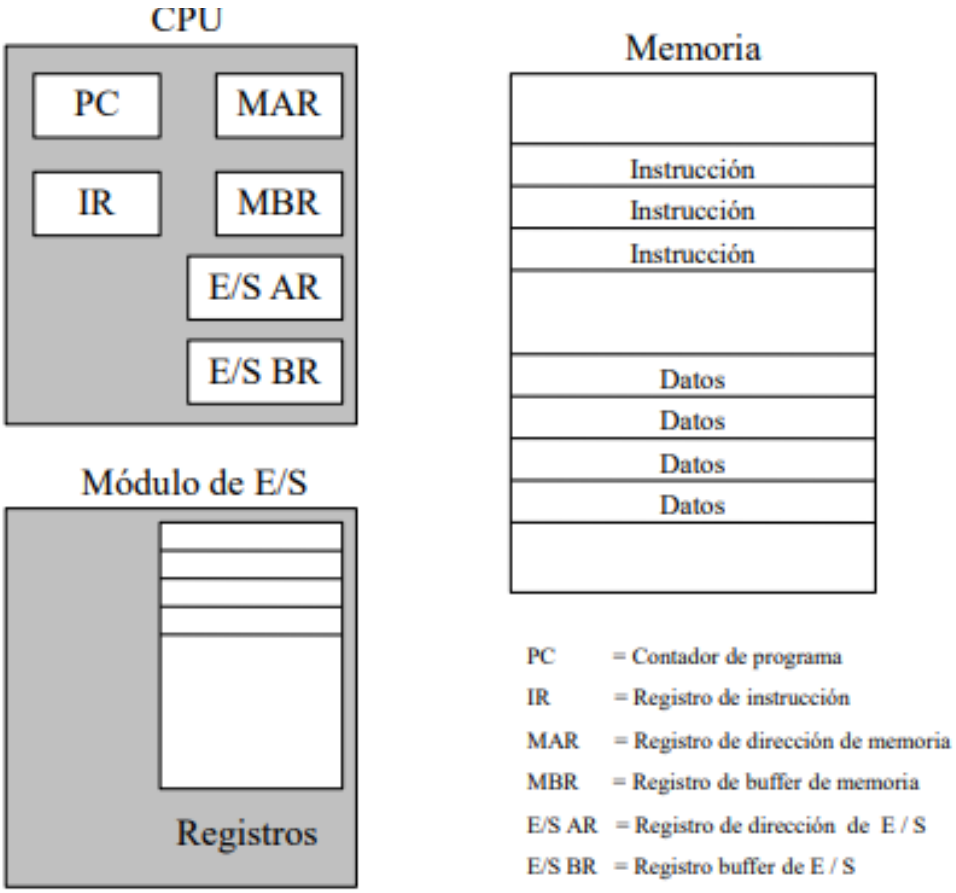
Conceptos de Arquitectura de Computadoras (CAC)

- Computadora: máquina digital (sistema de ceros y unos, relacionado con el voltaje), sincrónica (ejecuta las acciones de manera uniforme con el tiempo y en un momento en particular mediante una entidad), con capacidad de cálculo numérico y lógico, está controlada por un programa y tiene comunicación con el exterior.
- Arquitectura: atributos de un sistema que son visibles para un programador. Aquellos atributos que tienen un impacto directo en la ejecución lógica de un programa. Por ejemplo: el conjunto de instrucciones, el número de bits usados para representar los tipos de datos, los mecanismos de E/S y las técnicas de direccionamiento de memoria.
- Organización: se refiere a las unidades funcionales y sus interconexiones. Es la forma en la que se organizan y son implementados los atributos de un sistema. Es lo más cercano al hardware posible y puede interesarle al programador cuando necesite conocer la performance (cantidad de ciclos de instrucción) del sistema. Por ejemplo: señales de control, interfaces y tecnología de memoria.
- Algoritmo: especificación rigurosa (de forma clara y unívoca) de la secuencia de pasos (instrucciones) a realizar sobre un autómata para alcanzar un resultado deseado en un tiempo finito.
  Cuando el autómata es una computadora, el algoritmo se escribe en un lenguaje de máquina y pasa a ser un programa.
- Programa: conjunto de instrucciones u órdenes ejecutables (expresadas en un lenguaje de programación concreto) sobre una computadora, que permite cumplir con una función específica.
- Arquitectura Von Neumann: implementa el concepto de "programa almacenado". Propone la estructura de una computadora con 5 componentes principales:
  - Unidad de Entrada/Salida: provee las instrucciones y los datos; y envían los resultados respectivamente. Realizan la transferencia de información con unas unidades exteriores llamadas periféricos, lo que permite, entre

otras cosas, cargar datos y programas a la memoria principal e imprimir resultados en pantalla.

- Unidad de memoria: es donde se almacenan datos e instrucciones. Es una unidad dividida en celdas que se identifican mediante una dirección expresada en BCH (Binario codificado a Hexadecimal). Todas las celdas son del mismo tamaño (igual número de bits) y almacenan tanto datos como instrucciones de máquina.
- Unidad aritmético-lógica: se encarga de procesar los datos. Permite realizar una serie de operaciones elementales. Los datos sobre los que opera provienen de la memoria principal y pueden estar almacenados en forma temporal en algunos registros de la unidad.
- Unidad de control: es la encargada de dirigir la operación. Se ocupa de leer las instrucciones de maquina almacenadas en la memoria y de generar las señales de control necesarias para que el computador funcione y ejecute las instrucciones leídas.

La unidad central de procesamiento (CPU) está constituida por la unidad de control (UC) y la unidad aritmético-lógica (ALU).



- BUS: sistema digital que transfiere datos entre los componentes de una computadora. Es un medio de transmisión compartido. Se pueden usar varias líneas del bus para transmitir dígitos binarios (1 o 0, cada una) simultáneamente (en paralelo). Por ejemplo: un dato de 8 bits, es decir, 1 byte, puede transmitirse mediante ocho líneas del bus.

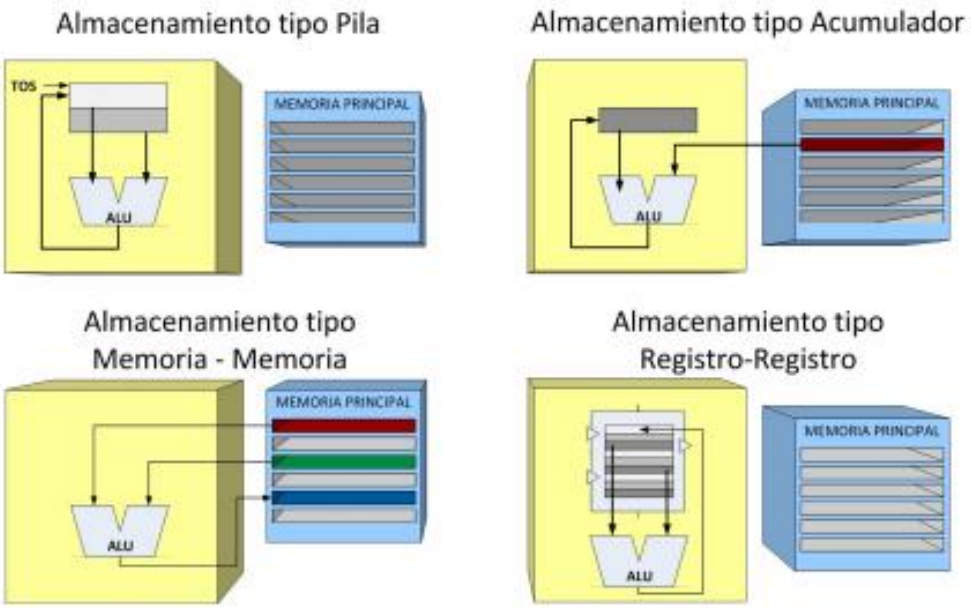
- Instrucción: es una única operación de un procesador definida por un conjunto de instrucciones de una arquitectura. Es interpretada por una parte del hardware (intérprete de instrucciones) que genera señales de control que la máquina puede ejecutar a partir de la instrucción.

  Las instrucciones son elementos AUTOCONTENIDOS, es decir, contienen toda la información necesaria para que la maquina sea capaz de tomarla, procesarla (cumplir las órdenes) y continuar con la siguiente instrucción.

  En la Arquitectura Von Neumann las instrucciones de un programa que respondan a lo que conocemos como algoritmo tienen que estar almacenadas en la memoria. El funcionamiento de la CPU está determinado por las instrucciones que ejecuta, que se denominan instrucciones maquina o del computador. Al conjunto completo de instrucciones distintas que puede ejecutar la CPU se le denomina repertorio de instrucciones. Cada una está subdividida en conjuntos de bits donde se definen:

  - Código de operación (COD OP): define a la ALU que operación tiene que realizar y sabe cuántos operandos (fuentes) necesita y donde poner el resultado.
  - Referencia a operandos fuentes: operandos a procesar para generar un resultado, datos que deben estar en la misma memoria del programa con las instrucciones que implementan el algoritmo. Pueden almacenarse en la memoria ppal., virtual o caché, en un registro de la CPU o pueden obtenerse mediante un dispositivo de E/S.
  - Referencia al operando resultado: indica donde se almacenará el resultado obtenido (de la operación entre los datos) en memoria.

- Referencia a la siguiente instrucción: indicaciones a la UC para la búsqueda de la siguiente instrucción tras completarse la ejecución de la instrucción actual.



- Tipos de Instrucciones: Pueden ser para el procesamiento de datos (aritmético-lógicas), el almacenamiento de datos (instrucciones de memoria), la transferencia de datos (instrucciones de E/S), o de control (instrucciones de testeo y flujo de programa).

- Diseño de un conjunto de instrucciones
  - Tipos de operandos
    - Números: asociados estrictamente a datos, ya sea en punto fijo o en punto flotante.
    - Direcciones: número sin signo que nos indica una celda o lugar de almacenamiento donde podremos almacenar o rescatar la información en ella almacenada.
    - Lógicos: se considera una unidad de n bits como n elementos de 1 bit, donde cada elemento toma el valor 1 o 0; flags o indicadores.
    - Caracteres: codificaciones de grupos o secuencias de bits que representan determinados elementos (ASCII, elemento codificador de patrones binarios referido a caracteres).

→ **Orden de los bytes:**

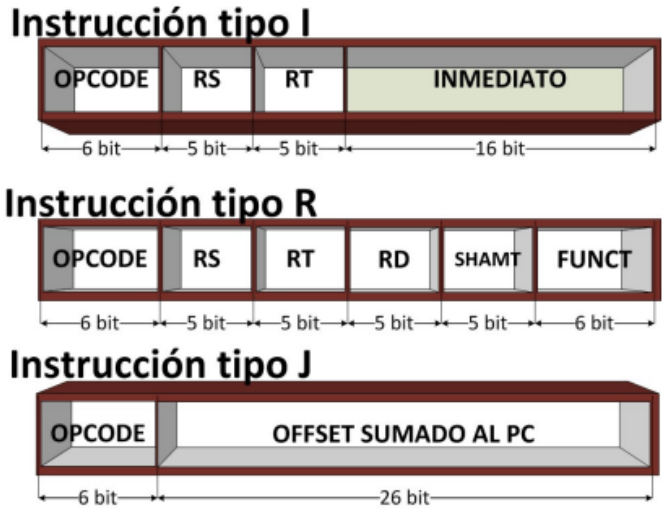
- BIG ENDIAN: el byte más significativo en la dirección con valor numérico más bajo.
- LITTLE ENDIAN: el byte menos significativo en la dirección con valor numérico más bajo.

↳ Repertorio de operaciones (implica saber cuántas operaciones se considerarán, cuáles y cuán complejas son).

