IR: registro de la CPU donde se almacena temporalmente la instrucción.
El bloque control puede “leer” IR y así saber qué hacer, dónde están los operandos y dónde poner el resultado.
PC: registro en el que se encuentra la próxima instrucción a ejecutar.
La CPU tiene registros internos de propósito general que pueden ser referenciados por el programador. Estos registros son lugares de almacenamiento temporario.
La CPU interactúa con la memoria a través de un par de registros que no son visibles para el programador.
MAR: registro de dirección de memoria.
MBR: registro de dato de memoria.
Estos registros están conectados a los buses
Además la CPU tiene otros registros que permiten almacenar direcciones, para poder brindar flexibilidad.
Los programas están compuestos de instrucciones almacenadas en memoria.
La CPU procesa dichas instrucciones, las debe traer desde memoria una por vez y debe cumplirlas.
Procesamiento de instrucciones:
-Búsqueda: leer desde memoria.
-Ejecución: dependiendo de la instrucción puede implicar varias operaciones.
Ciclo de instrucción: ciclo de búsqueda y ciclo de ejecución.
Al principio de cada ciclo, la CPU busca una instrucción en memoria.
En la CPU hay un registro (PC) que tiene la dirección de la próxima instrucción a buscar.
La CPU, después de buscar cada instrucción, incrementa el valor contenido en PC, así podrá buscar la siguiente instrucción en secuencia.
La instrucción buscada se carga dentro de un registro de la CPU, llamado registro de instrucción (IR).
La CPU interpreta cada instrucción y lleva a cabo las acciones requeridas.
Las acciones caen 4 tipos:
CPU – Memoria: datos pueden transferirse entre memoria y CPU.
CPU – E/S: datos pueden transferirse entre CPU y entrada/salida.
Procesamiento de datos: CPU efectúa operaciones aritméticas o lógicas en datos.
Control: alterar la secuencia de ejecución de instrucciones.

Comentado [ACM1]: CICLO DE INSTRUCCIÓN

Diagrama de estados:
1. cálculo dirección instrucción: determina la dirección de la siguiente instrucción a ejecutarse
2. búsqueda instrucción: lee la instrucción de su posición de memoria a la cpu.
3. decodificación de la instrucción: analiza la instrucción para determinar el tipo de operación a realizar y los operandos que se usarán.
4. cálculo dirección operando: si la operación implica la referencia a un operando en la memoria ó e/s, entonces se determina la dirección.
5. búsqueda del operando: busca el operando en la memoria ó e/s.
6. operación sobre los datos: ejecuta la instrucción.
7. cálculo dirección resultado.
8. almacenamiento resultado.

• Los estados en la parte superior implican un intercambio entre la cpu y la memoria ó e/s.
• Los estados en la parte inferior implican sólo operaciones internas en la cpu.

Elementos de una instrucción de máquina:
-Código de operación:
- especifica la operación a realizar (ej. suma).

- es un código binario.

-Referencia del operando fuente

-Establece dónde se encuentra el operando.

-la operación puede involucrar uno ó más operando fuente (o de entrada).

-Referencia del operando resultado

-establece dónde almacenar el resultado

-Referencia de la siguiente instrucción

-le dice a la CPU donde buscar la siguiente instrucción después de la ejecución de la instrucción anterior.

-en la mayoría de los casos se ubica a continuación de la instrucción actual.

-Los operandos fuente y resultado pueden estar en tres lugares:

- Memoria
- Registro de la CPU
- Dispositivo de E/S

Representación de instrucciones:

-Dentro de la computadora cada instrucción está representada mediante una secuencia de bits

-La secuencia se divide en campos en correspondencia a los elementos que la componen.

-Este esquema se conoce como formato de la instrucción

-Es difícil para el programador tratar con las representaciones binarias de las instrucciones de máquina. Por lo tanto, se usa una representación simbólica.

-Los códigos de operación se representan por medio de abreviaturas, llamadas mnemónicos que indican la operación.

