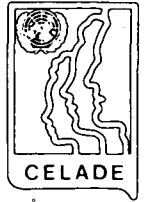


3225 c-2

# Centro Latinoamericano de Demografía

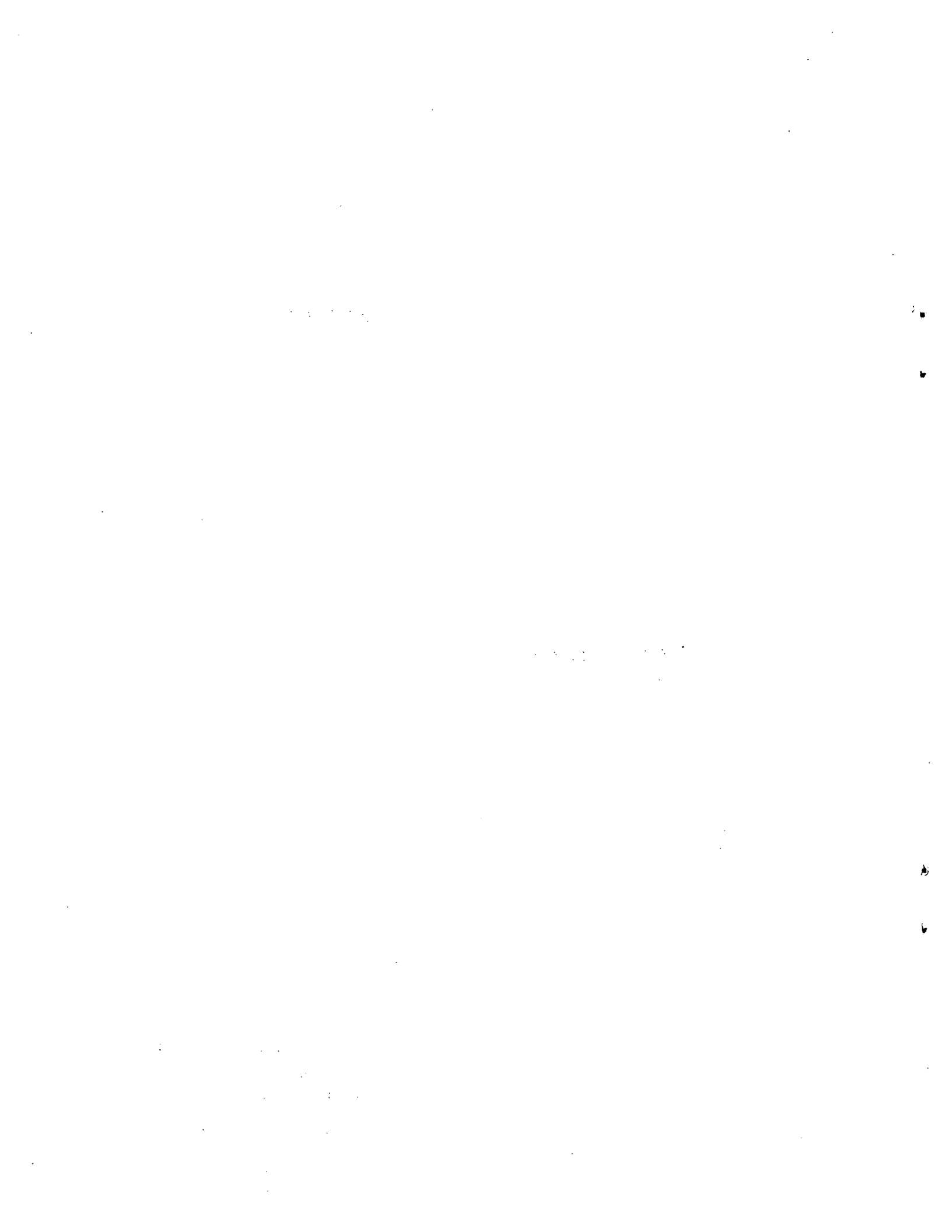


Documentos de Seminarios

SISTEMAS DE OPERACION

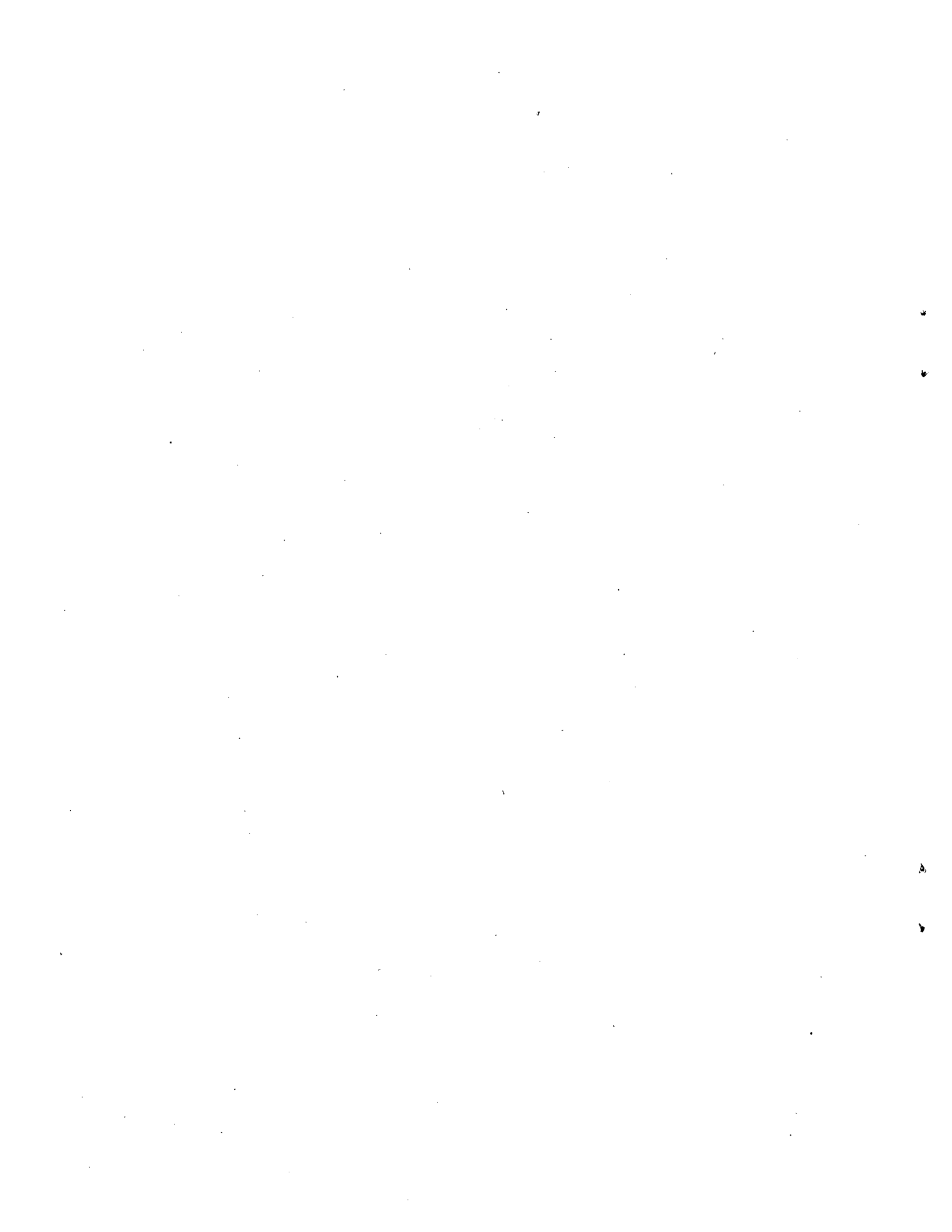
DS/9  
200  
1975.

CURSO LATINOAMERICANO DE  
PROCESAMIENTO DE DATOS (PED)  
APLICADO A LAS CIENCIAS SOCIALES



## I N D I C E

	<u>Página</u>
I. INTRODUCCION .....	1
II. FUNCIONES DEL SISTEMA DE OPERACION .....	7
1. Función de planificación .....	7
2. Función de administración de recursos .....	8
A. Administración de memoria .....	8
B. Administración de tiempo de la UCP .....	11
C. Administración de dispositivos de entrada/salida .....	12
D. Administración de archivos de información .....	12
3. Función de carga .....	13
4. Función de término de programas .....	13
5. Función de comunicación .....	14
III. PUESTA EN MARCHA DEL SISTEMA .....	14
IV. GENERACION DEL SISTEMA .....	14
V. UN SISTEMA DE OPERACION: DISK OPERATING SYSTEM/VIRTUAL STORAGE DE IBM (DOS/VS) .....	15
1. Programas componentes del sistema DOS/VS .....	15
A. Programas de control .....	15
B. Programas de procesamiento .....	15
C. Rutinas de administración de datos .....	16
2. Funciones del sistema de operación .....	16
A. Funciones de control .....	16
B. Utilización de recursos .....	17
3. Uso del sistema de operación .....	25
A. Cargador de programa inicial .....	25
B. Comandos de IPL .....	26
C. Programa Job Control .....	29
D. Programa Linkage Editor .....	40
E. Librarian .....	45
F. Problemas resueltos .....	49
4. Tipos de macro instrucciones en DOS/VS .....	60
A. Macros declarativas DTF (Define the file) y de generación de módulos .....	61
B. Macros imperativas .....	75
C. Problemas resueltos .....	78
BIBLIOGRAFIA .....	82



## I. INTRODUCCION

Se define como Sistema de Operación a un conjunto integrado de programas, diseñado para mejorar la efectividad de la operación total de un Sistema Electrónico de Procesamiento de Datos (computador).

En general está constituido por:

- a) Sistemas de programación (traductores de lenguajes) para ayudar a la formulación de problemas, que serán resueltos mediante el computador.
- b) Programas de control para ayudar a la operación, mantención y administración del sistema.
- c) Programas de chequeo para recuperación de errores.
- d) Rutinas de servicio de bibliotecas (residentes en el computador) para almacenar información para procesos posteriores y para contener al propio Sistema de Operación.
- e) Rutinas de mantención de bibliotecas, que permiten tenerlas actualizadas.

Varios hechos han ido señalizando las distintas etapas que ha tenido el desarrollo de los Sistemas de Operación. Inicialmente la programación se efectuaba en lenguaje de máquina, lenguaje árido de difícil manejo y fuente continua de errores, más aún, complicado y complejo para ubicar esos mismos errores. Esto trajo como consecuencia la creación de programas traductores que se diseñaron para convertir programas escritos en lenguajes simbólicos a los lenguajes de máquina utilizados hasta ese momento. Esos nuevos lenguajes se denominaron "de ensamble" y estaban orientados a la máquina. Les siguieron los lenguajes simbólicos generales, que permitían a los usuarios escribir sus programas en forma parecida a la utilizada en matemáticas (ALGOL, FORTRAN) o en el planteamiento de problemas de tipo comercial (COBOL).

En esta forma los usuarios de computadores tuvieron la posibilidad de escribir ellos mismos sus programas, en muchos casos sin necesitar la ayuda o asistencia de programadores profesionales, quienes, al mismo tiempo, estuvieron en condiciones de desarrollar un trabajo más creativo.

Junto con los programas traductores de lenguajes se crearon los llamados sistemas de control de entrada/salida de información, cuyo objetivo fundamental era disminuir el tiempo ocioso de la Unidad Central de Proceso, debido a que las operaciones de entrada/salida eran relativamente lentas en comparación con la velocidad de ejecución de la UCP. Para lograr la reducción del tiempo ocioso, en primer lugar, se desarrollaron los sistemas de computación, esto es, los equipos, que permitieron realizar en forma paralela operaciones de entrada/salida y de procesamiento de datos. En segundo lugar, se generaron los sistemas de control de entrada/salida de tal manera que permitieron sincronizar en forma automática las operaciones de entrada/salida con las de procesamiento, esto es, el proceso de los datos de entrada no se iniciaba mientras no se hubiera efectuado totalmente la transferencia de ellos desde el dispositivo de entrada y, en forma similar, los resultados no eran destruidos mientras no hubieran sido transferidos a un dispositivo de salida.

También fueron desarrollados programas que correspondían a procesos de uso frecuente de los distintos usuarios. Estos procesos podían ser de cálculo, de clasificación/intercalación o rutinarios, como transcribir información desde un medio de almacenamiento a otro. Los primeros se conocen como subrutinas y los últimos como utilitarios.

Todas estas facilidades de programación permitieron que el volumen de trabajos de procesamiento de datos se incrementara notablemente. Los problemas que se empezaron a presentar fueron de otro tipo. Se relacionaron fundamentalmente con la mayor exigencia que se hacía de parte de los usuarios de una disminución del tiempo de respuesta del sistema de computación. Ocurría que el computador perdía una gran parte de su potencialidad por la dependencia que existía de la acción del operador fuera ésta montaje de carretes de cinta, colocación de tarjetas u otra similar. Esto era extremadamente grave en instalaciones donde se procesaban muchos trabajos chicos, lo que significaba mayor intervención del operador.

Si se analiza cuales son las acciones normales de un operador en un computador sin Sistema de Operación se tiene lo siguiente:

- 1) En un proceso de traducción de un programa a lenguaje de máquina
  - a) Leer instrucciones de partida
  - b) Montar la cinta que tiene el compilador
  - c) Colocar programa fuente en la lectora

Se inicia el proceso y si hay detención

- d) Análisis de la detención (fin de proceso, error)
- e) Se cancela el proceso si hay error.

2) En un proceso de un programa ya traducido

- a) Leer instrucciones de partida
- b) Montar la cinta de datos
- c) Colocar programa objeto en la lectora

Se inicia el proceso y si hay detención

- d) Análisis de la detención (fin de proceso, error)
- e) Se cancela el proceso si hay error.

De los dos ejemplos se concluye claramente que un alto porcentaje de tiempo, posible tiempo productivo, se pierde por intervención del operador. Esto mismo hizo pensar entonces que la forma de obtener un menor tiempo de respuesta era eliminando al máximo la acción del operador y para ello se requería procesar todos los trabajos como uno solo.

Para reducir el tiempo ocioso del sistema de computación se diseñó un programa de control que dirigía la acción de otros programas como los traductores, utilitarios, etc. Junto con ello, como se trataba de reemplazar en lo posible al operador, se creó un lenguaje que permitía la comunicación directa entre el programador o usuario y el programa de control. Mediante este lenguaje (lenguaje de control) se podía entregar al programa de control una definición exacta del trabajo que se deseaba ejecutar a través de instrucciones o proposiciones de control que se registraban en tarjetas perforadas. El programa de control debía estar "capacitado" para interpretar y actuar en conformidad con las proposiciones entregadas.

Otra función que debía cumplir el programa de control era efectuar automáticamente la transición de un trabajo a otro y verificar que se hiciera correctamente. Esto significaba que debía haber una cola de espera de trabajos que necesitaban procesamiento. La cola, o lote (batch) o pila (stack) de trabajos se colocaba en un dispositivo de entrada del sistema, asignado exclusivamente para ese objeto. De este modo, el programa de control podía leer y ejecutar un trabajo tras otro con intervención mínima del operador. Esta técnica se conoce como "procesamiento de trabajos loteados (batched job)" o "procesamiento de trabajos apilados (stacked job)".

Así como se asignó un dispositivo de entrada del sistema, se asignó otro de salida, a través del cual el sistema se comunicaba con el exterior. En algunos casos se utilizó una perforadora de tarjetas, que entregaba normalmente paquetes de tarjetas perforadas correspondientes al resultado de la traducción de un lenguaje. En otros, la mayoría, se usó una impresora que permitía entregar listados del programa fuente, mensajes del sistema de operación, etc.

Dado que en muchos trabajos se requerían unidades de entrada/salida adicionales, aparte de aquéllas asignadas al sistema de operación y que eran "compartidas" por todos los trabajos, el operador inicialmente debía asignar esas unidades, antes de la iniciación del proceso que las utilizaría. Posteriormente, el programa de control se diseñó de tal manera que podía mantener registrada la situación de cada dispositivo (ocupado o desocupado). En esta forma si era requerido, uno de ellos, era asignado desde un conjunto de unidades desocupadas que correspondiera al tipo solicitado. El operador entonces no necesitaba asignarla en forma anticipada.

En algunos sistemas de operación era posible colocar datos de entrada en el mismo dispositivo que se utilizaba para la entrada de las proposiciones de control. Con esto se logró eliminar o disminuir el número de dispositivos de entrada adicionales y disminuyó también la intervención del operador.

Aquellos programas de uso general tales como traductores de lenguajes, utilitarios, programas de clasificación/intercalación, etc. se almacenaron en dispositivos de memoria auxiliar y constituyeron la biblioteca de programas.

En la mayoría de los sistemas de operación el programa de control se subdividió en dos partes: una, llamada "núcleo" porque permanecía siempre en la memoria principal y la otra "transitoria", que se cargaba en memoria principal sólo durante el intervalo entre un trabajo y otro o entre etapas de ellos (job steps). El núcleo estaba compuesto fundamentalmente por rutinas de uso frecuente cuya función era procesar las interrupciones, cargar programas desde la biblioteca a memoria principal, transmitir mensajes al operador, etc. Contenía, además, información sobre los dispositivos de entrada/salida y la fecha de cada día. También formaban parte del núcleo rutinas de supervisión necesarias para controlar las operaciones de entrada/salida. En cuanto a la parte transitoria, su función primordial era interpretar y ejecutar las proposiciones de control leídas. La ejecución de las proposiciones de control constituirá el paso de un trabajo a



otro o de una etapa de un trabajo a otra. Una vez realizada dicha transición, el espacio de memoria principal ocupado por la parte transitoria quedaba disponible para ser ocupado en la ejecución del trabajo o etapa de éste (job o job step).

Un gran impulso al desarrollo de los sistemas de operación le dieron algunas aplicaciones en las que se necesitaba una respuesta rápida a consultas hechas desde dispositivos de entrada/salida locales o remotos. En estas aplicaciones se utilizaron sistemas de operación que se comunicaban directamente con la fuente y destino de los datos que se procesaban. Estos eran enviados y recibidos desde distintos lugares que podían estar cercanos o remotos al computador, en este último caso conectados mediante líneas de telecomunicación. Estos sistemas para poder dar respuestas rápidas hicieron uso, en primer lugar, de los dispositivos de acceso directo que permiten llegar a los registros sin que éstos tengan que ser clasificados en un orden determinado con anterioridad a su utilización. En segundo lugar, se aplicaron nuevas técnicas que permitieron procesar varias transacciones en forma paralela. Para esto, el programa de control mantenía canales de recursos de información y de equipo y los asignaba a medida que fueran requeridos. Si se suspendía el proceso de una transacción, los recursos que dejaba disponibles se asignaban para iniciar el proceso de otra o continuar alguno que hubiera sido interrumpido previamente. Los sistemas que se obtuvieron se conocen como sistemas de tiempo compartido (time-sharing) y se refieren en general a aquéllos en los cuales los usuarios son independientes entre sí respecto al uso del computador o más claramente cada usuario entra información, la procesa y recibe resultados tal como si todo el computador estuviera a su disposición. Como ejemplo típico de estos sistemas está el desarrollado para resolver el problema de venta de pasajes de líneas aéreas desde distintas agencias. En este caso se requerían respuestas rápidas del sistema de tal manera que en una agencia no se vendiera un pasaje para un asiento que acababa de ser usado por otra agencia.

Otro tipo de aplicación es la de tiempo real (real-time) que corresponde a procesos que generan información que es elaborada por el computador y cuyos resultados permiten controlar o alterar los mismos procesos. En esta aplicación el computador está conectado directamente con la fuente emisora y receptora de información; la velocidad de la respuesta varía de acuerdo con el tipo de proceso, que puede ser el control de una máquina herramienta, de una fábrica de papel, de una industria petroquímica o de las luces de tráfico de una ciudad.