→ Tipos de operaciones

- Transferencia de datos: debe especificarse ubicación del operando fuente y operando destino (podría ser de memoria, un registro o la cabecera de la pila), tamaño de los datos y modo de direccionamiento.
- Aritméticas: Add, Sub, Mul, Div, Inc, Dec, Neg, Abs, Shift Left/Right.
- Lógicas/Conversión: operaciones que manipulan bits individualmente, ya sean booleanas o rotate L/R. Operaciones para cambiar formatos de datos.
- E/S: IN, OUT, MOVE (realizadas por instrucciones de movimiento de datos), controlador DMA (direct memory access).
- Control de flujo: modifican el valor contenido en el registro PC, cambiando la secuencia de ejecución de instrucciones (por ejemplo: JMP, JZ, CALL, RET).

↳ Formatos de instrucciones (longitud de instrucción, número de direcciones, tamaño de los campos).



En el MSX88 tenemos instrucciones de 32 bits.

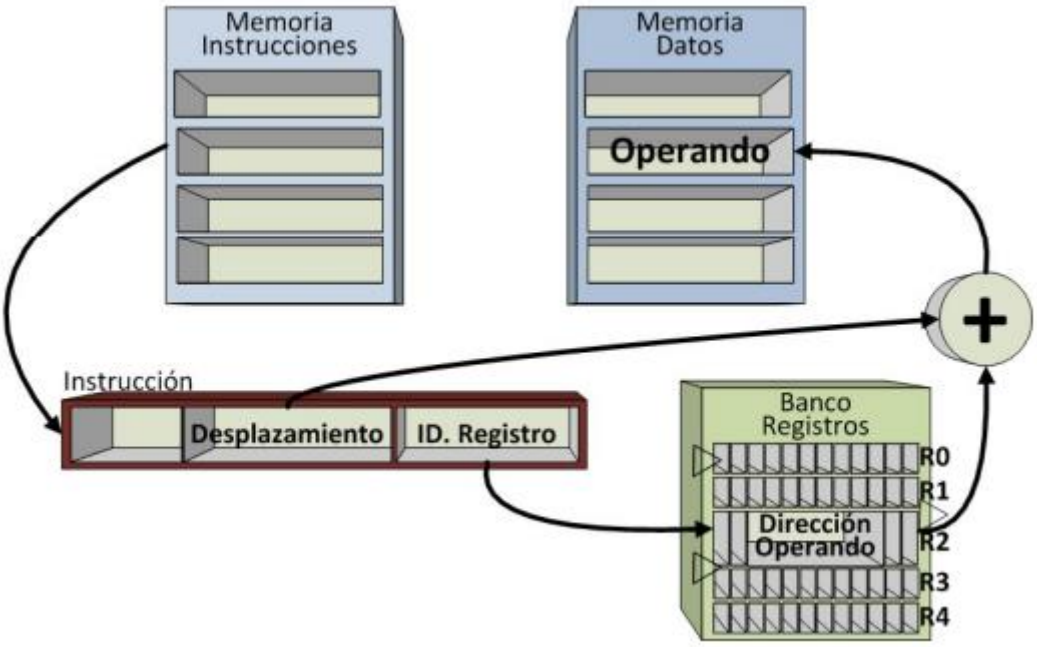
- De tipo I: operando en el valor inmediato de la instrucción, RS registro fuente (operando/sobreescritura de resultado), RT registro temporario (escritura de resultado).
- De tipo R: utiliza 3 registros; máquina de 3 direcciones. Operaciones de procesamiento de datos. FUNCT define que operación hago. SHAMT: shift amount.
- De tipo J: para instrucciones de instrucciones, OPCODE modifica la secuencia original del contador del programa, la sig. instrucción a ejecutar no es la siguiente línea, OFFSET sumado al program counter.

↪ Registros (número de registros referenciables, y categorización por tipo de operación que puedo ejecutar en cada uno, AX, BX, CX, DX siendo accumulator, base, counter y data respectivamente).

↪ **Modos de direccionamiento** (son maneras de explicitar los lugares relacionados con las direcciones de memoria. Informa la dirección de los operandos a usar en una determinada operación que utiliza un determinado formato de instrucción e inclusive requiere determinados registros dentro de la CPU).

- Inmediato: la instrucción contiene el valor del dato que la máquina tiene que utilizar, el valor del operando está escrito en la instrucción. Se utiliza para entregar valores constantes.
- Directo de memoria o Absoluto: contienen en la instrucción una descripción explícita de la dirección de memoria donde está el operando requerido. Debe hacerse un acceso a la memoria.
- Directo de registro: el campo de la instrucción que tomamos de la memoria de instrucciones hace referencia a un operando que está en el banco de registros (un conjunto de registros que se encuentra dentro de la CPU). No debe salir de la CPU, busca la información en el banco de registros interno a ella.
- Indirecto de memoria: en desuso, ir a memoria a buscar la dirección de memoria para buscar el operando, doble acceso a memoria.

- Indirecto con registro: acceder al registro a buscar la dirección de memoria donde está el operando, acceso al registro y luego a memoria.
- Indirecto con desplazamiento: la dirección de memoria donde se encuentra el operando que voy a buscar se obtiene por uso de registros del banco y por uso de la ALU con la operación de suma. En la instrucción encuentro un identificador que registro que contiene una dirección, y un valor numérico llamado desplazamiento (tipo inmediato) que se van a sumar para dar como resultado la *dirección efectiva* de memoria donde se encuentra el operando. Es indirecto con registro más el desplazamiento que le agrego, pila (relativo al stack pointer).



Pregunta de FINAL = En una instrucción de MSX88 ¿Cuántos modos de direccionamiento hay? Hay un modo de direccionamiento por cada operando. OJO con la limitación memoria/memoria! ¡No se puede!

RISC (Reduced Instruction Set Computer)

<>

CISC (Complex Instruction Set Computer)

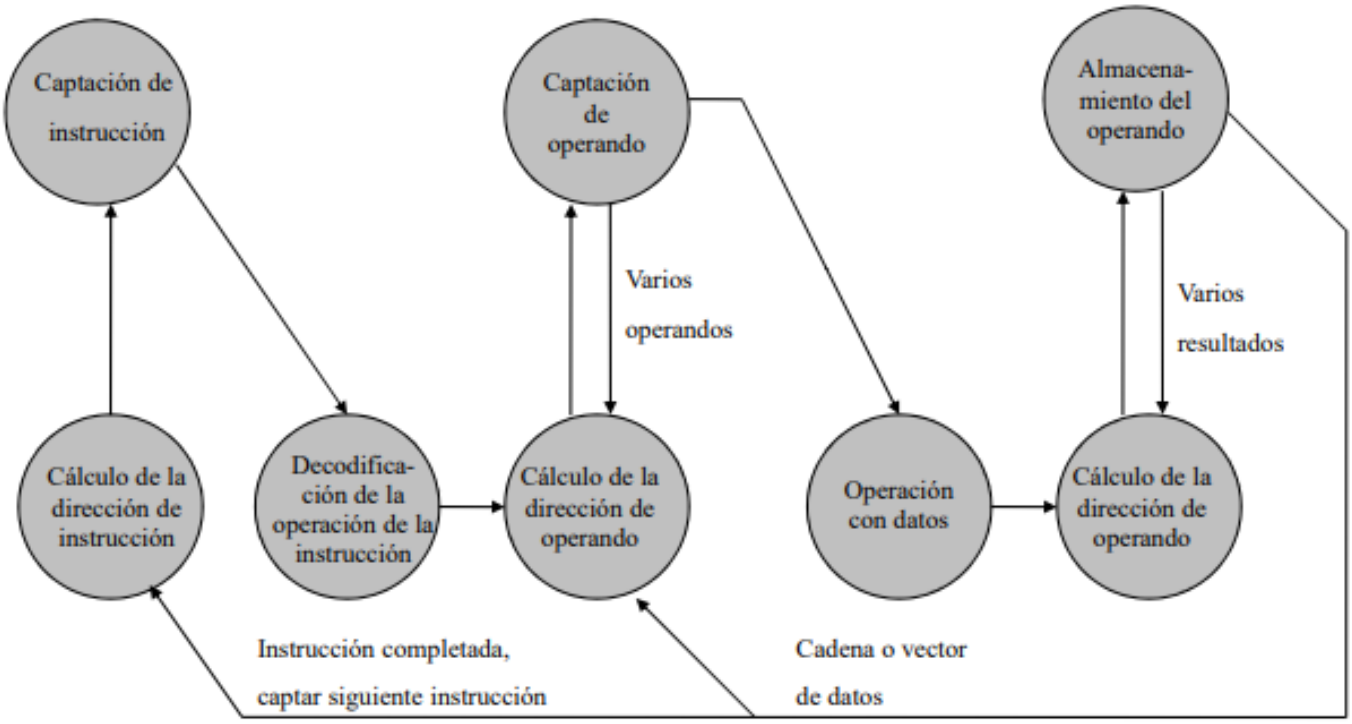
- Máquina de 4 direcciones: es únicamente teórica, sus direcciones hacen referencia a memoria, y la instrucción está en una dirección de memoria.

  No tiene como obligación la escritura secuencial de los pasos ya que en cada instrucción está contenida la siguiente a procesar. HALT solo tiene cód.. de operación y detiene la ejecución del programa.

  Los códigos de operación tienen 8 bits, 256 operaciones diferentes. Direcciones de 32 bits y los datos de 4 bytes (32 bits). Son muchos bytes destinados a las direcciones, de ahí surge el escribir secuencial para ahorrar el campo que define la siguiente.
- Surge la máquina de 3 direcciones: que tiene un registro interno que sigue el orden de la siguiente instrucción a ejecutar el program counter (PC). Hacer una celda de 13 bytes sigue siendo muy grande por lo que se toma la decisión de sobrescribir el resultado en un operando.
- Surge la máquina de 2 direcciones: puede ser que el resultado se escriba sobre el primer operando o sobre el segundo, depende de la máquina. La instrucción termina con 9 bytes de longitud y los datos tienen 4 bytes.
- La máquina de 1 dirección: IAS, trabaja con código de operación y una referencia a un operando, PC para seguir la secuencia de instrucciones, además de un registro acumulador que contiene un operando y recibe todos los resultados de la operación. LOAD lo lleva al acumulador, STORE almacena el valor del acumulador en la dirección que le asigno. Las instrucciones son de 5 bytes.
- Máquina de 0 direcciones: polaco inversa. Primero pongo los datos y luego las operaciones. Tengo registro pc, pila y nuevas operaciones. PUSH toma algo de memoria y lo apila. POP desapila y coloca en memoria. Las instrucciones son de 1 byte. Tengo una memoria y un registro acumulador que me permite hacer operaciones con 2 operandos y acumular el resultado.

- Condicionan la forma de programar y el hardware a construir.

- Ciclo de instrucción: compuesto por el ciclo de captación y el ciclo de ejecución. El ciclo de captación establece todas las operaciones que hay que hacer para que la unidad de control (UC) permita que la CPU vaya a una celda de memoria a buscar el patrón de bits que representa la instrucción y la guarde en la CPU misma. Luego, el ciclo de ejecución realiza los pasos necesarios para completar lo que la instrucción pide en su código de operación, utilizando los operandos que posee y dejando el resultado en el lugar en donde la misma instrucción establece.

