

Estructura de Computadores

Tema 1. Introducción

Rendimiento

- ▶ ¿Cómo sabemos si un computador A ofrece mejor rendimiento que un computador B?

Rendimiento

- ▶ Tiempo de ejecución
 - ▶ Tiempo transcurrido entre el inicio y el final de una tarea
 - ▶ Rendimiento es la inversa del tiempo de ejecución

Rendimiento

- ▶ Tiempo de ejecución
 - ▶ Tiempo transcurrido entre el inicio y el final de una tarea
 - ▶ Rendimiento es la inversa del tiempo de ejecución
- ▶ Productividad (Throughput)
 - ▶ Número de tareas completadas por unidad de tiempo
 - ▶ Rendimiento en servidores web, bases de datos...

Rendimiento

- Un computador A tiene 1 CPU que tarda 10 us en completar una tarea. Un computador B tiene 200 CPUs que tardan 20 us en completar la misma tarea. ¿Cuál tiene mayor rendimiento?

Relación entre tiempo de ejecución y productividad

- ▶ A menor tiempo de ejecución, mayor productividad
- ▶ Aumentar la productividad puede reducir el tiempo de ejecución, pero solo en caso de congestión, ya que se reduce el tiempo de espera en las colas

Tiempo de ejecución

- ▶ **Tiempo de CPU** + Tiempo de espera E/S + Tiempo de espera mientras se ejecutan otras tareas
- ▶ En EC, solo tendremos en cuenta el **tiempo de CPU**

Tiempo de ejecución

- ▶ **Tiempo de CPU** + Tiempo de espera E/S + Tiempo de espera mientras se ejecutan otras tareas
- ▶ En EC, solo tendremos en cuenta el **tiempo de CPU**
- ▶ $Rendimiento = \frac{1}{T_{ejecución}}$

Speedup

- ▶ Mejora de rendimiento o speedup
 - ▶ Cuántas veces más rápido se ejecuta una tarea al introducir una mejora en el programa o en la arquitectura

Speedup

- ▶ Mejora de rendimiento o speedup
 - ▶ Cuántas veces más rápido se ejecuta una tarea al introducir una mejora en el programa o en la arquitectura
- ▶ $Speedup = \frac{Rendimiento_{mejorado}}{Rendimiento_{original}} = \frac{T_{original}}{T_{mejorado}}$

Speedup

- ▶ Mejora de rendimiento o speedup
 - ▶ Cuántas veces más rápido se ejecuta una tarea al introducir una mejora en el programa o en la arquitectura
- ▶ $Speedup = \frac{Rendimiento_{mejorado}}{Rendimiento_{original}} = \frac{T_{original}}{T_{mejorado}}$
- ▶ Ejemplo
 - ▶ $T_{original} = 10s$
 - ▶ $T_{mejorado} = 5s$
 - ▶ $Speedup = \frac{10}{5} = 2$

Factores que influyen en el tiempo de ejecución

- ▶ $t_{eje} = n_{ciclos} \times t_c = n_{ciclos} / f_{clock}$
 - ▶ n_{ciclos} : Número de ciclos de reloj que tarda la ejecución
 - ▶ t_c : Tiempo de ciclo
 - ▶ f_{clock} : Frecuencia de reloj

Factores que influyen en el tiempo de ejecución

- ▶ $t_{eje} = n_{ciclos} \times t_c = n_{ciclos} / f_{clock}$
 - ▶ n_{ciclos} : Número de ciclos de reloj que tarda la ejecución
 - ▶ t_c : Tiempo de ciclo
 - ▶ f_{clock} : Frecuencia de reloj
- ▶ Dos formas de reducir el tiempo de ejecución
 - ▶ Reducir el número de ciclos
 - ▶ Aumentar la frecuencia de reloj (reducir tiempo de ciclo)

Reducir n_{ciclos}

- ▶ Ejecutar menor número de instrucciones (n_{ins})
- ▶ Reducir el número de ciclos por instrucción (CPI)
- ▶ $n_{ciclos} = n_{ins} \times CPI$
 - ▶ CPI es el promedio de ciclos por instrucción de todo el programa

Reducir n_{ciclos}

- ▶ Ejecutar menor número de instrucciones (n_{ins})
- ▶ Reducir el número de ciclos por instrucción (CPI)
- ▶ $n_{ciclos} = n_{ins} \times CPI$
 - ▶ CPI es el promedio de ciclos por instrucción de todo el programa
- ▶ $t_{eje} = n_{ciclos} \times t_c$

Reducir n_{ciclos}

- ▶ Ejecutar menor número de instrucciones (n_{ins})
- ▶ Reducir el número de ciclos por instrucción (CPI)
- ▶ $n_{ciclos} = n_{ins} \times CPI$
 - ▶ CPI es el promedio de ciclos por instrucción de todo el programa
- ▶ $t_{eje} = n_{ciclos} \times t_c$
- ▶ $t_{eje} = n_{ins} \times CPI \times t_c$

Reducir n_{ciclos}

► $n_{ciclos} = n_{ins} \times CPI$

Reducir n_{ciclos}

- ▶ $n_{ciclos} = n_{ins} \times CPI$
- ▶ Para reducir n_{ins} hay que mejorar el compilador

Reducir n_{ciclos}

- ▶ $n_{ciclos} = n_{ins} \times CPI$
- ▶ Para reducir n_{ins} hay que mejorar el compilador
- ▶ El CPI depende del retardo de cada tipo de instrucción y del número de instrucciones de cada tipo
 - ▶ $CPI = (n_1 \times CPI_1 + n_2 \times CPI_2 + \dots + n_m \times CPI_m) / n_{ins}$
 - ▶ Para reducir el CPI:
 - ▶ Mejorar la microarquitectura
 - ▶ Sustituir instrucciones costosas por instrucciones simples (mul por sll)

Reducir n_{ciclos}

- ▶ $n_{ciclos} = n_{ins} \times CPI$
- ▶ Para reducir n_{ins} hay que mejorar el compilador
- ▶ El CPI depende del retardo de cada tipo de instrucción y del número de instrucciones de cada tipo
 - ▶ $CPI = (n_1 \times CPI_1 + n_2 \times CPI_2 + \dots + n_m \times CPI_m) / n_{ins}$
 - ▶ Para reducir el CPI:
 - ▶ Mejorar la microarquitectura
 - ▶ Sustituir instrucciones costosas por instrucciones simples (mul por sll)
- ▶ $t_{eje} = n_{ins} \times CPI \times t_c$

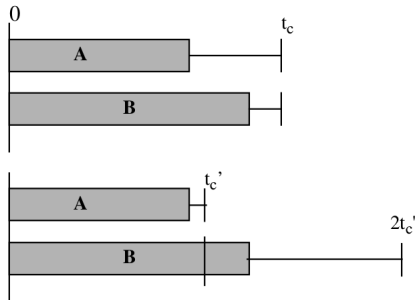
Reducir n_{ciclos}

- ▶ $n_{ciclos} = n_{ins} \times CPI$
- ▶ Para reducir n_{ins} hay que mejorar el compilador
- ▶ El CPI depende del retardo de cada tipo de instrucción y del número de instrucciones de cada tipo
 - ▶ $CPI = (n_1 \times CPI_1 + n_2 \times CPI_2 + \dots + n_m \times CPI_m) / n_{ins}$
 - ▶ Para reducir el CPI:
 - ▶ Mejorar la microarquitectura
 - ▶ Sustituir instrucciones costosas por instrucciones simples (mul por sll)
- ▶ $t_{eje} = n_{ins} \times CPI \times t_c$
- ▶ $t_{eje} = (n_1 \times CPI_1 + n_2 \times CPI_2 + \dots + n_m \times CPI_m) \times t_c$

Ejemplo

- ▶ Queremos comparar dos versiones de un programa en un mismo computador, que dispone de 3 tipos de instrucciones A, B y C con CPI 1, 2 y 3 respectivamente. La versión 1 consta de: 2 instrucciones de A, 1 de B y 2 de C. La versión 2 consta de: 4 instrucciones de A, 1 de B y 1 de C. ¿Cuál es más rápido? ¿Cuáles son los CPI promedio de ambas versiones?

Aumentar la frecuencia de reloj (reducir el tiempo de ciclo)



- ▶ Se reduce la latencia para las instrucciones de tipo A
- ▶ Las instrucciones de tipo B requieren 2 ciclos en lugar de 1
- ▶ Beneficio depende del número de instrucciones de tipo A y B
- ▶ No siempre se mejora el rendimiento

Ejemplo

- ▶ El procesador A tiene $tc_A = 500ps$ y $CPI_A = 2$. Supongamos que lo rediseñamos para que se use un menor tiempo de ciclo. El nuevo procesador B tiene $tc_B = 250ps$. Pero esto requiere aumentar el número de ciclos del programa de test, aumentando el CPI promedio: $CPI_B = 3$. ¿Es más rápido el nuevo diseño?

Ejemplo

- Supongamos que un computador A, con $f_A = 2\text{Ghz}$, ejecuta un programa en $t_{eje} = 10\text{s}$. Con ciertas mejoras en el circuito, se ejecuta en 6s, pero aumenta el número de ciclos en un factor 1.2. ¿Cuál es la frecuencia de reloj del nuevo diseño?

Ley de Amdahl

- ▶ “El máximo speedup que se puede conseguir minimizando el retardo de una parte está limitado por la fracción de tiempo que representa dicha parte sobre el tiempo total”
- ▶ Speedup total al mejorar una parte P en un factor S :

$$S_t = \frac{1}{(1 - P) + \frac{P}{S}}$$

Ley de Amdahl

- ▶ “El máximo speedup que se puede conseguir minimizando el retardo de una parte está limitado por la fracción de tiempo que representa dicha parte sobre el tiempo total”
- ▶ Speedup total al mejorar una parte P en un factor S :

$$S_t = \frac{1}{(1 - P) + \frac{P}{S}}$$

- ▶ Máximo speedup posible al mejorar una parte P :

$$S_{max} = \frac{1}{1 - P}$$

Ley de Amdahl

- ▶ “El máximo speedup que se puede conseguir minimizando el retardo de una parte está limitado por la fracción de tiempo que representa dicha parte sobre el tiempo total”
- ▶ Speedup total al mejorar una parte P en un factor S :

$$S_t = \frac{1}{(1 - P) + \frac{P}{S}}$$

- ▶ Máximo speedup posible al mejorar una parte P :

$$S_{max} = \frac{1}{1 - P}$$

- ▶ ¿Máximo speedup al mejorar una parte de un programa que representa el 80 % del tiempo de ejecución?

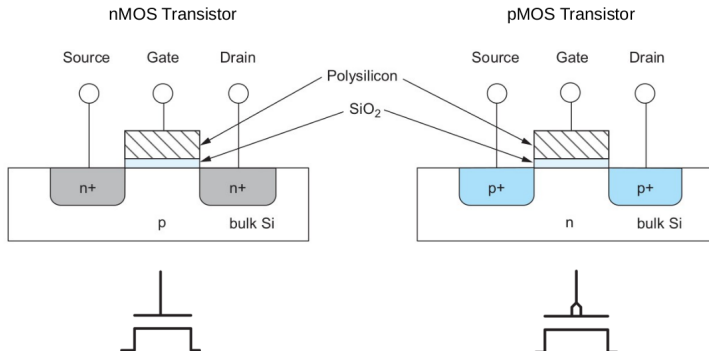
“Ley” de Moore

“La densidad de transistores en un chip se duplica cada 2 años”

- ▶ Gordon Moore, cofundador de Intel, 1965

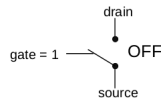
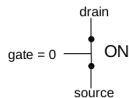
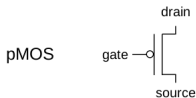
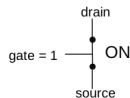
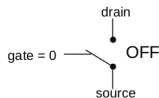
Transistores nMOS y pMOS

► MOS: Metal–Oxide–Semiconductor



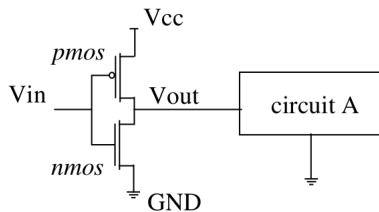
Transistores nMOS y pMOS

► MOS: Metal–Oxide–Semiconductor



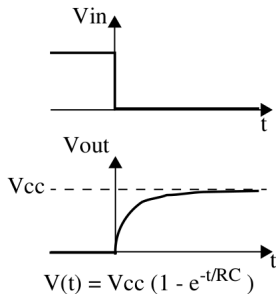
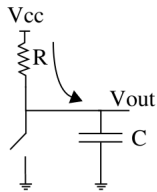
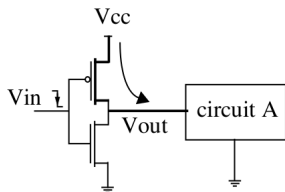
Tecnología CMOS

► Complementary Metal–Oxide–Semiconductor



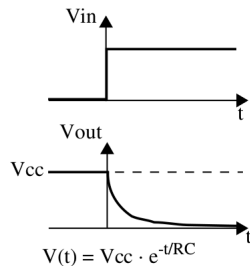
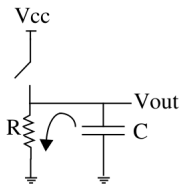
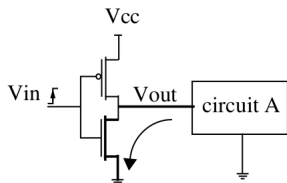
Tecnología CMOS - Carga

► Complementary Metal–Oxide–Semiconductor



Tecnología CMOS - Descarga

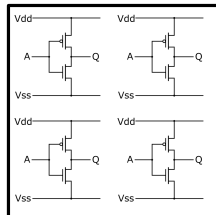
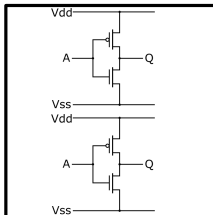
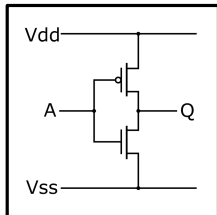
► Complementary Metal–Oxide–Semiconductor



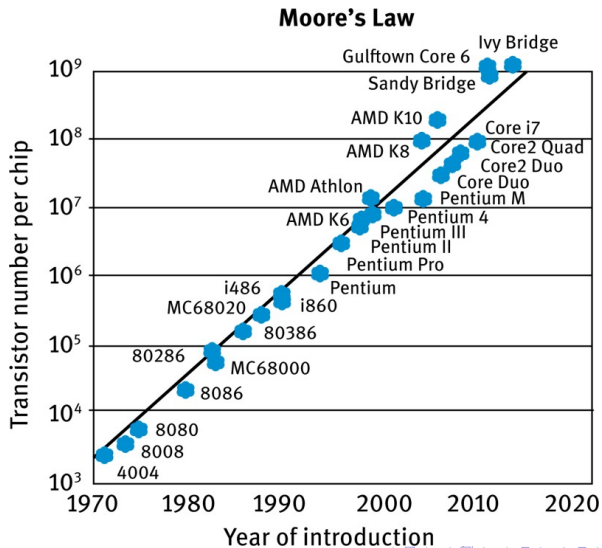
“Ley” de Moore

“La densidad de transistores en un chip se duplica cada 2 años”

► Gordon Moore, cofundador de Intel, 1965



“Ley” de Moore



Consecuencias de la “Ley” de Moore

- ▶ Transistores más pequeños
- ▶ Reducción del tiempo de conmutación de los transistores
- ▶ Mayor frecuencia de reloj
- ▶ Mayor disipación de potencia y consumo energético

Potencia dinámica

- ▶ Consumo de energía producido durante los ciclos de carga/descarga de los transistores (conmutación)
 - ▶ P: Potencia dinámica (Watts)
 - ▶ α : Fracción de transistores que conmutan por ciclo
 - ▶ C: Capacitancia (Farads)
 - ▶ V: Voltaje (Volts)
 - ▶ f_{clk} : Frecuencia de reloj

$$P_{dinámica} = \alpha \cdot C \cdot V^2 \cdot f_{clk}$$

Potencia dinámica vs Potencia estática

► Potencia dinámica

$$P_{\text{total}} = \alpha \cdot C \cdot V^2 \cdot f_{\text{clk}} + I_{\text{leak}} \cdot V$$

► Potencia estática

- I_{leak} : corriente parásita que circula por el transistor en circuito abierto

$$P_{\text{estática}} = I_{\text{leak}} \cdot V$$

Potencia dinámica vs Potencia estática

- ▶ Potencia dinámica

$$P_{\text{total}} = \alpha \cdot C \cdot V^2 \cdot f_{\text{clk}} + I_{\text{leak}} \cdot V$$