Es necesario definir el concepto de multiprogramación y éste corresponde a la ejecución en forma paralela de más de un programa. Para ello hay intercalación de ejecución de un programa con otro de acuerdo con las prioridades de ellos. Es fácil confundir multiprogramación con tiempo compartido, considerando que en ambos casos se trata de efectuar procesos paralelos; sin embargo, en el caso de tiempo compartido hay interacción entre el usuario y su programa, esto es, el usuario está en todo momento al tanto del progreso de su programa, de los errores que ocurren, que puede corregir inmediatamente, y de los resultados. En tiempo compartido hay una conversación entre el usuario y el computador. Se puede decir, sin temor a cometer un error, que tiempo compartido es una opción que puede funcionar bajo multiprogramación. Otro concepto es el de multiprocesamiento que corresponde a una técnica mediante la cual el procesamiento de datos es realizado entre dos o más UCP interconectadas de tal manera que la comunicación entre ellas, directa o indirecta, es realizada por el sistema de operación sin interrupción humana.

Finalmente, se han puesto en ejecución sistemas en los cuales se utiliza el concepto de memoria virtual, esto es, que permiten la ejecución de programas que exceden la capacidad de la memoria principal real disponible. Para ello se hace uso de la técnica de paginación, en la cual el programa se divide físicamente en páginas de las que sólo algunas necesitan residir en determinados momentos en la memoria principal, o la técnica de segmentación, en que el programa se divide en segmentos en vez de páginas; o de ambas técnicas combinadas. Algunos fabricantes establecen una diferencia entre los conceptos de página y segmento. La primera constituiría una unidad de tamaño fijo. En cambio, el segmento representaría partes de programa, relacionadas con la idea que tiene el usuario de la construcción lógica de su programa. Además, el usuario puede referirse a dichos segmentos mediante nombres que los identifican.

Fundamental en la aplicación del concepto de memoria virtual fue la "traducción dinámica de direcciones" (direcciones de memoria virtual a ubicaciones físicas) mediante el mismo equipo (hardware).

## II. FUNCIONES DEL SISTEMA DE OPERACION

### 1. Función de planificación

El objetivo de la función de planificación es seleccionar un trabajo (job) de aquéllos que se encuentren disponibles para ser procesados y dejarlo en condiciones de ser ejecutado. El criterio aplicado para planificar depende de la modalidad de operación: procesamiento loteado (batch), tiempo real o tiempo compartido. Lo normal es asignar prioridades a cada tipo de programa o suceso que ocurra.

En el procesamiento batch la prioridad de ejecución puede estar dada por el orden que tenga el trabajo en la cola de entrada, lo que significa que son procesados a medida que van entrando. Para alterar ese orden, la prioridad de ejecución puede hacerse a través de un parámetro; en este caso, el lote de trabajos se lee y almacena en memoria auxiliar y a continuación son ejecutados de acuerdo con la prioridad que tienen.

Otras posibilidades de planificación importantes son la condicional y la algorítmica. En la primera la ejecución de una etapa del trabajo (job step) puede estar condicionada a la presencia o ausencia de errores en una etapa previa o a la modificación de indicadores internos realizada por esas etapas. En la planificación algorítmica se selecciona el programa de acuerdo con la relación que exista entre determinados factores, relación que es el algoritmo de planificación donde los factores pueden ser, por ejemplo, tiempo estimado de proceso, tiempo que el trabajo ha estado en la cola, etc.

En el procesamiento en tiempo real lo normal es que la función de planificación se obtenga mediante el mismo equipo (hardware). Sin embargo, existen también sistemas en que la planificación se realiza mediante programas (software). Si la función de planificación es asignación de prioridades, se ejecuta siempre el suceso que tenga la más alta, si se presenta uno con prioridad mayor se suspende el que se estaba ejecutando y los recursos del sistema quedan disponibles para el nuevo suceso.

En el procesamiento en tiempo compartido se tiene una rutina de planificación que determina en primer lugar si el sistema está saturado en el momento en que se presenta un nuevo usuario, para permitirle proceder o no. En algunos sistemas, si se autoriza la entrada al usuario, éste debe dar a conocer sus requerimientos

de recursos y no lo planifican mientras esos recursos no estén disponibles. En otros sistemas el usuario queda colocado inmediatamente en una modalidad de ejecución en espera de que los recursos que necesita queden disponibles. Normalmente, en los sistemas que soportan aplicaciones que son de tiempo compartido y aplicaciones que no lo son, tienen prioridad los requerimientos de los usuarios del primer tipo debido a su modalidad de conversación.

La mayoría de los sistemas que trabajan en multiprogramación manejan más de un tipo de modalidad de operación, por ejemplo, procesamiento batch con tiempo compartido. En esos casos, si una modalidad de operación no tiene carga de trabajo, otra de las modalidades que están siendo manejadas utiliza los recursos que han quedado disponibles.

## 2. Función de administración de recursos

En general, los recursos que deben ser administrados por el sistema de operación son: memoria principal, tiempo de Unidad Central de Proceso (UCP), dispositivos de entrada/salida (input/output-I/O) y archivos de información.

### A. Administración de memoria

El problema fundamental que se presenta en la administración de memoria es el de tratarla como un recurso que debe ser desocupado antes de asignarla a un programa distinto del que la está ocupando. El problema, evidentemente, no se presenta en un sistema de proceso serial, en el que todos los recursos están disponibles para el problema que se está ejecutando, sino en los sistemas de multiprogramación de tiempo compartido o de tiempo real, en los cuales hay más de un programa o suceso que está compitiendo por recursos.

La dificultad que se puede presentar es la de entrar en un círculo vicioso, que se origina, por ejemplo, al tener dos programas en memoria, procesándose en forma simultánea. Si ocurre que en un momento determinado los dos programas requieren más memoria, ninguno de los dos puede terminarse hasta que no haya obtenido la memoria que le hace falta y que se la puede proporcionar el otro.

Otro problema que es necesario resolver es el tropiezo que implica la necesidad de volver a utilizar las mismas posiciones de memoria que se tenían antes de desocuparla. Si esta necesidad no se elimina, no es posible atender en forma paralela dos programas que están simultáneamente en memoria y requieren más

almacenamiento, pues uno de ellos, el de prioridad más baja, tendrá que ceder su espacio al de prioridad mayor y no podrá ser cargado nuevamente hasta que no sea desocupada "su" área de memoria. Este problema se conoce como "estancamiento" (deadlock).

Existen cinco métodos básicos que permiten asignar memoria. En el primer método se asignan, en forma estática, áreas fijas de memoria para proceso y esta asignación implica al mismo tiempo prioridad de ejecución. Las áreas se denominan "de primer término" (foreground) y "de último término" (background). Las áreas de primer término están destinadas, en un sistema complejo de multiprogramación, a procesos de tiempo real o de tiempo compartido y el área de último término a proceso batch. En sistemas más sencillos de multiprogramación, las áreas de primer término están destinadas a procesos que tienen un alto porcentaje de uso de la UCP y bajo porcentaje de operaciones I/O; en cambio, el área de último término se destina a procesos de características inversas.

En el segundo método se tiene la memoria dividida en áreas fijas (particiones) de tamaño fijo, en algunos sistemas, y de tamaño variable en otros. Cada partición puede tener un flujo de entrada aparte, o los programas se reciben de un flujo único y se asignan a las distintas particiones de acuerdo con parámetros proporcionados junto con el trabajo o estos se asignan a las particiones disponibles más pequeñas que puedan contenerlos.

Se puede decir que ambos métodos presentan los defectos siguientes:

- a) Cada trabajo debe diseñarse de tal manera que no exceda en su requerimiento de memoria, la máxima que le corresponde a su partición
- b) Durante el tiempo ocioso del trabajo se continúa reservando la memoria de la partición a la que está asignado
- c) En el caso de particiones fijas de tamaño fijo, debe distribuirse la memoria de modo que haya pocas particiones grandes y bastantes particiones pequeñas, pero, esto trae como consecuencia que si hay muchos trabajos chicos el sistema no puede atender a una cantidad mayor debido a las particiones grandes y, por el contrario, si hay trabajos grandes el sistema no puede atender a una cantidad mayor a causa de las particiones pequeñas.

En el tercer método se tiene la memoria libre como un solo grupo de almacenamiento. De este grupo se le asigna a cada trabajo la cantidad exacta de memoria que necesita y cuando el trabajo termina devuelve al grupo la memoria que le ha sido asignada.

A pesar de ser este método más dinámico que los anteriores, se sigue presentando un problema que se denomina de fragmentación. Dado que una vez que se asignó espacio de memoria éste debe quedar reservado hasta que el trabajo sea concluido, existirá una serie de pequeñas áreas disponibles (fragmentos) que en conjunto podrían permitir atender otro trabajo pero que por estar reservadas no pueden ocuparse y además no son contiguas.

En el cuarto método se dividen, tanto la memoria principal como el programa y los datos, en páginas, que son áreas de memoria de tamaño relativamente pequeño. Con este método se soluciona en gran parte el problema de fragmentación por cuanto al ser dividido un programa en páginas se necesitará bastante menos memoria que en el caso de estar agrupado el programa completo; luego, todo el espacio que no se usa queda disponible para otros programas.

Hablar de paginación es hacerlo de memoria virtual y de los conceptos que ella implica. En primer lugar se establece un "espacio de direcciones" (memoria virtual) que empieza en cero y se extiende hasta el máximo permitido por el método de dirección del sistema. Todas las referencias a la memoria principal deben ser hechas en términos de direcciones de memoria virtual. Considerando que la memoria virtual es mayor que la memoria real (física) se debe efectuar o establecer una correspondencia entre la dirección de memoria virtual y la dirección de memoria física. De ahí que se tiene que la definición de dirección de memoria virtual dice que "es un identificador de un pedazo de información requerido, pero no una descripción del punto de memoria principal en que está ese pedazo". La correspondencia es obtenida a través del mismo equipo (hardware), lo que se conoce como "traducción dinámica de direcciones". La parte de memoria virtual que excede a la memoria física se almacena en memoria auxiliar de tal manera que si se solicita una dirección que no esté en la memoria principal se produce una interrupción que permite cargar la página que contiene la información solicitada.

En el quinto método se divide tanto la memoria principal como el programa y los datos en segmentos, los cuales pueden ser identificados mediante nombres por el usuario. La ventaja de este método, aparte de la posibilidad de identificar las particiones, es la de efectuar una división más ligada a la estructura modular que le da el usuario a su programa.

En los sistemas de operación desarrollados para sistemas de computación grandes se ha aplicado una combinación de los dos últimos métodos, esto es, una división en segmentos y dentro de ellos una subdivisión en páginas.

#### B. Administración de tiempo de la UCP

Se puede decir que la administración de tiempo de la UCP es la función modular en un sistema de multiprogramación, tiempo real o tiempo compartido. Se trata de distribuir o compartir el tiempo entre programas que se están ejecutando en forma simultánea. La acción de asignar tiempo a esos programas se denomina "despacho" (dispatch).

Para poder asignar tiempo de UCP se utilizan distintos sistemas de despacho de "colas" de trabajos, los cuales están ordenados dentro de ellas de acuerdo con su prioridad, desde la más alta hasta la más baja. Además de la prioridad en la cola, se tiene la prioridad que le corresponde a la partición y en algunos casos dentro de la partición la prioridad que le ha sido asignada a la clase de trabajo. Así, la cola de trabajos correspondería a los de la misma clase en una partición.

El programa es ejecutado hasta que se produce una interrupción generada por un programa de mayor prioridad (de otra clase o de otra partición) o por el mismo programa debido a necesidad de servicios del programa de control (supervisor). En ese momento la rutina de interrupción coloca al programa cuyo proceso se ha suspendido al final de la cola de despacho.

Hay varias técnicas de despacho que están relacionadas con estructuras de almacenamiento de datos. Se define así la pila (stack) que es una lista lineal en la cual todos los agregados y eliminaciones de elementos se hacen en un extremo de la lista, otra es la cola (queue) que es una lista en la cual todos los agregados de elementos se efectúan en un extremo de la lista y las eliminaciones en el otro extremo. La lista lineal "pila" recibe otros nombres de los cuales los más conocidos son "lista push-down" y "LIFO (last-in-first-out)" y la lista lineal "cola" se conoce como "almacenamiento circular" o como "FIFO (first-in-first-out)".

C. Administración de dispositivos de entrada/salida

La administración de dispositivos de entrada/salida depende de la modalidad de operación que se está utilizando y del método que en dicha modalidad se ha empleado. En el método de división de la memoria en áreas de primer término (foreground) y área de último término (background) los dispositivos de entrada/salida normalmente se asignan en forma permanente a cada una de las áreas. Esas asignaciones pueden ser modificadas mediante comandos que entrega el operador a través del dispositivo de comunicación con el sistema. Con los otros métodos la asignación puede ser estática, en el momento de iniciar un trabajo, o dinámica, durante su ejecución.

D. Administración de archivos de información

Los archivos de información que pueden ser compartidos dentro de un sistema de operación consisten en: bases de datos y recursos de programación como compiladores, rutinas de entrada/salida, etc. Si se trata de bases de datos es necesario contar con algún mecanismo que impida que se intente grabar o leer alguna parte de la base de datos, por un proceso, mientras ella está siendo modificada por otro. Sin embargo, existen sistemas de operación que permiten el acceso al mismo archivo de datos a varios programas que se ejecuten en forma simultánea; para ello normalmente se acepta la lectura del archivo por parte de todos los programas; en cambio, la grabación se autoriza a uno solo de ellos una vez que se ha concluido la operación de leer.

Los recursos de programación pueden ser compartidos en dos formas. Una los considera como recursos que pueden ser utilizados en forma sucesiva de manera serial. Para esto es necesario inicializar aquellas variables cuyo valor haya sido cambiado por procesos previos o de lo contrario tener algún mecanismo que impida el acceso al recurso a un segundo proceso, mientras el primero no haya finalizado. La otra forma es dividir el recurso en dos partes: procedimiento y datos. El procedimiento contiene las instrucciones y los datos corresponden a todas las variables del programa que va a ser compartido. Cada proceso que va a utilizar el recurso se engancha a una copia común de la parte procedimiento y a su vez debe proporcionar "su" copia de la parte datos. Esta técnica ha sido ampliamente aplicada en la programación de subrutinas recursivas, esto es, que pueden llamarse a sí mismas.



### 3. Función de carga

Después de cumplidas las funciones de planificación y de administración de recursos que es posible asignar antes de iniciar la ejecución del trabajo éste se carga en memoria principal.

La carga se efectúa de alguna de las bibliotecas o de archivos temporales y es realizada por un programa "cargador".

La mayoría de los sistemas proporciona facilidades para carga reubicable, esto es, un programa puede ser cargado en cualquier parte de la memoria.

Es posible cargar un programa con una estructura simple (un sólo módulo), con estructura de traslapos (overlays) en la que el programa se divide en varios módulos y estos se cargan a medida que se utilizan, cubriendo total o parcialmente a módulos anteriores, con división del programa en páginas, en segmentos o en una combinación de ambos. Si se utiliza división de páginas o segmentos, o ambos a la vez, se hace uso de tablas que contienen los atributos de las páginas y de los segmentos, por ejemplo, dirección absoluta de memoria, longitud, derecho de acceso, etc.

Otra función que debe cumplir el programa "cargador" es la resolución de los enganches que existen entre las distintas partes del programa cuando éste ha sido seccionado. En otras palabras, combina los módulos objeto (resultados de procesos de traducción) para obtener un programa ejecutable (módulo de carga).

### 4. Función de término de programas

Existen dos posibilidades de término de un programa: normal y anormal. En el primer caso el programa devuelve el control al sistema de operación. En el segundo caso es el sistema de operación el que pone término al proceso de acuerdo con algún tipo de suceso ocurrido, interno o externo (división por cero, registro inexistente, comando del operador, etc.).

Cualquiera que sea el tipo de término, los recursos que han sido asignados al trabajo, como áreas de memoria, dispositivos de entrada/salida, son devueltos al sistema para que puedan ser asignados a otro trabajo. Se entrega, además, por parte del sistema, un resumen de los recursos asignados y utilizados, estadísticas

de errores y los mensajes correspondientes. Conjuntamente con los resultados parciales o totales obtenidos, en algunos casos se proporciona un vaciado (dump) de las zonas de memoria ocupadas, contenidos de registros, indicadores, etc.

### 5. Función de comunicación

La función de comunicación comprende todo el intercambio de información entre el programador o el operador y el sistema de operación. La información puede ser para controlar la ejecución de los trabajos, para configurar los recursos que se necesitarán o para dar cuenta de distintos estados o sucesos del trabajo.

Se tiene comunicación no interactiva e interactiva. La primera corresponde a aquellas modalidades de procesamiento en las cuales la información se entrega al sistema de operación a través de tarjetas de control o de comandos proporcionados mediante una operación de teclado manejado por el operador en su dispositivo de entrada/salida y se recibe del sistema por medio del mismo dispositivo, por una impresora rápida o por otro dispositivo de salida. La segunda se refiere a la comunicación existente en una modalidad de tiempo compartido.

### III. PUESTA EN MARCHA DEL SISTEMA

El operador efectúa la puesta en marcha para iniciar el sistema de operación para proceso normal. El proceso generalmente se realiza cada día una vez que el computador ha sido conectado, principalmente porque la carga de trabajo no es suficiente como para tener el computador funcionando las veinticuatro horas del día o porque es necesario inicializar algún tipo especial de rutinas. En los sistemas de tiempo real, en cambio, se pone en marcha el sistema de operación, que continúa funcionando día tras día. En las dos modalidades, sin embargo, se debe realizar la puesta en marcha cada vez que hay una caída del sistema (detención total por algún tipo de anomalía).

### IV. GENERACION DEL SISTEMA

La generación del sistema consiste en construir un sistema de operación de acuerdo con la configuración del equipo con que se cuenta y con las necesidades específicas de los usuarios de dicho equipo.

La firma vendedora proporciona un sistema de operación maestro a partir del cual se construye el que se utilizará en la instalación. Ese sistema de operación maestro contiene todas las rutinas requeridas para atender cualquier dispositivo de la configuración, como asimismo rutinas opcionales desarrolladas por la firma vendedora. Contiene, además, un conjunto de programas que procesarán la información proporcionada por el usuario para generar el sistema que desea. Esa información estará compuesta por características físicas del equipo (límites de memoria, tamaños de las particiones, tipo de dispositivo, etc.) nombres simbólicos asignados a dispositivos o a clases de dispositivos, traductores que serán incorporados, programas utilitarios, etc.

#### V. UN SISTEMA DE OPERACION: DISK OPERATING SYSTEM/VIRTUAL STORAGE DE IBM (DOS/VS)

##### 1. Programas componentes del sistema DOS/VS

###### A. Programas de control

- a) Cargador de programa inicial (Initial program loader - IPL) que se utiliza para la puesta en marcha del sistema. Carga el programa Supervisor desde un dispositivo de acceso directo a memoria.
- b) Supervisor, controla la operación total del sistema y proporciona funciones generales requeridas por el programa de control de trabajos (jobs) y todos los programas de procesamiento. Reside en el área más baja de memoria denominada área del supervisor.
- c) Programa de control de trabajos (job control programs) es cargado por el supervisor para iniciar la ejecución de cada programa y para establecer cuales facilidades del sistema van a ser invocadas mientras el programa está corriendo.

###### B. Programas de procesamiento

Se puede dividir en tres categorías:

- a) Traductores de lenguajes, que son los que traducen los programas fuente escritos en algún lenguaje de programación a lenguaje de máquina (programa objeto).
- b) Programas de servicio. Entre los más importantes están:
  - i) El Linkage Editor, que convierte los programas objeto a programas objeto ejecutables.

