – Organização de um Data Set com Endereçamento Direto

Primeiramente temos de conhecer o número de registros por trilha do nosso data set. No caso do exemplo temos 8 registros por trilha. Portanto vejamos a localização das chaves propostas.

chave 6 6 $\begin{array}{|c|c|} \hline 8 & \\ \hline 6 & 0 \end{array}$

onde 0 é o endereço relativo da trilha e $6 + 1 = 7$ é o número de registro.

concluindo — a chave 6 é o 7º registro da trilha 0.

chave 19 19 $\begin{array}{|c|c|} \hline 8 & \\ \hline 3 & 2 \end{array}$

a chave 19 portanto é o 4º registro da trilha 2

chave 32 32 $\begin{array}{|c|c|} \hline 8 & \\ \hline 0 & 4 \end{array}$

a chave 32 é o primeiro registro da trilha 4

chave 44 44 $\begin{array}{|c|c|} \hline 8 & \\ \hline 4 & 5 \end{array}$

a chave 44 é o 5º registro da trilha 5

O endereçamento direto permite um processamento randômico com um mínimo de gasto de tempo de dispositivo e é ideal para processamento seqüencial pelo fato de os registros serem escritos em seqüência de chave

Por outro lado este tipo de endereçamento é aconselhável somente quando a porcentagem de endereços não usados for baixa, caso contrário teremos uma quantidade muito grande de áreas não utilizadas.

5.2.3.2 — Endereçamento Indireto

É utilizado principalmente nos casos de data sets que apresentam registros cujas chaves incluem uma porcentagem de endereços tão alta que torna impraticável o uso do tipo direto de endereçamento. Por exemplo: se tivermos 3000 registros compreendidos entre as chaves 0001 e 9999 — a mais baixa e a mais alta respectivamente — não podemos usar o endereçamento direto porque assim procedendo estaremos perdendo 6999 endereços, já que reservamos 9999 para utilizar somente os 3000.

Um outro fato que leva à utilização do endereçamento indireto é a utilização de registros cujas chaves, por motivos vários, não podem ser numéricas.

No endereçamento indireto não existe relação entre chave e endereço, ou melhor, a relação existe mas não é imediata, depende do cálculo de um algoritmo pré estabelecido. É evidente que para localizar um registro do data set é preciso utilizar o mesmo algoritmo utilizado para a gravação do referido registro.

A esta técnica de utilização de algoritmos de cálculo para determinação de endereço chamamos RANDOMIZAÇÃO.

A randomização permite diminuir os intervalos de variação das chaves mas apresenta dois inconvenientes: o primeiro diz respeito ao número de endereços não utilizados. Isto se explica pelo fato de, ao calcular os algoritmos, nenhum dos resultados pode convergir em determinados endereços e consequentemente estes permanecem inutilizados. O segundo inconveniente diz respeito à presença de

sinônimos, isto é: chaves que randomizam no mesmo endereço ou, em outras palavras, dois ou mais registros cujas chaves, após o cálculo do algoritmo, convergem para o mesmo endereço. Neste caso, o próprio sistema se encarrega de resolver o problema automaticamente, sem transtornos para o usuário. Porém, é preciso considerar que sempre há o inconveniente do tempo gasto para resolver o endereço final do sinônimo ou sinônimos.

A título de ilustração vejamos um exemplo de um tipo de randomização possível.

Localizar o registro de chave 25.463.514 num data set com 10.000 registros distribuídos em 12 registros por trilha.

a) – o primeiro passo é descobrir o número primo imediatamente inferior a 10.000 que é 9973.

b) – dividir a chave do registro pelo número primo.

$$\begin{array}{r} 25.463.514 \quad | \quad 9.973 \\ - \quad - \quad - \quad - \quad 2.553 \\ \hline 2445 \end{array}$$

Assim procedendo obtemos um resto que nos fornece a localização relativa do registro no data set. Neste caso o registro de chave 25.463.514 é o 2446º registro do data set.

c) – dividir o resto da divisão do item b pelo número de registros por trilha.

$$\begin{array}{r} 2445 \quad | \quad 12 \\ - \quad - \quad 203 \\ \hline 9 \end{array}$$

Onde 203 é a trilha relativa onde se encontra o registro e 9 é a posição relativa do registro dentro da trilha.

d) – portanto, usando este tipo de randomização o registro de chave 25.463.513 é o 2446º registro do data set e, mais precisamente, é o 10º registro da 204ª trilha.

OBS. – não esquecer que o primeiro registro é o R0 e a primeira trilha é a T0 daí ser o 10º registro da 204ª trilha e não o 9º registro da 203ª trilha.

Evidentemente existem vários algoritmos de cálculo, ou técnicas de randomização, que possibilitam o endereçamento indireto em data sets organizados diretamente. Alguns mais simples, outros mais complexos, mas todos válidos. O melhor, o mais eficiente, será, sem dúvida, aquele que, por força da própria regra de formação, acarretar o menor número possível de endereços não utilizados bem como o menor número possível de sinônimos.

RESUMO: – A organização direta

I – Permite acessar rapidamente registros particulares não sendo eficiente para processamento seqüencial.

II – Permite a inserção, a eliminação e a atualização de registros sem necessidade de recriação do data set.

III – Suporta formatos: F, FB, V, VB, U

IV – Usa métodos de acesso: BSAM — para criação — e BDAM — para pesquisa e atualização randômica.

V – Pode estar contida apenas em dispositivos de acesso direto.

5.2.4 – Organização Particionada

Esta organização reúne características de seqüencial e seqüencial indexada, sendo possível sua utilização somente em dispositivos de acesso direto

Um data set particionado (P.D.S.) é um conjunto de data sets independentes, organizados seqüencialmente, chamados membros e que são reconhecidos individualmente, para serem processados. Os registros dentro dos membros são organizados seqüencialmente e são recuperados ou guardados sucessivamente de acordo com a seqüência física.

É um tipo de organização utilizado principalmente para armazenar dados seqüenciais tais como: programas, subrotinas e tabelas.

Os nomes dos membros, ordenados em seqüência ascendente, bem como o endereço inicial de cada um constitui uma tabela chamada DIRETÓRIO e situada nas posições iniciais do P.D.S.

O diretório é formado por blocos de 256 bytes cada, contendo a entrada para os membros.

Cada entrada é formada por um "NOME DE MEMBRO", ocupando os primeiros 8 bytes, seguido por um "APONTADOR", com 3 bytes, que aponta para o primeiro registro do MEMBRO, especificando a trilha relativa dentro do data set particionado. Pelo fato de especificar a trilha relativa e não o endereço absoluto da mesma, o P.D.S. pode ser movido sem necessidade de relocar o endereço dos membros. Isto confere uma grande versatilidade a este tipo de organização de data set.

Um data set particionado não pode ter seus membros atualizados "no lugar" (in place). A atualização é executada e o membro atualizado é regravado inteiramente após o último membro existente no P.D.S., desde que haja espaço suficiente. Desta forma, resta um espaço liberado mas não utilizável na área ocupada anteriormente pelo membro utilizado e agora regravado no final do P.D.S. Isto faz com que periodicamente, um P.D.S. sofra reorganização para eliminar estes espaços e transformá-los em área final disponível.

Toda vez que houver a inclusão ou a exclusão de um membro ou quando for o caso de uma reorganização do data set, o diretório será atualizado e reclassificado.

RESUMO: A Organização particionada

I – Permite acessar um membro através no nome e endereço constantes no diretório.

II – Exige uma recriação periódica para eliminar espaços deixados por membros deletados ou atualizados.

III – Suporta formatos: F, FB, V, VB.

IV – Usa métodos de acesso: BSAM — para criação — e BPAM — para atualização.

V – Pode estar contida apenas em dispositivos de acesso direto.

