



Módulo 01

Introducción (Pt. 2)



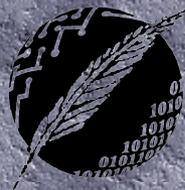
Organización de Computadoras
Depto. Cs. e Ing. de la Comp.
Universidad Nacional del Sur



Copyright

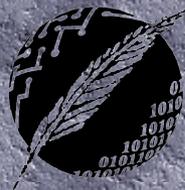
- Copyright © 2011-2024 A. G. Stankevicius
- Se asegura la libertad para copiar, distribuir y modificar este documento de acuerdo a los términos de la **GNU Free Documentation License**, Versión 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera
- Una copia de esta licencia está siempre disponible en la página <http://www.gnu.org/copyleft/fdl.html>
- La versión transparente de este documento puede ser obtenida de la siguiente dirección:

<http://cs.uns.edu.ar/~ags/teaching>



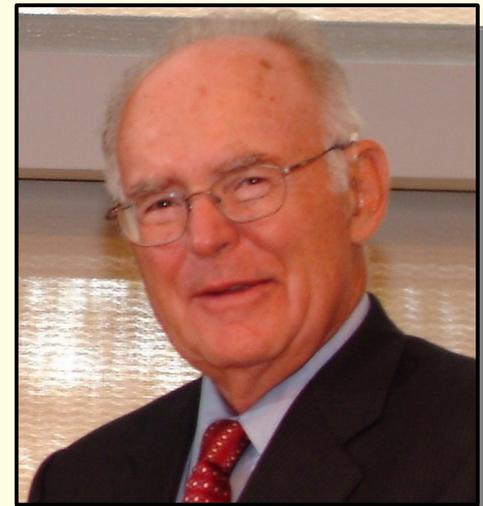
Contenidos

- Historia de la computación
- Arquitectura von Neumann
- Generaciones
- Ley de Moore
- Evolución de las arquitecturas
- Arquitecturas **CISC** vs. **RISC**
- Ciclo básico de un **CPU**
- Organización de la memoria



Ley de Moore

- Gordon Moore (1929-2023), fue uno de los fundadores de Intel, observó que la densidad de los circuitos integrados crecía con el tiempo
- Predijo que la cantidad de transistores en los chips iba a **duplicarse cada año**
 - ➔ Tan precisa fue esa predicción que hoy se la conoce como la **Ley de Moore**
- Desde la década del '70 este crecimiento se desaceleró:
 - ➔ ¡Ahora **sólo se duplica cada 18 meses!**



Transistores por CPU

- ¿Qué beneficio genera contar dentro de un mismo chip con más transistores?
 - Los transistores de los integrados **CMOS** cuanto más pequeños **resultan más veloces** y **disipan menos calor**
 - Al ser más pequeños todos los componentes, sus **interconexiones** son mas cortas, por ende pueden tener un mejor **desempeño**
 - El costo de fabricar un cm^2 de chip se ha mantenido relativamente constante en moneda dura, por lo que al aumentar el nivel de integración **baja el costo**



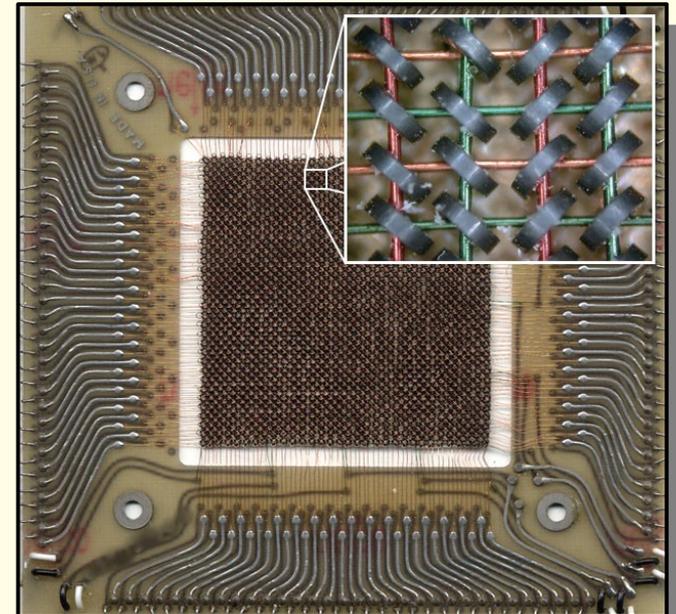
Ley de... ¿Alejandro?

- La ley de Moore nos permite contar con **CPUs** más veloces...
 - ➔ ¿Por qué razón más pequeño resulta más rápido?
- La forma más simple de recordarlo es teniendo en cuenta la siguiente aseveración:
 - “No se puede ir rápido y lejos al mismo tiempo”
- Volveremos a este principio numerosas veces a lo largo del resto de la carrera



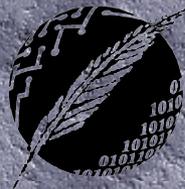
Evolución de la memoria

- La tecnología de la memoria principal de las primeras computadora era bien diversa
- Una de las alternativas más populares de la época (1955-1975) fue la **memoria de núcleo**
 - Cada bit se almacenaba en un pequeño toroide magnético
 - Se trata de una memoria **persistente**
 - El contenido **se destruye al leerlo**



Evolución de la memoria

- En la década del '70 finalmente se pudo aplicar la tecnología de los circuitos integrados a la producción de memoria
 - La tecnología de los semiconductores permitió empaquetar unos 256 bits en el espacio ocupado por un único toroide de la memoria de núcleo
 - Las lecturas no eran destructivas y además el ciclo de lectura era mucho más veloz
 - Eso si, se trata de una memoria dinámica, el contenido se pierde pasado un cierto tiempo

