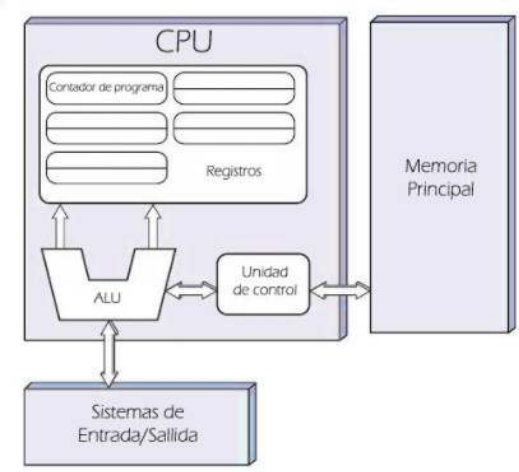
Parcial 2 Arq. de computadores

Arquitectura De Computadores (Universidad Argentina de la Empresa)

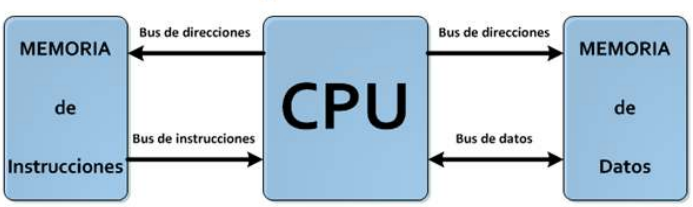
ARQUITECTURA VON NEUMANN



- Procesadores con **mismo** almacenamiento para datos e instrucciones.
- Se almacenan en el mismo formato dentro de la memoria
- Utilizan un único bus de datos para mantenerse en contacto con el CPU.
- Compuestos por la ALU, la unidad de control, una memoria, un dispositivo de entrada y salida y un bus de datos que los comunica. Los 4 componentes se vinculan a través de 3 buses, Bus de control (lectura o escritura), Bus de dirección y Bus de datos (a través de este viajan instrucciones y datos)

- Más lenta, pero más flexible

ARQUITECTURA HARVARD (ARM -> chips de celulares, Apple)



- Procesadores con **diferente** almacenamiento para datos e instrucciones.
- Se almacenan en memorias separadas
- Se utilizan diferentes buses de información.

- Compuestos por la ALU, la unidad de control, 2 memorias, un dispositivo de entrada y salida y un único bus de direcciones, con un control que pueda diferenciar entre ambas memorias
- Las instrucciones y los datos se almacenan en caché separados para mejorar el rendimiento.
- Para unir los 5 componentes utiliza el Bus de control, Bus de direcciones, Bus de datos: Une I/O con memoria de datos, y memoria de datos con ALU, y el Bus de instrucciones: Une Memoria de instrucciones con UC

➔ **Von Neumann**: tengo una sola memoria ≠ **Harvard**: tengo dos memorias
  - Paginación de memoria (Sectorizar en bloques de memoria) → La UC pagará la memoria en páginas de x tamaño. En determinadas páginas se guardarán instrucciones, y en determinadas páginas datos.

MEMORIAS

Tipos

- ➔ **Memoria Volátil:** Aquella que una vez que le quito la energía pierde los datos. Es más rápida que la memoria no volátil. Todas son memorias RAM
- ➔ **Memoria No Volátil:** Aquella que si le quito la energía retiene los datos. Puedo tener memorias ROM o RAM

- **RAM** → random access memory
  - SRAM: Static RAM. Muy rápida, usa transistores; pero consume muchos recursos (requiere potencias muy altas) y permite almacenar pocos datos. Ej: memorias cache
  - DRAM: Dynamic RAM. Utiliza capacitores en lugar de transistores (los capacitores pueden mantener la energía un rato a pesar de que se la quite). Es más barata y puedo almacenar más datos; pero debe refrescar el dato cada un rato (se debe copiar una y otra vez)
  - SDRAM: Sincrónicas con el procesador.
    [DDR SDRAM = double data rate synchronic dynamic RAM]
- **ROM** → read only memory [BIOS]
  - PROM: Memoria ROM programable (una sola vez)
  - EPROM: Memoria ROM programable que se puede borrar (erasable) y volver a programar. Se borra completamente
  - EEPROM: Memoria ROM Electrical Erasable Programable, permite modificar ciertos bytes de la memoria. El BIOS corresponde a una EEPROM
    - BIOS: (Basic Input Output System) es un chip que viene instalado en una memoria ROM. En otras palabras, es el primer programa que ejecuta un computador, diciendo al software cómo trabajar con el hardware.
  - FLASH: EEPROM con tiempos de escritura mucho más rápido, pero más lento de borrar. Es rentable en tiempos, borrar toda la memoria y escribirla de vuelta. Ej: SSD – Pendrive

