Apuntes
Conceptos de Arquitectura de Computadoras

Polanis, Iván Valentín

Índice

1. Introducción a la Arquitectura de Computadoras **3**
- 1.1. Arquitectura Von Neumann . . . 3
- 1.2. Repertorio de instrucciones . . . 3
  - 1.2.1. Elementos de las instrucciones de maquina . . . 4
  - 1.2.2. Tipos de instrucciones . . . 4
  - 1.2.3. Número de direcciones . . . 5
  - 1.2.4. Diseño del repertorio de instrucciones . . . 5
- 1.3. Tipos de operando . . . 5
  - 1.3.1. Direcciones . . . 6
  - 1.3.2. Números . . . 6
  - 1.3.3. Caracteres . . . 6
  - 1.3.4. Datos lógicos . . . 6
- 1.4. Orden de los bytes . . . 6
- 1.5. Modos de direccionamiento . . . 7
  - 1.5.1. Inmediato . . . 7
  - 1.5.2. Directo . . . 7
  - 1.5.3. Indirecto . . . 7
  - 1.5.4. Registro . . . 7
  - 1.5.5. Indirecto con registro . . . 7
  - 1.5.6. Indirecto con desplazamiento . . . 8
  - 1.5.7. Pila . . . 8
- 1.6. Tipos de operaciones . . . 8
- 1.7. Organización del procesador . . . 8
- 1.8. Organización de los registros . . . 9
  - 1.8.1. Registros visibles por el usuario . . . 9
  - 1.8.2. Registros de control y de estado . . . 9
- 1.9. Ciclo de instrucción . . . 9

2. Interrupciones **11**
- 2.1. Tipos de interrupciones . . . 11
  - 2.1.1. Interrupciones por hardware . . . 11
- 2.2. Interrupciones por software . . . 12
- 2.3. Interrupciones múltiples . . . 12
- 2.4. Funcionamiento de las E/S . . . 13
- 2.5. Reconocimiento de interrupciones . . . 13

3. Entrada y salida **15**
- 3.1. Dispositivos externos . . . 15
- 3.2. Funciones de un módulo de E/S . . . 16
- 3.3. Entrada y salida programada . . . 16
  - 3.3.1. Órdenes de E/S . . . 17
- 3.4. Entrada y salida con interrupciones . . . 18
  - 3.4.1. Cuestiones de diseño . . . 18
- 3.5. Acceso directo a memoria . . . 18
  - 3.5.1. DMA modo ráfaga . . . 19
  - 3.5.2. DMA modo robo de ciclo . . . 19
- 3.6. Canales de E/S . . . 20

A. Pilas **21**

B. Interfaces de espacio **21**

C. PIO **22**

1. Introducción a la Arquitectura de Computadoras

Cuando se describe un computador, frecuentemente se distingue entre *arquitectura* y *organización*.

La *arquitectura* de computadoras se refiere a los atributos de un sistema que son visibles a un programador, o para decirlo de otra manera,aquellos atributos que tiene un impacto directo en la ejecución lógica de un programa. En cambio, la *organización* de computadores se refiere a las unidades funcionales y sus interconexiones, que dan lugar a especificaciones arquitectónicas.

1.1. Arquitectura Von Neumann

La tarea de cargar y modificar programas para el ENIAC[1] era extremadamente tediosa. El proceso de programación podría ser más fácil si el programa se representara en una forma adecuada parta ser guardado en la memoria junto con los datos. Esta idea conocida como *concepto de programa-almacenado*, se atribuye al matemático y físico alemán John von Neumann.

La **unidad central de procesamiento (CPU)** está constituida por la **unidad de control (UC)** y la unidad **aritmético-lógica (ALU)**. Los datos e instrucciones deben introducirse en el sistema y los resultados se proporcionarán mediante componentes de **entrada/salida (E/S)**. Para almacenar temporalmente los datos e instrucciones, se utiliza una **memoria principal**.

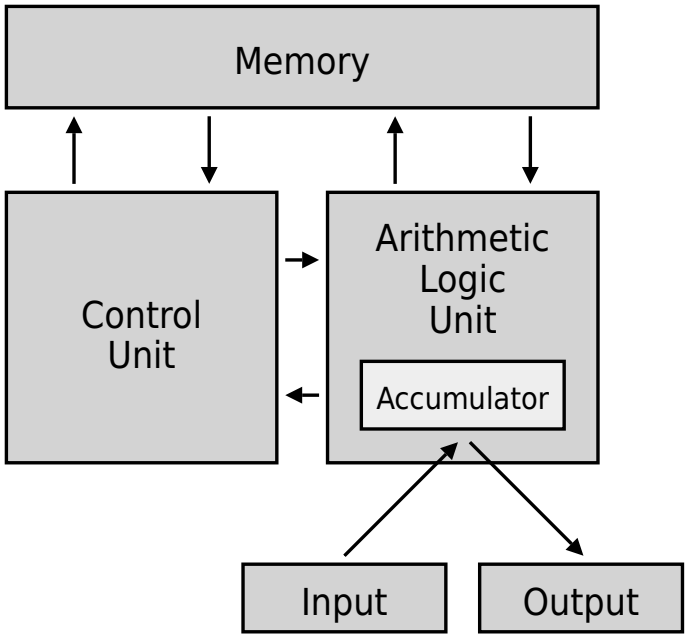


Figura 1: Arquitectura Von Neumann

1.2. Repertorio de instrucciones

El funcionamiento del procesador está determinado por las instrucciones que ejecuta. Estas instrucciones se denominan *instrucciones de maquina*. Al conjunto de instrucciones distintas que puede ejecutar el procesador se denomina *repertorio de instrucciones*.

[1]Electronic Numerical Integrator And Computer, diseñado y construido bajo la supervisión de John Mauchly y John Presper Eckert en la Universidad de Pennsylvania, fue el primer computador electrónico de propósito general del mundo.

1.2.1. Elementos de las instrucciones de maquina

Cada instrucción debe contener la información que necesita el procesador para su ejecución. Dichos elementos son:

- **Código de operación**: especifica la operación a realizar (suma, E/S, etc.).
- **Referencia a operandos fuente u origen**: la operación puede implicar a uno o más operandos origen, es decir operandos que son entradas para la instrucción.
- **Referencia al operando de destino**: la operación puede producir un resultado que debe ser almacenado en un operando destino.
- **Referencia a la siguiente instrucción**: especifica la ubicación de la siguiente instrucción a ejecutar, (generalmente la misma está implícita).

