**Resumen arquitectura**

**Clase 1**

¿Qué es la computadora? Es una maquina digital y sincrónica que tiene cierta capacidad de realizar operaciones aritméticas o lógicamente, que nos sirven para resolver problemas del mundo real.

* Sincrónica: es la interrelación entre elementos y estas son las q hacen funcionar a nuestra computadora, cada uno de estos elementos q componen la máquina, van a tener q interactuar de una manera ordenada, coordinada para cumplir el objetivo de la computadora. Esa coordinación se hace a través de un reloj, el reloj marca la evolución temporal de los procesos o cosas q va a hacer la computadora, la velocidad del reloj es un índice del funcionamiento (pero no quiere decir q sea una mejor maquina o q funcione mejor q otras en una determinada condición, por q depende del campo de aplicación).
* Digital: porque la información q vamos a guardar tiene un esquema de representación con números y esos números son fijos, ósea q los valores q vamos a representar son valores q no vamos a poder cambiar salvo q queramos cambiarlo. (esos datos van a representar un estado fijo durante un determinado estante de tiempo, ese tiempo puede ser corto de modo tal q a nosotros nos parezca q se modifica, pero la maquina sigue siendo digital). La máquina lo q hace es tomar esos números y generar otros números con ellos, ósea q la idea de la computadora es q procese valores numéricos. ¿qué es procesar? Es tomar algo q tiene determinadas características y a partir de él, generar algo q tiene características distintas porqué fue procesado.

La idea de la computadora es que pueda sumar y sería ideal q reste, para eso está la unidad Aritmética q tiene la computadora. Hablando en hardware un sumador es distinto a un restador, tengo un circuito hardware para sumar y otro para restar, ¿Qué pasa si quiero una maquina q sume y reste? Tengo q hacer los dos circuitos, como era muy caro la única solución q quedaba era platear q la representación de números en dígitos se haga con un sistema q me permita implementar o representar números negativos, las representaciones numéricas para números negativos son binario con signo, complemento a 1, complemento a 2, exceso.

(punto fijo ponen el numero con una cantidad de dígitos en donde se asume q la coma q determina parte entera y parte fraccionaria está en un determinado lugar.

Punto flotante la diferencia con el punto fijo es q el punto q separa la parte entera de la fraccionaria esta predeterminada a partir de varios componentes: matiza, la base y el exponente.

La diferencia entre punto fijo y Punto flotante es en cómo voy a poner los dígitos q representan el valor en cada uno).

Complemento a 2 es el único sistema q nos permite utilizar un circuito sumador para poder hacer restas. Entonces en la ALU voy a poder sumar y restar, pero solo haciendo uso del sumador, de esta manera en la cadena de dígitos q ponga, digo el valor y establezco el signo, además de la ventaja de tener un solo cero. (la ALU utiliza el complemento a dos para una representación de datos)

Dígitos binarios: son los q usamos en una computadora para almacenar información o para realizar operaciones con datos numéricos y si ponemos punto flotante la diferencia con el punto fijo es q el punto q separa la parte entera de la fraccionaria esta predeterminada a partir de varios componentes: matiza, la base y el exponente.

Para q la maquina digital y sincrónica pueda resolver un problema voy a tener q programar las acciones q tiene q realizar ese programa, dándole instrucciones de modo tal q la unidad de control q está en la CPU, sepa q operación quiere hacer y donde están los operandos q quiere utiliza, esta las procesa y genera un resultado (todo esto lo debe hacer con las instrucciones).

**Anexo clase 1**

¿Cómo son esas instrucciones?

Primero ¿Qué es una instrucción? las instrucciones son un elemento autocontenido, ósea q en esa representación de la instrucción, yo tengo q tener toda la información necesaria para q la UC pueda efectuar esa tarea. En la instrucción voy a tener q operación quiero hacer, con q operando voy a realizar esa operación, como o donde dejo el resultado de esa operación y defino cual es la siguiente instrucción (máquina de 4 direcciones).

Von Neumann planteo la arquitectura de la computadora, establecía cinco elementos: UC, ALU, memoria, módulo de Entrada y módulo de Salida, dijo q los datos e instrucciones tienen q estar en la memoria para funcionar.

La UC y la ALU se fusionaron en la CPU y se le incorporan registros. Esa CPU se tiene q conectar con la memoria o con entrada o con salida o con E/S. Internamente CPU, memoria y E/S se interconectan a través de los buces del sistema.

Este modelo de Arquitectura se puede simplificar en un modelo basado en 3 Unidades funcionales.

La UNIDAD CENTRAL DE PROCESAMIENTO (CPU) constituida por:

* Unidad de Control (UC)
* Unidad Aritmético-Lógica (ALU)
* Registros.
* La UNIDAD DE ENTRADA/SALIDA (E/S) para introducir y extraer los datos e instrucciones.
* La UNIDAD DE MEMORIA para almacenar temporalmente datos e instrucciones.

¿Cómo hacemos las instrucciones para q la CPU las ejecute o cumpla?

(asumimos q los datos e instrucciones y resultados van a ir todas en memoria por lo q Von Neumann dijo)

Máquina de 4 direcciones tengo q poner la dirección de la próxima instrucción, es decir q puedo no tener instrucciones secuenciales, esto es un problema ya q no cumple las condiciones q puso Von Neumann a los programas, los programas no solo tienen q estar almacenado en memoria si no q también son una secuencia de instrucciones q resuelven el programa, q debe ser en forma lineal (lo cual la máquina de 4 direcciones no cumple porque le pones la dirección vos, entonces puede ir de la instrucción 1 a la 5 y luego ir a la 3 y así).

Máquina de 3 direcciones Von Neuman le agrego a la estructura de la UC un registro “Contador del programa” (pc), es un registro extra que hace al funcionamiento secuencial de las instrucciones, en ese registro voy a tener el valor de la instrucción q voy a ejecutar a continuación. El tamaño de las instrucciones (medida en binario) es menor, por lo cual el tamaño de la memoria q va a almacenar esta información es menor (barata).

Máquina de 2 direcciones, guarda el resultado en una dirección de operando, ósea sobrescribe, para esto utiliza nuevas instrucciones una llamada MOV, (debe tener el pc). Se puede realizar esto porque la lectura de los operandos se hace antes de almacenar el resultado el problema es, si necesito la dirección de algún operando después, en ese caso tendré q tomar algunos recaudos, hacer alguna copia.