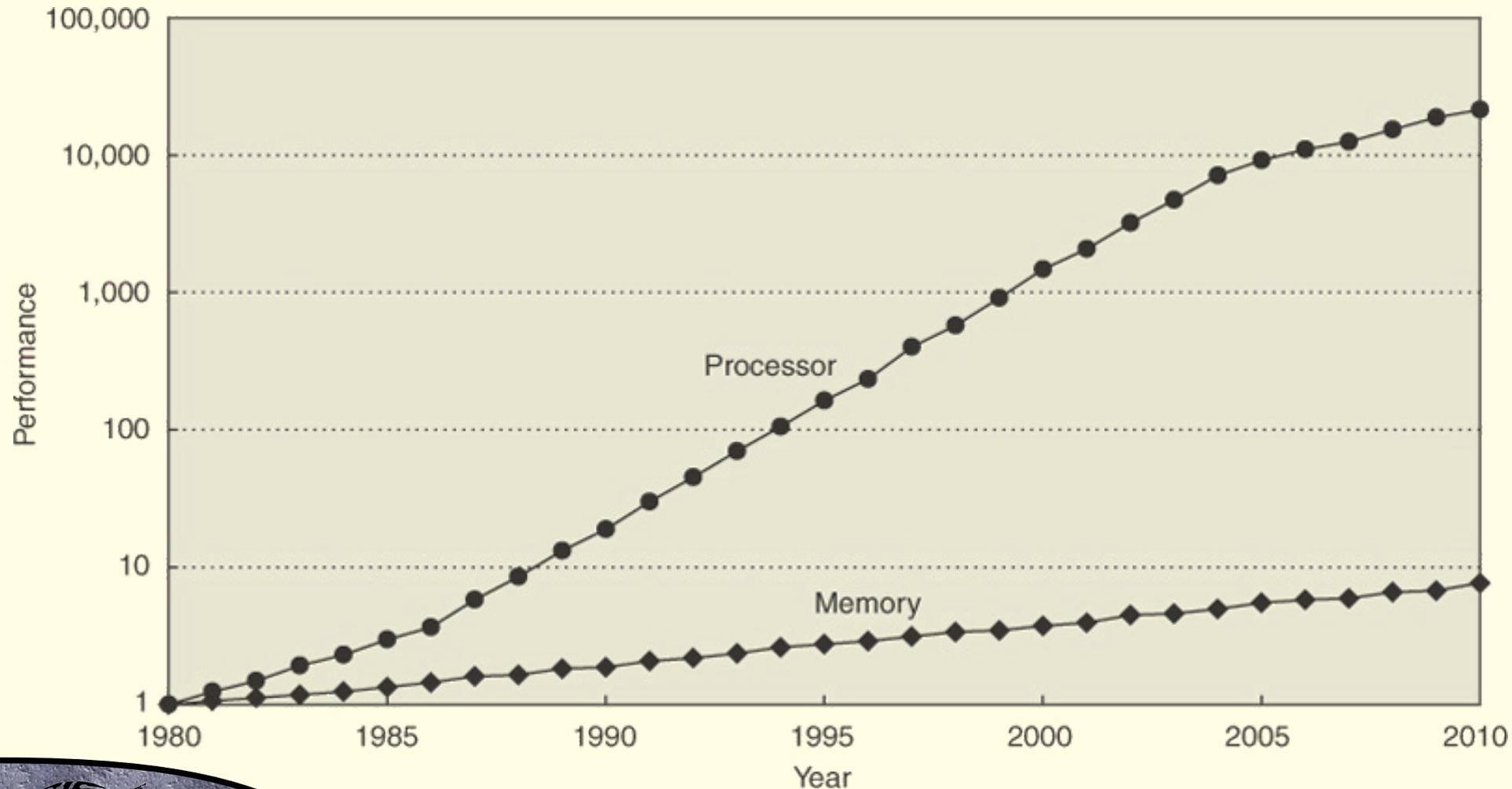


Evolución de la memoria

- Una vez implementada la memoria dentro de un circuito integrado, se pudo sacar provecho de los avances en el nivel de integración:
 - ➔ La capacidad de los módulos de memoria **se duplica cada unos 12 meses**
- No obstante, sólo la capacidad de los módulos se incrementa de forma exponencial, no así su velocidad de acceso
 - ➔ Existe una separación (un **gap**) cada vez mayor entre las velocidades del procesador y de la memoria



CPU vs. Memoria



Soluciones ensayadas

- A lo largo del tiempo se han ensayado distintas alternativas para atemperar el impacto de esta creciente diferencia de desempeño:
 - ➔ Traer más información a la vez (que los módulos de memoria tengan más pines o “patitas”)
 - ➔ Incorporar múltiples niveles de memoria cache
 - ➔ Minimizar los accesos a memoria (tarea tanto del programador como del compilador)
 - ➔ Mejorar la interconexión entre **CPU** y memoria (por ejemplo, haciendo uso de múltiples canales)



Intel

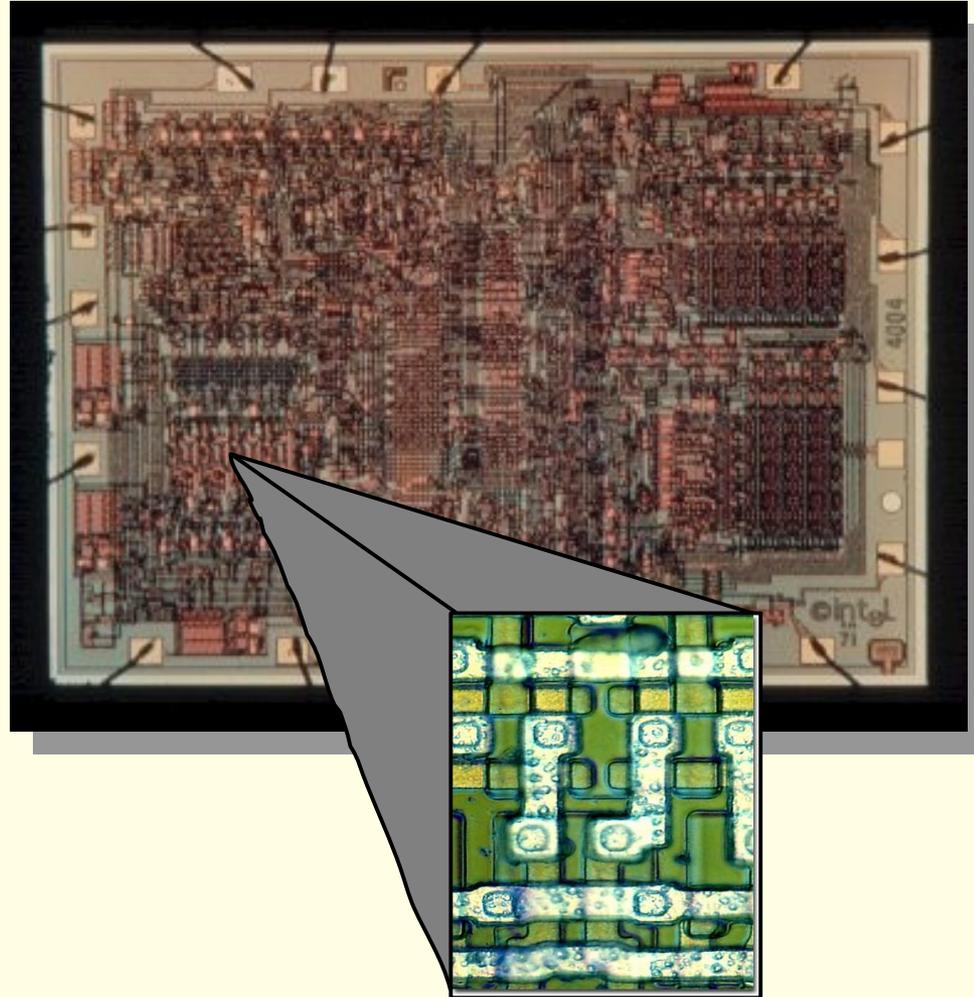
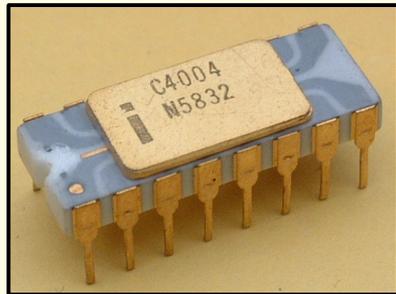
- Intel introdujo en 1971 el **i4004**, el **primer microprocesador**
 - Un microprocesador contiene todos los componentes del **CPU** dentro de un único chip
 - Se trata de una arquitectura de 4 bits
- En 1972 introdujo el **i8008**, de 8 bits
 - Tanto el **i4004** como el **i8008** eran de propósito específico
- Luego, en 1974 introdujo el **i8080**, el **primer microprocesador de propósito general**