La siguiente instrucción a captar está en memoria principal o, en el caso de un sistema de memoria virtual, bien en memoria principal. Los operandos origen y destino pueden estar en alguna de las tres áreas siguientes:

- **Memoria principal o virtual**: como en las referencias a instrucciones siguientes, debe indicarse la dirección de memoria principal.
- **Registro del procesador**: un procesador contiene uno o más registros que pueden ser referenciados por instrucciones máquina.
- **Dispositivo de E/S**: la instrucción debe especificar el módulo y dispositivo de E/S para la operación.

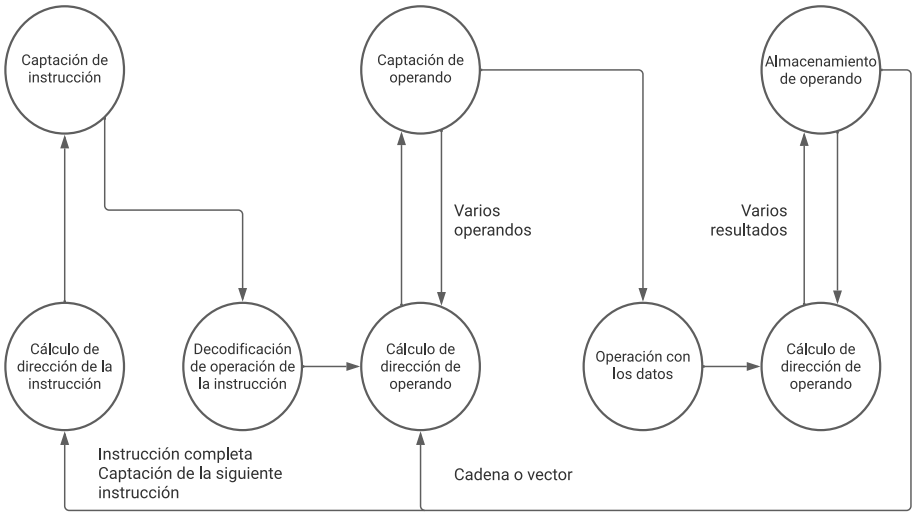


Figura 2: Ciclo de instrucción

1.2.2. Tipos de instrucciones

Cualquier programa escrito en alto nivel debe traducirse a lenguaje máquina para ser ejecutado. El repertorio de instrucciones máquina debe ser suficientemente amplio como para expresar cualquiera de las instrucciones e un lenguaje de alto nivel. Teniendo esto presente, los tipos de instrucciones se pueden clasificar de la siguiente manera:

- **Procesamiento de datos**: son instrucciones aritméticas y lógicas.
- **Almacenamiento de datos**: son instrucciones de memoria.
- **Transferencia de datos**: son instrucciones de E/S.
- **Control**: son instrucciones de comprobación y bifurcación.

Las instrucciones *aritméticas* proporcionan capacidad computacional para procesar datos numéricos. Las instrucciones *lógicas* operan con los bits de una palabra en lugar considerarlos como números. Las instrucciones *memoria* permiten transferir los datos entre la memoria y los registros. Las instrucciones *E/S* se necesitan para transferir programas y datos a memoria y devolver resultados de los cálculos al usuario.

1.2.3. Número de direcciones

Una de las formas tradicionales de describir la arquitectura de un procesador es en términos del número de direcciones contenidas en cada instrucción. Esta dimensión se va haciendo menos significativa a medida que aumenta la complejidad del diseño del procesador.

La mayoría de las instrucciones tienen una, dos o tre direcciones, estando implícita la dirección de la instrucción siguiente (obtenida a través del contador de programa)

- **Instrucciones de una dirección**: para funcionar, una segunda dirección debe estar implícita. La dirección implícita es el registro del procesador *acumulador*. El acumulador contiene uno de los operandos y se emplea para almacenar el resultado.
- **Instrucciones de dos direcciones**: para operaciones binarias una de las direcciones debe hacer el servicio doble de uno de los operandos y del resultado.
- **Instrucciones de tres direcciones**: no son comunes tal que requieren formatos relativamente largos para albergar las tres referencias. Una dirección es el destino y las otras dos son los operandos.

1.2.4. Diseño del repertorio de instrucciones

El repertorio de instrucciones es el medio que tiene el programador para controlar el procesador. En consecuencia deben considerarse las necesidades del programador a la hora de diseñar el repertorio de instrucciones.

Los aspectos fundamentales de diseño más importantes son:

- El **repertorio de operaciones**: cuántas y qué operaciones considerar, cuán complejas deben ser.
- Los **tipos de datos**: los distintos tipos de datos con los que se efectúan operaciones.
- Los **formatos de instrucciones**: longitud de la instrucción (en bits), número de direcciones, tamaño de los distintos campos, etc.
- Los **registros**: número de registros del procesador que pueden ser referenciados por las instrucciones, y su uso.
- El **direccionamiento**: el modo o modos de direccionamiento mediante los cuales puede especificarse la dirección de un operando.

1.3. Tipos de operando

Las instrucciones máquina operan con datos. Las categorías más importantes de datos son:

- Direcciones
- Números
- Caracteres
- Datos lógicos

1.3.1. Direcciones

1.3.2. Números

En los computadores son usuales tres tipos de datos numéricos:

- Enteros o en coma fija.
- En coma flotante.
- En decimal.

Hay que aclarar que hay un límite para la magnitud de los números representables en una máquina y en el caso de números de coma flotante, su precisión está limitada. Por tanto, el programador debe ser consciente de las consecuencias del redondeo, el desbordamiento o el desbordamiento a cero.

1.3.3. Caracteres

Una forma bastante común de datos es el texto o secuencias de caracteres. Aunque la información textual sea más conveniente para las personas, los computadores trabajan mejor con números. Por tanto, los caracteres se representan mediante números. En la mayoría de los sistemas, los caracteres se representan mediante un código de 8 bits. El código ASCII[2] es el más común.

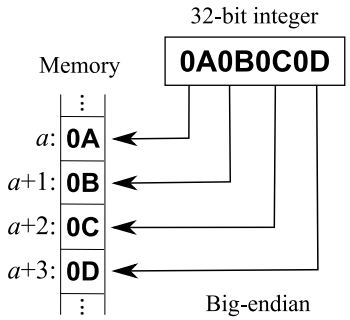
1.3.4. Datos lógicos

A veces resulta útil considerar una unidad de n bits como n elementos o datos de un bit, donde cada elemento tiene un valor de 1 o 0. Cuando los datos son vistos de esta manera, se consideran datos *lógicos*. Las principal ventaja es que nos permiten almacenar una matriz de elementos binarios o booleanos, por lo que la memoria puede ser utilizada de forma eficiente. También hay ocasiones en las que queremos manipular bits individuales de un dato.