El programa crece ya q necesitamos nuevas operaciones para cumplir el funcionamiento, y el hardware tiene q empezar a cambiar. Agregar nuevas operaciones obliga a q la UC sea un poco más compleja porque va a tener q “entender” una cosa más, y por ende va a costar más, entonces ahorro memoria, pero lo gasto en UC.

Maquina 1 además de contar con el registro PC, voy a tener para un operando y el resultado de la operación el registro acumulador (registro especial) dentro de la CPU, vamos a volver al mismo problema. Necesitamos otras operaciones, como la copia de memoria a un registro es una cosa y la copia de un registro a memoria es otra, esto lo identificó a través de las instrucciones LOAD y STRORE.

Al contar con un registro más y nuevas operaciones se hace más complicada la UC, lo que venía ahorrando ya no sirve. Está es la primera máquina q cumplió con la arquitectura de Von Neuman. A pesar de tener más instrucciones, este es mas corto en bits q el anterior.

Máquina de 0, usamos la pila q se desarrolla y se arma dentro de la CPU, se ingresan valores almacenados dentro de la CPU, lo cual es un problema por q esta pila define hardware (ósea q al ser de hardware no se puede modificar). Utiliza el lenguaje polaco inverso porque primero se utilizan los datos y luego se ponen las operaciones. (son programas más largos, pero ocupan menos bits).

**Clase 1**

(concepto de programa)

En un Sistema de cómputo, la secuencia de tareas está almacenada en la maquina como instrucciones.

Las instrucciones son interpretadas por la propia maquina en una unidad especial (Unidad de Control CU o Interprete de instrucciones por hardware).

Las instrucciones son convertidas en señales que comandan: la entrada de los datos, el tipo de operación, y la salida de los resultados.

La Unidad de cálculo es universal y capaz de resolver distintos tipos de operaciones. El tipo de operación a resolver lo obtiene desde el intérprete de instrucciones a partir de la instrucción leída.

programa: es una secuencia de instrucciones almacenadas en memoria en el cual, un cambio de alguna tarea requiere solo modificar la secuencia de instrucciones. A partir de los códigos de instrucción se debe hacer un “interprete de instrucción” en donde la UC se encarga de convertir (decodifica) la instrucción en señales de control requeridas por el hardware, diferentes de estas señales se necesitan para cada operación. Luego la UC obtiene la información necesaria para resolver (ejecutar) la acción definida en la instrucción.

El banco de registros de almacenamiento en la CPU estaba compuesto por:

* Registro Buffer de Memoria (MBR)
* Registro de Direcciones de Memoria (MAR)
* Registros de Instrucción y Buffer de Instrucción
* Registro Contador de Programa (Program Counter)
* Registros Acumulador y Multiplicador/Cociente
* **Modos de direccionamiento(falta)**

Debo escribir un programa de texto, en el lenguaje Assemblico correspondiente al procesador con eso mnemónicos, para poder meterlo en la maquina voy a tener q ensamblar. En el lenguaje de alto nivel, una vez q se tiene el programa se compila, acá como nuestro lenguaje de alto nivel es de los más bajo que hay, se ensambla porque sabe q trabaja con binarios exclusivamente. Ese ensamblado nos va a dejar un archivo convertido a binario, pero q no puede ejecutarse todavía porque hay q darle todo el entorno de funcionamiento a un programa en una computadora q ya está hecha, entonces se hace el lincado o enlazado (en un lenguaje de alto nivel cuando compilamos un programa también se hace el lincado automáticamente) y el resultado nos da un programa ejecutable. Luego lo debemos cargar en memoria y una vez q se haga eso tenemos q hacer q el PC tenga la dirección donde comienza el primer byte de la primera instrucción a ejecutar. (la memoria de este simulador es memoria q guarda de a bytes).

SP (stack pointer) puntero de pila: nos da la dirección de la cabeza de la pila, es una dirección de memoria.

IR (registro de instrucción): contiene la dirección q hay q ejecutar.

Ciclo de instrucción: es la secuencia de pasos q va a tener q realizar la UC para cumplir con una instrucción, la instrucción tiene q ser como el formato q vimos, según la maquina q las construya tenemos distintas cosas q se marcan en el ciclo de instrucción.

(sea el formato de instrucción q sea, debe cumplir con este ciclo de instrucción).

Secuencia de pasos:

* Captar la instrucción
* Decodificar la instrucción para saber q hacer

Captar instrucción:

* La dirección de la instrucción que se debe captar se carga en el registro Contador de Programa (PC).
* La UC envía al bus de direcciones el valor almacenado en el PC.
* La UC lee la instrucción desde la Memoria, y la almacena en un registro temporal denominado Registro de Instrucción (IR).
* El PC se incrementa en 1 (se prepara para apuntar a la próxima instrucción).
* La UC decodifica (“interpreta”) la instrucción captada.

Cuando decodifico y me di cuenta q operaciones y cuáles son las direcciones de los operandos, procedo a buscarla en memoria (una vez q le digo donde están, los voy a buscar) esto se realiza tantas veces como operandos necesito.

Ahora la UC esta en condiciones para realizar la operación con los datos, q realmente los procese, dejándome un resultado en la ALU.

Ese resultado lo tendré q guardar en la dirección en donde diga la instrucción, una vez q se la dirección lo guardo, cuando termine de guardar el dato se terminó la instrucción, entonces busco la siguiente instrucción en el PC. Vuelve a empezar

Ruta de datos\*

Multiplexores: son dispositivos, circuitos lógicos, combinatorios q permite de varias entradas q una sola salga como salida.

**Subrutina:**

Para poder implementar las funciones o procedimientos vamos a necesitar un par de instrucciones nuevas, una para invocar y otra para retornar.

Puede invocarse una subrutina desde cualquier punto del programa, mediante la instrucción Call, a continuación, se pone la etiqueta de la subrutina especifica (operando), con esto lo q logramos es q la siguiente instrucción no sea la q viene a continuación en el cuerpo de programa, por ejemplo, si no la dirección de la primera instrucción de la subrutina q llamamos.

La dirección CALL tiene un único operando q va a ser la dirección destino del salto, es decir tendría q ser la dirección q el contador del programa va a tener q adquirir para buscar la siguiente instrucción y la CPU comienza a ejecutar las instrucciones de la subrutina. Ese nombre se puede determinar con la lista de nombres.

A ese procedimiento o función nosotros le tenemos q pasar argumentos, tienen dos tipos

Por valor: es una copia del valor q queremos intercambiar.

Por referencia: es la dirección en donde el valor este.