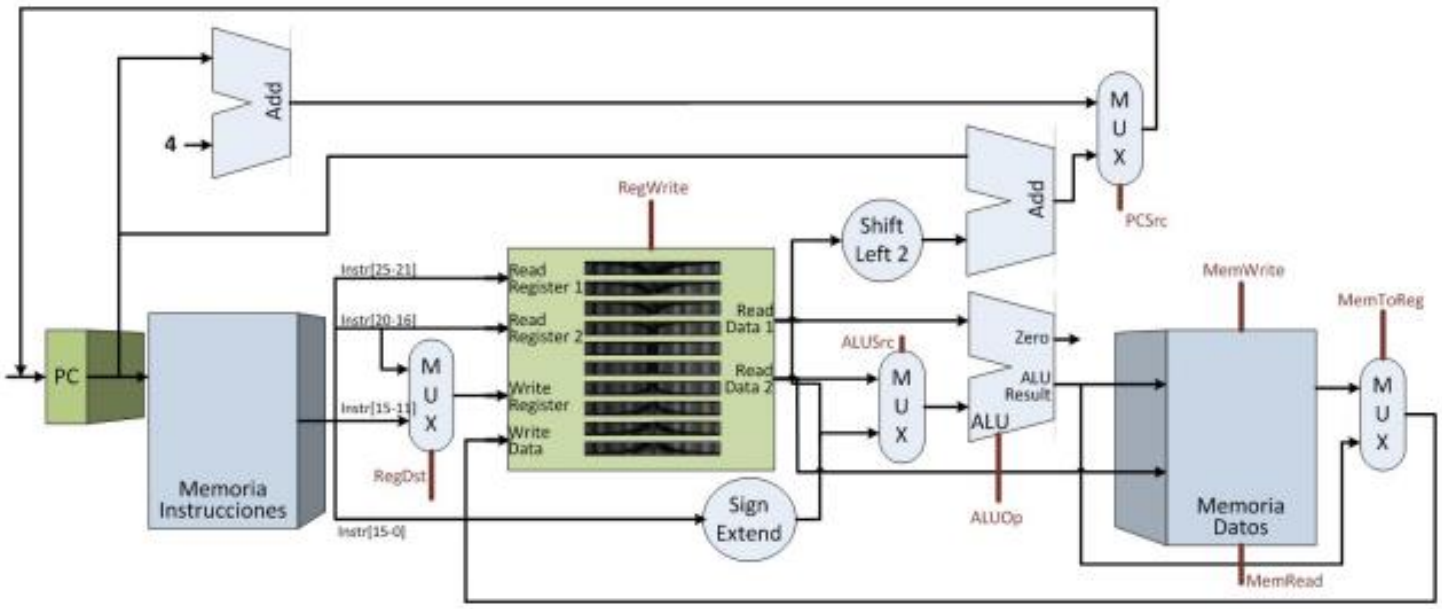


El *diagrama de estados* manifiesta la secuencia de pasos a realizar para poder concretar/cumplir una única instrucción. Es el algoritmo que realiza la unidad de control (UC) para poder ejecutar una instrucción. Ejecución: uso de la ALU para concretar una operación definida.

La máquina siempre está en algún punto del ciclo. Todas las instrucciones definidas en el repertorio de instrucciones tienen que pasar/cumplir estos estados del ciclo de instrucción.

Comienza por el cálculo de la dirección de instrucción, sigue por la captación..., operación de los datos en la ALU (ejecución), culmina buscando la siguiente instrucción.

Ruta de Datos



Muestra la evolución del ciclo de instrucción a través de poner algo de izquierda a derecha comenzando en el PC a través del cual, a medida de que trascurre el tiempo, pasaremos por la memoria de instrucciones y como contenido de la celda de memoria apuntada por el PC va a salir un patrón binario que representa una instrucción, esa instrucción tiene campos y cada uno va a salir por las diferentes líneas horizontales a la derecha de la memoria de instrucciones que pueden referirse al banco de registros de los cuales podremos sacar información.

De allí, la información irá a distintos posibles lugares receptores, en particular la ALU, que tiene entrada para dos operandos y una única salida a su izquierda, realizada una operación en la ALU a través de elementos del banco de registros, el resultado se puede almacenar en la memoria de datos o, a través del último multiplexor (MUX) tomo un camino que vuelve a ingresar al banco de registros. Con el multiplexor culmina la ruta de datos.

Subrutinas: son programas autocontenidos, algoritmos efectivos para la realización de una tarea en particular. Brindan economía (reutilización del código) y modularidad (subdivisión en unidades pequeñas). Se invocan desde cualquier punto del prog. ppal. con la instrucción CALL (Jump To Subrutine, Jump With Return, JSR).

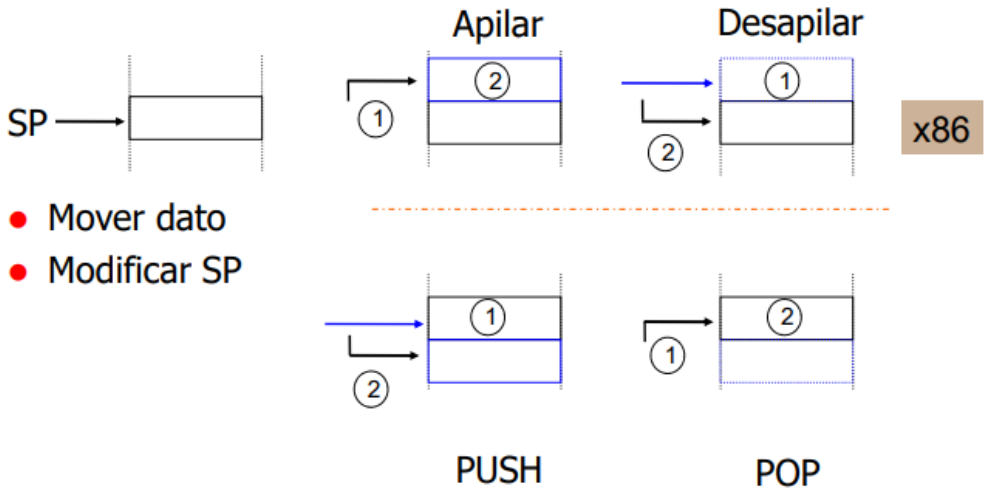
Requieren del pasaje de argumentos/parámetros: por valor (copia de la variable) o por referencia (dirección de la variable). El pasaje de argumentos debe referenciar el lugar físico donde el elemento va a estar disponible para

la subrutina. Se declaran antes del programa ppal. El pasaje de parámetros puede ser:

- Vía memoria: no presenta limitaciones con respecto a la cantidad de argumentos, se utiliza un área definida (RAM), es necesaria la documentación, presenta problemas de estandarización debido a las magnitudes de esa zona de memoria.
- Vía registros: limitación en la cantidad de registros, importante la documentación del usaje.
- Vía pila: método de almacenamiento de valores en una zona de memoria, dinámico, por demanda, que requiere un puntero de acceso, que indica el último elemento utilizado (SP), con mecanismo LIFO.
  Mediante ella se intercambiará la información y se pasarán los argumentos. Es independiente de la memoria, por el tamaño, al ser por demanda y no hace referencia a ningún registro. Método más usado.
  El Stack Pointer (SP) es un registro puntero que contiene la dirección del tope de la pila. Uno de los operandos está *de forma implícita* (apuntado por el SP) en la cabeza/tope de pila. Las operaciones PUSH y POP utilizan el puntero de pila y apilarán o desapilarán respectivamente el elemento ingresado en la instrucción.

  → **Funcionamiento de una pila:** es una secuencia de dos acciones:

  1) Movimiento de datos de un registro a un lugar de memoria indicado por el SP o al revés, de memoria a un registro.
  2) *Automodificación* del SP antes/después de 1) para poder continuar con las operaciones siguientes. Es la capacidad de incrementarse o decrementarse en uno o dos valores cuando se ponen o sacan elementos.



Assembly no diferencia procedimientos y funciones más allá de la instrucción de retorno. Subrutina es el rótulo de una dirección de memoria. Salto con retorno (CALL): utiliza la pila por sí misma, toma el valor de la dirección del rotulo y la pone en el program counter, pero antes ubica la dirección anterior en la pila para guardarla y poder accederla nuevamente. RET saca de la pila la dirección de retorno y la coloca en el PC.

- **Interrupciones:** mecanismo mediante el cual se puede cambiar el procesamiento normal (ejecución secuencial de instrucciones) de la CPU. Corta el lazo secuencial para cumplir con otra serie de instrucciones que tienen algún fundamento. LAS INTERRUPCIONES SON RUTINAS. Pueden ser de origen interno o externo a la CPU. Por resultado de una ejecución de una instrucción, por temporizador interno del procesador (WatchClock), por operación de E/S, por fallo del hardware.
- Gestor de Interrupciones: respuesta a una necesidad inmediata. Hay un gestor por cada tipo de interrupción que existe donde se define la respuesta a la solicitud de interrupción que se pueda generar. Tengo que salvar el *"estado del procesador"* (Contexto: próxima instrucción a ejecutar y el estado del procesador. Reflejo mínimo, tener una idea general del estado, no se guardan todos los registros, se guardan los flags, también el PC y se hace por pila) antes de transferir el control al programa gestor.
  La función del Gestor de Interrupciones es:
  - Corregir (o responder a) la causa que ocasionó la solicitud de interrupción.
  - Guardar/restaurar si usa/modifica valores en registros.
  - Retornar a la ejecución normal del programa interrumpido.

  IRET retorno de interrupción, saca de la pila el *estado de procesador* y la dirección de retorno, para volver al programa. Retorno a la **siguiente instrucción a ejecutar del programa interrumpido** como si fuese el retorno de una rutina o procedimiento.
- Tipos de Interrupciones
  - No enmascarables: no se pueden ignorar, son esenciales/ de alta prioridad para el funcionamiento del programa. Requieren respuesta eficiente y rápida.

- Enmascarables: pueden ser ignoradas, sus eventos no configuran peligro o pueden esperar, la solicitud de interrupción puede inhibirse con instrucciones especiales.

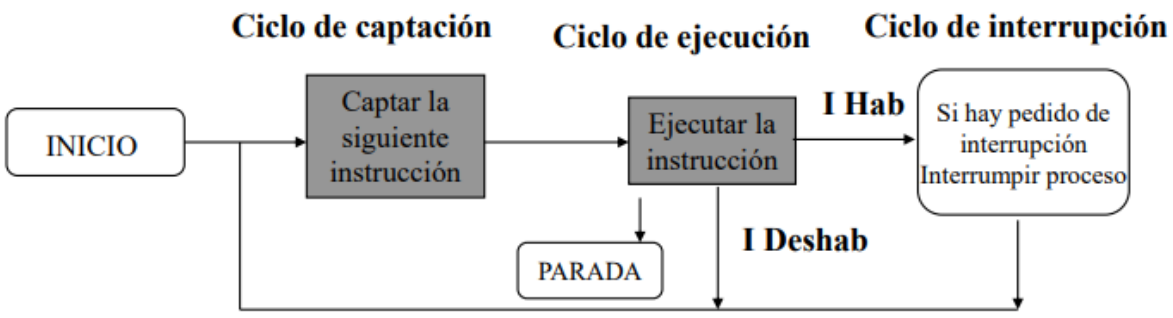
➥Tipos de Interrupciones: clasificación por origen

- Por hardware (*interrupt request*): generadas por dispositivos de E/S, son asíncronas al programa, eventos no planeados, no están relacionadas con el proceso de ejecución en ese momento.

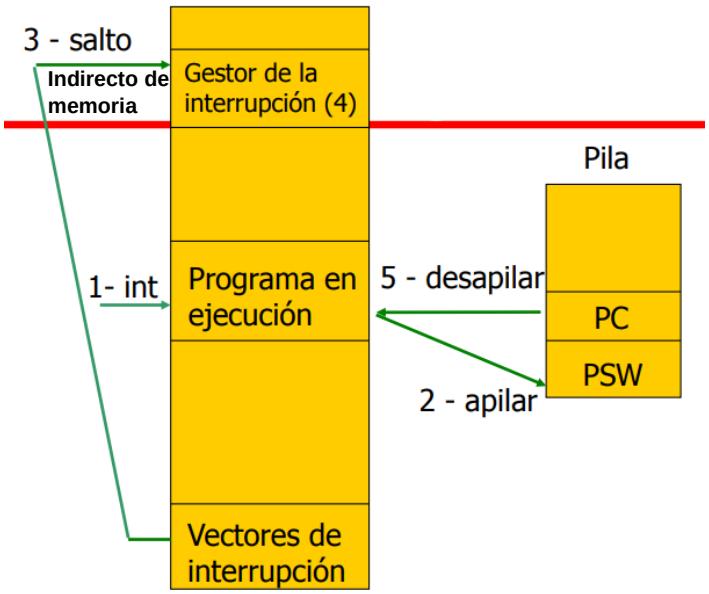
  ↘ Traps/Excepciones: interrupciones por hardware creadas por el procesador moderno en respuesta a ciertos eventos internos: condiciones excepcionales (overflow en ALU de punto flotante), falla de programa (instrucciones no definidas), fallas de hardware (error de paridad de memoria), accesos no alineados o zonas de memoria protegidas. Son del estilo hardware, pero no pertenecen a dispositivos externos, sino que son internas.