Buses → se definen por ancho (bits) y velocidad

- ➢ **Bus de dirección:** unidad de control
  - Establece la dirección de memoria del dato en tránsito
- ➢ **Bus de control**: unidad de control
- ➢ **Bus de validación:** unidad de control
  - Activa todas las celdas de cada cajón
- ➢ **Bus de datos:** lo disparan dispositivos que tengan datos para entregar
- ➢ **Bus clock:** independiente (marca pulsos)
  - Marca los pulsos que establecen la frecuencia de trabajo

REGISTROS → memoria mas rapida que necesito en el momento

Operaciones

- Puedo pedir que rote el número [Ej: AH ←→ AL]
- Complemento A2

Tipos

- **Registro de estado**: FLAGS
  - Le permite a la UC supervisar a la ALU
  - Estados de los bits más importantes:
    - Bit 11 → overflow
    - Bit 7 → negativo (módulo y signo)
    - Bit 6 → cero
    - Bit 0 → carry
- **Registro temporal**
  - Guarda el resultado hasta que la UC le dice a la ALU que hacer
  - La unidad de control decodifica las instrucciones y la ALU realiza las operaciones lógicas. Luego, pasan al registro temporal para esperar la orden. Después, utilizo el DI (destination index) para ubicar el resultado.

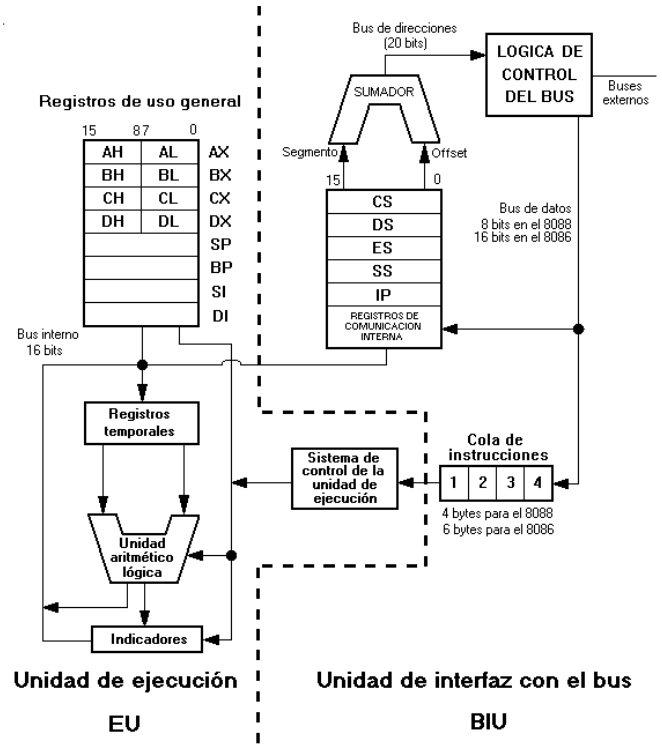
Para grabar un dato:

- Control: para que lea y escriba
- Validación: que valide y llame al registro
- Se lee de a 1 registro, no se pueden activar 2 a la vez
- Se lee recorriendo toda la lista
- Overflow: Es que la lista está llena. El pulso de clock determina cuando la unidad de control decodifica instrucciones.

REGISTROS ≠ MEMORIA

REGISTROS	MEMORIA
Los uso para almacenar datos	La uso para almacenar datos
Data bus	Data bus
-	Address bus
Operaciones (complemento, incremento)	-
Copiar dos registros por vez	-

ARQUITECTURA i8086



- ➢ Lo que vemos de la raya para la izquierda es igual al 4004, el resto es todo nuevo
  - 4004 → cuello de botella: bus de datos y direccion juntos en uno [AD]
- ➢ Pasa de 16 bits a 8 bits y a una cola de instrucción más chica
- Sumador: Es el encargado de sumar los stacks (suma desplazada para generar direcciones de 20 bits)
- **Pipeline (8086)** Mientras está ejecutando la primera instrucción ya se está buscando la segunda. BIU = Busca la siguiente. ALU = La ejecuta
- **SIN Pipeline:** Fetch (busca la instrucción), D = dirección, decodifica y, E = ejecuta.