Por definición de Von Neumann los datos deben estar en la memoria, y con el avance tecnológico pusimos registros dentro de la CPU que se pueden convertir en contenedores de estos datos q queremos intercambiar. Entonces podemos utilizar tanto memoria como registro para intercambiar, pero debemos tener cosas en cuenta:

Registros son rápida, pero tenemos una cantidad limitada de registros físico en la máquina, por lo tanto, nos da una limitación a la cantidad de argumentos q le quiero pasar a la subrutina.

Para eliminar el problema de la limitación usamos la memoria, pero la memoria es difícil de estandarizar.

Se hizo un estándar que consiste en pasar el argumento por medio de una pila o stack, es el método más utilizado para mandar argumentos entre subrutina. Se la llama también “pasaje de parámetros”, si no utiliza la pila se le llama pasaje de argumentos.

¿que es una pila?

**La pila**: es una zona de memoria q almacena datos en forma transitoria por demanda a través de un puntero (el stack pointer, SP) q indica la unidad de acceso de la misma. Es una zona de memoria q se modifica o crece por demanda, se puede realizar dos operaciones básicas PUSH operación de apilar y POP operación de desapilar, siempre van a estar basados en el puntero de la pila, son operaciones inversas. Utilizan el registro SP como indicador de dónde va el valor o de donde lo saco, luego se pone un operando, un registro q va a tener el valor a guardar o el q va a sacar el elemento del tope de la pila y la va a poner en ese registro.

**SP:** registro Puntero de Pila, apunta al último lugar de la pila q fue utilizado, (contiene la dirección de la cabeza de la pila).

(Si pongo un pop sin un Push antes, no va a tirar un error porque es una instrucción valida, sino q va a sacar de la pila un dato no valido).

Para volver de la subrutina vamos a tener q generar una nueva instrucción, RET este lo q hace es volver a la siguiente instrucción después del Call, en ese momento el PC se carga con la dirección de la instrucción siguiente al CALL en el programa principal, ubicada en el tope de la pila. De esa manera el programa principal continua con su tarea en el punto siguiente a la invocación de la subrutina.

Cuando hago un Call a un rotulo antes de ir, la UC guarda en la pila el pc (tiene la instrucción q le sigue al Call) en la pila, si yo agrego datos a la pila voy a tener q desapilarlo teniendo en cuenta q tengo este dato si no, lo voy a perder y no voy a poder volver.

El uso de la pila de la UC consta de dos acciones q tiene q controlar la UC, por un lado, va a tener q analizar un movimiento de datos entre el registro y una dirección de memoria o la inversa. (la dirección de memoria es donde se desarrolla la pila)

La pila va a crecer dependiendo de quien arma la unidad de control, él Push y el pop por definición trabajan con 2 byte ósea con todo el registro.

En la familia Intel tengo que mover el dato y modificar el SP, entonces en un Push lo primero q se hace es modificar el SP, se decrementa el valor y como segunda acción, en esa nueva dirección q apunta el puntero de pila, se guarda el valor q puede ser de 1 o 2 bytes, así apilo. En la 8000h no hay memoria, para desapilar aplico la instrucción pop, la cual primero lee la dirección q apunta el SP (que apunta a la última usada), copio lo q está en memoria y lo pongo en el registro del pop y el puntero de modifica para mostrar q el dato ya se sacó.

BP: es un puntero de referencia y sirve para acceder a los datos locales a la pila y parámetros, (es el puntero por el cual te vas a poder mover en la pila).

**Clase 2**

En condiciones “normales”, la CPU lee y ejecuta una instrucción a continuación de la otra de manera ininterrumpida (el “bucle interno del ciclo de instrucción”).

¿Qué es una interrupción?

Una interrupción es un mecanismo que permite alterar ese proceso de “ejecución normal” de la CPU. Este mecanismo permite que la CPU suspenda la tarea que está haciendo y responda a una solicitud de atención para resolver (ejecutar) otra tarea (servicio de la interrupción). Una vez completado el servicio de la interrupción, el procesador retoma la tarea suspendida, en el punto donde se detuvo (de una manera similar al llamado a subrutina). Por lo tanto se requiere de 3 acciones:

1) Detener (suspender) la tarea que está ejecutando el procesador (suspender, no terminar ni abortar).

2) Bifurcar (saltar) a otra tarea, asociada a la solicitud de interrupción (comúnmente conocida como servicio de interrupción).

3) Restablecer la tarea suspendida en las condiciones en las que se encontraba en el momento en el que se la detuvo.

(se solicitan las interrupciones a la maquina y la CPU ve si acata o no el pedido, si la acata las acciones q haga el gestor deben ser lo más eficientes posibles y luego vuelvan al programa principal)

(el objetivo de una interrupción es q la maquina deje de hacer algo para q el usuario tome el control)

Porque podemos llegar a pedir una interrupción? Dependiendo del ámbito de aplicación puede ser:

* Por resultado de una ejecución de una instrucción.
* Por un temporizador interno del procesador.
* Por una operación de E/S.
* Por un fallo de hardware.

**¿Qué hacer si interrumpe?** Primero debemos salvar el estado del procesador (se guarda en la pila), luego se cambia el control de la maquina y dejamos el programa interrumpido para empezar a ejecutar las instrucciones de lo que llamamos gestor.

El gestor es un programa q va a responder a un caso particular de interrupción q a veces se puede generalizar, restaura el estado original del procesador y retorna a la ejecución normal del programa interrumpido.

* El estado del procesador: salvamos el estado del contador del programa para volver y un registro de estado q refleja las banderas y otras cosas q tienen los procesadores. (conforma al estado del registro).

La función de un gestor es corregir la causa q ocasiona el pedido de interrupción, esta acción tiene q reponer o guardar todo lo q valla a utilizar en esta respuesta rápida.

La UC guarda el estado antes de darle el control al gestor, para salir del gestor y volver al programa principal, se saca de la pila la dirección y la pone en el PC, para poder ir a la siguiente instrucción luego del CALL, en el programa principal.

La UC (q es la q maneja a toda la computadora) debe ser capaz de resolver el problema original por el cual interrumpimos y después devolver el control a lo q se estaba haciendo antes de interrumpir ¿Cómo lo vamos a hacer? Esto se va a hacer a través de forzar a q la UC ejecute un programa especial para sacarla de esa situación incontrolada, lo llamamos gestor. Por un lado, va a ver una solicitud de interrupción a la maquina y cuando la maquina deje q se interrumpa, se va a generar el disparo de un programa q va a intentar solucionar el problema, si no puede resolver el problema debe minimizarlo lo más rápido q se pueda ante esta condición.