- ii) El Librarian, que realiza funciones de mantención y de servicio de las bibliotecas del sistema. Se tienen tres bibliotecas: la biblioteca imagen de memoria (Core Image Library - CIL) donde están todos los programas objeto ejecutables, la biblioteca reubicable (Relocatable Library - RL) donde están los programas objeto reubicables, esto es, los resultados de procesos de compilación, y la biblioteca de proposiciones fuente (Source Statement Library) donde están programas en lenguaje fuente.
- iii) El Power (Priority Output Writers, Execution Processors and Input Readers) que proporciona la posibilidad de leer y grabar flujos de entrada y salida en dispositivos de memoria auxiliar en forma paralela con la ejecución de trabajos. Esto se conoce como "spooling".
- iv) Emuladores que permiten ejecutar en el Sistema/370 programas escritos para otros sistemas de computación.
- v) Programas de aplicación, escritos por usuarios y en algunos casos proporcionados por IBM, para resolver problemas de tipo científico o comercial.

### C. Rutinas de administración de datos

Permiten liberar al programador de escribir programas para tareas rutinarias de transferencia de datos entre memoria auxiliar y programas.

## 2. Funciones del sistema de operación

### A. Funciones de control

Después que el sistema se pone en marcha por medio del IPL, está listo para aceptar información para ser procesada.

La unidad de trabajo que el usuario entrega para que sea procesada se conoce como "job" y el conjunto de jobs se identifica como "flujo de jobs" (job stream).

Cada job y el ambiente en el cual va a ser procesado se define por medio de "proposiciones de control de jobs" con los cuales se obtiene:

- a) Transición de job a job con intervención mínima por parte del operador.

El comienzo de un job se indica por medio de una proposición de control.

// JOB

Cada job debe estar subdividido en pasos (job step) cada uno de los cuales corresponde a un programa. El job step se identifica con una proposición de control.

// EXEC

- b) Asignación de dispositivos de entrada/salida a nombres simbólicos de dispositivos especificados en los programas.
- c) Carga de programas ejecutables desde bibliotecas a memoria.
- d) Manejo de término de programas.

#### B. Utilización de recursos

a) Administración de memoria. Existen dos modalidades de procesamiento y la memoria es organizada de acuerdo con esas modalidades.

i) Procesamiento batch. La memoria se subdivide en dos áreas. La de orden más bajo se destina al Supervisor y el área restante (background) a los programas que van a ser procesados. En memoria principal puede estar sólo un programa a ejecutar. Se proporciona un sistema de protección de memoria que impide que por efectos del proceso del programa pueda ser destruido el Supervisor.

ii) Procesamiento en multiprogramación. En esta modalidad el área que no es ocupada por el Supervisor puede subdividirse en un máximo de cinco particiones. El área de orden más bajo es el área de último término (background) o de menor prioridad de ejecución; las cuatro áreas restantes o áreas de primer término (foreground) se identifican con los nombres FOREGROUND-4 hasta FOREGROUND-1 y siguen en ese orden al área de último término con su respectiva prioridad de ejecución en el orden inverso.

Cada partición puede contener un programa separado, lo que significa que pueden ser procesados en forma paralela hasta cinco programas.

Como una ampliación de la multiprogramación se puede considerar la modalidad de multitareas (multitasking), en la que se tiene el proceso paralelo de programas o partes de programas en una sola partición. Los programas o partes de programas reciben el nombre de tareas y se hace una diferencia entre los que se inician por el programa control de jobs y los que lo hacen con la macro instrucción ATTACH, que reciben el nombre de subtareas (subtasks). Las subtareas pueden estar relacionadas entre sí en forma lógica o pueden ser totalmente independientes.

El número total de tareas depende del número de particiones; a su vez la cantidad de subtareas no puede exceder de quince.

iii) Memoria virtual. Los programas para computador, en cualquier lenguaje en que ellos estén escritos, contienen instrucciones, descripciones de datos y operaciones de entrada/salida. En esos programas se tienen nombres de datos, de archivos de registros y de instrucciones. Se puede decir, entonces, que los programas fuente residen en un espacio creado por el programador. Ese espacio se denomina "espacio de nombres simbólicos" y normalmente se almacena en tarjetas que sirven de datos al compilador.

El compilador convierte los elementos simbólicos del espacio de nombres en instrucciones, datos, áreas y bloques de control. Después de este proceso llamado traducción, el espacio comprendido entre la dirección más baja y la dirección más alta correspondiente al programa recibe el nombre de "espacio de direcciones". Normalmente este espacio empieza con la dirección cero.

Se pueden combinar espacios de direcciones que han sido resultados de distintos procesos de traducción, operación que realiza, generalmente, un programa de nombre Linkage Editor. El espacio de direcciones del programa se puede almacenar en tarjetas perforadas, en cinta magnética o en dispositivos de acceso directo, desde donde debe ser cargado en la memoria principal para ser ejecutado.

La operación de traducir direcciones del espacio de direcciones en ubicaciones específicas de la memoria real se define como "reubicación" y el tipo de reubicación dependerá del instante en que se efectúe la traducción. Si la traducción se realiza cuando se carga el programa para su ejecución, recibe el nombre de "reubicación estática" y si se realiza durante la ejecución se llama "reubicación dinámica". En el primer caso, el programa está limitado a la capacidad de la memoria real, pues para ser ejecutado debe estar cargado en su totalidad y si no puede ser cargado completo, porque excede la capacidad de memoria real, el programador debe recurrir a técnicas de traslapo (overlays) de secciones del programa. En la reubicación dinámica en cambio, el programador se despreocupa de los límites de la memoria real.

El programa Linkage Editor ofrece la posibilidad de modificar el punto de carga del programa, esto es, cambia la dirección cero, que era el límite inferior del espacio de direcciones del programa, lo cual a su vez permite la modalidad de multiprogramación. A pesar de esta característica el tipo de reubicación continúa siendo estática pues se efectúa una sola vez, antes de la ejecución del programa.

Si se trata de reubicación dinámica, el espacio de direcciones del programa se divide en secciones (segmentos, páginas o ambos) que se cargan en memoria real, en el área que esté disponible, con sus direcciones relativas a la dirección cero del espacio de direcciones, y que sólo se traducen a medida que se está ejecutando la sección y de ahí el nombre de reubicación dinámica, la que puede lograrse con un dispositivo especial conocido como mecanismo de "traducción dinámica de direcciones". El programador, se decía anteriormente, que no se preocupa de los límites de la memoria real porque si en alguna de las secciones se hace referencia a una dirección que está en otra sección que no se encuentra en ese momento en la memoria real, la sección se busca en el dispositivo de acceso directo y se carga en la memoria real.

#### 1) Segmentación

En cualquier sistema de segmentación cada segmento de un programa debe tener un identificador. De esta manera, la dirección de una instrucción, de un área de resultados o de un dato tendrá dos partes:

- la parte ns que representa el nombre del segmento, y
- la parte d que representa la dirección, dentro del segmento.

En el Sistema/370 la estructura de dirección tiene cuatro bytes, de los cuales los dos bytes de orden superior contienen el nombre o "número del segmento" y los restantes la dirección o "desplazamiento".

A medida que los segmentos de un espacio de direcciones se cargan en la memoria real, el sistema de operación construye una tabla de segmentos. Cada entrada en esta tabla contiene la identificación del segmento y el punto de origen de ese segmento en la memoria real. La tabla también se construye en la memoria real y a esa operación se la denomina mapping. Además de la tabla de segmentos se utiliza un registro de control que contiene el punto de origen de la tabla (Segment Table Origin Register-STOR).

Suponiendo que se ejecuta un programa que tiene cuatro segmentos y en el segmento 1 se hace referencia a una dirección relativa 8000, que está dentro del segmento, los pasos que se realizan son los siguientes:

- El registro STOR apunta a la dirección donde está la tabla de segmentos del programa.

- Se busca en la tabla la entrada que corresponde al segmento, en este ejemplo la primera entrada, y se obtiene ahí la dirección donde está cargado el segmento en memoria real.
- La dirección relativa 8000 se suma a la dirección de carga del segmento.

## 2) Paginación

La paginación es otro método que permite al programador construir sus programas despreocupándose de los límites de la memoria real. Consiste en subdividir el programa en partes de tamaño fijo llamadas "páginas". La memoria real también es subdividida en partes de igual tamaño que las páginas y estas partes reciben el nombre de "marcos de página".

El sistema de traducción de direcciones es similar al de la segmentación, esto es, se tiene una tabla de páginas cuyas entradas son el número de la página y el punto de carga de ella, además, se utiliza un registro que contiene la dirección de la tabla de páginas. Si bien es cierto que se pierde la posibilidad de subdividir el programa en secciones cuyo tamaño está en relación con la lógica de estructuración del mismo programa, se tiene en cambio la ventaja de poder cargar las páginas en cualquier marco, pues todos son de igual porte. Al mismo tiempo se pueden hacer subdivisiones más pequeñas.

## 3) Segmentación y paginación

Combinando ambos métodos se logran las ventajas de los dos. La estructura de las direcciones relativas, dentro de cada página, debe ser dada por:

- la parte ns que representa el nombre del segmento,
- la parte np que representa el nombre de la página, y
- la parte d que representa la dirección dentro de la página.

En el Sistema/370 en DOS/VS la estructura de la dirección tiene cuatro bytes, de los cuales los dos de orden superior contienen el nombre o "número del segmento", los cuatro bits que siguen indican el nombre o "número de la página" y los restantes proporcionan el "desplazamiento". La dirección de la tabla de segmentos está en el registro de control número uno.

Cada entrada en la tabla de segmentos está formada por:

- la identificación del segmento,
- la longitud de la tabla de páginas, y
- la dirección de la tabla de páginas, en memoria real.



Cada entrada en la tabla de páginas está formada por:

- la dirección de la página, en memoria real y
- un indicador para saber si la página está en memoria real o no.

Cada segmento tiene 64 K bytes y cada página tiene 2 K bytes, de tal manera que un segmento puede contener hasta 32 páginas.

DOS/VS tiene un espacio de direcciones llamado Memoria Virtual, que empieza en cero y se puede extender hasta 16777216.

La memoria principal se denomina Memoria Real y la parte de Memoria Virtual que puede ser igualada directamente con la Real se llama Area Real de Direcciones. El excedente recibe el nombre de Area Virtual de Direcciones y corresponde a la suma de los tamaños de las particiones virtuales (una partición virtual por partición real) y el tamaño del área virtual compartida (shared virtual area-SVA) que está ubicada en las posiciones de orden superior del Area Virtual de Direcciones y contiene programas ejecutables que pueden ser compartidos entre particiones. La dimensión mínima que debe tener cada partición virtual, incluida como partición la SVA, es de 64 K.

El área virtual de direcciones reside en discos, y el área que ocupa se conoce como "conjunto de datos de página" y contendrá todos aquellos programas o partes de programa que se procesan en modo virtual y para los cuales no hay memoria real disponible. Como se dijo al comienzo, el almacenamiento se divide en segmentos y éstos en secciones que reciben el nombre de páginas, cada una de las cuales tiene 2 K bytes de longitud. El área de memoria real en la que el sistema carga una página para que sea procesada se denomina marco de página (page frame) y también es de 2 K bytes de longitud. El conjunto de marcos de página forma una "agrupación de páginas" (page pool).

La agrupación de páginas está formada por el área que no ocupa el Supervisor y que no se asigna a ninguna partición y debe tener como mínimo 16 K bytes de longitud. A esta área se agregan la memoria real asignada a una partición cuyos programas serán procesados en modo virtual y de cualquier memoria real asignada a una partición la que excede al programa que se procesa en ella.

Si un programa que se está procesando en modo virtual hace referencia a una dirección de memoria contenida en una página que no se encuentra en memoria real, ocurre una "falla de página" (page fault). En este caso se produce una interrupción del proceso, se ubica la página requerida en el conjunto de datos de página

y después se lleva a un marco de página en la memoria real para continuar el proceso interrumpido. Si todos los marcos de página, esto es, la agrupación de páginas, estuvieran ocupados, se ubica un marco de página al cual no se hubiera hecho referencia recientemente y se lo deja disponible para la página que vendrá. Si es necesario, se salvan primero los contenidos del marco ubicado.

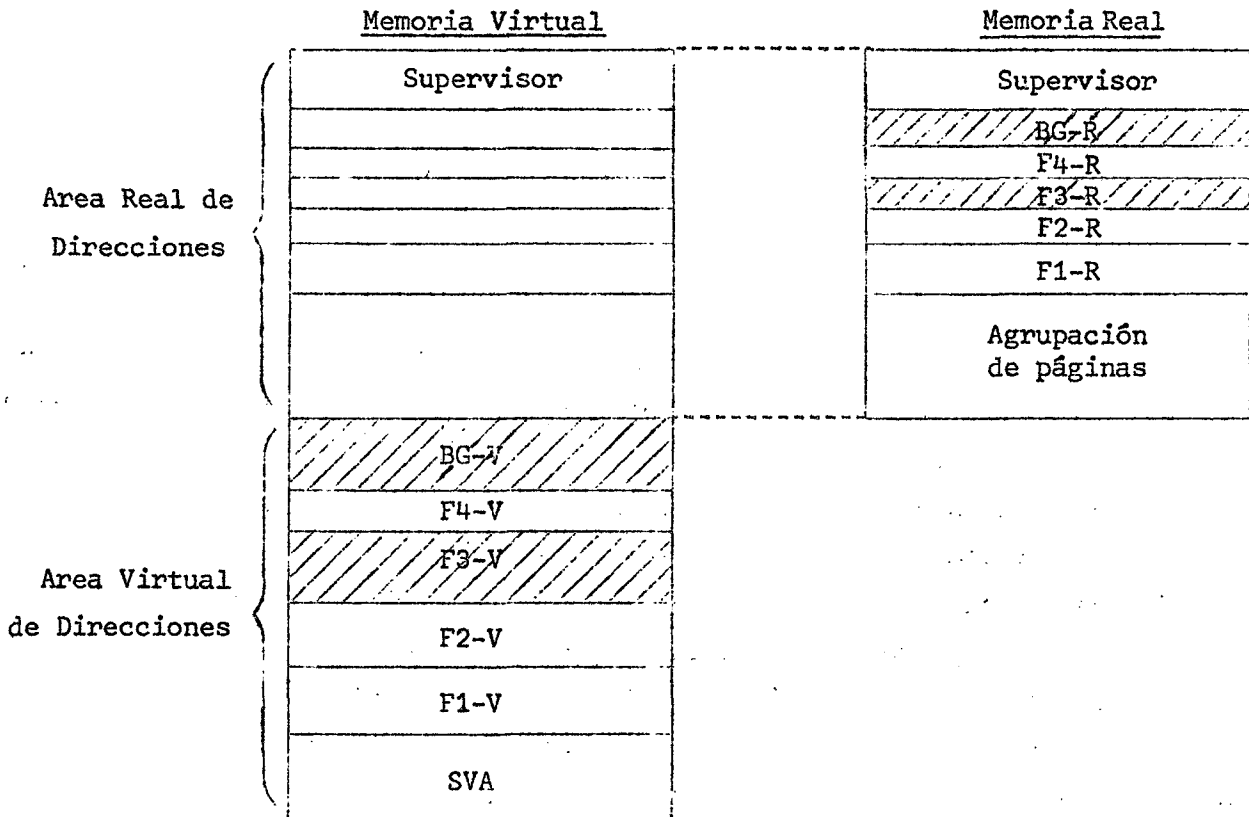
Tomando en cuenta que las páginas de un programa que corre en modo virtual pueden ser cargadas en cualquier marco, no es posible que las direcciones simbólicas contenidas en ellas sean traducidas con anticipación, sino sólo en el momento en que se conoce cuál marco de página ocupará. Este hecho obliga a una "traducción de direcciones", la que ocurre en el transcurso de la ejecución del programa y se obtiene a través de un mecanismo de traducción incorporado en el diseño del Sistema/370.

Las siguientes etapas tienen lugar en la ejecución de un programa en modo virtual:

- Proceso de ensamble o compilación
- Proceso de enganche - edición. El programa Linkage-editor combina (engancha) los resultados de un proceso de ensamble o compilación (módulos objeto) en fases (módulos objeto ejecutables) y los edita en formato "imagen de memoria" en la biblioteca de imágenes de memoria (CIL).
- A medida que el programa se carga para ser ejecutado en modo virtual, se transfieren las páginas desde la CIL a los marcos de página de la agrupación de páginas en la memoria real. Si faltan marcos las páginas son llevadas al conjunto de datos de página hasta que queda cargado el programa completo.
- Al continuar la ejecución del programa aquellas páginas que no están en la memoria real se cargan por el sistema (page-in) desde el conjunto de datos de página a medida que son requeridas. Si el contenido de un marco de página ha sido alterado durante la ejecución, entonces se copia (page-out) en el conjunto de datos antes de cargar la nueva página.
- La traducción de las direcciones relativas se efectúa al ejecutar cada una de ellas, para lo cual se busca en la tabla de segmentos, cuya dirección tiene el registro de control número uno, el segmento que interesa. Junto con la identificación del segmento se encuentra la dirección de la tabla de páginas correspondientes y en ella se busca la página y la

dirección en memoria real donde ha sido cargada. La dirección de carga de la página se suma con la dirección relativa para obtener la dirección absoluta.

Es posible ejecutar programas en modo real en cuyo caso no se efectúa paginación. Sin embargo, la partición real tiene siempre la correspondiente partición virtual, aun cuando no la usa. En la práctica hay programas que tienen que correr en modo real, como es el caso del Supervisor u otros programas cuya efectividad depende del tiempo.



Por ejemplo, en la figura anterior los programas que corren en las particiones background (BG-R) y foreground 3 (F3-R) lo harán en modo real, luego, las áreas virtuales correspondientes (BG-V y F3-V) no serán utilizadas.

iv) POWER. Una manera de disminuir la gran diferencia que existe entre la alta velocidad de la UCP y las velocidades de los dispositivos de entrada/salida que son relativamente bajos, es haciendo uso del programa POWER que efectúa una operación denominada SPOOL (Simultaneous Peripheral Operation on Line) que consiste en leer y grabar flujos de entrada y salida en dispositivos de memoria auxiliar, en forma paralela con la ejecución de los trabajos.

POWER permite la multiprogramación, dado que puede atender en forma simultánea hasta cuatro particiones. Los pasos que se realizan con POWER son los siguientes:

- Lee los flujos de trabajos para cada partición y los almacena en una cola en disco.
- Transfiere los trabajos desde disco a las particiones y los ejecuta.
- Almacena en disco los resultados de los trabajos antes de que sean perforados, impresos o ambas cosas.

v) Bibliotecas. DOS/VS acepta cuatro tipos de bibliotecas: la de imágenes de memoria (core image), la de módulos reubicables (relocatable), la de proposiciones fuente (source statement) y la de procedimientos (procedure). Las tres primeras pueden ser del sistema y privadas, la última puede existir sólo como biblioteca del sistema. Estas están almacenadas en el archivo de residencia del sistema en discos, cuyo nombre simbólico es SYSRES y a ellas pueden tener acceso todas las particiones. Las bibliotecas privadas pueden estar almacenadas en paquetes de discos separados y a ellas pueden tener acceso sólo aquellos programas que están en particiones a las cuales han sido asignadas las bibliotecas.

El resultado de una traducción es un módulo objeto y éste puede ser obtenido en tarjetas perforadas a través de un dispositivo que tiene el nombre simbólico SYSPCH, puede ser catalogado en la biblioteca de módulos reubicables o puede ser almacenado en un área de discos de nombre simbólico SYSLNK.

El programa Linkage-editor que toma la información desde SYSLNK y opcionalmente desde la biblioteca de módulos reubicables realiza el proceso siguiente. El resultado es el programa objeto ejecutable formado por fases, cada una de las cuales tiene una dirección de carga en memoria real y se almacenan en la biblioteca de imágenes de memoria en forma permanente o temporal.

