Resumen de Arquitectura I

- Sistema de cómputo

Existen 3 subsistemas: CPU, Memoria y E/S, que están conectados por un bus, que es un dispositivo con un cierto grado de inteligencia.

La principal tarea de un sistema de cómputo es manipular datos para producir resultados.

Una *celda binaria* es una celda capaz de almacenar 1 bit de información. Tiene una entrada y una salida y se pueden hacer 2 operaciones básicas: leer y escribir. Para leer retiene la entrada y para escribir hace una copia de la entrada en la salida. Para su control se tienen señales de control: READ/WRITE define el tipo de operación que se va a realizar, y HABILITACIÓN nos dice cuándo se va a realizar.

Un *registro* es un arreglo unidimensional de celdas binarias.

- Memoria

A la memoria se la puede interpretar como un arreglo unidimensional de registros o como un arreglo bidimensional de celdas binarias. Se accede a un registro por vez. Se tienen 2 variables combinatorias como entrada que decodifican y que producen la dirección (a esto se lo llama *información de dirección*). Si tenemos m registros de n celdas cada uno, las variables deben ser de p bits, donde $2^p = m$. Luego p define el ancho del bus de direcciones y n el del bus de datos.

La CPU es la que provee la información de dirección para los otros subsistemas (memoria y E/S). Los buses de datos y de direcciones transmiten información que es interpretada en forma conjunta, en cambio en el bus de control se pueden interpretar los bits por separado.

Dentro de la CPU hay un módulo funcional llamado *Unidad de Control* que se encarga de administrar el control de los 3 subsistemas. También está la ALU (Unidad Aritmético-Lógica) y un conjunto de registros.

- Formatos de instrucción

Una instrucción es un conjunto de bits que le dice al procesador qué es lo que tiene que hacer y con qué para que produzca un resultado. Para ello se necesita un código de operación, la dirección de los operandos, del resultado y de la próxima instrucción.

La *máquina de 4 direcciones* contiene todos estos campos, pero es un concepto teórico. La de *3 direcciones* le quita la dirección de la próxima instrucción, que se encuentra en el PC (Program Counter o contador del programa) y que se va incrementando automáticamente en uno, lo que implica que la ejecución es secuencial. Si quiero hacer un salto debo agregar una instrucción que lo haga. La *máquina de 2 direcciones* le quita la dirección del primer operando o la del resultado, la cual está implícita. Este tipo de máquina es muy común. Por última la *máquina de 1 dirección* le quita la dirección del segundo operando, y se tiene un registro acumulador que va acumulando los resultados parciales de operaciones sucesivas.

Cuanto menos direcciones tengan las máquinas, más lentas van a ser por la cantidad de instrucciones que tienen que hacer, aunque con las velocidad actuales esto no es tan importante.

- Ciclo de instrucción

1) IF (Instruction Fetch): Busca una instrucción en la memoria.

2) Decod: Interpreta la instrucción, lo que implica resolver la información del IR (Instruction Register).
3) OF (Operand Fetch): Busca los operandos.
4) EX (Execute) Ejecuta la instrucción.

Esta secuencia la repite indefinidamente la Unidad de Control.

Para aumentar la performance de una máquina, debemos tener en cuenta los conceptos de paralelismo y de concurrencia. El *paralelismo* es hacer la mayor cantidad de cosas en un mismo intervalo de tiempo. La *concurrencia* es hacer más de una cosa en forma simultánea. Un tipo de procesamiento en paralelo es el "pipeline", que es el procesamiento en línea.

- Modos de direccionamiento

Los modos de direccionamiento son las diferentes estrategias que tiene un procesador para referenciar un operando. Los modos *simples* tienen un solo componente, o sea que tiene un solo elemento para indicar dónde está el operando, y los *calculados* tienen más de un componente, lo que requieren una composición de los mismos para obtener la dirección del operando. El modo de direccionamiento puede estar con el código de la operación o puede tener un campo aparte. Ejemplos de modos simples son el vía registro, el absoluto o directo, el inmediato y el indirecto, y ejemplos de modos calculados son el paginado, el base, el indexado y el relativo al PC.