Una vez q se pasó el control al gestor, este corrige o responde al motivo de la interrupción, se toma la molestia de si va a utilizar hardware especifico (registro) para hacer cuentas, por ejemplo, usa el recurso y luego restaura los valores originales para poder dar la orden y retornar al programa q nos pidió la interrupción.

(Para cada interrupción debemos poner un gestor).

Si tenemos múltiples fuentes q solicitan interrupción se establece cuales son importantes. Se considera:

* No enmascarables: estas son las q NO se pueden ignorar, indican eventos peligrosos o de alta prioridad, requieren respuestas eficientes y rápidas. Las voy a atender siempre q se pida interrumpir, habilitando el gestor (eje: la temperatura del horno)
* Enmascarables: pueden ser ignoradas, sus eventos no configuran peligro para el configuramiento del sistema o pueden esperar, se hacen unas instrucciones específicas para q la UC se entere de q no tenemos q atender determinada solicitud de interrupción, se hace con instrucciones especiales si no es maquina porque las interrupciones normalmente van a ser señales eléctricas, la única manera q no se hagan es no estar conectado.

**¿Cuál es la fuente real de una solicitud de interrupción?**

Aparecieron las verdaderas interrupciones q son las q provenían de dispositivos de E/S y se las llamaban INTERRUPCIONES POR HARDWARE. Nuestro sistema, ósea nuestra CPU o UC, debe ser capaz de manejar o administrar están solicitudes q son no planeadas o asincrónicas (las solicitudes de interrupción son asincrónicas, ósea no tiene nada q ver con el funcionamiento de la computadora). No están relacionadas con el proceso en ejecución en ese momento. Entonces la señal de interrupción es una señal eléctrica, q de alguna manera llega a la CPU.

Aparecieron algunas interrupciones por hardware, pero no son externas a la máquina sino internas, a estas se las llama Traps o excepciones. Estas son dadas la complejidad de cosas q tenemos dentro de la CPU, pero requieren la misma respuesta de un gestor para solucionar el problema. Por ejemplo:

* Condiciones excepcionales: overflow en ALU de puntos flotante.
* Falla de programa: tratar de ejecutar una instrucción no definida.
* Falla de hardware: error de paridad de memoria.
* Accesos no alineados o a zonas de memoria protegidos.
* Alineamiento: cada procesador procesa un tamaño de palabras q necesitamos. Esa palabra la tenemos q llevar de una vez. Es el acceso a memoria con el tamaño de la palabra q nosotros necesitamos, q no siempre se da. Algunos procesadores no admiten q el acceso a la memoria no este alineados, algunos te dejan, pero te dan una advertencia.

**Interrupción por software:** son nuevos pedidos de interrupción q se le hacen a la UC, pero no por las tradicionales interrupciones. Estás no las administramos ¿Qué son? Son interrupciones provocadas por una instrucción q generalmente el resultado genera los mismos problemas q una interrupción por hardware. No todos los procesadores tienen estas interrupciones, tenemos q agregar más cosas al conjunto de instrucciones.

Son instrucciones explicitas q afectan a la UC de la misma manera q las interrupciones por hardware. Con esto vamos a poder probar si funcionan los gestores de interrupción, estas también se pueden usar para hacer llamados a funciones del sistema operativo (SO). Se podría entonces, utilizar las interrupciones por software para poder invocar y llamar a esas rutinas q serian reutilizar funciones q tiene el SO. (solo pueden existir en procesadores q tiene instrucciones específicas para hacer esto).

**¿Qué pasa si no tengo interrupciones por software y no tuviera sistema operativo?**

Se va a tener q escribir todo los que necesito en lenguaje de máquina para hacer todo lo q nos gusta hacer. Entonces en cada procesador, en cada módulo de E/S vamos a tener q escribir cual es la relación entre teclado y la CPU. Una vez q escribí todas las funciones en el programa q va a hacer uso de esas funciones, debo ir poniendo las direcciones de comienzo, como si la estuviera llamando a la subrutina (no con el esquema de interrupción), voy a usar tipo un CALL.

Nos van a servir porque nos van a permitir utilizar cosas q hizo otro sin quebrantar las leyes de copia y principalmente nos va a permitir depurarlos, saber q van a hacer.

* Sea cual sea la interrupción, debo escribir el gestor

(El q realiza las tareas es la UC, este le pasa el control al gestor de interrupciones, la CPU solo se encarga de administrar que rutina de atención voy a hacer).

(nuestra maquina siempre está realizando interrupciones, ósea realiza el famoso ciclo de instrucción)

Ciclo de instrucción:

Ahora en una máquina q va a tener la posibilidad de administrar interrupciones, va tener tres ciclos:

* Captación: capta la instrucción.
* Ejecución: ejecuta la instrucción.
* gestión de interrupción: interrumpe el ciclo de instrucción si hay un pedido de interrupción, se da en los casos q las interrupciones estén habilitadas (que no estén enmascaradas, si las tengo enmascaradas estoy en el viejo procesador).

En ese ciclo de instrucción se va a comprobar si se ha solicitado alguna interrupción q es una señal eléctrica, cuando llegan al procesador modifican un Flag (i) se pone en 1 cuando tenemos un pedido de interrupción. Si no tenemos ese pedido de interrupción, el ciclo sigue y busca la siguiente instrucción. Entonces la instrucción puede venir en cualquier instante de tiempo, pero la UC se va a fijar si hay un pedido de interrupción una vez guardado el resultado de la instrucción q estaba haciendo, después q guardo el resultado y antes de realizar la siguiente instrucción, en ese instante se fija. Si no hay, sigue el ciclo de instrucción y busca la siguiente, si hay un pedido debo atender esa interrupción, suspendiendo lo q estaba haciendo, ósea voy a suspender la ejecución del programa en curso y voy a guardar el contexto de ese programa (el estado del procesador (registro de estado) y ósea guardo la dirección de la próxima instrucción a ejecutar) en la pila, se guarda automáticamente. El PC debe tener almacenado la dirección de la primera instrucción a ejecutar de la secuencia para poder realizar lo del gestor de interrupción, automáticamente se inhiben otras solicitudes de interrupciones cuando se está atendiendo una interrupción (cuando vuelve, se vuelve a habilitar), para q sea rápida. (habitualmente pasa esto, pero se puede llegar a ir a una interrupción y en la primera línea habilitar las interrupciones, generando una cadena).

¿Cómo opera la interrupción?

1.- La CPU recibe, mientras está ejecutando una tarea (el programa en ejecución), un pedido de interrupción.

