



Tema 1. Introducción

Arquitectura de los Computadores

Tema1. Introducción

Arquitectura

Diseño

Introducción

- **1. Arquitectura de computadores**
- **2. El diseño de computadores**

Tema1. Introducción

Arquitectura

Diseño

Introducción

1.Arquitectura de computadores

- 1.1 Niveles de descripción de un computador
- 1.2 Definición de arquitectura
- 1.3 Clasificación de arquitecturas

2.El diseño de computadores

- 2.1 El proceso de diseño
 - a)Establecer requerimientos funcionales
 - b)Decisiones de implementación
 - c)Consideración de las nuevas tendencias
- 2.2 Principios de diseño
 - a) Acelerar el caso común
 - b) Ley de rendimientos decrecientes
 - c) Localidad de referencia

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

El concepto de arquitectura

- ❖ **Acuñado por IBM en 1964.** Parte del repertorio de instrucciones visible por el programador:

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

- ❖ **Unificar** las diferentes **divisiones de IBM** en una **arquitectura común**
- ❖ **Referencia exclusivamente** al diseño del **repertorio de instrucciones**
- ❖ **Errores de diseño** por falta de consideración de aspectos de implementación

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

El concepto de arquitectura

- ❖ Disciplina que trata el **diseño** de máquinas para ejecutar programas con criterios de optimización de rendimiento y coste

❖ Aspectos relacionados

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

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

Niveles estructurales de Bell y Newell [Bell, 71]:

- Descripción del computador mediante una aproximación por capas
- Cada capa utiliza los servicios que proporciona la del nivel inferior
- **Propone 5 niveles:**
 - De componente
 - Electrónico
 - Digital
 - Transferencia entre registros (RT)
 - Procesador-Memoria-Interconexión (PMS)

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

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
- **Se distinguen 5 niveles:**
 - Aplicaciones
 - Lenguajes de alto nivel
 - Sistema Operativo
 - Instrucciones máquina
 - Microinstrucciones
- Estos niveles son similares a los niveles funcionales de [Tanenbaum 86, 99, 00].

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

Niveles de abstracción para un computador:

Integra la orientación estructural de los niveles de Bell y Newell y el punto de vista funcional de los niveles de Levy y Tanenbaum.

SW

ARQUITECTURA

TECNOLOGÍA

HW

Niv. Soft. Superiores

Llamadas al Sistema, Comandos,...

Sistema Operativo

Sistemas de cómputo, Ensamblador,...

Sist. Computador

Procesadores, Interfaces E/S, Datapath

Nivel RT

ALUs, registros, memorias, MUX,...

Digital

Puertas lógicas, inversores, biestables

Electrónica

Uniones P N, Metal, Polisilicio

Componente

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

Niveles de abstracción para un computador:

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.

Componentes físicos:

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

SW

ARQUITECTURA

TECNOLOGÍA

HW

Niv. Soft. Superiores

Llamadas al Sistema, Comandos,...

Sistema Operativo

Sistemas de cómputo, Ensamblador,...

Sist. Computador

Procesadores, Interfaces E/S, Datapath

Nivel RT

ALUs, registros, memorias, MUX,...

Digital

Puertas lógicas, inversores, biestables

Electrónica

Uniones P N, Metal, Polisilicio

Componente

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

Niveles de abstracción para un computador:

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 componentes 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.

SW

ARQUITECTURA

TECNOLOGÍA

HW

Niv. Soft. Superiores

Llamadas al Sistema, Comandos,...

Sistema Operativo

Sistemas de cómputo, Ensamblador,...

Sist. Computador

Procesadores, Interfaces E/S, Datapath

Nivel RT

ALUs, registros, memorias, MUX,...

Digital

Puertas lógicas, inversores, biestables

Electrónica

Uniones P N, Metal, Polisilicio

Componente

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

Niveles de abstracción para un computador:

Transferencia entre registros (RT)

- 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.

SW

ARQUITECTURA

TECNOLOGÍA

HW

Niv. Soft. Superiores

Llamadas al Sistema, Comandos,...

Sistema Operativo

Sistemas de cómputo, Ensamblador,...

Sist. Computador

Procesadores, Interfaces E/S, Datapath

Nivel RT

ALUs, registros, memorias, MUX,...

Digital

Puertas lógicas, inversores, biestables

Electrónica

Uniones P N, Metal, Polisilicio

Componente

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

Niveles de abstracción para un computador:

Sistema computador

- Especificación de componentes (memorias, procesador, buses, redes de interconexión, periféricos, etc). interconexión entre ellos y operación del sistema completo.
- Programación a bajo nivel (lenguaje máquina y ensamblador).

SW

ARQUITECTURA

TECNOLOGÍA

HW

Niv. Soft. Superiores

Llamadas al Sistema, Comandos,...

Sistema Operativo

Sistemas de cómputo, Ensamblador,...

Sist. Computador

Procesadores, Interfaces E/S, Datapath

Nivel RT

ALUs, registros, memorias, MUX,...

Digital

Puertas lógicas, inversores, biestables

Electrónica

Uniones P N, Metal, Polisilicio

Componente

1.1 Niveles de descripción de un computador

Arquitectura

Diseño

Introducción

Niveles de abstracción para un computador:

Niveles 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 → incremento de prestaciones.

Sistema operativo

- 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.

SW

ARQUITECTURA

TECNOLOGÍA

HW

Niv. Soft. Superiores

Llamadas al Sistema, Comandos,...

Sistema Operativo

Sistemas de cómputo, Ensamblador,...

Sist. Computador

Procesadores, Interfaces E/S, Datapath

Nivel RT

ALUs, registros, memorias, MUX,...

Digital

Puertas lógicas, inversores, biestables

Electrónica

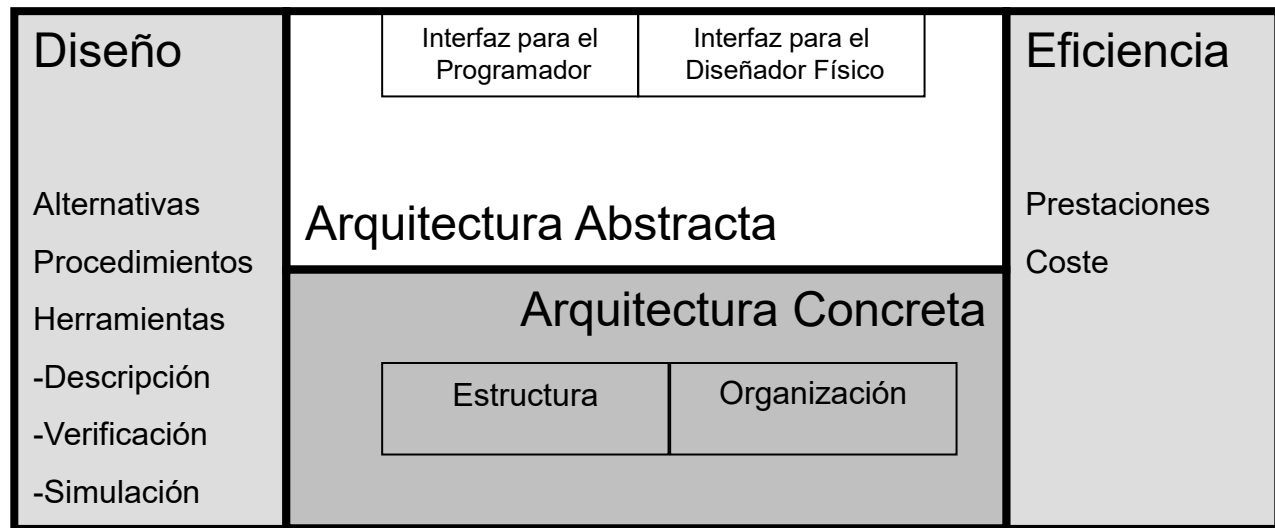
Uniones P N, Metal, Polisilicio

Componente

1.1 Niveles de descripción de un computador

Arquitectura

Diseño



Introducción

1.2 Definición de arquitectura

Arquitectura

Diseño

Introducción

Niveles que abarca la Arquitectura

The diagram illustrates the levels of computer architecture, showing the relationship between the abstract system level and the concrete microarchitecture level.

Arquitectura abstracta a nivel SC (System Level):

- Computador:** The overall system, represented by a large box.
- Procesador:** The processor, represented by a circle labeled 'P'.
- Cache (C):** The cache, represented by a green rectangle.
- Adaptador de Bus (AB):** The bus adapter, represented by a light blue rectangle.
- Memoria (M):** The memory, represented by a pink rectangle.

Arquitectura concreta a nivel SC (Nivel PMS):

- Cache (C):** The cache, represented by a green rectangle.

Arquitectura abstracta de procesador:

- P:** The processor, represented by a circle.