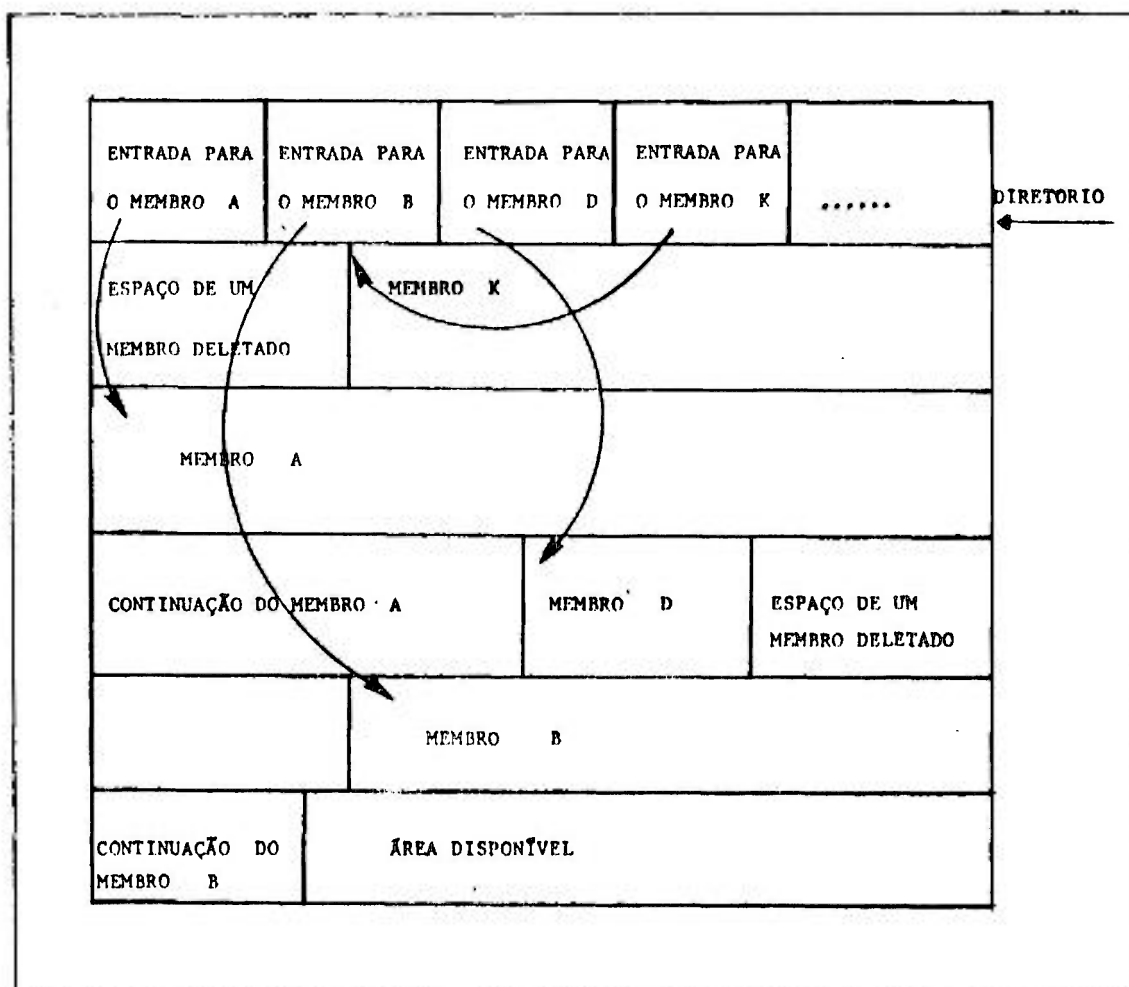


Figura 31 – Esquema de um Data Set Particionado

5.3 – Técnicas de Acesso

Para a atualização de qualquer um dos tipos de organização de data sets mencionados, o O.S. fornece ao usuário macro-instruções de entrada e saída que podem ser divididas em duas categorias, ou técnicas, que se diferenciam entre si pelos recursos que oferecem. As técnicas em questão são: técnica QUEUED e técnica BASIC.

5.3.1 – Técnica Queued

Esta técnica utiliza as macros GET e PUT para efetuar a transmissão de dados entre a memória principal e as unidades de entrada e saída e vice-versa.

- Aplica-se apenas para processamento seqüencial.
- Possibilita o acesso ao registro lógico através de blocagem e deblocagem automáticos.
- Sincroniza as operações de entrada e saída com o processamento.
- Obtém e controla Buffers automaticamente.
- Deteta e recupera erros automaticamente.
- Manipula condições de fim de arquivo (EOF) e fim de volume (EOV).

Estas características não impedem que o programador tenha de se preocupar com problemas tais como:

- Escolha de técnica de bufferização
- Escolha do modo de transmissão
- Escrever a DCB
- Usar, de modo apropriado, os macros GET e PUT.

5.3.2 – Técnica Basic

Esta técnica faz uso das macro READ e WRITE para transmitir dados entre a memória principal e as unidades de entrada e saída e vice-versa. Cumpre salientar porém que as macros READ e WRITE apenas iniciam as operações de entrada e saída, cabendo ao programa problema a sincronização das mesmas.

Possui as seguintes características:

- Trabalha com qualquer tipo de organização.
- Fornece acesso apenas ao registro físico, não efetuando blocagem nem deblocagem automática.
- Não providencia alocação nem controle de buffers de forma automática.
- Não executa a detecção nem a recuperação de erros.
- Não testa condições de fim de arquivo (EOF) nem de volume (EOV).

Através destas características é fácil perceber que o programador fica encarregado de todas as operações desejadas.

Esta técnica, por suas características, é usada nos casos em que o sistema não tem condições de prever a ordem em que os registros serão processados ou então, nos casos em que alguma das funções automáticas da Técnica QUEUED não for desejada.

***BPAM – BASIC PARTITIONED ACCESS METHOD**

Permite a manipulação do diretório de um data set particionado, acessando e alterando informações do mesmo.

***BTAM – BASIC TELECOMMUNICATION ACCESS METHOD**

Este método é utilizado para manipular terminais usando as facilidades de acesso BASIC.

Além dos métodos de acesso acima citados, que são todos métodos de acesso próprios do sistema, é possível ao usuário escrever seu próprio método de acesso, em conformidade com certas características peculiares de processamento. É possível, por exemplo, que o usuário deseje construir um programa de canal ou inventar um algoritmo qualquer de acesso a data set, etc. Isto é possível através do uso da macro EXCP – Execute Channel Program.

O uso satisfatório desta macro exige conhecimentos detalhados de:

- Controle do dispositivo
- Funções do sistema
- Estrutura dos blocos de controle

5.5 – Labels

Os labels são identificadores (rótulos, etiquetas) de volumes ou de data sets contidos em volumes.

Os labels de volume são utilizados para identificar o volume sendo obrigatórios em unidades de acesso direto e opcionais em fita magnética.

O label de volume oferece segurança na medida em que evita que volumes indevidos sejam utilizados por engano.

É aconselhável que cada volume tenha uma identificação única dentro de uma instalação. Este conselho de torna uma obrigatoriedade quando os volumes estão montados em linha (on-line).

Os labels de data sets por sua vez identificam o data set dentro de um volume específico. Quando os data sets estão em fita magnética, sua disposição é seqüencial e os labels são utilizados para distinguir cada data set bem como para conferir-lhe suas próprias características.

Os labels podem ser divididos em dois grupos: labels para fitas magnéticas – TAPE LABELS – e labels para dispositivos de acesso direto – DASD LABELS.

Data sets oriundos de equipamentos do tipo UNIT RECORD não tem labels.

5.5.1 – Tape Labels

Com o O.S. podem ser processados volumes com os seguintes tipos de labels:

- I – SL – Standard Label

II – SUL – Standard User Label

III – NSL – Non Standard Label

IV – NL – No Label

I – STANDARD LABEL – São registros de 80 bytes constituídos de um label de volume e um grupo de labels do data set.

O label do volume é o primeiro registro na fita e identifica o volume e o proprietário. Os labels do data set antecedem e sucedem cada data set no volume, além de identificá-lo e descrevê-lo.

Os labels do data set são:

- Header Labels – aqueles que antecedem o data set.
- STANDARD USER LABEL – opcional.
- Trailer Label – aqueles que sucedem o data set.

II – NON STANDARD LABEL – Pode ser de qualquer tamanho e é processado por rotinas escritas pelo usuário.

No cartão DD e JCL o usuário especificará o parâmetro LABEL = (, NSL).

III – NO LABEL – Um volume de fita magnética sem label pode conter um ou mais data sets (separados por Tape Marks).

No cartão DD de JCL o usuário especificará o parâmetro LABEL = (, NL). Para ler um certo data set devemos contar o número de tape marks que o procedem. Ex: LABEL = (2, NL) fará com que o sistema leia o segundo data set do volume.