-Los ejemplos más comunes son: (algunos ya los vimos en el Ingreso)

-ADD adición (suma)

-SUB sustracción (resta)

-MOV movimiento de datos

-AND, OR, XOR operaciones lógicas

-Los operandos también se pueden representar de manera simbólica.

Ej: MOV reg1 , memoY

-instrucción que copia el valor contenido en la posición de memoria llamada memoY, a un registro denominado reg1.

Comentado [ACM2]: FORMATO DE INSTRUCCIÓN

Tipos de instrucciones

- En lenguajes de alto nivel escribimos:

  X := X + Y

  -Esta instrucción suma los valores almacenados en las posiciones de memoria X e Y.

  -Esto puede implicar cargar registros, sumarlos y luego almacenar el resultado en memoria.

Una instrucción de alto nivel puede requerir varias instrucciones de máquina.

El lenguaje de alto nivel expresa operaciones en forma “concisa” usando variables.

El lenguaje de máquina expresa las operaciones en forma “básica” involucrando movimiento de datos y uso de registros.

Cualquier programa escrito en lenguaje de alto nivel se debe convertir a un lenguaje de máquina para ser ejecutado.

El conjunto de instrucciones de máquina debe ser capaz de expresar cualquiera de las instrucciones de un lenguaje de alto nivel.

Podemos categorizar las instrucciones de máquina como de:

- Procesamiento de datos: operaciones aritméticas y lógicas.
- Almacenamiento de datos: transferencias dentro del sistema.
- Instrucciones de E/S: transferencia de datos entre la computadora y los mecanismos externos.
- Control

¿Cuántas direcciones se necesitan?

Máquina para 4 direcciones:

- Dos direcciones para hacer referencia a los operandos, una donde almacenar el resultado y la dirección de la próxima instrucción. Por lo tanto necesitaríamos cuatro direcciones:
- Add DirRes, DirOp1, DirOp2, DirPróxIns
- Direcciones explícitas para operandos, resultado y próxima instrucción.
- Son "raras", cada campo de dirección tiene que tener bits para "acomodar" una dirección completa.
- Ej. si dirección = 24 bits, la instrucción tiene 96 bits de referencias.

Máquina para 3 direcciones:

- Add DirRes, DirOp1, DirOp2
- Dirección de la próxima instrucción está almacenada en un registro de la CPU, llamado Contador de Programa PC.
- Referencias = 72 bits. Todavía larga.

Máquina para 2 direcciones:

- Add DirOp1, DirOp2:
- Reduce el tamaño de la instrucción.
- 48 bits de referencias.
- Hay que mover el Op1 a un registro temporal.
- Menos elección donde guardar el resultado.

Máquina para 1 dirección:

- Add DirOp1
- Registros especiales en la CPU (acumulador)
- Instrucciones para cargar y descargar el acumulador.
- Un operando y resultado en lugar predefinido
- Instrucción más corta (24 bits de referencias)

Diseño del conjunto de instrucciones:

El conjunto de instrucciones es el medio que tiene el programador para controlar la CPU.

Hay que tener en cuenta:

- Tipos de operaciones: cuántas y cuáles
- Tipos de datos: cuáles
- Formato de instrucciones: longitud (bits), No de direcciones, tamaño de cada campo
- Registros: cantidad que se pueden referenciar mediante instrucciones y su uso
- Direccionamiento: la manera de especificar la dirección de un operando o una instrucción (la próxima).

Tipos de operaciones:

- Transferencia de datos: Mov (load/store)
- Aritméticas: Add, Sub, Inc, Dec, Mul
- Lógicas: And, Or, Xor, Not
- Conversión
- E/S: In, Out
- Transferencia de control: salto, bifurcación

Comentado [ACM3]: TIPOS DE INSTRUCCIONES

Comentado [ACM4]: MÁQUINAS DE 1/2/3/4 DIRECCIONES

Comentado [ACM5]: DISEÑO DEL CONJUNTO DE INSTRUCCIONES

- Control del sistema: usadas por S.O.