Intel 4004 (1971)

● Características:

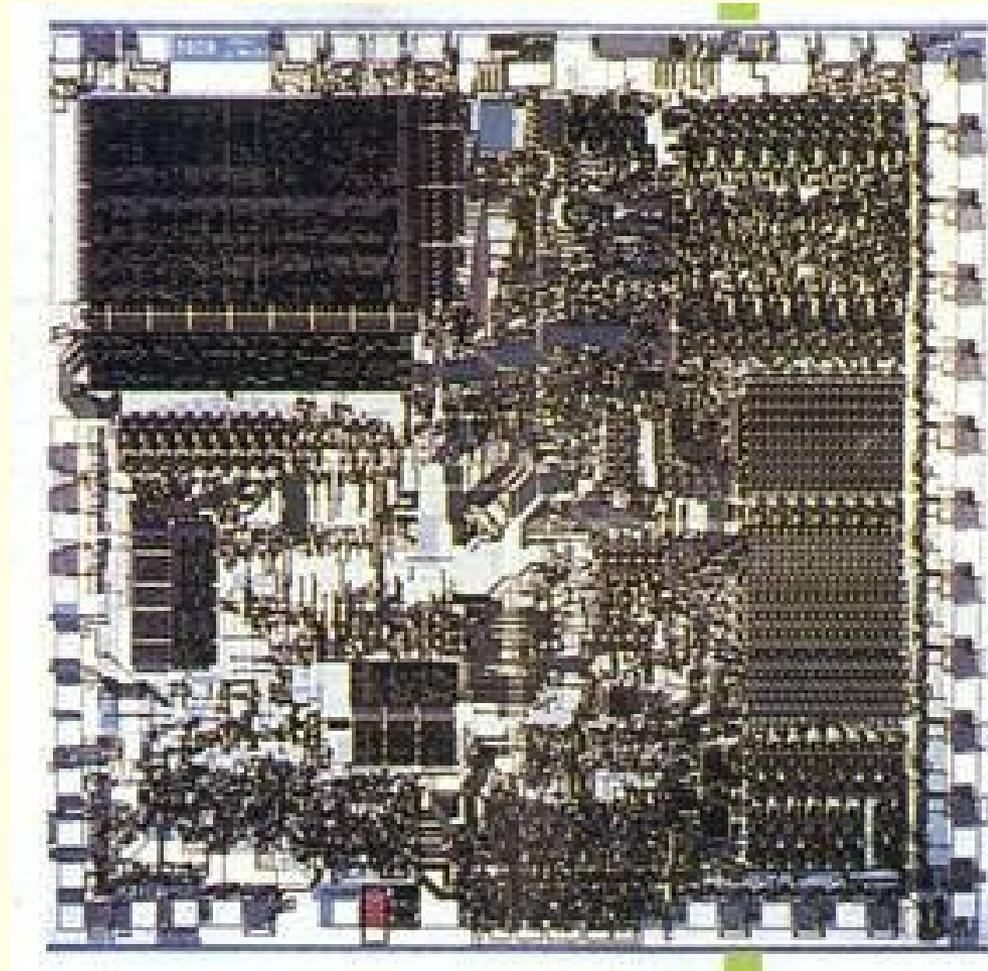
- ➔ Arquitectura de 4 bits
- ➔ Capacidad de memoria: 4 KB (12 bits)
- ➔ Transistores: 2.250
- ➔ Tamaño: 12 mm²
- ➔ Frecuencia: 108 KHz



Intel 8086 (1978)

● Características:

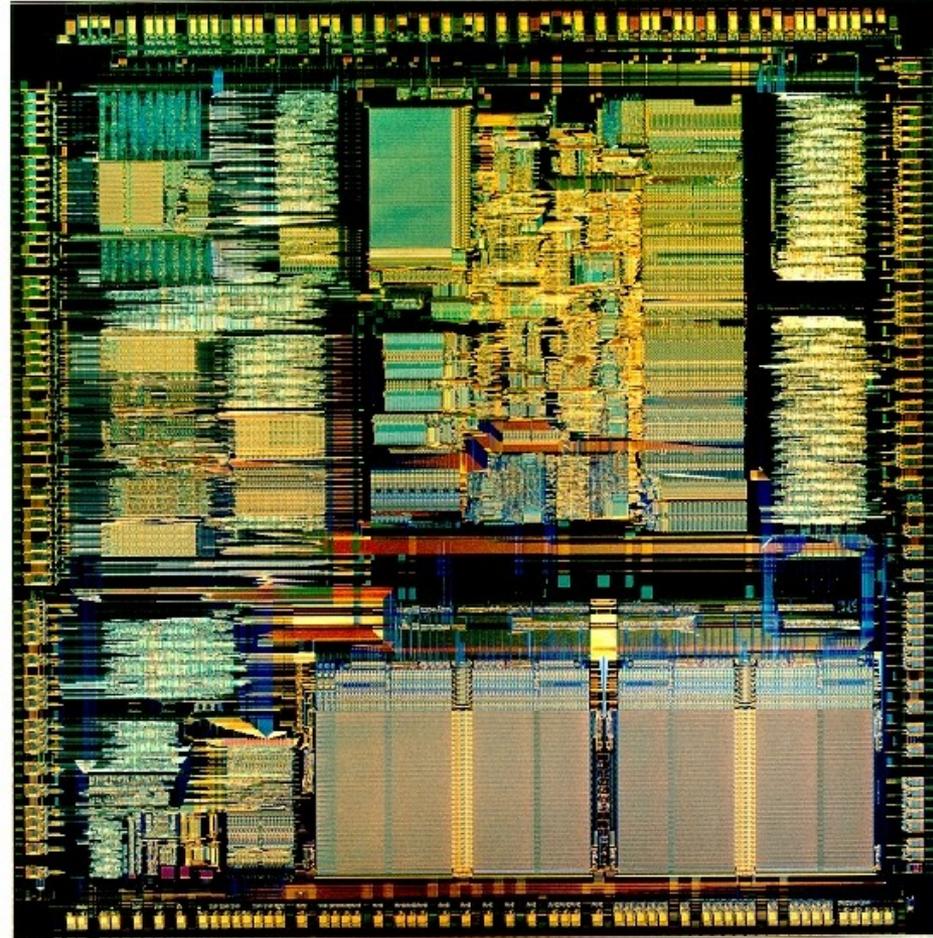
- Arquitectura de 16 bits
- Capacidad de memoria: 1 MB (20 bits)
- Transistores: 29.000
- Tamaño: 23 mm²
- Frecuencia: 5.000 KHz



