Resumen final - Arquitectura de los Computadores (Fórmulas)

Tema 1. Introducción

- Ley de Amdahl:

$$Aceleración\,rendimiento = \frac{Rendimiento\,con\,mejora}{Rendimiento\,\text{sin}\,mejora} = \frac{Tiempo\,ejecución\,\text{sin}\,mejora}{Tiempo\,ejecución\,con\,mejora}$$

Tema 2. Análisis del Rendimiento

- Concepto de rendimiento: el tiempo es la medida más fiable del rendimiento

$$Rendimiento = \frac{1}{tiempo}$$

- Relación de rendimientos entre máquinas:

"X es más rápida que Y" $t_{ex} < t_{ey}$

"X es n% más rápida que Y" $t_{ex}\ n\% < t_{ey}$

Porcentaje incremental: $tiempo\,ejecución_X + \frac{n}{100} tiempo\,ejecución_X = tiempo\,ejecución_Y$

Aceleración: $\frac{tiempo\,ejecución_Y}{tiempo\,ejecucion_X} = 1 + \frac{n}{100}$

En términos de rendimiento:

$$1 + \frac{n}{100} = \frac{tiempo\,ejecución_Y}{tiempo\,ejecución_X} = \frac{\frac{1}{Rendimiento_Y}}{\frac{1}{Rendimiento_X}} = \frac{Rendimiento_X}{Rendimiento_Y}$$

$$n = 100\frac{Rendimiento_X - Rendimiento_Y}{Rendimiento_Y}$$

Expresado en tiempos de ejecución: $n = 100\frac{Tiempo\,ejecución_Y - Tiempo\,ejecución_X}{Tiempo\,ejecución_X}$

- Incremento anual: $\Delta_{anual} = \sqrt[n]{\frac{rend_{an}}{rend_{a0}}} = \sqrt[n]{\frac{te_{a0}}{te_{an}}}$

- Ley de Amdahl:

Fracción mejorada: (siempre menor o igual a 1)

Aceleración mejorada: (mayor que 1)

Tiempo de ejecución nuevo:

$$TE_{nuevo} = TE_{antiguo}\left(\left(1 - Fracción_{mejorada}\right) + \frac{Fracción_{mejorada}}{Aceleración_{mejorada}}\right)$$

$$Aceleración_{global} = \frac{TE_{antiguo}}{TE_{nuevo}} = \frac{1}{\left(1 - Fracción_{mejorada}\right) + \frac{Fracción_{mejorada}}{Aceleración_{mejorada}}}$$

$$fracción_{mejorada} = 1 \rightarrow aceleración_{global} = aceleración_{mejorada}$$

$$fracción_{mejorada} = 0 \rightarrow aceleración_{global} = 1$$

- Coste de un circuito integrado:

$$Coste\,de\,CI = \frac{Coste\,dado + Coste\,dado\,prueba + Coste\,empaquetado\,y\,final\,test}{Prueba\,rendimiento\,final}$$

$$Coste\,del\,dado = \frac{Coste\,de\,oblea}{Dados\,por\,oblea * rendimiento\,dado}$$

$$Dados\,por\,oblea = \frac{\pi\left(Diámetro\,oblea/2\right)^2}{Área\,del\,dado} - \frac{\pi * Diámetro\,oblea}{\sqrt{\left(2 * Área\,dado\right)}}$$

$$Rendimiento\,dado = \frac{Rendimiento\,oblea * 1}{\left(1 + Defectos\,por\,unidad\,de\,área * Área\,dado\right)^N}$$

- Tiempo de programa/CPU/CPI:

$$Tiempo_{CPU} = Ciclos\,reloj\,CPU\,para\,un\,programa * Duración\,ciclo\,de\,reloj$$

$$Tiempo_{CPU} = \frac{Ciclos\,reloj\,CPU\,para\,un\,programa}{Frecuencia\,de\,reloj}$$

$$CPI = \frac{Ciclos\,de\,reloj\,CPU\,para\,un\,programa}{Recuento\,de\,instrucciones}$$

$$Tiempo_{CPU} = Recuento\,de\,Instrucciones * CPI * Duración\,del\,ciclo\,de\,reloj$$

$$Tiempo_{CPU} = RI * CPI * CLK$$

- Alternativas para la medida del rendimiento:

MIPS: Millones de instrucciones por segundo: $MIPS = \frac{Recuento\,Instrucciones}{Tiempo\,ejecución * 10^6}$

Considerando: $Recuento\,Instrucciones = \frac{tiempo\,ejecución}{CPI * ciclo\,de\,reloj}$

$$MIPS = \frac{Frecuencia\,de\,reloj}{CPI * 10^6}$$

- MIPS relativos y MIPS nativos:

Tiempo de referencia: tiempo ejecución de un programa en la máquina de referencia

Tiempo no estimado: tiempo ejecución del mismo programa en la máquina que se va a medir.