Arquitectura concreta de procesador o Microarquitectura:

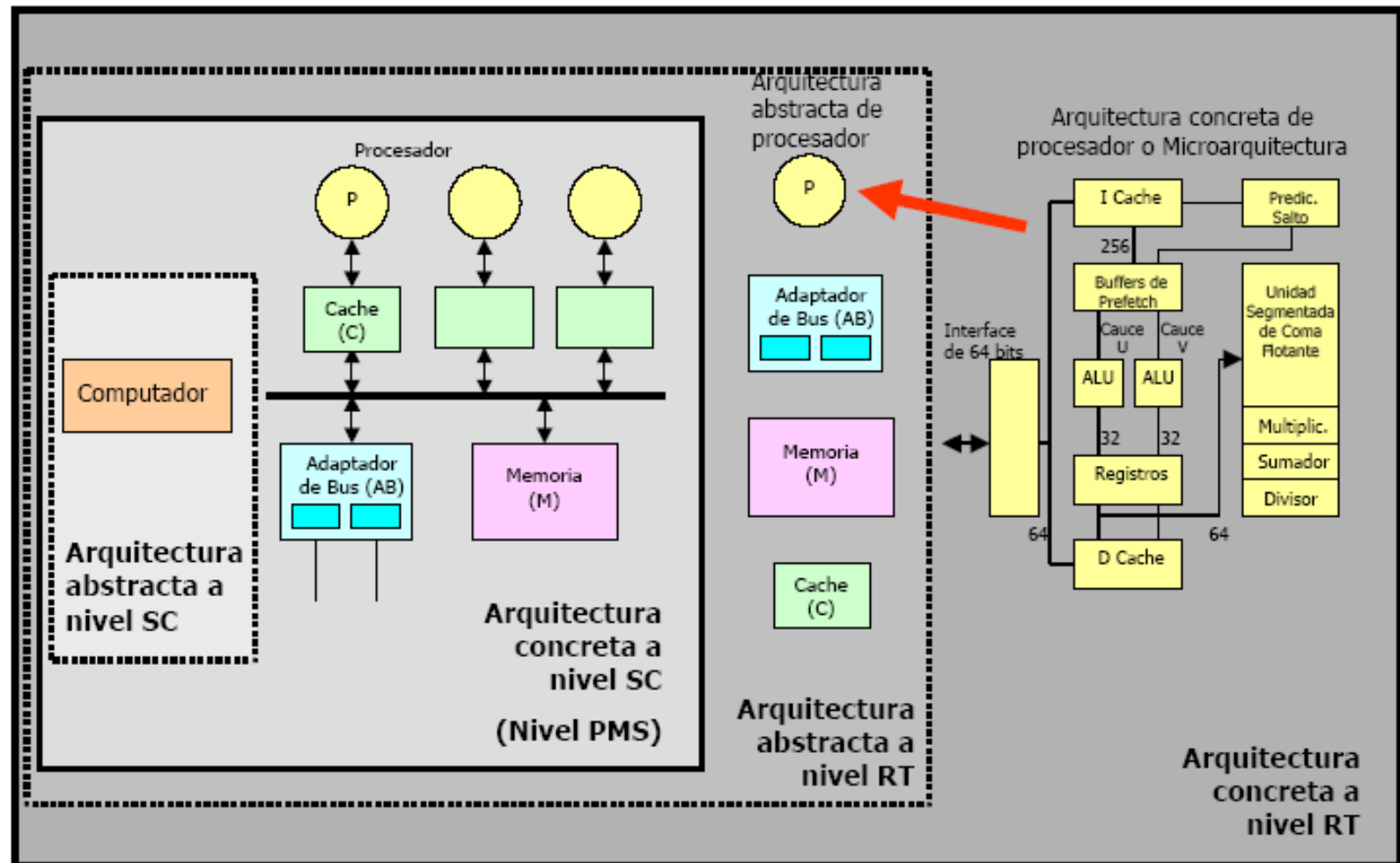
- I Cache:** Instruction Cache.
- Predic. Salto:** Branch Predictor.
- 256:** Branch Target Cache size.
- Buffers de Prefetch:** Prefetch buffers.
- Cauce U:** Instruction stream.
- Cauce V:** Data stream.
- ALU:** Arithmetic Logic Unit.
- 32:** Register file size.
- Registros:** Register file.
- D Cache:** Data Cache.
- Unidad Segmentada de Coma Flotante:** Floating-point unit.
- Multiplic.:** Multiplier.
- Sumador:** Adder.
- Divisor:** Divider.

Arquitectura abstracta a nivel RT (Register Transfer Level):

- Cache (C):** The cache, represented by a green rectangle.

Arquitectura concreta a nivel RT:

- Cache (C):** The cache, represented by a green rectangle.



1.2 Definición de arquitectura

Arquitectura

Diseño

- ❖ **Definición de arquitectura** “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” [Ortega, 2005]

Introducción

1.2 Definición de arquitectura

Arquitectura

Diseño

- ❖ **Definición de microarquitectura:** “Conjunto de recursos y métodos utilizados para satisfacer las especificaciones que establece la arquitectura. El término incluye tanto la forma en que se organizan los recursos como las técnicas utilizadas para alcanzar los objetivos de costes y prestaciones planteados. La microarquitectura define las especificaciones para la implementación lógica” [Ortega, 2005]

Introducción

1.2 Definición de arquitectura

Arquitectura

Diseño

Introducción

Ámbito de la arquitectura

Arquitectura a nivel lenguaje máquina

Repertorio de
instrucciones

Organización

alto nivel del diseño de un
computador

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

Hardware

componentes específicos de
una máquina

Diseño lógico
detallado, la
tecnología de
encapsulamiento...

Implementación

1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Introducción

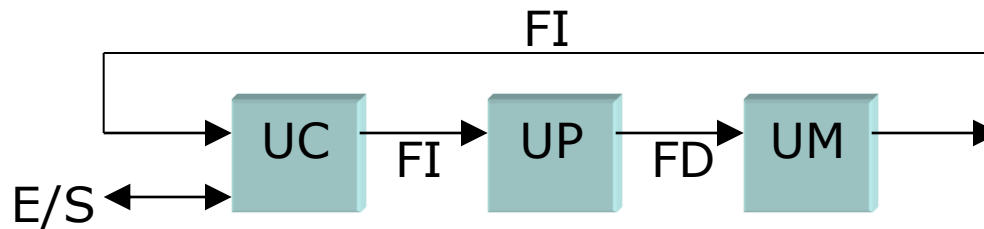
- Como toda clasificación, una clasificación (o taxonomía) de arquitecturas persigue dividir el conjunto de los computadores en una serie de clases de forma que, si se sabe la clase a la que pertenece un computador, automáticamente se conocen una serie de características interesantes del mismo
- La clasificación más extendida "**Taxonomía de Flynn**":
 - SISD
 - SIMD
 - MISD
 - MIMD

1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Computadores SISD: 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;
```

Introducción

1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Computadores SIMD: un único flujo de instrucciones procesa operandos y genera resultados, definiendo varios flujos de datos, dado 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;
```

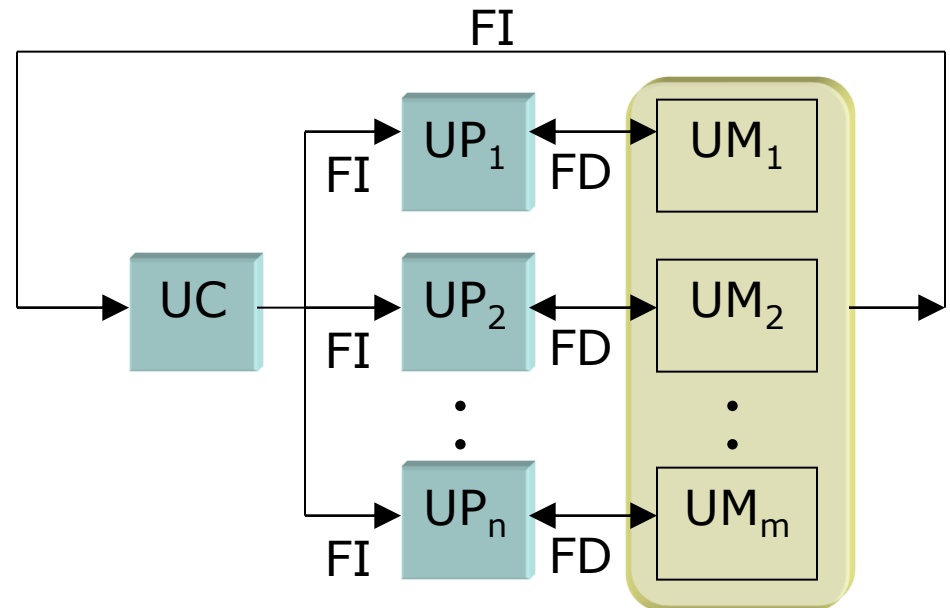
Procesadores matriciales

```
ADDV C,A,B
```

```
SUBV F,D,E
```

```
MULV G,K,H
```