2.- La CPU salva todo o parte del estado de la CPU correspondiente a la tarea a ser suspendida. Al menos salva el Contador de programa (PC) y el registro de estado (PSW)(es un registro q por lo menos tiene 16 bits), típicamente en la Pila del sistema . Esto lo hace porque lo necesita para restaurar la tarea suspendida.

3.- La CPU busca, en un área de memoria definida, la dirección de comienzo (el “vector”) del servicio de la interrupción, y comienza a ejecutar dicho servicio. (direccionamiento indirecto vía memoria).

4.- Cuando la CPU termina el servicio de la interrupción, tiene que retornar al programa interrumpido. Esto se hace mediante una instrucción especial de Retorno de interrupción (RTI).

5.- La ejecución de la instrucción RTI desapila exactamente lo apilado cuando atendió́ la interrupción (es decir, todo o parte del estado de la CPU). Como mínimo recupera el PC y el PSW. De esta manera retoma la tarea suspendida (el programa en ejecución) en el punto en que fue interrumpida.

En el diagrama de estado (completar):

* Calculo la dirección de la instrucción a buscar en memoria.
* Capto la instrucción
* Decodifico la instrucción para saber q operandos necesito para esa instrucción
* Capto los operandos
* Hago la operación de los datos
* Calculo la dirección del resultado
* Almaceno el resultado en la memoria
* Compruebo si hay solicitud de interrupción: Si no hay pedido pendiente (Flag inactivo) se inicia el ciclo de captación de la siguiente instrucción (proceso “normal” de ejecución). Si hay algún pedido de interrupción pendiente, el procesador guarda en la pila del Sistema, el “estado del proceso”. Existen 2 estrategias de guardado del estado del proceso:
  + Guardar solo la próxima instrucción a ejecutar y algún registro critico (por ejemplo, el registro de estado).
  + Guardar todos los registros del procesador.

El objetivo de esta operación es el de restablecer el estado del procesador al terminar el servicio de la interrupción. Obtiene la dirección donde comienza la rutina de la interrupción y carga el PC con este valor, bifurcando de esta manera, al servicio de la interrupción. Existen varias técnicas para obtener la dirección donde comienza el servicio de la interrupción. En general se dispone de un área de memoria reservada, donde están estas direcciones. Tener en cuenta que son varias direcciones porque el procesador es capaz de atender varias interrupciones, y habrá una dirección por interrupción. Está área de memoria se llama área de vectores de interrupciones.

* Interrupciones inhabilitadas: el procesador no se va a enterar q alguien quiere interrumpir por lo cual nunca va a disparar el proceso de pasarle el control al gestor. Sí alguien solicito una interrupción, al estar las interrupciones inhabilitadas lo q pasa es q el pedido queda pendiente hasta q se habilitan las interrupciones.
* Interrupciones habilitadas: Cuando viene una solicitud de interrupción, se inhabilitan (para q nadie pueda volver a interrumpir), luego se gestiona la misma (el gestor realiza todo lo q tiene q hacer) y luego se habilitan otra vez. Este mecanismo lo q permite q todas las interrupciones q se generan se atienden en orden secuencial estricto.

**Interrupciones múltiples**

Nadie me asegura q va a ser solo una interrupción la q se vaya a manejar, entonces cuando yo habilito y tengo dos interrupciones para atender, ¿Qué hago? Entonces el orden secuencial estricto solo aplica cuando tenemos una sola interrupción, cuando tengo más de una solicitud simultanea la única manera de decidir cual atiendo es asignando una prioridad (establezco prioridades para la solicitud de interrupciones):

* A cada una le doy un orden de prioridad distinto, no hay dos q tengan el mismo orden de prioridad.
* La de interrupción más alta puede interrumpir a aquel gestor de interrupción q tiene menos prioridad. Cuando se ha gestionado la interrupción de prioridad mas alta, el procesador vuelve a las interrupciones previas (de menor prioridad).
* Entonces las interrupciones en este caso se manejan anidando interrupciones.

Dentro del proceso de gestión de las interrupciones, lo primero que tiene que hacer la CPU es detectar el pedido de interrupción. Cuando hay múltiples fuentes de interrupción, hay varias formas para identificar el origen del pedido. Los más comunes son:

* Opción 1: 1 señal física de entrada a la CPU por cada Interrupción.
* Opción 2: 1 única señal física de entrada a la CPU para todas las interrupciones e identificación por software.
* Opción 3: 1 única señal física de entrada a la CPU para todas las interrupciones e identificación por hardware.

Opción 1: Se tiene 1 señal física de entrada a la CPU por cada interrupción, es decir que hay múltiples líneas de pedido de interrupción en la CPU. Dado que disponer de líneas en la CPU para interrupción es costoso, la cantidad de señales para ser usadas en interrupciones se acota normalmente a un número reducido (por ejemplo 3 o 4).

* + Cada dispositivo que puede provocar interrupción tiene una entrada física de interrupción conectada directamente a la CPU.
  + La implementación es bastante sencilla.
  + Restringida por la cantidad de líneas disponibles en la CPU.

Opción 2: Hay 1 sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos.

* Para poder identificar la fuente de la solicitud, la CPU debe “preguntar” a cada dispositivo si ha producido el pedido de interrupción. Este método de consulta se conoce como encuesta o polling, y es básicamente un programa que ejecuta la CPU dentro de los servicios de interrupción.
* Al ser un esquema de detección por software, el tener que consultar uno por uno todos los dispositivos lo hacen relativamente lento e ineficiente.

Opción 3: Hay 1 sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos.

* + Para poder identificar la fuente de la solicitud, la CPU recibe a continuación, típicamente a través del bus de datos, un número que identifica la fuente de la interrupción (conocido como “vector de la interrupción”).
  + El vector es provisto por el periférico que generó el pedido, o por algún dispositivo que se ocupe de generar el número dependiendo de la interrupción a ser atendida.

Esta se conoce como interrupción vectorizada, el escenario que se tiene es el siguiente:

* + El procesador tiene una única entrada de pedido de interrupciones.
  + Hay varios “productores” de interrupciones.
  + Un “dispositivo especial” administra las necesidades propias de la interrupción. En la familia Intel, este dispositivo se conoce como Controlador Programable de Interrupciones (‘PIC’). El PIC se encarga, entre otras cosas, de generar el vector, administrar prioridades, habilitar interrupciones, etc.