Procesador

- **EU = execution unit**
  - ALU: módulo que ejecuta las operaciones, tiene varios circuitos que se prenden cuando la UC indica con el codigo de operacion, deja pasar el resultado de acuerdo al decodificador
  - UC
  - Bus de direcciones (20 bits)
  - Memoria → 8 registros
    - Primeros 4 (8 bits): registros de uso general, sirven a la ALU
      - **Ax:** acumulador → aca guardo el dato
      - **Bx:** base → maneja la paginación
      - **Cx:** contador → ciclos, cuantas veces use algo
      - **Dx:** data → para todo lo demás (auxiliar)
    - Últimos 4: sirven a la UC
      - **SP:** stack pointer → marca la dirección del capitulo donde inicia la pila
      - **BP:** base pointer → apunta a la pagina del SP
      - **DI:** destination index → a donde va el dato (donde lo

guardo)
        - **SI:** source index → de donde viene el dato
- **BIU = bus interface unit**
    - Instruction queue: cola de instrucciones → pipeline
    - Se encarga de la salida
    - Registros
        - Segmentos: (capítulos)
            - CS: code segment → si lo incremento cargo la próxima instrucción en la cola de instrucciones
            - DS: data segment → donde esta el dato [se combina con DI y SI]
            - SS: stack segment → marca el segmento de la pila (del 1 al 16) [se combina con SP y BP]
            - ES: extra segment → auxiliar
        - Offset: (paginas)
            - IP: instruction pointer → puntero de pila, de instrucción (próxima instrucción) [se combina con CS para llenar la cola de instrucciones]

SUMA DESPLAZADA

- Para correrlo hacia la izquierda debo multiplicarlo por la base (binario, decimal. Hexa)
- Tengo 16 bits para significar un segmento y 16 para significar el offset.
    - ➔ Seg: A B C D x16 = A B C D 0
    - ➔ Off: 1 2 3 4 + 1 2 3 4
    - A C F 1 4
- Dirección lógica: A B C D 1 2 3 4
- Dirección física: A C F 1 4
- Ventajas: Permite proteger datos reales de la memoria para que no estén accesibles para el usuario. Desventajas: Tiempo de generación de la dirección física.

ARQUITECTURA BÁSICA Y ARQUITECTURA 8086

ARQUITECTURA BASICA	ARQUITECTURA 8086
RDIM: registro de instrucciones. Guarda la dirección de la instrucción en la que estamos **Salto o contador de programa:** Indicará la próxima dirección de la instrucción. Por lo cual el RDIM dependerá del estado anterior	CS: IP
RDAM: registro de almacenamiento de	DS: SI

direcciones de datos. Guarda la dirección del dato en el que estamos. RDIM y RDAM dependen del bus de direcciones	DS: DI SS: BP SS: SP

INTERRUPCIONES

- Interrupción = pedirle algo que influya en el funcionamiento normal
- Tipos
  - **Enmascarable:** dejo de hacer una cosa por otra más prioritaria → interrumpir la interrupción
  - **No enmascarable:** no las puedo interrumpir ni tocarlas

ACCESO A MEMORIA

DMA = direct access memory

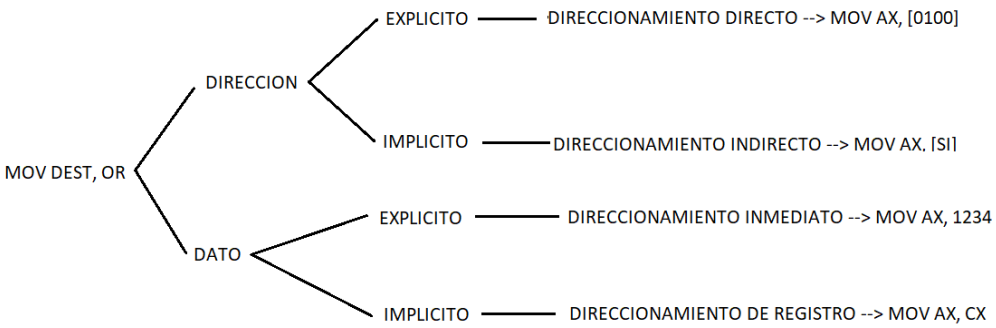
- Única función → transferir datos, pasa datos de dispositivo A a dispositivo B
- Que necesita?
  1. Dirección de memoria I/O L/E: de donde saco los datos, el origen
  2. Cantidad de bytes a transferir: cuantos mas tengo, mas tiempo tardo
  3. Interrupciones de los dispositivos I/O que intervienen en la operación
  4. Dispositivo destino L/E