Introducción



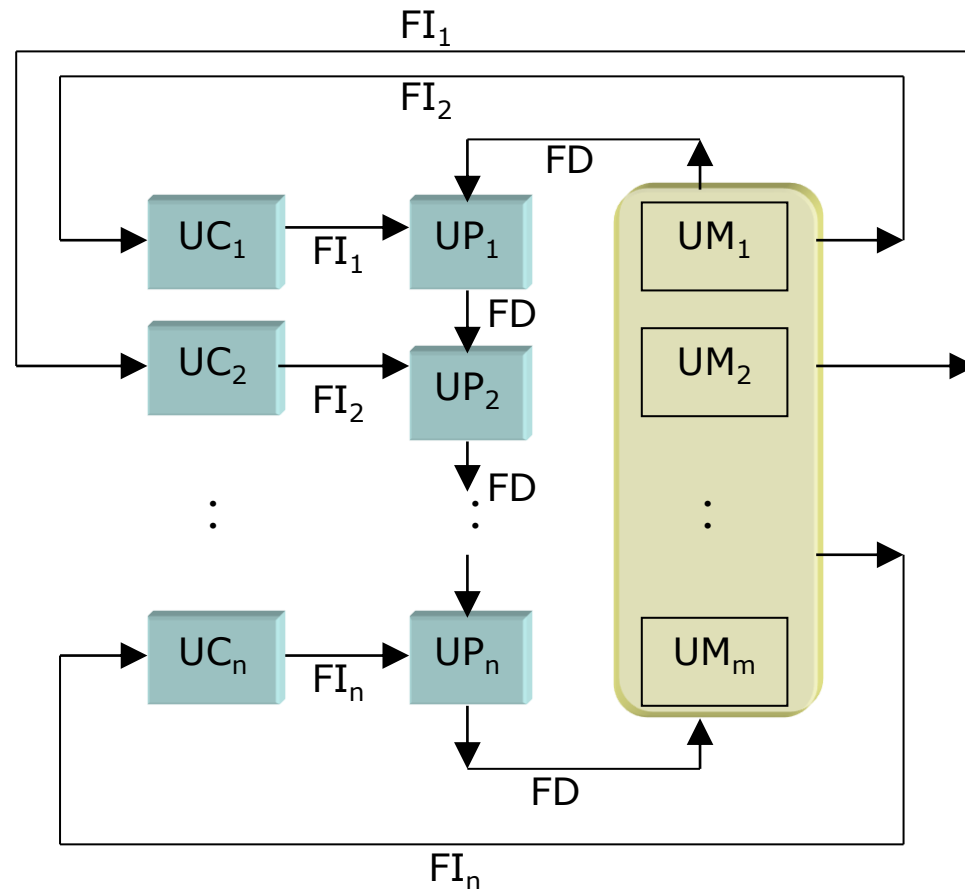
1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Introducción

Computadores MISD: 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



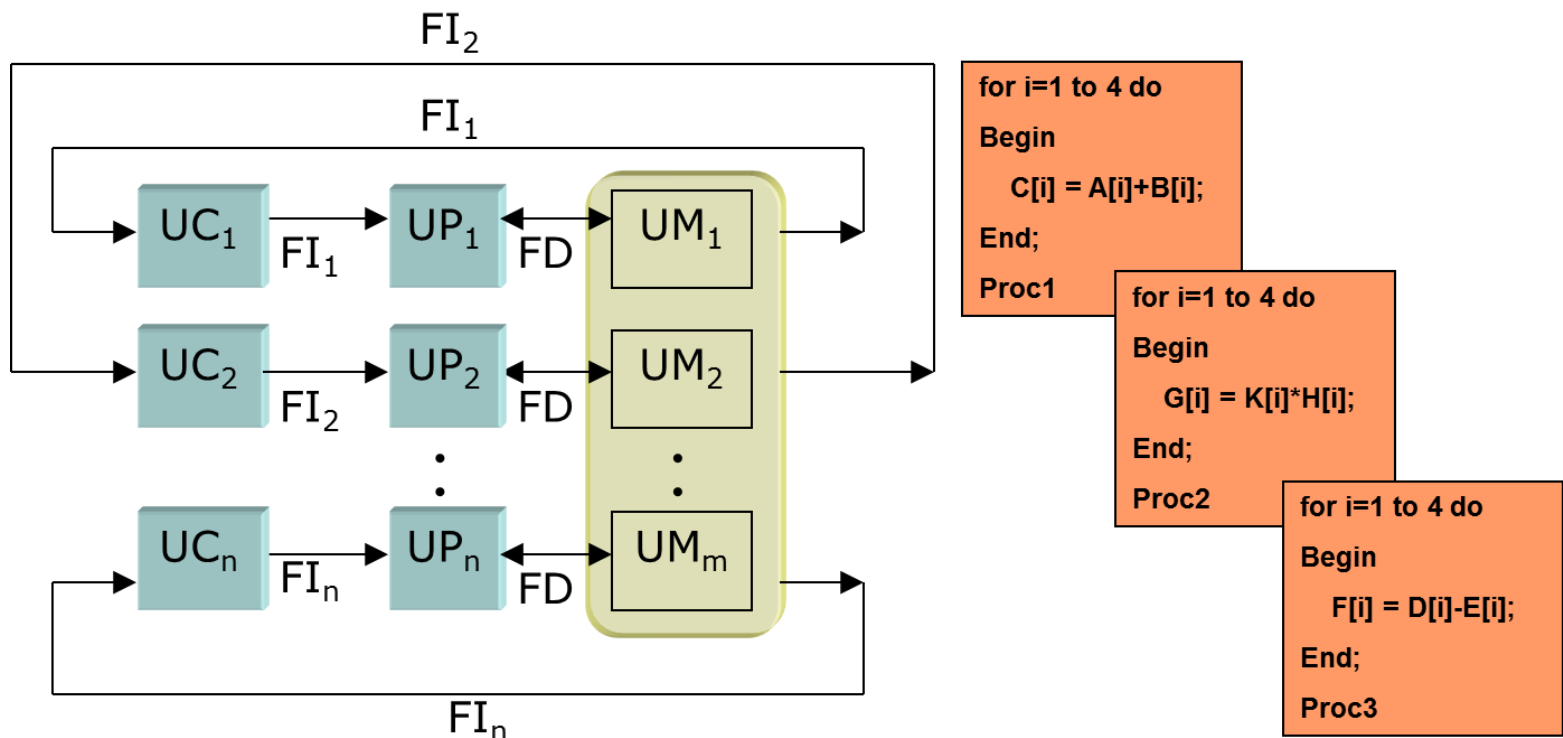
1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Introducción

Computadores MIMD: 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.



1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Introducción

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

1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Introducción

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

- **Nivel de instrucción (ILP)** – se ejecutan en paralelo las instrucciones de un programa. Granularidad fina.
- **Nivel de bucle o hebra (Thread)** – se ejecutan en paralelo 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 paralelo programas diferentes que pueden corresponder, o no, a una misma aplicación. Granularidad gruesa.

1.3 Clasificación de arquitecturas

Arquitectura

Diseño

Introducción

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, video compartido, juegos multiusuario
- **Embebidos**
 - Presentes en máquinas: microondas, lavadoras, impresoras, switches, coches ... Amplio espectro coste rendimiento

2. Diseño de computadores

Arquitectura

Diseño

Introducción

• Tarea de diseño

• Propuesta conjunta ACM-IEEE

- Informática basada en tres paradigmas: teoría, abstracción y diseño (no prima ninguno sobre el otro)

• Teoría: Fuerte base matemática. Ciencias formales

- Definición
- Teorema
- Demostración
- Interpretación

2. Diseño de computadores

Arquitectura

Diseño

Introducción

• Tarea de diseño

• Propuesta conjunta ACM-IEEE

• Informática basada en tres paradigmas: teoría, abstracción y diseño (no prima ninguno sobre el otro)

• **Abstracción: Ciencias experimentales. (Física, química)**

- Hipótesis
- Construcción de un modelo y realización de predicciones
- Diseño de experimentos y recogida de resultados
- Análisis de resultados

2. Diseño de computadores

Arquitectura

Diseño

Introducción

• Tarea de diseño

• Propuesta conjunta ACM-IEEE

- Informática basada en tres paradigmas: teoría, abstracción y diseño (no prima ninguno sobre el otro)

• Diseño: Ingenierías. Ciencias aplicadas

- Establecer requerimientos
- Especificar
- Realización del sistema
- Prueba del sistema

2.1 El proceso de diseño de computadores

Arquitectura

Diseño

- ❖ **Establecer requerimientos funcionales**
- ❖ **Especificar el sistema**
- ❖ **Realización del sistema**
- ❖ **Prueba del sistema**

Introducción

2.1 El proceso de diseño de computadores

Arquitectura

Diseño

a. Establecer requerimientos funcionales y especificar

- ❖ **Funcionalidades inspiradas por el mercado y el software de aplicación que determinan características específicas del sistema**
- ❖ **Especificación en base a criterios de coste, rendimiento, consumo y disponibilidad para el mercado pensado**

Introducción

2.1 El proceso de diseño de computadores

a. Establecer requerimientos funcionales y especificar

Arquitectura

Diseño

Introducción

Requerimientos funcionales	Características típicas requeridas o soportadas
Área de aplicación <ul style="list-style-type: none">• Móviles• PCs• Servidores• Clusters/Warehouse-Scale Computers• Computación embebida	Objetivo del computador <ul style="list-style-type: none">• 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.

2.1 El proceso de diseño de computadores

b. Decisiones de implementación

¿Como se implementa mejor una funcionalidad requerida?