- Repertorio de instrucciones

Las instrucciones se clasifican según su función:

1) Movimiento de datos → MOV
2) Aritméticas → ADD, DAC, SUB, SUBB, MUL, IMUL, DIV, IDIV, INC, DEC, NEG, CMP
3) Lógicas → AND, OR, XOR, TEST
4) Rotación y desplazamiento → ROL, RCL, ROR, RCR, SAL, SAR, SHL, SHR
5) Control de programa → JMP, CALL, RET, JC, JNC, JO, JNO, JS, JNS, JZ, JNZ
6) E/S → IN, OUT
7) Control de la CPU → CLC, STC, CMC, CLI, STI, HLT, NOP

La cantidad de instrucciones fue creciendo con la tecnología de circuitos integrados, teniendo en cuenta la performance del sistema y su facilidad de uso. Con la performance se aumenta la velocidad y con la facilidad de uso se trata de implementar las instrucciones de alto nivel en lenguaje de máquina con la menor cantidad de instrucciones posibles. A este proceso de conversión se lo llama *GAP semántico*, donde GAP es una distancia entre 2 zonas o niveles, y es el grado de dificultad para convertir estas sentencias de alto nivel con la menor cantidad posible de instrucciones a nivel máquina. Esto tiene 2 impactos: el costo y la velocidad de operación. El costo va a se despreciable, pero la velocidad de operación es más notable, ya que disminuye la performance. Igualmente va a ser más rápido si utilizo una operación implementada a bajo nivel que la misma en alto nivel, ya que llevaría varias instrucciones.

Con respecto a las instrucciones, existen 2 tipos de máquinas: las CISC (Complex Instruction Ser Computers) y las RISC (Reduced Instruction Set Computers). La diferencia es la cantidad de instrucciones de cada una, los CISC tienen una gran cantidad de modos de direccionamiento, los RISC tienen el formato de instrucción fijo (en los CISC

es variable), los RISC son del tipo LOAD y STORE (los CISC son de tipo acumulador y registro). Los RISC tienen una gran performance, debido a que permite el uso de pipe line.

Por otro lado, el uso del área del chip es muy importante, ya que una parte del mismo es reservada para la Unidad de Control y la otra parte para los registros. Luego, cuanto más registros, menos accesos a memoria van a haber. El RISC tiene un banco de registros mucho más amplio, pero no tienen interrupciones ni E/S. Además el pipe line con números en punto flotante es sumamente complicado. También el uso intensivo de los registros es para determinados programas.

Para los RISC se hizo las siguiente estadística: las instrucciones más usadas son las de movimiento de datos (45%), seguido de las de control de programas (25%), luego las aritmético-lógicas y de rotación y desplazamiento (20%) y lo que queda para el resto.

Otro aspecto es el tiempo de ejecución de cada instrucción. Las que más tardan son las de control de programa, seguidas de las de movimiento de datos y luego las aritmético-lógicas y de rotación y desplazamiento.

- Interrupciones

Una interrupción es una alteración en el proceso de ejecución de un programa por efecto de un evento aleatorio que requiere atención de la CPU. Están pensadas para administrar operaciones de E/S sobre periféricos.

El uso de interrupciones implica una mayor eficiencia de la CPU. La mayor desventaja surge de la naturaleza aleatoria de una interrupción, ya que debemos modificar el estado de un programa en el momento en que ésta ocurre.

Es difícil verificar que un programa funcione correctamente con el uso de la administración de interrupciones. El modo de la interrupción se llama *servicio de interrupción.*

Las interrupciones pueden ser externas o internas. Las *externas* están vinculadas a operaciones de E/S y se dan por hardware mediante señales de entrada a la CPU, mientras que las *internas* están relacionadas con acciones internas al sistema de cómputo. Estas últimas pueden ser detectadas o programas. Las *detectadas* están vinculadas a problemas especiales en la ejecución de instrucciones, como por ejemplo un error de lectura en memoria. Las *programadas* son instrucciones que producen un efecto similar al de una interrupción

Tratamiento de las interrupciones

1) Detectar la interruppción → el procesador analiza si existen interrupciones al comienzo o al final de una instrucción
2) Detener y salvar el proceso a interrumpir → el PC no se carga con la próxima instrucción del programa, sino que se apila para ejecutar la interrupción, el cual se guarda como mínimo junto con los flags
3) Detectar su fuente → como pueden haber varios dispositivos conectados a una señal de interrupción, existen 2 métodos: el polling o encuesta y vectorizado. El de *polling* o *encuesta* es el que usa Motorola y cada periférico debe disponer de una línea capaz de generar un pedido de atención a la CPU; la suma lógica de los pedidos de interrupción es examinada por la CPU antes de ejecutar una instrucción. Cuando llega un 1 por la línea INT se realiza la encuesta, que consiste en preguntar a cada pedido de servicio para determinar qué periférico generó la interrupción, lo que lo hace muy costoso si se tienen muchos dispositivos. La mayor prioridad la tiene el periférico 1 y la menor el periférico N. El *vectorizado* es el que usa Intel y posee un vector que identifica el pedido de interrupción de cualquiera de los dispositivos

periféricos. Cuando llega un 1 por la línea INT se genera una señal de reconocimiento de la interrupción (INTA) para que el controlador de interrupciones le devuelva el identificador de la fuente. Si hay más de un pedido se establece un orden de prioridades.