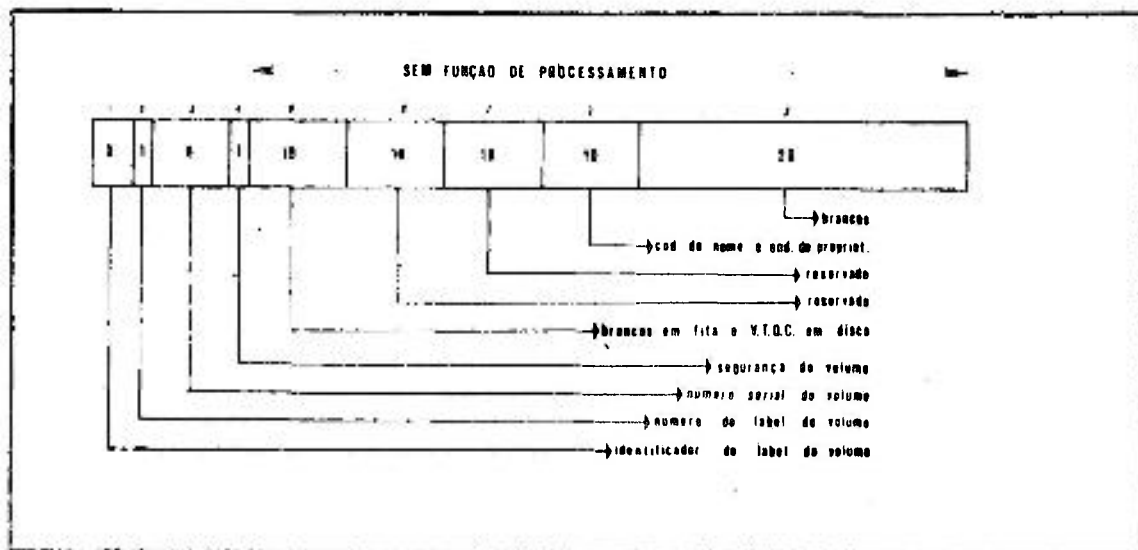


Figura 33 – Label de Volume para Fita e Disco

5.4 – Métodos de Acesso

Após estarmos de posse de alguns conceitos fundamentais sobre organização de data sets e sobre técnicas de acesso, podemos conceituar o que sejam métodos de acesso.

Chamamos de método de acesso à combinação de uma técnica de acesso com uma certa organização de data set.

ORGANIZAÇÃO DO DATA SET	TÉCNICAS DE ACESSO	
	QUEUED	BASIC
SEQUENCIAL	QSAM	BSAM
SEQUENCIAL INDEXADA	QISAM	BISAM
DIRETA		BDAM
PARTICIONADA		BPAM
TELECOMUNICAÇÕES	QTAM	
	TCAM	BTAM

Figura 32 – Métodos de Acesso do O.S.

Os métodos de acesso disponíveis no O.S. são os seguintes:

***QSAM – QUEUED SEQUENTIAL ACCESS METHODS**

É o método que permite processar seqüencialmente um data set organizado de forma seqüencial.

***QISAM – QUEUED INDEXED SEQUENTIAL ACCESS METHOD**

Permite processar seqüencialmente um data set organizado de forma seqüencial indexada.

***QTAM – QUEUED TELECOMMUNICATIONS ACCESS METHOD**

***TCAM – TELECOMMUNICATION ACCESS METHOD**

Estes dois métodos são utilizados quando temos entradas e saídas procedentes de terminais remotos, fornecendo diversas macros e as facilidades da técnica de acesso QUEUED.

***BSAM – BASIC SEQUENTIAL ACCESS METHOD**

Permite processar seqüencialmente um data set organizado de forma seqüencial.

***BISAM – BASIC INDEXED SEQUENTIAL ACCESS METHOD**

Permite processar randomicamente um data set organizado de forma seqüencial indexada.

***BDAM – BASIC DIRECT ACCESS METHOD**

É o único método de acesso com o qual podemos processar data sets com organização direta.

5.5.2 – Dasd Label

Cada volume de acesso direto é identificado por um label de volume com 80 caracteres e que se localiza no terceiro registro da trilha zero, cilindro zero. Este label é análogo ao Tape Label de volume com a diferença que o campo 5, que na fita não é utilizado, aqui é utilizado para conter o apontador da VTOC.

A VTOC é composta por registros de DSCB – Data Set Control Block – de todos os arquivos residentes no volume.

Conforme vimos acima o label de volume tem localização fixa, mas a VTOC não, por isso existe no label de volume um apontador para a VTOC. Normalmente a VTOC é colocada em área adjacente ao label de volume para minimizar o tempo de SEEK.

Quando o data set contém labels do usuário, estes ocupam área imediatamente anterior à do data set.

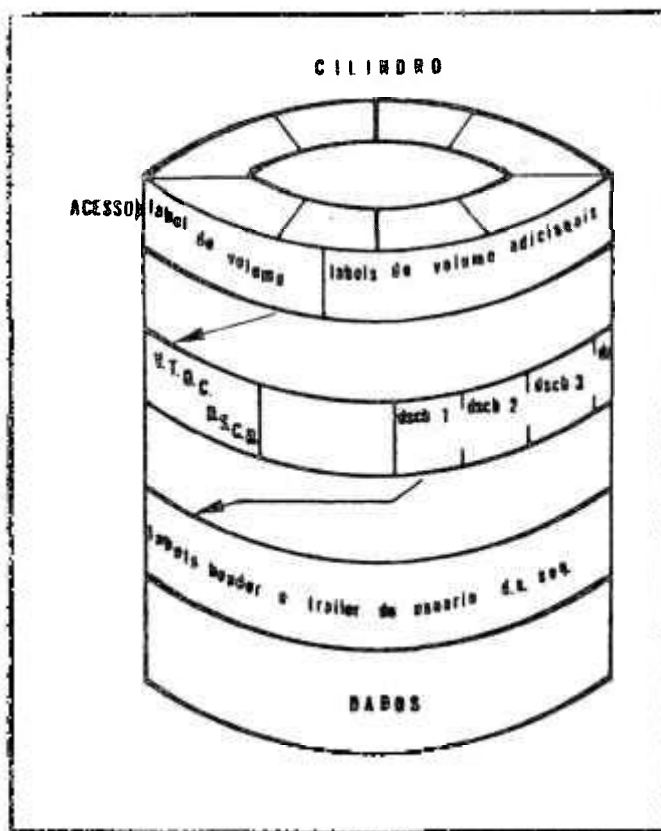


Figura 34 — Label de Dasd

5.6 — Data Sets do Sistema

Existem 23 data sets do sistema dos quais 8 são obrigatórios e 15 opcionais.

Independentemente do fato de serem obrigatórios ou opcionais, todos os data sets do sistema, com exceção do SYSCTLG, tem nomes qualificados compostos de dois nomes simples ligados por ponto, dos quais o primeiro é sempre SYS1.

5.6.1 – Data Sets Obrigatórios

*SYSCTLG – é um data set seqüencial que contém o catálogo do sistema.

*SYS1.NUCLEUS – é um data set particionado que contém as rotinas residentes do programa de controle (NÚCLEO).

*SYS1.SVCLIB – data set particionado contendo rotinas não residentes que são carregadas à medida da necessidade.

*SYS1.LOGREC – contém estatísticas quanto ao funcionamento do equipamento. É organizado seqüencialmente.

*SYS1.LINKLIB – contém os módulos executáveis (LOAD MODULES) ou seja, os programas de processamento da instalação. Sua organização é particionada.

*SYS1.SYSJOBQE – data set seqüencial contendo os cartões de controle separados por classes e ordenados por prioridades.

*SYS1.PROCLIB – é um data set particionado que contém os procedimentos catalogados.

*SYS1.PARMLIB – contém os parâmetros utilizados pelo sistema como por exemplo PRESRES – lista de volumes PRIVATE. Sua organização é particionada.

Observação: – Estes data sets obrigatórios do sistema não necessitam residir obrigatoriamente no mesmo volume de DASD. Porém, o volume de DASD que contém o Sistema Operacional – SYSRES – deve conter obrigatoriamente:

- IPL
- SYSCTLG
- SYS1.NUCLEUS
- SYS1.SVCLIB
- SYS1.LOGREC

É aconselhável que o data set SYS1.SYSJOBQE seja criado no dispositivo de acesso direto mais rápido da instalação e, de preferência, separado do SYSRES para evitar interferências.

5.6.2 – Data Sets Opcionais

*SYS1.MACLIB – macros para ASSEMBLER

*SYS1.FORTLIB – rotinas para FORTRAN

*SYS1.COBLIB – rotinas para COBOL

*SYS1.PLILIB – rotinas para PL/I

*SYS1.SORTLIB – rotinas para SORT/MERGE

*SYS1,ALGLIB — rotinas para ALGOL

*SYS1.TELCMLIB – rotinas para Teleprocessamento

*SYS1.ASRLIB – certos módulos do núcleo para rotinas de manuseio do “Machine – check”

*SYS1.SYSVLOGX – } dados do "system log"

*SYS1.SYSVLOGY --

*SYS1.ROLLOUT – função ROLLOUT/ROLLIN

*SYS1.ACCT – informações de contabilização

*SYS1.MANX —
 } SYSTEM MANAGEMENT FACILITIES

*SYS1.MANY —

*SYS1.DUMP – DUMP da memória (Abend)

6 – CONTROLE DO SISTEMA

6.1 – Introdução

O computador é uma máquina preparada para realizar certas tarefas, mais ou menos complexas, de forma rápida e precisa. Porém, como toda máquina, é desprovido de inteligência e por esta razão necessita ser instruído e controlado.