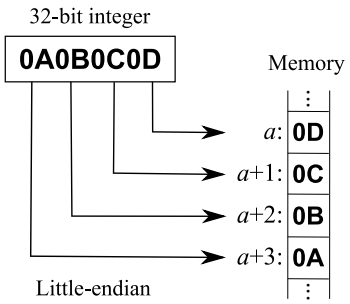
1.4. Orden de los bytes

Supongamos una memoria direccionable de a byte, es decir, que cada byte tiene un dirección única. ¿Cómo se leerán los números que ocupan más de un byte? ¿Cómo se escribirán los números que ocupan más de un byte? Para este problema se crearon dos formatos de almacenamiento:

- **Big endian**: el byte más significativo se almacena en la dirección con valor numérico más bajo.
- **Little endian**: el byte menos significativo se almacena en la dirección con valor numérico más bajo.



(a) Big endian



(b) Little endian

[2] American Standard Code for Information Interchange

1.5. Modos de direccionamiento

El campo o campos de direcciones en un formato de instrucción usual está bastante limitado. Para poder referenciar un rango elevado de posiciones se han empleado diversas técnicas de direccionamiento. Las más comunes son:

- **Inmediato**
- **Directo**
- **Indirecto**
- **Registro**
- **Indirecto con registro**
- **Indirecto con desplazamiento**
- **Pila**

1.5.1. Inmediato

El operando se encuentra presente en la propia instrucción. La ventaja es que una vez captada la instrucción, no se requiere una referencia a memoria para obtener un operando. La desventaja es que el tamaño del número está restringido a la longitud del campo de direcciones.

1.5.2. Directo

El campo de direcciones contiene la dirección efectiva del operando. La ventaja es que solo se requiere una referencia a memoria y no necesita ningún cálculo especial. La limitación es que proporciona un espacio de direcciones restringido.

1.5.3. Indirecto

El campo de direcciones referencia a la dirección de una palabra en memoria, la cual contiene la dirección completa del operando.

La ventaja de esta aproximación es que para una longitud de palabra Nbits, se dispone ahora un espacio de direcciones de 2^N. La desventaja es que la ejecución de la instrucción requiere dos referencias a memoria para captar el operando.

1.5.4. Registro

El campo de direcciones contiene el número de registro que contiene el operando. La ventaja es que la ejecución de la instrucción no requiere ninguna referencia a memoria. La desventaja es que el número de registros es limitado.

1.5.5. Indirecto con registro

El campo de direcciones contiene el número de registro que contiene la dirección del operando. La ventaja es que la ejecución de la instrucción requiere una referencia a memoria. La desventaja es que el número de registros es limitado.

1.5.6. Indirecto con desplazamiento

El direccionamiento con desplazamiento requiere que las instrucciones tengan dos campos de direcciones, al menos uno de los cuales explícito. El valor contenido en uno de los campos de direcciones se utiliza directamente. El otro campo, se refiere a un registro cuyo contenido se suma al valor del campo de direcciones. El resultado de la suma se utiliza como la dirección del operando.

- **Direccionamiento relativo**: el registro referenciado implícitamente es el **PC**.
- **Direccionamiento con registro base**: el registro referenciado contiene una dirección de memoria, y el campo de dirección contiene un desplazamiento desde dicha dirección. La referencia a registro puede ser explícita o implícita.
- **Indexado**: el campo de dirección referencia una dirección de memoria principal, y el registro referenciado contiene un desplazamiento positivo desde esa dirección.

1.5.7. Pila

El modo de direccionamiento de pila, (explicado en el ApéndiceA), es una forma de direccionamiento implícito. Las instrucciones máquina no necesitan incluir una referencia a memoria sino que operan implícitamente con la cabecera de la pila.

1.6. Tipos de operaciones

Las operaciones que se pueden realizar en un computador se pueden clasificar en:

- **Transferencia de datos**: mueven datos entre registros, registros a memoria y memoria a registros. Se debe especificar la ubicación del operando fuente y destino, el tamaño de los datos y el modo de direccionamiento.
- **Aritméticas**: son las operaciones ADD, SUB, MUL y DIV. Puede incluirse también las operaciones INC o DEC (que solamente necesitan un operando), NEG (cambia el signo en Ca2) y ABS (valor absoluto).
- **Lógicas**: son las operaciones AND, OR, XOR, NOT. Puede incluirse también las operaciones SHL, SHR, ROL, ROR (que solamente necesitan un operando).
- **Conversión**: son operaciones para cambiar formatos de datos, por ejemplo de EBCDIC[3] a ASCII.
- **E/S**: son operaciones para leer o escribir datos en dispositivos de E/S. Son las instrucciones IN y OUT. Se pueden realizar a través de un controlador aparte *DMA*[4].
- **Control**: son las instrucciones de salto, llamadas a subrutinas, etc. Algunas de ellas pueden ser JMP, JZ, JNZ, CALL, RET, etc.

1.7. Organización del procesador

Para comprender la organización del procesador, consideremos los requisitos que ha de cumplir:

- **Captar instrucción**: el procesador lee una instrucción de la memoria.
- **Interpretar instrucción**: la instrucción se decodifica para determinar qué acción es necesaria.
- **Captar datos**: la ejecución de una instrucción puede exigir leer datos de la memoria o de un módulo de E/S.

[3] Extended Binary Coded Decimal Interchange Code
[4] Direct Memory Access

- **Procesar datos**: la ejecución de una instrucción puede exigir llevar a cabo alguna operación aritmética o lógica.
- **Escribir datos**: los resultados de una ejecución pueden exigir escribir datos en la memoria o en un módulo de E/S.

1.8. Organización de los registros

Los registros del procesador son de dos tipos:

- **Registros visibles por el usuario**: permiten a programador de lenguaje máquina minimizar las referencias a memoria principal por medio de la optimización del usa de registros.
- **Registros de control y de estado**: son utilizados por la unidad de control para controlar el funcionamiento del procesador y por programas privilegiados del sistema operativo para controlar la ejecución de programas.

1.8.1. Registros visibles por el usuario

Un registro visible por el usuario es aquél que puede ser referenciado por medio del lenguaje máquina que ejecuta el procesador.

Los **registros de uso general** pueden ser asignados por el programador a diversas funciones. Su uso dentro del repertorio de instrucciones puede ser ortogonal a la operación. Por ejemplo, el registro R_0 puede ser usado como un registro de acumulación, un registro de índice o un registro de apuntador.

