# ARQUITECTURA DEL COMPUTADOR

## 1 Definición

El concepto de la arquitectura de las computadoras consiste en un conjunto de técnicas que permiten construir máquinas lógicas generales programables en forma práctica. En materia terminológica se distingue la "arquitectura" de un computador de la "organización" de un computador. Se puede decir que la **arquitectura** es la visión funcional (el conjunto de recursos que "ve" el programador), mientras que la **organización** es la forma en que se construye una cierta arquitectura en base a circuitos lógicos.

## 2 Arquitectura de von Neumann

### 2.1 Introducción

Esta arquitectura fue la propuesta por el matemático John von Neumann para la construcción de la computadora EDVAC en 1945 (la máquina se terminó de construir en 1949), sucesora de la que se considera la primer computadora electrónica, la ENIAC (1946).

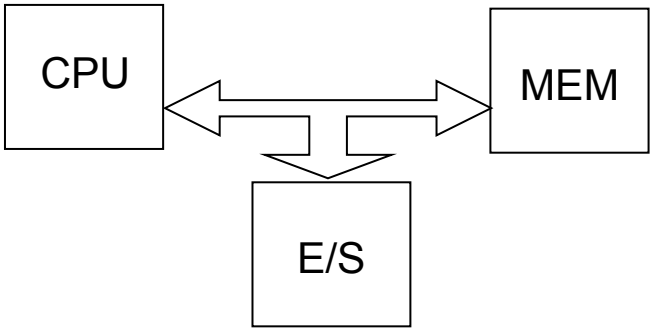
Los conceptos que propuso von Neumann han tenido una vigencia mucho más allá de la esperada en una industria como la de las tecnologías de la información. Por más que llame la atención, todos los computadores modernos disponibles comercialmente poseen en el fondo la misma arquitectura que la EDVAC, definida en 1945. Mucho se ha adelantado en materia de los circuitos lógicos (la organización) que implementan la arquitectura, pero casi nada en los principios de diseño que planteó von Neumann hace más de 60 años (que en materia tecnológica representan más de una decena de generaciones completas).

Se puede decir que la idea fundamental de von Neumann se apoya en el hecho que una operación compleja normalmente se puede dividir como una secuencia ordenada de operaciones más simples. En otras palabras lo que propuso fue construir una máquina capaz de ejecutar algoritmos en forma explícita. Para ello introdujo el concepto de "programa almacenado" como una "secuencia lógicamente ordenada de instrucciones", siendo las "instrucciones" las operaciones básicas que implementa el hardware a través de sus circuitos lógicos.

Una Arquitectura de von Neumann tiene tres bloques constructivos básicos: la Unidad Central de Proceso (ó CPU por su sigla en inglés), la Memoria y la Entrada/Salida.

Las funciones de cada bloque son:

- **CPU**: se encarga de ejecutar los programas.
- **Memoria**: almacena el programa (conjunto de instrucciones ordenado lógicamente) y los datos (operadores y resultados de la ejecución de las instrucciones).
- **Entrada/Salida**: comunica el computador con el mundo exterior, permitiendo la interacción con los usuarios y con otras computadoras.



En temas posteriores veremos en detalle como se implementa la **Memoria,** la **CPU** y el sub-sistema de **Entrada/Salida**.

Estos tres sub-sistemas se interconectan por medio de un **bus**, que contiene líneas de **datos**, de **dirección** y de **control**.

## 2.2 Caracterización de una Arquitectura von Neumann

Como ya expresamos todas las computadoras actuales tienen en esencia la misma arquitectura. Sin embargo dentro de la idea general de von Neumann que todas respetan, se diferencian entre sí por decisiones de diseño que afectan la cantidad y cualidad de sus elementos componentes.