Intel 80386 (1985)

● Características:

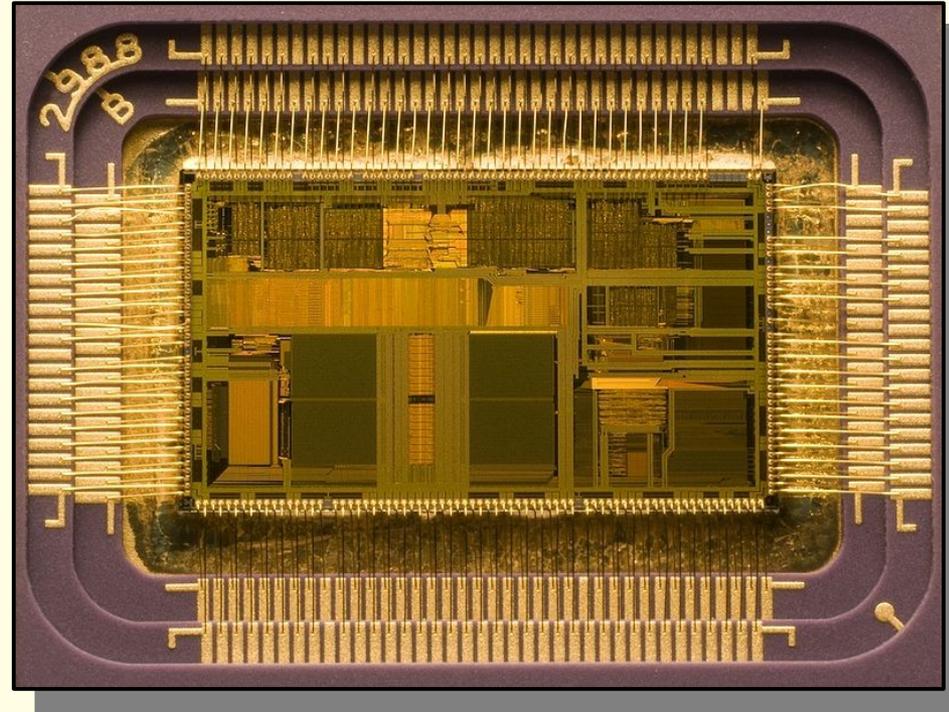
- Arquitectura de 32 bits
- Capacidad de memoria: 4 GB (32 bits)
- Transistores: 275.000
- Tamaño: 104-39 mm²
- Frecuencia: 12-33 MHz



Intel 80486 (1989)

● Características:

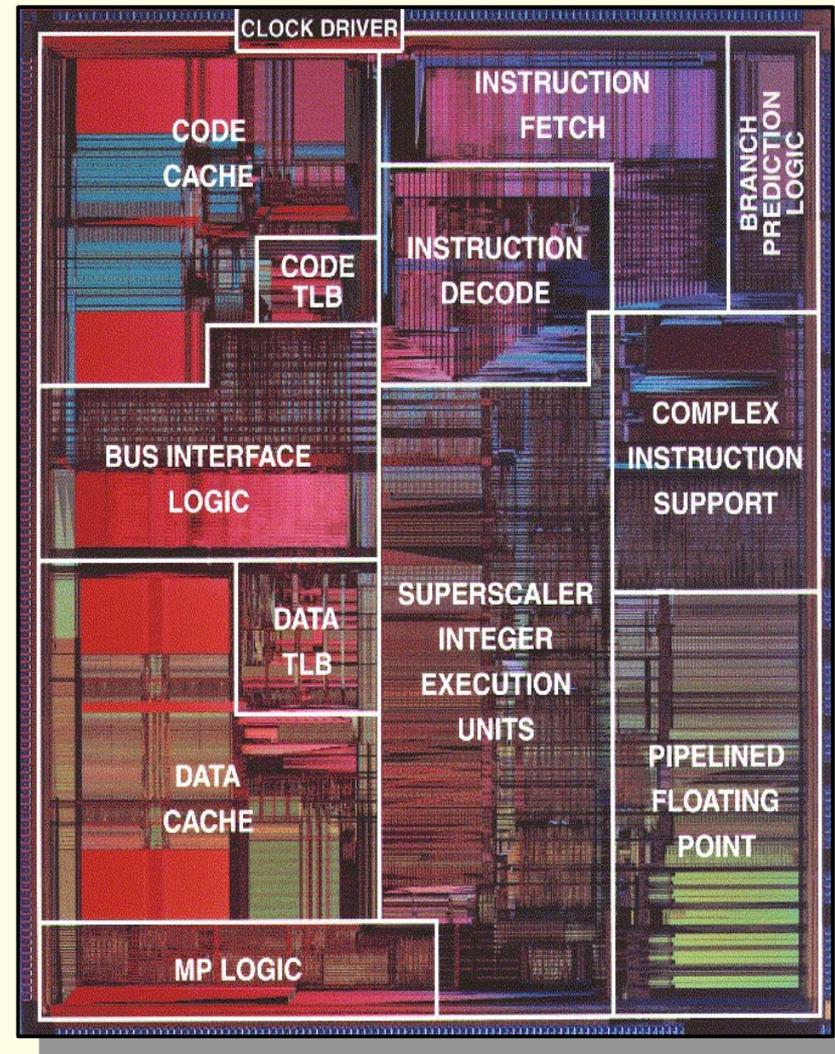
- Arquitectura de 32 bits
- Capacidad de memoria: 4 GB (32 bits)
- Transistores: 1.200.000
- Tamaño: 81 mm²
- Frecuencia: 16-100 MHz
- Cache: 8 KB
- Incluye por primera vez un **coprocesador matemático** en el mismo chip