### 3. Uso del sistema de operación

#### A. Cargador de programa inicial (Initial Program Loader-IPL)

La operación del DOS/VS se inicia mediante el procedimiento de carga del programa inicial desde el paquete de discos donde reside el sistema (SYSRES). El operador monta el paquete de discos SYSRES en una unidad y monta además el paquete que contiene el conjunto de datos de página en la unidad asignada al nombre simbólico SYSVIS. Pone la dirección de la unidad donde está SYSRES utilizando las perillas de carga de unidad (load unit switches) presionando a continuación la tecla LOAD en el panel de control del sistema. Esto causa que el primer registro en la pista cero del paquete de discos sea leído a memoria principal a los bytes 0-23. La información transferida consiste de una PSW de IPL y dos comandos de canal (CCW), los que a su vez producen la lectura y carga del IPL en las posiciones más altas de memoria.

Cuando se enciende la luz de WAIT en el panel de control del sistema, se presiona la tecla REQUEST en el dispositivo de comunicación operador-sistema (máquina 3210 ó 3215) a través del cual el sistema envía el mensaje

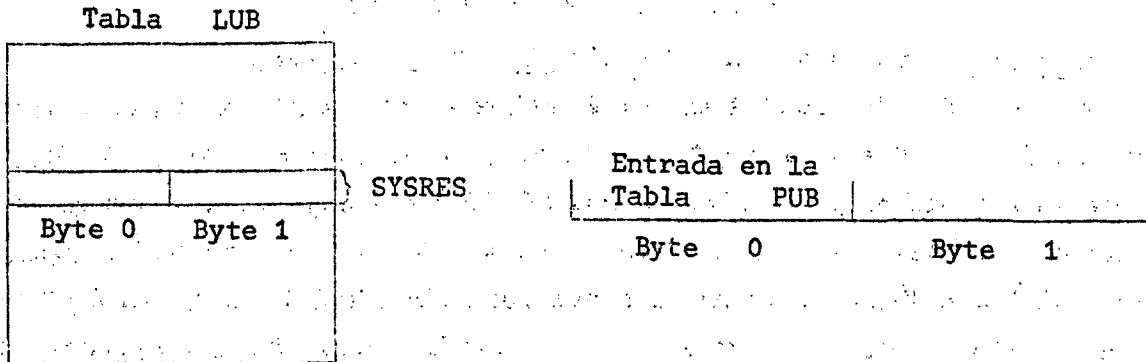
OIO3A SPECIFY SUPERVISOR NAME

el operador entra el nombre del supervisor y después presiona END. Si no se especifica nombre de supervisor y se presiona END asume el sistema el nombre \$\$A\$SUP1.

IPL lee el núcleo del supervisor a las posiciones más bajas de memoria. Si se detecta un error mientras se lee el supervisor, se entra en estado de espera y se pone un código de error en la primera palabra de memoria. Si esto ocurre, se debe reiniciar IPL.

Si no ocurre nada anormal en la transferencia a memoria del núcleo del supervisor, IPL realiza las siguientes operaciones:

i) En la tabla "Bloque de unidades lógicas (Logical Unit Block-LUB)", en los dos bytes que corresponden al nombre simbólico SYSRES, coloca el punto de entrada de ese mismo nombre en otra tabla llamada "Bloque de unidades físicas (Physical Unit Block-PUB)" en la que aparecen las direcciones reales o físicas de los distintos dispositivos que configuran el sistema. Esa dirección real está formada por el número del canal al cual está acoplada la unidad, y el número de esa unidad.



El objeto de esta primera operación de IPL es establecer la dirección real del sistema residente, ya que el paquete de discos puede ser cargado en distintas unidades. Obtiene la dirección de las perillas de carga de unidad donde la estableció el operador.

ii) Coloca la unidad de proceso en estado de espera con todos los códigos de interrupción enmascarados, excepto el de entrada/salida. Se enciende la luz de WAIT y debe presionarse nuevamente REQUEST.

El sistema responderá identificando el archivo SYSRES y la UCP, indicará si se requieren la fecha y la hora y si es necesario un operando relacionado con la ubicación geográfica del sistema, además entregará el mensaje.

OI10A GIVE IPL COMMANDS

A continuación se entregan los comandos que sean necesarios, se presiona la tecla END y el sistema responderá

DOS/VS IPL COMPLETE

o el mensaje

DOS/VS IPL COMPLETE  
1T00A WARM START COPY OF SVA FOUND

después de lo cual el operador entregará los comandos que sean necesarios dependiendo de si se desea mantener o no la copia existente del Area Virtual Compartida (Shared Virtual Area - SVA) y de si se quiere o no hacer uso de las listas de directorio del sistema proporcionadas por IBM.

B. Comandos de IPL

La estructura de los comandos es: código de operación, escrito a partir de la primera posición y seguido al menos por un blanco, y a continuación operandos separados entre sí por coma.

i) Comando ADD. Permite agregar dispositivos a la tabla PUB.

Estructura:

ADD X'cuu' [(k)], tipo de dispositivo  $\left[ \begin{array}{l} ,X'ss' \\ ,X'ssss' \\ ,X'ssssss' \end{array} \right]$

donde:

ADD es el código de operación

X'cuu' son los números de canal (c) y unidad (uu)

k operando optativo, es igual a S si el dispositivo va a ser conmutable, esto es, que está asignado a dos canales adyacentes. El canal designado es el más bajo de los dos. Si el dispositivo no es conmutable, k varía de 0 a 255, siendo 0 la prioridad de atención más alta. Si no se especifica k, se supone prioridad 255.

tipo de dispositivo      código del dispositivo actual      Ej. 2400T9  
3340R, etc.

X'ss' operando optativo. Se utiliza para dar especificaciones de dispositivo, X'ssss' por ejemplo, para indicar el modo de trabajo en cintas magnéticas de X'ssssss, siete o de nueve canales, etc.

ii) Comando DEL. Se utiliza para borrar un dispositivo de la tabla PUB.

Estructura:

DEL X'cuu'

donde:

X'cuu' son los números de canal (c) y unidad (uu)

iii) Comando CAT. DOS/VS proporciona un nuevo método de acceso de archivos, el "Método de acceso de almacenamiento virtual (Virtual Storage Access Method-VSAM)" el cual tiene archivos ordenados secuencialmente por llave, de tal modo que a ellos se puede tener acceso en forma secuencial (SAM) o directamente (DAM). Si se va a utilizar VSAM el comando CAT permite asignar el catálogo de VSAM al archivo de nombre simbólico SYSCAT. El comando CAT debe ser proporcionado después del comando SET y antes de DPD.

Estructura:

CAT UNIT = X'cuu'

donde:

X'cuu' números de canal (c) y unidad (uu) del disco que será asignado a SYSCAT

iv) Comando SET

Estructura:

SET [DATE=valor1,CLOCK=valor2] [,ZONE=  $\left\{ \begin{array}{l} \text{EAST} \\ \text{WEST} \end{array} \right\}$  /hh/mm]

donde:

valor1 especifica día, mes y año con uno de los formatos siguientes:

mm/dd/aa  
dd/mm/aa

en que:

mm=mes, dd=día, aa=año

valor2: especifica la hora con el formato hh/mm/ss

en que:

hh=hora, mm=minuto, ss=segundo

EAST indica que el equipo está ubicado al este de Greenwich

WEST indica que el equipo está ubicado al oeste de Greenwich

hh/mm valor decimal que indica la diferencia en horas y minutos entre la hora local y la hora de Greenwich. El operando hh debe estar en el rango 0-12 y mm en el rango 0-59

v). Comando DPD. Permite definir el conjunto de datos de página. Se debe especificar siempre durante IPL y debe ser el último comando entregado.

Estructura:

DPD.  $\left[ \text{TYPE} = \left\{ \begin{matrix} N \\ F \end{matrix} \right\} \right] \left[ \begin{matrix} \text{,UNIT=X'cuu' ,CYL=xxx} \\ \text{,VOLID=xxxxxxx} \end{matrix} \right]$

donde:

TYPE =N indica que el conjunto de datos de página no necesita ser formateado y los límites del área no han sido cambiados. Cualquiera de las dos condiciones que no se cumpla TYPE=N es ignorado. En este caso los operandos UNIT y CYL deben ser especificados, si es que no lo fueron en el momento de generación del sistema

=F indica que el conjunto de datos de página va a ser formateado durante IPL. Esto es necesario si el conjunto va a ser ampliado o reubicado

UNIT =X'cuu' especifica el número de canal y de unidad del dispositivo en el cual va a estar el conjunto de datos de página. Si se especifica UNIT se debe indicar CYL

CYL =xxx especifica el número de secuencia del cilindro, relativo a cero, donde el conjunto de datos de página va a empezar, el sistema calcula el tamaño del área de discos que se va a utilizar a base de la fórmula:

$$\frac{VSIZE}{2} = \text{número de páginas (bloques de 2 K bytes)}$$

VSIZE es un parámetro con el que se define el tamaño del área virtual en el momento de generar el sistema.



VOLID =xxxxxx especifica el número de serie de volumen del paquete de discos que contiene al conjunto de datos de página

Después de completar el procedimiento de IPL durante el cual se carga el programa de control de trabajos (JOB CONTROL), el sistema está listo para aceptar jobs para ser procesados en el área background. Si se desea procesar en el área foreground, el programa de control de trabajos se carga en la partición deseada en respuesta a los comandos BATCH o START que deben darse después de presionar REQUEST y recibir el mensaje

AR 1I60A READY FOR COMMUNICATIONS

### C. Programa Job Control

Todos los programas que se procesan en el área background se cargan desde la CIL. Si un programa se cataloga en la CIL, job control construye un directorio de fases que ocupa las primeras pistas del área asignada a la biblioteca. Para ejecutar el programa, job control dirige al programa de carga del sistema a dicho directorio, donde se encuentra: la dirección en la biblioteca, el tamaño, el punto de carga, etc. Si el programa está almacenado temporalmente en CIL, se construye un directorio de fases de ese programa con entradas en el directorio de la biblioteca.

El directorio de fases para un programa catalogado incluye una entrada por cada fase de programa. Los nombres de esas fases tienen los mismos cuatro primeros caracteres del nombre que aparece en la proposición de control EXEC.

Job Control se encarga también de la asignación de unidades de entrada/salida físicas. Los programas no se refieren a los dispositivos por sus direcciones físicas, sino por nombres simbólicos. Esto es ventajoso tanto para el operador como para el programador. El programador elige los nombres simbólicos desde un conjunto fijo de ellos, de tal forma que puede escribir programas que sean dependientes sólo del tipo de dispositivo y no de la dirección que tenga uno de ellos en particular.

Sólo en el momento de ejecución el operador o el programador determina cuál es el dispositivo específico que va a ser asignado al nombre simbólico. Esta información se comunica al programa Job Control por medio de las proposiciones de control.

i) Nombres simbólicos (unidades lógicas)

- SYSRDR** Lectora de tarjetas, unidad de cinta magnética, área de discos o área de diskette. Es utilizada por job control para leer sus proposiciones.
- SYSIPT** Lectora de tarjetas, unidad de cinta magnética, área de discos o área de diskette. Es utilizada por los programas como unidad de entrada.
- SYSPCH** Perforadora de tarjetas, unidad de cinta magnética, área de discos o área de diskette. Es utilizada como la unidad principal para salida perforada.
- SYSLST** Impresora, unidad de cinta magnética, área de discos o área de diskette. Es utilizada como la unidad principal para salida impresa.
- SYSLOG** Impresora de teclado, consola de despliegue del operador o impresora. Es utilizada para la comunicación operador-sistema y para imprimir las proposiciones de job control.
- SYSLNK** Área de discos. Usada como dispositivo de entrada por el programa Linkage Editor.
- SYSRES** Área de discos. Dispositivo de residencia del sistema.
- SYSVIS** Área de discos. Usada para soporte de memoria virtual.
- SYSCAT** Área de discos. Se utiliza para el catálogo de VSAM.
- SYSCLB** Área de discos. Se utiliza para la biblioteca imagen de memoria privada.
- SYSRLB** Área de discos. Se utiliza para la biblioteca reubicable privada.
- SYS SLB** Área de discos. Se utiliza para la biblioteca de proposiciones fuente privada.
- SYSREC** Área de discos. Se utiliza para registros de error de transferencia de información.
- SYS000-SYSnnn** Cualquier dispositivo en el sistema. Son unidades lógicas del programador en contraposición a las anteriores que son unidades lógicas del sistema.

Para conveniencia del usuario se definen dos nombres adicionales para asignación de programas en el área background. Ellos son:

- SYSIN** Nombre que puede ser usado cuando SYSRDR y SYSIPT son asignados a la misma lectora de tarjetas o unidad de cinta magnética. Debe ser usado cuando ambos son asignados a la misma área de discos o diskette.
- SYSOUT** Nombre que debe ser usado cuando SYSPCH y SYSLST son asignados a la misma unidad de cinta magnética.

ii) Formato general de las proposiciones

1) Nombre: Dos operadores de división (//) identifican la proposición como proposición de control. Deben estar en las columnas uno y dos respectivamente. Al menos un blanco debe seguir al segundo operador.

Excepciones:

Final de datos (End of data)	/*
Final de trabajo (End of job)	/ε
Comentario	*
Final de procedimiento	/+

2) Operación: Código que describe la operación que va a ser realizada. Puede tener hasta ocho caracteres de largo. Al menos un blanco debe seguir al último carácter.

3) Operandos: Puede ser blanco o contener uno o más parámetros separados entre sí por coma. La información no puede sobrepasar la columna 71, exceptuando las proposiciones de rótulos de archivo, TPLAB y DLAB, cuya información no alcanza a ser contenida en una tarjeta. Para indicar continuación se coloca un carácter distinto de blanco en la columna 72 y en la tarjeta de continuación se empieza en la columna 16.

iii) Lista de las proposiciones

ASSGN	JOB	RSTRT	MTC
CLOSE	LBLTYP	TLBL	/ε
DLAB	LISTIO	TPLAB	/+
DLBL	OPTION	UPSI	/*
DATE	OVEND	VOL	*
EXTENT	PAUSE	XTENT	
EXEC	RESET	ZONE	

iv) Descripción de algunas proposiciones

1) Proposición JOB. Identifica el comienzo de un trabajo.

Estructura:

// JOB nombre del trabajo

donde:

nombre del trabajo: Debe ser de uno a ocho caracteres alfanuméricos. Cualquier comentario del usuario puede aparecer en la proposición JOB después del nombre del trabajo. Si se tiene el medidor de tiempo (timer) aparece la hora en las posiciones 73 a 100 de SYSLST y en las posiciones 1 a 28 de SYSLOG. En ambos casos el formato utilizado es

DATE mm/dd/aa, CLOCK hh/mm/ss

2) Proposición EXEC. Permite iniciar una "etapa de trabajo (job step)".

Precediendo a EXEC puede haber cualquier proposición necesaria para preparar la ejecución del job step. La única limitación es la secuencia para las proposiciones de información de rótulos. Deben estar en el orden que se indica:

```
// VOL
// TPLAB
// VOL
// DLAB
// XTENT
```

Estructura:

```
// EXEC [ [PGM=] nombre de programa ] [ ,REAL ] [ ,SIZE=tamaño ]
        [ PROC=nombre de procedimiento ] [ ,OV ]
```

donde:

nombre de programa: Es el programa, que está en la CIL, que va a ser ejecutado. Puede ser de uno a ocho caracteres alfanuméricos, el primero alfabético. Si el programa que va a ser ejecutado acaba de ser procesado por el programa Linkage Editor, el operando de EXEC debe ser blanco.

REAL: Indica que el job step será ejecutado en modo real. Si el operando se omite, el job step se ejecuta en modo virtual.

tamaño: Define de qué porte es necesaria una partición para el programa que se va a ejecutar. Si se ha especificado REAL, indica el porte de la sección de la parte real que será necesaria para procesar el job step. Si se omite SIZE, la partición real completa se destina al job step. Si no se ha especificado REAL, indica el porte de la sección de la parte virtual que estará disponible para el job step. Si se omite SIZE, la partición virtual completa se destina al job step.

El parámetro SIZE puede especificarse en una de las formas siguientes:

SIZE = nK

SIZE = AUTO

SIZE = (AUTO,nK)

donde:

n: debe ser múltiplo de dos y distinto de cero

AUTO: indica que el tamaño del programa será calculado por el sistema a base de la información contenida en el directorio de la CIL.

nombre de procedimiento: es el procedimiento, que está en la biblioteca de procedimientos, que va a ser ejecutado. Debe ser de uno a ocho caracteres alfanuméricos

OV indica que las proposiciones que siguen a EXEC servirán para modificar el procedimiento catalogado. Las proposiciones que modifican deben tener el mismo identificador que las proposiciones a las que se refieren. El identificador debe aparecer en las columnas 73 a 79. En la columna 80 se puede tener el carácter siguiente:

- A para insertar después de la proposición de referencia
- B para insertar antes de la proposición de referencia
- D para borrar la proposición de referencia
- otro, para reemplazar la proposición de referencia.

3) Proposición ASSGN. Cuando se compilan los programas, usan nombres simbólicos para referirse a dispositivos de entrada/salida. En el momento de ejecución se hace uso de esta proposición para asignar la dirección de un dispositivo específico a los nombres simbólicos usados.

Estructura:

```
// ASSGN SYSxxx, X'cuu' ,TEMP
                    (lista de direc- ,PERM
                    ciones)          ,VOL=número
UA                    ,SHR
IGN                    ,X'ss'
SYSyyy                ,ALT
clase de dispo-      ,H1
sitivo
tipo de dispositivo ,H2
```

donde:

SYSxxx: es el nombre simbólico de la unidad (unidades lógicas)

X'cuu': son los números de canal (c) y unidad (uu)

c=0 a 6      uu = 00 a FE

(lista de direcciones): se pueden especificar hasta siete direcciones de dispositivo en la forma X'cuu' y separadas por coma. El sistema busca en la tabla PUB solamente aquellas unidades indicadas en la lista. Al encontrar una unidad libre la asigna a SYSxxx. En el caso de discos si se especifica SHR se asigna la primera unidad indicada en la lista, aun cuando esté ocupada

UA indica que la unidad lógica va a ser desasignada. Cualquier operación que se refiera a este dispositivo causará la cancelación del job

- IGN indica que la unidad lógica va a ser desasignada y que todas las referencias del programa hacia el dispositivo lógico van a ser ignoradas. La opción no es válida para SYSRDR, SYSIPT, SYSIN y SYSCLB ni para programas en PL/I
- SYSyyy puede ser cualquier unidad lógica. Si se especifica, SYSxxx se asigna al mismo dispositivo al que está SYSyyy
- Clase de dispositivo: se especifica una de las siguientes palabras claves: READER, PRINTER, PUNCH, TAPE, DISK o DISKETTE. El sistema busca en la tabla PUB el primer dispositivo desasignado de la clase indicada y lo asigna a SYSxxx
- Tipo de dispositivo: se especifica un código de dispositivo, tal como 2400T9, 3420T7, etc. El sistema busca en la tabla PUB y asigna a SYSxxx el primer dispositivo libre del tipo especificado. En el caso de discos, si se especifica SHR se asigna el primer dispositivo del tipo indicado, aun cuando esté ocupado
- TEMP: se indica si la asignación va a ser temporal (TEMP) o
- PERM: permanente (PERM)
- número: se especifica el número de serie de volumen, del dispositivo requerido. Este parámetro se puede especificar sólo si se utilizan cintas o discos. El sistema efectúa un chequeo del rótulo del volumen montado para verificar si es el que se ha pedido con el mensaje
- ```
1T50A MOUNT nnnnnn ON X'cuu'
```
- SHR: se especifica sólo para dispositivos de discos y se utiliza en combinación con: lista de direcciones, clase de dispositivo y tipo de dispositivo. Se asigna a SYSxxx la primera unidad del tipo, clase o dirección aun cuando esté ocupada. Si el parámetro no se indica, el sistema asigna la unidad a un dispositivo de discos privado
- X'ss': número hexadecimal que permite indicar características de cinta magnética. Si no se especifica, el sistema asume las características definidas en el momento de generación del mismo
- ALT: permite especificar una unidad de cinta magnética alternativa de aquella asignada a SYSxxx. La cinta alternativa se utiliza cuando la capacidad de la cinta original se ha completado. El parámetro no es válido para SYSRDR, SYSIPT, SYSIN, SYSLNK, SYSCLB y SYSLOG
- H1: se indica que en las máquinas multifuncionales 2560 ó 5425, se utilizará

H2: para entrada, el depósito uno (H1) o el depósito dos (H2). Si no se especifica ninguno de los dos parámetros, el sistema supone H1. Si el programa POWER sustenta la partición, no se puede especificar H2.

