



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Departament d'Arquitectura de Computadors

Estructura de Computadores

Tema 1: Rendimiento, Ley de Amdahl y Consumo

Agustín Fernández

Departament d'Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya



Rendimiento

- ❑ ¿Cómo sabemos que una CPU es mejor que otra?
- ❑ ¿Cómo medimos el RENDIMIENTO de un diseño hardware respecto a otro?

Depende de nuestro objetivo, tenemos 2 métricas diferentes

- ❑ **TIEMPO de EJECUCIÓN** (Execution Time)
 - T_{exe} . Tiempo transcurrido desde el inicio de una tarea hasta su finalización
 - Rendimiento = $1/T_{exe}$
- ❑ **PRODUCTIVIDAD** (Throughput)
 - Tareas completadas por unidad de tiempo

Rendimiento

- ❑ Ejemplo. Tenemos 2 fábricas de coches. La fábrica A tarda 1 día en producir un coche. La fábrica B tarda 3 días en producir 5 coches.
 - La fábrica A tiene menor tiempo de ejecución.
 - La fábrica B tiene más productividad (1,66 coches/día)

- ❑ Ejemplo. Tenemos 2 Computadores. El computador A tienen 1 CPU que tarda 10 μ s en realizar una operación. El computador B tiene 100 CPUs que tardan cada una 20 μ s en realizar una operación.
 - El computador A tiene menor tiempo de ejecución (productividad 10^5 ops/s).
 - El computador B tiene más productividad (productividad $5 \cdot 10^6$ ops/s)

Tiempo de Ejecución

- El tiempo de ejecución normalmente incluye:

$$T_{EXE} = T_{CPU} + T_{E/S} + T_{resto}$$

- T_{CPU} . **Tiempo de CPU (CPU time)**. El tiempo que consumimos ejecutando el programa.
 - $T_{E/S}$. Tiempo de E/S. El tiempo que pasamos esperando el disco o la red (E/S en general)
 - T_{resto} . Resto de tiempo que consume el SO, esperando a conseguir CPU, ...
-
- En este curso, cuando hablemos de Tiempo de Ejecución sólo tendremos en cuenta el Tiempo de CPU.

Rendimiento y Speedup

- El rendimiento se evalúa utilizando el Tiempo de Ejecución:

$$\text{Rendimiento} = 1/T_{\text{EXE}}$$

- Speedup (Aceleración, Ganancia de Rendimiento):

$$\text{Speedup} = \frac{\text{Tiempo}_{\text{ORIGINAL}}}{\text{Tiempo}_{\text{MEJORADO}}} = \frac{\text{Rendimiento}_{\text{MEJORADO}}}{\text{Rendimiento}_{\text{ORIGINAL}}}$$

Ejemplo:

- $T_{\text{original}} = 10\text{s}$
- $T_{\text{mejorado}} = 5\text{s}$
- $\text{Speedup} = 10/5 = 2$

- Las mejoras se pueden producir a muchos niveles:

- En el programa (algoritmos, estructuras de datos, lenguajes de programación, ...)
- En el compilador (optimización de bucles, de llamadas, ...)
- En la microarquitectura (ALUs, pipeline, cache, especulación, predictores, ...)
- En la tecnología (frecuencia de reloj, tamaño transistores, ...)

Factores que influyen en T_{EXE}

- Podemos expresar el Tiempo de ejecución de la siguiente forma:

$$T_{EXE} = n_{ciclos} \cdot T_c = n_{ciclos} / f_{clock}$$

siendo:

- n_{ciclos} = Número total de ciclos de reloj que tarda la ejecución del programa
- T_c = Tiempo de ciclo (o Periodo de reloj)
- f_{clock} = Frecuencia de reloj = $1 / T_c$

¿En el mundo real
cómo podemos
saber el número de
ciclos que tarda un
programa?

- ¿Cómo podemos mejorar el Tiempo de ejecución?

- Reduciendo el número de ciclos
- Aumentando la frecuencia de reloj (reduciendo el tiempo de ciclo)

Reducción del número de ciclos

- El número de ciclos se puede expresar como:

$$n_{\text{ciclos}} = N \cdot CPI$$



$$T_{\text{EXE}} = N \cdot CPI \cdot T_c$$

, siendo:

- N, el número total de instrucciones ejecutadas
- CPI, el promedio de ciclos por instrucción

- Cada tipo de instrucción i , tiene un CPI_i diferente

$$n_{\text{ciclos}} = \sum CPI_i \cdot n_i$$

, siendo:

- CPI_i , ciclos que tarda una instrucción de tipo i
- n_i , número total de instrucciones del tipo i ejecutadas

¿Cómo reducimos n_{ciclos} ?