4) Transferir el control a la rutina de servicio de la interrupción → busca la dirección de la primera instrucción de servicio de atención de la interrupción, la cual depende del procesador.
5) Devolver el control → carga el valor de la próxima dirección del proceso interrumpido, o sea que desapila el estado anterior.

Habilitación de interrupciones (enmascaramiento)

La interrupciones pueden ser enmascarables o no enmascarables. Las *enmascarables* se atienden siempre, mientras que las *no enmascarables* se atienden siempre y cuando el flag de habilitación (IF = Interrupt Flag) se lo permita. IF = 0 ⇒ deshabilitada, IF = 1 ⇒ habilitada.

Interrupciones en el 8086

- **Externas:**
  - Reset → es la de mayor prioridad y fuerza a la CPU a tener un estado fijo y conocido.
  - NMI → es un pedido de interrupción no enmascarable y lo que hace es apilar los flags y acceder a la 2° posición del vector de interrupciones, que salva el PC en la pila.
  - INTR → es un pedido de interrupción enmascarable y lo que hace es apilar los flags y acceder a la posición del vector que le envía el PIC (Programmable Interrupt Controller).
- **Internas:**
  - **Programadas:** Son generadas por la instrucción INT N, donde N toma los valores enteros del 0 al 255 para apuntar a la rutina de atención de la interrupción. Transfieren el control al S.O., el cual se comunica con el BIOS (donde están las rutinas de atención).
    - INTO → interrrupción por overflow
    - BREAKPOINT → puntos de parada en un debbuger
  - **Detectadas:** La Unidad de Control genera señales internas como producto de la ejecución de alguna instrucción.
    - División por cero → excepción o trap
    - Trace → ejecución de instrucciones paso por paso (se habilita con el flag TF = Trace Flag)

Bus de control del 8086:

Sus líneas más importantes son:

- **R/W:** Indica el sentido de la transferencia.
  = 1 ⇒ lee (mem. → CPU o E/S)
  = 0 ⇒ escribe (CPU → mem. o E/S)
- **Interrupciones:** señales de entrada → NMI y INTR
  señales de salida → INTA (indica que está en un proceso de atención de una interrupción)
- **IO/M:** Señal de salida que selecciona si va a direccionar a la memoria o a la E/S

- **Reloj:** Sincroniza los instantes de tiempo en los que se realiza una actividad
  = 1 ⇒ accesos a memoria
  = 0 ⇒ transferencia de CPU a memoria
- **HOLD:** Se indica a la CPU que se desconecte de los buses
- **Reset:** Pone el IP en 0 y el CS en FFFF, enmascara las interrupciones que se pueden llegar a generar y busca la próxima instrucción a ejecutar en la dirección FFFF:0000
- **Ready:** Se usa con dispositivos muy lentos
  = 1 ⇒ se puede utilizar la memoria o la E/S
  = 0 ⇒ se deben esperar ciclos de reloj para poder acceder a la memoria o a la E/S

Subistema de E/S

El subsistema de E/S controla los dispositivos externos e intercambia datos entre éstos y la memoria principal y/o los registros de la CPU. Se tienen 2 interfaces: una interna (CPU y memoria principal) y otra externa (dispositivo). El *módulo de E/S* permite que la CPU visualice un amplio rango de dispositivos de manera sencilla. Un *controlador de dispositivo* es un módulo primitivo que requiere un control detallado.

Puertas de E/S

Son dispositivos que se conectan por un lado al bus del sistema y por el otro con un periférico. Su objetivo es intercambiar datos entre las partes a las que se conectan. Para una correcta transferencia se requiere una información de control (status).