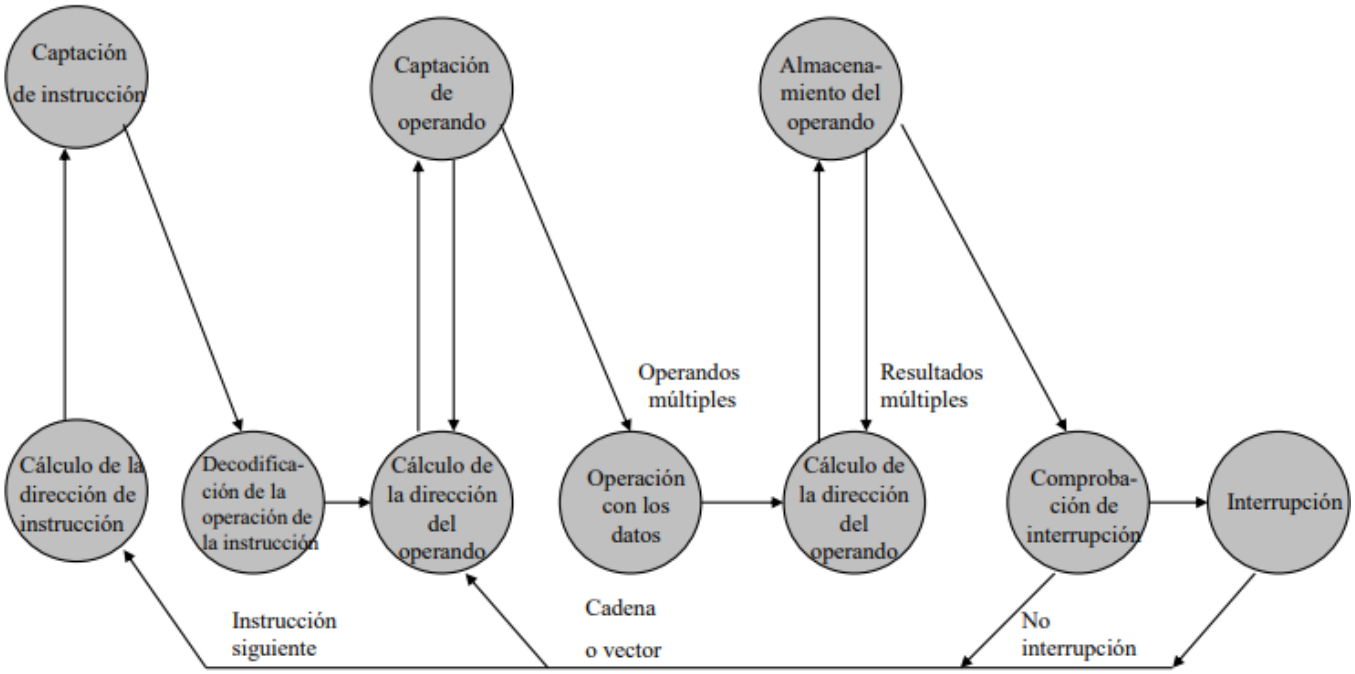
- Por software: son instrucciones explicitas de la máquina que afectan al procesador de la misma manera que las interrupciones por hardware. La diferencia es que ya no es asincrónico, dan como resultado de su ejecución por el ciclo de instrucción la respuesta que generaría la solicitud de interrupción. Permiten depurar los gestores de interrupción, ejecutándolos sin que el origen real de la interrupción se produzca. No requieren conocer la dirección de la rutina en tiempo de ejecución. Tienen la confianza de que la respuesta que se da es la misma que la que se da con las interrupciones por hardware.

➥Ciclo de Interrupción: al finalizar el ciclo de ejecución pregunto si hay una solicitud de interrupción (flag de pedido de interrupción), si es que están habilitadas. De ser así se ejecuta el ciclo de interrupción y se ejecuta su respuesta a esa interrupción. Al ejecutar el ciclo se inhiben los demás pedidos de interrupción, se ejecuta uno por vez. Si no hay señal se capta la siguiente instrucción.



- Vectores de Interrupción: identifican para cada número de vector una dirección donde está el programa de gestor de interrupción asociado a la interrupción. Se hace por salto indirecto vía memoria. PSW (Processor Status Word).





- Vectores de Interrupción, ¿cómo se arma?

  Es el nexo entre tipo de interrupción (0..255) y el procedimiento designado para atenderla. Cada entrada es una doble palabra (4 bytes, los 2 superiores son cero y los 2 inferiores son la dirección lógica/física).

  Vectores preasignados (gestores) MEMORIA DE LA 0 A LA 1023.

  - Tipo 0 – finaliza ejecución de programa (INT 00 alternativa a HLT)
  - 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.

- Ambiente de Interrupciones múltiples
  - Con interrupciones inhabilitadas: El procesador puede y debe ignorar la señal de petición de interrupción si se produce una interrupción en ese momento. Si se hubiera generado una interrupción se mantiene pendiente y se examinará luego una vez que se hayan habilitado nuevamente.
  - Con interrupciones habilitadas: Ante una solicitud, se inhabilitan. Se gestiona la misma y luego se habilitan otra vez.

  La implementación de una cola de solicitudes me permite atender las solicitudes en **orden secuencial estricto.**

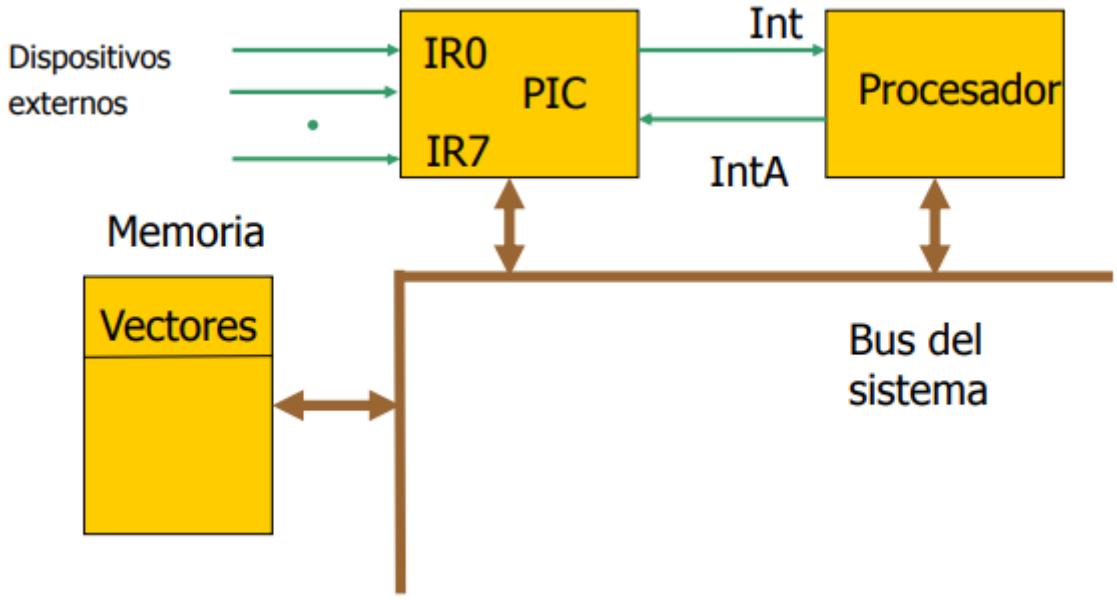
  - Con prioridades: Una interrupción de prioridad más alta puede interrumpir a un gestor de interrupción de prioridad menor. Cuando se termina la gestión de interrupción de mayor prioridad continua con la gestión de menor jerarquía. Las interrupciones se **gestionan**/manejan **anidando gestores**.

  De acuerdo con cómo se manejen las interrupciones es como se modifica la pila, que tiene guardados los estados de nuestras máquinas, a medida que van pasando las interrupciones.

- Reconocimiento de Solicitudes de Interrupciones
  - Interrupciones multinivel: puede ser fácil de implementar físicamente pero muy caro ya que por cada dispositivo que puede provocar una interrupción hay una entrada física de interrupción conectada a la CPU. Cualquier periférico puede ser un generador de solicitudes de interrupción por hardware.
  - Línea de interrupción única: una sola entrada física de pedido de interrupción a la que están conectados todos los dispositivos; se encuesta/pregunta a cada dispositivo si se ha producido el pedido de interrupción, hay varias técnicas de *polling*. **Enmascarables y No enmascarables tienen cada una su línea fija de interrupción. Separa los tipos de respuesta.**

- Interrupciones vectorizadas: se tiene un gran conjunto de elementos independientes que se pueden identificar, un vector que tiene tantos elementos como posibles solicitadores de pedidos de interrupción y va a poseer la dirección de comienzo de cada una de las rutinas de interrupción. Da lugar a 256 posibles pedidores de interrupción.
  El dispositivo que quiere interrumpir además de la señal de pedido de interrupción debe colocar en el bus de datos un identificador (vector, que identifica la fuente solicitadora de interrupción e informa cuál es el gestor que va a atender la solicitud). El valor del identificador lo coloca en el bus de datos el mismo periférico o un elemento interno a la CPU (Controlador de Interrupciones).

- **Controlador de interrupciones** (**PIC**, programmable interrupt controller): is an integrated circuit that helps a microprocessor (or CPU) handle interrupt requests coming from multiple different sources (like external I/O devices) which may occur simultaneously.
  Dispositivo que se conecta con el procesador a través de dos líneas: **Int** (línea fija de interrupción para el **pedido** de interrupción) e **IntA** (Internal Acknowledge o reconocimiento de pedido de interrupción, línea fija de **respuesta** al pedido de interrupción).
  Se relaciona con 8 registros IR (Interruption Request).



- Registros Internos PIC: usa 4 de los 8 posibles registros para las señales externas (0...3, por orden de prioridad el 0 con más, el 7 con menos) y los otros 4 son:

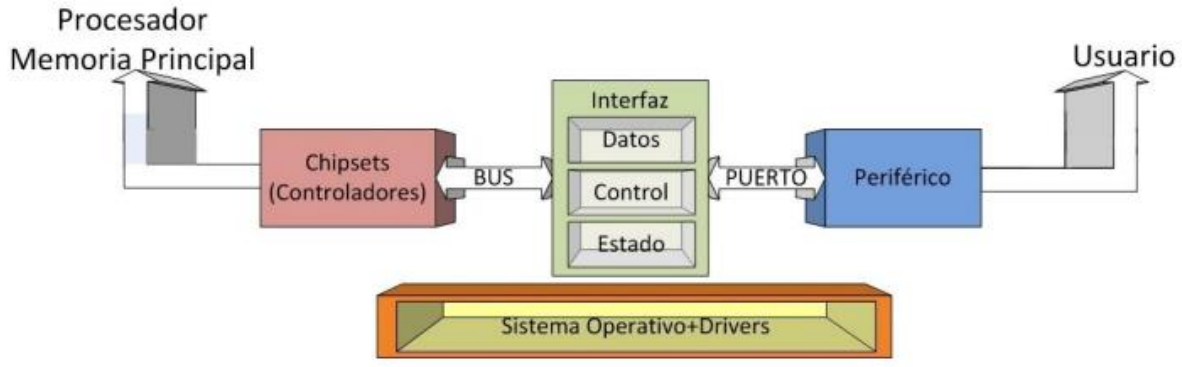
- EOI: para comandos, está en la dirección 20H, lo usamos para fin de rutina de interrupción.
- IMR: máscara, está en la dirección 21H, registro de 1 byte; por cada solicitud, cada bit puesto en 1 enmascara la solicitud, con cero desenmascaro la solicitud para que pueda ingresar.
- IRR: request, está en la dirección 22H, usado para petición de interrupción, se pone en uno el bit asociado a la solicitud de interrupción.
- ISR: service, está en la dirección 23H, indicador del solicitador que se está atendiendo, el del bit que pasó por prioridad, indica con el bit en 1.