Así una arquitectura particular (ej: Intel x86, PowerPC, SPARC, MIPS, etc) establece en forma diferenciada los siguientes elementos característicos, los que deben ser conocidos por los programadores "de bajo nivel" para poder escribir programas para una de esas arquitecturas:

- **Set de Instrucciones**: la cantidad de instrucciones disponibles y la calidad y complejidad de las operaciones implementadas en el hardware de la CPU.
- **Formato de Instrucción**: la forma en que se codifican las instrucciones.
- **Set de Registros**: la cantidad de registros disponibles al programador, así como la función que pueden cumplir (ej: origen o destino de operaciones, almacenamiento de direcciones de operandos).
- **Modos de Direccionamiento**: formas de generar las direcciones para hallar los operandos o almacenar los resultados de las operaciones.
- **Manejo de la Entrada/Salida**: forma de comunicación con los "periféricos" (dispositivos que implementan la interacción con el mundo exterior).
- **Manejo de Interrupciones**: manejo de una forma particular de invocar a ciertas sub-rutinas de los programas que estudiaremos mas adelante.

### 2.2.1 Set de Instrucciones

Históricamente desde la EDVAC hasta principios de la década del 80 la tendencia fue ir, progresivamente, aumentando la cantidad de instrucciones disponibles en el hardware y su complejidad. Es así que las arquitecturas más difundidas a fines de los 70 (Intel 8080, Motorola 6800, Digital VAX, IBM 370) disponían de cientos de instrucciones y soportaban por hardware operaciones complejas tales como la comparación de strings y la búsqueda de un elemento en un array.

En ese momento se pensaba que cuanto más "potente" era el hardware (en cuanto a la variedad de operaciones disponibles), más eficientemente se podrían ejecutar los programas, logrando una mayor "performance" del sistema.

Sin embargo en 1980 se publicaron, casi simultáneamente, trabajos de investigación de las universidades de Berkeley y Stanford (ambas en California, EE.UU.) que proponían un enfoque radicalmente distinto: lo mejor era disponer de un conjunto mínimo de instrucciones que estuvieran implementadas en forma óptima. Estos trabajos acuñaron el término **RISC** (**R**educed **I**nstruction **S**et **C**omputer) para referirse a los diseños basados en este concepto. A partir de ese momento el resto de las arquitecturas pasaron a denominarse, por contraposición, **CISC** (**C**omplex **I**nstruction **S**et **C**omputer).

Durante muchos meses corrieron ríos de tinta en publicaciones de todo tipo

discutiendo cuáles arquitecturas eran mejor: si las basadas en la filosofía RISC o las CISC. La historia de esta batalla "tecnológica" tiene aspectos muy interesantes y un resultado doblemente paradójico: si bien las primeras implementaciones prácticas de RISC (incluyendo un chip diseñado por un grupo de estudiantes de Berkeley) demostraron que las RISC eran mucho mas eficientes (considerando su desempeño en relación con la frecuencia del reloj utilizado), el mercado siguió siendo de las CISC, en particular de la que se convirtió en un estándar de facto: la arquitectura x86 de Intel (la de los actuales procesadores “Pentium” y “Core”). La segunda paradoja es que para lograr los niveles actuales de desempeño, Intel debió incorporar conceptos de RISC en su diseño. Hoy se puede decir que un procesador Core es, en el fondo, una máquina RISC que ejecuta un conjunto de "micro-rutinas" que "emulan" las instrucciones de una máquina de arquitectura x86.

### 2.2.2 Formato de Instrucciones

El formato de las instrucciones refiere a la codificación de las distintas instrucciones para su almacenamiento en la memoria del sistema.

Al igual que para los demás tipos de datos manipulados que vimos (caracteres, números) los computadores trabajan con una representación de las instrucciones mediante un código binario. Un código de instrucción es un grupo de bits que instruye a la computadora sobre cómo ejecutar una operación específica.

El código binario reserva una serie de bits para identificar la operación realizada por la instrucción, otros indican los operandos a utilizar y sus direcciones, así como la indicación de dónde se almacena el resultado.

El código de operación de una instrucción es un grupo de bits que define operaciones como sumar, restar, multiplicar, desplazar y complementar. El número de bits requeridos para el código de operación de una instrucción depende de la cantidad total de operaciones disponibles en la computadora. El código de operación debe estar formado de por lo menos n bits para un conjunto dado de $2^n$ (o menor) operaciones diferentes. Como ejemplo, consideremos una computadora con 64 operaciones diferentes, una de las cuales es la operación suma (ADD). El código de operación consta de seis bits, con una configuración de bits 110010 asignada a la operación ADD.