Los **registros de datos** pueden usarse unicamente para contener datos y no se pueden emplear en el cálculo de la dirección de un operando.

Los **registros de dirección** pueden ser de uso más o menos general, o pueden estar dedicados a un modo de direccionamiento particular.

Los **registros de código de condición** son bits fijados por el hardware del procesador como resultado de alguna operación. Estos bits son usados por el programador para controlar el flujo de ejecución de un programa.

1.8.2. Registros de control y de estado

Hay diversos registros del procesador que se emplean para controlar su funcionamiento. La mayoría de ellos, no son visibles por el usuario. Para la ejecución de una instrucción son esenciales cuatro registros:

- **PC**: contiene la dirección de la instrucción que se está ejecutando.
- **IR**: contiene la instrucción que se está ejecutando.
- **MAR**: contiene la dirección de memoria que se está accediendo.
- **MBR**: contiene la palabra de datos a escribir en memoria o la palabra leída más recientemente.

1.9. Ciclo de instrucción

El ciclo de instrucción es el conjunto de operaciones que se llevan a cabo para ejecutar una instrucción. El ciclo de instrucción se divide en tres etapas:

- **Captación**: llevar la siguiente instrucción de la memoria al procesador. La instrucción se almacena en el registro IR. El registro PC se incrementa, (a no ser que se indique lo contrario). La UC interpreta la instrucción captada y lleva a cabo las operaciones necesarias para ejecutarla.
- **Ejecución**: interpretar el código de operación y llevar a cabo la operación indicada. Las operaciones posibles son:

    - **CPU-Mem**: deben transferirse datos desde la CPU a la memoria o viceversa.
    - **CPU-E/S**: deben transferirse datos desde la CPU a un dispositivo de E/S o viceversa.
    - **Procesamiento de datos**: la CPU ha de realizar alguna operación aritmética o lógica con los datos.
    - **Control**: una instrucción puede especificar que la secuencia de ejecución se altere.
- **Interrupción**: si las interrupciones están habilitadas y ha ocurrido una interrupción, guardar el estado del proceso actual y atender la instrucción.

2. Interrupciones

Con el uso de interrupciones, el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S está en curso. El procesador puede ser notificado cuando la operación de E/S ha terminado, y puede continuar con la ejecución de la instrucción que se interrumpió. Cuando el dispositivo externo está listo para aceptar más datos del procesador, el módulo de E/S de este dispositivo externo envía una señal de *petición de interrupción* al procesador. El procesador responde suspendiendo la operación del programa que estaba ejecutando y salta a un programa, conocido como gestor de interrupción, que da servicio a ese dispositivo concreto, y prosigue con la ejecución del programa original después de haber dado dicho servicio al dispositivo.

Para permitir el uso de interrupciones, se añade un *ciclo de interrupción* al ciclo de instrucción. En el ciclo de interrupción el procesador comprueba si se ha generado alguna interrupción, indicada por la presencia una señal de interrupción. Si no hay señales, el procesador continúa con el ciclo de captación y accede a la siguiente instrucción del programa. Si hay alguna petición, el procesador suspende la ejecución del programa en curso y guarda su contexto. El procesador carga el contador de programa con la dirección de comienzo de una rutina de interrupción.

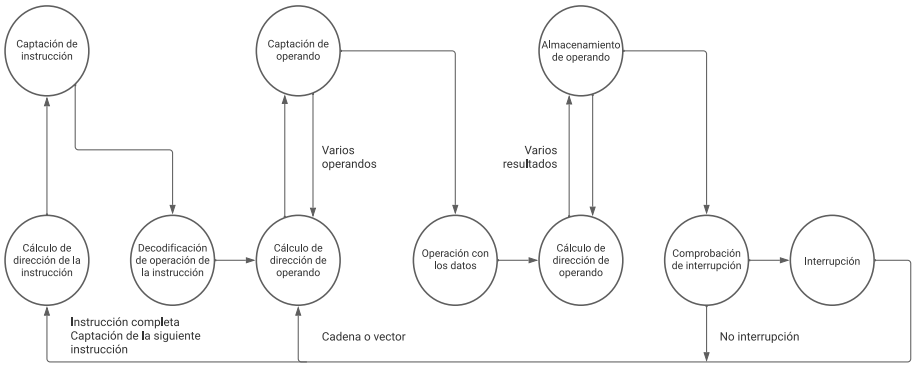


Figura 3: Ciclo de instrucción (con interrupciones)

2.1. Tipos de interrupciones

Podemos clasificar las interrupciones dependiendo de su importancia:

- **No enmascarables**: son aquellas interrupciones que no pueden ser ignoradas. Indican eventos peligros o de alta prioridad y requieren un respuesta eficiente y rápida.
- **Enmascarables**: son aquellas interrupciones que pueden ser ignoradas. Sus eventos no configuran peligro ó pueden esperar. La posible solicitud de interrupción puede inhibirse con instrucciones especiales.

2.1.1. Interrupciones por hardware

Las interrupciones por hardware son las generadas por los dispositivos de E/S. El sistema de cómputo tiene que manejar estos eventos externos 'no planeados'. Además, las mismas no están relacionadas con el proceso que se encuentra ejecutándose en ese momento. Son conocidas como *interrupt request*.

Las **traps/excepciones** son interrupciones por hardware creadas por el procesador moderno en respuesta a ciertos eventos internos:

- **Condiciones excepcionales**: overflow en el ALU de punto flotante.
- **Falla de programa**: tratar de ejecutar una instrucción no definida.
- **Falla de hardware**: error de paridad de memoria.

- **Accesos no alineados ó a zonas de memoria protegidas**: acceso a memoria no permitido.

2.2. Interrupciones por software

Las interrupciones por software son instrucciones explícitas que afectan al procesador de la misma manera que las interrupciones por hardware. La instrucción INT n permite a las interrupciones ser generadas desde dentro del software utilizando el número del vector de interrupción como un operando.

Estas interrupciones permiten depurar los gestores de interrupción. También se utilizan para invocar funciones del sistema operativo, permitiendo asi cargar subrutinas del sistema operativo en algún lugar y puedan utilizarse.