Intel Pentium (1993)

● Características:

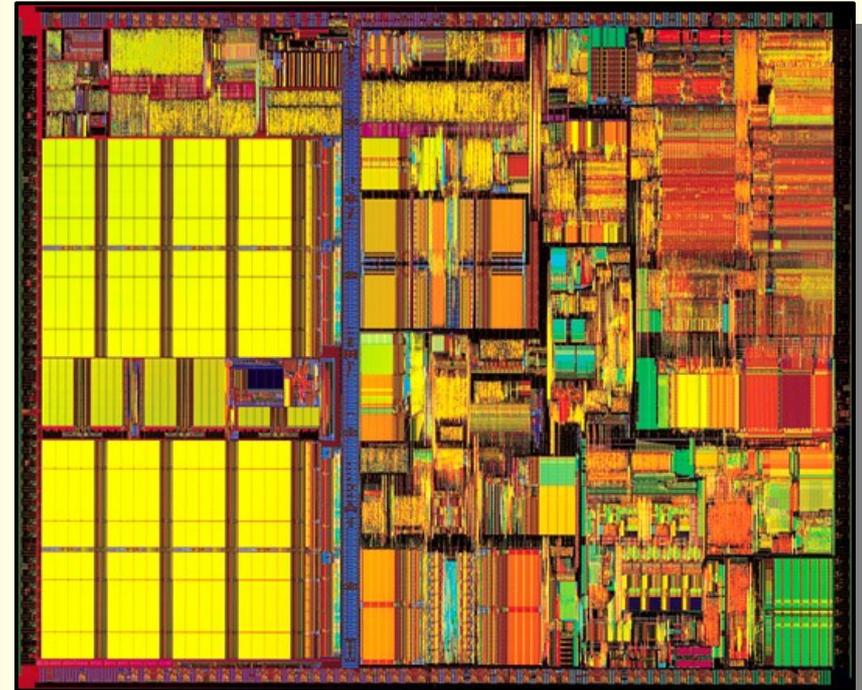
- Arquitectura de 32 bits
- Capacidad de memoria: 4 GB (32 bits)
- Transistores: 3.100.000
- Tamaño: 294 mm²
- Frecuencia: 60-233 MHz
- Cache: 16-32 KB
- Primer arquitectura **x86 superescalar**



Intel Pentium III (1999)

● Características:

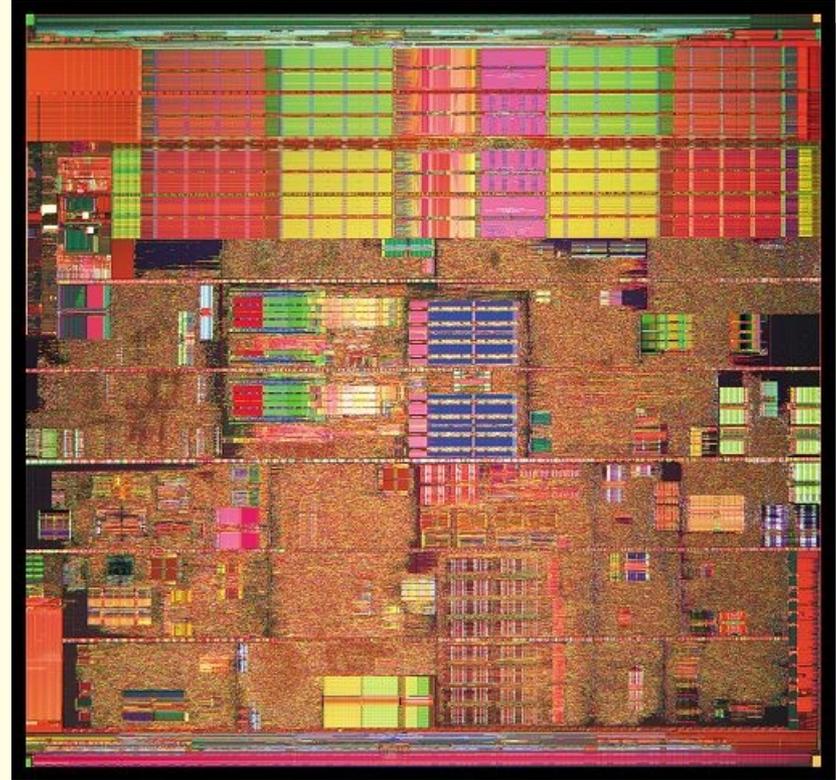
- Arquitectura de 32 bits.
- Capacidad de memoria: 4 GB (32 bits)
- Transistores: 9.500.000
- Tamaño: 125 mm²
- Frecuencia: 450-1400 MHz
- Cache: 256-512 KB



Intel Pentium IV (2000)

● Características:

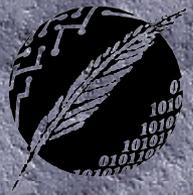
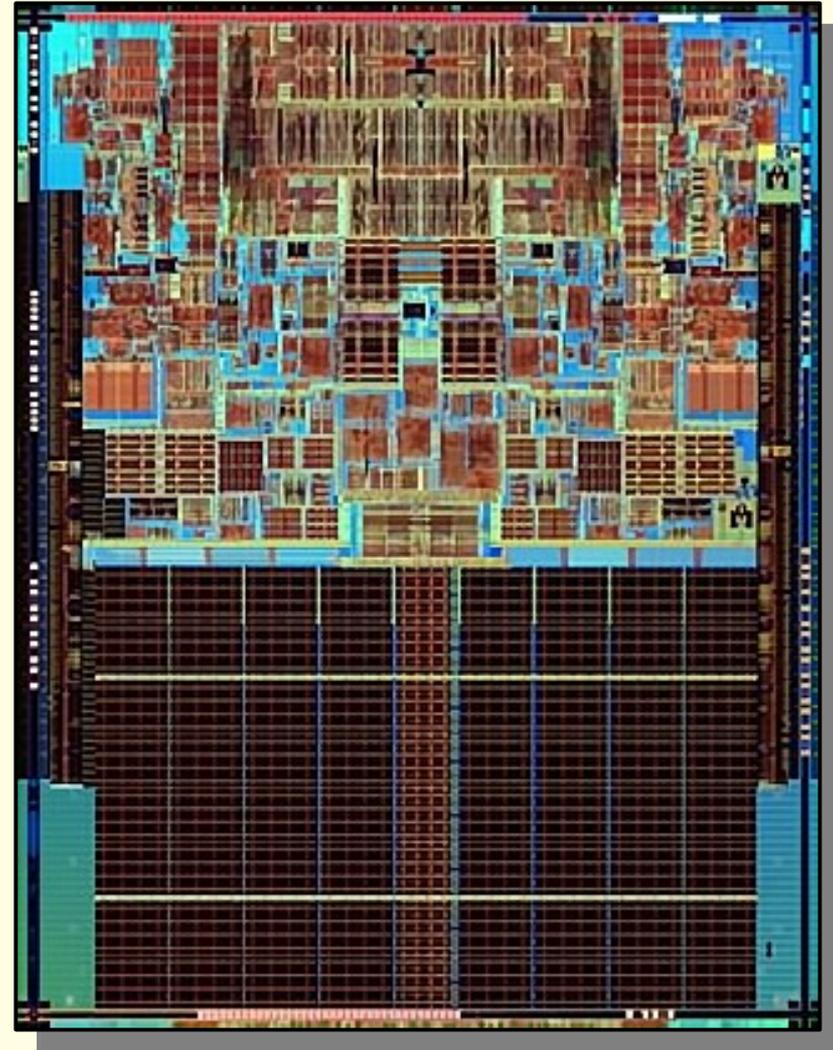
- Arquitectura de 32 bits
- Capacidad de memoria: 4 GB (32 bits)
- Transistores: 42.000.000
- Tamaño: 145 mm²
- Frecuencia: 1.3-3.8 GHz
- Cache: 256-2048 KB
- Primera implementación de la técnica de **hyperthreading**



Intel Core2 (2006)

● Características:

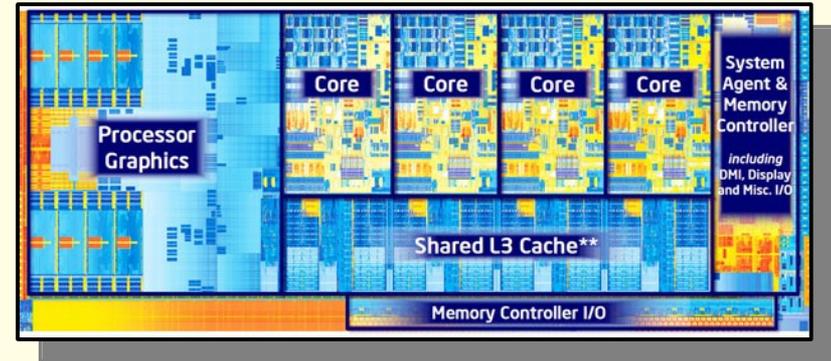
- Arquitectura de 64 bits
- Capacidad de memoria: 64 GB - 8 EB (36-64 bits)
- Transistores: 291.000.000
- Tamaño: 143 mm²
- Frecuencia: 1.06-3.5 GHz
- Cache: 1-12 MB
- Núcleos: 1, 2 ó 4



Intel Core i7 (2008)

● Características:

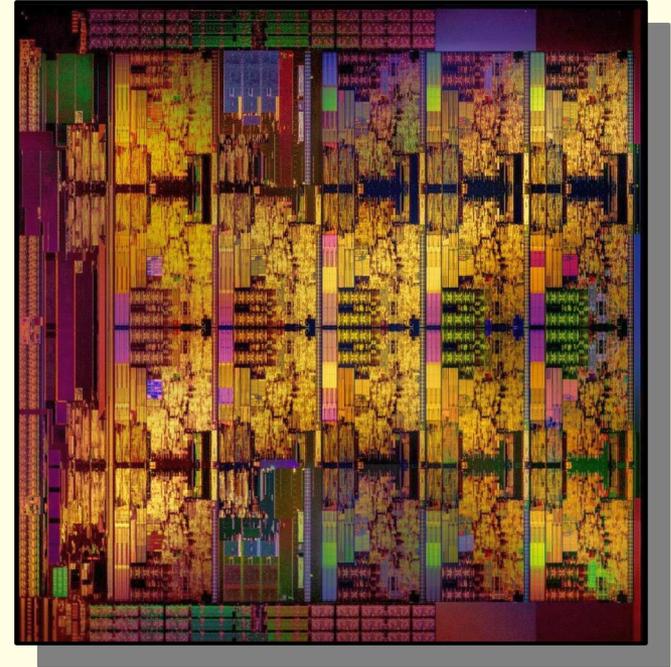
- Arquitectura de 64 bits
- Capacidad de memoria: 64 GB - 8 EB (36-64 bits)
- Transistores: 1.600.000.000
- Tamaño: 160 mm²
- Frecuencia: 1.06-3.5 GHz
- Cache: 5-15 MB
- Núcleos: 2, 4 ó 6



Intel Core i9 (2017)

● Características:

- Arquitectura de 64 bits
- Capacidad de memoria: 64 GB - 8 EB (36-64 bits)
- Transistores:
- Tamaño: 322-484 mm²
- Frecuencia: 2.6-6 GHz
- Cache: 24-68 MB
- Núcleos: 10-24



La catástrofe de Intel...



ARM Cortex-A9 (2007)

● Características:

- Arquitectura de 32 bits
- Capacidad de almacenamiento:
1 GB de RAM / 16 o 32 GB de datos
- Transistores: 26.000.000
- Tamaño: 31 mm²
- Frecuencia: 1.5 GHz
- Cache: 16-64 KB
- Núcleos: 2



A15 (2010) + A7 (2007)

● Características:

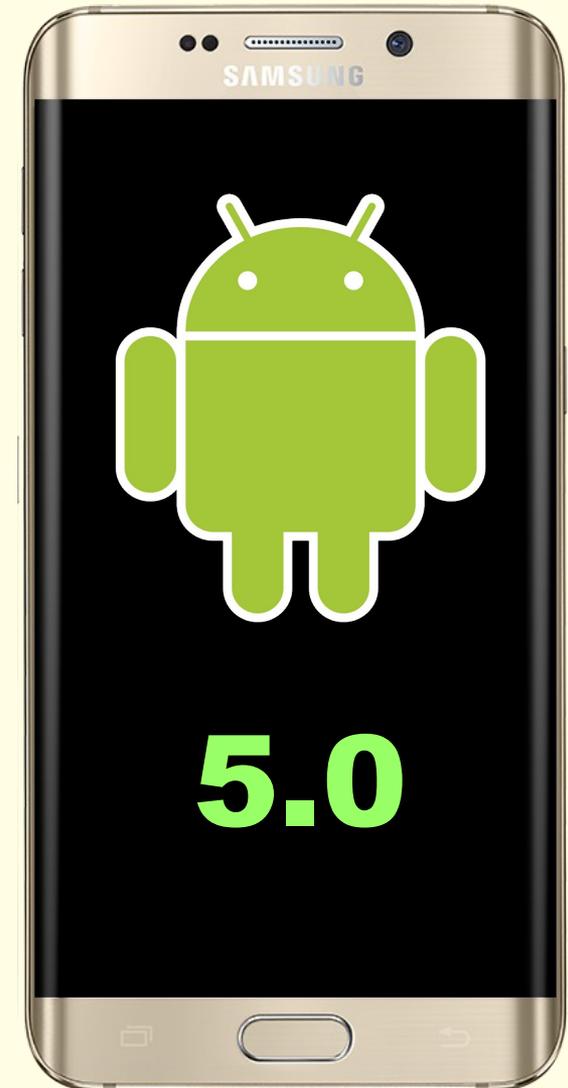
- Arquitectura de 32 bits
- Capacidad de almacenamiento:
2 GB de RAM / 16 o 32 GB de datos
- Transistores: itop secret!
- Tamaño: itop secret!
- Frecuencia: 1.6 GHz + 1.2 GHz
- Cache: 512 KB - 4 MB
- Núcleos: 4 + 4



A57 (2012) + A53 (2012)

● Características:

- Arquitectura de 64 bits
- Capacidad de almacenamiento:
4 GB de RAM / 32 o 64 GB de datos
- Transistores: itop secret!
- Tamaño: itop secret!
- Frecuencia: 2.1 GHz + 1.5 GHz
- Cache: 512 KB - 2 MB
- Núcleos: 4 + 4



M2 (2017) + A53 (2012)

• Características:

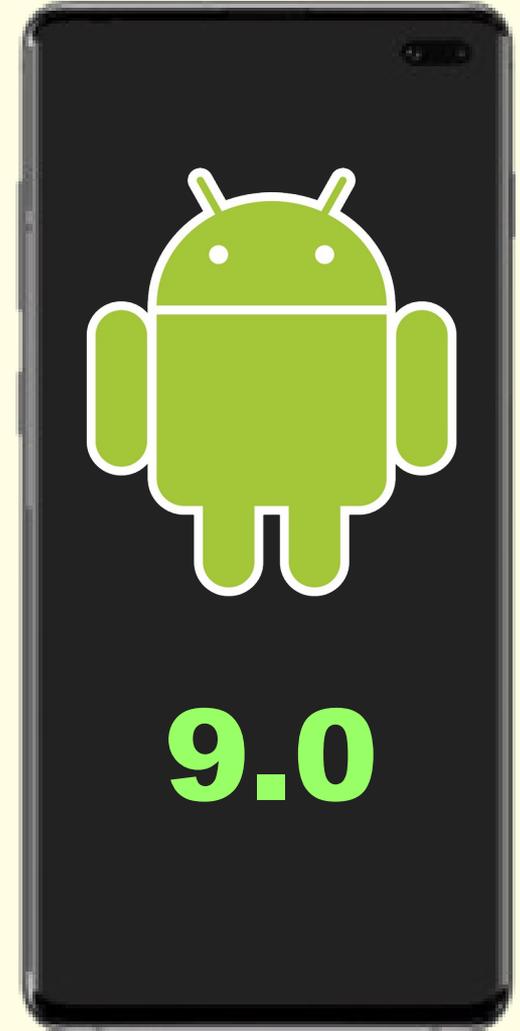
- Arquitectura de 64 bits
- Capacidad de almacenamiento:
4-6 GB de RAM / 64-128 GB de datos
- Transistores: itop secret!
- Tamaño: itop secret!
- Frecuencia: 2.3 GHz + 1.7 GHz
- Cache: 2 MB
- Núcleos: 4 + 4



M4 (2019) + A75 (2017) + A55 (2017)

● Características:

- Arquitectura de 64 bits
- Capacidad de almacenamiento:
8-12 GB de RAM / 128GB-1TB datos
- Transistores: itop secret!
- Tamaño: itop secret!
- Frecuencia: 2.73/2.71/1.95 GHz
- Cache: 2-4 MB
- Núcleos: 2 + 2 + 4



M5 (2020) + A76 (2018) + A55 (2017)

● Características:

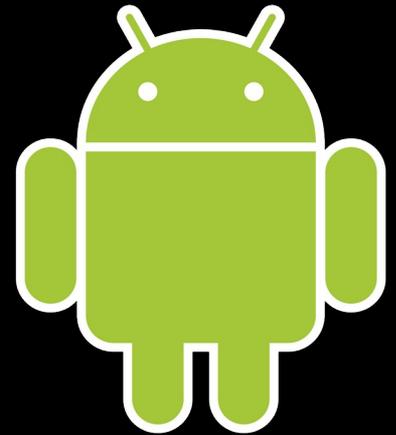
- Arquitectura de 64 bits
- Capacidad de almacenamiento:
8-16 GB de RAM / 128GB-512GB datos
- Transistores: itop secret!
- Tamaño: itop secret!
- Frecuencia: 2.73/2.60/2.00 GHz
- Cache: 2-4 MB
- Núcleos: 2 + 2 + 4



X2 (2021)+A710 (2021)+A510 (2021)

● Características:

- Arquitectura de 64 bits
- Capacidad de almacenamiento:
8-12 GB de RAM / 128GB-1TB datos
- Transistores: itop secret!
- Tamaño: itop secret!
- Frecuencia: 2.8/2.52/1.82 GHz
- Cache: 6 MB
- Núcleos: 1 + 3 + 4



10.0



Hardware vs. Software

- Las computadoras hoy en día se componen esencialmente de **hardware** y de **software**
- Denominamos hardware a todo componente **tangible**, es decir, que se puede tocar
 - Por caso, el procesador, la memoria, el teclado, etc.
- En contraste, denominamos software a los restantes componentes **intangibles**
 - Por caso, el sistema operativo, los programas de aplicación, etc.



Principio de equivalencia

- El principio de equivalencia nos brinda un puente entre el hardware y el software:
 - “Todo lo que pueda ser implementado a nivel de software también puede ser reimplementado a nivel de hardware y todo lo que pueda ser implementado a nivel de hardware también puede ser reimplementado a nivel de software”*
- La decisión de qué implementar en hardware y qué en software depende de otros parámetros:
 - ➔ Velocidad, costo, facilidad de mantenimiento, etc.