- Reduciendo N
 - ✓ Mejorando el compilador
- Reduciendo el CPI
 - ✓ Mejorando la microarquitectura, usando instrucciones más rápidas (mult por sll)

$$T_{EXE} = N \cdot CPI \cdot T_c$$

$$T_{EXE} = N \cdot CPI \cdot T_c$$

$$T_{EXE} = \frac{\text{instrucciones}}{\text{programa}} \cdot \frac{\text{ciclos}}{\text{instrucción}} \cdot \frac{\text{segundos}}{\text{ciclo}} = \frac{\text{segundos}}{\text{programa}}$$

Reducción del número de ciclos. Ejemplo

- Comparad el rendimiento de 2 versiones P1 y P2 de un mismo programa con un computador que tiene 3 tipos de instrucciones (A, B y C).

tipo inst.	CPI	n _i	
		P1	P2
A	1	2·10 ⁹	4·10 ⁹
B	2	1·10 ⁹	1·10 ⁹
C	3	2·10 ⁹	1·10 ⁹

- $n_{\text{ciclosP1}} = CPI_A \cdot n_A + CPI_B \cdot n_B + CPI_C \cdot n_C = (1 \cdot 2 + 2 \cdot 1 + 3 \cdot 2) \cdot 10^9 = 10 \cdot 10^9 \text{ ciclos}$
- $n_{\text{ciclosP2}} = CPI_A \cdot n_A + CPI_B \cdot n_B + CPI_C \cdot n_C = (1 \cdot 4 + 2 \cdot 1 + 3 \cdot 1) \cdot 10^9 = 9 \cdot 10^9 \text{ ciclos}$
- $CPI_{P1} = n_{\text{ciclosP1}} / N_{P1} = 10 \cdot 10^9 / 5 \cdot 10^9 = 2 \text{ ciclos/inst}$
- $CPI_{P2} = n_{\text{ciclosP2}} / N_{P2} = 9 \cdot 10^9 / 6 \cdot 10^9 = 1,5 \text{ ciclos/inst}$

Reducción del número de ciclos. Ejemplo

- Comparad el rendimiento de 2 versiones P1 y P2 de un mismo programa con un computador que tiene 3 tipos de instrucciones (A, B y C).

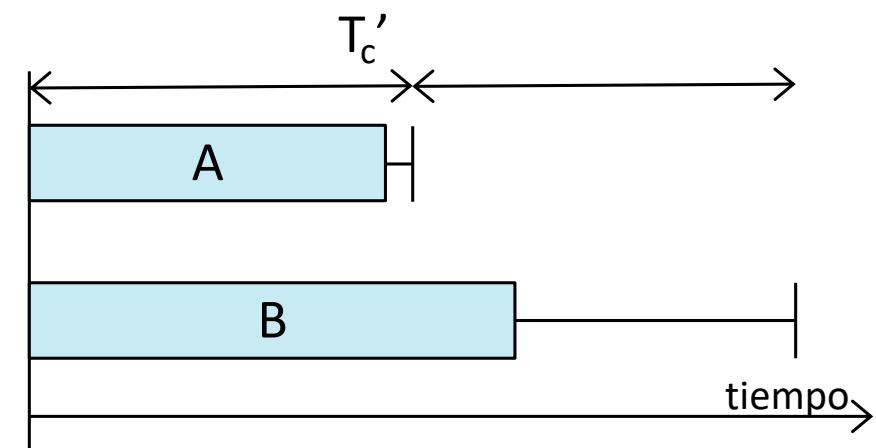
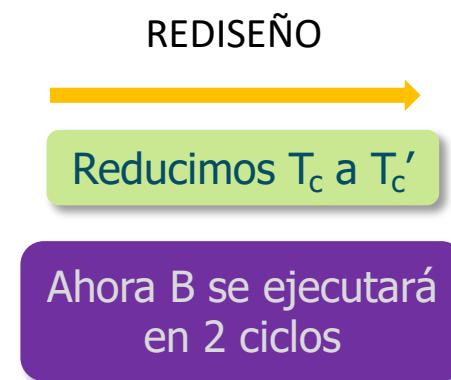
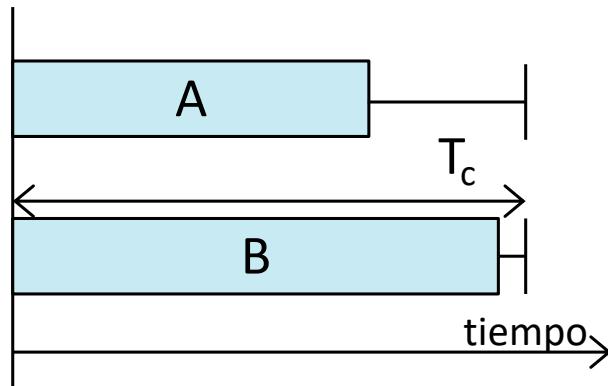
tipo inst.	CPI	n_i	
		P1	P2
A	1	$2 \cdot 10^9$	$4 \cdot 10^9$
B	2	$1 \cdot 10^9$	$1 \cdot 10^9$
C	3	$2 \cdot 10^9$	$1 \cdot 10^9$

	P1	P2
n_{ciclos}	$10 \cdot 10^9$	$9 \cdot 10^9$
N_{inst}	$5 \cdot 10^9$	$6 \cdot 10^9$
CPI	2 c/i	1,5 c/i

- Suponiendo que $f_{clock} = 2\text{GHz}$ (ciclos/s), calculad T_{EXE} y el Speedup de P2 respecto de P1
 - $T_{EXEP1} = n_{ciclosP1} / f_{clock} = 10 \cdot 10^9 / 2 \cdot 10^9 = 5\text{s}$
 - $T_{EXEP2} = n_{ciclosP2} / f_{clock} = 9 \cdot 10^9 / 2 \cdot 10^9 = 4,5\text{s}$
 - Speedup = $T_{EXEP1} / T_{EXEP2} = 5 / 4,5 = 1,11$

Aumentar el f_{clock} → Reducir T_c

- ¿De que depende el T_c ?
 - Todos los computadores están gobernados por un reloj ($f_{clock} = 1/T_c$)
 - Podemos dividir la ejecución de 1 instrucción en fases
 - En cada una de esas fases tardamos un cierto tiempo, el T_c podría ser el tiempo que tarda la fase más lenta de todas.
- ¿Cómo podemos reducir el T_c ?