Tipos de datos:

Los más importantes:

- Direcciones
- Números: enteros, p. fijo, p. flotante
- Caracteres: ASCII, BCD.
- Datos lógicos

Modos de direccionamiento:

- Como vimos, en una instrucción se utilizan bits para expresar el código de operación: nos dice qué hacer. También se necesitan una "gran" cantidad de bits para especificar de dónde provienen los datos.
- ¿Cómo podemos reducir el tamaño de estas especificaciones?
  - Hay 2 métodos generales:
    - 1.Si un operando va a usarse varias veces puede colocarse en un registro. Usar registro para una variable tiene 2 ventajas:
      -el acceso es más rápido
      -se necesitan menos bits.
      Ej. si hay 32 reg. se necesitan 5 bits para especificar c/u de ellos (menos bits que las dir. de mem.).
    - 2. Especificar uno ó más operandos en forma implícita.
      Ejemplos: reg2 = reg2 + fuente1; el acumulador.
    - Los mdd tienen como objetivo:
      -disminuir la cantidad de bits en la instrucción
      -la dirección puede que no se conozca hasta el momento de ejecutar el programa
      -manejo más eficiente de datos (arreglos)
- Inmediato
- Directo
- Por registro
- Indirecto por memoria
- Indirecto por registro
- Por desplazamiento
- Del stack

Mdd Inmediato:

- El operando se obtiene automáticamente de la memoria al mismo tiempo que la instrucción. No requiere una referencia extra a memoria de datos
- Se utiliza para definir constantes y para inicializar variables.
- Desventaja: tamaño del operando limitado por el tamaño del campo de direccionamiento.

Mdd Directo:

- El campo de dirección tiene la dirección efectiva del operando.
- Es simple, pero tiene un espacio limitado de direcciones por cantidad de bits del campo.
- Uso: acceder a variables globales, cuya dirección se conoce en el momento de compilación.

Mdd Por registro:

- Conceptualmente igual al directo, pero se especifica un registro en lugar de una posición de memoria.

Comentado [ACM6]: TIPOS DE OPERACIONES

Comentado [ACM7]: TIPOS DE DATOS

Comentado [ACM8]: MODOS DE DIRECCIONAMIENTO

- La referencia a registro usa menos bits que la especificación de la dirección y no requiere acceso a memoria de datos.
- Desventaja: los registros no son muchos y es un recurso preciado.

Mdd Indirecto por memoria:

- En la instrucción está la dirección de la dirección del operando. Trata de solucionar el problema del directo. Así, con una dirección de menos bits en la instrucción, se apunta a una dirección de más bits.
- Ventaja: espacio de direccionamiento mayor
- Desventaja: múltiples accesos a memoria.

Mdd Indirecto por registro:

- En la instrucción se especifica el registro que tiene almacenada la dirección.
- Ventaja: menos bits para especificar el registro que la posición de memoria. Espacio de direccionamiento grande, accede una vez menos a memoria que el indirecto. La dirección así usada se llama apuntador.

Mdd Por desplazamiento:

- Combina capacidades de indirecto y directo. Requiere que la instrucción tenga dos campos de dirección. Estos dos campos se suman para producir la dirección efectiva.
  Los más comunes:
  -Relativo
  -De registro base
  -Indexado

Comentado [ACM9]: DESCRIPCIÓN MODOS DE DIRECCIONAMIENTO

Relativo:
-El registro referenciado de manera implícita es el contador de programa PC.
-La dirección de la instrucción actual se suma al campo de dirección para producir la dirección efectiva.
-El campo de dirección se trata como un número en Ca2.
De registro base:
-El registro referenciado contiene una dirección de memoria y el campo de dirección tiene un desplazamiento.
Indexado:
-Se direcciona la memoria con un registro más un desplazamiento.
-Es "igual" al anterior pero se intercambian los papeles del registro y del desplazamiento.
-La Indexación proporciona un mecanismo eficiente para realizar operaciones iterativas.
-Se utiliza un registro llamado índice
algunas máquinas incrementan ó decrementan este registro como parte de la instrucción (Autoindexación)
Del stack:
-El stack ó pila es un arreglo lineal de localidades de memoria. Es una lista ó cola donde el último en entrar es el primero en salir. Es una zona de memoria reservada.
-Asociado con la pila o stack hay un registro apuntador (o registro puntero de pila), cuyo valor es la dirección tope de pila o stack.