Interrupciones vectorizadas con el PIC

* El PIC recibe los pedidos de interrupción, típicamente de periféricos que piden atención.
* El PIC solicita atención a la CPU con la única señal de pedido de interrupción IntR (Interrupt request)
* Cuando la CPU está lista para atender la interrupción, le avisa al PIC mediante la señal IntA (Interrupt acknowledge).
* El PIC genera en el bus de datos el número de la interrupción (vector) a ser atendida. La CPU lee ese número y busca en la memoria el vector correspondiente al servicio de esa interrupción.
* Dado que se hace por hardware es mucho más rápido.

(el PIC tiene líneas físicas, señales eléctricas, conectadas a los dispositivos externos q podían pedir interrupción al procesador)

Interrupciones del MSX88

Interrupciones por hardware

* Línea INT (interrupción enmascarable) con respuesta de reconocimiento de interrupción INTA
* Línea NMI (interrupción no-enmascarable)

Interrupciones por software

* Instrucción: INT xx
* Retorno de interrupción: IRET

Cada entrada (o vector) es una palabra doble (4 bytes), que contiene la dirección del procedimiento que brinda el servicio. La parte alta del vector es 0.

Ej.: 0000yyyy, donde yyyy es la dirección lógica/física.

Vectores preasignados:

* Tipo 0 – finaliza ejecución de programa
* Tipo 3 – punto de parada para depuración/seguimiento
* Tipo 6 – lectura de entrada STD. Requiere el uso de BX.
* Tipo 7 – escritura de salida STD. Requiere BX y AL.

**Registros internos PIC:**

* INT0...INT7: Vector de cada interrupción
* IRR: petición de interrupción: Indica con bit en 1. Indican cuales dispositivos externos solicitan interrumpir.
* ISR: interrupción en servicio: Indica con bit en 1. Indica q dispositivo externo está siendo atendida.
* IMR: máscara de interrupción: enmascara con 1. Habilita o deshabilita alguna interrupción.
* EOI: para comandos: para fin de interrupción escribir 20H. Le avisa al PIC q la interrupción ya fue atendida.

Los registros internos del PIC se sitúan a partir de la dirección 20h. Son accedidos con operaciones lectura y escritura en el espacio de E/S (in y out).

Interrupciones hardware asignadas:

* Int 0: tecla f10
* Int 1: timer
* Int 2: handshake
* Int 3: dma
* Int 4 a int 7 no se usan.

**Clase 3**

**SUBSISTEMA E/S:**

El subsistema de E/S comprende los dispositivos que están conectados al bus del sistema y proveen los servicios de transferencia de datos con los Periféricos.

Debido a la gran variedad de Periféricos con los que se requiere intercambiar información,

el subsistema de E/S tiene que ser lo suficientemente flexible para permitir:

➢ Trasmisión de diferentes cantidades de datos

➢ Rango de velocidades de transmisión muy amplio.

➢ Diferentes formatos de dato y tamaño de palabra.

En general, todos los Periféricos son más lentos que la CPU y la Memoria. Los dispositivos que forman parte del subsistema de E/S permiten descongestionar el trabajo de la CPU.

**Módulo de interfaz (“Puerto”) de E/S:**

Los dispositivos más sencillos para implementar las transferencias de E/S con periféricos son los Puertos de E/S.

Los puertos de E/S realizan la interfaz entre el procesador/memoria y un periférico. Estos son, por lo general, administrados por el SO a través de drivers específicos.

La conexión con el Periférico provee 2 tipos de informaciones:

➢ 1.- Datos: Información útil a transferir

➢ 2.- Control y Estado: información que permite realizar la transferencia (en lo posible libre de errores). Por ejemplo: sentido de la transferencia (de entrada, de salida), operación (de lectura, de escritura), estado del periférico (listo, no-listo, en falla).

Puerto de E/S:

El Puerto de E/S debe ejecutar 2 tipos de comunicaciones:

1.- Hacia el Periférico (comunicaciones externas con la periferia):

• Transferir datos con el periférico (incluye adaptación eléctrica).

• Controlar y temporizar uno o más dispositivos externos.

• Almacenar temporalmente datos (“buffer”).

• Detectar errores.

2.- Hacia el Bus (internas con CPU y Memoria):

• Interpretar las órdenes que recibe de CPU y transmitirlas al periférico.

• Transferir datos con la CPU (registros) y Memoria.

• Informar a la CPU del estado del periférico.

Tipos de Puerto de E/S básicos:

**Puerto paralelo:** hay varias líneas de datos (n) que transfieren n bits simultáneamente entre el puerto de E/S y el periférico. (Ejemplos: impresora paralelo, scanner, etc.)

Se requiere disponer de una conexión mediante un cable que incluya al menos los n bits de datos, lo que lo hace un método bastante costoso.

**Puerto serie:** hay 1 línea de dato para la transferencia entre el puerto de E/S y el periférico. (Ejemplos: impresora serie, red Ethernet, mouse, teclado, etc.)

Se requiere disponer de una conexión mediante un cable sencillo. El costo es mucho menor y los datos deben serializarse (transmitirse de 1 bit por vez, uno a continuación del otro) lo que en teoría sería mucho más lento que transmitir de a n bits simultáneamente, como lo hace el puerto paralelo.

Registros de un Puerto de E/S:

Desde el punto de vista de la CPU, una operación de E/S requiere acceder a los registros internos del Módulo de Interfaz de E/S. Los registros pueden ser de lectura y/o escritura.

Dentro de Módulo de E/S hay 2 tipos de registros:

➢ de DATOS: interviene en la transferencia de entrada o de salida del dato a intercambiar entre el Sistema de cómputo y el periférico.

➢ de CONTROL: registros que controlan y registran el funcionamiento del módulo, la transferencia, y el periférico.

de Datos: la transferencia de un dato entre el Sistema de cómputo y el periférico consiste básicamente en:

➢ Operación de entrada: lectura de un registro de dato de entrada (es decir un registro escrito por el periférico y leído por la CPU)

➢ Operación de salida: escritura de un registro de dato de salida (es decir, un registro escrito por la CPU y leído por el periférico).

de Control y Estado: la supervisión y/o control de la transferencia requiere de:

➢ Control: adecuar la configuración del módulo para ajustar formatos, sincronizaciones, etc.

➢ Estado: registrar el estado operativo del módulo y del periférico.

Acceso al Subsistema de E/S

Desde el punto de vista de la CPU, el Subsistema de E/S está compuesto por un conjunto de registros a los que accede para una operación de entrada o de salida.