La decisión de implementación software o hardware

⚙ **Ventajas implementación software**

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

⚙ **Ventajas implementación hardware**

- ⚙ Rendimiento

Arquitectura

Diseño

Introducción

2.1 El proceso de diseño de computadores

Arquitectura

Diseño

Introducción

c. Consideración de las nuevas tendencias

⚙ **Diseñador consciente de tendencias:**

- ⚙ Utilización del computador
- ⚙ Tecnología de computadores
- ⚙ Una arquitectura a nivel lenguaje máquina con éxito puede durar decenas de años (núcleo de la IBM 360 desde 1964, 50 años)

2.1 El proceso de diseño de computadores

Arquitectura

Diseño

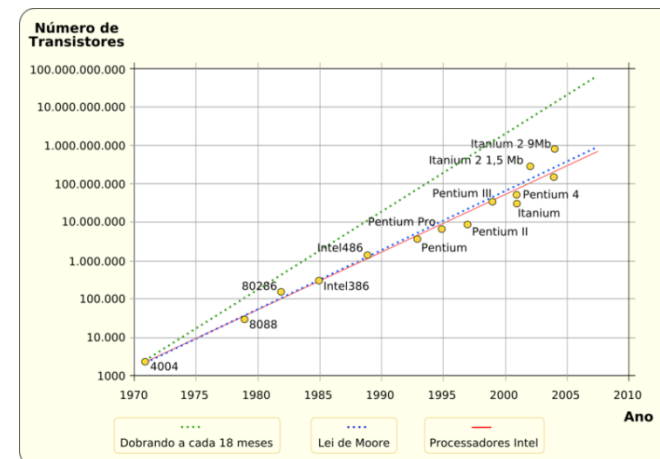
c. Consideración de las nuevas tendencias

◆ 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 cómputo y las densidades de almacenamiento se duplican cada 18 meses



Introducción

Arquitectura

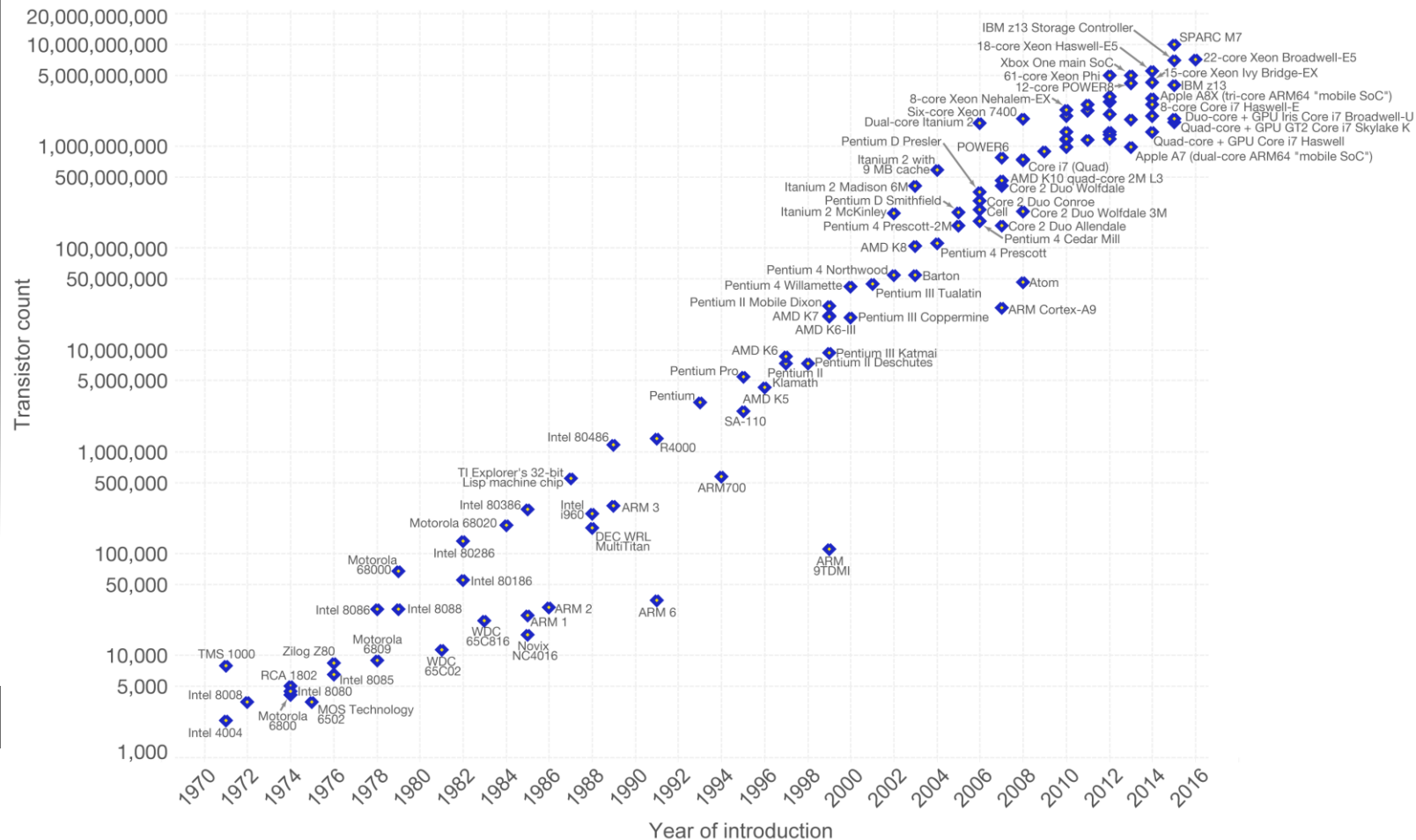
Diseño

Introducción

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World
in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

The data visualization is available at [OurWorldinData.org](https://ourworldindata.org). There you find more visualizations and research on this topic.

Licensed under [CC-BY-SA](#) by the author Max Roser.

2.1 El proceso de diseño de computadores

Arquitectura

Diseño

Introducción

c. Consideración de las nuevas tendencias

⚙️ **Tendencias software**

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

2.2 Principios de diseño de computadores

Arquitectura

Diseño

a. Acelerar el caso común

◆ Favorecer el caso frecuente

- ◆ **Ejemplo:** El desbordamiento de la suma es poco frecuentemente.
- ◆ El principio indicaría la optimización del caso sin desbordamiento
- ◆ La cuantificación de este principio se conoce como la ley de Amdahl

◆ 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 más rápido está limitada por la fracción de tiempo en que se puede utilizar ese modo más rápido

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

Introducción

2.2 Principios de diseño de computadores

Arquitectura

Diseño

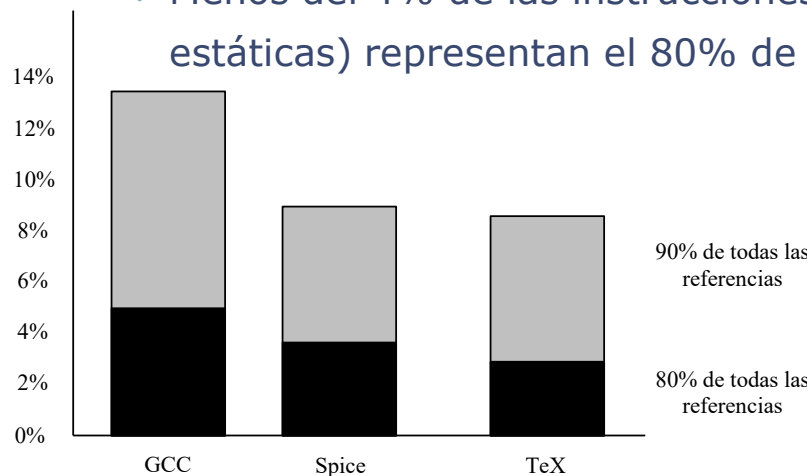
Introducción

b. Ley de rendimientos decrecientes

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

c. 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 Spice (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.



Tema 2

Análisis del rendimiento

Arquitectura de los Computadores

Tema 2. Análisis del rendimiento

🏠 Objetivos

- 🏠 Entender el concepto de rendimiento, la evolución del rendimiento en los computadores en los últimos años y su relación con el coste
- 🏠 Saber cuantificar la ganancia de rendimiento o aceleración que puede obtenerse al mejorar alguna característica de un computador
- 🏠 Mostrar al alumno distintas métricas para evaluar el rendimiento de una arquitectura, observando la relación que existe entre ellas.
- 🏠 Adquirir conciencia de la necesidad de establecer métricas para llevar a cabo procesos de evaluación y comparación objetiva y contrastada de sistemas computacionales

Tema 2. Análisis del rendimiento

🏠 Contenido

🏠 2.1. Rendimiento. Concepto y definiciones

- 🏠 Concepto de rendimiento
- 🏠 Ley de Amdhal
- 🏠 Relación entre rendimiento y coste

🏠 2.2. Evaluación del rendimiento

- 🏠 Medidas del rendimiento
- 🏠 Programas para evaluar el rendimiento
- 🏠 Formulación de resultados

2.1. Rendimiento. Concepto y definiciones

Tema 2 Análisis del rendimiento

Arquitectura de los Computadores

2.1 Rendimiento. Concepto y definiciones

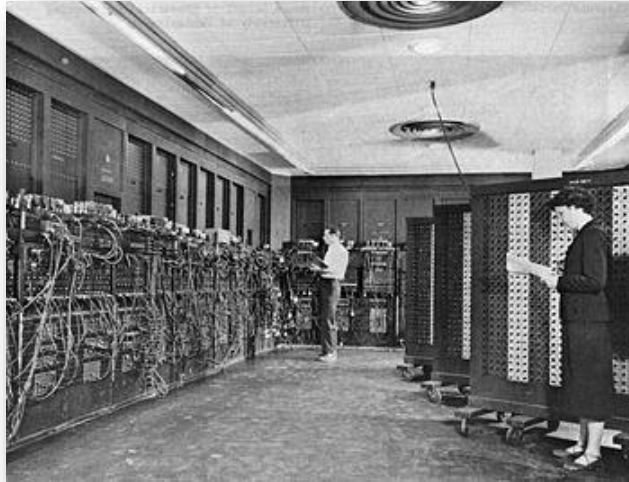
Concepto

Amdahl

Relación

Análisis de
rendimiento

Evolución del rendimiento



- La tecnología de computadores ha tenido un increíble progreso en los últimos 70 años

- Por menos de 500€ es posible comprar un portátil que tiene más rendimiento, memoria y capacidad de disco que un computador que costaba 50 millones de € en 1993.