Será mejor o peor dependiendo del % de ops de cada tipo que tengamos

Aumentar el f_{clock} → Reducir T_c . Ejemplo

- El procesador A tiene $T_{cA} = 0,5\text{ns} = 0,5 \cdot 10^{-9}\text{s}$ ($f_A=2\text{GHz}$) y $CPI_A=2$. Rediseñamos el procesador para reducir el Tiempo de Ciclo. El nuevo procesador B tiene $T_{cB} = 0,25\text{ns} = 0,25 \cdot 10^{-9}\text{s}$ ($f_B=4\text{GHz}$), pero como efecto lateral el CPI aumenta a $CPI_B=3$. ¿Cuál es más rápido?
 - Podemos suponer que ejecutamos un programa con N instrucciones
 - $T_{EXEA} = N \cdot CPI_A \cdot T_{cA} = N \cdot 2 \cdot 0,5 \cdot 10^{-9} = N \cdot 10^{-9} \text{ s}$
 - $T_{EXEB} = N \cdot CPI_B \cdot T_{cB} = N \cdot 3 \cdot 0,25 \cdot 10^{-9} = N \cdot 0,75 \cdot 10^{-9} \text{ s}$
 - **El computador B es más rápido que el A**
 - **Speedup = $T_{EXEA} / T_{EXEB} = 1/0,75 = 1,33$**

Ley de Amdahl

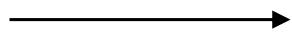
□ Supongamos que tenemos

- una tarea que se ejecuta en un Tiempo T_0 ,
- que una fracción (P) de esa tarea puede ser mejorada, en consecuencia el resto ($1-P$) quedará igual,
- y que el speedup de la fracción mejorada es S .

□ ¿Cuál es el speedup de la tarea completa?

Expresión formal del sentido común.

$$T_0 = (1-P) \cdot T_0 + P \cdot T_0$$



$$T_M = (1-P) \cdot T_0 + P \cdot T_0 / S$$

$$S_G = \frac{T_0}{T_M} = \frac{T_0}{(1 - P) \cdot T_0 + \frac{P \cdot T_0}{S}} = \frac{1}{(1 - P) + \frac{P}{S}}$$

Ley de Amdahl

- ❑ ¿Cuál es el MÁXIMO SPEEDUP TEÓRICO que podemos conseguir?

$$S_G = \frac{T_0}{T_M} = \frac{1}{(1 - P) + \frac{P}{S}}$$

$$S_{\max} = \lim_{S \rightarrow \infty} \frac{1}{(1 - P) + \frac{P}{S}} = \frac{1}{1 - P}$$

Ley de Amdahl. El máximo Speedup (S_{\max}) que podemos conseguir mejorando una parte de una tarea está limitado por el tiempo de ejecución que representa esta parte en el total (P).

P	20%	50%	75%	90%	99%
S_{\max}	1,25	2	4	10	100

Ley de Amdahl

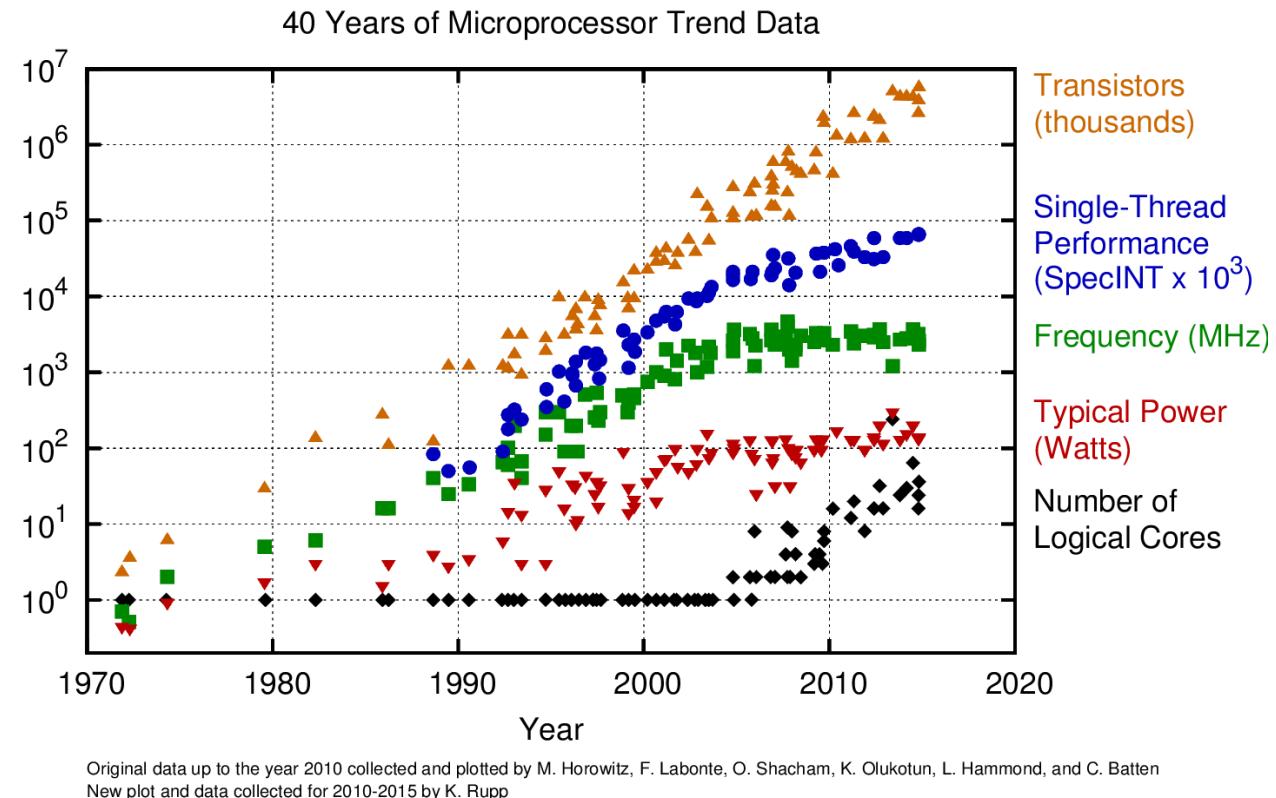
Regla de Diseño: Optimizar el caso más frecuente