A instrução é definida pelos programas de aplicação que determinam o que deve ser feito. Cada programa portanto instruirá o computador de uma certa maneira visando alcançar os objetivos para os quais ele, programa, foi criado.

O controle é exercido por comandos externos que determinam como e com que meios fazer aquilo que o programa de aplicação requer.

Nota-se portanto um vínculo estreito entre o programa de aplicação e os comandos de controle. Considerando que o programa de aplicação prevalece sobre os comandos de controle na medida em que, estes, nada mais são do que auxiliares do computador para atingir os objetivos definidos por aquele: parece evidente que o controle seja efetuado em função das instruções e não ao contrário. Então, para resumir este pensamento, a situação se apresenta da seguinte maneira: existe um problema e existe um computador com o qual podemos resolver o problema. Primeiramente é preciso traduzir o problema de modo inteligível para a máquina. Isto é feito através de uma seqüência de instruções numa linguagem qualquer pertencente ao conjunto das linguagens que a máquina está apta a interpretar. Em segundo lugar é preciso estabelecer uma comunicação com a máquina para orientá-la quanto às necessidades do problema e dos recursos, bem como para que ela nos informe sobre possíveis erros ou omissões ou peça instruções complementares. Isto é feito, pelo menos no nosso caso, através da utilização de uma linguagem de controle de trabalho denominada JOB CONTROL LANGUAGE — J.C.L. — , composta de uma série limitada de comandos dentre os quais utilizamos aqueles de que necessitamos num trabalho determinado. Estes comandos são enviados ao computador via cartões perfurados que recebem o nome de JOB CONTROL CARDS — J.C.C. — . Por este motivo é que geralmente falamos indiferentemente em comandos de controle ou cartões de controle apesar de, a rigor, o cartão ser o veículo sobre o qual circula o comando e tudo mais a este associado.

DATA SET DO SISTEMA	OBRIGATÓRIO	ORGANIZAÇÃO	RESIDEM NO SYSRES	ALOCÇÃO SE- CUNDÁRIA PER MITIDA	CATALOGADO
SYSCTIG	SIM	SEQ.	SIM	SIM	NÃO
SYS1.NUCLEUS	SIM	PDS	SIM	NÃO	OPCIONALMENTE
SYS1.SVCLIB	SIM	PDS	SIM	SIM	PREFERIVELMENTE
SYS1.LOGREC	SIM	SEQ.	SIM	NÃO	NÃO
SYS1.LINKLIB	SIM	PDS	OPCIONALMENTE	SIM	SIM
SYS1.PARMLIB	SIM	PDS	OPCIONALMENTE	NÃO	PREFERIVELMENTE
SYS1.PROCLIB	SIM	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.SYSJOBQE	SIM	SEQ.	OPCIONALMENTE	NÃO	PREFERIVELMENTE
SYS1.MACLIB	NÃO	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.SORTLIB	NÃO	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.ALGLIB	NÃO	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.COBLIB	NÃO	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.FORTLIB	NÃO	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.PLILIB	NÃO	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.TELCLIB	NÃO	PDS	OPCIONALMENTE	SIM	PREFERIVELMENTE
SYS1.SYSVLOGX	NÃO	SEQ.	OPCIONALMENTE	NÃO	SIM
SYS1.SYSVLOGY	NÃO	SEQ.	OPCIONALMENTE	NÃO	SIM
SYS1.ROLLOUT	NÃO	SEQ.	OPCIONALMENTE	NÃO	SIM
SYS1.ASRLIB	NÃO	SEQ.	SIM	NÃO	NÃO
SYS1.ACCT	NÃO	SEQ.	OPCIONALMENTE	NÃO	NÃO
SYS1.MANX	NÃO	SEQ.	OPCIONALMENTE	NÃO	PREFERIVELMENTE
SYS1.MANY	NÃO	SEQ.	OPCIONALMENTE	NÃO	PREFERIVELMENTE
SYS1.DUMP	NÃO	SEQ.	OPCIONALMENTE	NÃO	SIM

Figura 35 — Data Sets do Sistema

Em resumo podemos dizer que o controle do sistema é feito através de comandos de JOB CONTROL LANGUAGE os quais, pelo papel que exercem, precisam ser muito bem conhecidos por quem deles for utilizar-se, para poder obter do sistema a melhor performance possível.

Este conhecimento detalhado necessário, não será fornecido nas presentes notas porque o J.C.L. deve ser tratado a parte, tal sua importância e extensão, não podendo figurar aqui senão sob forma introdutória, muito mais visando mostrar o potencial do O.S., do que propriamente com a preocupação de ensinar sua utilização. E é assim que será tratado.

6.2 – Job Control Language

O J.C.L. é uma linguagem que permite estabelecer uma comunicação entre o usuário e o sistema operacional.

Mais precisamente podemos dizer que através do J.C.L. o usuário se comunica com o JOB MANAGEMENT e especialmente com o JOB SCHEDULER.

6.2.1 – Cartões de J.C.L.

Existem nove comandos ou, usualmente falando apesar de indevido, nove cartões de J.C.L.

– CARTÃO JOB

Identifica o JOB, marca seu início e o fim do precedente.

– CARTÃO EXEC

Marca o início de um step e o fim do step anterior. Identifica o programa ou o procedimento catalogado a ser chamado.

– CARTÃO DD

Identifica um data set e descreve seus atributos.

– CARTÃO PROC

Marca o início de um procedimento "in-stream" e opcionalmente de um procedimento catalogado.

No primeiro caso pode ser usado para atribuir valores e parâmetros simbólicos. No segundo caso é usado para esse fim.

– CARTÃO PEND

Marca o fim de um procedimento in-stream.

– CARTÃO COMANDO

Usado para entrar com comandos através do input stream.

– CARTÃO DELIMITADOR

Usado no input stream para indicar fim dos dados.

– CARTÃO COMENTÁRIO

Usado para entrar com informações auxiliares.

– CARTÃO NULO

Pode ser usado para indicar o fim de um Job Stream.

6.2.2 – Campos no Cartão de J.C.L.

Existem sete campos compondo um cartão de J.C.L.

– CAMPO INICIAL

Este campo ocupa as colunas 1 e 2 e contém sempre //, com exceção do cartão delimitador que contém /*.

– CAMPO DE NOME

Identifica o cartão permitindo que seja referenciado. Este campo deve iniciar na coluna 3 e pode comportar tanto nomes simples como qualificados, obedecendo às características de formação de cada um deles.

– CAMPO DE OPERAÇÃO

Especifica o tipo de cartão de controle ou, em se tratando de um cartão de comando, o comando. Este campo deve ser precedido e seguido de pelo menos um branco.

– CAMPO DE OPERANDO

Contém os parâmetros separados por vírgulas. Deve ser precedido e seguido de pelo menos um branco.

– CAMPO DE COMENTÁRIOS

Pode conter qualquer informação que seja útil. Deve ser precedido de pelo menos, um branco.

– CAMPO DE CONTINUAÇÃO

Este campo, ocupando a coluna 72, pode conter qualquer carácter, inclusive brancos, quando o cartão que o contém tiver continuação. Se não, este campo deve conter apenas branco. Não necessita estar entre brancos.

– CAMPO DE IDENTIFICAÇÃO

Ocupando as colunas 73 a 80, pode ser usado para conter códigos que identifiquem o cartão. Não precisa ser precedido nem seguido de brancos.

Embora sejam estes os campos existentes, nem todos os cartões os contém na sua totalidade.

6.2.3 – Regras para Continuar um Cartão de J.C.L.

- A interrupção somente pode se dar após a codificação de um parâmetro ou subparâmetro completo, isto é, após uma vírgula.
- A vírgula não deve ultrapassar a coluna 71.
- O cartão seguinte deve ter // codificados na coluna 1 e 2.
- A continuação deve ser indicada entre as colunas 4 e 16.

Observações:

1 – Os comandos:

- COMANDO (//)
- DELIMITADOR (//*)
- COMENTÁRIO (//*)
- NULO (//)

não têm continuação. No caso dos cartões: COMANDO e COMENTÁRIO, embora não possam ter continuação, podemos codificar tantos cartões quantos sejam necessários.

2 – Não é necessário perfurar um carácter diferente de branco, na coluna 72, para indicar que há continuação no próximo cartão.