MSX88:

-inst. de transferencia
-Inst. aritméticas y lógicas
-Inst. transf. de control

Comentado [ACM10]: INSTRUCCIONES MSX

Organización de registros:
Registros visibles al usuario: son utilizados por el programador.

Registros de control y estado: son utilizados por la UC para controlar la operación de la CPU (no son visibles por el programador).
Registros visibles al usuario:

- Propósito general
- Datos
- Dirección
- Códigos de condición

Registros visibles al usuario:
-Pueden ser asignados a una variedad de funciones:

- cualquier registro de propósito general puede contener el operando para cualquier código de operación (verdadero propósito)
- pueden existir restricciones (ej. registros dedicados a operaciones en PF)
- se pueden utilizar para direccionamiento (ej. indirecto de registro)
- sólo para datos ó sólo para direcciones
- los registros de dirección pueden ser asignados para un mdd (ej. reg índice para direccionamiento autoindexado)

Comentado [ACM11]: REGISTROS

Discusión:
¿Todos los registros de propósito general ó especializar su uso?
Todos de propósito general: afecta al tamaño de las instrucciones.
Especializados: puede estar implícito en el código de operación a qué registro se refiere (ej. Acumulador). Se ahorran bits. Limitan la flexibilidad del programador.
No hay una receta.
Número de registros:
-Afecta al tamaño de la instrucción.
-Mayor N° de registros, más bits para especificarlos en la instrucción.
-Pocos registros: más referencias a memoria
-N° óptimo: entre 8 y 32 reg. Más, no hay gran mejora (aumenta tamaño de la instrucción).
Longitud de los registros:

- De direcciones: deben ser capaces de almacenar la dirección más grande.
- De datos: deben estar habilitados para almacenar la mayoría de los tipos de datos.
- Algunas máquinas permiten 2 registros contiguos utilizados como un solo registro para almacenar valores de doble longitud.

Bits de condición (banderas):
-Bits establecidos por la CPU como resultado de operaciones.
-Pueden ser utilizados por las instrucciones de bifurcación condicional.
-Generalmente no son alterados por el programador.

Comentado [ACM12]: BANDERAS

Registros de control y estado:
Empleados para controlar la operación de la CPU. En la mayoría de las máquinas no son visibles al usuario.
Los 4 esenciales para la ejecución de instrucciones:
• Contador de programa (PC)
• Registro de instrucción (IR)
• Registro de dirección de memoria (MAR)
• Registro buffer de memoria (MBR)
-Los 4 reg recién mencionados se emplean para el movimiento de datos entre la cpu y memoria.

-Dentro de la CPU los datos se deben presentar a la ALU para procesamiento, ésta puede acceder al MBR y a los reg visibles por el usuario. Puede haber también reg temporales adicionales para intercambiar datos con el MBR y demás reg visibles.
AX, BX, CX, DX: De uso general
CS, SS, DS, ES, FS, GS: Segmentos
EIP, EFLAGS: PC y banderas
• AX: acumulador, es el principal en las operaciones aritméticas
• BX: puntero base (dir de memoria)
• CX: contador, interviene en instrucciones de ciclo
• DX: datos, participa en multiplicación y división
-SI y DI: apuntadores que utilizan las instrucciones que recorren arreglos o tablas
-BP y SP: también son apuntadores a memoria, pero a una zona especial: pila ó stack
-E: reg de 32 bits
Organización de registros CPU MOTOROLA 68000:
• 8 registros de 32 bits de datos
• 9 registros de direcciones
• 2 stacks: uno para usuario y otro para S.O.