2.3. Interrupciones múltiples

Hasta ahora únicamente se ha discutido la existencia de una sola interrupción. Sin embargo, es posible que se produzcan varias interrupciones simultáneamente. Se pueden seguir dos alternativas para tratar las interrupciones múltiples:

- **Interrupción inhabilitada**: mientras se está atendiendo una interrupción, se deshabilitan las interrupciones. Si se produce una interrupción, queda pendiente y será examinada por el procesador una vez que este haya activado las interrupciones nuevamente. El inconveniente de este enfoque es que no tiene en cuenta la prioridad relativa ni las solicitudes con un tiempo crítico.
- **Interrupciones con prioridades**: se definen prioridades para las interrupciones y permite que una interrupción de prioridad más alta pueda interrumpir a un gestor de interrupción de prioridad menor. Cuando se ha gestionado la interrupción de prioridad más alta, el procesador vuelve a las interrupciones previas.

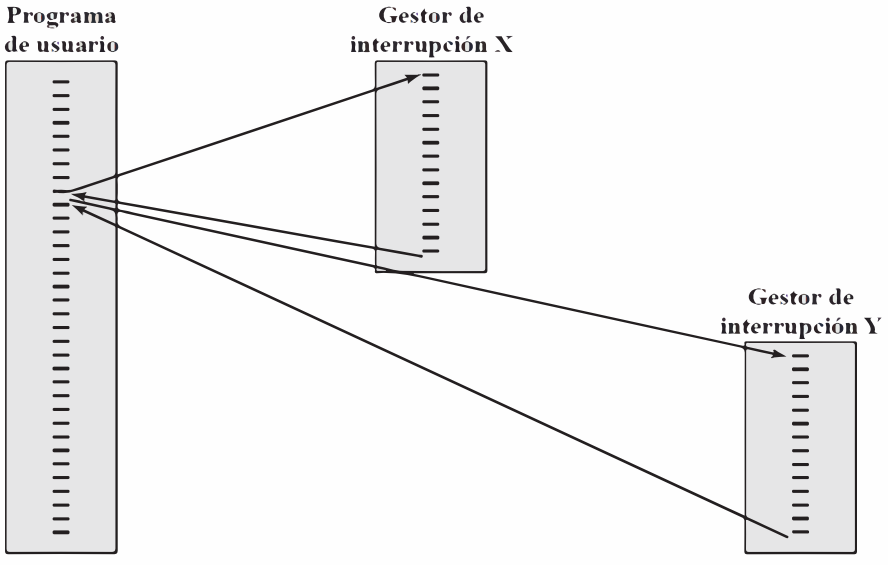


Figura 4: Interrupciones secuenciales

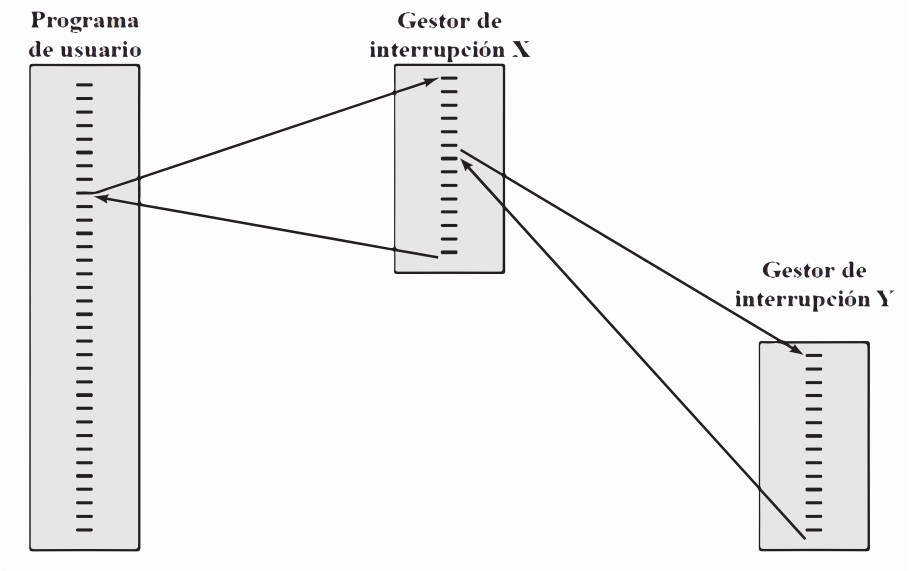


Figura 5: Interrupciones con prioridades

2.4. Funcionamiento de las E/S

Un módulo de E/S puede intercambiar datos directamente con el procesador. Igual que el procesador puede iniciar una lectura o escritura en memoria, también puede leer o escribir datos de un módulo de E/S. En algunos casos, es deseable permitir que los intercambios de E/S se produzcan directamente con la memoria. En ese caso, el procesador cede a un módulo de E/S la autoridad para leer de o escribir en memoria, para que así la transferencia de E/S-memoria pueda producirse sin la intervención sel procesador. Durante esas transferencias, el módulo de E/S proporciona a la memoria las órdenes de lectura o escritura, liberando al procesador de cualquier responsabilidad en el intercambio. Esta operación se conoce con el nombre de *DMA* (Direct Memory Access).

2.5. Reconocimiento de interrupciones

- **Interrupciones multinivel**: cada dispositivo que puede provocar una interrupción tiene una entrada física de interrupción conectada a la CPU. Es muy sencillo, pero a la vez muy caro.
- **Línea de interrupción única**: hay una sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos. Es necesario preguntar a cada dispositivo si ha producido el pedido de interrupción (técnica de *polling*).
- **Interrupciones vectorizadas**: el dispositivo que quiere interrumpir además de la señal de pedido de interrupción, debe colocar en el bus de datos un identificador.

El encargado de manejar todas las interrupciones es el **PIC**[5]. Las tareas realizas por el PIC son:

- Puesto que existen muchos dispositivos que pueden solicitar interrupciones, es responsabilidad del PIC priorizarlas cuando existen varias IRQ's simultáneas.
- Después de enviar una solicitud de interrupción, debe enviar un número de interrupción (número de vector) cuando el procesador indica que está listo para atender la petición.

[5]Programmable Interrupt Controller

- Mantiene un registro de que se está procesando una interrupción; cuando esto no sucede, no envía más peticiones hasta que este le responde con una señal EOI[6], indicando que la rutina de servicio precedente ha terminado o puede aceptar otra interrupción.
- Puede enmascarar de forma selectiva cualquiera de las 8 IRQ's que tiene conectadas.

El **PIC** tiene varios registros internos, ellos son:

- **EOI**: recibe 20H cuando el procesador ha terminado de atender la interrupción.
- **Interrupt Mask Register**: contiene los bits de máscara de las IRQ's.
- **Interrupt Request Register**: contiene los bits de estado de las IRQ's.
- **Interrupt Status Register**: contiene el bit de estado de la IRQ que se está atendiendo.
- **INT0...INT7**: son los 8 vectores de interrupción.