ADMINISTRACIÓN DE DATOS: ENDIANNESS

➔ Formas de guardar datos y leer:

- **Big endian:** primero guardo los más significativos (RISC, MAC)
- **Little endian:** primero guardo los menos significativos (intel, amd)
- **Mixed endian:** el procesador define de acuerdo a lo necesario (celulares, debido al poco storage, android)

MODOS DE DIRECCIONAMIENTO



- **Modo directo:** toma los datos directo de la memoria.
- **Modo implícito:** toma el dato de la posición marcado con [ ]
- **Modo inmediato**: tal registro guarda tal valor que pongo a mano

- **Modo registro:** el dato de origen está en otro registro

CONCEPTO DE ORTOGONALIDAD

- Propiedad de las unidades centrales de procesamiento
- Las instrucciones son ortogonales cuando se puede utilizar cualquier modo de direccionamiento en cualquier instrucción

CARACTERES ASCII

- American standard code for information interchange
- Binario, de 7 bits
- Bit 5 → hace la diferencia entre mayusculas y minusculas [SHIFT: lo cambia]

VARIABLES

- Tienen: nombre, tipo expresión y valor
- Tipo
  - 8b = byte
  - 16b = DW word
  - 32b = DD double word
  - 64b = DQ quadruple word
- Valor
  - Valor = valor
  - Nada = ??

DECODIFICAR UNA INSTRUCCIÓN

➔ **Decodificador de instrucción:** Habilita el circuito necesario para hacer la operación que se quiere realizar.

Campos de una instrucción:

- **OP** = código de operación → indica el tipo de instrucción
- **MD** = modo de direccionamiento → Especifica a la UC donde devolver el dato a la memoria, que hago con el dato de resultado, como lo trato (piso el dato, lo guardo en una posición nueva)
- **CDE** = código dirección efectiva → Donde va a impactar efectivamente el dato

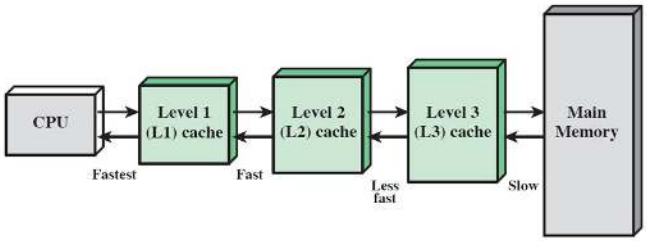
MEMORIA CACHE → memoria que acelera procesos

- Almacenamiento de datos de alta velocidad que almacena un subconjunto de datos,

normalmente transitorios, de modo que las solicitudes futuras de dichos datos se atienden con mayor rapidez que si se debe acceder a los datos desde la ubicación de almacenamiento principal.

- La memoria caché se divide en bloques, todos ellos contienen el mismo número de bytes. Cada bloque contiene un conjunto de datos ubicados en posiciones de memoria consecutivas. Cada vez que la CPU intenta acceder a una dirección de memoria que no está en ningún bloque de la caché se produce un **fallo** de caché, y en caso contrario un **acierto** de caché

Niveles



- **Cache L1**: son dos memorias → datos e instrucciones, que corren a lo mismo que el procesador
- **Caché L2:** datos e instrucciones todo junto

➔ Cuanto más cerca del procesador, más rápido

Como funciona:

- Cuando iniciamos un proceso o programa en nuestro ordenador, este empieza a ejecutar una serie de instrucciones que son gestionadas por el procesador. Esa información se carga primero en la RAM y luego pasa al procesador, de manera que para agilizar este proceso, las instrucciones principales y las que más se utilizan se copian en la memoria caché, así el procesador podrá tener acceso inmediato a ellas y no tener que ir a buscar las originales almacenadas en la RAM.
- Podemos decir que la memoria caché funciona como una memoria de instrucciones transitoria.