La parte operativa de un código de instrucción especifica la operación que se va a realizar. Esta operación debe ejecutarse sobre algunos datos almacenados en los registros del procesador o en la memoria. Por lo tanto, un código de instrucción debe especificar no sólo la operación, sino también los registros o palabras de la memoria donde se van a encontrar los operandos, al igual que que el registro o la palabra de memoria donde se va a almacenar el resultado. Pueden especificarse palabras de memoria en los códigos de instrucción mediante sus direcciones. Pueden especificarse registros del procesador al asignar a la instrucción otro código binario de k bits que especifique uno de $2^k$ registros.

Existen muchas variaciones para “componer” el código binario de instrucciones, y cada computadora tiene su propio formato de código de instrucciónes particular. Los atributos que definen el formato de las instrucciones incluyen aspectos tales como si los códigos binarios asociados son de largo fijo o variable y si las instrucciones tienen operandos y destino independientes (en cuyo caso se habla de "arquitectura de 3 direcciones") ó solapados (correspondiente a "arquitectura de 2 direcciones").

Las arquitecturas RISC promocionaron la utilización de formatos de instrucción de largo fijo, como forma de simplificar la circuitería de decodificación y permitir la utilización

eficaz de técnicas de optimización (pipeline, superescalaridad). Las arquitecturas CISC normalmente tienen instrucciones de largo variable.

Las arquitecturas RISC también utilizan formatos de “3 direcciones”, mientras que en las CISC es más común encontrar implementaciones de “2 direcciones” (como es el caso de Intel x86).

### 2.2.3 Set de Registros

Los registros son, como mencionamos, posiciones "especializadas" de memoria, ubicadas dentro de la propia CPU y que poseen una manera de direccionarlas distinta de la memoria "normal" del sistema.

Los primeros diseños siguieron el ejemplo de la EDVAC de von Neumann y utilizaron registros "con personalidad". En la propuesta original de von Neumann la computadora poseía, entre otros, un registro **Acumulador** para las operaciones aritméticas, un registro **Contador** para operaciones que implicaran contar y un registro **Indice** para ser utilizado para contener direcciones de operandos en memoria. Esas ideas son las que llevaron a Intel a nombrar los registros de su primer microprocesador 8080 con letras: A (Acumulador), B (Base, para direcciones), C (Contador, en instrucciones de string), D (Data, para almacenar datos, es decir operandos en general) y así sucesivamente. Luego la arquitectura x86 de 16 bits tomó esos nombres y le agregó una X (de eXtended, en referencia a que pasaron de ser de 8 bits a 16 bits), quedando AX, BX, CX y DX, a los que se agregaron SI (Source Index) y DI (Destination Index) entre otros.

La característica de los registros "con personalidad" es que su función dentro de las instrucciones (como operandos fuente, operandos destino, contadores o parte del direccionamiento de los operandos) está condicionada y no todos sirven para cualquier función. Su aplicabilidad es restringida y específica.

Las arquitecturas RISC promovieron desde el comienzo el uso de registros "despersonalizados", de uso general, aplicables todos y cada uno a cualquiera de las funciones antedichas. Intel incorporó la idea a partir del procesador 80386, primer representante de 32 bits de su familia arquitectónica. Los registros de 32 bits de la arquitectura x86 se denominan EAX, EBX, ECX, EDX, ESI, EDI, etc (la "E" significa "Enhanced") y pueden intercambiarse casi completamente en su uso en las instrucciones.

### 2.2.4 Modos de Direccionamiento

Los modos de direccionamiento establecen las formas en que se puede, a nivel de las instrucciones, especificar la dirección de un operando o del lugar donde colocar el resultado de la operación correspondiente a la instrucción.