[6]End of Interrupt

3. Entrada y salida

Un módulo de E/S no es únicamente un conector mecánico que permite enchufar el dispositivo al bus del sistema; sino que además está dotado de cierta «inteligencia», que contiene la lógica necesaria para permitir la comunicación entre el periférico y el bus.

Las razones por las que los periféricos no se conectan directamente al bus del sistema son:

- Hay una pila variedad de periféricos con formas de funcionamiento diferentes. Podría ser imposible incorporar la lógica necesaria dentro del procesador para controlar tal diversidad de dispositivos.
- Generalmente, la velocidad de transferencia de datos de los periféricos es mucho menor que la de la memoria o el procesador.
- Por otro lado, la velocidad de transferencia de algunos periféricos es mayor que la de la memoria o el procesador.
- Los periféricos suelen utilizar datos con formatos y tamaños de palabra diferentes de los del computador a los que se conectan.

En consecuencia, se necesita un módulo de E/S. Este módulo tiene dos funciones principales:

- Realizar la interfaz entre el procesador y la memoria y los periféricos.
- Realizar la interfaz entre uno o más dispositivos periféricos.

3.1. Dispositivos externos

Un dispositivo externo se conecta al computador mediante un enlace a un módulo de E/S. El enlace se utiliza para intercambiar señales de control, estado, y datos entre el módulo de E/S y el dispositivo externo. Los dispositivos externos pueden ser:

- **E/S básicos**: monitor, mouse, teclado, etc.
- **E/S de almacenamiento**: discos duros, CD-ROM, etc.
- **Impresión**: impresoras, escáneres, etc.
- **Comunicaciones**: modems, acceso/interfaz de red, etc.
- **Multimedia**: micrófonos, altavoces, etc.
- **Automatización y control**: sensores, alarmas, etc.

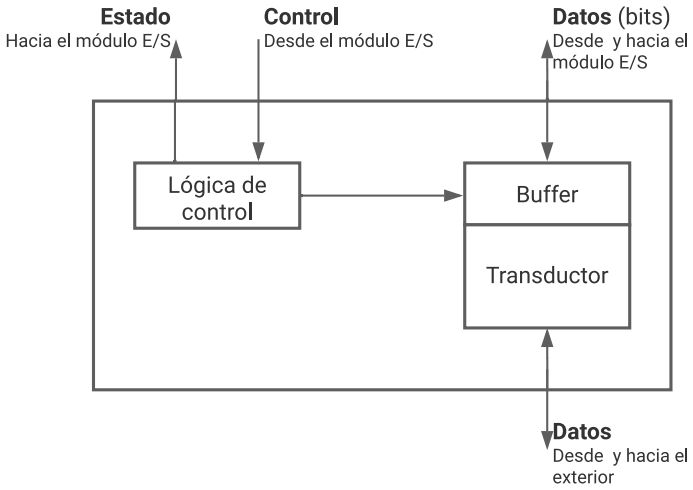


Figura 6: Dispositivo externo

La forma de un dispositivo externo se indica en la Figura 6. La conexión con el módulo de E/S se realiza a través de señales de control, estado y datos.

La *lógica de control* asociada al dispositivo controla su operación en respuesta a las indicaciones del módulo de E/S. El *transductor* convierte las señales eléctricas asociadas al dato a otra forma de energía en el caso de una salida y viceversa en el caso de unta entrada. Existe un buffer asociado al transductor para almacenar temporalmente el dato que se esta transfiriendo ente el módulo de E/S y el exterior.

3.2. Funciones de un módulo de E/S

Las principales funciones y requisitos de un módulo de E/S se encuentran dentro de las siguientes categorías:

- **Control y temporización**: coordina el tráfico entre los recursos internos y los dispositivos externos.
- **Comunicación con el procesador**: implica la decodificación de órdenes, el intercambio de datos, información del Estado y el reconocimiento de direcciones.
- **Comunicación con los dispositivos**: esta comunicación implica intercambiar órdenes, información del estado y datos.
- **Almacenamiento temporal de datos**: los datos que se transfieren entre el módulo de E/S y los dispositivos externos se almacenan temporalmente en un buffer.
- **Detección de errores**: detecta los errores e informa al procesador.

La complejidad de los módulos de E/S y el número de dispositivos externos que controlan varían considerablemente.

El funcionamiento de un módulo de E/S permite que el procesador vea a una amplia gama de dispositivos de forma simplificada. El módulo debe ocultar los detalles de temporización, formatos y electromecánica de los dispositivos externos para que el procesador pueda funcionar únicamente en términos de órdenes de lectura y escritura.

Un módulo de E/S que se encarga de la mayoría de los detalles del procesamiento, presentando al procesador una interfaz de alto nivel, se denomina *canal de E/S* o *procesador de E/S*. Un módulo que sea bastante simple qy requiera control detallado normalmente se denomina *controlador de E/S*.

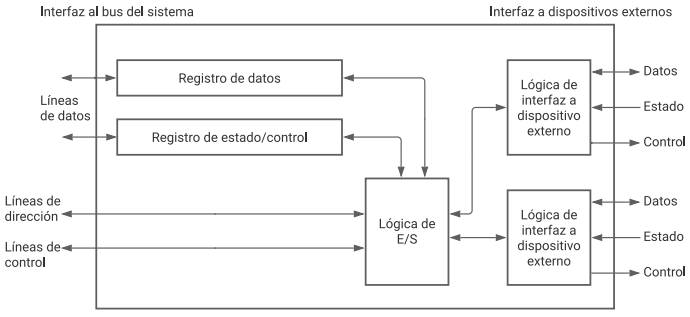


Figura 7: Estructura de un módulo de E/S

3.3. Entrada y salida programada

Los datos se intercambian entre el procesador y el módulo de E/S. El procesador ejecuta un programa que controla directamente la operación de E/S:

- Comprobación del estado del dispositivo.

- Envío de una orden de lectura o escritura.
- Transferencia del dato.

El procesador espera que el módulo de E/S termine la operación.
Los detalles de la E/S programada se muestran en la Figura 8:

- La CPU solicita la operación de E/S.
- El módulo E/S realiza la operación.
- El módulo E/S activa los bits de estado del dispositivo direccionado y espera.
- La CPU comprueba periódicamente el estado de esos bits, hasta que detecta que la operación fue completada.
- En caso contrario la CPU espera y vuelve a comprobarlo más tarde.