Objetivo:

Aumentar el rendimiento de recuperación de datos para evitar tener que acceder a la capa subyacente de almacenamiento, que es más lenta.

Tipos

- **Caché de disco:** Se trata de utilizar una porción de memoria RAM asociada a un disco particular
- **Caché de pista:** Es un tipo de memoria caché sólida (similar a la RAM), que se emplea en supercomputadores.
- **Caché de web:** Es la memoria caché que se ocupa de guardar los datos de las páginas web que hemos visitado de manera reciente, así se agiliza la carga en siguientes visitas y se ahorra ancho de banda.

Correlación entre datos de memoria y caché

- **Correlación directa:** Cada dato se almacena en solo una posición o bloque del caché → es muy rápido porque no usa todos los datos, el TAG es chico, no tengo política de reemplazo
- **Correlacion dinámica:** Aprovecho el lugar, importa el orden de llegada, guardo el primero que llega en la primera posición, cuando se acaban los lugares reemplazo al primero, el TAG es toda la direccion
- **Correlación asociativa por conjunto:** aprovecho el espacio y dentro de cada conjunto me comporto dinámicamente → parto mi memoria en conjuntos
    - Política de reemplazo: se va el que entró primero

RISC Y CISC

- CISC son los procesadores Intel Core y AMD Ryzen
- RISC son los utilizados para smartphones (Qualcomm Snapdragon y Samsung Exynos)

RISC agiliza y acelera significativamente el procesamiento de datos al minimizar el número de instrucciones almacenadas permanentemente en el microprocesador y se basa más en instrucciones no residentes, es decir, códigos o programas de software. Es posiblemente la tecnología de microprocesador más rápida y eficiente disponible en la actualidad.

Risc: (cálculo de conjuntos de instrucciones reducidos)

- Instrucciones pequeñas definidas
- Compilador complejo
- Se requiere **un ciclo de reloj** para ejecutar una instrucción. Cada ciclo de reloj incluye un método de obtención, decodificación y ejecución de la instrucción. (Tiempo fijo)
- Mismo tamaño de instrucciones
- Técnica de canalización para ejecutar etapas de instrucciones para más eficiencia.
- Se encuentran **muchos** GPR (registros generales) para almacenar instrucciones
- Modo de direccionamiento simple
- Usa instrucciones **LOAD** y **STORE** para acceder a la memoria
- **Muchos** pipeline
- Operaciones, control de flujo (estado del programa)
- Las operaciones aritméticas y lógicas sólo utilizan operandos de registro.

VENTAJAS:

- Ofrecen mejor rendimiento por el menor número de instrucciones
- Requieren de menos transistores (los hace mas economicos)

- Menos consumo de energía y generan menos calor.

DESVENTAJAS

- El rendimiento del procesador depende del código que se ejecuta, ya que las instrucciones posteriores que se ejecuten pueden depender de una instrucción anterior.
- Actualmente la mayoría de software y compiladores hacen uso de instrucciones complejas.
- Necesitan de memorias muy rápidas para almacenar diferentes cantidades de instrucciones, que requieren de una gran cantidad de memoria caché para responder a la instrucción en el menor tiempo posible
- Las instrucciones RISC no permiten la transferencia directa de memoria a memoria; requiere instrucciones de carga y almacenamiento para hacerlo.

Cisc: (cálculo de conjuntos de instrucciones complejas)

- Microcodigo
- Diferentes tamaños de instrucciones
- Requiere varios ciclos de reloj para ejecutar una instrucción (Tiempo variable)
- Se encuentran **pocos** GPR
- **Pocos** pipeline
- Las operaciones aritméticas y lógicas se pueden aplicar tanto a la memoria como a los operadores de registro.

VENTAJAS

- Codigo pequeño → baja necesidad de memoria RAM
- El compilador se requiere de poco esfuerzo
- Se permite ajustar la velocidad y el voltaje del reloj

DESVENTAJAS

- Pueden requerir varios ciclos de reloj para completar una instrucción
- Requiere muchos más transistores
- Más generacion de temperatura, mayor consumo y mayor requisito de espacio físico