- ❑ Pero, no es buena idea dedicar todos los esfuerzos al caso más frecuente y no hacer nada con el resto.
- ❑ Es mejor, aumentar la fracción de tarea que mejoramos, que mejorar excesivamente una sola parte.

- ❑ Supongamos una tarea con una parte secuencial ($1-P$) y otra parte totalmente paralelizable (P).
 - Con $P=20\%$ paralelizable, si lo ejecutamos en 10.000 CPUs y el 80% secuencial en 1 CPU, el speedup que obtendremos será 1,25.
 - Con $P=99\%$ paralelizable, si lo ejecutamos en 2 CPUs y el 1% secuencial en 1 CPU, el speedup que obtendremos será 1,98.

Ley de Moore

- En 1965 Gordon E. Moore (cofundador de Intel) afirmó que el número de transistores por unidad de superficie se duplicaría cada año:
 - Gordon E. Moore. "Cramming more components onto integrated circuits" Electronics Magazine, Apr 1965.
- En 1975, modificó su afirmación diciendo que se duplicaría cada 24 meses.



Más que una ley es un objetivo

Implicaciones de la Ley de Moore

Tamaño Transistor ↓



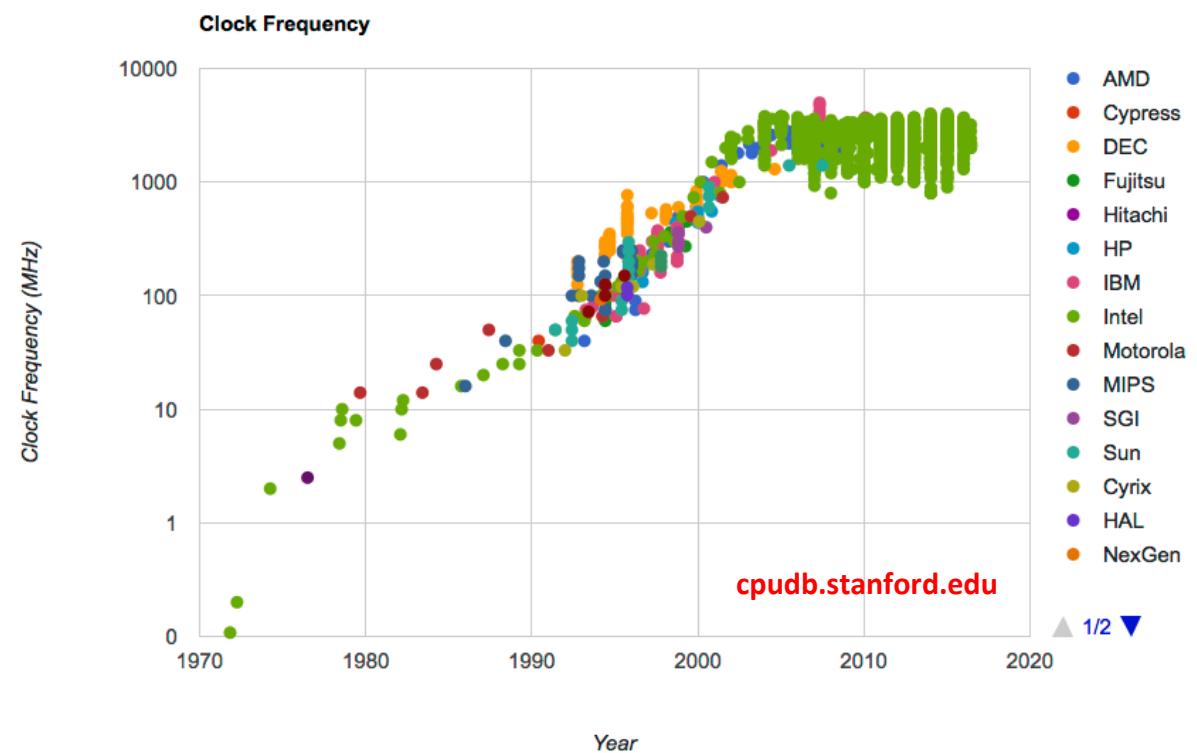
Tiempo Conmutación ↓



Frecuencia ↑



Consumo ↑



The Power Wall

Disipación de Potencia

- En una CPU tenemos dos tipos de Potencia Disipada
 - **Potencia Dinámica (P_d)**. Causada por la conmutación de los transistores.
 - **Potencia Estática (P_s)**. También conocida como “Leakage Current” (corrientes de fuga).