6.2.4 – Tipos de Parâmetros no J.C.L.

Existem 4 tipos de parâmetros em J.C.L.

- OPCIONAIS – usados somente quando necessário.
- OBRIGATÓRIOS – sempre usados.
- POSICIONAIS – cujo significado depende da posição relativa dentro do conjunto de parâmetros.
- PALAVRAS CHAVES – cujo significado é dado pela própria palavra, independentemente da sua posição relativa.

6.2.5 – Uso do Parênteses e Apóstrofe em J.C.L.

- I – Os parênteses são comumente utilizados nos casos de subparâmetros dentro de parâmetros.

- II – O apóstrofe é usado principalmente para casos em que seja necessário o uso de caracteres especiais ou nomes com mais de 8 caracteres.

Observação: Há casos em que os apóstrofes podem ser substituídos por parênteses.

6.2.6 – Procedimentos Catalogados

Os procedimentos catalogados são conjuntos de comandos de controle gravados num data set particionado denominado biblioteca de procedimentos (SYS1.PROCLIB).

Um procedimento catalogado pode ser chamado e intercalado num job, por meio do cartão EXEC no qual o parâmetro PROC especifica o nome do membro do P.D.S. SYS1.PROCLIB.

Para atender a certas necessidades momentâneas, os parâmetros da procedure podem ser alterados (OVERRIDE) pelo usuário e o sistema executará as alterações indicadas ao intercalar o procedimento no JOB. Na biblioteca o procedimento permanece inalterado.

Nas posições iniciais da listagem dos cartões de controle emitida pelo computador temos:

// indicando os J.C.C. do input stream

XX indicando os J.C.C. da SYS1.PROCLIB sem OVERRIDE

X/ indicando os J.C.C. da SYS1.PROCLIB com OVERRIDE

Um exemplo de procedimento catalogado são as etapas de compilação, linkedição e execução dos programas de aplicação. Estas etapas, pelo fato de serem idênticas para todos os programas, e para evitar que as mesmas tivessem que ser anexadas ao Input Stream toda vez que um programa ia ser testado, foram convertidas em procedimentos catalogados. Em decorrência disto temos procedimentos catalogados tais como:

COBUC – para compilação em COBOL

COBUCL – para compilação e linkedição em COBOL

COBUCLG – para compilação, linkedição e execução em COBOL.

etc.

6.2.7 – Bibliotecas Particulares

Os módulos de carga são guardados normalmente na biblioteca do sistema SYS1.LINKLIB. Isto não impede que o usuário crie e use bibliotecas particulares de acordo com suas necessidades e conveniências. O O.S. permite ao usuário criar bibliotecas particulares distintas contendo os módulos de carga de programas.

Normalmente o sistema pesquisa o programa a ser carregado na biblioteca SYS1.LINKLIB. Quando forem usadas bibliotecas particulares é preciso informar ao sistema que o programa a ser carregado não está na biblioteca do sistema mas sim numa biblioteca particular, além disso temos de especificar o nome da biblioteca particular. Desta forma o sistema irá procurar o programa a carregar, primeiro na biblioteca particular e, somente se não o encontrar lá, irá procurá-lo no SYS1.LINKLIB.

A maneira de informar o sistema da existência de bibliotecas particulares é especificar o data set através de cartões DD especiais:

– **JOBLIB** – situado entre o cartão JOB e o primeiro EXEC deste JOB. A biblioteca especificada com este cartão DD é válida para todos os steps do JOB exceto aqueles que tiverem o cartão STEPLIB.

– **STEPLIB** – situado imediatamente após o cartão EXEC. A biblioteca especificada com este cartão DD é válida apenas para step ao qual pertence o STEPLIB.

Exemplo:

//BIBLPART	JOB	
//JOBLIB	DD	DSN=MINHA.BIBLIOT
//PASSO1	EXEC	PGM=PAGTO
	DD	
	DD	
//PASSO2	EXEC	PGM=DESCTO
	DD	
	DD	
//PASSO3	EXEC	PGM=SALDO
//STEPLIB	DD	DSN=BIBLIOT.DE.SALDO
	DD	
	DD	
//PASSO4	EXEC	PGM=FINAL
	DD	
	DD	

6.2.8 – Data Set Concatenado

Possibilita o processamento de data sets como se se tratasse de um único data set de entrada.

Os data sets não necessitam ser semelhantes mas é preciso que sua organização seja seqüencial ou particionada.

Para concatenar data sets basta usar um cartão DD normal para o primeiro data set (aquele ao qual os demais serão concatenados) e cartões DD com o campo de nome do DD – DDname – em branco.

Exemplo:

//ENTRADA	DD	DSN=ABC
//	DD	DSN=DEF
//	DD	DSN=GHI
//	DD	

7 – NOÇÕES DE TELEPROCESSAMENTO

Teleprocessamento é um processamento de dados a distância com a utilização de um terminal mais próximo da origem dos dados do que dos meios de cálculo.

Um sistema de Teleprocessamento não é muito diferente de um sistema de processamento comum, é mais uma extensão deste utilizando as técnicas atuais de comunicação. Daí conceituarmos também T.P. como sendo um sistema de processamento de dados mais comunicação.

A grande novidade do T.P. são os terminais que nada mais são do que unidades de entrada/saída colocados a distância da U.C.P. do sistema central.

7.1 – Elementos Básicos de um Sistema de T.P.

Os elementos básicos de um sistema T.P. são:

TERMINAL

MODEM – ligação entre terminal ou sistema e linha

LINHA

MODEM – ligação entre linha e terminal ou sistema

UCT – unidade de controle de transmissão

UCP – do sistema

DASD e componentes de entrada e saída do sistema.

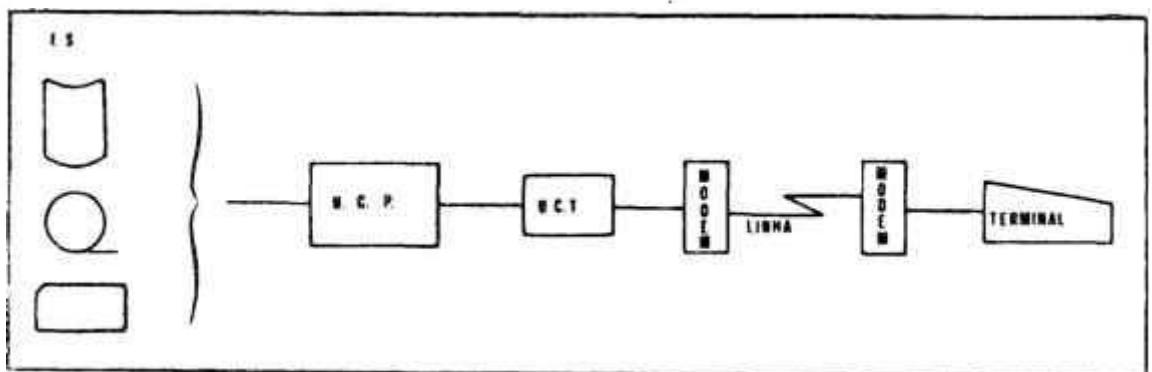


Figura 36 – Componentes de um Sistema de T.P.

***TERMINAL** — é o responsável pela entrada e/ou saída remota de dados num sistema de T.P.

Como exemplo de modalidades de entrada e saída temos:

- cartão perfurado
- fita perfurada
- fita magnética
- teclado
- vídeo
- impressora
- telefone
- U.C.P.
- etc.

***MODEM** — é o elemento de ligação entre o terminal ou o sistema e a linha. Tem por função modular ou demodular o sinal, geralmente digital, que parte ou chega — do terminal — ou da UCP.

***LINHA** — cabos ou canais hertzianos sobre os quais circulam as informações — dados — num sistema de T.P.

***U C T** — são dispositivos que servem para fazer a interface entre a UCP e as linhas de comunicação. Em outras palavras, compatibilizam os sistemas de processamento de dados convencionais com os sistemas de teleprocessamento.

***U C P** — também pode funcionar como terminal sendo o elemento básico mais importante em Teleprocessamento.

7.2 — Principais Aplicações de Teleprocessamento

As principais aplicações de T.P. podem ser agrupadas em três categorias:

I — **DATA ENTRY** — permite ao operador do terminal enviar dados da fonte original à U.C.P. diretamente.

II — **INQUIRY** — os dados, centralizados, podem ser examinados através de terminais remotos.

III — **RECORD UPDATE** — permite atualizar dados armazenados em determinados arquivos.

Para exemplificar estas aplicações tomemos um banco de dados de qualquer espécie, dados estatísticos de um estado, por exemplo.