Existen 2 técnicas de acceso a estos registros:

➢ Espacio de E/S compartido con memoria (memory-mapped)

➢ Espacio de E/S separada de la memoria

Espacio de E/S compartido con memoria (memory-mapped)

En esta técnica los registros de los dispositivos de E/S y memoria comparten un único espacio de direcciones.

➢ Los registros de la E/S se comportan idéntico a una memoria de lectura/escritura.

➢ No hay instrucciones específicas para E/S, se usan las mismas instrucciones de movimiento de datos a memoria.

Espacio de E/S separado de memoria (modelo Intel)

En esta técnica los registros de los dispositivos de E/S y la memoria están en diferentes espacios de direcciones.

➢ Dado que el bus de direcciones es compartido por la memoria y el subsistema de E/S, se requieren señales de control adicionales para identificar a donde está accediendo la CPU: a la memoria, o a la E/S.

➢ Hay instrucciones específicas de E/S, distintas de las instrucciones de acceso a la memoria.

➢ El procesador dispone de instrucciones específicas para acceder a los registros que están en el subsistema de E/S.

➢ Cuando se ejecutan estas instrucciones específicas, en el bus de control se identifica el acceso al mapa de direcciones de E/S. Para el resto de las instrucciones en el bus de control se identifica el acceso a la Memoria.

**Gestión de la transferencia:**

Desde el punto de vista de la gestión para transferir datos entre el Sistema de cómputo y el periférico, existen 3 estrategias básicas de implementación:

➢ E/S Programada y espera de respuesta

➢ E/S Programada y administrada por interrupción

➢ E/S con acceso directo a memoria (DMA)

Las 2 primeras opciones requieren intervención directa de la CPU, es decir, que la CPU participa en la transferencia de todos los datos (byte o word) a transferir.

E/S Programada y espera de respuesta:

La CPU interviene directamente en la transferencia de cada unidad de información (byte, Word) con el módulo. Es decir que la CPU tiene control casi directo sobre la operación de E/S. Entre otras acciones hace:

➢ Comprueba el estado del dispositivo

➢ Envía los comandos requeridos (por ejemplo, de lectura, escritura)

➢ Realiza la transferencia de todos los datos (de a uno)

En cada dato que es transferido, la CPU espera que el módulo E/S termine la operación, típicamente que el periférico “acepte” el dato. Durante la espera, la CPU permanece ociosa (no deseable).

La secuencia de acciones que ejecuta la CPU son:

1. La CPU verifica el estado de periférico (preparado/no-preparado) leyendo un registro del módulo de interfaz.

2. Examina el estado del periférico chequeando el bit (o bits) que identifican dicho estado.

3. Si el dispositivo no está listo (por ejemplo, bit=0), la CPU vuelve al paso 1. Este lazo significa que la CPU “espera” hasta que el periférico se pone en “preparado”, es decir, listo para la transferencia.

4. Cuando el dispositivo está listo, la CPU transfiere 1 dato hacia o desde el módulo de interfaz.

5. Si hay más datos que transferir vuelve al paso 1.

6.- Si se completó la transferencia, termina el servicio de E/S.

**ACCESO DIRECTO A MEMORIA (DMA):**

El controlador de DMA es un dispositivo capaz de controlar una transferencia de datos entre un periférico y memoria sin intervención de la CPU.

El Controlador de DMA (DMAC) debe actuar como maestro del bus durante la transferencia DMA y debe ser capaz de

• Solicitar el uso del bus mediante las señales y la lógica de arbitraje necesarias

• Especificar la dirección de memoria sobre la que se realiza la transferencia

• Generar las señales de control del bus

• Tipo de operación (lectura/escritura)

• Señales de sincronización de la transferencia

Dado que la transferencia por DMA requiere el uso del Bus, tanto el DMAC como la CPU pueden tomarlo. El DMAC y la CPU “compiten” por el uso del Bus.

Cuando el DMAC toma el Bus, actúa como “master” durante la transferencia por DMA, y debe ser capaz de:

➢ Solicitar el uso del bus mediante las señales y la lógica de arbitraje necesarias

➢ Especificar la dirección de memoria sobre la que se realiza la transferencia

➢ Generar las señales de control del bus

➢ Especificar el tipo de operación (lectura/escritura)

➢ Generar las señales de sincronización de la transferencia

Cuando la CPU entrega el bus al DMAC, se desconecta lógicamente del mismo, y es el DMAC el que toma el control del bus.

El proceso de transferencia requiere realizar una serie de acciones o fases relativamente complejas. Las principales son:

➢ 1.- Fase de Inicialización

➢ 2.- Fase de ejecución de la transferencia

➢ 3.- Fase de finalización y análisis de la transferencia

**➢ 1.- Fase de inicialización**

En la fase de inicialización la CPU debe configurar el módulo de E/S y el DMAC con los parámetros de la transferencia.

Inicialización interfaz de E/S:

➢Tipo de transferencia (lectura/escritura)

➢Configuración del periférico

➢Otra información de control para el periférico (por ejemplo si es un disco se especifica el número de pista, sector, etc.)

Inicialización DMAC:

➢Numero de bytes o palabras a transferir

➢Tipo de transferencia (lectura/escritura)

➢Dirección de memoria inicial para la transferencia

➢Otra información la transferencia.

**➢ 2.- Fase de ejecución de la transferencia**

Cuando el periférico está listo, pide al DMAC iniciar la transferencia mediante una señal física. Cuando el DMAC recibe el pedido del periférico, pide el control del bus mediante alguna señal especial a la CPU. La CPU típicamente dispone de algunas señales destinadas a implementar las transferencias por DMA. Cuando reconoce el pedido de DMA, la CPU entrega (libera) el bus y se “desconecta” (ya no controla el bus) lógicamente del mismo. La CPU avisa al DMAC que liberó el bus mediante otra señal especial y al liberar la CPU el bus, el DMAC toma el control del bus y ejecuta la transferencia hasta terminarla.

El DMAC avisa al periférico que puede iniciar la transferencia. El periférico comienza a transferir los datos, a través de bus, con la memoria, de a uno por vez.

La transferencia implica que: Bus master: DMAC + Periférico - Bus Slave: Memoria

Después de la transferencia de cada palabra se actualizan los registros del DMAC:

➢No de bytes faltantes (o cuenta de los que se transfirieron)

➢Próxima dirección de memoria a donde guardar el dato (anterior o posterior de la corriente)

Cuando el número de bytes faltantes es igual a 0 significa que transfirió todos los datos y terminó la transferencia.