$$P = P_d + P_s$$

- Potencia y Energía
 - La Energía se mide en unidades de trabajo: **Julios**
 - Potencia es la energía consumida por unidad de tiempo: **Watios (Julios/seg)**
 - Si la potencia es constante:

$$\text{Energía} = \text{Potencia} \times \text{tiempo}$$

Potencia Dinámica

- Consumo de energía producido por los ciclos de carga/descarga de los transistores (comutación)

$$P_d = \alpha \cdot C \cdot V^2 \cdot f_{clock}$$

siendo,

- P_d : Potencia Dinámica (Watios, W)
- α : Fracción de transistores que comutan por ciclo
- C : Capacitancia (Faradios, F)
- V : Voltaje (Voltios, V)
- f_{clock} : Frecuencia de reloj (ciclos/s)

Potencia Dinámica vs Potencia Estática

□ Potencia Dinámica (P_d):

$$P_d = \alpha \cdot C \cdot V^2 \cdot f_{clock}$$

- Normalmente ($\alpha \cdot C$) aparecen juntos en un solo factor C.

□ Potencia Total (W):

$$P = P_d + P_e$$

□ Potencia Estática (P_e):

$$P_e = I_{leak} \cdot V$$

siendo,

- I_{leak} : Corriente parásita que circula por los transistores en circuito abierto (A, Amperios)

□ Energía consumida en un tiempo t, (J, julios):

$$E = P \cdot t$$

Implicaciones

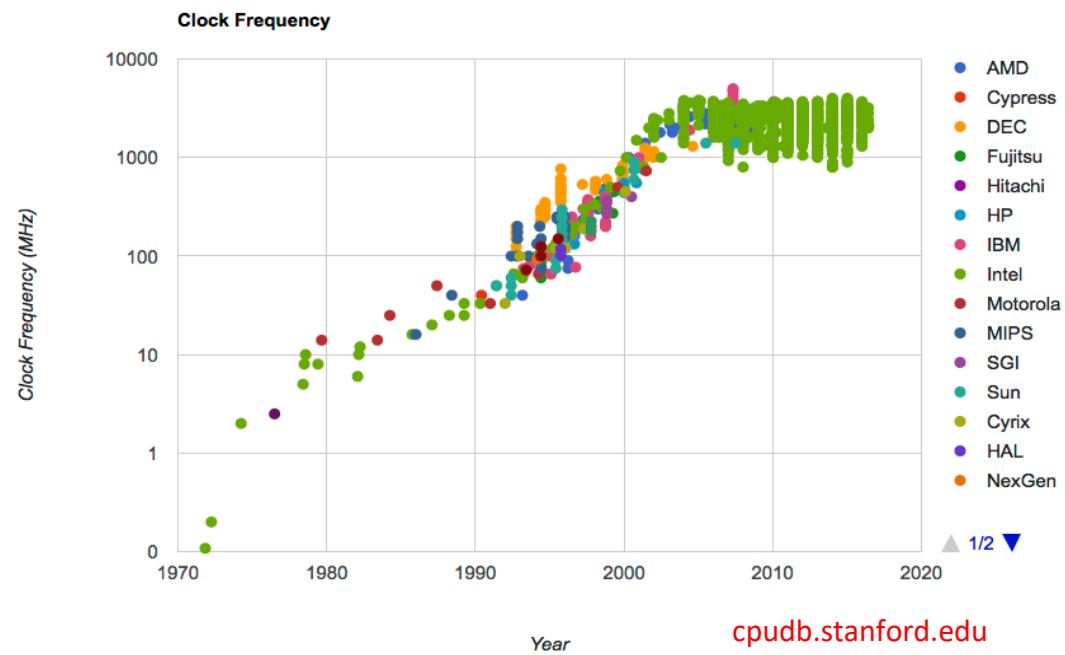
- ❑ Transistores más pequeños implica:
 - $C \downarrow$, pero como hay muchos más transistores la capacidad agregada C no varía mucho
 - El tiempo de computación $\downarrow \Rightarrow$ Frecuencia $\uparrow \Rightarrow$ Potencia \uparrow 😞
 - Para compensar el aumento de consumo: Voltaje \downarrow (de 5V a 1V)
 - $V \downarrow \Rightarrow V_t$ (tensión umbral del transistor) \downarrow (de 0,7V a 0,4V) \Rightarrow Problema de Consumo Estático.
 - ✓ Con un $V_t=0,2V$, el consumo estático podría llegar al 50% del total.
- ❑ La energía consumida se convierte en calor
 - Sistema de refrigeración para evitar el exceso de temperatura
 - Thermal Design Power (TDP)

Implicaciones

- ❑ La energía consumida se convierte en calor
 - Sistema de refrigeración para evitar el exceso de temperatura
 - Thermal Design Power (TDP)
 - ✓ Mide la cantidad de calor, medida en W, que produce una CPU

La frecuencia máxima de funcionamiento, se ha estancado en los últimos años.

