Arquitectura de los Computadores

Subtitulo

Universitat d'Alacant
Universidad de Alicante

Resumen

Explicación del documento

Eduardo Espuch

Curso 2019-2020

Índice

1. Introducción

En este tema daremos conceptos básicos sobre la arquitectura de los computadores, algunas funciones e iniciarnos en el diseño de los computadores.

1.1. Arquitectura de computadores

Para este apartado vamos a:

1. Ver el concepto de arquitectura acuñado por IBM.
2. Los distintos niveles propuestos y el producto final de integrarlas en un modelo.
3. Distintas definiciones sobre la arquitectura y relacionados.
4. Clasificación de arquitecturas.
5. Definiciones de paralelismo de datos y funcional.
6. Distintos tipos de computadores
7. Tarea de de diseño, propuesta conjunta de AM-IEEE. Introducir el paradigma de diseño en mas detalle. Ademas de otros conceptos importantes como las decisiones de implementacion y cla importancia de considerar nuevas tendencias.
8. Los principios de diseño de computadores, acelerar el caso comun, ley de rendimiento decrecientes y localidad de referencia.

1.1.1. Niveles de descripción de un computador

El concepto de arquitectura

Introducido por IBM en 1964, concebían como la arquitectura a la parte del repertorio de instrucciones visible por el programador.

[Amadhal,64]. Presentación del IBM S/360. *La arquitectura de un computador es la estructura del computador que un programador en lenguaje maquina debe conocer para escribir un programa correcto.*

Con esto, pretendían unificar las diferentes divisiones de IBM en una arquitectura común, pero se referían exclusivamente al diseño del repertorio de instrucciones. Cabe añadir que presentaba errores de diseño debido a la falta de consideración de aspectos de implementación.

La acabaremos tratando como una disciplina que trata el diseño de maquinas para ejecutar programas con criterios de optimización de rendimiento y coste, teniendo los siguientes aspectos relacionados:

- Diseño del repertorio de instrucciones
- Diseño de la organización funcional
- Diseño lógico
- Implementación

Niveles estructurales de Bell y Newell [Bell, 71]

Se describe el computador mediante una aproximacion por capas, utilizando cada capa los servicios que proporciona la capa del nivel inferior. Se proponen los siquientes niveles:

- De componente
- Electronico
- Digital
- Transferencia entre registros **(RT)**
- Procesador-Memoria-Interconexión **(PMS)**

Niveles de Interpretación de Levy **[Bell 78, De Miguel 01]**

Contemplan al computador desde un punto de vista funcional, constituido por una serie de máquinas virtuales superpuestas.

Cada máquina interpreta las instrucciones de su nivel, proporcionando servicios a la máquina de nivel superior y aprovechando los de la máquina de nivel inferior. Distinguiremos los siguientes niveles:

- Aplicaciones
- Lenguajes de alto nivel
- Sistema Operativo
- Instrucciones máquina
- Microinstrucciones

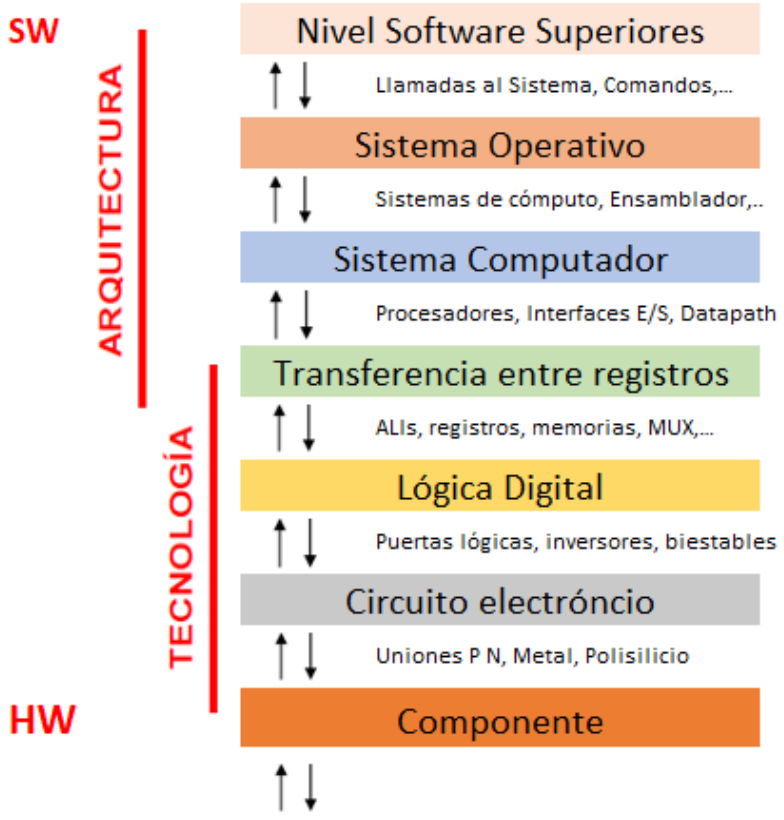
Similar a los niveles funcionales de [**Tanenbaum 86,99,00**]

Niveles de abstracción para un computador

Este nivel integra la orientación estructural de los niveles de Bell y Newell y el punto de vista funcional de los niveles de Levy y TanenBaum.

En el esquema, cada nivel lo vemos conectado a otro mediante el uso de ciertos recursos.

Vamos a ver con mas detalle los niveles.



Componentes físicos:

- Semiconductores de tipo n y p, metales, polisilicio, etc...
- A partir de estos componentes se construyen bloques: transistores, resistencias, etc.
- Las leyes que lo rigen son las de la electrónica física.

Circuito electrónico:

- Puertas lógicas, biestables,etc. Utilizan componentes del nivel anterior.
- Las leyes que lo rigen son las de la electricidad, de naturaleza continua.
- El comportamiento del circuito se describe en términos de corrientes, tensiones y frecuencias.

Lógica digital:

- Las leyes que lo rigen son las del Álgebra de Boole
- Se divide en 2 partes: circuitos combinacionales y secuenciales
  - **Nivel combinacional:** se utilizan como componetes las puertas NAND, NOR, NOT, etc, para generar bloques como, multiplexores, decodificadores, conversores de códigos y circuitos aritméticos.
  - **Nivel secuencial:** se utilizan como componentes elementos de memoria (biestables) y bloques del nivel anterior para obtener circuitos secuenciales, como registros, contadores, memorias, etc.

Transferencia entre registros:

- Estudio del comportamiento de las unidades de un computador en términos de transferencia de información entre registros.
- Utiliza los componentes del nivel anterior, registros, circuitos aritméticos, memorias, etc, para crear componentes del procesador o de otros elementos del computador (interfaces).
- En este nivel se incluye como un posible subnivel la microprogramación.