Comentado [ACM13]: REGISTROS

Instrucciones - Intel
Tienen la forma: instrucción destino,fuente
-destino y fuente son 2 operandos, donde c/u de ellos está especificado por alguno de los mdd vistos, el otro operando es un registro de la CPU
Llamando:
• mem = especificación de una dirección de memoria
• reg = registro de la CPU
• inm = dato inmediato
Las instrucciones tienen la forma:
• Instrucción mem, reg
• Instrucción reg , mem
• Instrucción reg , reg
• Instrucción reg , inm
• Instrucción mem, inm
El nombre destino y fuente proviene del hecho que si hay un movimiento de datos, es desde la derecha (fuente) hacia la izquierda (destino).
En una suma hay 2 operandos y el resultado se almacena en el lugar del operando izquierdo (destino).
Ejemplos:
Direccionamiento por registro:
• ADD AX,BX AX=AX+BX
• ADD AL,AH AL=AL+AH
• MOV AL,CH AL=CH
• SUB AX,BX AX=AX - BX
Direccionamiento inmediato:
• ADD AX,35AFh AX=AX+35AFh
• ADD AL,15 AL=AL+15
• MOV AL,3Eh AL=3Eh
• SUB AX,1234h AX=AX - 1234h
Direccionamiento directo:
• ADD AX, [35AFh] AX = AX + contenido direcc. 35AFh y 35B0h

• ADD AL, DATO AL = AL + contenido variable DATO (8 bits)
• MOV CH, NUM1 CH = contenido variable NUM1 (8 bits)

Direccionamiento indirecto por registro:

• ADD AX, [BX] AX = AX + dato almacenado en dirección contenida en BX y la que sigue
• MOV [BX], AL dato en la dirección contenida en BX = AL

Direccionamiento base + índice:

• MOV CX, [BX+SI] CX = dato almacenado en la direcc. BX+SI y la siguiente
• MOV [BX+DI], AL dato almacenado en la direcc. BX+DI = AL

Direccionamiento relativo por registro:

• MOV AL, [BX+2] AL=dato almacenado en dir BX+2
• MOV [BX+2Ah], AX dato almacenado en dir BX+2Ah y la que sigue = AX (16 bits)

Direccionamiento relativo base+índice:

• MOV AL, [BX+SI+2] AL = dato almacenado en la dir BX+SI+2
• MOV [BX+DI+2Ah], AX dato almacenado en la dir BX+DI+2Ah y la que sigue = AX (16 bits)

Formatos de instrucción- Criterios de diseño:

-¿Instrucciones cortas ó largas?
-N° de bits/seg: ancho de banda de la memoria
-Velocidad procesador/Velocidad memoria
-Instrucciones más cortas: el procesador “parece” más rápido.
-Suficientes bits para expresar todas las operaciones deseadas.
-La experiencia demuestra dejar bits libres para el futuro.
-Cantidad de bits de datos.

Comentado [ACM14]: INSTRUCCIONES Y DIRECCIONAMIENTOS

Comentado [ACM15]: FORMATOS DE INSTRUCCIÓN CRITERIOS DE DISEÑO

Memoria

A medida que aumenta la brecha entre las velocidades del procesador y de la memoria, las distintas arquitecturas buscan tender un puente sobre esta brecha. Una computadora típica suele tener distintos tipos de memoria, desde una rápida y cara (registros) hasta una barata y lenta (discos).