Buscad en google: problemas snapdragon 810



cpudb.stanford.edu

Algunas Técnicas para Reducir el Consumo

❑ Clock Gating.

- Inhabilitan partes del chip para reducir el α de la potencia dinámica

❑ Dynamic Voltage and Frequency Scaling (DVFS).

- Bajan selectivamente el voltaje y la frecuencia de una parte del circuito que puede ir más lenta.

❑ Power Gating.

- Permite reducir la potencia dinámica y estática, apagando completamente (cortando la corriente) de partes del chip cuando no hay que usarlos.

Problema Rendimiento y Consumo

Tenim un codi en MIPS que treballa sobre matrius. Hem fet dues versions del codi, una usant accés aleatori (AA) i una altra usant accés seqüencial (AS). El total d'instruccions utilitzades a cada versió les podeu trobar a la següent taula:

	Load / Store	Mult	Salts que salten	Salts que no salten	Altres instruccions
AA	500.000	250.000	250.000	250.000	3.750.000
AS	500.000	0	500.000	500.000	6.000.000

Suposem que a la nostra arquitectura, cada instrucció de memòria costa 5 cicles, cada multiplicació 16 cicles, els salts 1 o 2 cicles depenen de si salta (2) o no salta (1) i la resta d'instruccions 1 cicle.

- Quin seria el CPI de cada versió del programa? Quin seria el speed-up de la versió d'accés seqüencial respecte a la d'accés aleatori?
- Quant de temps, en segons, triga en executar-se cada programa si el rellotge va a una freqüència de 2GHz?
- Si el processador dissipa 3 watts (Joules per segon) sigui quina sigui la instrucció executada, quin ha estat el consum total del processador en Joules de cada versió del programa?
- Ens diuen que es pot dissenyar una arquitectura alternativa on podríem reduir el temps de cicle de les operacions de multiplicació. Quin és el màxim nombre de cicles de multiplicació admissible per a la nova arquitectura per tal que el codi AA sigui al menys igual de ràpid que el codi AS?

Problema Rendimiento y Consumo

Tenim un codi en MIPS que treballa sobre matrius. Hem fet dues versions del codi, una usant accés aleatori (AA) i una altra usant accés seqüencial (AS). El total d'instruccions utilitzades a cada versió les podeu trobar a la següent taula:

	Load / Store	Mult	Salts que salten	Salts que no salten	Altres instruccions
AA	500.000	250.000	250.000	250.000	3.750.000
AS	500.000	0	500.000	500.000	6.000.000

Suposem que a la nostra arquitectura, cada instrucció de memòria costa 5 cicles, cada multiplicació 16 cicles, els salts 1 o 2 cicles depenen de si salta (2) o no salta (1) i la resta d'instruccions 1 cicle.

- a) Quin seria el CPI de cada versió del programa? Quin seria el speed-up de la versió d'accés seqüencial respecte a la d'accés aleatori?

$$N(AA) = 500.000 + 250.000 + 250.000 + 250.000 + 3.750.000 = 5 \cdot 10^6 \text{ instrucciones}$$

$$\text{Ciclos (AA)} = 500.000 \cdot 5 + 250.000 \cdot 16 + 250.000 \cdot 2 + 250.000 \cdot 1 + 3.750.000 \cdot 1 = 11 \cdot 10^6 \text{ ciclos}$$

$$N(AS) = 500.000 + 0 + 500.000 + 500.000 + 6.000.000 = 7,5 \cdot 10^6 \text{ instrucciones}$$

$$\text{Ciclos (AS)} = 500.000 \cdot 5 + 0 + 500.000 \cdot 2 + 500.000 \cdot 1 + 6.000.000 \cdot 1 = 10 \cdot 10^6 \text{ ciclos}$$

$$\text{CPI(AA)} = 11 \cdot 10^6 / 5 \cdot 10^6 = 2,2 \text{ c/i}$$

$$\text{CPI(AS)} = 10 \cdot 10^6 / 7,5 \cdot 10^6 = 1,33 \text{ c/i}$$

Problema Rendimiento y Consumo

Tenim un codi en MIPS que treballa sobre matrius. Hem fet dues versions del codi, una usant accés aleatori (AA) i una altra usant accés seqüencial (AS). El total d'instruccions utilitzades a cada versió les podeu trobar a la següent taula:

	Load / Store	Mult	Salts que salten	Salts que no salten	Altres instruccions
AA	500.000	250.000	250.000	250.000	3.750.000
AS	500.000	0	500.000	500.000	6.000.000

Suposem que a la nostra arquitectura, cada instrucció de memòria costa 5 cicles, cada multiplicació 16 cicles, els salts 1 o 2 cicles depenen de si salta (2) o no salta (1) i la resta d'instruccions 1 cicle.