↪ Conexionado y direccionamiento: Interrupciones hardware asignadas

- INT0 – Tecla F10
- INT1 – Timer
- INT2 – Handshake
- INT3 – DMA acceso directo a memoria
- INT4 a INT7 no usadas

↪ Entrada/Salida

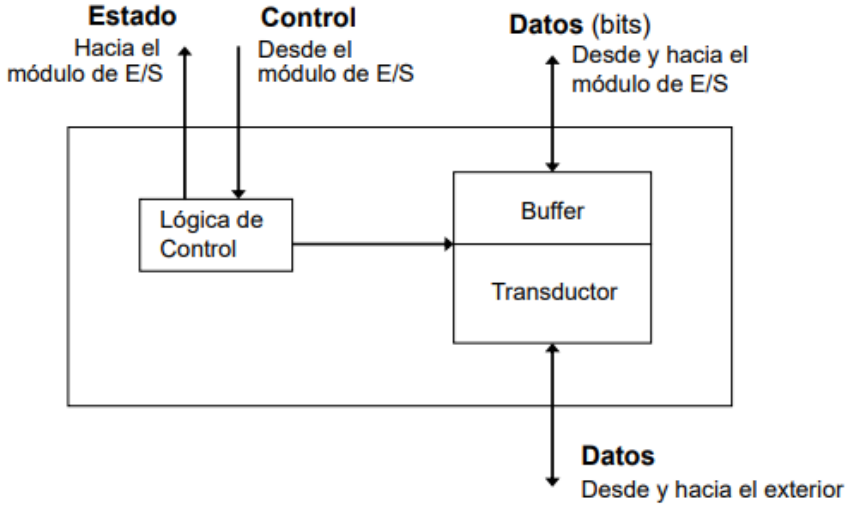
Gran variedad de periféricos con varios métodos de operación, más **lentos** que la CPU y la RAM, necesidad de módulos intermedios con "inteligencia" para administrar el ingreso a la CPU. El módulo de entrada salida realiza la interfaz entra el procesador y la memoria (bus) y los periféricos.



Interfaz: definición estructural de lo que ve el procesador para conectarse con el periférico que no ve. Está conectada al periférico por un puerto. 3 registros: datos, control y estado. El uso de la interfaz esta administrado por los drivers asociados a un sistema operativo.

Tipos de Dispositivos Externos

- E/S básicos: monitor/pantalla, mouse, teclado.
- Almacenamiento: disco duro, CD, DVD.
- Impresión: impresora, escáner.
- Comunicación con Dispositivos Remotos: modem, acceso/interfaz de red.
- Multimedia: micrófono, parlantes.
- Automatización y Control: sensores, alarmas, adquisición de datos.



Dispositivo externo tipo o general, que se encarga de la recepción de información. La parte superior son señales que ingresan al módulo de E/S cumpliendo determinadas características. Los datos son los caminos físicos por los cuales se va a realizar la transferencia de información en formato de bits. Esa información irá en un sentido o el otro dependiendo de **señales de control** (INPUT, OUTPUT) que envíe el módulo de E/S, quién también puede recibir **información del estado** (READY, NOT READY) del dispositivo externo.

El dispositivo externo posee una parte de lógica de control que puede recibir órdenes a través de las señales de control que envía el módulo de E/S o puede enviar información de estado del funcionamiento del dispositivo hacia el módulo de E/S. Mediante las **señales de control lógico** se administra el flujo de introducción (o expresión) de datos, en formato bit, que ingresan al módulo de E/S y que se relacionan con los datos del exterior. El proceso de adaptación se hace a través de un sistema de almacenamiento de bits a enviar al módulo o a recibir del módulo de E/S, llamado **buffer**; y un **transductor** que es el nexo entre la representación de datos interna y externa. La información se guarda en el buffer que contiene los datos que el transductor convirtió y que enviará a través de la señal del puerto a la CPU.

↪ Funciones de un Módulo de E/S

- Control y temporización de uno o más dispositivos externos
- Interpretar las órdenes que recibe de CPU y transmitirlas al periférico
- Comunicación con la CPU (registros) y Memoria con los datos del periférico
- Controlar las transferencias de datos entre CPU y el periférico (convertir formatos, adaptar velocidades)
- Comunicación con los dispositivos (periféricos)
- Informar a la CPU del estado del periférico
- Almacenamiento temporal (buffering) de datos del periférico hacia o desde la CPU.
- Detección de errores

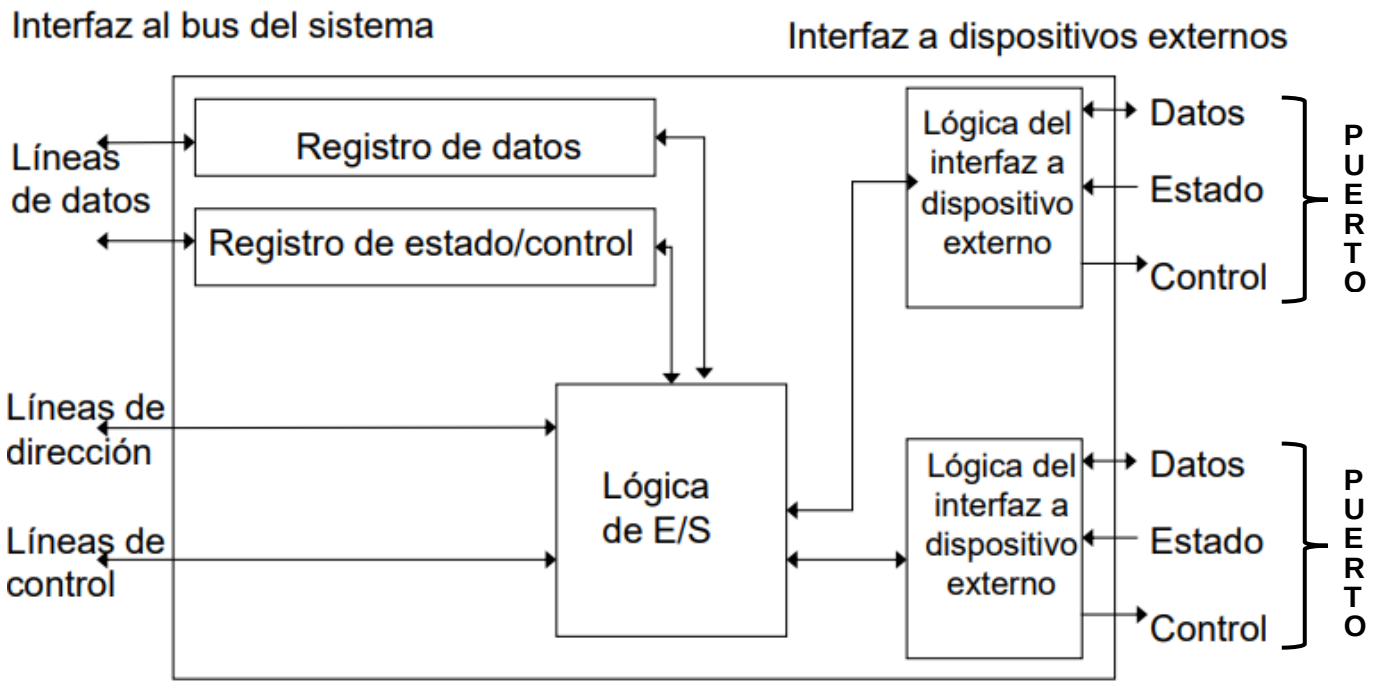


Diagrama en bloques de un módulo de entrada/salida.

↪ Capacidades de un módulo de E/S

- Ocultar las propiedades del dispositivo a la CPU
- Ocuparse de uno o varios dispositivos
- Controlar o no las funciones del dispositivo

  ↘ Canales de E/S o procesador de E/S (manejo de parte importante de la carga del procesamiento). Presentes en Mainframes.

  ↘ Controlador de E/S o controlador de dispositivo (manejo primitivo). Presentes en microcomputadoras.

Operación de Entrada o Salida

Toda operación ya sea de entrada o salida requiere de 3 pasos:

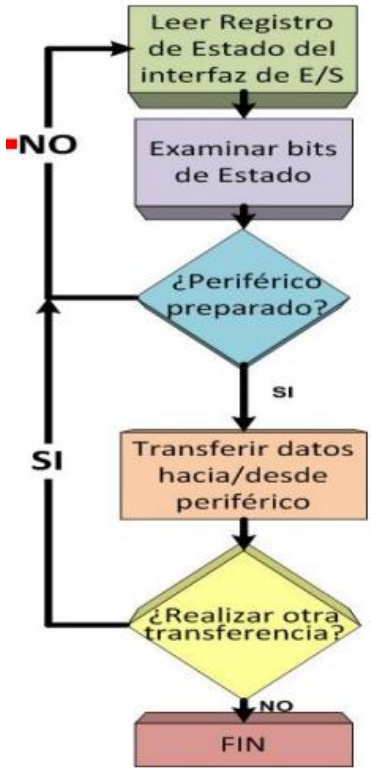
- Direccionamiento: es establecer con quién se quiere hacer la operación. Hay dos formas de analizar la manera de ubicar los dispositivos de E/S.
  - E/S mapeada en memoria (memory-mapped): típicamente usado por Motorola y Apple. La memoria ppal. y los dispositivos de E/S comparten un único espacio de direcciones. Es mucho más estricta la declaración de la representación de la memoria, ya que puede ser zona de datos de entrada, de salida o de ambas. La ventaja es que no hay ordenes o instrucciones específicas para E/S, entonces tengo variedad de órdenes de acceso a memoria (programación eficiente) ya que todas las operaciones que involucran una celda de memoria puede ser una celda de E/S. No hace diferenciación entre lectura/escritura de una celda de memoria y una E/S de un periférico.
  - E/S aislada (MSX88): todas las direcciones a las que hacemos referencia con respecto a módulos de E/S o que tengan que ver con periféricos como el controlador de interrupciones (PIC) tienen un espacio de direcciones separado de las direcciones de memoria, ésta forma de armar la conexión con el módulo de E/S necesita de la utilización de líneas especiales de E/S y de memoria. Tendremos líneas de control para dar órdenes específicas a E/S e inclusive instrucciones específicas (un conjunto limitado) para realizar las operaciones de E/S.
- Transferencia de información: consta de hacer una operación de lectura o escritura en función de si la operación es de entrada o de salida. Siempre visto desde la CPU.
- Gestión de la transferencia: cómo van a ser los mecanismos para sincronizar y controlar la transferencia correcta de los datos entre la CPU, o la memoria donde se encuentran los datos, y el periférico.

Técnicas de gestión de E/S

- E/S Programada con espera de respuesta: Se produce un intercambio de datos entre la CPU y el periférico a través del módulo de E/S. La CPU tiene control directo sobre la operación de E/S. Tiene 3 partes:

- **Comprobación del estado del dispositivo:** verificar que el dispositivo se encuentre en condiciones de recibir una tarea a realizar sea de lectura o escritura por parte de la CPU. Se hace a través del módulo de E/S pidiéndole a través de un comando que me diga cuál es el estado del periférico con el cuál quiero conectarme. Para eso el módulo de E/S se contactará con el periférico en cuestión a través del puerto correspondiente y recibirá y actualizará en el registro de estado del módulo de E/S la información del periférico.
- **Envío de comandos de lectura/escritura:** una vez verificado que el periférico está listo para realizar una trasferencia con la CPU, ésta envía un comando al módulo de E/S para que se ejecute la operación de lectura/escritura del periférico en cuestión.
- **Trasferencia de datos:** una vez que el módulo de E/S recibe el comando, lo implementará produciendo la transferencia de datos hacia el periférico o desde el periférico al módulo de E/S y colocándolo en el registro de estado que le da la CPU.