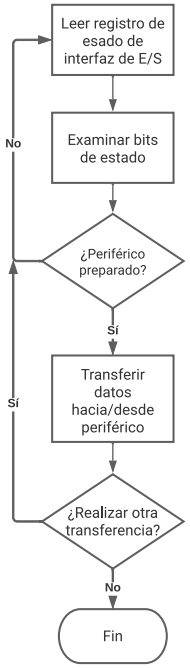


Figura 8: E/S programada

3.3.1. Órdenes de E/S

Al ejecutar una instrucción relacionada con una E/S, el procesador proporciona una dirección, especificando el módulo de E/S particular y el dispositivo externo, y una orden de E/S. Hay cuatro tipos de órdenes de E/S que puede recibir un módulo de E/S cuando es direccionado por el procesador:

- **Control**: se utiliza para activar el periférico e indicarle qué hacer. Estas órdenes son específicas del tipo particular de periférico.
- **Test**: se utiliza para comprobar diversas condiciones de estado asociadas con el módulo de e/S y sus periféricos.
- **Lectura**: hace que el módulo de E/S capte un dato de un periférico y lo sitúe en un buffer interno. Después el procesador puede obtener el dato solicitando que el módulo de E/S lo ponga en el bus de datos.
- **Escritura**: hace que el módulo de E/S capte un dato del bus de datos y posteriormente lo transmita al periférico.

3.4. Entrada y salida con interrupciones

El problema con la E/S programada es que el procesador tiene que esperar un tiempo considerable a que el módulo de E/S en cuestión esté preparado para recibir o transmitir los datos.

En la E/S con interrupciones, el módulo de E/S interrumpirá al procesador para solicitar su servicio cuando esté preparado para intercambiar datos con él.

Desde el punto de vista del procesador, las acciones son las que siguen:

- Envía una orden READ de lectura. El módulo E/S obtiene los datos del periférico mientras la CPU realiza otro trabajo.
- La CPU chequea si hay pedidos de interrupciones pendientes al final de cada ciclo de instrucción.
- La CPU detecta el pedido, guarda el contexto, interrumpe el proceso y realiza la gestión de la interrupción.
- La CPU solicita los datos. El módulo E/S transfiere los datos.

3.4.1. Cuestiones de diseño

En la implementación de las E/S mediante interrupciones aparecen dos cuestiones. Primero, cómo determina el procesador qué dispositivo ha provocado la interrupción. Y segundo, si se han producido varias interrupciones, como decide el procesador la que debe atender.

Hay cuatro tipos de técnicas que se utilizan comúnmente:

- **Múltiples líneas de interrupción**: consiste en proporcionar varias líneas de interrupción entre el procesador y los módulos de E/S. No resulta práctico dedicar más de unas pocas líneas del bus o terminales del procesador a ser líneas de interrupción.
- **Software poll**: cuando el procesador detecta una interrupción, se produce una bifurcación a una rutina de servicio de interrupción que se encarga de consultar cada módulo de E/S para determinar cuál ha provocado la interrupción. Este método es muy lento y no es muy eficiente.
- **Daisy chain**: es una alternativa a la técnica anterior. Todos los módulos de E/S comparten una línea común para solicitar interrupciones. La línea de reconocimiento de interrupción se conecta encadenando los módulos uno tras otro. Cuando el procesador recibe una interrupción, activa el reconocimiento de interrupción. Esta señal se propaga a través de la secuencia de módulos de E/S hasta que se alcanza un módulo que solicitó la interrupción. El módulo responderá colocando un vector, en el bus, que lo identifica. La CPU emplea el vector como puntero para acceder a la rutina de servicio.
- **Arbitraje de bus**: un módulo de E/S debe en primer lugar disponer del control del bus antes de poder activar la línea de petición de interrupción. Así, solo un módulo puede activar la línea en un mismo instante. Cuando el procesador detecta la interrupción, responde mediante la línea de reconocimiento de interrupción. Después, el módulo que solicitó la interrupción sitúa su vector en las líneas de datos.

3.5. Acceso directo a memoria

El DMA requiere un módulo adicional en el bus del sistema. El módulo o controlador de DMA es capaz de imitar al procesador y de recibir el control del sistema cedido por el procesador. El módulo de DMA utiliza sólo cuando el procesador no lo necesita, o debe forzar al procesador a que suspenda temporalmente su funcionamiento. Esta última técnica es la más común y se denomina *robo de ciclo*.

Cuando el procesador desea leer o escribir un bloque de datos, envía una orden al módulo de DMA, incluyendo la siguiente información:

- Si se solicita una lectura o una escritura, utilizando la línea de control de lectura o escritura el procesador y el módulo de DMA.
- La dirección del dispositivo de E/S en cuestión, indicada a través de las lías de datos.
- La posición inicial de memoria a partir de donde se lee o se escribe indicada a través de las lineas de datos y almacenada por el módulo de DMA en su registro de direcciones.
- El número de palabras a leer o escribir, también indicando a través de las líneas de datos y almacenado en el registro de cuenta de datos.

El procesador delega la operación de E/S al módulo de DMA, que se encargará de ella. El módulo de DMA transfiere el bloque completo de datos, directamente desde o hacia la memoria, sin que tenga que pasar a través del procesador. Una vez que la transferencia termina, el módulo de DMA envía una señal de interrupción al procesador.

Para una transferencia de E/S de varias palabras, el DMA es mucho más eficiente que la E/S mediante interrupciones o la E/S programada.

Aunque la transferencia por DMA puede ser más eficientes, puede haber ciertos problemas. Se puede degradar el rendimiento de la CPU si el DMA hace uso intensivo del bus.

El problema se reduce con el uso de memoria cache. La mayor parte del tiempo, la CPU lee instrucciones de la cache por lo que no necesita usar el bus de memoria. El DMA puede aprovechar estos intervalos en los que la CPU está leyendo instrucciones de la cache para realizar las transferencias.

En caso de computadores sin cache, el procesador no utiliza el bus en todas las fases de la ejecución de una instrucción. El DMA puede aprovechar las fases de ejecución de una instrucción en las que la CPU no utiliza el bus para realizar sus transferencias.

Si el DMA 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

Si pueden distinguir dos tipos de transferencias:

- Por ráfagas
- Por robo de ciclo

3.5.1. DMA modo ráfaga

Cuando la CPU concede el bus, el DMA no lo libera hasta haber finalizado la transferencia de todo el bloque de datos completo. La ventaja que tiene es que la transferencia se realiza de forma rápida. Pero la desventaja es que durante el tiempo que dura la transferencia la CPU no puede utilizar el bus con memoria, lo que puede degradar el rendimiento del sistema.