MIPS: Estimación de los MIPS en la máquina de referencia

$$MIPS_{relativos} = \frac{Tiempo_{referencia}}{Tiempo_{no\ estimado}} * MIPS_{referencia}$$

Los MIPS relativos se apoyan en el tiempo de ejecución.

- FLOPS:

Flops: Operaciones de punto flotante por segundo.

$$MFLOPS = \frac{N\acute{u}mero\ de\ operaciones\ en\ punto\ flotante\ de\ un\ programa}{Tiempo\ ejecuci\acute{o}n * 10^6}$$

$$GFLOPS = \frac{MFLOPS}{10^6}$$

- Resúmenes del rendimiento:

Tiempo medio de ejecución: $\frac{1}{n}\sum_{i=1}^{n} Tiempo_i$

Tiempo de ejecución ponderado: $\sum_{i=1}^{n} w_i * Tiempo_i$ donde:

w_i = frecuencia del programa i-ésimo de la carga de trabajo

$Tiempo_i$ = tiempo ejecución del programa i-ésimo

Media geométrica: $\sqrt[n]{\prod_{i=1}^{n} Tiempo_i}$ $\frac{MG(x_i)}{MG(y_i)} = MG(\frac{x_i}{y_i})$

Tema 3. Diseño del repertorio de instrucciones

-Codificación de las instrucciones (DLX):

Instrucción tipo I: (aritmético-logicas)

6	5	5	16
Cod. op.	RS1	RD	Inmediato

- Cargas y almacenamientos (byte, media palabra, palabra)
- ALU's con operandos inmediatos
- Instrucciones de salto condicional (BQEZ, BNEZ) – RS1 registro implicado, RD no se utiliza.
- Saltos a registro – RD=0; Inmediato=0; RS1=destino

Instrucción tipo R: (acceso a memoria, saltos condicionales, inmediatas)

6	5	5	5	11
Cód. op.	RS1	RS2	RD	func

- Aritméticas y lógicas entre registros: RS1=fuente1, RS2=fuente2, RD=RegistroDestino,func=operación del flujo de datos

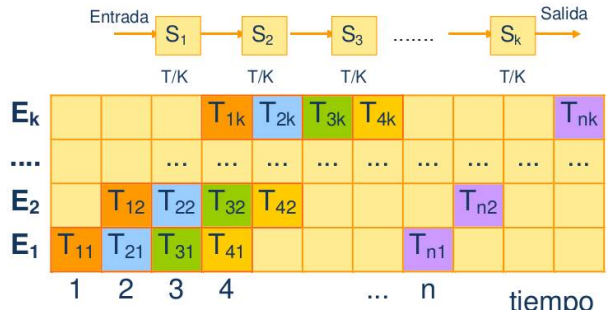
Instrucción tipo J: (saltos incondicionales)

6	26
Cód. op.	Desplazamiento añadido al PC

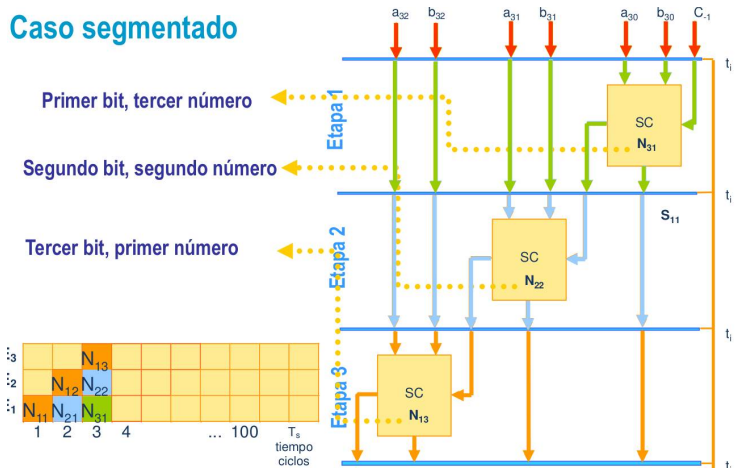
- Instrucciones de salto
- Desplazamiento de 26 bits con signo añadido al PC
- JAL – Salto incondicional y enlace R31
- J – Salto incondicional
- Trap - Interrupciones

Tema 4. Segmentación

- **Segmentación:** el comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa.