La CPU espera que el módulo de E/S termine la operación (esto sucede cuando se actualiza el registro de estado del módulo de E/S con el valor que la CPU esperaba), por lo tanto, permanece ociosa durante un período de tiempo (queda en un estado de verificación constante, no deseable).



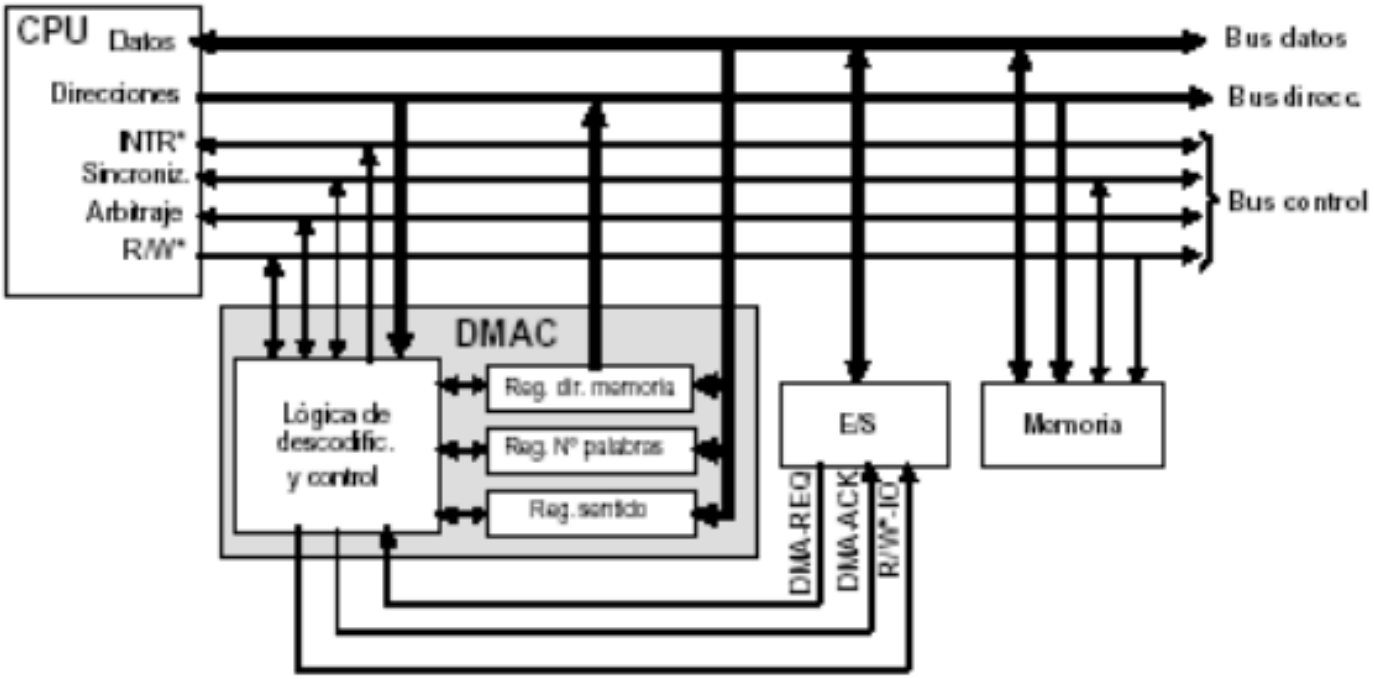
- La CPU solicita la operación de E/S al módulo.
- El módulo E/S realiza la operación.
- El módulo E/S activa los bits de estado del dispositivo direccionado y espera.
- La CPU comprueba periódicamente el estado de esos bits, hasta que detecta que la operación fue completada.
- En caso contrario la CPU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

- E/S con interrupciones: La diferencia con el caso programado es que vamos a programar solamente el comienzo de la operación y vamos a dar la orden de que el módulo de E/S trabaje con el periférico y que la CPU o el programa ppal. que se va a correr en la CPU pueda realizar alguna otra actividad mientras el periférico trabaja. Cuando éste termine de hacer lo que se le pidió, va a solicitar vía interrupción la continuación de la tarea u operación de E/S. El gestor de interrupción será quien administre el envío o recolección de datos con un periférico; su respuesta a la CPU determinará la continuación o culminación de una operación de E/S. La CPU no tiene que esperar la finalización de la tarea de E/S, puede seguir procesando. No se repite la comprobación de los estados de los módulos.
    - ¿Qué hace la CPU?: La CPU envía una orden de lectura (READ). El módulo E/S obtiene los datos del periférico mientras que la CPU realiza otro trabajo. La CPU chequea si hay pedidos de interrupciones pendientes al final de cada ciclo de instrucción. El módulo E/S emite un pedido de interrupción a la CPU. 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.

Si hacemos un análisis de estos dos tipos de operaciones de E/S, las efectuadas mediante interrupciones son más efectivas que las programadas, pero ambas necesitan la intervención directa de la CPU por lo que la velocidad de transferencia es limitada y la CPU permanece ocupada mucho tiempo durante la operación.

- E/S con acceso directo a memoria (DMA): la transferencia de datos entre la memoria y un dispositivo de E/S se hace sin intervención alguna de la CPU. Se incorpora un controlador para realizar esta tarea, que tiene que ser capaz de hacer uso del bus del sistema mediante señales y lógica de arbitraje necesarias para solicitar el uso del bus de datos para trasladar el dato en sí mismo; debe manejar o administrar el bus de direcciones para poder acceder a la memoria y leerla/escribirla para realizar la transferencia; debe generar señales de control del bus para definir si es lectura o escritura y sincronizar la transferencia con el módulo de E/S. DMA es un módulo de E/S en sí mismo ya que está

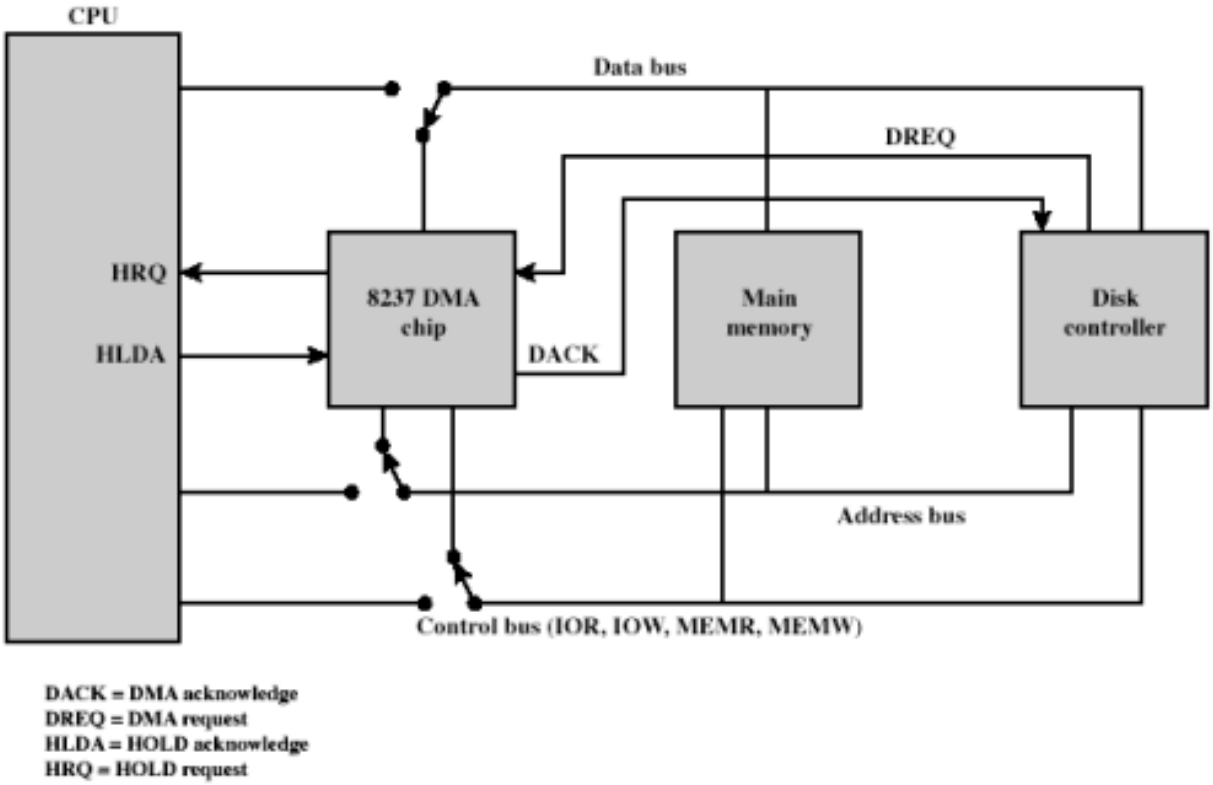
pensado inicialmente para periféricos que requieren alto volumen de transferencia de datos.



➥Etapas de una transferencia DMA

- **Inicialización de la transferencia**: La CPU debe enviar a la interfaz del periférico (Mód. E/S) y al DMAC los parámetros de la transferencia.
  Inicialización de la interfaz (Bus master: CPU - Bus slave: Interfaz)
  Envía Nº de bytes a transferir, tipo de transferencia (lectura/escritura), otra información de control (pista, sector, etc.).
  Inicialización del controlador DMA (Bus master: CPU - Bus slave: DMAC)
  Envía Nº de bytes o palabras a transferir, tipo de transferencia (lectura/escritura), dirección de memoria inicial para la transferencia, Nº de canal (para DMAs con varios canales).
  Después de la inicialización la CPU retorna a sus tareas y ya no se preocupa más de la evolución de la transferencia.
- **Realización de la transferencia**: Cuando el periférico está listo para realizar la transferencia se lo indica al DMAC (DMA request). El DMAC pide el control del bus y se realiza la transferencia entre el periférico y la memoria.
  Bus master: DMAC + Periférico - Bus slave: Memoria.
  Después de la transferencia de cada palabra se actualizan los registros del DMAC (Nº de bytes o palabras a transferir, dirección de memoria)

- **Finalización de la transferencia**: El DMAC libera el bus y devuelve el control a la CPU. El DMAC suele activar una señal de interrupción para indicar a la CPU la finalización de la operación de E/S solicitada.



Si la CPU no puede acceder (ya que no está conectada) al controlador ni a la memoria, se está degradando su rendimiento. El DMAC hace uso intensivo del bus, si este está ocupado en una trasferencia DMA, la CPU no puede acceder a memoria para leer instrucciones o datos.

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 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 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 para realizar sus transferencias.

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

Se distinguen dos tipos de transferencias: por ráfagas (burst) o por robo de ciclo (cycle-stealing).

- DMA modo ráfaga: 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 rápida.
  - 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.

  Si se sabe que la CPU no necesita acceder a la memoria, este modo de trasferencia no afecta notablemente el rendimiento de la computadora.