- Esta rápida mejora se debe fundamentalmente a:

- Avances en la tecnología** (casi constante) usada para construir computadores

- Tamaño de los elementos en el chip (feature size), velocidad del reloj

- Innovaciones en el diseño** (menos consistentes)

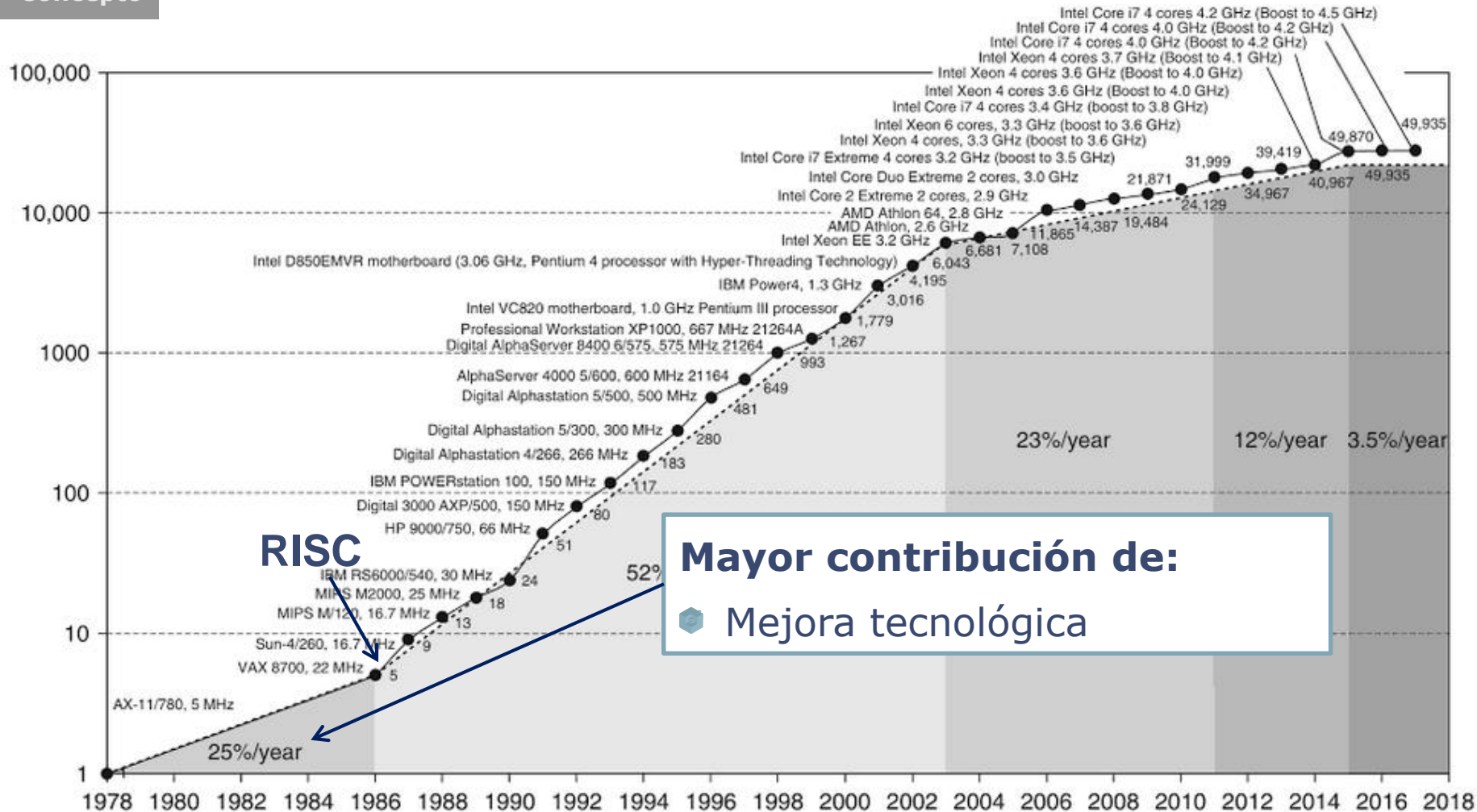
- Compiladores de lenguajes de alto nivel, UNIX

- Provocado por arquitecturas RISC

2.1 Rendimiento. Concepto y definiciones

Concepto

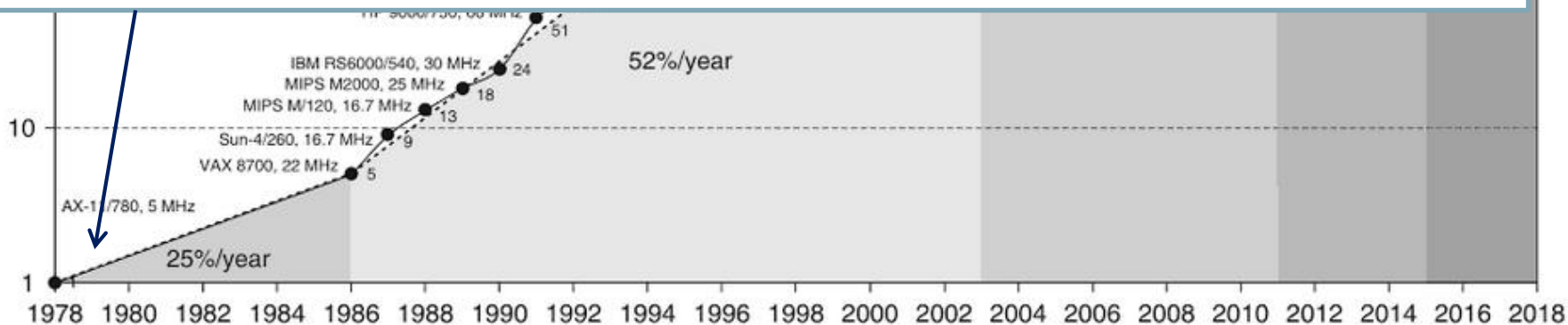
Evolución del rendimiento



Aparición del microprocesador (μ P) (finales 1970)

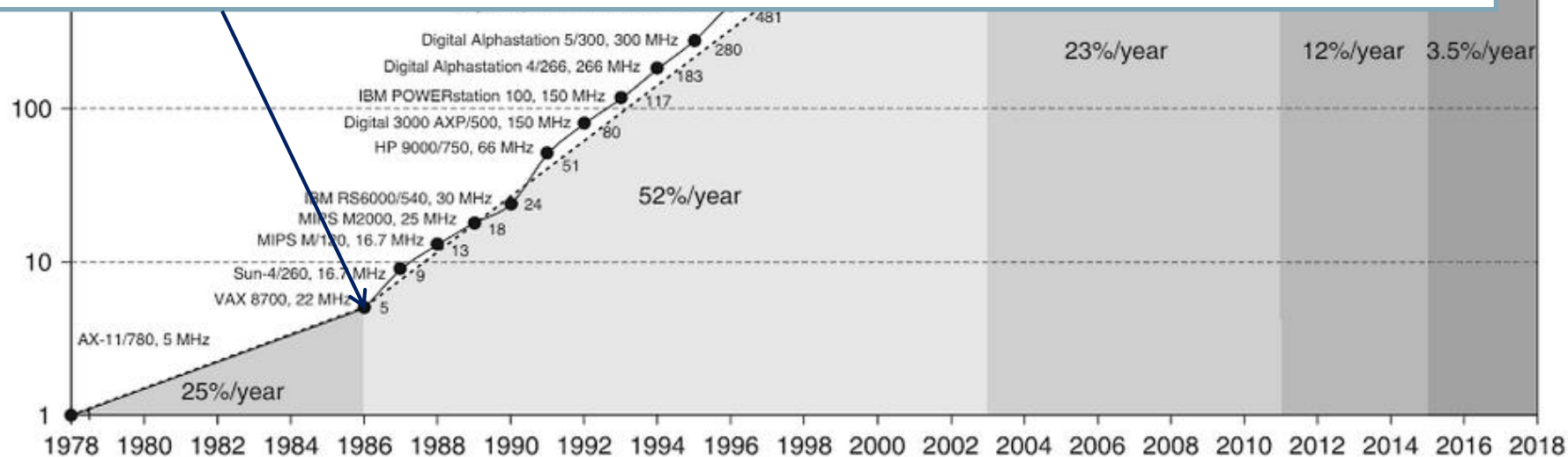
- Capacidad de dirigir los avances en la tecnología de circuitos integrados
- Tasa más alta de mejora del rendimiento (35% anual)
- Ventajas en el coste debido a la producción masiva de μ Ps
- Aumenta el número de computadores basados en μ Ps.
- Además, dos cambios significantes:
 - Eliminación virtual de la programación en lenguaje ensamblador
 - Reduce la necesidad de la compatibilidad en el código objeto
 - Aparición de sistemas operativos estandarizados como UNIX
 - Reducen el coste y el riesgo en la aparición de una nueva arquitectura