4) Proposición OPTION. Permite especificar una o más opciones de control de trabajos. Las opciones tienen vigencia sólo durante el job.

Estructura:

```
// OPTION      opción 1  [,opción 2...]
```

las opciones pueden ser:

LOG para listar todas las proposiciones y comandos de control en SYSLST

NOLOG no se imprimen las proposiciones y comandos de control válidos, sólo se imprimen los no válidos

DUMP en el caso de un término anormal de programa, se vacían en SYSLST los contenidos de los registros y de las particiones virtual y real provisoria, si fueron asignadas

NODUMP suprime el efecto de la opción DUMP

LINK si se especifica, el resultado de procesos de traducción se graba en SYSLNK, de donde lo tomará posteriormente el programa Linkage Editor. Esta opción siempre debe preceder a la proposición EXEC LNKEDT en el flujo de entrada. La opción se cancela después de EXEC sin operando

NOLINK suprime el efecto de la opción LINK

DECK el resultado de procesos de traducción se obtiene en SYSPCH. Si se ha especificado LINK, la opción es aceptada por PL/I, FORTRAN IV, COBOL ANS y DOS/VS y el lenguaje de ensamble

NODECK suprime el efecto de la opción DECK

EDECK permite que el lenguaje de ensamble perfore todas las definiciones de macro fuentes en formato de edición en SYSPCH. El resultado se puede catalogar en la sub-biblioteca E de la biblioteca de proposiciones fuente

NOEDECK suprime el efecto de la opción EDECK

ALIGN permite que el lenguaje de ensamble deje constantes y áreas con el alineamiento de direcciones utilizadas en instrucciones de máquina

NOALIGN suprime el efecto de la opción ALIGN

LIST si se especifica, los traductores de lenguajes entregan los módulos fuente en SYSLST. El lenguaje de ensamble entrega, además, el listado del módulo objeto en hexadecimal y junto con FORTRAN proporciona un resumen de todos los errores encontrados en el programa fuente

- NOLIST suprime el efecto de la opción LIST. Anula también la impresión del diccionario de símbolos externos, del diccionario de lista de reubicación y de la lista de referencias cruzadas (XREF).
- LISTX si se especifica, los traductores COBOL ANS y DOS/VS entregan el mapa de la PROCEDURE DIVISION en SYSLST. Los traductores PL/I y FORTRAN entregan los módulos objeto en SYSLST
- NOLISTX suprime el efecto de la opción LISTX
- SYM si se especifica, los traductores COBOL ANS y DOS/VS y COBOL entregan el mapa de la DATA DIVISION en SYSLST. El traductor PL/I entrega la tabla de símbolos en SYSLST
- NOSYM suprime el efecto de la opción SYM
- XREF si se especifica, el lenguaje de ensamble entrega la lista de referencias cruzadas simbólicas en SYSLST
- NOXREF suprime el efecto de la opción XREF
- ERRS si se especifica, los traductores FORTRAN, COBOL ANS y DOS/VS y PL/I entregan un resumen de los errores contenidos en el programa fuente, en SYSLST
- NOERRS suprime el efecto de la opción ERRS
- ACANCEL indica que el job debe ser cancelado si se intenta asignar un dispositivo en forma infructuosa. Esto puede ser debido a dispositivo no definido, a estado no válido de dispositivo, etc.
- NOACANCEL suprime el efecto de la opción ACANCEL
- CATAL indica que una fase o programa va a ser catalogado en la CIL al término de un proceso de LinkageEditor, CATAL implica la opción LINK. La opción se cancela después de EXEC sin operando.
- STDLABEL indica que todos los rótulos de cinta o de dispositivos de almacenamiento de acceso directo (DASD), proporcionados después de este punto, se graban al comienzo de la pista de rótulos estándar. Se vuelve a la opción USRLABEL (rótulos del usuario) al término del job o job step.
- USRLABEL indica que todos los rótulos de cinta o DASD, proporcionados después de este punto, se graban al comienzo de la pista de rótulos del usuario
- PARSTD indica que todos los rótulos de cinta DASD, proporcionados después de este punto, se graban al comienzo de la pista de rótulos estándar de la partición. Se vuelve a la opción USRLABEL al término del job o job step



48C se utiliza con PL/I y especifica que se tiene el conjunto de 48 caracteres en SYSIPT

60C se utiliza con PL/I y especifica que se tiene el conjunto de 60 caracteres en SYSIPT

SYSPARM = 'cadena de caracteres'

especifica un valor para el símbolo variable del lenguaje de ensamble, \$SYSPARM. La cadena puede contener hasta ocho caracteres EBCDIC encerrados entre comillas

SUBLIB=DF indica que el lenguaje de ensamble DOS/VS y el programa ESERV recuperan macros no editadas y copia "libros" desde la sub-biblioteca D de la biblioteca de proposiciones fuente (SSL) y para recuperar macros editadas desde la sub-biblioteca F en vez de las sub-bibliotecas A y E respectivamente. La opción permanece vigente hasta el término del job o hasta encontrar la opción SUBLIB=AE

SUBLIB=AE indica que el lenguaje de ensamble y el programa ESERV recupera las macros no editadas y copia "libros" desde la sub-biblioteca A y recupera macros editadas desde la sub-biblioteca E.

5) Proposición LISTIO. Permite obtener un listado de todas las asignaciones hechas de entrada/salida. La proposición se ignora si no se ha asignado SYSLST.

// LISTIO

{  
 SYS  
 PROG  
 BG  
 F1  
 F2  
 F3  
 F4  
 ALL  
 SYSxxx  
 UNITS  
 DOWN  
 UA  
 X'cuu'

donde:

SYS: se listan las unidades físicas asignadas a todas las unidades lógicas del sistema en background

PROG: se listan las unidades físicas asignadas a todas las unidades lógicas del programador en background

BG se listan las unidades físicas asignadas a todas las unidades lógicas en background

F1-F4 se listan las unidades físicas asignadas a todas las unidades lógicas de la respectiva partición foreground

ALL se listan las unidades físicas asignadas a todas las unidades lógicas

SYSxxx se listan las unidades físicas asignadas a la unidad lógica especificada. No es válido el parámetro para SYSOUT y SYSIN

UNITS se listan las unidades lógicas asignadas a todas las unidades físicas

DOWN se listan todas las unidades físicas indicadas como no operativas

UA se listan todas las unidades físicas que no están asignadas a una unidad lógica

X'cuu' se listan las unidades lógicas asignadas a la unidad física especificada.

6) Proposición PAUSE. Permite producir una interrupción temporal (pausa) del proceso. La proposición aparece en SYSLOG. El proceso continúa al presionar la tecla END o la ENTER en SYSLOG.

Estructura:

```
// PAUSE [cualquier comentario del programador]
```

7) Proposición RESET. Permite volver las asignaciones hechas en la partición donde se entrega RESET, a las estándares, que son aquéllas que se especifican en el momento de generar el sistema, más cualquier modificación hecha por el operador mediante un comando ASSGN sin la opción TEMP.

Estructura:

```
// RESET {
        SYS
        PROG
        ALL
        SYSxxx
    }
```

donde:

SYS vuelve todas las unidades lógicas del sistema a sus asignaciones estándares

PROG vuelve todas las unidades lógicas del programador a sus asignaciones estándares

ALL vuelve todas las unidades lógicas a sus asignaciones estándares

SYSxxx vuelve la unidad lógica indicada a su asignación estándar. No se puede especificar SYSIN o SYSOUT

8) Proposición CLOSE. Se utiliza para cerrar una unidad lógica asignada a cinta magnética. La operación consiste en grabar una "marca de cinta" (tape mark) un registro de salida EOV (end of volume), dos marcas de cinta, rebobinar y descargar la cinta.

Estructura:

```
// CLOSE  SYSxxx  [ ,X'cuu' ,X'ss'
                   ,UA
                   ,IGN
                   ,ALT ]
```

donde:

SYSxxx puede ser: SYSPCH, SYSLST, SYSOUT o SYS000 a SYSnnn

X'cuu' indica que la unidad lógica, después de haber sido cerrada se asignará al canal (0-6) y unidad (0-254, en hexadecimal 00-FE) especificados

X'ss' ver proposición ASSGN

UA indica que la unidad lógica va a ser cerrada y desasignada

IGN indica que la unidad lógica va a ser cerrada y quedar ignorada. El parámetro no es válido para SYSRDR, SYSIPT o SYSIN

ALT indica que la unidad lógica va a ser cerrada y se va a abrir como unidad alternativa. El parámetro es válido sólo para SYSPCH, SYSLST o SYSOUT asignadas a cinta magnética.

9) Proposición DATE. Permite colocar una fecha en el área de comunicaciones del Supervisor. La fecha se aplica sólo al job que se está ejecutando. Tiene dos formatos posibles.

Estructura:

```
// DATE      mm/dd/aa
// DATE      dd/mm/aa
```

donde:

dd = día (01 a 31)

mm = mes (01 a 12)

aa = año (00 a 99)

10) Proposición MTC. Permite realizar operaciones de control en cinta magnética.

Estructura:

```
// MTC  operación ,SYSxxx [ ,nn ]
```

donde:

operación es uno de los códigos siguientes:

BSF retrocede un archivo (backspace file)

BSR retrocede un registro físico (backspace record)

ERG borra un trozo de cinta (erase gap)

FSF avanza un archivo (forward space file)

FSR avanza un registro físico (forward space record)

RUN rebobina y descarga (rewind and unlcad)

REW rebobina (rewind)

WTM graba marca de cinta (write tape mark)

SYSxxx nombre simbólico de la unidad donde se efectúa la operación

nn es un número decimal de dos dígitos (01 a 99) que indica la cantidad de veces que se realizará la operación. Si no se especifica se supone "una vez".

#### D. Programa Linkage Editor

Todo programa antes de ser ejecutado debe ser procesado por el linkage editor el cual transforma módulos objeto reubicables en módulos objeto ejecutables (fases). Los módulos objeto reubicables, que son resultados de procesos de traducción de lenguajes de computador, pueden estar catalogados en forma permanente en la biblioteca de módulos reubicables o pueden haber sido dejados temporalmente en el área de discos SYSLNK. El resultado del proceso de linkage editor lo deja éste en la CIL, en forma temporal o en forma permanente si se especifica la opción CATAL.

El programa Linkage Editor puede correr tanto en el área background como en foreground. En foreground se debe asignar una biblioteca privada de imagen de memoria (SYSCLB).

Los módulos reubicables que resultan de procesos de traducción están formados por los siguientes tipos de tarjetas:

ESD diccionario de símbolos externos, el cual permite resolver los enganches que quedaron pendientes durante la traducción

TXT tarjetas de texto propiamente tales

RLD diccionario de lista de reubicación, el cual permite modificar los símbolos reubicables (constantes de dirección)

END final del módulo

Entre las tarjetas RLD y END el programador puede colocar tarjetas de reemplazo (REP) de texto.

Además de las tarjetas mencionadas, el módulo que procesa linkage editor también puede contener proposiciones de control de dicho programa. El formato de las proposiciones de control es similar al de las proposiciones del lenguaje de ensamble. El campo de operación debe estar precedido y seguido por uno o más blancos y el campo de operandos no puede continuar más allá de la columna 71 y no debe contener blancos.

Las proposiciones de linkage editor son: PHASE, INCLUDE, ACTION y ENTRY.

i) Proposición PHASE. Indica el comienzo de una fase. Debe preceder al primer módulo reubicable de cada fase procesada. Proporciona el nombre de la fase y la dirección en memoria donde va a ser cargada.

Estructura:

PHASE nombre, origen [,NOAUTO] [,SVA] [,PBDY]

donde:

nombre: es el identificador de la fase. Puede ser de uno a ocho caracteres alfanuméricos

origen: indica la dirección en memoria donde será cargada la fase. Si se da la dirección en relación al comienzo de una partición, el linkage editor utiliza la dirección de comienzo de la partición virtual para calcular el punto de carga de la fase.

Si se desea correr el programa en una partición real, debe ser procesado por linkage editor con ACTION REL de modo que pueda ser reubicado a una partición real cuando se cargue en memoria o escribir el programa de modo que sea auto-reubicable o procesarlo con linkage editor con una proposición PHASE que tenga la dirección absoluta donde el programa será cargado en la partición real.

El parámetro puede ser entregado en una de las formas siguientes:

- |                                     |                            |
|-------------------------------------|----------------------------|
| 1) símbolo [(fase)] [+ reubicación] | } direcciones<br>relativas |
| 2) * [+ reubicación]                |                            |
| 3) S [+ reubicación]                |                            |
| 4) ROOT                             |                            |
| 5) + desplazamiento                 | } direcciones<br>absolutas |
| 6) F + dirección                    |                            |

- en que:
- símbolo puede ser el nombre de una fase definida previamente, el de una sección de control o un rótulo externo que es el operando de una proposición ENTRY en un programa fuente
- (fase) si el parámetro anterior (símbolo) es un nombre de sección de control o de rótulo externo que aparece en más de una fase, el nombre de la fase específica que debe haber sido definida anteriormente se indica entre paréntesis
- reubicación indica que el origen estará desplazado con respecto al símbolo, hacia atrás o hacia adelante, en el valor de la constante hexadecimal (uno a seis dígitos X'hhhhh') o de la decimal (uno a ocho dígitos ddddddd) o de nK
- \* se asigna como origen de la fase la siguiente ubicación de memoria en la partición virtual, alineada en una dirección múltiplo de doble palabra.  
Si se trata de la primera proposición PHASE en el área background, \* indica que el origen estará en la primera doble palabra después del área de salvación de rótulos, si los hay, y del área común asignada, si la hay.
- S Si se especifica, el origen se determina en la misma forma que con el subparámetro \* en una primera proposición PHASE
- ROOT indica que la fase será fase raíz, esto es, ninguna fase posterior podrá traslaparla, de lo contrario se especificará un diagnóstico de desastre. El origen se determina en la misma forma que con el subparámetro S. Sólo la primera fase puede ser raíz
- desplazamiento se especifica el punto de origen como una dirección absoluta, relativa a cero. El desplazamiento se da como una constante hexadecimal, como una constante decimal o como nK en igual forma que el subparámetro reubicación
- F+dirección permite que el origen del programa que se está procesando por linkage editor en una partición sea ubicado al comienzo de otra partición que no está asignada. La dirección se puede especificar como una constante hexadecimal (cuatro a seis dígitos), como una constante decimal (cinco a ocho dígitos), o como nK (n, de dos a cuatro dígitos)

NOAUTO indica que la búsqueda automática en las bibliotecas reubicables se suprime. La búsqueda se efectúa para resolver las referencias externas sin solucionar

SVA indica que la fase será procesada en el área SVA (shared virtual área)

PBDY se especifica que la fase será procesada por linkage editor en alineamiento de página.

ii) Proposición INCLUDE. Permite incluir módulos objeto reubicables para que sean procesados por linkage editor. Tiene dos parámetros como operandos, que pueden ser omitidos. Si se omiten, le indican al programa job control que copie en el área SYSLNK lo que hay en SYSIPT. La operación termina al detectarse fin de datos. Si está solamente el primer operando, se incluye un módulo desde la biblioteca reubicable. Si están los dos operandos, el módulo se lee desde la biblioteca reubicable y de él se extraen las secciones de control especificadas en el segundo operando. Si se indica solamente el segundo operando, debe estar precedido por una coma. En este caso se extraen las secciones de control del módulo que figura a continuación de la proposición INCLUDE en el área SYSLNK, previamente el programa job control debe haber copiado todo desde SYSIPT a causa de un INCLUDE sin operandos.

Estructura:

INCLUDE [nombre de módulo] [,(lista de nombres)]

donde:

nombre de módulo es el nombre con el cual se recuperará el módulo desde la biblioteca reubicable.

(lista de nombres) nombres de las secciones de control que serán extraídas del módulo reubicable. La cantidad total de secciones de control especificadas en la lista no puede exceder de cinco, sin embargo, puede indicarse cualquier número de proposiciones INCLUDE. Una fase puede contener un módulo completo y además algunas secciones de control del mismo módulo.

iii) Proposición ENTRY. Todo conjunto de datos de entrada para el programa linkage editor debe terminar en la proposición ENTRY. Si el programador no la incluye, se utiliza la que genera el programa job control cuando es leída la proposición EXEC LNKEDT.

## Estructura:

ENTRY [punto de entrada]

## donde:

punto de entrada: nombre simbólico del punto al cual se le entregará el control (punto de entrada) en el momento de la ejecución. Debe ser el nombre de una sección de control o de una definición de rótulo que aparezcan en la primera fase. Si el parámetro no se especifica, el linkage editor utiliza como punto de entrada la dirección proporcionada en la tarjeta END. Si END no tiene la dirección buscada, el punto de entrada será la dirección de carga de la primera fase.

iv) Proposición ACTION. Se utiliza para indicar distintas opciones a linkage editor. Debe ser la primera proposición en el conjunto de datos de entrada para el linkage editor. Si es necesario, se pueden especificar varias proposiciones ACTION seguidas.

## Estructura:

ACTION {  $\frac{REL}{NOREL}$  } [ ,CLEAR ] [ ,MAP  
[ ,NOMAP ] [ ,NOAUTO ] [ ,CANCEL ] [ ,BG  
[ ,Fn ]

## donde:

REL indica que la fase que se va a producir será reubicable, lo cual depende del origen especificado en la proposición PHASE. Si en el sistema se tiene el programa que permite cargar en forma reubicable (relocating loader), se supone la opción REL

NOREL indica que la fase que se va a producir no será reubicable. Si en el sistema no se tiene el relocating loader, se supone la opción NOREL

CLEAR indica que el área no utilizada de la CIL se llenará con ceros binarios antes del proceso del linkage editor. La opción produce gran consumo de tiempo por lo cual, en lo posible, debe evitarse su uso

MAP permite que los mensajes de diagnóstico y un mapa de memoria virtual se obtengan a través de SYSLST. El mapa contiene los nombres de las secciones de control que componen las fases y los nombres de los puntos de entrada en cada sección de control

NOMAP suprime el efecto de la opción MAP. Los mensajes de error se obtienen a través de SYSLOG

NOAUTO suprime la característica de AUTOLINK para el programa completo



CANCEL se cancela automáticamente el job si se produce algún error del tipo 2100I al 2170I. Si el parámetro no se especifica y ocurre alguno de los tipos de errores indicados, el job continúa. Algunos errores del tipo 2100I al 2170I son, por ejemplo, operación no válida en la proposición, la proposición se extiende más allá de la columna 71, nombre de fase duplicado, la biblioteca reubicable no está presente, etc.

BG se coloca la dirección de término del supervisor al comienzo de la  
F1-F4 partición virtual especificada más los bytes ocupados por el área de salvación y los ocupados por el área de rótulos.

Los programas que tienen un origen de fase igual a S o a\* se cargan en la partición virtual indicada.