- **Puertas serie:** tienen una sola línea de transferencia, lo que obliga a secuenciar la transferencia de N bits a intervalos regulares de tiempo.
  - **De salida:** Posee un registro de transmisión y una única línea de salida llamada Serial Out. El registro se carga en paralelo y se van desplazando los bits a la derecha hasta completar la transmisión.
  - **De entrada:** Tiene un registro de recepción que se carga en forma serie a través de una única línea llamada Serial In y se lee a través del bus de datos.
  - **Transmisión sincrónica:** Existe un reloj común entre el transmisor y el receptor, el cual identifica en qué instante se transmite cada bit. Se envían bloques de datos con un encabezado para el sincronismo y un fin de mensaje para la detección y corrección de errores. El receptor debe realizar la misma función que el transmisor para comparar los valores.
  - **Transmisión asincrónica:** Cada dispositivo tiene su propio reloj. Se envían bytes, donde cada uno tiene un bit de arranque (0), los bits del dato, un bit de paridad y 2 bits de parada (11). Los relojes se sincronizan con el bit de arranque. Cuando no hay transmisión, la línea está permanentemente en 1, y todo mensaje comienza con un 0. Es menos eficiente que el anterior y más difícil de implementar.
- **Puertas paralelo:** Transfiere simultáneamente tantos bits como bits tiene el bus de datos.
  - **De salida:** Tiene un registro de datos de tantos bits como bits tenga el bus de datos, y éstos se cargan simultáneamente.
  - **De entrada:** Está compuesta por un conjunto de llaves cuyas entradas están controladas por el periférico, las cuales se cierran cuando se selecciona el dispositivo al que se conecta la puerta. La información es

controlada por el periférico: La CPU lee, no escribe. No es necesario ningún registro porque la CPU es más rápida que el periférico y los datos pasan al bus de datos mediante una interrupción.

- **De E/S:** Es lo que se tiene en la realidad. Dentro de la lógica de control existe un registro que indica entrada o salida, el cual controla el funcionamiento de la puerta. Los N bits del registro se programan para que sean de entrada o de salida.

La puerta paralelo es más rápida que la serie, pero necesita N líneas de transmisión. Para los dispositivos que están lejos del sistema de cómputo se utilizan una conexión serie.

- Técnicas de acceso al subsistema de E/S

- **E/S mapeada en memoria (Motorola):** El espacio de memoria es común para la memoria y para la E/S, donde uan parte es para la memoria y otra para la E/S. Las instrucciones para los registros de las puertas de E/S son las mismas que se usan para acceder a la memoria. La ventaja es que tengo un amplio repertorio de instrucciones para las puertas, y la desventaja es que el espacio guardado para la E/S se pierde para la memoria, aunque en realidad es mínimo.
- **E/S mapeada independientemente de la memoria (Intel):** Las instrucciones para las puertas de E/S son distintas de las direcciones de memoria. La desventaja es la cantidad de líneas necesarias para manejar ambos buses. Las líneas de dirección y de datos son iguales, por lo tanto se usan señales de control para ver qué tipo de acceso va a ser.

- UART 8250

Es una combinación de una puerta de salida y una de entrada serie. Está basada en la norma de interface RS232C (una norma de transmisión que especifica la distribución de las señales eléctricas). Convierte los datos paralelos que llegan desde el sistema a datos en serie para salida hacia un circuito de comunicación de datos. De forma similar, convierte los datos en serie que llegan de otra unidad a datos en paralelo que salen hacia el sistema.

Tiene 10 registros internos, los cuales se programan a través de su escritura para establecer el protocolo de comunicaciones. Cuando se detectan errores o la llegada de datos, se debe comunicar a la CPU.

Las señales sIN y sOUT son de enlace para la comunicación serie. Las líneas A_0, A_1 y A_2 son las que seleccionan el registro interno en cuestión. La líneas A_3 a A_9 del bus de direcciones habilita la puerta cuando todas valen 1.

Otras líneas son:

DTR = Data Terminal Ready
DSR = Data Set Ready
INTR → genera una interrupción en la CPU
RESET→ inicializa los registros
R/W = Read/Write → líneas del bus de control

Los registros internos son:

LCR = Line Control Register → controla lo relacionado con el formato de transmisión y recepción
bits 0 y 1 → longitud del mensaje (5, 6, 7)
bit 2 → bits de parada (1, 2)

bits 3, 4 y 5 → paridad
bit 7 → acceso al divisor

LSR = Line Status Register → reporta cómo se están recibiendo los datos, o sea el estado de la puerta en cuanto a la transmisión o a la recepción
bit 0 → DR lleno (dato listo para ser recibido)
bits 1, 2 y 3 → error de recepción
bit 5 → TE = Trasnmitter Emmpty; se activa cuando el TH está vacío, lo que implica que está listo para transmitir