Sistema Computador:

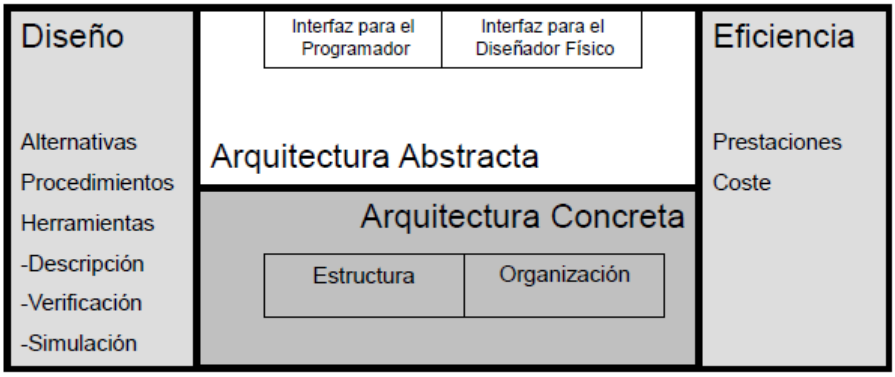
- Especificación de componentes (memoria, procesador, buses, redes de interconexión, periféricos, etc), intercomunicador entre éllos y operación del sistema completo.
- Programación a bajo nivel (lenguaje maquina y ensamblador).

Sistema operativa:

- Interfaz entre hardware y software.
- Encargado de facilitar el uso eficiente de los recursos hardware por parte de los usuarios y de los programas de aplicación.

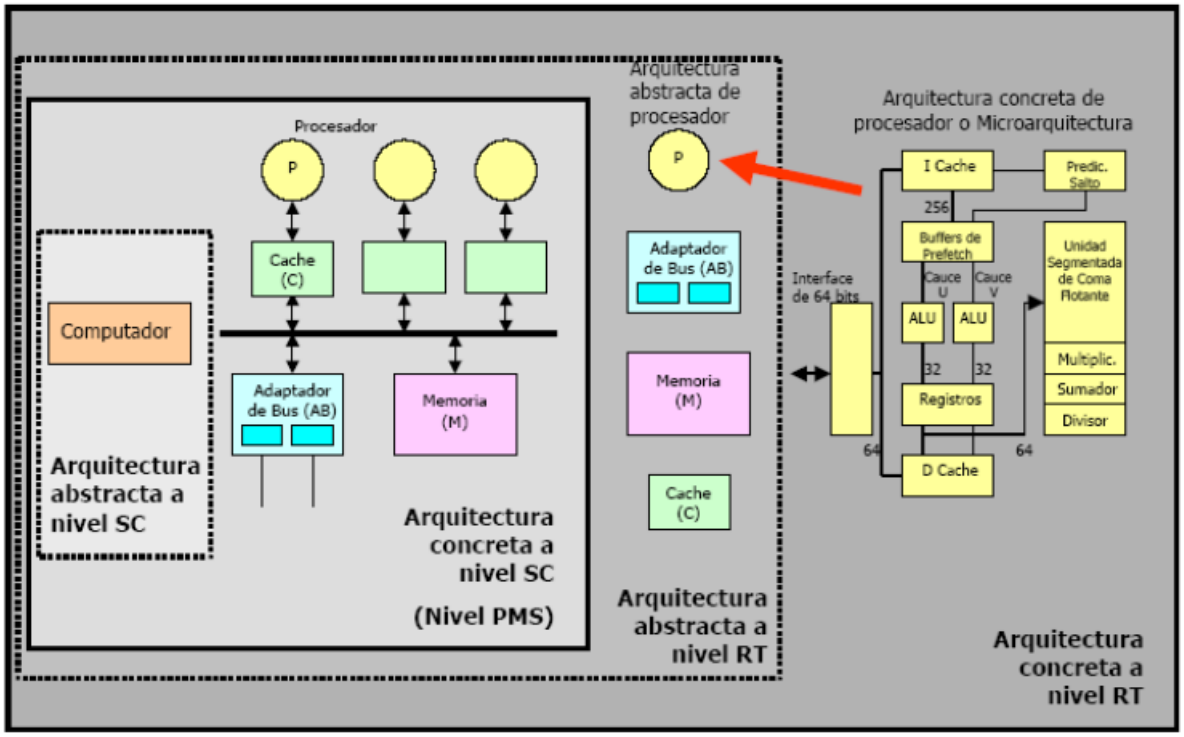
Niveles software Superiores:

- Niveles software: compiladores, programas escritos en lenguajes de alto nivel, etc.
- Realización de programas y compiladores eficientes requieren el conocimiento de la arquitectura del computador, incrementando de prestaciones.



1.1.2. Definición de arquitectura

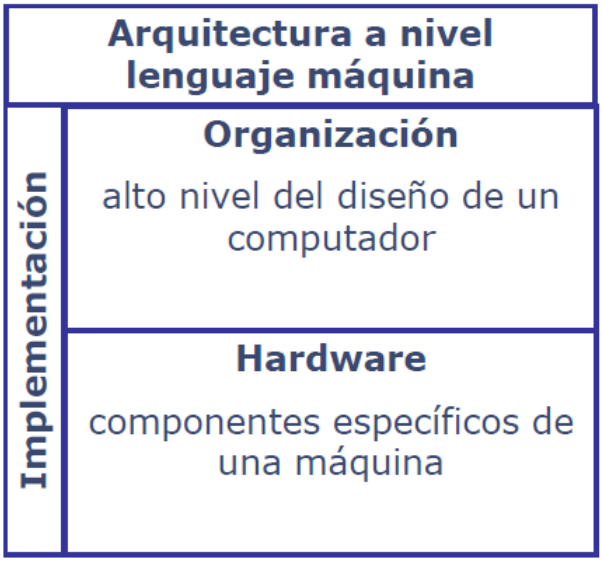
Niveles que abarca la Aquitectura



Definiremos como arquitectura al conjunto de instrucciones, recursos y características del procesador que son visibles al software que se ejecuta en el mismo. Por tanto, la arquitectura determina el software que el procesador puede ejecutar directamente, y esencialmente define las especificaciones a las que debe ajustarse la microarquitectura. Esta definición viene dada por Ortega, en 2005.

Es mas, también definiremos como microarquitectura al conjunto de recursos y métodos utilizados para satisfacer las especificaciones que establece la arquitectura. El termino incluye tanto la forma en que se organizan los recursos como las técnicas utilizadas para alcanzar los objetivos de costes y prestaciones planteados. Con esto se define las especificaciones para la implementacion lógica.

Ámbito de la arquitectura



Repertorio de instrucciones

Sistema de memoria, estructura del Bus, diseño interno de la CPU...