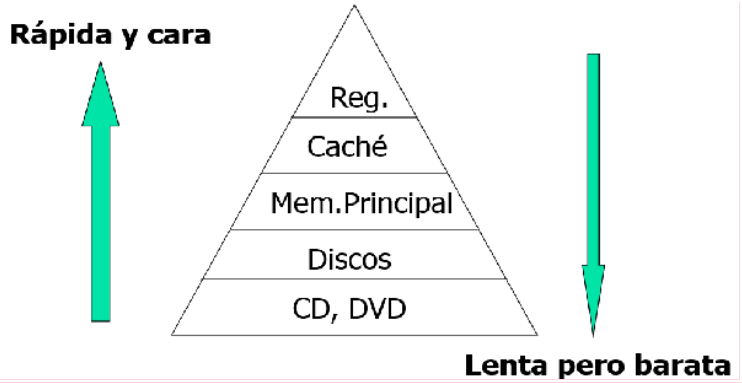
La interacción entre los diferentes tipos de memoria se aprovecha de forma tal que se logra un comportamiento, por parte de la computadora, equivalente al que tendría con una memoria única, grande y rápida, cuando en realidad tiene distintos tipos de memoria trabajando en forma coordinada.

Jerarquía de memorias:

La forma en que se organizan estos distintos tipos de memoria es lo que se conoce como jerarquía de memoria.

Comentado [ACM16]: JERARQUÍA DE MEMORIAS - DEFINCIÓN

En la cima de la jerarquía están los registros. En la base, las memorias secundarias (discos magnéticos) y de almacenamiento “off line” (CD, DVD, cintas).



A medida que ascendemos tenemos mayor rendimiento y más costo por bit. Entre la memoria principal y la secundaria hay otro tipo de memoria para salvar la brecha.
Cuando ascendemos, también aumenta la frecuencia de accesos a ese tipo de memoria.

Comentado [ACM17]: PIRÁMIDE DE MEMORIA

-Memoria del computador:
• Tecnologías diferentes
• Fundamentos físicos distintos
• Localización en lugares distintos
-Objetivo:
• Capacidad de almacenamiento
• Tiempo de acceso reducido
-Características:
Duración de la información:
• Memorias volátiles: RAM
• Memorias no volátiles: discos, cintas
• Memorias permanentes: ROM, EPROM
Modo de acceso:
• Acceso por palabra: memoria principal
• Acceso por bloque: discos, caché
Velocidad:
Memorias semiconductoras:
Tiempo de acceso: tiempo máximo que transcurre desde que se inicia la operación de lec/esc hasta obtener/almacenar el dato.
Tiempo de ciclo: tiempo mínimo que tiene que haber entre dos operaciones sucesivas sobre la memoria
(t sub-ciclo > t sub-acceso)
Memorias magnéticas:
Tiempo de acceso: tiempo de posicionar el cabezal + tiempo de latencia (+ tiempo de lectura)
Velocidad de transferencia: bytes/seg
Métodos de acceso
• Acceso aleatorio: el tiempo para acceder a una locación dada es independiente de la secuencia de accesos anteriores y es constante. Ejemplo la memoria principal.
• Acceso secuencial: el acceso debe hacerse en una secuencia lineal específica. Variable. Ejemplo son las unidades de cinta.

• Acceso directo: los bloques ó registros individuales tienen una dirección única que se basa en la localización física. Variable. Ejemplo los discos magnéticos.
• Acceso asociativo: memoria caché

Memoria de acceso aleatorio

RAM (Random Access Memory). Aleatorio significa que se puede acceder a cualquier celda de memoria en el mismo tiempo, independientemente de la posición en la estructura de la memoria.

-Basada en flip flops: memoria estática (SRAM)

-Basada en transistores: memoria dinámica (DRAM).

Cargas almacenadas en transistores (capacitores)

El acceso a las ROM también es de éste tipo

-DRAM almacena más información que SRAM en la misma superficie.

Los 'capacitores' son más chicos que los flip flop

-DRAM hay que "refrescarla"

Los 'capacitores' se descargan

Usada como memoria principal

-SRAM más rápida

Usada como memoria caché

Organización:

El elemento básico de una memoria de semiconductor es la celda de memoria.

Todas las celdas de memoria de semiconductor comparten 3 propiedades:

- Dos estados estables: para representar al uno (1) y al cero (0).
- 2. Se puede escribir en ellas, al menos una vez
- 3. Se pueden leer para conocer el estado.