- b) Quant de temps, en segons, triga en executar-se cada programa si el rellotge va a una freqüència de 2GHz?

$$T_{exe} = N \cdot CPI \cdot T_c = N \cdot CPI / Freq$$

$$N(AA) = 5 \cdot 10^6 \text{ instrucciones}$$
$$N(AS) = 7,5 \cdot 10^6 \text{ instrucciones}$$

$$T(AA) = 5 \cdot 10^6 \cdot 2,2 / 2 \cdot 10^9 = 5,5 \cdot 10^{-3} \text{ s} = 5,5 \text{ ms}$$

$$CPI(AA) = 2,2 \text{ c/i}$$

$$T(AS) = 7,5 \cdot 10^6 \cdot 1,33 / 2 \cdot 10^9 = 10^{-3} \text{ s} = 5 \text{ ms}$$

$$CPI(AS) = 1,33 \text{ c/i}$$

Problema Rendimiento y Consumo

Tenim un codi en MIPS que treballa sobre matrius. Hem fet dues versions del codi, una usant accés aleatori (AA) i una altra usant accés seqüencial (AS). El total d'instruccions utilitzades a cada versió les podeu trobar a la següent taula:

	Load / Store	Mult	Salts que salten	Salts que no salten	Altres instruccions
AA	500.000	250.000	250.000	250.000	3.750.000
AS	500.000	0	500.000	500.000	6.000.000

Suposem que a la nostra arquitectura, cada instrucció de memòria costa 5 cicles, cada multiplicació 16 cicles, els salts 1 o 2 cicles depenen de si salta (2) o no salta (1) i la resta d'instruccions 1 cicle.

- c) Si el processador dissipa 3 watts (Joules per segon) sigui quina sigui la instrucció executada, quin ha estat el consum total del processador en Joules de cada versió del programa?

$$\text{Consumo} = \text{Energía} = P \cdot t$$

$$E(AA) = 3 \cdot 5,5 \text{ ms} = 16,5 \cdot 10^{-3} \text{ J}$$

$$E(AS) = 3 \cdot 5 \text{ ms} = 15 \cdot 10^{-3} \text{ J}$$

$$N(AA) = 5 \cdot 10^6 \text{ instrucciones}$$

$$N(AS) = 7,5 \cdot 10^6 \text{ instrucciones}$$

$$CPI(AA) = 2,2 \text{ c/i}$$

$$CPI(AS) = 1,33 \text{ c/i}$$

$$T(AA) = 5,5 \text{ ms}$$

$$T(AS) = 5 \text{ ms}$$

Problema Rendimiento y Consumo

Tenim un codi en MIPS que treballa sobre matrius. Hem fet dues versions del codi, una usant accés aleatori (AA) i una altra usant accés seqüencial (AS). El total d'instruccions utilitzades a cada versió les podeu trobar a la següent taula:

	Load / Store	Mult	Salts que salten	Salts que no salten	Altres instruccions
AA	500.000	250.000	250.000	250.000	3.750.000
AS	500.000	0	500.000	500.000	6.000.000

Suposem que a la nostra arquitectura, cada instrucció de memòria costa 5 cicles, cada multiplicació 16 cicles, els salts 1 o 2 cicles depenen de si salta (2) o no salta (1) i la resta d'instruccions 1 cicle.

- d) Ens diuen que es pot dissenyar una arquitectura alternativa on podríem reduir el temps de cicle de les operacions de multiplicació. Quin és el màxim nombre de cicles de multiplicació admissible per a la nova arquitectura per tal que el codi AA sigui al menys igual de ràpid que el codi AS?

$$\text{Necesitamos que } N(\text{AA}) \cdot CPI(\text{AA}) = N(\text{AS}) \cdot CPI(\text{AS})$$

$$N(\text{AS}) \cdot CPI(\text{AS}) = 10 \cdot 10^6 \text{ ciclos}$$

$$10 \cdot 10^6 = (0,5 \cdot 5 + 0,25 \cdot CPI_M + 0,25 \cdot 2 + 0,25 \cdot 1 + 3,75 \cdot 1) \cdot 10^6$$

$$10 = 2,5 + 0,5 + 4 + 0,25 \cdot CPI_M \Rightarrow CPI_M = 3/0,25 = 12 \text{ c/i}$$

$$N(\text{AA}) = 5 \cdot 10^6 \text{ instrucciones}$$

$$N(\text{AS}) = 7,5 \cdot 10^6 \text{ instrucciones}$$

$$CPI(\text{AA}) = 2,2 \text{ c/i}$$

$$CPI(\text{AS}) = 1,33 \text{ c/i}$$

$$T(\text{AA}) = 5,5 \text{ ms}$$

$$T(\text{AS}) = 5 \text{ ms}$$

Problema Rendimiento y Consumo

Un processador ha estat dissenyat per poder funcionar correctament sols a les següents combinacions de freqüències i voltatges d'alimentació (les freqüències estan expressades en forma de fracció per facilitar la simplificació dels càlculs, sense la calculadora):

combinació	voltatge (V)	freqüència (GHz)
1	1,0	6/3
2	1,1	7/3
3	1,2	8/3
4	1,3	9/3

Suposem que la potència estàtica consumida és zero. Sabent que la potència dinàmica consumida amb la combinació número 1 és de $P_1 = 60\text{W}$, es demana:

- Quines són les potències dinàmiques consumides en les combinacions 2, 3 i 4, en watts?
Quina és la combinació amb màxima freqüència i voltatge a la qual podrà funcionar la CPU sense sobrepassar el màxim consum permès pel dissipador, que són 120W?
- (0,5 pts) Quin és el guany de rendiment (o speedup) que s'obté amb la combinació número 4 respecte de la combinació número 1, executant el mateix programa?
- (0,5 pts) Amb la combinació número 1, quin és el temps d'execució (en segons) d'un programa que executa 10^{10} instruccions i que té un CPI promig de 4?