#### E. Librarian

Se denomina así a un conjunto de programas que atienden las bibliotecas del sistema realizando funciones de mantenimiento, servicio y copia.

i) El programa MAINT, que es llamado mediante la proposición

```
// EXEC MAINT
```

ejecuta las funciones de mantenimiento.

Esas funciones son:

- catalogar (catalog)
- borrar (delete)
- renombrar (rename)
- condensar (condense)
- reubicar (reallocate)
- actualizar (update)

Algunas proposiciones de uso frecuente:

1) Proposición CATALR. Permite catalogar un módulo en la biblioteca de módulos reubicables.

Estructura:

CATALR nombre del módulo [,v.m]

donde:

nombre del módulo es el identificador del módulo y tiene de uno a ocho caracteres y no debe tener asterisco como primer carácter

v.m indica número de versión (v = 0-127) y número de modificación (m = 0-255). Si el parámetro no se especifica, el sistema supone 0.0.

2) Proposición CATALS. Permite catalogar un libro en una sub-biblioteca de la biblioteca de proposiciones fuente.

**Estructura:**

CATALS sub-biblioteca, nombre del libro [,v.m [c]]

donde:

sub-biblioteca puede ser cualquier carácter alfanumérico que represente una sub-biblioteca

nombre del libro es el identificador del libro y tiene de uno a ocho caracteres alfanuméricos, el primero de los cuales debe ser alfabético

v.m indica número de versión y de modificación

C indica que los números especificados en v.m deben ser verificados.

Si se van a catalogar definiciones de macros en la sub-biblioteca de lenguaje de ensamble, deben estar precedidas por la proposición MACRO y seguidas por la proposición MEND. Si se trata de otro tipo de libros, deben estar precedidos y seguidos por la proposición BKEND, cuya estructura es la siguiente:

BKEND [sub-biblioteca, nombre del libro], [verifica secuencia], [cantidad], [CMPRSD]

donde:

sub-biblioteca, nombre del libro: es el mismo parámetro de la proposición CATALS

verifica secuencia: puede ser SEQNFS para verificar la secuencia en las columnas 73 a 78 o SEQNCE, para verificar en las columnas 77 a 80

cantidad: indica el número de tarjetas imagen en el libro, incluidas las dos proposiciones BKEND

CMPRSD: indica que el libro está en formato comprimido, esto es, que se han eliminado todos los caracteres blanco.

Todos los parámetros son opcionales y deben estar en el orden indicado. Se utilizan sólo en la proposición que encabeza el libro.

3) Proposición DELETR. Permite borrar módulos de la biblioteca reubicable.

Estructura: Tiene tres formatos posibles.

DELETR nombre de módulo [, nombre de módulo, ...]

DELETR prog1.ALL [, prog2.ALL, ...]

DELETR ALL

La primera permite borrar uno o más módulos, la segunda uno o más programas completos y la tercera toda la biblioteca.

4) Proposición DELETS. Permite borrar libros de la biblioteca de proposiciones fuente

Estructura: Tiene tres formatos posibles

DELETS sub-biblioteca.libro1 [,sub-biblioteca.libro2,...]

DELETS sub-biblioteca.ALL

DELETS ALL

La primera permite borrar uno o más libros, la segunda una sub-biblioteca completa y la tercera toda la biblioteca.

5) Proposiciones RENAMC y RENAMR. Permiten cambiarle el nombre a una fase en la CIL y a un módulo en la RL respectivamente.

Estructura:

RENAMC nombre antiguo,nombre nuevo [,nombre antiguo,nombre nuevo,...]

RENAMR nombre antiguo,nombre nuevo [,nombre antiguo,nombre nuevo,...]

6) Proposición RENAMS. Permite cambiarle el nombre a un libro en la SSL.

Estructura:

RENAMS sub-biblioteca.nombre antiguo, sub-biblioteca.nombre nuevo [,sub-biblioteca.nombre antiguo, sub-biblioteca.nombre nuevo,...]

7) Proposiciones CONDS. Permite condensar, esto es, eliminar los espacios desocupados producto de la función borrar (delete), de una o más bibliotecas.

Estructura:

CONDS biblioteca [,biblioteca,...]

ii) Seis programas realizan la función de servicio. Ellos son:

DSERV para desplegar los directorios de cada una de las bibliotecas

CSERV para desplegar o perforar fases desde la CIL o para ambas cosas a la vez

RSERV para desplegar o perforar módulos desde la RL o para ambas cosas a la vez

SSERV para desplegar o perforar libros desde la SSL o para ambas cosas a la vez

ESERV para desplegar o perforar, verificar, etc. macros en lenguaje de ensamblaje editadas desde la SSL, o para ambas cosas a la vez

PSERV para desplegar o perforar procedimientos desde la biblioteca de procedimientos (PL), o para ambas cosas al mismo tiempo todos los programas son llamados con // EXEC

Algunas proposiciones de uso frecuente

1) Proposición DSPLY. Permite desplegar directorios o contenidos de bibliotecas.

Ejemplo. Despliegue de contenido de la CIL.

Se utiliza la proposición DSPLY con uno de los formatos siguientes:

```
DSPLY fase1 [,fase 2,...]
DSPLY prog1.ALL [,prog2.ALL,...]
DSPLY ALL
```

La primera permite desplegar una o más fases, la segunda uno o más programas completos y la tercera toda la biblioteca.

2) Proposición PUNCH. Permite perforar contenidos de bibliotecas.

Ejemplo. Perforación de contenidos de la RL.

Se utiliza la proposición PUNCH con uno de los formatos siguientes:

```
PUNCH módulo 1 [,módulo 2,...]
PUNCH prog1.ALL [,prog2.ALL,...]
PUNCH ALL
```

La primera permite perforar uno o más módulos, la segunda uno o más programas completos y la tercera toda la biblioteca.

3) Proposición DSPCH. Permite desplegar y perforar contenidos de bibliotecas.

Ejemplo. Despliegue y perforación de contenidos de la SSL.

Se utiliza la proposición DSPCH con uno de los formatos siguientes:

```
DSPCH sub-biblioteca.libro1 [,sub-biblioteca,libro2,...] [,CMPRSD]
DSPCH sub-biblioteca.ALL [,sub-biblioteca.ALL,...] [,CMPRSD]
DSPCH ALL [,CMPRSD]
```

La primera despliega y perfora uno o más libros, la segunda una o más sub-bibliotecas completas y la tercera toda la biblioteca. En los tres casos se puede perforar en formato comprimido utilizando el parámetro CMPRSD.

iii) El programa CORGZ ejecuta la función de copia. Realiza las siguientes operaciones:

- Definir o crear un paquete de sistema nuevo
- Definir o crear bibliotecas privadas
- Transferir fases, módulos o libros entre bibliotecas asignadas.

El programa se llama mediante la proposición // EXEC. Junto con la que se deben proporcionar otras como // DLBL y // EXTENT de job control, que no han sido tratadas en el texto.

#### F. Problemas resueltos

Se utilizan las siguientes asignaciones: X'001' lectora de tarjetas (SYSRDR y SYSIPT), X'00A' lecto-perforadora de tarjetas (SYSPCH), X'180' y X'181' cintas magnéticas.

1) Obtener el deck de la compilación de un programa en FORTRAN y de otro en ASSEMBLER. Catalogar en CIL el módulo de assembler más un módulo MOD1 de la RL. Ejecutar el módulo de fortran más un módulo MOD2 de la RL. Ejecutar el programa catalogado en CIL.

```

SYSRDR,SYSIPT
// JOB EJ1
// OPTION DECK
// EXEC FFORTRAN
    } programa en fortran
/*
// EXEC ASSEMBLY
    } programa en assembler
/*
* PONER DECK DE ASS.
// PAUSE EN X'00A'
// ASSGN SYSIPT,X'00A'
// OPTION CATAL
    PHASE PRØG1,S
    INCLUDE
    INCLUDE MØD1
// EXEC LNKEDT

```

```

* PØNER DECK DE FØRT.
// PAUSE EN X'00A'
// ØPTION LINK
  INCLUDE
  INCLUDE MØD2
// EXEC LNKEDT
// RESET SYS
// EXEC
  }datos
/*
// EXEC PRØG1
  }datos
/*
/ε

```

2) En el programa que figura a continuación indicar: ¿Qué módulos quedan en la X'180'? ¿Qué módulos quedan en la X'181'? ¿Cuál es el conjunto de datos de entrada para el segundo // EXEC LNKEDT? ¿Cuál es la estructura del programa FASE1? ¿Qué lenguajes componen el programa FASE2?

```

  SYSRDR, SYSIPT
// JØB EJ2
// ØPTION DECK
// ASSGN SYSPCH, X'180'
// ASSGN SYSIPT, X'00A'
// EXEC ASSEMBLY
// CLØSE SYSPCH, X'181'
// EXEC FFØRTRAN
* MØNTAR CARRETE EN LA
// PAUSE UNIDAD X'180'
// CLØSE SYSPCH, X'180'
// MTC BSF, SYSPCH
// EXEC CØBØL
// CLØSE SYSPCH, X'00A'
// ASSGN SYSIPT, X'001'

```

```
// EXEC ASSEMBLY
    } programa en assembler
      CSG,CSH
```

```
/*
```

```
* PONER DECK DE ASS.
```

```
// PAUSE EN LA X'00A'
```

```
// OPTIØN CATAL
```

```
// ASSGN SYSIPT,X'00A'
```

```
ACTIØN MAP
```

```
PHASE FASE1,RØØT
```

```
INCLUDE
```

```
// EXEC LNKEDT
```

```
* MØNTAR CARRETE EN
```

```
// PAUSE X'180'
```

```
// ASSGN SYSIPT,X'180'
```

```
// MTC FSF,SYSIPT
```

```
ACTIØN CLEAR
```

```
PHASE FASE2,*
```

```
INCLUDE
```

```
// MTC BSF,SYSIPT,2
```

```
INCLUDE
```

```
// EXEC LNKEDT
```

```
* MØNTAR CARRETE EN
```

```
// PAUSE X'181'
```

```
// ASSGN SYSIPT,X'181'
```

```
ACTIØN NOMAP
```

```
PHASE FASE3,*
```

```
INCLUDE
```

```
// EXEC LNKEDT
```

```
/&
```

```
En la X'00A' se tiene:
```

```
    } programa en assembler
      CSA,CSB
```

```
/*
```

```
    } programa en fortran
      CSC,CSD
```

```

/* } programa en cobol
   } CSE,CSF
/* }
   } tarjetas en blanco

```

Al ejecutarse el programa se tienen los siguientes resultados:

| <u>X'180'</u> | <u>X'181'</u>     | <u>SYSLNK (2)</u> |
|---------------|-------------------|-------------------|
| ESD           | ESD               | ACTION CLEAR      |
| TXT CSA       | TXT CSC           | PHASE FASE2,*     |
| TXT CSB       | TXT CSD           | ESD               |
| RLD           | RLD               | TXT CSE           |
| END           | END               | TXT CSF           |
| TM            | TM                | RLD               |
| EØV           | EØV               | END               |
| TM            | TM                | ESD               |
| TM            | TM                | TXT CSA           |
| ESD           |                   | TXT CSB           |
| TXT CSE       | <u>SYSLNK (1)</u> | RLD               |
| TXT CSF       | ACTION MAP        | END               |
| RLD           | PHASE FASE1,ROØT  | ENTRY             |
| END           | ESD               |                   |
| TM            | TXT CSG           | <u>SYSLNK (3)</u> |
| EØV           | TXT CSH           | ACTION NØMAP      |
| TM            | RLD               | PHASE FASE3,*     |
| TM            | END               | ESD               |
|               | ENTRY             | TXT CSC           |
|               |                   | TXT CSD           |
|               |                   | RLD               |
|               |                   | END               |
|               |                   | ENTRY             |

De acuerdo con los resultados obtenidos, las respuestas son: En la unidad X'180' quedan los módulos de assembler y de cobol. En la unidad X'181' queda el módulo de fortran. El conjunto de datos de entrada para el segundo // EXEC LNKEDT se muestra en SYSLNK (2). El programa FASE1 está formado por las secciones de control CSG y CSH. El programa FASE2 está compuesto por los módulos de cobol y assembler.



SYSRDR, SYSIPT

```

3) // JØB EJ3
    // ØPTIØN LINK,LIST
    // ASSGN SYSIPT,X'00A'
    PHASE FASE1,S
    INCLUDE MØD1,(CSA,CSC)
    PHASE FASE2,FASE1
    INCLUDE
    // EXEC ASSEMBLY
    INCLUDE
    PHASE FASE3,FASE(FASE2)
    INCLUDE,(CSE,CSG,CSH)
    INCLUDE
    // EXEC LNKEDT
    // RESET ALL
    // EXEC
    }datos
/*
/ε

```

En la X'00A' (lecto-perforadora) se tiene:

```

    INCLUDE ,(CSR,CST,CSW)
    ESD
    TXT CSR
    TXT CSS
    TXT CST
    TXT CSU
    TXT CSW
    RLD
    END
    PHASE FASE4,*
/*
    } programa en assembler
    } CSK,CSL
/*
    INCLUDE MØD2

```

```

/* ESD
  TXT CSE
  TXT CSF
  TXT CSG
  TXT CSH
  TXT CSI
  RLD
  END

```

/\*

En la biblioteca de módulos reubicables se tiene:

| <u>MØD1</u> | <u>MØD2</u>  | <u>MØD3</u> |
|-------------|--------------|-------------|
| ESD         | INCLUDE MØD3 | ESD         |
| TXT CSA     |              | TXT CSX     |
| TXT CSB     |              | TXT CSY     |
| TXT CSC     |              | TXT CSZ     |
| TXT CSD     |              | RLD         |
| RLD         |              | END         |
| END         |              |             |

Al ser ejecutado el programa, se asigna a SYSIPT la unidad física X'00A' después de abrir el área SYSLNK con la opción LINK.

Se tiene como resultado en SYSLNK:

|                        |   |                                          |
|------------------------|---|------------------------------------------|
| PHASE FASE1,S          | } | copiadas por el job control desde SYSRDR |
| INCLUDE MØD1,(CSA,CSC) |   |                                          |
| PHASE FASE2,FASE1      |   |                                          |
| INCLUDE ,(CSR,CST,CSW) | } | copiadas por job control desde SYSIPT    |
| ESD                    |   |                                          |
| TXT CSR                |   |                                          |
| TXT CSS                |   |                                          |
| TXT CST                |   |                                          |
| TXT CSU                |   |                                          |
| TXT CSW                |   |                                          |
| RLD                    |   |                                          |
| END                    |   |                                          |
| PHASE FASE4,*          |   |                                          |

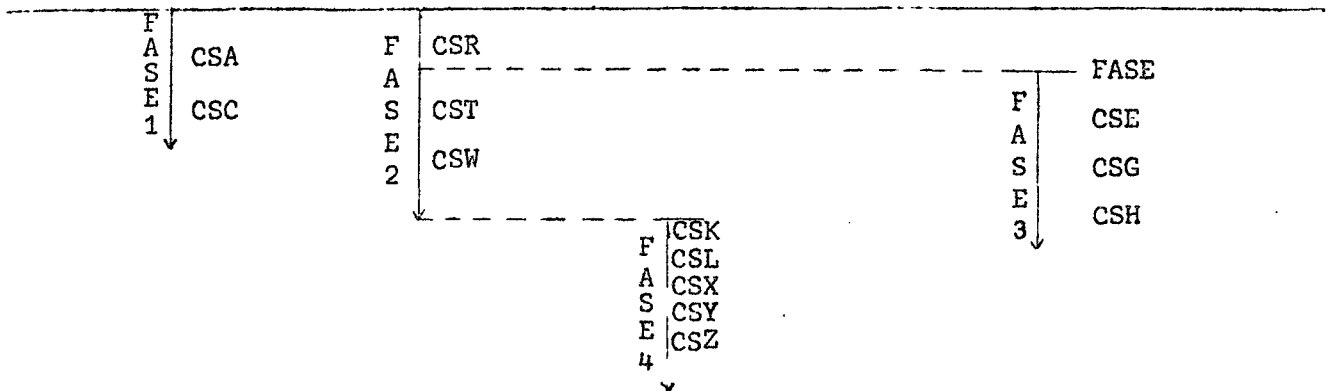
|                         |   |                                                                          |
|-------------------------|---|--------------------------------------------------------------------------|
| ESD                     | } | resultado del proceso<br>de ensamble                                     |
| TXT CSK                 |   |                                                                          |
| TXT CSL                 |   |                                                                          |
| RLD                     | } | copiado por job control desde<br>copiado por job control<br>desde SYSRDR |
| END                     |   |                                                                          |
| INCLUDE MØD2            |   |                                                                          |
| PHASE FASE3,FASE(FASE2) | } | copiado por job control<br>desde SYSRDR                                  |
| INCLUDE ,(CSE,CSG,CSH)  |   |                                                                          |
| ESD                     |   |                                                                          |
| TXT CSE                 | } | copiado por job control<br>desde SYSIPT                                  |
| TXT CSF                 |   |                                                                          |
| TXT CSG                 |   |                                                                          |
| TXT CSH                 |   |                                                                          |
| TXT CSI                 |   |                                                                          |
| RLD                     |   |                                                                          |
| END                     | } | generado por job control al aparecer<br>// EXEC LNKEDT                   |
| ENTRY                   |   |                                                                          |

El programa linkage editor genera las fases en forma temporal en la CIL. La estructura de las fases es la siguiente:

| FASE1 | FASE2 | FASE4 | FASE3 |
|-------|-------|-------|-------|
| CSA   | CSR   | CSK   | CSE   |
| CSC   | CST   | CSL   | CSG   |
|       | CSW   | CSX   | CSH   |
|       |       | CSY   |       |
|       |       | CSZ   |       |

Al ser cargadas en memoria formarán la estructura que se indica a continuación:

SUPERVISOR



Se ha supuesto la existencia de un rótulo FASE dentro de FASE2 y que FASE4 se ejecuta a continuación de FASE2

4) SYSRDR, SYSIPT

```
// JOB EJ4
// ASSGN SYSIPT,X'00A'
// OPTION LINK,LIST,SYM,NODUMP
ACTION CANCEL
PHASE FASE1,R00T
INCLUDE
// RESET ALL
// EXEC FF0RTRAN
    } programa en fortran
      CSC,CSD
/*
INCLUDE
PHASE FASE2,*
INCLUDE M0D1,(CSF,CSH)
/*
// ASSGN SYSIPT,X'00A'
// EXEC ASSEMBLY
// ASSGN SYS008,X'180'
// ASSGN SYS008,X'181',ALT
* MONTAR CARRETES EN
// PAUSE X'180' Y X'181'
// RESET SYS
INCLUDE
PHASE FASE3,*+1024
INCLUDE M0D3
/*
// EXEC LNKEDT
// EXEC
    } datos
/*
/£
```

En la X'00A' se tiene:

ESD  
TXT CSA  
TXT CSB  
RLD  
END

```
/*      } programa en assembler
/*      }      CSJ,CSK
```

En la biblioteca de módulos reubicables se tiene:

| <u>MØD 1</u> | <u>MØD 2</u> | <u>MØD 3</u> |
|--------------|--------------|--------------|
| INCLUDE MOD2 | ESD          | ESD          |
| ESD          | TXT CSE      | TXT CSL      |
| TXT CSH      | TXT CSF      | RLD          |
| TXT CSI      | TXT CSG      | END          |
| TXT CSJ      | RLD          |              |
| RLD          | END          |              |
| END          |              |              |

Con la ejecución se tiene primero en SYSLNK

```
ACTION CANCEL
PHASE FASE1,RØØT
ESD
TXT CSA
TXT CSB
RLD
END
ESD
TXT CSC
TXT CSD
RLD
END
PHASE FASE2,*
INCLUDE MØD1,(CSF,CSH)
```