Performance (vs. VAX-11/780)



RISC (Principios 1980)

- Cambios anteriores permiten el desarrollo de forma satisfactoria de un nuevo conjunto de arquitecturas con instrucciones más simples: arquitecturas RISC (Reduced Instruction Set Computer).
- Los diseñadores de máquinas RISC se centraron en dos técnicas clave para la mejora del rendimiento:
 - La explotación del paralelismo a nivel de instrucción
 - El uso de cachés
- El aumento del rendimiento forzó a las arquitecturas previas a mantener el ritmo o a desaparecer



2.1 Rendimiento. Concepto y definiciones

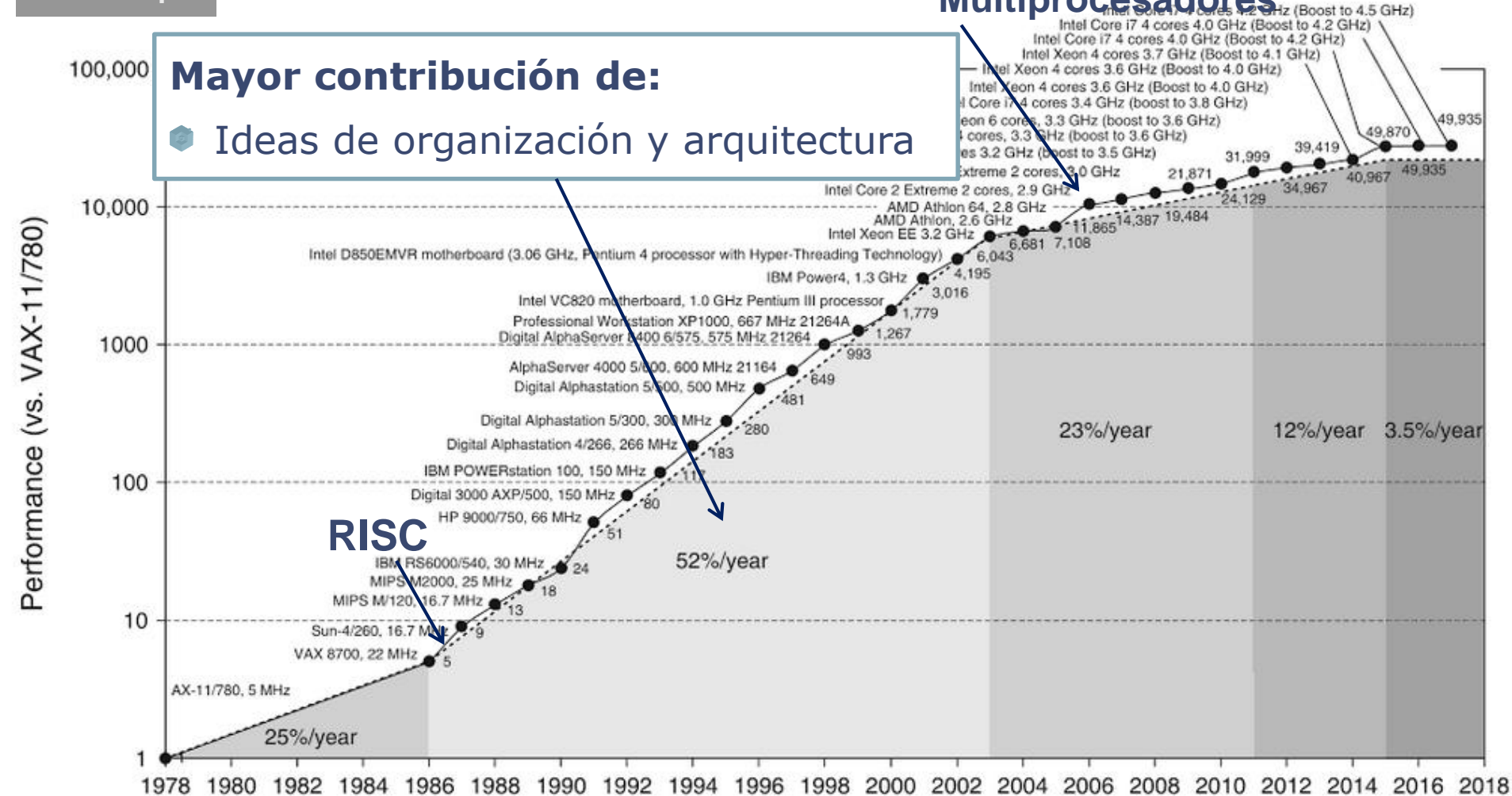
Concepto

Evolución del rendimiento

Mayor contribución de:

➡ Ideas de organización y arquitectura

Multiprocesadores



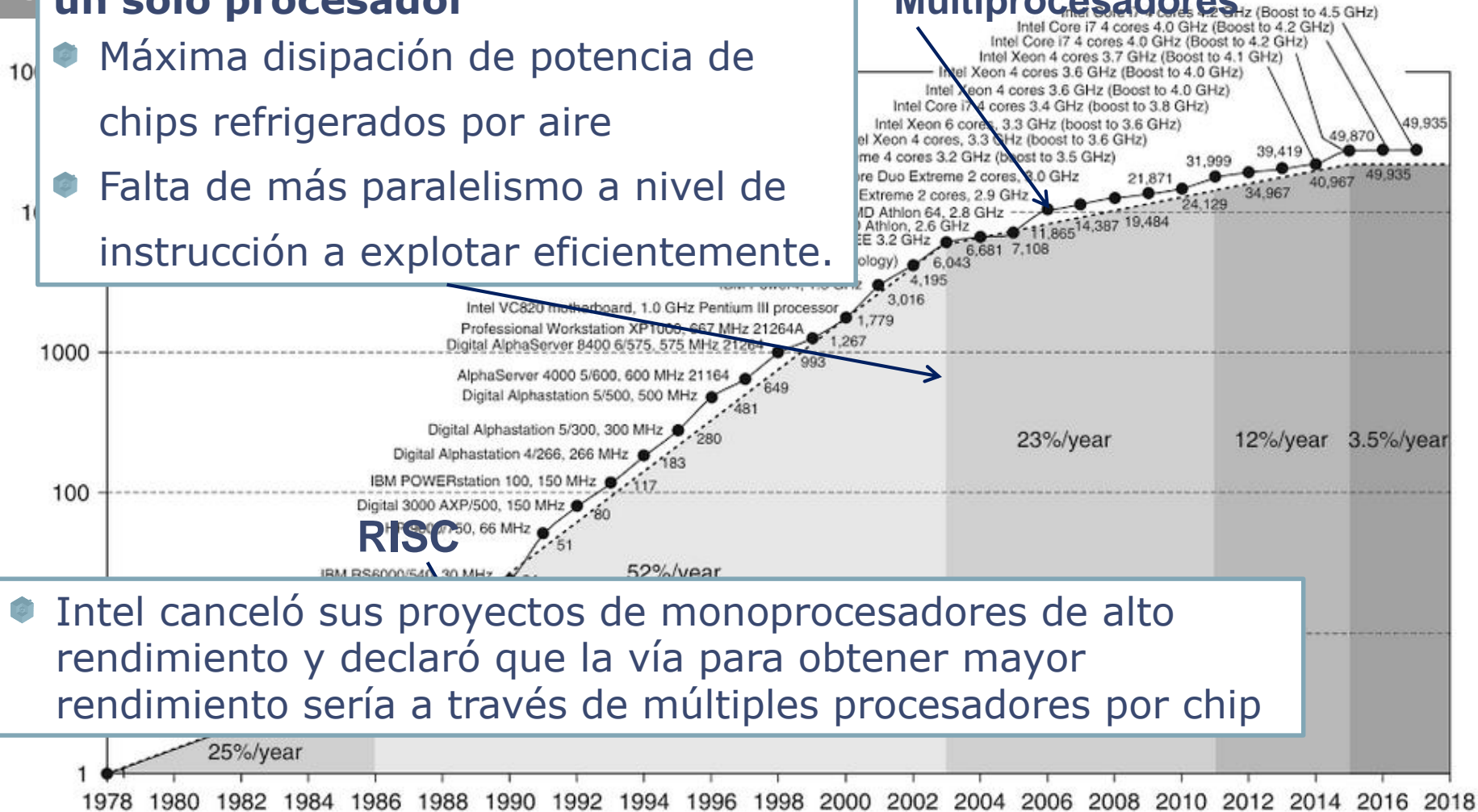
2.1 Rendimiento. Concepto y definiciones

Cae la mejora del rendimiento para un sólo procesador

- Máxima disipación de potencia de chips refrigerados por aire
- Falta de más paralelismo a nivel de instrucción a explotar eficientemente.