A criação do banco de dados usa o **DATA ENTRY**; as consultas a este banco de dados são efetuadas via técnica **INQUIRY** e, finalmente a atualização de registros deste banco de dados é feita através do **RECORD UPDATE**.

Outras aplicações de T.P. são:

***Processamento remoto por lotes** — onde os dados são enviados para um processamento posterior por lotes.

***Sistema de coleção de dados** — análogo ao anterior com a diferença que os dados são enviados à “estação mestre” à medida que são coletados.

***Computação remota** — Remote Job Entry — o usuário utiliza o sistema via terminal, cabendo-lhe a programação. Desta forma vários usuários remotos utilizam uma grande U.C.P.

***Management Information System — MIS** — permite ao usuário utilizar-se do sistema sem necessidade de percorrer o caminho normal usuário → chefe do C.P.D. → analista → programador → operador.

A escolha de um sistema completo de T.P. deve ser precedido por uma análise precisa e objetiva das necessidades reais que requerem tal sistema.

Podemos resumir a três as características essenciais que devem ser consideradas na escolha e na implantação de um sistema de teleprocessamento:

O valor da informação

A economia da transmissão

A segurança das informações

Finalizando achamos interessante chamar a atenção sobre o fato de o teleprocessamento ser, via de regra, encarado apenas como centralizador de processamentos. Seria interessante porém pensar no teleprocessamento também como um descentralizador de responsabilidades. Isto é importante considerando que na ciência, na administração pública e privada, na política, etc. a filosofia mais adotada modernamente é a da descentralização de responsabilidades.

8 — ALGUMAS CONSIDERAÇÕES SOBRE O /370

No que se refere à operacionalidade do sistema, não existem praticamente diferenças entre o IBM/360 e o IBM/370. As diferenças existem na parte física da máquina isto é, no hardware.

A seguir serão apresentadas sucintamente algumas características do sistema/370 que o diferenciam das anteriores — /360.

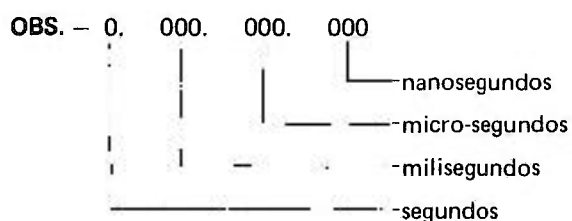
*O /370 é mais veloz que o /360.

*Uma parte da U.C.P. do /370 é de 4ª geração com circuitos monolíticos.

*O ciclo de U.C.P. do /370 é aproximadamente 4 vezes mais rápido que do /360.

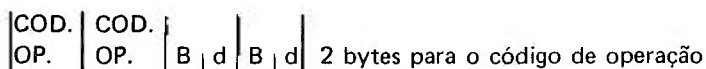
O ciclo de U.C.P. de /360 modelo 65 é de 650 nanosegundos.

O ciclo de U.C.P. do /370 modelo 165 é de 160 nanosegundos.



*O /370 possui, além dos canais Seletor e Multiplexor, o canal Block-Multiplexor que pode trabalhar tanto com unidades de alta como de baixa velocidade.

*Além dos 5 tipos de instruções do /360, o /370 possui um sexto tipo chamado tipo S.



No /360 base + deslocamento = endereço absoluto, correspondendo ao Endereço Real.

No /370 base + deslocamento = endereço absoluto + extensão, correspondendo ao Endereço Real.

*A extensão é acrescentada por hardware e graças a ela é possível mudar um programa inteiro de uma parte a outra da memória sem modificação alguma.

*No /370 existem mais 16 registradores de privilégio do Supervisor, chamados Registradores de Controle.

ANEXO 1

Tabela de Aplicações Usuais dos Utilitários

AÇÃO	APLICAÇÃO	UTILITÁRIO
ALTERA	Nº serial de volume em DASDR	IEHDASD
	Organização de data sets	IEHBUPDTE
	Tamanho do registro lógico	IEBGENER
ANALISAR	Trilhas em acesso direto	IEHATLAS, IEHDASDR, IBCDASDI
ATUALISAR	Data sets particionados, "in place"	IEBUPDTE
CARREGAR	Data sets indexados sequenciais	IEBISAM
CATALOGAR	Data sets	IEHPROGM
COMPARAR	Data sets particionados	IEBCOMPR
	Data sets sequenciais	IEBCOMPR
COMPRIMIR	Data sets particionados, "in place"	IEBCOPY
CONVERTER	Data set indexado sequencial para sequencial	IEBISAM, IEBDG
	Data set particionado para sequencial	IEBUPDTE
	Data set sequencial para particionado	IEBUPDTE, IEBGENER
COPIAR	Catálogo	IEHMOVE
	Data set catalogado	IEHMOVE
	Data set indexado sequencial	IEBISAM
	Data set particionado	IEBCOPY, IEHMOVE
	Data set sequencial	IEBGENER, IEHMOVE, IEBUPDTE
	"Dump" de dados, de fita para DASD	IEHDASDR, IBCDMPRS
	"Job Step"	IEBEDIT
	Membros	IEBUPDTE, IEBGENER, IEBUPDTE, IEBDG
	Volume de acesso direto	IEHDASDR, IEHMOVE, IBCDMPRS
CRIAR	Data set sequencial de saída	IEBDG
	Índice	IEHPROGM
	"Job stream" de saída	IEBEDIT
	Membros	IEBDG
DELETAR	Registros de um data set particionado	IEBUPDTE
	Registro de um membro	IEBUPDTE
DESCARREGAR	Data set indexado sequencial	IEBISAM
	Data set particionado	IEBCOPY, IEHMOVE
	Data set sequencial	IEHMOVE
DESCATALOGAR	Data sets	IEHPROGM

continuação

EDITAR E COPIAR	Data set sequencial	IEBGENER, IEBUPDTE
	"Job stream"	IEBEDIT
EDITAR E IMPRIMIR	Data set sequencial	IEBTPCH
EDITAR E PERFURAR	Data set sequencial	IEBTPCH
EXCLUIR	Membros de um P.D.S. numa operação de cópia . . .	IEBCOPY, IEHMOVE
GERAR	Dados para teste	IEBDG
IMPRIMIR	Data set particionado	IEBTPCH
	Data set sequencial	IEBGENER, IEBUPDTE, IEBTPCH
	Registros selecionados	IEBTPCH
INICIALIZAR	Fita magnética	IEHINITT
	Volumes de acesso direto	IEHDASDR, IBCDASDI
INSERIR	Registros de um data set particionado	IEBUPDTE
INTERCALAR	Data set particionado	IEHMOVE, IEBCOPY
LISTAR	V.T.O.C.	IEHLIST
MODIFICAR	Data set particionado ou sequencial	IEBUPDTE
MOVER	Data set catalogado	IEHMOVE
	Data set particionado	IEHMOVE
	Data set sequencial	IEHMOVE
PERFURAR	Data set sequencial	IEBTPCH
	Membros de um data set particionado	IEBTPCH
	Registros selecionados	IEBTPCH
RENOMEAR	Data set particionado ou sequencial	IEHPROGM
	Membros copiados ou movidos	IEHMOVE
	Membros de data set particionado	IEBCOPY, IEHPROGM
SUBSTITUIR	Membros	IEBUPDTE, IEBUPDATE
	Registros lógicos	IEBUPDTE
'SCRATCH'	Data sets	IEHPROGM
	V.T.O.C.	IEHPROGM

ANEXO 2

COMPARAÇÃO ENTRE LINGUAGENS QUANTO A FACILIDADES
DE "DATA MANAGEMENT"

FACILIDADES DE "DATA MANAGEMENT"	LINGUAGENS				
	ALGOL	COBOL ANS	FORTRAN E, G, H	PL/I	R P G
BUFFER AUTOMÁTICO	SIM	SIM	SIM	SIM	SIM
TÉCNICA DE ACESSO BASIC	SIM	SIM	SIM	SIM	SIM
CONTROLE DE "BUFFER"	SIM	SIM	SIM	SIM	NÃO
CONCATENAÇÃO SEQUENCIAL DE "DATA SETS"	SIM	SIM	SIM	SIM	SIM
PROTEÇÃO DE SENHA ("PASSWORD")	SIM	SIM	SIM	SIM	SIM
PROCESSAMENTO DE DATA SETS MULTIVOLUME	SIM	SIM	SIM	SIM	SIM
GRUPO DE GERAÇÃO DE DADOS (G. D. G.)	SIM	SIM	SIM	SIM	SIM
TROCA DE BUFFERS	NÃO	NÃO	NÃO	NÃO	NÃO
PROCESSAMENTO DE DATA SETS ORGANIZADOS DE FORMA DIRETA	NÃO	SIM	SIM	SIM	SIM
PROCESSAMENTO DE P. D. S.	NÃO	NÃO	NÃO	NÃO	NÃO
PROCESSAMENTO DE DATA SET SEQUENCIAL INDEXADO	NÃO	SIM	NÃO	SIM	SIM
TÉCNICA DE ACESSO QUEUED	NÃO	SIM	NÃO	SIM	SIM
DATA SET SEQUENCIAL: CONTROLE DE DISPOSITIVO	SIM	SIM	SIM	SIM	SIM
DATA SET SEQUENCIAL: INDEPENDÊNCIA DE DISPOSITIVO	SIM	SIM	SIM	SIM	SIM
"BUFFERIZAÇÃO" SIMPLES	SIM	SIM	SIM	SIM	SIM
REGISTROS EM "OVERFLOW"	NÃO	SIM	SIM	SIM	NÃO
OBS.: TODAS AS FACILIDADES SÃO DISPONÍVEIS AO USUÁRIO COM O USO DA LINGUAGEM ASSEMBLER					