Microprogramación

- Una aplicación inmediata de la ley de equivalencia se puede observar, por caso, en las minicomputadoras de la compañía **DEC**
- En ocasiones, las instrucciones máquina más complicadas se implementaban en un software de muy bajo nivel llamado **microcódigo**
 - Parte de las instrucciones estaban implementadas directamente en hardware, pero parte también se implementaban a través de microprogramación
 - La idea era **balancear costo vs. desempeño**



Microprogramación

- La técnica de microprogramación floreció principalmente durante el período que la memoria principal era excesivamente lenta
 - El contar con un set de instrucciones complejo permite generar programas más compactos y con menor requerimiento de ancho de banda a memoria
 - Este tipo de arquitectura se denomina casualmente **Complex Instruction Set Computer (CISC)**
 - La familia de procesadores Intel es el ejemplo más conocido de las arquitecturas **CISC**



Microprogramación

- La utilidad de la microprogramación empezó a ceder de la mano de dos avances:
 - Por un lado el **crecimiento exponencial de la cantidad de transistores disponibles** permitió reimplementar en hardware más y más funcionalidad
 - Pero el mayor impacto vino de la mano de la **mejora en el desempeño de la memoria principal**
 - En la actualidad se opta por **simplificar el set de instrucciones (RISC)**, mejorando el desempeño mediante el uso de pipeline y de múltiples núcleos
 - La arquitectura **ARM** pone en práctica esta filosofía



Ciclo básico de un CPU

- Las computadoras llevan adelante infinitas veces su **ciclo básico de operación**:
 - ➔ Etapa Fetch: se busca la próxima instrucción a ser ejecutada, la cual es apuntada por el registro **PC**
 - ➔ Etapa Decode: luego, se determina de qué tipo de instrucción se trata
 - ➔ Etapa Effective Address: se calcula la dirección efectiva referida por la instrucción (si es que alguna) y/o se busca en memoria los operandos para poder ejecutar la instrucción



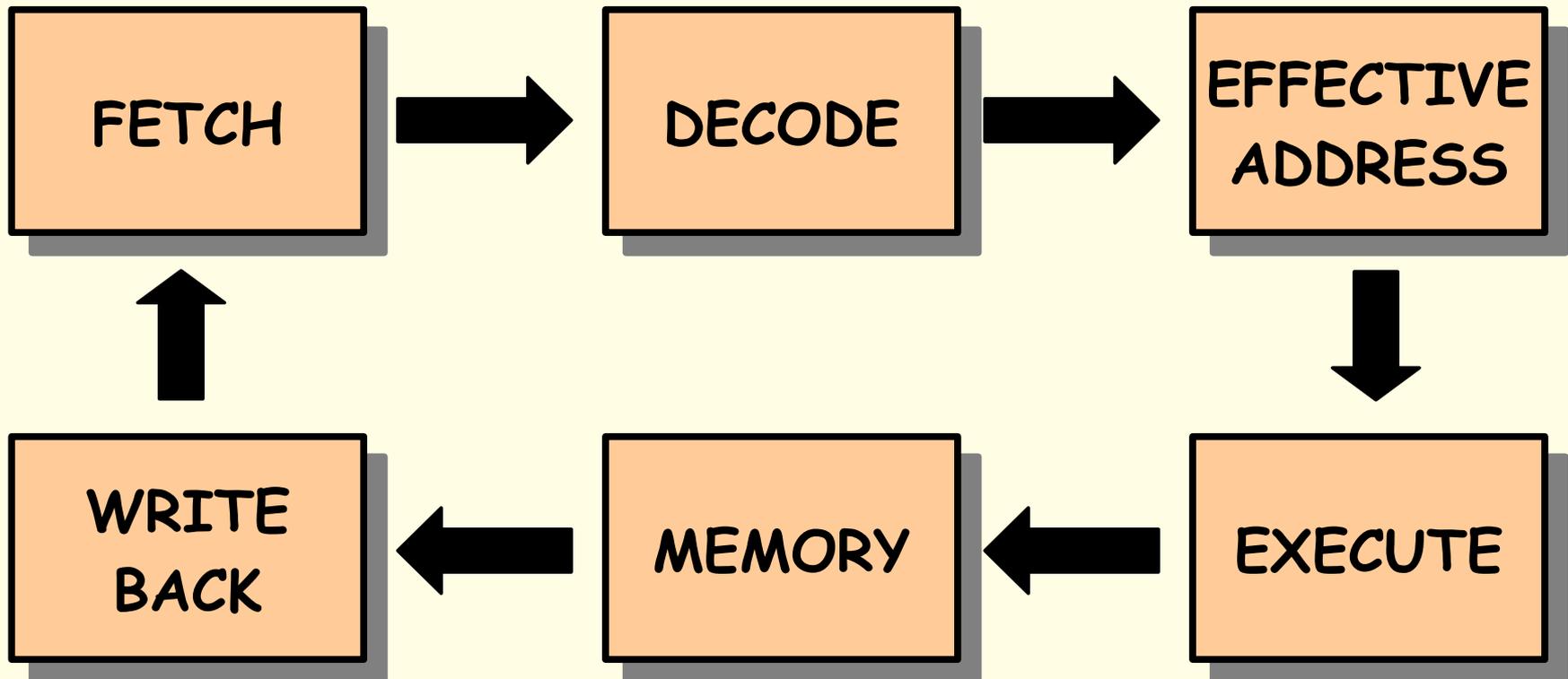
Ciclo básico de un CPU

● Continúa:

- Etapa Execute: sabiendo de qué instrucción se trata y contando con los operandos que se necesiten se puede enviar todo a la **ALU** para que sea ejecutado
- Etapa Memory: las arquitecturas que no permiten acceder a operandos en memoria cuentan con una etapa específica donde se accede a memoria para leer o escribir una determinada locación
- Etapa Write-Back: usualmente el resultado de las operaciones se almacena en alguno de los registros de la computadora durante esta etapa



Ciclo básico de un CPU



Organización de la memoria

● La memoria se organiza como un arreglo de **n** locaciones, cada una de **k** bits

→ Cada locación cuenta con un identificador que la caracteriza denominado **dirección**

→ De igual forma, cada locación almacena un **contenido** de **k** bits

0000:	01110010
0001:	11100101
0010:	00011001
0011:	01110010
0100:	00100101
⋮	⋮
1110:	00111001
1111:	10010110

● Las operaciones básicas con la memoria son:

→ **Leer** un valor de la memoria

→ **Escribir** un valor en la memoria



¿Preguntas?