**➢ 3.- Finalización de la transferencia**

Una vez que termina la fase 2, el DMAC libera el bus y le avisa a la CPU por medio de una señal física. La CPU retoma el control del bus.

El DMAC suele activar, además, una señal de interrupción para indicar a la CPU la finalización de la operación de E/S solicitada.

La CPU, mediante la interrupción, verifica el resultado de la transferencia vía los registros internos del DMAC. Algunos resultados a verificar son:

➢Transferencia OK/fallida?

➢Errores? ¿Tipo de errores?

➢Estado periférico

**VENTAJAS Y DESVENTAJAS DEL DMA:**

La principal ventaja: es la eficiencia, dado que la CPU se libera de tener que controlar la transferencia de los datos. Solo prepara la transmisión, y verifica el resultado de la misma.

La principal desventaja: se origina en el uso del bus. Como las transferencias por DMA pueden tener mayor prioridad que la CPU, se puede degradar el rendimiento de la CPU si el DMAC hace uso intensivo del bus.

Sin embargo, no necesariamente la CPU necesite todo el tiempo el bus porque:

En el caso de Computadoras con memoria caché:

La mayor parte del tiempo, la CPU lee instrucciones de la cache, por lo que no necesita usar uco el bus de memoria.

El DMAC puede aprovechar estos intervalos en los que la CPU está leyendo instrucciones de la cache (y por tanto no usa el bus de memoria) para realizar las transferencias.

En el caso de computadores sin cache:

El procesador no utiliza el bus en todas las fases de la ejecución de una instrucción.

El DMAC puede aprovechar las fases de ejecución de una instrucción en las que la CPU no utiliza el bus.

Si el DMAC sólo toma el control del bus durante los intervalos de tiempo en los que la CPU no hace uso del mismo el rendimiento del sistema no sufrirá degradación alguna.

Se distinguen dos tipos de transferencias:

• Por ráfagas (burst)

• Por robo de ciclo (cycle-stealing)

Método de transferencia de DMA por ráfaga

Es la vista hasta ahora. El DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus, el DMAC no lo libera hasta haber finalizado la transferencia de todo el bloque de datos completo.

VENTAJAS: La transferencia se realiza de forma muy rápida, limitada por la velocidad del periférico.

DESVENTAJAS: Durante el tiempo que dura la transferencia la CPU no puede utilizar el bus con memoria, lo que puede degradar el rendimiento del sistema.

Método de transferencia de DMA por robo de ciclo

El DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus al DMAC, se realiza la transferencia de una única palabra y después el DMAC libera el bus. El DMAC solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo. El uso del bus se reparte entre la CPU y el DMAC.

VENTAJAS: No se degrada tanto el rendimiento del sistema y de la CPU.

DESVENTAJAS: La transferencia puede tardar un poco más de tiempo.

\*\*(➢ Notar que, dado que la toma y liberación del bus por parte de la CPU no es una interrupción, es decir que el procesador no debe guardar el contexto (es decir, no está interrumpiendo su tarea).

➢ Si bien el trabajo de la CPU es más lento (que si no estuviera presente la transferencia por DMA), no será tanto como si ella tuviera que estar desconectada del bus todo el tiempo.

➢ En general, para transferencia de E/S de múltiples palabras, la técnica por robo de ciclo es la más eficiente, ya que permite implementar la transferencia por DMA al mismo tiempo que la CPU continúa trabajando en su tarea.)\*\*

**Canales de E/S:**

Las transferencias de E/S se pueden dividir, en función de la capacidad para interactuar con los periféricos, en varios niveles:

Nivel 1: CPU + Módulo de Interfaz de E/S o controlador: la CPU controla directamente los periféricos e interfaz, y administra la transferencia por programa (con espera).

Nivel 2: CPU + Módulo de Interfaz E/S o controlador con interrupción: la CPU controla directamente los periféricos y administra la transferencia con programa e interrupciones.

Nivel 3: DMA (DMAC + Módulo de E/S): la CPU no interviene directamente, solo prepara y supervisa la transferencia.

Nivel 4: Canal de E/S básico (Procesador básico + módulo de E/S): la CPU interviene mínimamente.

Nivel 5: Canal de E/S inteligente (Procesador inteligente + módulo de E/S): la CPU no interviene, excepto situaciones especiales.

En el nivel más alto de la escala de transferencias de E/S están los canales de E/S, que representan una extensión al concepto de DMA.

Los canales de E/S tienen la habilidad de ejecutar programas de servicios de E/S, lo que les permite tener un completo control de la transferencia de datos. La CPU no ejecuta las instrucciones de E/S, las realiza el procesador incluido en el canal. El programa que ejecuta el procesador interno del canal está almacenado en la memoria principal.

La CPU solamente interviene para iniciar la transferencia, y dar la orden de ejecutar el programa de E/S que está en memoria. El programa de servicio de E/S especifica dispositivos, áreas de memoria a usar, prioridades y acciones ante errores.

Hay 2 tipos básicos de canales de E/S:

El Selector y el Multiplexor

Canal selector de E/S:

El canal controla varios dispositivos de alta velocidad, de a uno por vez.

El canal selecciona un dispositivo y efectúa la transferencia sobre el dispositivo seleccionado. Cada dispositivo tiene asociado un controlador o módulo de E/S que lo maneja, por lo tanto, el canal de E/S ocupa el lugar de la CPU en el control del módulo de E/S.

Canal Multiplexor de E/S

El canal controla varios dispositivos de alta velocidad, incluso simultáneamente.

El canal mutiplexa la atención entre los dispositivos seleccionados. El multiplexado puede ser:

Multiplexor de bytes: acepta y transmite de a caracteres.

Multiplexor de bloques: intercala bloques de datos desde distintos dispositivos.

\*\***Un canal selector** controla varios dispositivos de velocidad elevada y, en un instante dado, se dedica a transferir datos a uno de esos dispositivos. Es decir, el canal de E/S selecciona un dispositivo y efectúa la transferencia de datos. Cada dispositivo, o pequeño grupo de dispositivos, es manejado por un controlador, o módulo de E/S, que es similar a los módulos de E/S que se han discutido. Así, el canal de E/S se utiliza en lugar de la CPU para controlar estos controladores de E/S. **Un canal multiplexor** puede manejar las E/S de varios dispositivos al mismo tiempo. Para dispositivos de velocidad reducida, un multiplexor de byte acepta o transmite caracteres tan rápido como es posible a varios dispositivos. Por ejemplo, la cadena de caracteres resultante a partir de tres dispositivos con diferentes velocidades y cadenas\*\* LIBRO