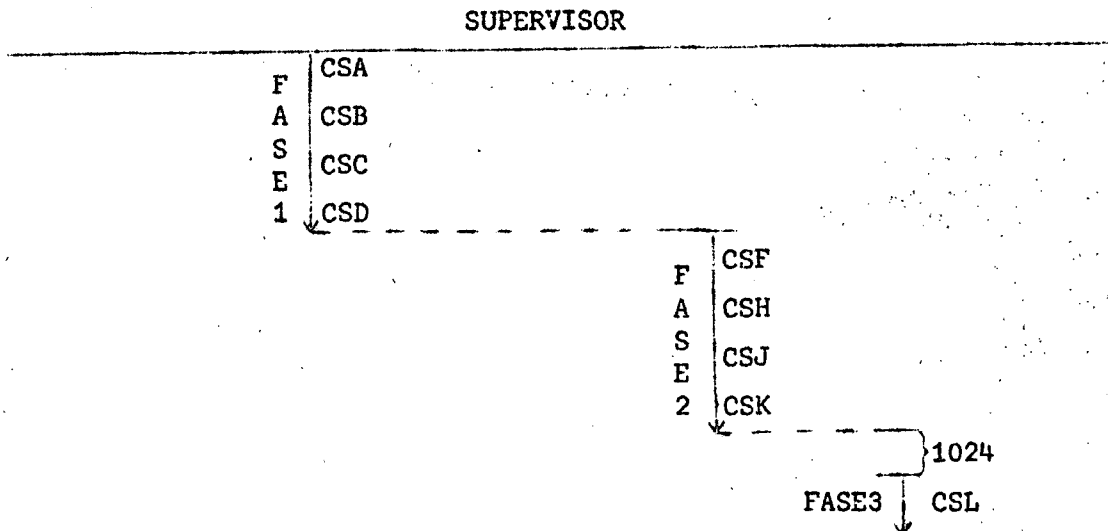
```

ESD
TXT CSJ
TXT CSK
RLD
END
PHASE FASE3,*+1024
INCLUDE MØD3
ENTRY
    
```

El programa linkage editor genera las fases en forma temporal en la CIL. La estructura de las fases es la siguiente:

| FASE1 | FASE2 | FASE3 |
|-------|-------|-------|
| CSA   | CSF   | CSL   |
| CSB   | CSH   |       |
| CSC   | CSJ   |       |
| CSD   | CSK   |       |

Al ser cargadas en memoria formarán la estructura que se indica a continuación:



- 5) Se tiene un programa en fortran y un programa en cobol. Se pide:
- compilar el programa en fortran y catalogarlo en la RL con el nombre de MØDFØR
  - catalogar el programa en cobol en la sub-biblioteca C de la SSL con el nombre de LIBCOB

- cambiarle el nombre al módulo ALFA por el de BETA
- cambiarle el nombre al libro CINCO de la sub-biblioteca A por el de FIVE

- listar los directorios de las bibliotecas CIL,RL y SSL

```
// JØB EJ5
// ØPTION DECK
// EXEC FFØRTRAN
    }programa en fortran
/*
* PØNER DECK DE FØRTRAN
* EN LA UNIDAD X'00A'
* EN LA SIGUIENTE FØRMA:
* CATALR MØDFØR
* DECK
* /*
// PAUSE LEER CØMENTARIØ
// ASSGN SYSIPT,X'00A'
// EXEC MAINT
// RESET SYS
// EXEC MAINT
    CATALS C.LIBCØB
    }programa en cobol
/*
// EXEC MAINT
    RENAMR ALFA,BETA
/*
// EXEC MAINT
    RENAMS A.CINCØ,A.FIVE
/*
// EXEC DSERV
    DSPLY CD,RD,SD
/*
/ε
```

#### 4. Tipos de macro instrucciones en DOS/VS

Una macro instrucción es un conjunto de instrucciones que se identifica con un nombre, de tal manera que, si se desea realizar el proceso que efectúa dicho conjunto basta con referirse a él por su nombre proporcionándole al mismo tiempo los parámetros que sean necesarios.

El sistema de macro instrucciones de DOS/VS está compuesto de dos partes básicas:

- Definiciones de macro instrucciones (en lo sucesivo macros) que son rutinas generales escritas en lenguaje simbólico y almacenadas en la sub-biblioteca de macros de la SSL. El conjunto de proposiciones define el nombre, formato y condiciones para generar instrucciones en lenguaje de ensamble.
- Macros del programa fuente que son las que el programador especifica en su programa para indicar al ensamblador cuáles son las definiciones de macros que se llaman desde la SSL.

Se tienen los siguientes tipos de macros: macros de comunicación con el Supervisor, comunican con éste y proporcionan acceso al área de comunicación; macros declarativas DTFxx, del sistema de control de entrada/salida, que definen las características del archivo que va a ser procesado (usadas con Logical Input Output Control System - LIOCS y Physical Input Output Control System - PIOCS); macros declarativas de generación de módulos lógicos, (usadas con LIOCS) proporcionan información acerca del tipo de módulo que va a ser generado, éste se define como una rutina en lenguaje de máquina, la cual maneja las condiciones especificadas en la macro de generación del módulo; macros imperativas, de control de entrada/salida, que indican lo que la operación de entrada/salida desea; macros declarativas del método de acceso de almacenamiento virtual (VSAM).

Durante el proceso de compilación, la macro especifica cuál definición debe ser extraída de la SSL para ser insertada en el programa. La inserción, que puede ser un módulo, una tabla o una rutina, recibe el nombre de expansión de la macro. El programa queda constituido así por proposiciones fuente y proposiciones en lenguaje de ensamble generadas desde la definición de las macros. Posteriormente el programa es traducido a lenguaje de máquina.



A. Macros declarativas DTF (Define the file) y de generación de módulos

Si se hace uso de LIOCS, el archivo debe ser descrito con una macro DTF. Con ella se indican las características y el tipo de procesamiento del archivo, y además se especifican las áreas y rutinas de memoria virtual que se van a usar en el proceso del archivo, como también los dispositivos que se emplean para almacenar en memoria o recuperar desde ella.

Macros utilizadas en procesamiento secuencial

DTFSR dispositivo de tipo serial (sólo para compatibilidad con Basic Operating y Basic Program System)  
DTFCD dispositivo de tarjetas  
DTFMT dispositivo de cinta magnética  
DTFPR impresora  
DTFCN impresora de teclado de la consola y consola de despliegue del operador  
DTFSD dispositivo DASD secuencial  
DTFPT lectora-perforadora de cinta de papel  
DTFOR lectora óptica de caracteres (excepto la 3886)  
DTFDR lectora 3886  
DTFMR lectora de caracteres magnéticos y lectora/clasificadora óptica  
DTFDI dispositivo independiente  
DTFDU unidad de entrada/salida diskette 3540

Macro utilizada en acceso directo

DTFDA dispositivo DASD random

Macro utilizada en sistema secuencial con índices

DTFIS dispositivo DASD con sistema de administración de archivo secuencial con índice

Macro utilizada con PIOCS

DTFPH para verificación o grabación de rótulos estándar en DASD o cinta magnética, o si el archivo DASD está protegido.

A continuación se describen las macros DTFCD, DTFCN, DTFMT y DTFPR con las macros de generación de módulos lógicos que tienen asociadas.

Los parámetros de las macros DTF y de generación de módulos pueden ser perforados en tarjetas con el mismo formato utilizado en las proposiciones del lenguaje de ensamble y pueden ser compiladas en cualquier orden.

En la primera tarjeta (de encabezamiento), en el campo de nombre se indica el nombre simbólico del archivo, que puede tener hasta siete caracteres, evitando usar IJ en los dos primeros pues se puede crear conflicto con los símbolos de IOCS. Evitar nombres que difieran de otro sólo en un último carácter adicional, pues IOCS genera nombres similares. A continuación del nombre, separado por un blanco al menos, está el código de operación (DTFxx) y después de él separado por un blanco al menos, los operandos. Si se desea, o si es necesario tener tarjetas de continuación (de detalle) se especifica carácter de continuación en la columna 72 y se empieza a partir de la columna 16 en las tarjetas de detalle.

Los operandos están constituidos por palabras claves seguida cada una de signo igual y después de éste el parámetro respectivo. Como son palabras claves se pueden especificar en cualquier orden como asimismo ser omitidos.

1) Macro DTFCD. Se utiliza para describir un archivo en tarjetas, sin embargo, si el programa es llamado por un procedimiento catalogado y lee desde SYSIPT se debe especificar la macro DTFDI.

#### Operandos de DTFCD

ASOCFLE = nombre del archivo

Se utiliza con el operando FUNC para definir archivos asociados en la 2560, 3525 o 5425. Un archivo asociado se tiene, cuando un archivo para una función se usa asociado con otro archivo para otra función, las que se realizarán en el mismo conjunto de tarjetas.

Nombre de archivo es el identificador del archivo asociado, de impresión, perforación o lectura.

Por ejemplo, si FUNC = PW, en el operando ASOCFLE de la DTFCD se especifica el nombre de la DTFPR y en ésta el de la DTFCD.

BLKSIZE = n

Se especifica la longitud del área de entrada/salida (IOAREA1). Si el formato del registro es variable o indefinido debe indicarse la longitud del registro más grande.

Si el operando se omite se supone longitud 80 bytes con las excepciones siguientes:

160 para modo columna binaria en la 2560, 3505 ó 3525.

96 para la 2536 ó 5425

900 para la 3881

CONTROL = YES

Se especifica si se va a utilizar la macro CNTRL en el proceso del archivo. En este caso debe omitirse CTLCHR.

CRDERR = RETRY

Se aplica para salida de tarjetas en la 2520 ó 2540. El sistema envía un mensaje al operador y entra en estado de espera. El operador puede cancelar el job, ignorar el error o pedirle al sistema que reperfore la tarjeta.

Si el operando no se especifica, el error se ignora.

CTLCHR = {ASA | YES}

Se especifica si se va a utilizar el primer carácter como carácter de control. ASA significa American National Standards Institute Inc. y YES corresponde a conjunto de caracteres del Sistema/370.

El operando CONTROL se debe omitir.

DEVADDR = {SYSIPT | SYSPCH | SYSRDR | SYSnnn}

Se especifica el nombre simbólico del dispositivo. El mismo nombre debe aparecer en la proposición ASSGN para asignar la dirección del dispositivo actual de entrada/salida al archivo.

DEVICE = {2540 | 1442 | 2501 | 2520 | 2560P | 2560S | 2596 |  
3504 | 3505 | 3525 | 5425P | 5425S | 3881}

Se especifica el dispositivo de entrada/salida asociado con el archivo. Los parámetros que tienen P ó S son dispositivos que tienen depósitos de entrada primario y secundario.

EOFADDR = nombre

Se especifica el nombre de la rutina a la cual se le entrega el control cuando el sistema detecta fin de archivo.

ERROPT = {IGNORE | SKIP | nombre}

Se especifica la acción a seguir al detectarse un error en un archivo de entrada o salida en una 2560, 3504, 3505, 3525 ó 5425. Las tres opciones se pueden especificar para archivos de entrada, en cambio, sólo IGNORE se puede indicar para archivos de salida.

IGNORE indica que el error debe ser ignorado

SKIP indica que el registro con error se debe saltar

nombre es una rutina que tratará el error y a la cual se le entrega el control al detectar uno.

FUNC = {R | P | I | RP | RW | RPW | PW}

Se especifica el tipo de archivo que va a ser procesado en una 2560, 3525 ó 5425, R indica lectura, P perforación y W impresión. Si se indica I significa que el archivo será perforado y además interpretado.

Los parámetros restantes se utilizan en conjunto con el operando ASOCFLE para especificar archivos asociados.

IOAREA1 = nombre

Se especifica el nombre del área de entrada o salida utilizada para el archivo.

IOAREA2 = nombre

Se especifica el nombre de una segunda área de entrada o salida.

IOREG = (r)

Si se utilizan dos áreas de entrada o salida y no se especifica área de trabajo, se debe indicar este operando, el parámetro corresponde a un RUG 2-12.

MODE = {E | C | O | R | EO | ER | CO | CR}

El operando indica el modo utilizado para procesar un archivo de entrada o salida en una 2560, 3504, 3505 ó 3525. E corresponde a EBCDIC, C columna binaria, O marca óptica, R eliminación de columna leída. En la 3504 y 3505 se pueden especificar combinaciones de los parámetros.

MODNAME = nombre

Se especifica el nombre del módulo lógico que será usado con la DTF para procesar el archivo. Si el módulo lógico es ensamblado junto con el programa el nombre debe ser el mismo de la macro CDMOD. Si el operando se omite el sistema genera un nombre estándar para llamar el módulo.

OUBLKSZ = n

Se especifica el número máximo de caracteres que se transferirán cada vez. El operando se utiliza en conjunto con IOAREA2 pero sólo para archivos combinados. Se define como tal un archivo en la 1442, 2520 ó 2540 en el cual se lee y perfora el mismo conjunto de tarjetas.

Si no se especifica IOAREA2 se supone la longitud indicada en BLKSIZE.  
RDONLY = YES

Se especifica si la DTF es usada con un módulo de lectura solamente. El RUG 13 debe tener la dirección de un área de 72 bytes con alineamiento de doble palabra.

RECFORM = { FIXUNB | VARUNB | UNDEF }

Se especifica el formato del registro del archivo: fijo desbloqueado, variable desbloqueado o indefinido.

RECSIZE = (r)

Si se tienen registros indefinidos, este operando especifica un RUG 2-12 que contiene la longitud del registro de salida, esto es, debe cargarse el RUG antes de cada llamada de la macro PUT en el programa.

SEPASMB = YES

Se especifica si la DTF va a ser ensamblada aparte. En este caso, una tarjeta CATALR con el nombre del archivo se perforará encabezando el módulo, además, se define el nombre del archivo como un punto de entrada en el ensamblado.

SSELECT = n

Se especifica el carácter seleccionador de depósito que es válido para el archivo. Si no se indica se supone NR (normal read) o NP (normal punch).

TYPEFLE = { INPUT | OUTPUT | CMBND }

Se especifica el tipo de archivo: entrada, salida o combinado.

WORKA = YES

Se especifica si los registros de entrada o salida van a ser procesados en un área de trabajo.

2) Macro CDMOD

Operandos de CDMOD

CONTROL = YES

Se debe incluir si se utiliza la macro CNTRL. El módulo generado también procesa archivos para los cuales no se hace uso de la macro CNTRL.

CRDERR = RETRY

Se especifica si se incluyó en la DTF.

CTLCHR = {ASA | YES}

Se especifica con el mismo parámetro que en la DTF

DEVICE = {2540 | 1442 | 2501 | 2520 | 2560P | 2560S | 2596 |  
3504 | 3505 | 3525 | 5425P | 5425S | 3881}

Se especifica con el mismo parámetro que en la DTF

FUNC = {R | P | I | RP | RW | RPW | PW}

Se especifica con el mismo parámetro que en la DTF

IOAREA2 = YES

Se especifica si en la DTF se indicó IOAREA2

RDONLY = YES.

Se especifica si se incluyó en la DTF

RECFORM = {FIXUNB | VARUNB | UNDEF}

Se especifica con el mismo parámetro que en la DTF

SEPASMB = YES

Se especifica si el módulo va a ser ensamblado aparte. Se perfora el módulo con una tarjeta CATALR con el nombre del módulo al comienzo y además, el nombre como punto de entrada en el ensamblado

TYPEFLE = {INPUT | OUTPUT | CMBND}

Se especifica con el mismo parámetro que en la DTF

WORKA = YES

Se especifica si se incluyó en la DTF

3) Macro DTFCN. Se utiliza para describir un archivo de entrada o salida que es procesado en una impresora de teclado de la consola, 3210 ó 3215, o una consola de despliegue del operador.

Operandos de DTFCN

BLKSIZE = n

Se especifica la longitud del área de entrada/salida

DEVADDR = {SYSLOG | SYSnnn}

Se especifica el nombre simbólico del dispositivo

INPSIZE = n

Se especifica la longitud de la parte de entrada del área de entrada/salida, para uso de la macro PUTR

IOAREA1 = nombre

Se especifica el nombre del área de entrada/salida utilizada para el archivo. Para el uso de la macro PUTR la primera parte del área es para salida y la segunda para entrada. Las longitudes de ambas indican el BLKSIZE e INPSIZE respectivamente

MODNAME = nombre

Se especifica el nombre del módulo lógico que será generado por la DTFCN. Si el operando se omite el sistema genera un nombre estándar

RECFORM = {FIXUNB | UNDEF}

Se especifica el formato del registro del archivo: fijo desbloqueado o indefinido.

RECSIZE = (r)

Si se tienen registros indefinidos, este operando especifica un RUG 2-12 que contiene la longitud del registro.

TYPEFLE = {INPUT | OUTPUT | CMBND}

Se especifica el tipo de archivo: entrada, salida o combinado

WORKA = YES

Se especifica si los registros de entrada o salida van a ser procesados en un área de trabajo.

#### 4) Macro DTFPR

Operandos de DTFPR

ASOCFLE = nombre de archivo

Véase igual operando de la DTFCN.

BLKSIZE = n

Se especifica la longitud del área de entrada/salida (IOAREA1). Las longitudes máximas y supuestas para cada dispositivo son:

| Dispositivo | longitud máxima | longitud supuesta |
|-------------|-----------------|-------------------|
| 1403        | 132             | 121               |
| 1443        | 144             | 121               |
| 2245        | 800             | 121               |
| 2560        | 384             | 64                |
| 3203        | 132             | 121               |
| 3211        | 150             | 121               |
| 3525        | 64              | 64                |
| 5203        | 132             | 96                |
| 5425        | 128             | 96                |

CONTROL = YES

Véase igual operando de la DTFCD. No debe usarse con la 2560 ó 5425

CTLCHR = {YES | ASA}

Véase igual operando de la DTFCD. No debe usarse con la 2560 ó 5425. Si se indica ASA en la 3525 no está permitido el carácter +.

DEVADDR = {SYSLOG | SYSLST | SYSmmn}

Se especifica el nombre simbólico del dispositivo. Los dos primeros no deben indicarse para la 2245, 2560, 3525 ó 5425.

DEVICE = {1403 | 1443 | 2245 | 2560P | 2560S | 3203  
3211 | 3525 | 5203 | 5425P | 5425S}

Se especifica el dispositivo asociado con el archivo

ERROPT = {RETRY | IGNORE | nombre}

Se especifica la acción a seguir al detectarse un error de máquina.

RETRY sólo para la 3211, indica que el comando es recuperado una vez.

Si esto no tiene efecto, se entrega un mensaje y se cancela el job.

IGNORE sólo para la 3525, indica que el error será ignorado.

nombre sólo para la 3211, es una rutina que tratará el error y a la cual se le entrega el control al detectar uno.

FUNC = {W [T] | RW [T] | RPW [T] | PW [T]}

Véase igual operando de la DTFCD. El parámetro T se especifica sólo en la 3525 e indica una impresora opcional de dos líneas.



IOAREA1 = nombre

Se especifica el nombre del área de salida.

IOAREA2 = nombre

Se especifica el nombre de una segunda área de salida.

IOREG = (r)

Véase igual operando de la DTFCD.

MODNAME = nombre

Véase igual operando de la DTFCD.

PRINTOV = YES

Se especifica si se va a utilizar la macro PRTOV en el proceso del archivo.

El operando no puede indicarse con la 2560 ó la 5425.

RDONLY = YES

Véase igual operando de la DTFCD.

RECFORM = {FIXUNB | UNDEF | VARUNB}

Véase igual operando de la DTFCD.

RECSIZE = (r)

Véase igual operando de la DTFCD.

SEPASMB = YES

Véase igual operando de la DTFCD.