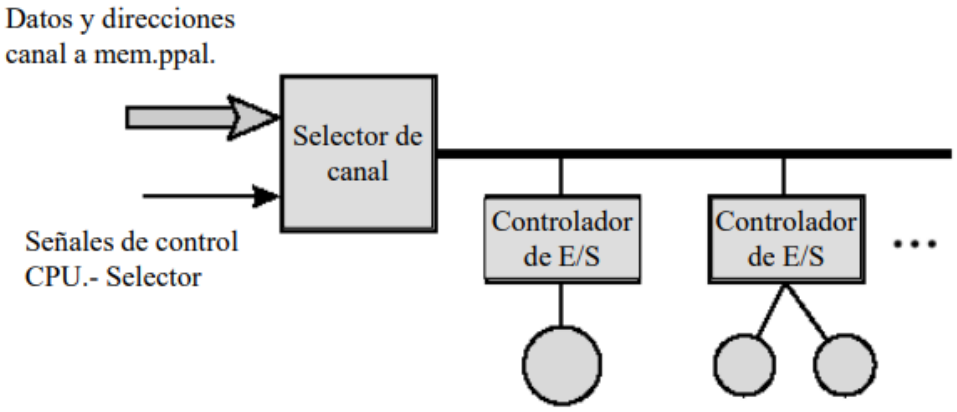
- DMA modo robo de ciclo: El DMAC solicita el control del bus a la CPU. Cuando la CPU concede el bus al DMAC, se realiza la transferencia de una única palabra y después el DMAC libera el bus. El DMAC solicita el control del bus tantas veces como sea necesario hasta finalizar la transferencia del bloque completo. Utiliza los ciclos de reloj en los que la CPU no ocupa el bus porque no necesita acceder a memoria.
  - VENTAJAS: No se degrada el rendimiento del sistema.
  - DESVENTAJAS: La transferencia tarda más tiempo en llevarse a cabo.

  Para la CPU no es una interrupción. El procesador no debe 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. Requiere alta sincronización del uso de bus ente el controlador de DMA y la CPU.

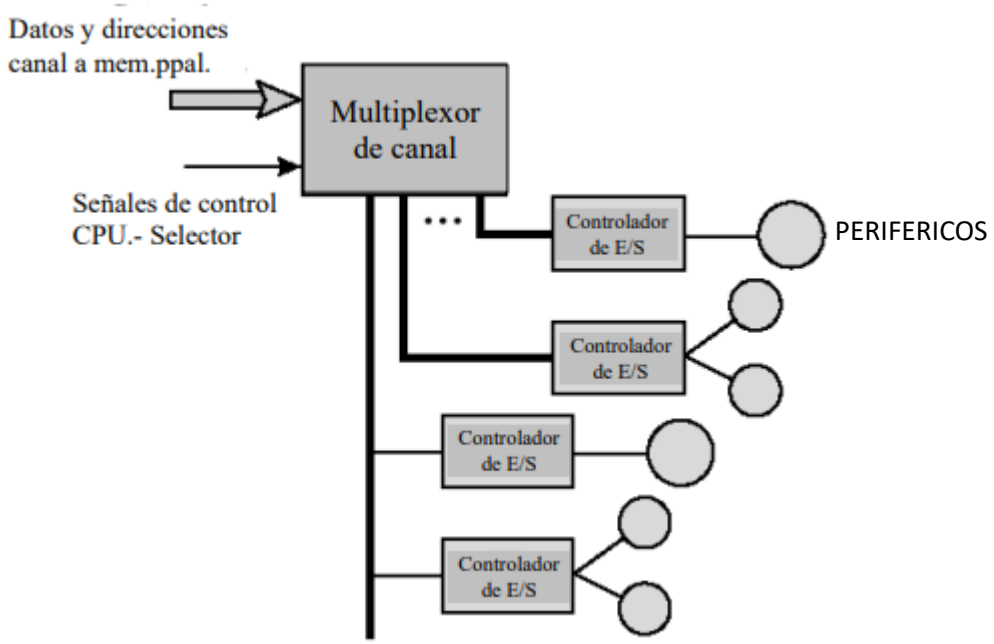
- Características de los Canales de E/S – Evolución
  1. La CPU controla directamente los periféricos.
  2. Se agrega un módulo de E/S o controlador.
  3. Idem 2 más llamado de interrupción.
  4. El módulo de E/S provee el acceso directo a memoria (DMA).
  5. El módulo de E/S tiene su propio procesador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

Representan una extensión al concepto de DMA, tienen la habilidad de ejecutar instrucciones de E/S. Tienen completo control de la transferencia de datos, por lo tanto, la CPU no ejecuta instrucciones de E/S. La CPU genera un programa almacenado en memoria principal, inicia la transferencia de E/S y ordena ejecutar el programa que está en memoria. El programa especifica dispositivos, áreas de memoria a usar, prioridades y acciones ante errores.

- Selector: controla varios dispositivos de alta velocidad y uno por vez, por lo tanto, el canal se dedica para la transferencia de datos de ese dispositivo. El canal selecciona un dispositivo y efectúa la transferencia. 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.



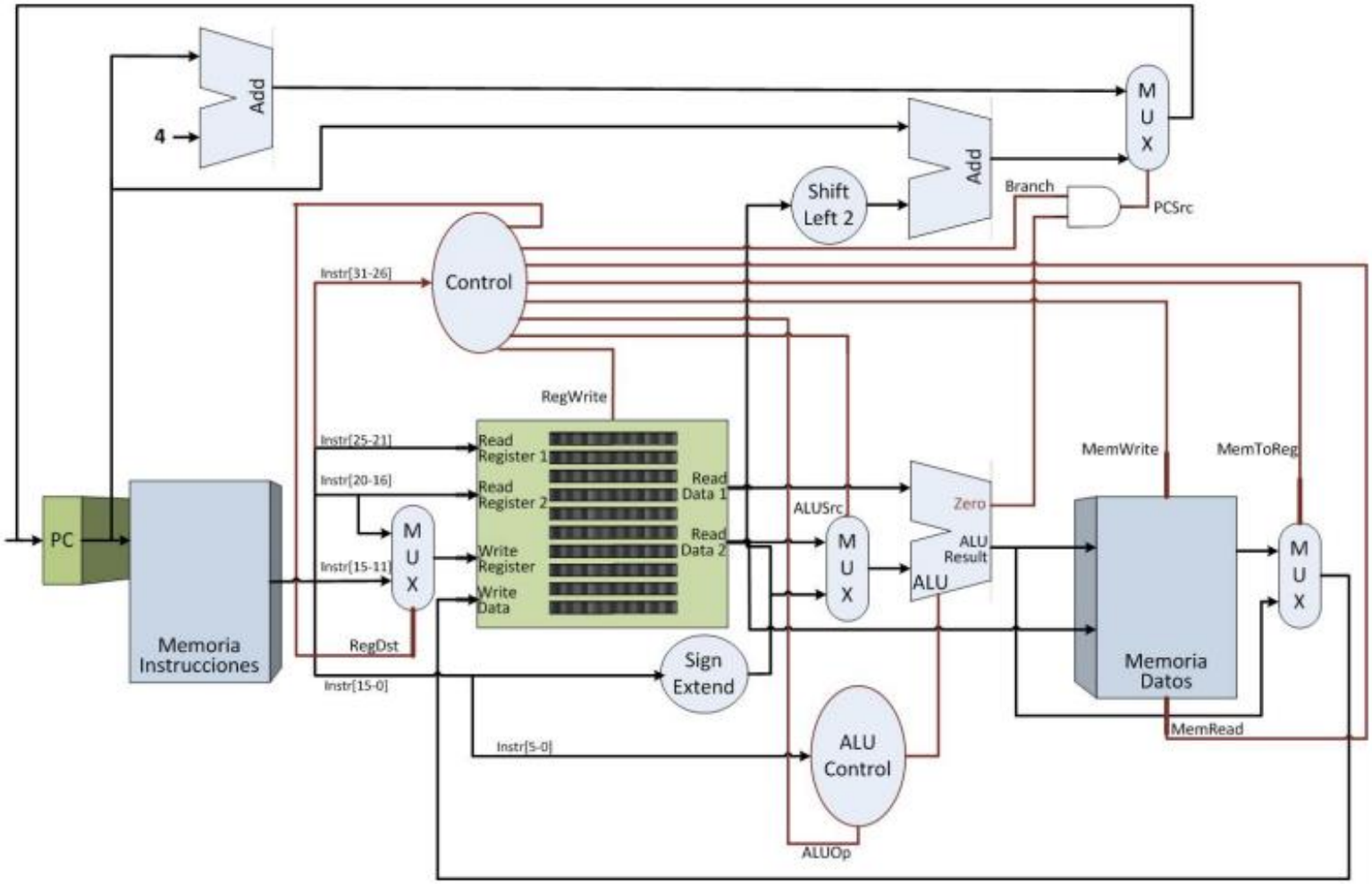
- Multiplexor: puede manejar E/S con varios dispositivos a la vez. El multiplexor de bytes acepta y transmite caracteres. El multiplexor de bloques intercala bloques de datos desde distintos dispositivos.



CLASE 4 02/05/2023 SEGMENTACIÓN DE INSTRUCCIONES

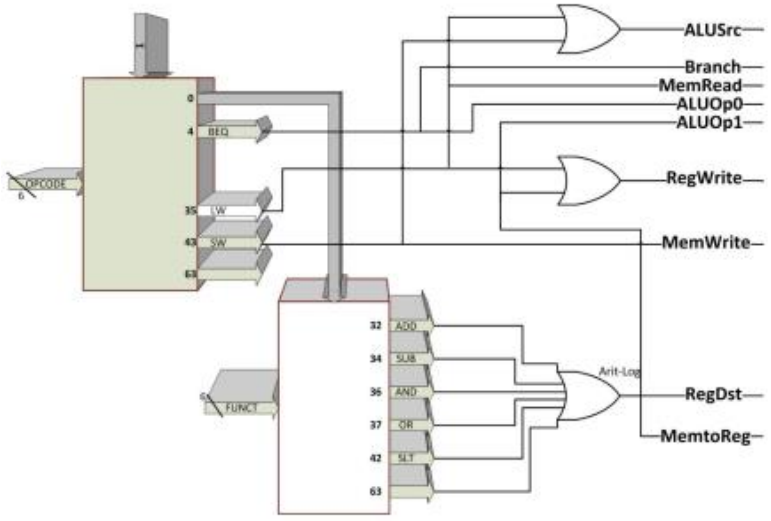
La mejora en el procesador viene de hacer un correcto análisis del ciclo de instrucción e implementarlo en hardware para sacarle el máximo provecho.
Análisis de tareas a realizar para cumplir con el ciclo, la primera columna es la de búsqueda de instrucción. La segunda columna se llama decodificación de instrucción y búsqueda de operandos, necesito saber qué operación quiero hacer y con que operandos. La tercera etapa es la de realizar la operación con los datos mediante la ALU, que genera un resultado que va a ser almacenado (Dirección o Indirección). Se puede guardar en la memoria ppal. o en los registros de la CPU.

Fetch, F (búsqueda) acceder a la memoria por la instrucción e incrementar el PC. Se puede hacer en simultáneo, traer la instrucción e incrementar el registro interno.
Decode, D (decodificación) Se accede al banco de registros por el/los operando/s (si es necesario) y se calcula el valor del operando inmediato con extensión de signo (se convierte a binario, también si hace falta).
Execute, X (ejecución) ejecución de la operación ene la ALU.
Memory Access, M (acceso a memoria) accedo a la memoria de ser necesario.
Writeback, W (almacenamiento) accedo al banco de registros para almacenar un resultado si es requerido.
En M y en W puedo acceder para hacer escritura. En general, accedo a memoria para buscar un dato y no para escribirlo. Si estoy en la segunda etapa y tengo que escribir en un registro, no puedo, tengo que esperar a la tercera etapa.
MUX (multiplexor) circuito combinatorio que permite de varias entradas posibles sacar una de ellas. Multiplexor 2 a 1 es el que se muestra en el gráfico de ruta de datos.
ADD son sumadores. Podemos tener operadores complejos u operadores simples.
Sign Extend: realiza la **extensión del signo** para poder colocar por ejemplo un valor de 4 (1000 = -8 en Ca2) bits en un registro de 16 bits, coloca los 12 bits a la izquierda en 1 y el número continúa siendo el mismo. Si tengo el 0100 = 4 en Ca2 agrego 12 ceros a la izquierda y se mantiene el número.
Shift Left 2: desplazamiento de dos lugares a la izquierda, agrego dos ceros a la derecha. 010000 = 16 en Ca2. Es el 4 que tenía originalmente multiplicado por 4, multiplico por 2 por cada lugar, cada cero.