3.5.2. DMA modo robo de ciclo

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

La ventaja que tiene es que no se degrada el rendimiento del sistema. Aunque por contraparte, la transferencia tarda más tiempo en llevarse a cabo.

Para la CPU no es considerado una interrupción, por lo que no es necesario guardar el contexto. Si bien el trabajo de la CPU es lento, no será tanto como si ella realizara la transferencia. Por lo tanto, para transferencia de E/S de múltiples palabras, es la técnica más eficiente.

3.6. Canales de E/S

A medida que los computadores han evolucionado, la complejidad y sofisticación de sus componentes se ha incrementado. En ningún lugar se hace más evidente que en el funcionamiento de las E/S. Sus etapas pueden resumirse como:

1. La CPU controla directamente los periféricos.
2. Se agrega un módulo de E/S o controlador.
3. Idem 2, más llamado de interrupción.
4. El módulo de E/S provee el acceso directo a memoria (DMA).
5. El módulo de E/S tiene su propio procesador con su pequeño conjunto de instrucciones.
6. El módulo además tiene su memoria local, (se convierte en una computadora en sí mismo).

Características de los canales de E/S

El canal de E/S representa una ampliación del concepto de DMA. Un canal de E/S puede ejecutar instrucciones E/S, lo que le confiere un control completo sobre las operaciones de E/S.

La CPU no ejecuta instrucciones de E/S, sino que los canales toman control completo sobre la transferencia de datos. Las instrucciones de E/S se encuentran almacenadas en la memoria principal para ser ejecutadas por un procesador de uso específico contenido en el propio canal de E/S.

La CPU inicia una transferencia de E/S indicando al canal de E/S que debe ejecutar un programa en la memoria. El programa especifica el dispositivo, el área de memoria para almacenamiento la prioridad y las acciones a realizar en ciertas situaciones de error.

Son comunes dos tipos de canales:

- **Canal selector**: controla varios dispositivos de alta velocidad y se dedica a transferir datos a uno de esos dispositivos. El canal de E/S selecciona un dispositivo y efectúa la transferencia de datos. Los dispositivos son manejados por un controlador o módulo de E/S. Por lo tanto, el canal de E/S ocupa el lugar de la CPU en el control de esos controladores.
- **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. Para dispositivos de velocidad elevada, un *multiplexor de bloque* entrelaza bloques de datos de los distintos dispositivos.

A. Pilas

Una **pila** es un conjunto ordenado de elementos, en el que solo uno de ellos es accesible en un instante dado. El punto de acceso se denomina *cabecera* de la pila. El número de elementos en la pila, se denomina *longitud*. Es una estructura de tipo *LIFO* (Last In First Out), es decir, el último elemento en entrar es el primero en salir.

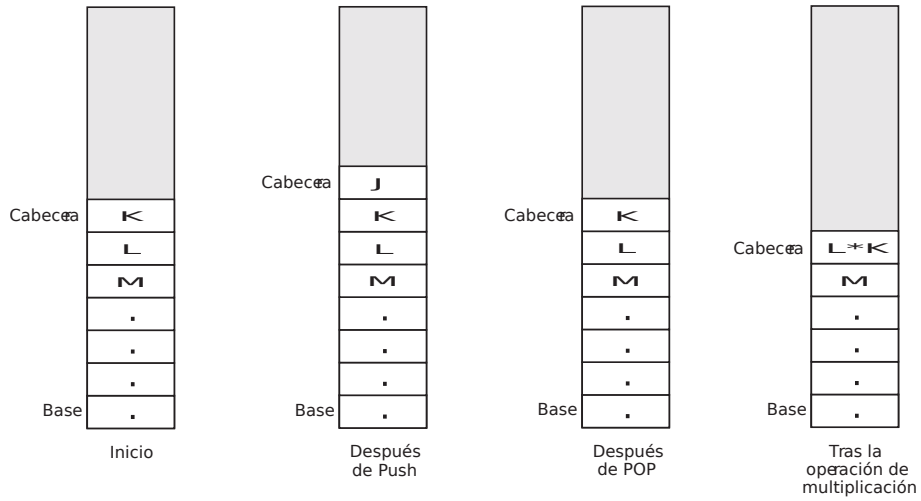


Figura 9: Pila

La pila es una estructura útil como parte de la implementación del procesador. Por ejemplo, en el procesamiento de llamadas a subrutinas, la pila se utiliza para almacenar la dirección de retorno de la subrutina.

La implementación de una pila requiere que exista un cierto conjunto de posiciones utilizadas para almacenar los elementos de la pila.En memoria principal se reserva un bloque de posiciones contiguas para la pila. Para un funcionamiento correcto se necesitan tres direcciones, normalmente memorizadas en registros del procesador:

- **Stack pointer (SP)**: contiene la dirección del tope o cabecera de pila.
- **Stack base (SB)**: contiene la dirección de la base de la pila.
- **Stack limit (SL)**: contiene la dirección de la ultima posición reservada de la pila.

B. Interfaces de espacio

Hay 2 métodos para hacer la interfaz del espacio de E/S:

E/S aislado

Esta técnica es utilizada por sistemas basados en procesadores Intel. En este tipo de interfaz, las posiciones de E/S se encuentran separadas de la memoria principal, en un espacio distinto de direcciones. Las direcciones de E/S llamadas puertos, están separadas de la memoria.

La principal ventaja de esta técnica es que todo el espacio de memoria se encuentra ocupado por la misma. La desventaja es que para transferir datos entre el μp y E/S se tienen que usar instrucciones especiales como IN y OUT.

Cuando la UC decodifica OUT ó IN, activa las líneas del bus de control iow=input/output write ó ior=input/output read.

Cuando la UC decodifica MOV, activa las líneas del bus de control mwr=memory write ó mrd=memory read.

E/S mapeada en memoria

En esta técnica, las direcciones de E/S están mapeadas en las direcciones de memoria, por lo que las direcciones de E/S pertenecen al espacio de memoria. La principal ventaja es que se puede usar todo el conjunto de instrucciones del procesador, porque todas las posiciones son tomadas como direcciones. La desventaja es que el espacio de memoria se ve reducido por la presencia de las direcciones de E/S.

C. PIO

Son 2 puertos paralelos de 8 bits: A y B. Se puede programar con bits por separados como entrada ó salida. Posee 4 registros internos de 8 bits cada un: 2 de datos, PA y PB; 2 de control, CA y CB.

Un bit en 0 en el registro CA selecciona como salida a la línea correspondiente de PA. Un bit en 1 en el registro CA selecciona como entrada a la línea correspondiente de PA. Los bits en CB controlan de la misma manera al registro PB.