TH → registro de transmisión; se escribe el dato que se quiere enviar
bits 0 y 1 → stop
bit 2 → paridad
bits 3, 4, 5 y 6 → TX (dato)
bit 7 → start

DR → registro de recepción; recibe la información y se extrae el byte que se está transmitiendo

IER = Interruption Enable Register → registro de habilitación de interrupciones
bit 0 → DR lleno
bit 1 → TH vacío
bit 2 → LSR (error de recepción)
bit 3 → MSR (cambios en las líneas de entrada asociadas al modem)

IIR = Indentify Interruption Register → identifica la fuente de la interrupción
bit 0 → pedido de interrupción pendiente
bit 1 y 2 → ID; identifica el pedido de interrupciones con más prioridad (la recepción tiene más prioridad que la transmisión)

MSR = MODEM Status Register → registro de estado del modem
bits 0 y 1 → ACTS y ADSR; variación de las señales DSR o CTS desde la última vez que este registro fue leído por la CPU
bits 4 y 5 → DSR y CTS; determinan el estado de las líneas asociadas al modem

MCR = Modem Control Register → registro de control del modem
bit 0 → maneja la línea DTR
bit 1 → maneja la línea RTS
bit 4 → LOOP; testea la puerta

DLA = Latch Divisor → registro de división; especifica la velocidad de la trasmisión serie, programando un número que divide la frecuencia patrón para generar la frecuencia de recepción, la cual viene dada por el reloj

- <u>Administración de operaciones de E/S</u>

Las operaciones de E/S pueden ser:

- **Administradas directamente por la CPU:** La CPU interviene directamente en cada transferencia de información
  - **Iniciadas por software (programadas):** Implica inhibir toda fuente de interrupción. Lo que hace es saltar a una subrutina correspondiente a un servicio de E/S como producto de la ejecución de una instrucción. La CPU controla que el dispositivo de E/S esté listo y va verificando periódicamente su estado para comprobar la finalización de la operación. Una gran desventaja de esto es que mantiene al procesador ocupado de manera innecesaria.
  - **Iniciadas por interrupción:** La CPU envía le un comando al módulo de E/S y ejecuta otras instrucciones hasta que es interrumpida por el módulo cuando éste pueda comenzar el intercambio de datos. Es más eficiente que el anterior porque sólo se bifurca a la rutina de servicio cuando se requiere una operación de E/S. El uso de interrupciones complica el manejo del sistema. El gasto de tiempo requerido puede ser considerable si el dispositivo es de alta velocidad.
- **Controladas por hardware (DMA):** La CPU no interviene directamente en la transferencia, sólo la supervisa. Esta técnica es indispensable cuando el periférico de muy alta velocidad, con lo que se transfieren bloques de información y no bytes sueltos.
  La CPU sólo se encarga de inicializar el proceso, donde se le manda la dirección de inicio y el tamaño del bloque a transferir, y verificar el resultado, que implica buscar los errores en el CDMA, que es el dispositivo que realiza la transferencia, y verificar Tanto la CPU como el DMA manejan el bus del sistema, por lo que debe haber un control del mismo. Lo que se hace es enviarle a la CPU una señal HALT para que libere el bus, el cual responde con una señal HALTA. Cuando la CPU se desconecta del bus, lo administra el CDMA y luego le devuelve el control a la CPU retirando el pedido de HALT.
  Si hay más de un periférico, se puede tener un CDMA para todos (administración sencilla) o un controlador por periférico, con lo cual se debe agregar un controlador del bus que va decidiendo a qué dispositivo le da el control del bus; su implementación es costosa.

Técnicas de administración de DMA:

- Por HALT → su implementación es sencilla, pero la CPU puede quedar ociosa por mucho tiempo
- Por robo de ciclo (Intel) → la CPU y el CDMA compiten por el uso del bus (el CDMA tiene mayor prioridad que la CPU) y se sincroniza la transferencia con el reloj del sistema; la CPU se puede retrasar pero no queda osciosa
- Acceso a fases diferentes dl reloj (Motorola) → tiene la máxima velocidad de transferencia porque ni la CPU ni el CDMA se retrasan, ya que la CPU accede al reloj cuando está en 1 y el CDMA cuando está en 0, aunque no siempre es posible sincronizar el CDMA con el reloj del sistema