Buscar bibliografía, diseño de arquitectura españoles.

Unidad de Control:



64 señales que salen de la ALU.

Instrucciones Mips:

SLT: SET IF LESS THAN

BEQ: BRUNCH IF EQUAL

RD: REGISTRO DESTINO

RS: REGISTRO FUENTE

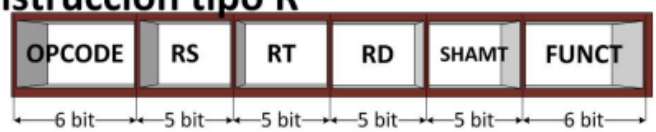
RT: REGISTRO TEMPORARIO

La mayoría usa modo de direccionamiento directo de registro.

LW usa modo de direccionamiento indirecto vía registro con desplazamiento.

Todas las instrucciones tienen 32 bits de largo y se dividen según la operación que quiero hacer. Son de tipo R en donde todos los operandos son registros.

Instrucción tipo R



Los tiempos del monociclo y multiciclo son similares, pero el multiciclo es más eficiente (pero no más rápido).

Comparación monociclo – multi ciclo: para el multi ciclo debo hacer modificaciones al hardware. Se optimiza el uso de tiempo y hardware para la ejecución.
NOP se utilizaba para calcular un tiempo a raíz de que la instrucción tiene un tiempo fijo y me puedo crear un timer.

Diagrama de estados del controlador
Lenguaje de transferencia de registros RTL
El load es la instrucción más lenta y la maquina se va a ajustar a esta instrucción. Load utiliza todos los recursos del hardware.

Pipelining/ segmentación de cauce: forma de organizar el hardware disponible para que se puedan realizar más de una tarea en simultáneo. Se descompone el fases o etapas. Paralelización de la ejecución de las instrucciones. Rompe con la secuencialidad de los algoritmos.
Características: La segmentación es una técnica de mejora de prestaciones a nivel de diseño hardware. El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones.
¿Qué es la segmentación? Separación de etapas mediante registros de segmentación en hardware, si no los tuviera sería procesador multi ciclo sin segmentación.

ATASCOS DEL CAUCE (STALL)
- ESTRUCTURALES
- POR DEPENDENCIA DE DATOS
- POR DEPENDENCIA DE CONTROL

Si resolvemos con paradas del cauce, disminuye el rendimiento teórico y el CPI>1.
Tengo una instrucción por cada uno de los ciclos de reloj de la máquina.
El rendimiento teórico es proporcional a la cantidad de etapas.

SOLUCIONES A RIESGOS ESTRUCTURALES
- Duplicación de recursos de hardware.
- Separación de memorias de instrucciones y datos.
- Turnar el acceso al banco de registros.

SOLUCIONES A RIESGOS DE DATOS

RAW: riesgos elementales, lectura después de una escritura.

Pueden ser **tratados** mediante:

HARDWARE → Modificación al hardware a partir de una unidad que detecta los riesgos. Adelantamiento de operandos (**forwarding**)

SOFTWARE → Tener un compilador más complejo que cuando hace la conversión del lenguaje fuente a lenguaje de maquina haga las debidas modificaciones para que no se produzcan los RAW. Instrucciones NOP o reordenamiento de código.

FORWARDING: Consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando. El dato no es guardado ni en un registro ni en memoria antes de ser pasado como operando, no es necesario esperar a que se almacene realmente.

Pares de instrucciones que tienen un dato en común y pueden presentar riesgos RAW, que soluciono con un forwarding:

LW R1, 100(R2)
SW R1, 0(R10)

LW R1, 100(R2)
ADD R3, R1, R8

LW R1,100(R2)
BEQ R1, R3,etiqueta

ADD R1, R2, R3
SW R1, 0(R10)

ADD R1, R2, R3
SUB R8, R1, R6

ADD R1, R2, R3
BEQ R1, R7, etiqueta

LW R1,0(R7)
LW R10, 100(R1)

ADD R1, R2, R3
LW R10, 100(R1)

1. Una vez que el registro es determinado por la instrucción de carga se pasa a la instrucción de escritura.

2. Suma de RI con otro Reg. Suma toma los operandos en la etapa de decodificación, pero el dato no esta y tengo que esperar un ciclo para repetir el intento de lectura. El dato está en el registro de segmentación.
Tomo el dato de R8 y del reg. de seg. tomo el RI y lo mando a ejecución. Pierdo un solo ciclo de reloj y no dos.

3. Tengo que esperar 3 ciclos de parada, pero con el forwarding pierdo solo 2.

En la mayoría de los casos el forwarding evita la dependencia de tipo RAW y si no lo elimina siempre lo disminuye.

RUTA DE DATOS CON LA UNIDAD DE ADELANTAMIENTO

Administra multiplexores que tienen tres posibles elementos a entrar en la ALU y una sola salida. La unidad es transparente al programador. Viene con el procesador y si no lo tiene lo único que tenemos es la solución por software para implementar.

TECNICA SOFWARE: evita los riesgos reordenando las instrucciones del código sin afectar los resultados Realizada por el compilador

Introducción de instrucciones NOP: se genera Retardo

Reordenación de instrucciones: Máxima separación de instrucciones con dependencia RAW. Cuidado con ejecución "fuera de orden" producida por la cantidad de separación que hay.

SOLUCIONES A RIESGOS DE CONTROL: son aquellas que NO podemos evitar. Siempre hay una parada de un ciclo como mínimo. Hay **Penalización por Salto.**

• Incondicional: La dirección de destino se debe determinar lo más pronto posible, dentro del cauce, para reducir la penalización.

• Condicional: Introduce riesgo adicional por la dependencia entre la condición de salto y el resultado de una instrucción previa. Es el peor caso.

Modificación sencilla para **reducir la cantidad de paradas a un solo ciclo**: adelantar la resolución de los saltos a la etapa D: se decodifica y se sabe que es un salto, s se evalúa la condición de salto y se calcula la dirección de salto. Pierdo el ciclo, pero es uno solo. Se agrega hardware, el restador y el sumador. Hace más trabajo usando más hardware.

Sabemos que ponemos una instrucción de salto y el siguiente ciclo de instrucción lo pierdo, pero lo utilizo para hacer algo que no implique a la ALU.

Se puede hacer por técnica de hardware: predicción de saltos para evitar la parada. Puede ejecutar algo, en el peor de los casos pierdo la ejecución porque no puedo llevarla a cabo y el el mejor de los casos gané un ciclo.

Técnicas estáticas

• Predecir que nunca se salta: Asume que el salto no se producirá. Siempre capta la siguiente instrucción.

• Predecir que siempre se salta: Asume que el salto se producirá. Siempre capta la instrucción destino del salto.

Si mi predicción no es correcta, tengo una parada y tengo que sobrescribir la instrucción.

O la técnica de software: salto retardado o de relleno de ranura de retardo. Slot. Puedo implementar una técnica que proponga una instrucción a ejecutar en cualquier caso después de la instrucción de salto. Lo utilizo de forma específica con algo que sé que se ejecuta y que me va a servir. Se generan modificaciones de hardware. La ALU es diferente. El salto se completa en el siguiente ciclo y aprovecho el tiempo muerto para ejecutar la ejecución del medio. El truco es ver que instrucción hago.

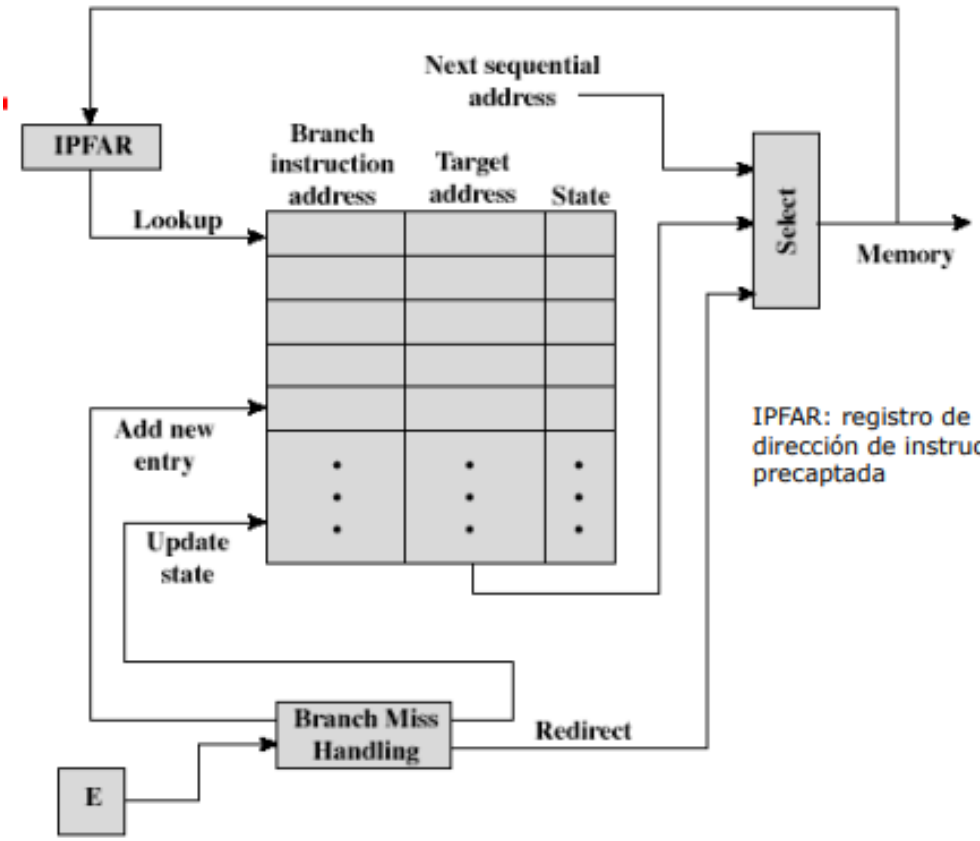
Técnicas dinámicas: me adapto al programa.

• Conmutador saltar/no saltar: Basado en la historia de las instrucciones. Si no acierto dos veces seguidas la predicción la cambio. Eficaz para los bucles.

• Tabla de historia de saltos (branch-target buffer): Que guarda la historia de los saltos previos, puedo repetirlo si estoy en un bucle. Pequeña cache asociada a la etapa de búsqueda (F). La caché tiene tres campos: Dirección de una instrucción de bifurcación; Información de la instrucción destino (dirección de la instrucción o la instrucción misma); N bits de estado (historia de uso). Tracking de ejecución de instrucciones. Se toman decisiones en función del uso.

//DIAGRAMA DE PREDICCIÓN SALTAR NO SALTAR// Funciona con pocos bucles, separados y no anidados.

TABLA DE HISTORIA DE DATOS BTB



En la etapa de búsqueda puedo saber si lo que voy a buscar es una instrucción de salto o no.

La primera vez "pierdo" pero luego puedo predecir. En esa primera vez pierdo el ciclo, por la penalización de salto.

SALTO RETARDADO El compilador trata de situar instrucciones útiles (que no dependan del salto) en los huecos de retardo (delay-slot). Si no es posible, se utilizan instrucciones NOP. Las instrucciones en los huecos de retardo de salto se captan siempre. Requiere el reordenamiento de instrucciones.

¡Las máquinas o tienen delay-slot o tienen BTB! ¡NO AMBAS!

Otras soluciones hardware

- Predecir según el código de operación: Hay instrucciones con más probabilidades de saltar. La tasa de acierto puede llegar a alcanzar un 75%
- Flujos múltiples
- Precaptar el destino del salto
- Buffer de bucles