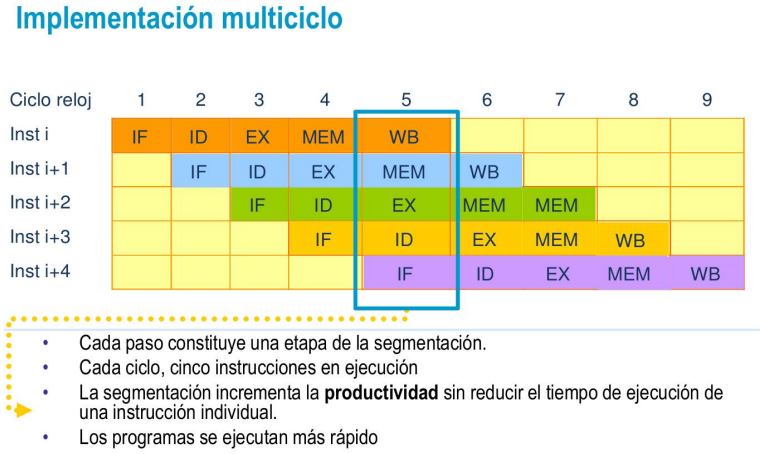
- **Segmentación aritmética:** sumador de tres bits con propagación de acarreo



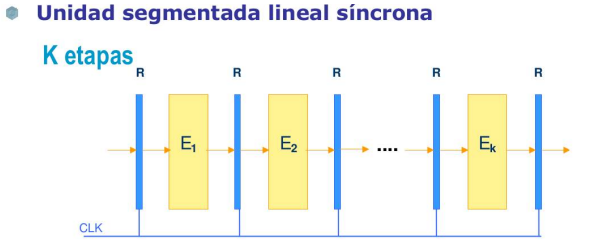
Tiempo secuencial para 100 números: $T_{secuencial} = 100(2T_c + T_s) = 200T_s \rightarrow Si\ T_c \approx 1/2\,T_s$

Tiempo segmentado para 100 números: $T_{segmentado} = 3T_s + 99T_s = 102T_s$

- **Segmentación de instrucciones:** consiste en solapar la ejecución de las instrucciones.



- **Análisis de prestaciones:**



R = Registros de almacenamiento intermedio

E_i = Circuitos combinacionales para operaciones

CLK = Señal de reloj que controla el flujo de datos.

t_i = Retardo temporal de cada etapa E_i

t_r = Es el retardo de cada registro R.

Periodo del reloj del cauce: $CLK = max[t_i]_{i=1}^{k} + t_r$

Sesgo de reloj (retardo del pulso): $CLK \geq max[t_i]_{i=1}^{k} + t_r + s$

Tiempo para procesar n tareas: $T_{SEG} = k * CLK + (n-1) * CLK$

Tiempo equivalente para un proceso no encauzado: $T_{SEC} = k * CLK * n$

Ganancia de velocidad de un cauce de k etapas: $G_k = \frac{T_{SEC}}{T_{SEG}} = \frac{n*k*CLK}{(k+n-1)CLK} = \frac{n*k}{k+n-1}$

Eficiencia: $T_{ocupado} = k*n*CLK \qquad T_{total} = k*(k+n-1)*CLK$

$$E_k = \frac{k*n*CLK}{k(k+n-1)CLK} = \frac{n}{k+n-1} = \frac{G_k}{k}$$

Productividad: (nº instrucciones por unidad de tiempo) $P_k = \frac{n}{(k+n-1)CLK} = \frac{E_k}{CLK}$

- Segmentación básica de MIPS

Iniciar una instrucción cada ciclo introduce problemas. Hay que determinar que operación esta realizando la máquina en cada ciclo, para asegurarnos que no esta intentando realizar dos operaciones diferentes con el mismo recurso.

Máquina sin segmentación:

$$Tiempo\,de\,ejecución\,medio\;por\;instrucción = T_{emi} = CLK * CPI$$

Implementación segmentada: El ciclo de reloj debe ir a la velocidad de la etapa más lenta más sobrecargas. Por eso la ganancia para la segmentación es:

$$G_s = \frac{tiempo\,medio\,instrucción \operatorname{sin} segmentación}{tiempo\,medio\,instrucción\,con\,segmentación}$$

La segmentación puede entenderse como una mejora del CPI, que es lo que típicamente entendemos o como una reducción del ciclo de reloj.

- Rendimiento de la segmentación con detenciones

$$G_s = \frac{tiempo\,medio\,instrucción \operatorname{sin} segmentación}{tiempo\,medio\,instrucción\,con\,segmentación}$$

$$G_s = \frac{CPI \operatorname{sin} segmentación * Ciclo\,de\,reloj \operatorname{sin} segmentación}{CPI\,con\,segmentación * Ciclo\,de\,reloj\,con\,segmentación}$$

$$G_s = \frac{Ciclo\,de\,reloj \operatorname{sin} segmentación}{Ciclo\,de\,reloj\,con\,segmentación} \cdot \frac{CPI \operatorname{sin} segmentación}{CPI\,con\,segmentación}$$

$$CPI\,con\,segmentación = CPI_{ideal} + Ciclos\,reloj\,detención\,segmentación\;por\,instrucción$$