ANEXO 3

TIPOS DE SISTEMAS OPERACIONAIS PARA COMPUTADORES IBM/360

I – BPS – BASIC PROGRAMMING SUPPORT

É um sistema utilizado em computadores do modelo 25 ao 50, com memórias de 8 até 64 K. Contém programas de utilidades e SORT/MERGE.

II – BOS – BASIC OPERATING SYSTEM

É o menor dos sistemas operacionais. Aplica-se a computadores do modelo 25 ao 50 com memórias de 8 a 64 K. Exige uma unidade residente em disco magnético. Permite utilizar programas de utilidades de SORT/MERGE.

III – DOS/TOS – DISK OPERATING SYSTEM/TAPE OPERATING SYSTEM

Estes sistemas são aplicados em computadores do modelo 25 ao 75 com memórias de 16 a 512 K. Permitem utilizar programas de utilidade, SORT/MERGE. e Tele-Processamento (só o DOS).

A unidade residente pode ser disco (DOS) ou fita magnéticas (TOS).

IV – OS – OPERATING SYSTEM

É o sistema operacional propriamente dito e o mais completo. Aplica-se a computadores do modelo 25 ao 75 com memória de 64 K em diante.

Permite utilizar todas as facilidades.

NOTA: Na figura abaixo vemos um resumo das linguagens e facilidades suportadas pelos diversos sistemas operacionais.

LINGUAGENS E FACILIDADES	TIPO DE SISTEMA OPERACIONAL			
	BPS	BOS	DOS/TOS	OS
ASSEMBLER				
FORTRAN IV				
CODOL				
PL/I				
RPG				
PROGRAMAS DE UTILIDADES				
SORT/MERGE				
TELEPROCESS- SAMENTO				
CONVERSION				

ANEXO 4

REPERTÓRIO DE ALGUMAS SIGLAS E ACRÔNIMOS MAIS UTILIZADOS EM COMPUTAÇÃO

ABNT ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS	BOS BASIC OPERATING SYSTEM
ACU AUTOMATIC CALLING UNIT	BPAM BASIC PARTITIONAL ACCESS METHOD
ADP AUTOMATIC DATA PROCESSING	BPI BIT PER INCH
ALC AUTOMATIC LAST CARD	BPS BASIC PROGRAMMING SUPPORT
ALGOL ALGORITHMIC – ORIENTED LANGUAGE	BSAM BASIC SEQUENTIAL ACCESS METHOD
ALU ARITHMETIC LOGIC UNIT	BTAM BASIC TELECOMMUNICATION ACCESS METHOD
ANSI AMERICAN NATIONAL STANDARD INSTITUTE	CAI COMPUTER ASSISTED INSTRUCTION
APR ALTERNATE PATH RETRY	CAW CHANNEL ADDRESS WORD
ASA AMERICAN STANDARDS ASSOCIATION	CCB COMMAND CONTROL BLOCK
ASB AUTOMATIC SYSIN BATCH	CCH CHANNEL – CHECK HANDLER
ASCII AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE	CCW CHANNEL COMMAND WORD
BCD BINARY CODED DECIMAL	CDE CONTENTS DIRECTORY ENTRY
BDAM BASIC DIRECT ACCESS METHOD	CHAP CHANGE DISPATCHING PRIORITY
BDW BLOCK DESCRIPTOR WORD	COBOL COMMON BUSINESS ORIENTED LANGUAGE
BISAM BASIC INDEXED SEQUENTIAL ACCESS METHOD	COCR CYLINDER OVERFLOW CONTROL RECORD
BIT BINARY DIGIT	CODASYL CONFERENCE ON DATA SYSTEMS LANGUAGES

COM
COMPUTER OUTPUT MICROFILM

CP
COMMAND PROCESSOR

CPB
CHANNEL PROGRAM BLOCK

CPD
CENTRO DE PROCESSAMENTO DE DADOS

CPI
CHARACTERS PER INCH

CPM
1 – CRITICAL PATH METHOD
2 – CARD PER MINUTE

CPS
CHARACTERS PER SECOND

CPU
CENTRAL PROCESSING UNIT

CRJE
CONVERSATIONAL REMOTE JOB ENTRY

CRT
CATHODE RAY TUBE

CSCB
COMMAND SCHEDULING CONTROL BLOCK

CSW
CHANNEL STATUS WORD

CVT
COMMUNICATION VECTOR TABLE

DADSM
DIRECT ACCESS DEVICE SPACE MANAGEMENT

DAR
DAMAGE ASSESSMENT ROUTINE

DASD
DIRECT ACCESS STORAGE DEVICE

DASDI
DIRECT ACCESS STORAGE DEVICE INITIALIZATION

DCB
DATA CONTROL BLOCK

DD
DATA DEFINITION

DDR
DYNAMIC DEVICE RECONFIGURATION

DEB
DATA EXTENT BLOCK

DFT
DIAGNOSTIC FUNCTION TEST

DOS
DISK OPERATION SYSTEM

DP
DATA PROCESSING

DPC
DATA PROCESSING CENTER

DPRTY
DISPATCHING PRIORITY

DPS
DATA PROCESSING SYSTEM

DS
DATA SET

DSCB
DATA SET CONTROL BLOCK

DSE
DATA SET EXTENSION

DSL
DATA SET LABEL

DSN
Ver DSNNAME

DSNAME
DATA SET NAME

EBCDIC
EXTENDED BINARY CODED DECIMAL INTERCHANGE
CODE

ECB
EVENT CONTROL BLOCK

ECT
ENVIRONMENT CONTROL TABLE

EDP
ELECTRONIC DATA PROCESSING

EDPM
ELECTRONIC DATA PROCESSING MACHINE

EOA
END OF ADDRESS

EOB
END OF BLOCK

EOF
END OF FILE

EOM
END OF MESSAGE

EOV
END OF VOLUME

ERP
ERROR RECOVERY PROCEDURE

ESD
EXTERNAL SYMBOL DICTIONARY

ESS
ELECTRONIC SWITCHING SYSTEM

EVS
ERROR STATISTIC BY VOLUME

ETXR
END-OF-TASK EXIT ROUTINE

EVA
ERROR VOLUME ANALYSIS

EXCP
EXECUTE CHANNEL PROGRAM

EXEC
EXECUTE

EXTRN
EXTERNAL REFERENCE

FBQE
FREE BLOCK QUEUE ELEMENT

FCB
FORM CONTROL BUFFER

FD
FIELD DEFINITION

FIFO
FIRST-IN, FIRST-OUT

FORTRAN
FORMULA TRANSLATION

FOSDIC
FILM OPTICAL SENSING DEVICE FOR INPUT TO
COMPUTER

FPM
FILE PROTECT MODE

FQE
FREE QUEUE ELEMENT

FS
FILE SEPARATOR

GAM
GRAFIC ACCESS METHOD

GDG
GENERATE DATA GROUP

CDS
CONFIGURATION DATA SET

GJP
GRAPHIC JOB PROCESSOR

GM
GROUP MARK

GPR
GENERAL PURPOSE REGISTER

IAS
IMMEDIATE ACCESS STORAGE

IBG
INTER BLOCK GAP

IBM
INTERNATIONAL BUSINESS MACHINES
CORPORATION

IC
INSTRUCTION COUNTER

ICA
INTERNATIONAL COMMUNICATION ASSOCIATION

IDP
INTEGRATED DATA PROCESSING

IMS
INFORMATION MANAGEMENT SYSTEM

INIT
INITIATOR

I/O
INPUT/OUTPUT

IOB
INPUT/OUTPUT BLOCK

IOCS
INPUT/OUTPUT CONTROL SYSTEM

IPL
INITIAL PROGRAM LOADING

IRMS
INFORMATION RETRIEVAL AND MANAGEMENT
SYSTEM

IS
INFORMATION SEPARATOR

ISO
INTERNATIONAL STANDARD ORGANIZATION

JCC
JOB CONTROL CARD

JCL
JOB CONTROL LANGUAGE

JCS
JOB CONTROL STATEMENT

JCT
JOB CONTROL TABLE

JFCB
JOB FILE CONTROL BLOCK

LCB
LINE CONTROL BLOCK

LCS
LARGE CAPACITY STORAGE

LD
LOAD

LE
LINKAGE EDITOR

LIFO
LAST-IN, FIRST-OUT

LIOCS
LOGICAL INPUT/OUTPUT CONTROL SYSTEM

LLE
LOAD LIST ELEMENT

LP
LINEAR PROGRAMMING

LPA
LINK PACK AREA

LPM
LINES PER MINUTE

LPSW
LOAD PROGRAM STATUS WORD

LSQA
LOCAL SYSTEM QUEUE AREA

LTPC
LOCAL TIME PSEUDO-CLOCK

MAR
MEMORY ADDRESS REGISTER

MCAR
MACHINE CHECK ANALYSIS AND RECORDING

MCH
MACHINE-CHECK HANDLER

MCF
MAGNETIC CARD FILE

MCP
MACHINE CONTROL PROGRAM

MCRR
MACHINE CHECK RECORDING AND RECOVERY

MCS
MULTIPLE CONSOLE SUPPORT

MFT
MULTIPROGRAMMING WITH A FIXED NUMBER OF
TASKS

MICR
MAGNETIC INK CHARACTER RECOGNITION

MIS
MANAGEMENT INFORMATION SYSTEM

MODEM
MODULATOR DEMODULATOR

MPS
MULTIPROGRAMMING SYSTEM

MVT
MULTIPROGRAMMING WITH A VARIABLE
NUMBER OF TASKS

NIP
NUCLEUS INITIALIZATION PROGRAM

NL
NO LABEL

NSL
NON STANDARD LABEL

OBR
OUTBOARD RECORDER

OCR
OPTICAL CHARACTER RECOGNITION

OLTEP
ONLINE TEST EXECUTIVE PROGRAM

OS
OPERATING SYSTEM

PAX
PRIVATE AUTOMATIC EXCHANGE

PCP
PRIMARY CONTROL PROGRAM

PDS
PARTITIONED DATA SET

PID
PROGRAM INFORMATION DEPARTMENT

PIOCS
PHISICAL INPUT / OUTPUT CONTROL SYSTEM

PL / I
PROGRAMMING LANGUAGE I

PQE
PARTITION QUEUE ELEMENT

PRTY
PRIORITY

PSA
PREFIXED STORAGE AREA

PSW
PROGRAM STATUS WORD

QCB
QUEUE CONTROL BLOCK

QISAM
QUEUE INDEXED SEQUENTIAL ACCESS METHOD

QSAM
QUEUE SEQUENTIAL ACCESS METHOD

QTAM
QUEUE TELECOMMUNICATION ACCESS METHOD

RAM
RANDON ACCESS MEMORY

RAT
READ ACCESS TIME

RB
REQUEST BLOCK

RCT
REGION CONTROL TASK

RDR
READER

RDW
RECORD DESCRIPTOR WORD

RJE
REMOTE JOB ENTRY

RMS
RECOVERY MANAGEMENT SUPPORT

RO
RECEIVE ONLY

RO / RI
ROLLOUT / ROLLIN

RPG
REPORT PROGRAM GENERATOR

RS
RECORD SEPARATOR

SCC
SEGMENT CONTROL CODE

SCT
STEP CONTROL TABLE

SDR
STATISTICAL DATA RECORDER

SDW
SEGMENT DESCRIPTOR WORD

SER
SYSTEM ENVIRONMENT RECORDING

SIO
START INPUT / OUTPUT

SIOT
STEP INPUT / OUTPUT TABLE

SL
STANDARD LABEL

SMB
SYSTEM MESSAGE BLOCK

SMF
SYSTEM MANAGEMENT FACILITIES

SPI
SINGLE PROGRAM INITIATOR

SPQE
SUB PULL QUEUE ELEMENT

SQA
SYSTEM QUEUE AREA

SUCESU
SOCIEDADE DOS USUÁRIOS DE COMPUTADORES E
EQUIPAMENTOS SUBSIDIÁRIOS

SUL
STANDARD USER LABEL

SVC
SUPERVISOR CALL

SYSCTLG
SYSTEM CATALOG

SYSGEN
SYSTEM GENERATION

SYSIN
SYSTEM INPUT

SYSOUT
SYSTEM OUTPUT

SYSRES
SYSTEM RESIDENCE (VOLUME)

TCAM
TELECOMMUNICATION ACCESS METHOD

TCB
TASK CONTROL BLOCK

TCU
TRANSMISSION CONTROL UNIT

TIOC
TERMINAL INPUT / OUTPUT COORDINATOR

TIOT
TASK INPUT / OUTPUT TABLE

TJID
TERMINAL JOB IDENTIFICATION

TMP
TERMINAL MONITOR PROGRAM

TOD
TIME OF DAY

TOS
TAPE OPERATING SYSTEM

TP
TELEPROCESSING

TQE
TIMER QUEUE ELEMENT

TSC
TIME SHARING CONTROL TASK

TSCE
TIME-SLICE CONTROL ELEMENT

TSD
TIME SHARING DRIVER

TSIP
TIME SHARING INTERFACE PROGRAM

TSO
TIME SHARING OPTION

TSS
TIME SHARING SYSTEM

80

UADS

USER ATTRIBUTE DATA SET

UCB

UNIT CONTROL BLOCK

UCS

UNIVERSAL CHARACTER SET

US

UNIT SEPARATOR

USASCII

VER ASCII

USASI

UNITED STATES OF AMERICA STANDARDS
INSTITUTE

USERID

USER IDENTIFICATION

VDU

VISUAL DISPLAY UNIT

VM

VIRTUAL MACHINE

VS

VIRTUAL STORAGE

VTOC

VOLUME TABLE OF CONTENT

WPM

WORDS PER MINUTE

WTR

WRITER

ABSTRACT

Basic concepts of the Operating System — O.S. — are given we intend to give to the reader a general view of the power and characteristics of the mentioned O.S.

The O.S. characteristics are described briefly but enough to be understood and used.

AGRADECIMENTOS

- Ao Eng^o CIBAR CÁCERES AGUILERA, Coordenador Geral do C.P.D. do I.E.A. pelo apoio e pelo incentivo que sempre prestou no decorrer da preparação do presente trabalho.
- À Sra. LÚCIA FARIA SILVA, por ter sugerido e apoiado a publicação do trabalho.
- À Srta. ELENICE MAZZILLI, pela revisão final do trabalho.
- À Srta. ODETE GUEDES, por ter oferecido condições para a realização do trabalho bem como pelas sugestões a respeito do conteúdo do mesmo.
- **Agradeço de maneira especial** à Srta. VELENTINA MONTIEL FERNANDEZ, com a qual foram discutidos muitos dos itens do trabalho e, principalmente, pela valiosa colaboração que prestou ao ler e sugerir alterações quanto à forma e conteúdo do mesmo.
- A todos aqueles que, de uma forma ou de outra, contribuíram para a realização deste trabalho, principalmente ao grupo de alunos que assistiram ao curso do qual esta publicação é outro resultado.

REFERÊNCIAS BIBLIOGRÁFICAS

1. FRANCO, R. S. *Sistema operacional OS-IBM/360: conceitos e controle de sistema*. São Paulo, PRODESP, Set. 1974.
2. INTERNATIONAL BUSINESS MACHINES, Poughkeepsie, N. Y. *IBM system/360 operating system: concepts and facilities*. 8.ed. Poughkeepsie, N. Y., Jan. 1971. (File n. S360-20, Order n. GC28-6535-8).
3. _____. *IBM system/360 operating system: introduction*. 3 ed. Poughkeepsie, N. Y., Oct. 1969. (File n. S360-20, Order n. GC28-6534-2).
4. _____. *IBM system/360 operating system: MFT guide*. 6.ed. Poughkeepsie, N. Y., July 1969. (File n. S360-36, Order n. C27-6939-5).
5. _____. *IBM system/360 operating system: MVT guide*. 4.ed. Poughkeepsie, N. Y., June. 1971. (File n. S360-36, Order n. GC28-6720-3).
6. INTERNATIONAL BUSINESS MACHINES, São Paulo. *Sumário de O.S.* São Paulo, Centro Educacional IBM, maio 1971.