STLIST = YES

Se especifica si se va a utilizar la característica de listado de cinta selectiva en la 1403. El operando RECFORM debe tener el parámetro FIXUNB

UCS = {OFF | ON}

Se indica una acción a tomar si se detecta error de datos en una 1403, 5203, 3203 ó 3211. ON significa que los errores de datos serán procesados con un mensaje al operador, OFF significa que los errores serán ignorados y será dejado en blanco el lugar del carácter no imprimible.

WORKA = YES

Véase igual operando de la DTFCD.

5) Macro PRMOD

Operandos de PRMOD

CONTROL = YES

Véase igual operando de la CDMOD

CTLCHR = {YES | ASA}

Se especifica con el mismo parámetro que en la DTF

DEVICE = {1403 | 1443 | 2245 | 2560P | 2560S | 3203 | 3211 | 3525 | 5203 | 5425P | 5425S}

Se especifica con el mismo parámetro que en la DTF.

ERROPT = YES

Se especifica si ERROPT = nombre en la DTF, se omite en cualquier otro caso.

FUNC = {W [T] | RW [T] | RPW [T] | PW [T]}

Se especifica con el mismo parámetro que en la DTF

IOAREA2 = YES

Se especifica si en la DTF se indicó IOAREA2

PRINTOV = YES

Se especifica si la macro PRTOV va a ser utilizada en el proceso del archivo.

RONLY = YES

Se especifica si se incluyó en la DTF.

RECFORM = {FIXUNB | VARUNB | UNDEF}

Se especifica con el mismo parámetro que en la DTF

SEPASMB = YES

Véase igual operando de la CDMOD

STLIST = YES

Se especifica si se incluyó en la DTF

WORKA = YES

Se especifica si se incluyó en la DTF

6) Macro DTFMT. Se utiliza para cada archivo de entrada o salida, en cinta magnética en código EBCDIC o ASCII, que va a ser procesado.

Operandos de DTFMT:

ASCII = YES

Se especifica si la cinta está en código ASCII. El operando no se permite para archivos de trabajo.

BLKSIZE = n

Se especifica la longitud del área de entrada/salida. Si el formato del registro es variable o indefinido, debe indicarse la longitud del registro más grande.

El tamaño máximo del bloque es 32767 bytes, el mínimo es de 12 bytes. En el caso de registros de salida variables, el tamaño mínimo es de 18 bytes

BUFOFF = { 0 | n }

Se especifica si ASCII = YES. En él se indica la longitud de un campo de 0-99 bytes que precede cada registro y que puede contener datos o la longitud física del registro en el caso de longitud variable.

CKPTREC = YES

Se especifica si la cinta de entrada tiene registros "checkpoint" entre los registros de datos. El operando se omite si ASCII = YES (registro "checkpoint" contiene la información necesaria para reiniciar un job sin tener que volver al comienzo de él).

DEVADDR = { SYSRDR | SYSIPT | SYSPCH | SYSLST | SYSnmm }

Véase igual operando de la DTFCD. Si es una cinta en código ASCII se debe especificar SYSnmm.

EOFADDR = nombre

Véase igual operando de la DTFCD.

ERREXT = YES

Se especifica para permitir que las rutinas indicadas en ERROPT o WLRERR retornen a MTMOD con la macro ERET (retorno de error).

ERROPT = { IGNORE | SKIP | nombre }

Se especifica la acción a seguir al detectarse un bloque erróneo (error de paridad). El significado de los parámetros es similar al indicado en el operando de la DTFCD.

FILABL = { NO | STD | NSTD }

Se especifica si el archivo tiene rótulos estándar, no estándar o si no tiene rótulos. ASCII = YES es incompatible con NSTD.

HDRINFO = YES

Se especifica para obtener el contenido de los campos 3-10 del rótulo de encabezamiento estándar en SYSLOG.

IOAREA1 = nombre

Se especifica el nombre del área de entrada/salida. Si se procesan registros de longitud variable deben considerarse cuatro bytes para indicar el tamaño del bloque. El operando no se aplica a archivos de trabajo.

IOAREA2 = nombre

Se especifica el nombre de una segunda área de entrada/salida.

IOREG = (r)

Se especifica un RUG 2-12 cuando:

- se tienen dos áreas de entrada/salida y no se tiene área de trabajo
- se tienen registros bloqueados y se procesan en las áreas de entrada/salida
- se leen registros de longitud variable desbloqueados
- se leen hacia atrás registros indefinidos
- no se especifica BUFOFF = 0 ni WORKA = YES para cinta en código ASCII.

LABADDR = nombre

Se especifica el nombre de la rutina que procesará rótulos no estándar o rótulos estándar del usuario.

LENCHK = YES

Se especifica si la longitud del bloque en una cinta de entrada, en código ASCII, será comparada con la longitud del registro físico. Debe haberse indicado BUFOFF = 4 y RECFORM = VARUNB o VARBLK

MODNAME = nombre

Véase igual operando de la DTFCD

NOTEPNT = {POINTS | YES}

Se aplica sólo a archivos de trabajo. YES indica que las macros NOTE, POINTW, POINTR o POINTS van a ser utilizadas. POINTS indica que sólo esa macro será especificada.

RDONLY = YES

Véase igual operando de la DTFCD

READ = { FORWARD | BACK }

Especifica la dirección en la cual es leída la cinta.

RECFORM = { FIXUNB | FIXBLK | VARUNB | VARBLK | SPNBLK | SPNUNB | UNDEF }

Se especifica la organización de los datos en la cinta

FIXUNB registros desbloqueados de longitud fija  
 FIXBLK registros bloqueados de longitud fija  
 VARUNB registros desbloqueados de longitud variable  
 VARBLK registros bloqueados de longitud variable  
 SPNBLK registros bloqueados de longitud variable  
 "spanned" (registros que pueden ser más largos que el tamaño del  
 bloque y ocupan, por lo tanto, uno o más bloques continuos)  
 SPNUNB registros desbloqueados de longitud variable "spanned"  
 UNDEF registros de longitud indefinida

Los archivos de trabajo pueden ser FIXUNB o UNDEF

RECSIZE = { n | (r) }

En el caso de registros bloqueados de longitud fija, se especifica el número de caracteres del registro.

En registros indefinidos o "spanned" se especifica un RUG 2-12 que contiene la longitud del registro

REWIND = { UNLOAD | NORWD }

La operación normal con una macro OPEN o CIOSE o condición de fin de volumen (EOV) es rebobinar la cinta al punto de carga pero no descargarla. Con el parámetro UNLOAD se descarga y con NORWD no se rebobina.

SEPASMB = YES

Véase igual operando de la DTFCD

TPMARK = NO

En cintas de salida con rótulos no estándar, normalmente se graba una "marca de cinta" (tape mark) al comienzo. Este operando evita esa operación. En cintas sin rótulos se supone este operando.

TYPEFLE = { INPUT | OUTPUT | WORK }

Véase igual operando de la DTFCD

VARBLD = (r)

Se especifica cuándo se van a construir registros bloqueados de longitud variable en el área de salida. Se indica el RUG 2-12 que contiene la longitud del espacio que queda disponible en el área.

WLRERR = nombre

Se especifica el nombre de una rutina a la que se le entrega el control si se detecta un registro de longitud errónea en una cinta de entrada.

WORKA = YES

Véase igual operando de la DTFC

7) Macro MTMOD

Operandos de MTMOD

ASCII = YES

Se especifica si se incluyó en la DTF

CKPTREC = YES

Se especifica si se incluyó en la DTF. El módulo también procesa archivos cuyas DTF no tienen este operando.

ERREXT = YES

Se especifica si se incluyó en la DTF.

ERROPT = YES

Se especifica si se incluyó en la DTF. El módulo también procesa archivos cuyas DTF no tienen este operando.

NOTEPNT = {YES | POINTS}

Se especifica si se incluyó en la DTF. El módulo también procesa archivos cuyas DTF no tienen este operando. Si el parámetro es YES, también procesa archivos en cuyas DTF se especificó solamente POINTS.

RDONLY = YES

Se especifica si se incluyó en la DTF.

READ = {FORWARD | BACK}

Si se especifica FORWARD, el módulo maneja cintas cuya DTF tiene el mismo parámetro. Con BACK se manejan cintas cuyas DTF tienen cualquiera de los dos parámetros

RECFORM = {FIXUNB | FIXBLK | VARUNB | VARBLK | SPNBLK | SPNUNB | UNDEF}

Si el operando se omite o si se especifica cualquiera de los dos primeros parámetros, el módulo generado permite procesar registros bloqueados o desbloqueados de longitud fija. Cualquiera de los dos parámetros que siguen produce un módulo que permite procesar registros bloqueados o desbloqueados de longitud variable o "spanned". Si se especifica UNDEF, el módulo generado permite procesar registros indefinidos.

SEPASMB = YES

Véase igual operando de la CDMOD

TYPEFLE = {OUTPUT | INPUT | WORK}

Si se especifica el parámetro WORK, se genera un módulo para procesar archivos de trabajo. Cualquiera de los otros dos parámetros genera el mismo módulo lógico y formato de tabla.

WORKA = YES

Se especifica si se incluyó en la DTF. El módulo también procesa archivos cuyas DTF no tienen este operando.

#### B. Macros imperativas

Se dividen en macros de iniciación, procesamiento y términos.

1) Macro de iniciación OPEN. Deja disponible el archivo para que sea procesado.

Estructura:

$$[\text{nombre}] \text{ OPEN } \left\{ \begin{array}{c} \text{nombre de archivo 1} \\ (r1) \end{array} \right\} \left[ , \left\{ \begin{array}{c} \text{nombre de archivo 2} \\ (r2) \end{array} \right\} \dots \right]$$

donde:

nombre: es el identificador de la macro OPEN

nombre de archivo i: es el identificador de la DTF respectiva

(ri): RUG en el cual se especifica el nombre del archivo. Se recomienda usar registros 2-12. Se puede abrir un máximo de dieciséis archivos.

2) Macro de procesamiento GET. Deja el siguiente registro lógico en secuencia, disponible para que sea procesado en un área de entrada o en un área de trabajo.

Estructura:

$$[\text{nombre}] \text{ GET } \left\{ \begin{array}{c} \text{nombre de archivo} \\ (1) \end{array} \right\} \left[ , \left\{ \begin{array}{c} \text{nombre de área de trabajo} \\ (0) \end{array} \right\} \right]$$

donde:

nombre: es el identificador de la macro GET

nombre de archivo: es el identificador de la DTF. El parámetro se puede especificar en el RUG 1.

nombre de área de trabajo: es el identificador del área de trabajo utilizada en el proceso del archivo. El parámetro se puede especificar en el RUG 0.

3) Macro de procesamiento PUT. Graba, perfora o imprime registros lógicos que han sido contruidos en un área de salida o en un área de trabajo.

$$[\text{nombre}] \text{ PUT } \left\{ \begin{array}{l} \text{nombre de archivo} \\ (1) \end{array} \right\} \left[ \begin{array}{l} \text{nombre de área de trabajo} \\ (0) \end{array} \right]$$

$$\left[ \begin{array}{l} , \left\{ \begin{array}{l} \text{STLSP} = \text{campo de control} \\ (r) \end{array} \right\} \\ , \left\{ \begin{array}{l} \text{STLSK} = \text{campo de control} \\ (r) \end{array} \right\} \end{array} \right]$$

donde:

nombre, nombre de archivo y nombre de área de trabajo tienen igual significado que en la macro GET

STLSP = campo de control, se especifica un byte de control que permite el espaciado mientras se hace uso de la característica de listado de cinta selectiva en la 1403. El operando STLST = YES debe indicarse en la DTF

STLSK = campo de control, se especifica un byte de control que permite saltar, o lo que es lo mismo, avanzar el carro después de imprimir mientras se hace uso de la característica de listado de cinta selectiva en la 1403. El operando STLST = YES debe indicarse en la DTF.

Ambos operandos se pueden especificar en un RUG 2-12.

4) Macro de procesamiento CNTRL. Proporciona comandos que realizan funciones tales como: rebobinado de cinta, selección de bolsillo receptor de tarjetas, espaciado de líneas, etc.

Estructura:

$$[\text{nombre}] \text{ CNTRL } \left\{ \begin{array}{l} \text{nombre de archivo} \\ (1) \end{array} \right\} , \text{código } [n1] [n2]$$



donde:

código, n1 y n2 se deben indicar de acuerdo con la tabla siguiente:

| Unidad                                                        | códigos                                                                | n1 | n2 | comando                                                                                                                                |
|---------------------------------------------------------------|------------------------------------------------------------------------|----|----|----------------------------------------------------------------------------------------------------------------------------------------|
| Cinta magnética 3420<br>y serie 2400                          | los mismos indicados en la proposición<br>MTC (n1 y n2 no se permiten) |    |    |                                                                                                                                        |
| Lecto-perforadora de<br>tarjetas 1442 y 2520                  | SS                                                                     |    | 1  | selecciona bolsillo 1                                                                                                                  |
|                                                               |                                                                        |    | 2  | selecciona bolsillo 2                                                                                                                  |
| Lecto-perforadora 2540<br>Lectoras 3504 y<br>Perforadora 3525 | E                                                                      |    |    | salto a bolsillo 1<br>(sólo la 1442)                                                                                                   |
|                                                               |                                                                        |    | 1  | selecciona bolsillo                                                                                                                    |
|                                                               |                                                                        |    | 2  | 1, 2 ó 3                                                                                                                               |
| Máquina de tarjetas<br>multi-funcional 2560                   | SS                                                                     |    | 3  |                                                                                                                                        |
|                                                               |                                                                        |    | 1  | selecciona bolsillo                                                                                                                    |
|                                                               |                                                                        |    | 2  | 1, 2, 3, 4 ó 5                                                                                                                         |
|                                                               |                                                                        |    | 3  |                                                                                                                                        |
|                                                               |                                                                        |    | 4  |                                                                                                                                        |
| Lecto-perforadora<br>2596                                     | SS                                                                     | 1  |    | selecciona bolsillo<br>1R ó 3P                                                                                                         |
|                                                               |                                                                        |    | 2  | selecciona bolsillo<br>2R ó 4P<br>(R = lectura, P = per-<br>foración)                                                                  |
| Unidad de tarjetas<br>multi-funcional 5425                    | SS                                                                     | 1  |    | selecciona bolsillo                                                                                                                    |
|                                                               |                                                                        |    | 2  | 1, 2, 3 ó 4                                                                                                                            |
|                                                               |                                                                        |    | 3  |                                                                                                                                        |
|                                                               |                                                                        |    | 4  |                                                                                                                                        |
| Impresoras 1403, 1443,<br>3203, 3211 y 5203                   | SP                                                                     | C  | d  | salta 1, 2 ó 3 líneas                                                                                                                  |
|                                                               | SK                                                                     | C  | d  | salta a canal c y/o d<br>(c es un entero que<br>indica antes de im-<br>primir y d es un ente-<br>ro que indica después<br>de imprimir) |
| Perforadora de tarjetas<br>3525 con impresión                 |                                                                        |    |    |                                                                                                                                        |

```
FINTAR CLØSE LECTØ, IMPRE
```

```
EØJ
```

```
A DS CL80
```

```
END INICIØ
```

```
/*
```

```
// EXEC LNKEDT
```

```
// EXEC
```

```
} datos
```

```
/*
```

```
/E
```

2) Se tiene un archivo en tarjetas. Se pide leer el archivo desde la unidad lgica SYS014 y grabarlo en la cinta magntica asignada a SYS009 bloqueado, diez registros de ochenta caracteres por bloque. Utilizar dos reas de entrada y rea de trabajo.

```
// JØB PRØB2
```

```
// ØPTION LINK, LIST, XREF, NØDECK
```

```
// EXEC ASSEMBLY
```

```
START
```

```
PRINT NØGEN
```

```
INPUT DTFCD BLKSIZE=80, DEVADDR=SYS014, x
```

```
IØAREA1=T1, IØAREA2=T2, x
```

```
WØRKA=YES, EØFADDR=FINAL, x
```

```
MØDNAME=MØDCD
```

```
ØUTPUT DTFMT BLKSIZE=800, RECSIZE=80, x
```

```
RECFØRM=FIXBLK, TYPEFLE=ØUTPUT, x
```

```
WØRKA=YES, IØAREA1=C1, x
```

```
IØAREA2=C2, FILABL=NØ, x
```

```
MØDNAME=MØDMT, DEVADDR=SYS009
```

```
MØDCD CDMØD WØRKA=YES
```

```
MØDMT MTMØD WØRKA=YES
```

```
T1 DS CL80
```

```
T2 DS CL80
```

```
C1 DS 10CL80
```

```
C2 DS 10CL80
```

```
INICIO BALR 2,0
        USING *,2
        OPEN INPUT,OUTPUT
LEE     GET  INPUT,B
        PUT  OUTPUT,B
        B    LEE
FINAL   CLØSE INPUT,OUTPUT
        EØJ
B       DS   CL80
        END  INICIO
```

/\*

// EXEC LINKEDT

// ASSGN SYS014,X'001'

// ASSGN SYS009,X'182'

\* MØNTAR CARRETE M-245 EN UNIDAD X'182'

// PAUSE Y LUEGØ DAR EØB

// EXEC

} datos

/\*

/E

BIBLIOGRAFIA

- Auslander, M.A. y Jaffe, J.F., Functional structure of the IBM Virtual Storage operating system, Parte I, IBM System Journal, Vol. 12, N° 4, 1973, págs. 368-380.
- Bensoussan, A., Clingen, C.T. y Daley, R.C., "The Multics Virtual Memory : Concepts and Design", en Communications of the ACM, Vol. 15, N° 5, mayo 1972, págs. 308-318.
- Cohen, Leo J., Operating System Analysis and Design, USA, Hayden Book Company INC., 1970, 182 págs.
- Colin, A.J.T., Introduction to Operating Systems, Inglaterra Redwood Press Limited, 1971, 120 págs.
- Denning, Peter J., "Virtual Memory", en Computing Surveys, Vol. 2, N° 3, septiembre de 1970, págs. 153-139.
- Denning, Peter J., "Third Generation Computer Systems", en Computing Surveys, Vol. 3, N° 4, diciembre de 1971, págs. 175-216.
- IBM System products division, Introduction to DOS/VS, Nueva York, 1973.
- IBM System products division, DOS/VS Supervisor and I/O Macros, Nueva York, 1973.
- IBM System products division, DOS/VS System Management Guide, Nueva York, 1974.
- IBM System products division, DOS/VS System Control Statements, Nueva York, 1973.
- IBM System products division, IBM System/360 Operating System, Introduction, Nueva York, 1971.
- Katzan, Harry Jr., Advanced Programming, USA, Van Nostrand Reinhold Company, 1970, 285 págs.
- Martin, James, Design of Real Time Computer Analysis, USA, Prentice-Hall, Inc., 1967, 629 págs.
- Mealy, G.H., Witt, B.Y. y Clark, W.A., The functional structure of OS/360, IBM System Journal, Vol. 5, N° 1, 1966, págs. 2-51.
- Rosen, Saul, "Programming Systems and Languages 1965-1975", en Communications of the ACM, Vol. 15, N° 7, julio 1972, págs. 591-610.
- Rosin, Robert F., "Supervisory and Monitor Systems", en Computing Surveys, Vol. 1, marzo, 1969, págs. 37-54.
- Parmelee, R.P., Peterson, T.Y., Tillman, C.C. y Hatfield, D.J., "Virtual storage and virtual machine concepts", en IBM System Journal N° 2, 1972, págs. 99-130.





**CENTRO LATINOAMERICANO DE DEMOGRAFIA**  
**CELADE: J.M. Infante 9. Casilla 91. Teléfono 257806**  
**Santiago (Chile)**  
**CELADE: Ciudad Universitaria Rodrigo Facio**  
**Apartado Postal 5249**  
**San José (Costa Rica)**