Si ignoramos el incremento potencial en el ciclo de reloj debido a la segmentación, y asumimos que las etapas están equilibradas, podemos igualar el ciclo de reloj de las dos máquinas:

$$G_s = \frac{CPI \operatorname{sin} segmentación}{CPI_{ideal} + Ciclos\,reloj\,detención\,segmentación\,por\,instrucción}$$

$$G_s = \frac{Profundidad\,de\,la\,segmentación}{CPI\,con\,segmentación}$$

$$Profundidad\,de\,la\,segmentación = \frac{Ciclo\,de\,reloj \operatorname{sin} segmentación}{Ciclo\,de\,reloj\,con\,segmentación}$$

$$Tiempo\,medio\,por\,instrucción\,ideal \operatorname{sin} detenciones = Ciclo\,de\,reloj\,ideal$$

Tema 5.Rendimiento de la jerarquía de memoria

- Definiciones:

Un acierto (hit) es un acceso con éxito a memoria del nivel superior, en caso contrario se produce un fallo (miss)

La tasa de aciertos (Hit Ratio) es el porcentaje de aciertos en accesos a memoria del nivel superior

La tasa de fallos (Miss Ratio) se define como: **Miss Ratio: 1 – Tasa de aciertos**

El tiempo de acierto (TA) = T. empleado en determinar si la información esta a ese nivel + T. empleado en acceder a esa información

La penalización de Fallo (PF) = T. empleado en sustituir un bloque del nivel superior por el bloque correspondiente del nivel inferior + T. proporcionar el bloque al dispositivo que lo ha pedido (CPU).

Se distingue: Tiempo de acceso (para acceder a la primera palabra del bloque) y tiempo de transferencia (para transferir el resto del bloque)

El tiempo medio de acceso a memoria (TMA) = tiempo de acierto + tasa de fallos * penalización de fallo

- Memoria caché:

Correspondencia directa: cada bloque sólo puede aparecer en un lugar en la caché

línea = dirección de la estructura de bloque MOD nº de líneas

Correspondencia asociativa por conjuntos: un bloque se puede colocar en un conjunto restringido de lugares en la caché:

línea = dirección de la estructura de bloque MOD nº de conjuntos

Correspondencia completamente asociativa: un bloque se puede colocar en cualquier parte de la caché

- Rendimiento de la caché:

Tiempo medio de acceso a memoria: **TMA = TA + FF * PF**

Tiempo de ejecución: **Tcpu = NI * CPI * Treloj = NI * (CPIejec + CPImem) * Treloj**

CPImem: ciclos en espera de CPU por referencias a memoria:

$$CPI_{mem} = \frac{ciclos\ detención\ debido\ a\ fallos}{N\,I} = CPI_{mem}(lecturas) + CPI_{mem}(escrituras)$$

$$CPI_{mem}(lecturas) = \frac{fallos\ en\ lecturas \cdot PF_{lecturas}}{N\,I} = \frac{lecturas \cdot FF_{lecturas} \cdot PF_{lecturas}}{N\,I}$$

$$CPI_{mem}(escrituras) = \frac{fallos\ escritura \cdot PF_{escritura}}{N\,I} = \frac{escrituras \cdot FF_{escrituras} \cdot PF_{escrituras}}{N\,I}$$

$$CPI_{mem} = \frac{NM \cdot FF \cdot PF}{N\,I}$$

$$T_{CPU} = N\,I \cdot (CPI_{ejec} + CPI_{mem}) \cdot T_{reloj} = N\,I \cdot (CPI_{ejec} + \frac{NM}{N\,I} \cdot FF \cdot PF) \cdot T_{reloj}$$

Álgunos diseñadores prefieren medir los fallos por instrucción en lugar de los fallos por acceso a memoria:

$$T_{CPU} = N\,I \cdot \left(CPI_{ejec} + \frac{fallos}{instrucción} \cdot PF\right) \cdot T_{reloj}$$

- **Optimizaciones:**

Tasa de Fallo = Nº Fallos / Nº de Accesos

Reducir la penalización de fallos con cachés multinivel:

$$TMA = TA_{L1} + FF_{L1} * PF_{L1}$$

$$PF_{L1} = TA_{L2} + FF_{L2} * PF_{L2}$$

Hay que diferenciar:

$$FF_{local} = \frac{nº\,de\,fallos}{nº\,de\,accesos\,a\,la\,caché} \qquad FF_{global} = \frac{nº\,de\,fallos}{nº\,total\,de\,accesos\,realizados\,por\,la\,CPU}$$

En general se cumple:

$$FF_{local} \geq FF_{global}$$

Y en particular:

$$FF_{local.L1} = FF_{global.L1} \qquad FF_{local.L2} > FF_{global.L2}$$