- ▶ Potencia estática

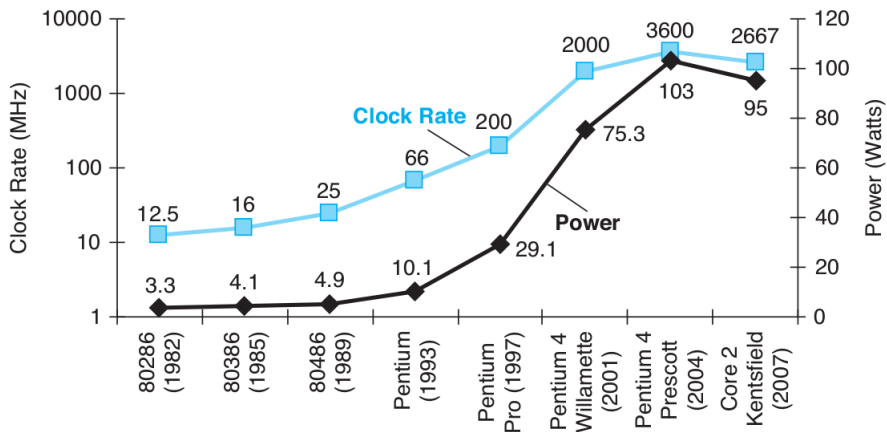
- ▶ I_{leak} : corriente parásita que circula por el transistor en circuito abierto

$$P_{\text{estática}} = I_{\text{leak}} \cdot V$$

- ▶ Potencia total: $P = P_{\text{dinámica}} + P_{\text{estática}}$

- ▶ Energía consumida en un tiempo t , en Joules:

$$E = t \cdot P$$



Potencia, Energía y Temperatura

- ▶ La energía consumida se convierte en calor
 - ▶ Sistema de refrigeración para evitar exceso de temperatura
 - ▶ Thermal Design Power (TDP)

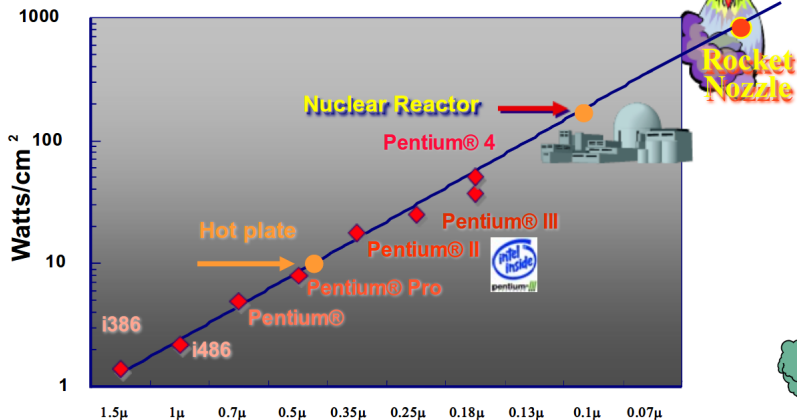
Potencia, Energía y Temperatura

- ▶ La energía consumida se convierte en calor
 - ▶ Sistema de refrigeración para evitar exceso de temperatura
 - ▶ Thermal Design Power (TDP)
- ▶ Dispositivos móviles
 - ▶ Alimentados por baterías
 - ▶ Espacio reducido para el sistema de refrigeración

Potencia, Energía y Temperatura

- ▶ La energía consumida se convierte en calor
 - ▶ Sistema de refrigeración para evitar exceso de temperatura
 - ▶ Thermal Design Power (TDP)
- ▶ Dispositivos móviles
 - ▶ Alimentados por baterías
 - ▶ Espacio reducido para el sistema de refrigeración
- ▶ Hemos alcanzado el punto en el que el calor no se puede disipar
 - ▶ La frecuencia máxima se ha estancado en los últimos años

Power Density








* "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999.

Técnicas de reducción de consumo







- ▶ Clock gating
- ▶ Power gating
- ▶ Dynamic Voltage and Frequency Scaling (DVFS)

Essentials

 [Export specifications](#)

Product Collection	9th Generation Intel® Core™ i9 Processors
Code Name	Products formerly Coffee Lake
Vertical Segment	Desktop
Processor Number	I9-9900K
Status	Launched
Launch Date 	Q4'18
Lithography 	14 nm
Included Items	Please note: The boxed product does not include a fan or heat sink
Use Conditions 	PC/Client/Tablet
Recommended Customer Price 	\$488.00 - \$499.00

Performance

# of Cores 	8
# of Threads 	16
Processor Base Frequency 	3.60 GHz
Max Turbo Frequency 	5.00 GHz
Cache 	16 MB SmartCache
Bus Speed 	8 GT/s DMI3
TDP 	95 W