En general la celda tiene 3 terminales funcionales capaces de llevar una señal eléctrica:

- Selección: selecciona una celda de memoria
- Control: especifica lectura ó escritura
- Escritura/Lectura de datos

Organización de la memoria:

Una memoria de 1 bit la implementamos con flip-flop y 'armamos' registros sencillos de n bits con flip-flops (notas de clase3).

Para construir memorias 'más grandes' se requiere una organización diferente, en la cual sea posible 'direccionar' palabras individuales.

Comentado [ACM18]: ORGANIZACIÓN DE MEMORIA

Organización del chip:

Cada chip contiene un arreglo de celdas de memoria.

En las memorias de semiconductor se han empleado dos enfoques organizacionales: 2D y 2½D.

Comentado [ACM19]: ORGANIZACIÓN DEL CHIP

Organización 2D:

-El arreglo está organizado en 2W palabras de B bits cada una. Cada línea horizontal (una de 2W) se conecta a cada posición de memoria, seleccionando un renglón.

-Las líneas verticales conectan cada bit a la salida.

-El decodificador que está en el chip, tiene 2W salidas para W entradas (bits del bus de direcciones).

Comentado [ACM20]: ORGANIZACIÓN 2D

Organización 2½D:

-El arreglo es 'cuadrado' y funciona igual que 2D.

-Los bits de una misma palabra están dispersos en distintos chips.

-La dirección se divide en dos partes: una selección de renglón y una selección de columna. Hay 2 decodificadores.

Comentado [ACM21]: ORGANIZACIÓN 2½D

Comparación:

- En 2D todos los bits están en el mismo chip.
- En 2½D los bits de una misma palabra estarán en distintos chips.
- 2D es muy larga y estrecha, No grande de palabras de pocos bits. Cada línea de selección de palabra tiene que tener un manejador y conectarse al decodificador. Ocupan mucha superficie.
- 2D dificulta el uso eficaz de los circuitos correctores de error. En 2½D al estar los bits dispersos en distintos chips hay menor probabilidad de error.
- En 2½D al usar decodificación separada de filas y columnas, reduce la complejidad de los decodificadores.

Comentado [ACM22]: COMPARACIONES ENTRE AMBAS

Problema 1:
-Cada módulo de memoria cubre el espacio de direccionamiento requerido, pero sólo cubre una parte de la palabra.
-Solución: usar varios módulos 'en paralelo'.
-Problema 2:
-La longitud de la palabra es la deseada, pero los módulos no tienen la capacidad deseada.
-Solución: cubrir un cierto rango de direcciones con módulos de memoria 'en serie'.
-Cada módulo 'estará' en direcciones distintas

Principios:

- El uso de la memoria caché se sustenta en dos principios ó propiedades que exhiben los programas:
  - 1.Principio de localidad espacial de referencia: cuando se accede a una palabra de memoria, es 'muy probable' que el próximo acceso sea en la vecindad de la palabra anterior.
  - 2.Principio de localidad temporal de referencia: cuando se accede a una posición de memoria, es 'muy probable' que un lapso de 'tiempo corto', dicha posición de memoria sea accedida nuevamente.

Comentado [ACM23]: PRINCIPIOS

Localidad espacial
Localidad espacial, se sustenta en:
-Ejecución secuencial del código
-Tendencia de los programadores a hacer próximas entre sí variables relacionadas
-Acceso a estructuras tipo matriz ó pila

Comentado [ACM24]: LOCALIDAD ESPACIAL

Localidad temporal
-Localidad temporal, se sustenta en:
-Formación de ciclos o bucles
-Subrutinas (Procedimientos o Funciones)
-Pilas

Comentado [ACM25]: LOCALIDAD TEMPORAL

Caché:
La idea general es que cuando se hace referencia a una palabra, ella y alguna de las vecinas se traen de la memoria grande y lenta a la caché, para que en el siguiente acceso la palabra buscada se encuentre en el caché.

Comentado [ACM26]: CACHÉ