Multiprocesadores

Performance (vs. VAX-11/780)



- Intel canceló sus proyectos de monoprocesadores de alto rendimiento y declaró que la vía para obtener mayor rendimiento sería a través de múltiples procesadores por chip

2.1 Rendimiento. Concepto y definiciones

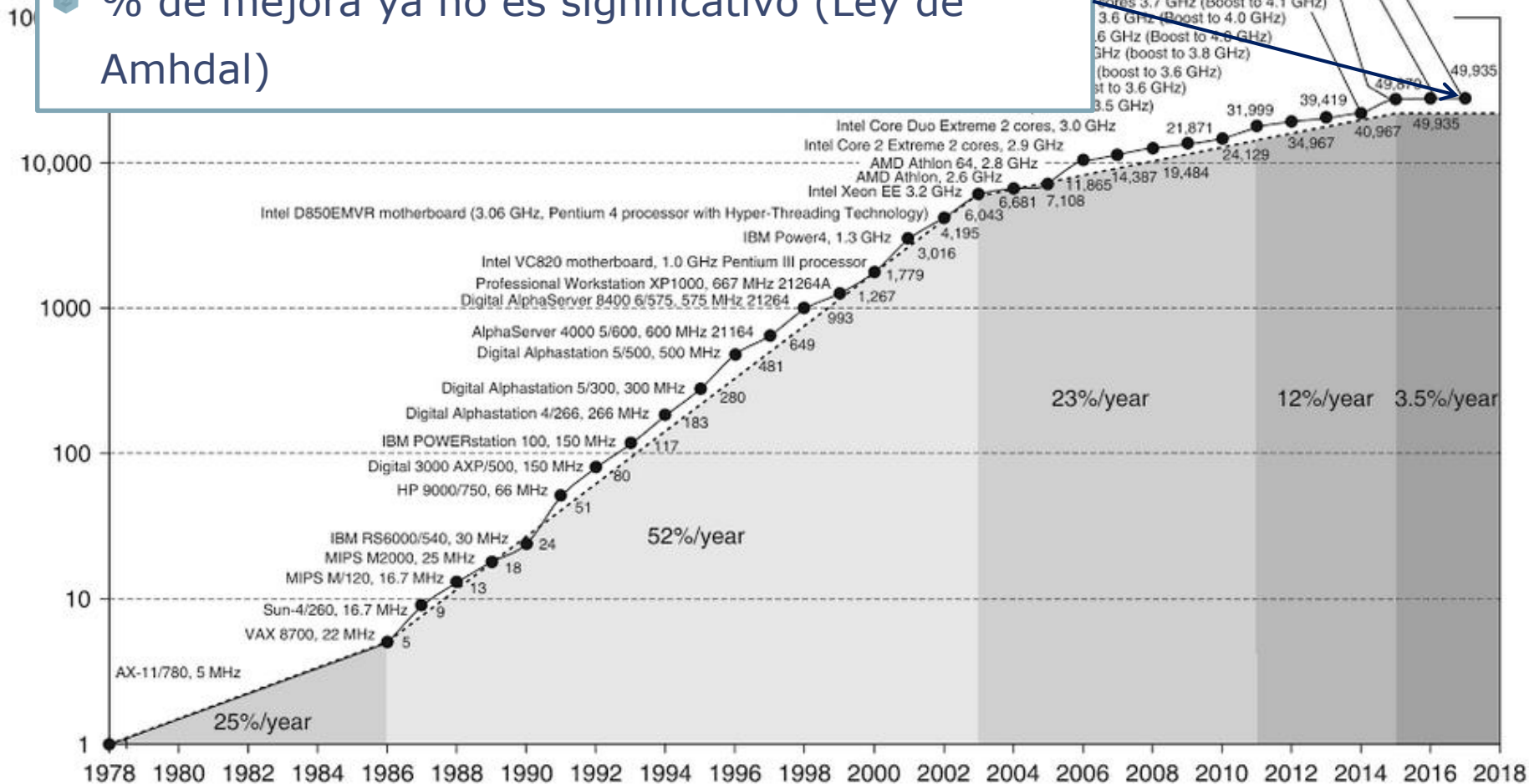
El rendimiento se estanca para un solo procesador

🏠 % de mejora ya no es significativo (Ley de Amhdal)

multiprocesadores

Intel Core i7 4 cores 4.2 GHz (Boost to 4.5 GHz)
Core i7 4 cores 4.0 GHz (Boost to 4.2 GHz)
Core i7 4 cores 3.7 GHz (Boost to 4.1 GHz)
Core i7 4 cores 3.6 GHz (Boost to 4.0 GHz)
Core i7 4 cores 3.6 GHz (Boost to 4.0 GHz)
Core i7 4 cores 3.5 GHz (Boost to 3.8 GHz)
Core i7 4 cores 3.5 GHz (Boost to 3.6 GHz)
Core i7 4 cores 3.5 GHz (Boost to 3.6 GHz)

Performance (vs. VAX-11/780)



2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Concepto de rendimiento

- Qué significa cuando decimos que un computador es más rápido que otro?:
 - **Usuario:**
 - Un computador es más rápido cuando un programa se ejecuta en **menos tiempo**
 - Interesado en reducir el **tiempo de respuesta/tiempo de ejecución**
 - Tiempo transcurrido entre el inicio y el final de un evento
 - **Administrador de un Cluster:**
 - Un computador es más rápido cuando completa **más transacciones por hora** (Google, Amazon...)
 - Interesado en aumentar la **productividad/throughput**
 - Cantidad total de trabajo realizado en un tiempo determinado

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Ejemplo:

- ¿Las siguientes mejoras en el rendimiento afectan a la productividad, al tiempo de respuesta o a ambas cosas?
 - 1. Ciclo de reloj más rápido
 - 2. Múltiples procesadores para tareas separadas (sistema de reservas de una compañía aérea)
 - 3. Procesamiento paralelo de problemas científicos
 - **La disminución del tiempo de respuesta habitualmente mejora la productividad.**
 - 1 y 3 mejoran el tiempo de respuesta y en consecuencia la productividad. En el caso 2 no mejora el tiempo de respuesta pero si la productividad.
- La influencia de factores no determinísticos aconseja hablar de estas medidas de rendimiento con **distribuciones de probabilidad.**
 - Por ejemplo el tiempo de respuesta en un disco para una operación de entrada salida depende de:
 - Actividad del disco en el instante de petición.
 - Número de tareas intentando acceder al disco
- Esta situación aconseja hablar de tiempo medio de respuesta de un acceso al disco

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Concepto de rendimiento

- ⚙ El tiempo es la medida más fiable del rendimiento
- ⚙ El tiempo de ejecución de un programa se mide en segundos
- ⚙ La relación entre el tiempo y el rendimiento es inversa
- ⚙ El rendimiento se mide como una frecuencia de eventos por segundo

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

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Relación de rendimientos entre máquinas

"X es más rápida que Y"

$$t_{ex} < t_{ey}$$

"X es n % más rápida que Y"

$$t_{ex} \cdot n\% < t_{ey}$$

Porcentaje incremental

$$tiempo\ ejecución_X + \frac{n}{100} tiempo\ ejecución_X = tiempo\ ejecución_Y$$

Aceleración

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

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Relación de rendimientos entre máquinas

En términos de rendimiento

$$1 + \frac{n}{100} = \frac{\text{tiempo ejecución}_Y}{\text{tiempo ejecución}_X} = \frac{\frac{1}{\text{Re } n \text{ dim iento}_Y}}{\frac{1}{\text{Re } n \text{ dim iento}_X}} = \frac{\text{Re } n \text{ dim iento}_X}{\text{Re } n \text{ dim iento}_Y}$$

Despejando

$$\frac{n}{100} = \frac{\text{Re } n \text{ dim iento}_X}{\text{Re } n \text{ dim iento}_Y} - 1 = \frac{\text{Re } n \text{ dim iento}_X - \text{Re } n \text{ dim iento}_Y}{\text{Re } n \text{ dim iento}_Y}$$

$$n = 100 \frac{\text{Re } n \text{ dim iento}_X - \text{Re } n \text{ dim iento}_Y}{\text{Re } n \text{ dim iento}_Y}$$

Expresado con tiempos de ejecución

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

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Relación de rendimientos entre máquinas

- Ejemplo: Si la máquina A ejecuta un programa en 10 segundos y la máquina B ejecuta un programa en 30 segundos. ¿Cuál de las siguientes afirmaciones es correcta?
 - A es el 30% más rápida que B
 - A es el 200% más rápida que B

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Relación de rendimientos entre máquinas

❖ "La productividad de X es el 30 por 100 superior que la de Y"

$$P = nt/t; P_x = 1,3 * P_y$$

Ejemplo: Si la máquina A ejecuta un programa en 10 segundos y la máquina B ejecuta el mismo programa en 15 segundos. ¿En que porcentaje es la máquina A más rápida que la B?

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Relación de rendimientos entre máquinas

❖ "La productividad de X es el 30 por 100 superior que la de Y"

$$P = nt/t; P_x = 1,3 * P_y$$

Ejemplo: Si la máquina A ejecuta un programa en 10 segundos y la máquina B ejecuta el mismo programa en 15 segundos. ¿En que porcentaje es la máquina A más rápida que la B?

$$n = 100 \frac{\text{tiempo ejecución}_B - \text{tiempo ejecución}_A}{\text{tiempo ejecución}_A}$$

$$n = 100 \frac{15 - 10}{10} = 50$$

"A es el 50% más rápida que B"

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- ❖ **Ejercicio 1:** Se dan los tiempos de ejecución en segundos del benchmark linpack y de 10.000 iteraciones del benchmark Dhrystone en dos modelos VAX y un procesador actual:

Modelo	Año	Tiempo linpack	Tiempo Dhrystone
VAX- 11/780	1978	4,9	5,69
VAX 8600	1985	1,43	1,35
Intel Xeon 4 cores	2010	$2,24 \times 10^{-4}$	$2,74 \times 10^{-4}$

- ❖ **A) ¿Cuántas veces es más rápido el 8600 que el 780 utilizando Linpack? ¿Que ocurre cuando se utiliza Dhrystone?**
- ❖ B) ¿Cuántas veces es más rápido el 8550 que el 8600 utilizando Linpack? ¿Que ocurre cuando se utiliza Dhrystone?
- ❖ C) ¿Cuántas veces es más rápido el Xeon que el 780 utilizando Linpack? ¿Que ocurre cuando se utiliza Dhrystone?
- ❖ **D) ¿Cual es el crecimiento medio del rendimiento por año entre el 780 y el 8600 utilizando linpack? ¿Que ocurre cuando se utiliza Dhrystone?**
- ❖ E) ¿Cual es el crecimiento medio del rendimiento por año entre el 8600 y el 8550 utilizando linpack? ¿Que ocurre cuando se utiliza Dhrystone?
- ❖ F) ¿Cual es el crecimiento medio del rendimiento por año entre el 780 y el Xeon utilizando linpack? ¿Que ocurre cuando se utiliza Dhrystone?