Existen tres modos de direccionamiento básicos: Inmediato, Directo e Indirecto. A continuación presentaremos las características de cada uno:

- **Inmediato:** en este modo en la instrucción se encuentra el propio operando (su valor). Se utiliza típicamente para constantes (ya que las instrucciones no se pueden modificar).
- **Directo:** en este modo en la instrucción se encuentra la dirección del operando. Se pueden distinguir dos tipos:
  - Directo a Registro: el operando está almacenado en un registro y la instrucción contiene el identificador del registro
  - Directo a Memoria: el operando está almacenado en memoria y la instrucción contiene la dirección donde se encuentra

- **Indirecto:** en este modo en la instrucción se encuentra la dirección del lugar donde se encuentra la dirección del operando. También se pueden distinguir dos tipos:
  - Indirecto por Registro: el operando está almacenado en una posición de memoria cuya dirección se encuentra en un registro y la instrucción contiene el identificador del registro. De los dos tipos de indirecto éste es el habitualmente implementado, utilizando mas de un registro (sumando sus contenidos) para formar la dirección. En algunos casos la arquitectura prevé multiplicar el contenido del registro por el tamaño en bytes del operando. En ese caso se habla de direccionamiento **indizado**.
  - Indirecto por Memoria: el operando está almacenado en una posición de memoria cuya dirección está en otra posición de memoria y la instrucción contiene la dirección de esta última. Esta variante no es implementada en las arquitecturas prácticas disponibles.

Los modos Directo e Indirecto (especialmente en su modalidad Indizado) se suelen combinar para establecer la dirección de un operando o de un destino. La dirección se forma sumando el contenido de uno o mas registros identificados en la instrucción con la "dirección base" contenida en la misma. Esto permite implementar fácilmente el acceso a estructuras tipo array. Se coloca la dirección de comienzo de la estructura en la instrucción (modo directo) y se cambia el contenido de un registro "índice" (que también debe estar referenciado en la instrucción) para ir recorriendo el array haciendo un "loop" y reutilizando la misma instrucción.

### 2.2.5 Manejo de Entrada/Salida

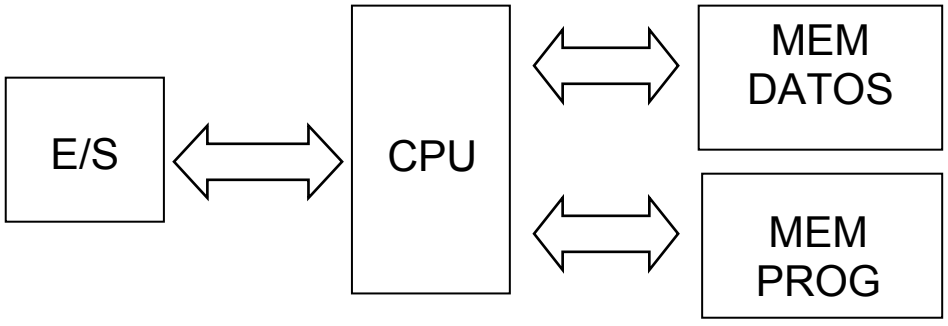
Más adelante veremos un capítulo completo dedicado a este tema. Por ahora interesa marcar que las distintas arquitecturas se distinguen en este punto básicamente por el hecho de tener o no instrucciones específicas, con su propio espacio de direccionamiento, para la comunicación con los dispositivos que permiten la interacción con el "mundo exterior".

### 2.2.6 Manejo de Interrupciones

Nos referiremos a este tema en un capítulo específico.

## 3 Arquitectura Harvard

La arquitectura Harvard se caracteriza porque existen, a diferencia de la von Neumann, dos unidades de memoria separadas: una para los datos y otra para las instrucciones:



Muchas veces se presenta la arquitectura Harvard como algo diferente de la arquitectura von Neumann. Sin embargo en el fondo no cambia la esencia de la von Neumann: siguen existiendo “programas” formados por una secuencia lógica de “instrucciones” las que son ejecutadas por una “CPU”.

Actualmente muchos procesadores implementan una ***arquitectura Harvard modificada***, que consiste en mantener ciertos caminos separados y paralelos para acceder a las instrucciones en forma simultánea con los datos, pero permitiendo que los datos y las instrucciones están almacenados en una memoria común. Los detalles de estas implementaciones se verán más adelante en el curso.