Memoria Caché

Las memorias caché son más chicas que la memoria principal, con una velocidad mucho mayor y suelen implementarse con memoria asociativas, donde el acceso se hace por contenido. El dato clave es la dirección donde se encuentra la información en la memoria principal y el dato asociado es el dato en sí. Lo que hace la memoria caché es almacenar los

datos más frecuentemente usados de la memoria principal para satisfacer más rápidamente los requerimiento de la Unidad de Control. Si al buscar un dato éste está en la caché, entonces ocurre un acierto (hit), y sino ocurre un fallo (miss). Existe un principio llamado de *localidad* que establece que cuando un bloque de datos es trasladado a la caché, es muy probable que las referencias posteriores sean respecto a otras palabras de ese bloque. Es por eso que se logra un alto porcentaje de aciertos.

Análisis de performance:

La performance depende de lo que se llama *probabilidad de acierto* (ϕ), el cual depende del tamaño de la memoria caché, del tipo de programa que se está corriendo, del estilo de programación, de la organización de la caché, etc. Esta probabilidad varía entre 0 y 1: es 0 cuando nunca encuentra la información, y es 1 cuando la encuentra.

Si T_c es el tiempo de acceso a la memoria caché, T_m el tiempo de acceso a la memoria principal, y T_e el tiempo de acceso efectivo (promedio de los tiempos de acceso), luego $T_c < T_m$.

Supongamos que tengo N accesos, donde $N = N_c + N_m$ (N_c =cant. De accesos a la memoria principal y N_m = cant. de accesos a la memoria caché). Luego tengo que:

$N * T_e = N_c * T_c + N_m * T_m \Rightarrow Te = N_c/N * T_c + N_m/N * T_m \Rightarrow T_e = \phi * T_c + (1 - \phi) * T_m$

Para demostrar la mejora que se obtiene con el uso de la caché, calculamos el factor de aceleración (S):

$S = T_m/T_e = T_m/(\phi * T_c + (1 - \phi) * T_m) = T_m/(T_m + (T_c - T_m)\ \phi) = 1/(1 + (T_c/T_m - 1)\ \phi) = 1/(1 - (1 - T_c/T_m)\ \phi)$

Organización lógica de la memoria caché:

Se la particiona en bloques del mismo tamaño, que es la mínima información que se transfiere desde la memoria caché a la memoria principal. La coherencia de los bloques de la caché y de la memoria principal es indispensable.

Para mapear las direcciones de una memoria a la otra es necesario un algoritmo. Para hacer la búsqueda en la caché, cada dirección de memoria principal se divide en campos. Se pueden usar 3 técnicas:

- **Mapeo directo:** En la memoria principal se forman grupos de N bloques cada uno, donde N es la cantidad total de bloques de la memoria caché. La dirección de la memoria principal consta de 3 campos: P bits para el grupo, Q bits para el bloque y S bits para la palabra. O sea que en la memoria principal se tienen 2^P grupos, 2^Q bloques por grupo y 2^S palabras por bloque. Para cada bloque de la memoria caché hay un registro (TAG) que contiene el número de grupo con que fue mapeado. El acceso a estos registros es aleatorio y es por dirección. Cuando una dirección de memoria es interceptada por la caché, se localiza el bloque de dicha dirección dentro de la caché; si coincide el número de grupo, entonces el número de palabra se utiliza para seleccionar la palabra solicitada.
  En el bloque i de la memoria caché sólo puede haber bloques i de cualquier grupo de la memoria principal. Por eso el mapeo es único para cada bloque de la memoria principal. Por más que la asignación de estos bloques sea muy rígida, el mapeo directo es uno de los más usados debido a su simplicidad y rapidez para la conversión de las direcciones.
- **Mapeo asociativo:** Permite mapear un bloque de la memoria principal en cualquier bloque de la caché. En los registros de la caché está el número de bloque de la

memoria principal asignado a cada bloque de la caché. Para determinar si un bloque está en la caché, se busca en todas las palabras al mismo tiempo, con lo que se requiere de una implementación costosa pero mejorada con respecto al mapeado directo.

- **Mapeo asociativo por grupos:** La memoria principal y la caché se particionan en grupos. El bloque i de la memoria principal puede ser mapeado en el bloque i de cualquiera de los grupos de la caché. Con el número de boque se indexa en todos los grupos de la caché, y si alguno da correcto, significa que en alguno está mapeado el grupo correspondiente al bloque. Este método es más complicado de administrar. Si se tiene un solo grupo se trata del mapeo asociativo, y si se tienen tantos grupos como bloques se trata del mapeo directo.