Problema Rendimiento y Consumo

Un processador ha estat dissenyat per poder funcionar correctament sols a les següents combinacions de freqüències i voltatges d'alimentació (les freqüències estan expressades en forma de fracció per facilitar la simplificació dels càlculs, sense la calculadora):

combinació	voltatge (V)	freqüència (GHz)
1	1,0	6/3
2	1,1	7/3
3	1,2	8/3
4	1,3	9/3

Suposem que la potència estàtica consumida és zero. Sabent que la potència dinàmica consumida amb la combinació número 1 és de $P_1 = 60\text{W}$, es demana:

- a) Quines són les potències dinàmiques consumides en les combinacions 2, 3 i 4, en watts?

Sabemos que la Potencia dinámica es $P = \alpha \cdot C \cdot V^2 \cdot F$

Desconocemos $\alpha \cdot C$, pero sabemos que $P_1 = 60\text{ W}$,

En consecuencia, $\alpha \cdot C = P / (V^2 \cdot F) = 60 / (1^2 \cdot 2 \cdot 10^9) = 30 \cdot 10^{-9} \text{ F}$

Usando este valor tenemos que:

- $P_2 = 30 \cdot 10^{-9} \cdot 1,1^2 \cdot (7/3) \cdot 10^9 = 84,7 \text{ W}$
- $P_3 = 30 \cdot 10^{-9} \cdot 1,2^2 \cdot (8/3) \cdot 10^9 = 115,2 \text{ W}$
- $P_4 = 30 \cdot 10^{-9} \cdot 1,3^2 \cdot (9/3) \cdot 10^9 = 152,1 \text{ W}$

Problema Rendimiento y Consumo

Un processador ha estat dissenyat per poder funcionar correctament sols a les següents combinacions de freqüències i voltatges d'alimentació (les freqüències estan expressades en forma de fracció per facilitar la simplificació dels càlculs, sense la calculadora):

combinació	voltatge (V)	freqüència (GHz)
1	1,0	6/3
2	1,1	7/3
3	1,2	8/3
4	1,3	9/3

Suposem que la potència estàtica consumida és zero. Sabent que la potència dinàmica consumida amb la combinació número 1 és de $P_1 = 60W$, es demana:

Quina és la combinació amb màxima freqüència i voltatge a la qual podrà funcionar la CPU sense sobrepassar el màxim consum permès pel dissipador, que són 120W?

- $P_1 = 60 \text{ W}$
- $P_2 = 30 \cdot 10^{-9} \cdot 1,1^2 \cdot (7/3) \cdot 10^9 = 84,7 \text{ W}$
- $P_3 = 30 \cdot 10^{-9} \cdot 1,2^2 \cdot (8/3) \cdot 10^9 = 115,2 \text{ W}$ 
- $P_4 = 30 \cdot 10^{-9} \cdot 1,3^2 \cdot (9/3) \cdot 10^9 = 152,1 \text{ W}$

Problema Rendimiento y Consumo

Un processador ha estat dissenyat per poder funcionar correctament sols a les següents combinacions de freqüències i voltatges d'alimentació (les freqüències estan expressades en forma de fracció per facilitar la simplificació dels càlculs, sense la calculadora):

combinació	voltatge (V)	freqüència (GHz)
1	1,0	6/3
2	1,1	7/3
3	1,2	8/3
4	1,3	9/3

Suposem que la potència estàtica consumida és zero. Sabent que la potència dinàmica consumida amb la combinació número 1 és de $P_1 = 60\text{W}$, es demana:

- b) (0,5 pts) Quin és el guany de rendiment (o speedup) que s'obté amb la combinació número 4 respecte de la combinació número 1, executant el mateix programa?

Mismo programa y misma CPU, implica mismo número de instrucciones y CPI.

Sólo cambia la frecuencia.

$$\text{Speedup} = T_1 / T_4 = (N_{\text{ciclos}} 1 / F_1) / (N_{\text{ciclos}} 2 / F_4) = 9/6 = 1,5$$

Problema Rendimiento y Consumo

Un processador ha estat dissenyat per poder funcionar correctament sols a les següents combinacions de freqüències i voltatges d'alimentació (les freqüències estan expressades en forma de fracció per facilitar la simplificació dels càlculs, sense la calculadora):

combinació	voltatge (V)	freqüència (GHz)
1	1,0	6/3
2	1,1	7/3
3	1,2	8/3
4	1,3	9/3

Suposem que la potència estàtica consumida és zero. Sabent que la potència dinàmica consumida amb la combinació número 1 és de $P_1 = 60\text{W}$, es demana:

- c) (0,5 pts) Amb la combinació número 1, quin és el temps d'execució (en segons) d'un programa que executa 10^{10} instruccions i que té un CPI promig de 4?

$$T_{exe} = N \cdot CPI \cdot T_c = N \cdot CPI / F$$

$$T_{exe} = 10^{10} \cdot 4 / 2 \cdot 10^9 = 40 / 2 = 20 \text{ s}$$



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Departament d'Arquitectura de Computadors

Estructura de Computadores

Tema 1: Rendimiento, Ley de Amdahl y Consumo

Agustín Fernández

Departament d'Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya