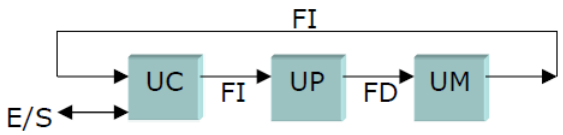
Diseño lógico detallado, la tecnologia de encapsulamiento...

1.1.3. Clasificación de arquitecturas

Como toda clasificación (o taxonomía) persigue dividir el conjunto de los computadores en una serie de clases de forma que, si se sabe la clase a la que pertenece un computador, automáticamente se conocen una serie de características interesantes del mismo.

La taxonomía de Flynn es la mas extendida, compuesta por:

- Computadores SISD (Single Instruction Single Data): un único flujo de instrucciones procesa operandos y genera resultados, definiendo un único flujo de datos.



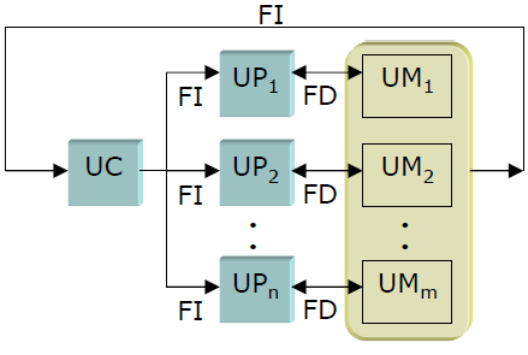

```
for i=1 to 4 do
Begin
   C[i] = A[i]+B[i];
   F[i] = D[i]-E[i];
   G[i] = K[i]*H[i];
End;
```

- Computadores SIMD (Single Instruction Multiple Data): un único flujo de instrucciones procesa operandos y genera resultados, definiendo varios flujos de datos, dato que cada instrucción codifica realmente varias operaciones iguales, cada una actuando sobre operadores distintos.

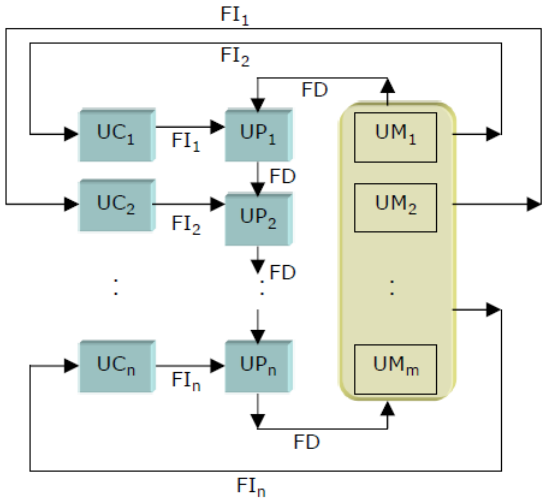
```
for all Epi(i=1 to 4) do
Begin
   C[i] = A[i]+B[i];
   F[i] = D[i]-E[i];
   G[i] = K[i]*H[i];
End;
```

Procesadores matriciales

```
ADDV C,A,B
SUBV F,D,E
MULV G,K,H
```

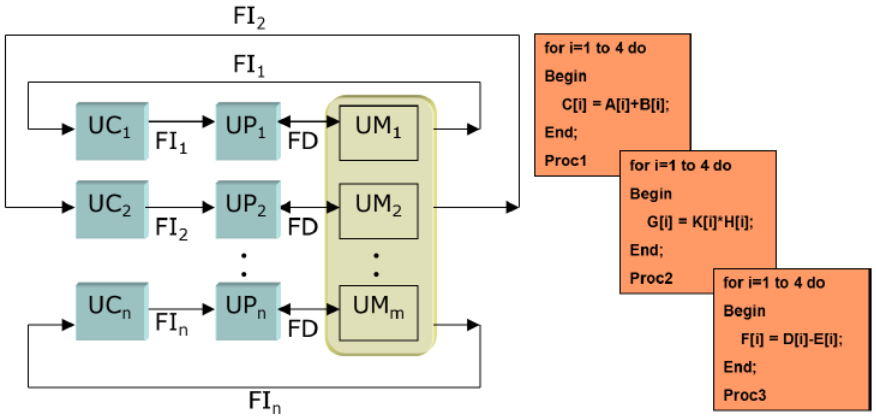


- Computadores MISD (Multiple Instruction Single Data): se ejecutan varios flujos distintos de instrucciones (MI) aunque todos actúan sobre el mismo flujo de datos.



Actualmente no existen computadores que funcionen bajo este esquema.

- Computadores MIMD (Multiple Instruction Multiple Data): el computador ejecuta varias secuencias o flujos distintos de instrucciones, y cada uno de ellos procesa operandos y genera resultados definiendo un único flujo de instrucciones, de forma que existen también varios flujos de datos uno por cada flujo de instrucciones.



Parelelismo de datos: La misma función, instruccion, etc. se ejecuta en paralelo pero en cada una de esas ejecuciones se aplica sobre un conjunto de datos distinto.

Parelelismo funcional: varias funciones, tareas, instrucciones, etc)iguales o distintas) se ejecutan en paralelo. Se distinguen los siguientes niveles (segun el tipo de entidades funcionales que se ejecutan en parelelo):

- Nivel de instrucción (Instruction Level Parallelism ILP), se ejecutan en paralelo las instrucciones de un programa. Granularidad fina.
- Nivel de bucle o hebra (Thread), se ejecutan en parelelo distintas iteraciones de un bucle o secuencias de instrucciones de un programa, Granularidad fina/media.
- Nivel de procedimiento (proceso), los distintos procedimientos que constituyen un programa se ejecutan simultáneamente. Granularidad media.

- Nivel de programa, la plataforma ejecuta en parelelo programas diferentes que pueden corresponder, o no, a una misma aplicación. Granularidad gruesa.

Tipos de computadores

- Dispositivos móviles personales: teléfonos móviles, tablets,... Coste y eficiencia energética.
- Ordenadores sobremesa: precio-rendimiento.
- Servidores: disponibilidad, escalabilidad, rendimiento
- Clusters: LANs de sobremesas y servidores actuando como un gran computador. SaaS: búsquedas, redes sociales, vídeo compartido, juegos multiusuario.
- Embedidos: presentes en maquinas: microondas, lavadoras, impresoras, switches, coches... Amplio espectro coste rendimiento.

1.2. El diseño de computadores

En la tarea de diseño, trataremos la propuesta conjunta ACM-IEEE, una informatica basada en tres paradigmas, cada uno igual de importante.

Teoría: ciencias formales	Abstracción: ciencias experimentales	Diseño: ciencias aplicadas
•Definición •Teorema •Demostración •Interpretación	•Hipótesis •Construcción de un modelo y realización de predicciones •Diseño de experimentos y recogida de resultados •Análisis de resultados	•Establecer requirimientos •Especificar •Realización del sistema •Prueba del sistema

1.2.1. El proceso de diseño de computadores

- Establecer requerimientos funcionales
- Especificar el sistema
- Realizacion del sistema
- Prueba del sistema

Establecer requerimientos funcionales y especificar

Funcionalidades inspiradas por el mercado y el software de aplicación que determinan características especificas del sistema.

Especificación en base a criterios de coste, rendimiento, consumo y disponibilidad para el mercado pensado.

Requerimientos funcionales	Características típicas requeridas o soportadas
Área de aplicación • Móviles • PCs • Servidores • Clusters/Warehouse-Scale Computers • Computación embebida	**Objetivo del computador** • Rendimiento en tiempo real para muchas tareas, incluyendo gráficos, video y audio; eficiencia energética. • Rendimiento equilibrado para muchas tareas, incluyendo gráficos, video, y audio. • Soporte para BB.DD. y transacciones; alta fiabilidad y disponibilidad; escalabilidad • Alta productividad para tareas independientes; corrección de errores en memoria; proporcionalidad energética • A menudo requiere soporte especial para video y audio (y otras extensiones específicas de aplicaciones); limitaciones en consumo y se puede requerir control de potencia; limitaciones de tiempo real.
Nivel de compatibilidad software En lenguaje de programación Código binario compatible	**Determina la cantidad de software existente para la máquina** Más flexible para el diseñador, necesita nuevo compilador La arquitectura está completamente definida (poca flexibilidad), pero no necesita invertir en software ni en portar programas
Requerimientos del S.O. Tamaño del espacio de direcciones Gestión de memoria Cambio de contexto Interrupciones Protección	**Características necesarias para soportar el S.O. requerido** Muy importante, puede limitar aplicaciones Para S.O. modernos; puede ser plana, paginada, segmentada. Requerido para interrumpir y recomenzar un programa Tipos de soporte impactan en el diseño hardware y S.O. Diferentes S.O. y necesidades de aplicación: protección de páginas frente a protección de segmentos.
Estándares Punto flotante Bus E/S Sistema operativo Redes Lenguajes de programación	**Ciertos estándares pueden ser requeridos por el mercado** Formato y aritmética: IEEE 754, aritmética especial para gráficos o procesamiento de señal Dispositivos E/S: Serial ATA, Serial Attach SCSI, PCI Express UNIX, Windows, Linux Distintas redes: Ethernet, Lenguajes (ANSI C, C++, Java, Fortran) afectan al repertorio de instrucciones.

Decisiones de implementación

Una funcionalidad requerida se puede implementar a nivel software o hardware, cada una ofreciendo una serie de ventajas.

Las ventajas de implementacion software son:

- El bajo coste errores
- Facilidad de diseño
- Actualización simple

Las ventajas de implementacion hardware son:

- Rendimiento

Consideración de las nuevas tendencias

Un diseñador debe de ser consciente de las tendencias tiene que estar al corriente de la utilización del computador y las tecnologías de estos.

Aparte de casos en los que surgen nuevas tecnologías y métodos de uso, podemos encontrarnos con casos en los que ciertos conceptos se mantienen durante años, como por ejemplo una arquitectura a nivel lenguaje maquina que duro 50 años, el núcleo de la IBM 360 desde 1964.

Tendencias en tecnologías hardware [Agarwall, 00]

Tecnología	Tendencias de rendimiento y densidad
Tecnología de CI	El número de transistores en un chip aumenta aproximadamente el 35% por año, x4 en 4 años. La velocidad de los dispositivos aumenta casi a esa rapidez. [Agarwall, 00] disminución tasa crecimiento a 12% anual (consecuencia de procesos de 0,035 micras previstos para 2014)
DRAM semiconductora	La densidad aumenta en un 60% por año, cuadruplicándose en tres años. 2011 25-40% x2 cada 2-3 años [Kim, 2005] La duración del ciclo ha mejorado muy lentamente, decreciendo aproximadamente una tercera parte en diez años.
Tecnología de almacenamiento secundario	La densidad del disco magnético aumenta desde 2004 40% año x2 cada 3 años. El tiempo de acceso ha mejorado un tercio en diez años. Crecimiento de los discos SSD (NAND SLC, MLC y TLC)

Ley de Moore

Las velocidades de computo y las densidades de almacenamiento se duplican cada 18 meses.

Tendencias en software

- Creciente cantidad de memoria utilizada por los programas y sus datos
- Sustitución del lenguaje ensamblador por los lenguajes de alto nivel
- Reorientacion de las arquitecturas hacia el soporte de los compiladores

1.2.2. Principios de diseño de computadores

Acelerar el caso común

Con la ley de Amdahl se cuantifica el principio en el cual se optimiza el caso mas frecuente. Por ejemplo, en un programa es poco frecuente que se de una suma con desbordamiento pero si una sin, por lo tanto se optimizaria este ultimo caso.

Ley de Amdahl define la ganancia de rendimiento o aceleración que puede obtenerse al mejorar alguna característica de un computador. La mejora obtenida en el rendimiento al utilizar algún modo de ejecución mas rápido esta limitada por la fracción de tiempo en que se puede utilizar ese modo mas rápido.

$$\text{Aceleración Rendimiento} = \frac{\text{Rendimiento con mejora}}{\text{Rendimiento sin mejora}} = \frac{\text{Tiempo ejecucion sin mejora}}{\text{Tiempo ejecucion con mejora}}$$

Ley de rendimientos decrecientes

La mejora incremental en la aceleración conseguida por una mejora adicional en la rendimiento de una parte del calculo disminuye tal y como se van añadiendo mejoras.

Localidad de referencia

Tendencia de los programas a reutilizar los datos e instrucciones usados recientemente. Los programas suelen emplear el 90 % de su tiempo de ejecución en el 10 % del código.

Menos del 4 % de las instrucciones del programa Spide (instrucciones estáticas) representan el 80 % de las instrucciones dinámicas.

Localidad temporal: Los elementos accedidos recientemente probablemente serán accedidos en un futuro próximo.

Localidad espacial: Los elementos cuyas direcciones son próximas tienden a ser referenciados juntos en el tiempo.

2. Análisis del rendimiento

Con este tema intentaremos:

- Entender el concepto de rendimiento, la evolucion del rendimiento en los compuadores en los utlimos años y su relacion con el coste
- Saber cuantificar la ganancia de rendimiento o acelaracion que puede obtenerse al mejorar alguna caracteristica de un computador
- Mostrar al alumno distintas metricas para evaluar el rendimiento de una arquitectura, observando la realcion que existe entre ellas
- Adquirir conciencia de la necesidad de establecer metricas para llevar a cabo procesos de evaluacion y comparacion objetiva y contrastada de sistemas computacionales

2.1. Rendimiento. Concepto y definiciones

Todos sabemos que en los ultimos 70 años la tecnologia de computadores han tenido un increible progreso, esta mejora se debe fundamentalmente a:

- Avances en la tecnologia usada para construir computadores. Tamaño de los elementos en el chip (feature size), velocidad del reloj.
- Innovaciones en el diseño (menos consistenes). Compiladores de lenguajes de alto nivel, UNIX. Provocado por arquitecturas **R**educed**I**nstruction**S**et**C**omputing.

Aparicion del microporcesador (μP) (finales 1970)

- Capacidad de dirigir los avances en la tecnologia de circuitos integrados, con una tasa mas alta de mejora del rendimiento (35 % anual)
- Ventajas en el coste debido a la producion masiva de μPs, aumentando el numero de computadores basados en éstos
- Cambios significantes:
  - Eliminacion virtual de la programacion en lenguaje ensamblador, reduciendo la necesidad de la compatibilidad en el codigo objeto.
  - Aparicion de sistemas operativos estandarizados como UNIX, reduciendo el coste y el riesgo en la aparicion de una nueva arquitectura.

RISC (Principios 1980)

- Cambios anteriores permiten el dearrollo de forma satisfactoria de un nuevo conjunto de arquitecturas con instrucciones mas simples: arquitecturas RISC(Reduces Instruction Set Computer)

- Los diseñadores de maquinas RISC se centraron en dos tecnicas clave para la mejora del rendimiento:
  - La explotacion del paralelismo a nivel de instruccion
  - El uso de caches
- El aumento del rendimiento forzo a las arquitecturas previas a mantener el ritmo o a desaparecer

Entre los 80 y 00 se produce la mayor constribucion de ideas de organizacion y arquitectura, con un 52 % anual)

Cae la mejora del rendimiento para un solo procesador (Inicios del 2000)

- Maxima disipacion de potencia en chips refrigerados por aire
- Falta de mas parelelismo a nivel de instruccion a explotar eficientemente
- Intel cancelo sus proyectos de monoprocesadores de alto rendimiento y declaro que la via para obtener mayor rendmiento seria a traves de multiples procesadores por chip.

El rendimiento se estanca par un solo procesador 2010

El porcentaje de mejora ya no es significativo, la ley de Amdhal ya no es aplicable en este aspecto.

2.1.1. Concepto de rendimiento

Para un usuario, un computador es mas rápido cuando un programa se ejecuta en menos tiempo. Estará interesado en reducir el **tiempo de respuesta/tiempo de ejecución**, el tiempo transcurrido entre el inicio y el final de un evento.

Para un administrador de un Cluster, un computador es mas rápido cuando completa mas transacciones por hora. Estará interesado en aumentar la **productividad/throughput**, la cantidad total de trabajo realizado en un tiempo determinado.

Para definir el concepto de rendimiento es importante:

- La medida mas fiable del rendimiento es el tiempo
- El tiempo de ejecucion de un programa se mide en segundos
- La relacion entre el tiempo y el rendimiento es inversa
- El rendimiento se mide como una frecuencia de eventos por segundo

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

Relacion de rendimientos entre maquinas

Diremos que X es mas rapida que Y si tenemos que el tiempo de ejecucion (t_e) de X es menor que el de Y

$$t_{eX} < t_{eY}$$

Es mas, podremos decir que X es n % mas rapida de Y cuando tenemos que

$$t_{eX} n\,\% < t_{eY}$$

- Porcentaje incremental.

$$t_{eX} + \frac{n}{100} t_{eX} = t_{eY}$$

- Aceleracion.

$$\frac{t_{eY}}{t_{eX}} = 1 + \frac{n}{100}$$

- Aceleracion en terminos de rendmiento.

$$1 + \frac{n}{100} = \frac{t_{eY}}{t_{eX}} = \frac{\frac{1}{\text{Rendimiento}_Y}}{\frac{1}{\text{Rendimiento}_X}} = \frac{\text{Rendimiento}_X}{\text{Rendimiento}_Y}$$

$$\frac{n}{100} = \frac{\text{Rendimiento}_X}{\text{Rendimiento}_Y} - 1 = \frac{\text{Rendimiento}_X - \text{Rendimiento}_Y}{\text{Rendimiento}_Y}$$

$$n = 100 \frac{\text{Rendimiento}_X - \text{Rendimiento}_Y}{\text{Rendimiento}_Y}$$

- Aceleracion en terminos de tiempo de ejecucion.

$$n = 100 \frac{t_{eY} - t_{eX}}{t_{eX}}$$

Resumen: denotaremos con t_e al tiempo de ejecucion, con *rend* al rendimiento, a n como porcentaje de incremento y a la expresion $1 + \frac{n}{100}$ como aceleracion que se puede obtener del porcentaje incremental.

Lo veo algo lioso como lo han querido expresar, pero con saber que n es el valor con el cual sabremos que X es mas rapido de Y y este valor lo obtenemos de las formulas de aceleracion que puede depender de tiempos de ejecucion o del rendimiento dado en tantos por ciento

De la productividad unicamente quedaros con que la productividad de X sera el n % superior que la de Y, y tomando la funcion de porcentaje incremental (que era para cuando X era mas rapido que Y, la interpretamos como que un computador sera mas productivo para cuando haga el mismo trabajo mas rapido. La formula pone que es $P = nt/t; P_x = mP_y$ pero no se indica de donde sacan las t y la m

La aceleracion se puede interpretar como incrementos si consideramos el rendimiento de un computador en distintos instantes de tiempo(anual generalmente). De otra forma, podemos usar la funcion

$$acel. = \Delta_{anual} =^n \sqrt{\frac{rend_n}{rend_0}} \iff (\Delta_{anual})^n = \frac{rend_n}{rend_0}$$

2.1.2. Ley de Amdhal

Esta relacionado con el prinicipio de diseño de favorecer el caso frecuente, define la ganancia de rendimiento o acelaracion que puede obtenerse al mejorar alguna caracteristica de un computador.

La mejora obtenida en el rendimiento al utilizar algun modo de ejecucion mas rapido esta limitada por la fraccion de tiempo en que se puede utilizar ese modo mas rapido.

La funcion para calcular la aceleracion de rendimiento se deriva de la funcion de rendimiento ya conocida pero considerando X el computador con mejora y a Y sin mejora, entonces

$$acel_{rend} = \frac{rend_{conMej}}{rend_{sinMej}}$$

Dicho esto, consideramos que la aceleracion definida por esta ley depende de dos factores:

- Fraccion mejorada, fraccion de tiempo de la opcion sin la mejora que puede utilizarse para aprovechar la mejora, siempre menor o igual que uno.

  Para esto denotaremos como t_eS al tiempo de ejecucion sin mejora, t_eC al tiempo de ejecucion con mejora, añadiremos g para indicar que es el global, f para indicar que es para la seccion mas frecuente y nf para la que no. Dicho, tendremos que la fraccion mejorada es

  $$F_M = \frac{t_{eSf}}{t_{eSg}}$$

- Aceleracion mejorada, la funcion de la aceleracion cuando unicamente tratamos el tiempo de ejecucion en la seccion mas frecuente, siendo

  $$A_M = \frac{t_{eCf}}{t_{eSf}}$$

El tiempo de ejecucion nuevo del computador con la mejora es el tiempo empleado sin utilizar la parte mejorada t_{eSnf} mas el tiempo empleado utilizando la parete mejorada t_{eCf}. Como se ha propuesto denotarlo previamente, este seria $t_{eCg} = t_{eCnf} + t_{eCf}$, $t_{eSnf} = t_{eCnf}$ porque en la parte no frecuente no se realiza mejora.

Otra formula para calcularlo es usando la fraccion y aceleracion mejorada respecto al anterior tiempo de ejecucion.

$$t_{eCg} = t_{eSg}\left((1 - F_M) + \frac{F_M}{A_M}\right)$$

Con esto, sabemos que la aceleracion definida previamente, la del rendimiento, es la aceleracion global que consideran todo el conjunto del tiempo, la parte con y sin mejora y, sin la fraccion mejorada es 1, sabremos que coincide con la mejorada. En el caso de ser la fraccion mejorada 0, la aceleracion global sera 1 ya que no se produce mejora sobre ninguna seccion.

La formula previamente la hemos introducido con el rendimiento, si lo hacemos con los tiempo, sabiendo que el tiempo de ejecucion nuevo esta en funcion del antiguo, usando esta funcion veremos que:

$$t_{eCg} = t_{eSg}\left((1 - F_M) + \frac{F_M}{A_M}\right) \Leftrightarrow \frac{t_{eSg}}{t_{eCg}} = \left(\frac{1}{(1 - F_M) + \frac{F_M}{A_M}}\right) = A_G$$

2.1.3. Relación entre rendimiento y coste

El diseño de computadores abarca desde el alto coste alto rendimiento hasta el bajo coste bajo rendimiento, consideremos esta tabla:

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of micro-processor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

- Para dispositivos moviles como telefonos, tabletes,... la principal preocupacion es el coste debido al precio del producto
- Para los sitemas embedidos, como microondas o impresonras, es el coste. Los requerimientos de rendimiento existen pero el objetivo principal es encontrar el rendimiento requerido al minimo coste.
- Para ordenadores de sobremesa (alta o baja gama) se debe de optimizar el ratio Coste/Rendimiento, los microprocesadores de mas alto rendimiento y aquellos de coste reducido aparecen en primer lugar en este tipo de computador.
- Servidores, centrados en rendimiento ya que estan diseñoadps para una productividad eficiente. El rendimiento global del servidor es crucial, rendimiento desde el punto de vista de un administrador (transacciones por minuto o pawinas web servidas por minuto).
- La relacion Coste/Rendimiento es crictica, tratamos con una collecion de computadores de sobremesa o servidores los cuales estan conectados por redes de area local y que actuan como un unico computador. Las redes sociales, videojuegos en linea, aplicaciones de busqueda,... depende de esto.
- Para los supercomputadores, el rendimiento es crucial, el coste no es importante. Enfatizar en el rendimiento del punto flotante, capaces de ejecturar grandes programas por lotes que pueden correr por semanas.

El **coste** es un parametro a tener muy en cuenta al diseñar un nuevo procesador o al modificar uno existente, siendo las factores principales:

- **Curva de aprendizaje**: costes de manufacturacion decrecen a lo largo del tiempo incluso sin mejoras en la tecnologia de implementacion basica, incrementandose el porcentaje de dispositivos manufacturados que pasan los procedimientos de prueba a lo largo del tiempo. Basicamente, al realizar mejor tu producto, podras aprovechar mejor el material, reduciendo el coste de realizacion de este.
- **Volumen**: al incrementar el volumen se reduce el tiempo necesario para bajar la curva de aprendizaje (cuanto mas lo hagas, antes aprenderas a ahcerlo mejor), incrementarse la eficiencia de compra y manufactura (un 10 % menos por cada doble de volumen) y reducirse la cantidad del coste de diseño que debe ser amortizado para cada computador.
- **Mercado altamente competitivo**: un mismo componente, o practicamente identico, es vendido por diversos fabricantes, bajando el precio de venta asemejandolo al de coste pero, a su vez, se reduce el coste ya que los componentes tienen tanto un gran volumen como una clara definicion.

Los **costes** de un **circuito integrado (Integrated Circuit)** se estan convirtiendo en una porcion cada vez mas grande del coste que varia entre computadores, siendo los factores que influeyen en el coste del silicio los siguientes:

- **El numero de puertas**: influye en el numero de transistores necesarios. Un aumento de estos requiere un area de silicio mayor.
- **Concexiones entre elementos**: el numero y la longitud.
- **Regularidad del diseño**: cuanto mas regular sea el diseño, menos area ocupara.

Los procesos del **Integrated Circuit** estan caracterizados por *'feature size'*: el tamaño minimo de un transistor o conexion sea en la dimension X o Y(10micras en 1971 a 0.014 micras en 2015)

El proceso básico de fabricacion del silicio no ha cambiado: la oblea es testeada y cortada en dados que son empaquetados. La oblea es una lamina delgada de cristal semiconductor(silicio)

$$\text{Coste del IC} = \frac{\text{Coste del dado} + \text{Coste de testeo del dado} + \text{Coste del empaquetado y testeo final}}{\text{Prueba final de rendimiento(FTY)}}$$

$$\text{Coste del dado} = \frac{\text{Coste de la oblea}}{\text{Dados por oblea x Rendimiento del dado}}$$

El numero de dados por oblea es aproximadamente el area de la oblea dividida por el area del dado, aproximadamente:

$$\text{Dados por oblea} = \frac{\pi \text{ x } (\text{radio de la oblea})^2}{\text{area del dado}} - \frac{\pi \text{x 2 x radio de la oblea}}{\sqrt{2 \text{ x area del dado}}}$$

NOTA: en el excel la formula es

$$\text{Dados por oblea} = \frac{\pi \text{ x } (\text{radio de la oblea})^2}{\text{area del dado}} - \frac{\mathbf{2} \text{ x } \pi \text{ x radio de la oblea}}{\sqrt{2 \text{ x area del dado}}}$$

teniendo en la primera parte area de oblea/area de dado (cuando dados caben dentro de la oblea) y en la segunda parte son aproximadamente los dados en la periferia.

Los defectos estan distribuidos aleatoriamentes y usaremos defectos por unidad de area y el rendimiento es inversamente proporcional a la complejidad del proceso de fabricacion.

$$\text{Rendimiento del dado} = \text{Rendimiento de la oblea x } \left(\frac{1}{(1 + \text{Defectos/ua x Area del dado})^N} \right)$$

con N siendo el factor de complejidad.

2.2. Evaluación del rendimiento

2.2.1. Medidas del rendimiento

Tiempo de ejecucion

El tiempo es la medida mas fiable del rendimiento, midiendose el tiempo de ejecucion de un programa en segundos.

La relacion entre el tiempo y el rendimiento es inversa, el rendimiento midiendose como una frecuencia de eventos por segundo.

- Tiempo de reloj, tiempo de respuesta, tiempo transcurrido: Latencia para contemplar una tarea incluyendolo todo: accesos a disco, accesos a memoria, actividades de entrada salida, gastos del sistema, multiprogramacion...

  El rendimiento del sistema es el termino que se utiliza para referenciar el tiempo transcurrido en un sistema no cargado.

- Tiempo de CPU
  - CPU usuario + CPU sistema
  - tiempo en que la CPU esta calculando sin incluir tiempos de espera para E/S o para ejecucion de otros programas
  - Tiempo de programa: este termino se refiere al tiempo de CPU del usuario (nos centramos en rendimiento de la CPU)

Por ejemnplo, al ejecutar *time* de Unix obtenemos la salida: Xu Ys Z W %, donde:

- Tiempo de CPU del usuario= X
- Tiempo de CPU utilizado por el sistema= Y
- Tiempo de CPU=$X + Y$
- Tiempo de respuesta=Z, expresado en mm:ss
- Tiempo de CPU en funcion al timepo de respuesta=Z x W
- Tiempo esperando operaciones de E/S y/o el tiempo ejecutando otras tareas $100 - W = W'$ del tiempo de respuesta = Z x W'

El tiempo de CPU de un programa puede expresarse en funcion del ciclo de reloj

$$\text{Tiempo de CPU} = \text{CRPrg x clk}$$

$$\text{Tiempo de CPU} = \text{CRPrg}/\text{f}_R$$

siendo CRPrg los ciclos de reloj de CPU para un programa, clk es la duracion del ciclo de reloj y R reloj (me quedaria sin espacio para las funciones si no empiezo a denotar parametros)

No tiene sentido mostrar el tiempo transcurrido en funcion del ciclo de reloj ya que la latencia de los dispositivos de E/S es independiente del ciclo de reloj de la CPU

El CPI (clock Cycles Per Instruction) es el numero promedio de ciclos de reloj por instruccion, expresandose en funcion del numero de ciclos de reloj y el numero de instrucciones ejecutadas

$$CPI = \text{CRPrg}/\text{RI}$$

siendo RI recuento de instrucciones, el numero total de instrucciones realizadas.

Con esto, podemos expresar el tiempo de CPU en funcion del CPI

$$T_{CPU} = \text{CRPrg x T}_{CR} = RI \text{ x } CPI \text{ x clk}$$

En ocasiones se detalla el CPI por cada tipo de instruccion estatica i

$$\text{Ciclos de reloj de la CPU} = \sum_{i=1}^{n} (CPI_i \text{x} I_i)$$

I_i =instrucciones dinamaicas para cada tipo de instruccion estatica i

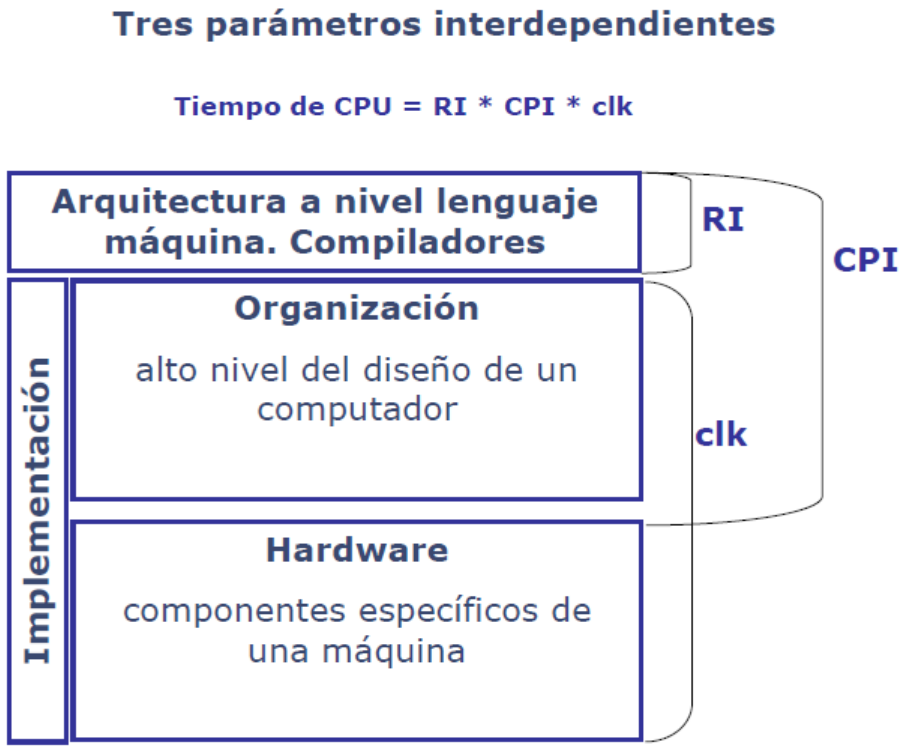
CPI_i =Numero medio de ciclos de reloj para la instruccion tipo i

Sabiendo la cantidad de ciclos de reloj que realiza la CPU y la duracion de estos ciclos, podemos obtener el tiempo de ejecucion con esto

$$T_{CPU} = \sum_{i=1}^{n}(CPI_i \text{x} I_i) \text{ x clk}$$

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \text{x} I_i)}{RI} = \sum_{i=1}^{n}\left(CPI_i \text{x} \frac{I_i}{RI}\right)$$

CPI_i medido, no obtenido de tabla de referencia. Deben considerarse fallos de cache y demas incidencias del sistema de memoria.



La medida mas fiable del rendimiento es el timepo de ejecucion de los programas reales pero alternativas como objetos de medida han conducido a errores en el diseño. Algunas alternativas son:

- **MIPS** Millones de instrucciones por segundo, se muestran como un parametro intuitivo para reflerejar el rendimiento. Maquinas mas rapidas tienen MIPS mas altos.

  Es una medida dependiente al repertorio de instrucciones por lo tanto no es aconsejable comparar los los MIPS de computadores con repertorios de instrucciones diferentes.

  - Reflejan el ritmo de ejecucion de intrucciones
  - No reflejan la efectividad del reperotio de recuento de instrucciones
  - Los MIPS pueden variar inversamente al rendimiento.

$$MIPS = \frac{RI}{t_e \text{ x } 10^6} = \frac{\frac{t_e}{CPI \text{ x } clk}}{t_e \text{ x } 10^6} = \frac{1}{CPI \text{ x } clk \text{ x } 10^6} = \frac{f_{clk}}{CPI \text{ x } 10^6}$$

Hz	KHz	MHz	GHz
1Hz	10^3Hz	10^6Hz	10^9Hz
1^{-1}s	10^{-3}Hz	10^{-6}Hz	10^{-9}Hz

Si queremos comparar dos computadores tenemos que tener en cuenta

- Tiempo referencia, tiempo de ejecucion de un programa en la maquina de referencia t_{ref}
- Tiempo no estimado, timepo de ejecucion del mismo programa en la maquina que se va a medir t_{NE}
- $MIPS_{ref}$, estimacion de los MIPS de la maquina de referencia

$$MIPS_{rel} = \frac{t_{ref}}{t_{NE}} \text{ x } MIPS_{ref}$$

- Los MIPS relativos se apoyan en el tiempo de ejecucion.

VAX-11/780 era la maquina dominante como referenciam con MIPS=1

- FLOPS, Operaciones de punto flotante por segundo, pero manipularemos principalmente por millones de segundo

$$MFLOPS = \frac{\text{nº operacions en punto flotante de un programa}}{t_e \text{ x } 10^6}$$

Veamos algunas problemas derivados de la utilizacion de los FLOPS:

- Son dependientes del repertorio de instrucciones, es un termino basado en operaciones con objetivo de de poder para comparar diferentes maquinas, pero el conjunto de este tipo de operaciones no es consistente en diferentes maquinas.
- Dependencia del programa, la estimacion de los MFLOPS cambia segun la mezcla de operaciones rapidas y lentas en punto flotante del programa, es decir, no es igual de rapido un programa con el codigo de operaciones rapidas y otro con operaciones lentas.

La solucion para esto es usar operaciones normalizadas las cuales tienen en cuenta que, segun el tipo de operacion, el numero de operaciones ejecutadas se trata con un valor constante dado por la siguiente tabla:

Operaciones reales PF	Operaciones normalizadas PF
ADD, SUB, COMPARE, MULT	1
DIVIDE, SQRT	4
EXP, SIN	8

$$MFLOPS_{\text{nativos}} = \frac{O_{PF}}{t_e \text{ x } 10^6}$$

$$MFLOPS_{\text{nativos}} = \frac{O_{PF\text{tipo 1}} + O_{PF\text{tipo 4}} \text{ x } 4 + O_{PF\text{tipo 8}} \text{ x } 8}{t_e \text{ x } 10^6}$$

2.2.2. Programas para evaluar el rendimiento

Tiempo de ejecucion de la carga de trabajo del usuario (workload) que es una mezcla de programas y ordenes del SO

- Programas reales, como compiladores de C, software de tratamiento de textos comoTeX (ahora mismo estoy usando eso) y herramientas CAD como Spice.
- Nucleos (Kernels), pequeños fragmentos clave de programas, como Livermore Loops y Linpack
- Benchmark reducidos (toys) 10 y 100 lineas de codigo. Criba de eatostenes, Puzzle y clasificacion rapida (quicksort)
- Benchmarks sinteticos, que se crean artificalmente intentando simular una frecuencia media de operaciones y operandos de un gran conjunto de programas, como Whetstone y Dhrystone.

Importancia para las empresas de los benchmarks: empleo de recursos para optimizar el funcionamiento de los benchmarks pero no de programas reales. Si se utilizaran programas reales para evaluar el rendimiento, las mejoras repercutirian en el usuario final.

Las razones para usar benchmarks pequeños son:

- Portabilidad: en el pasado lenguajes de programacion inconsistenes entre maquinas dificultando el transporte+
- Facil simulacion; cuando se diseña una nueva maquina
- Estandarizacion: es mas facil

Actualmente, la popularidad de los sitemas operativos estandares elimina la principal dificultad, portabilidad

NOTA: la forma de redactar en las transparencias es... unica, hare lo que pueda por dejarlo interpretable de forma mas comoda

Las colecciones de benchmarks permiten medir el rendimiento de los procesadores con una variedad de aplicaciones, en este caso estan formadas por programas que pueden ser nucleos, pero fundamentalmente son programas realas. La ventaja clave es la que la debilidad de un benchmark es minimizada por la presencia de otro tipo de benchmark en la coleccion. Un ejemplo de colleciones es el SPEC

2.2.3. Formulación de resultados

Basicamente, nos quedaremos con los resumenes del rendimiento. Si tenemos varios programas corriendo en un computador si sumamos el tiempo total y lo dividimos por el total de programas, obtenemos el tiempo medio:

$$\frac{1}{m}\sum_{i=1}^{m} t_{ei} = t_{e\ \text{medio}}$$

con m siendo el numero de programas realizados y t_{ei} el tiempo de ejecucion para el programa i.

Es mas, podemos ponderar, asignando a cada programa un factor de peso, indicando la frecuencia relativa del programa en la carga del trabajo, tal que

$$\sum_{i=1}^{m} w_i \text{ x } t_{ei} = t_{e \text{ medio}}$$

con w_i siendo la frecuenda del programa i de la carga de trabajo.

Otra opcion seria la media geometrica, siendo

$$\sqrt[m]{\prod_{i=1}^{m} t_{ie}}$$

y para comapra con otro valor, si denotamos a la funcion con MG

$$\frac{MG(x_i)}{MG(y_i)} = MG\left(\frac{x_i}{y_i}\right)$$

Ejemplo de ejercicios para parcial 1, T1 y 2

3. TEMA3

texto

3.1. Apartado1

texto

subapartado1

texto

subapartado2

texto

subapartado3

texto

subapartado4

texto

3.2. Apartado2

texto

subapartado1

texto

subapartado2

texto

subapartado3

texto

subapartado4

texto

3.3. Apartado3

texto

subapartado1

texto

subapartado2

texto

subapartado3

texto

subapartado4

texto

3.4. Apartado4

texto

subapartado1

texto

subapartado2

texto

subapartado3

texto

subapartado4

texto

4. Anexo

4.1. Formulario

añadir formulas

Referencias

[1] *X86 Assembly/Floating Point*