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

■ a y d) Comparación de rendimientos para el VAX 780 y el VAX 8600 (Según Linpack)

$$te_{VAX780} = 4,9 \text{ s}$$
$$rend_{VAX780} = \frac{1}{te_{VAX780}} = 0,204$$

$$te_{VAX8600} = 1,43 \text{ s}$$
$$rend_{VAX8600} = \frac{1}{te_{VAX8600}} = 0,699$$

■ Porcentaje de incremento del rendimiento entre las dos arquitecturas:

$$porcentaje = n = 100 \frac{te_{VAX780} - te_{VAX8600}}{te_{VAX8600}} = 100 \frac{4,9 - 1,43}{1,43} = 243\%$$

$$porcentaje = n = 100 \frac{rend_{VAX8600} - rend_{VAX780}}{rend_{VAX780}} = 100 \frac{0,699 - 0,204}{0,204} = 243\%$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

● Aceleración del rendimiento entre las dos arquitecturas

$$aceleracion = \left(1 + \frac{n}{100}\right) = \frac{te_{VAX780}}{te_{VAX8600}} = \frac{4,9}{1,43} = 3,43 = \frac{rend_{VAX8600}}{rend_{VAX780}} = \frac{0,699}{0,204}$$

$$rend_{VAX8600} = 3,43 \cdot rend_{VAX780} \Rightarrow 0,699 = 3,43 \cdot 0,204$$

└──────────┘
7 años

● Los incrementos anuales se aplican cada año sobre el anterior

$$rend_{an} = \Delta_{anual} \cdot rend_{an-1} \rightarrow rend_{a1} = \Delta_{anual} \cdot rend_{a0}$$

$$rend_{a2} = \Delta_{anual} \cdot rend_{a1} = (\Delta_{anual})^2 rend_{a0}$$

$$rend_{an} = \Delta_{anual} \cdot rend_{an-1} = (\Delta_{anual})^n rend_{a0}$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

El incremento anual es:

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

Si consideramos $n=7$

$$te_{a0} = te_{VAX780}$$

$$rend_{a0} = rend_{VAX780}$$

$$te_{a7} = te_{VAX8600}$$

$$rend_{a7} = rend_{VAX8600}$$

$$\Delta_{anual} = \sqrt[n]{\frac{rend_{a7}}{rend_{a0}}} = \sqrt[n]{\frac{te_{a0}}{te_{a7}}} = \sqrt[7]{\frac{rend_{VAX8600}}{rend_{VAX780}}} = \sqrt[7]{\frac{te_{VAX780}}{te_{VAX8600}}} = \sqrt[7]{3,43} = 1,193$$

$$aceleración = \left(1 + \frac{n}{100}\right) = \Delta_{anual} \Leftrightarrow n = (\Delta_{anual} - 1) \cdot 100 = 19,3\%$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- ❖ **Ejercicio 2:** Hay dos equipos de diseño en dos compañías diferentes. La gestión de la compañía más pequeña y más agresiva pide un ciclo de diseño de dos años para sus productos. La gestión de la compañía más grande y menos agresiva apuesta por un ciclo de diseño de cuatro años. Supongamos que en el mercado de hoy día se demanda 1.5 veces el rendimiento de un Intel Xeon 4 cores según Linpack.
- ❖ ¿Cuales deberían ser los objetivos de rendimiento para cada producto, si las frecuencias de crecimiento necesarias son el 22% por año?
- ❖ Supongamos que las compañías acaban de empezar a utilizar chips de DRAM de 2048 Megabits. Dado que la capacidad por chip se ha incrementado entre un 25% y un 40% por año recientemente y casi doblándose de 2 a 3 años. Supongamos que las frecuencias de crecimiento serán 30% y se duplicará en 3 años ¿Que tamaños de DRAM hay que planificar para utilizar en estos proyectos?. Obsérvese que el crecimiento de las DRAM es discreto

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- a) Actualmente 1.5 veces mas rendimiento que el Intel Xeon (según Linpack)

$$1.5 = \frac{te_{Xeon}}{te_{actual}} \Rightarrow te_{actual} = \frac{te_{Xeon}}{1.5} = \frac{2.24 \cdot 10^{-4}}{1.5} = 1.49 \cdot 10^{-4} s$$

- El rendimiento crece un 22% anual

$$rend_{a1} = rend_{act} + 0.22 \cdot rend_{act} = 1.22 \cdot rend_{act} \Rightarrow 1.22 = \frac{rend_{a1}}{rend_{act}} = \frac{te_{act}}{te_{a1}}$$

- El primer año $te_{a1} = \frac{1}{1.22} \cdot te_{act} = 0.82 \cdot te_{act}$

- El segundo año $te_{a2} = \frac{1}{1.22} \cdot te_{a1} = 0.82 \cdot te_{a1} = (0.82)^2 \cdot te_{act}$

- El año **n** $te_{an} = \frac{1}{1.22} \cdot te_{an-1} = 0.82 \cdot te_{an-1} = (0.82)^n \cdot te_{act}$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- a) Actualmente 1.5 veces mas rendimiento que el Intel Xeon (según Linpack)

$$1.5 = \frac{te_{Xeon}}{te_{actual}} \Rightarrow te_{actual} = \frac{te_{Xeon}}{1.5} = \frac{2.24 \cdot 10^{-4}}{1.5} = 1.49 \cdot 10^{-4} s$$

- El rendimiento crece un 22% anual

$$rend_{a1} = rend_{act} + 0.22 \cdot rend_{act} = 1.22 \cdot rend_{act} \Rightarrow 1.22 = \frac{rend_{a1}}{rend_{act}} = \frac{te_{act}}{te_{a1}}$$

- Empresa A (2 años)

$$te_{a2} = (0.82)^2 \cdot te_{act} = (0.82)^2 \cdot 1.49 \cdot 10^{-4} = 1 \cdot 10^{-4} s = 100 \mu s$$

- Empresa B (4 años)

$$te_{a4} = (0.82)^4 \cdot te_{act} = (0.82)^4 \cdot 1.49 \cdot 10^{-4} = 6.74 \cdot 10^{-5} s = 67.4 \mu s$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- b) El tamaño de la memoria actual es 2048 Mb

- Tamaño de la memoria año 1

$$tm_{a1} = tm_{act} + 0.3 \cdot tm_{act} \Rightarrow tm_{a1} = 1.3 \cdot tm_{act}$$

- Tamaño de la memoria año 2

$$tm_{a2} = tm_{a1} + 0.3 \cdot tm_{a1} \Rightarrow tm_{a2} = 1.3 \cdot tm_{a1} = (1.3)^2 \cdot tm_{act}$$

- Tamaño de la memoria año n

$$tm_{an} = tm_{an-1} + 0.3 \cdot tm_{an-1} \Rightarrow tm_{an} = 1.3 \cdot tm_{an-1} = (1.3)^n \cdot tm_{act}$$

- Evolución continua

$$tm_{a2} = (1.3)^2 \cdot tm_{act} = (1.3)^2 \cdot 2048 = 3461,12Mb$$

$$tm_{a3} = (1.3)^3 \cdot tm_{act} = (1.3)^3 \cdot 2048 = 4449,45Mb$$

$$tm_{a4} = (1.3)^4 \cdot tm_{act} = (1.3)^4 \cdot 2048 = 5849,29Mb$$

$$tm_{a6} = (1.3)^6 \cdot tm_{act} = (1.3)^6 \cdot 2048 = 9885,30Mb$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- b) El tamaño de la memoria actual es 2048 Mb

- Tamaño de la memoria año 1

$$tm_{a1} = tm_{act} + 0.3 \cdot tm_{act} \Rightarrow tm_{a1} = 1.3 \cdot tm_{act}$$

- Tamaño de la memoria año 2

$$tm_{a2} = tm_{a1} + 0.3 \cdot tm_{a1} \Rightarrow tm_{a2} = 1.3 \cdot tm_{a1} = (1.3)^2 \cdot tm_{act}$$

- Tamaño de la memoria año n

$$tm_{an} = tm_{an-1} + 0.3 \cdot tm_{an-1} \Rightarrow tm_{an} = 1.3 \cdot tm_{an-1} = (1.3)^n \cdot tm_{act}$$

- Evolución discreta (x2 cada 3 años)

- Empresa A (salto superior más cercano 3 años x 2)

$$tm_{EmpA} = 2048Mb_{act} \cdot 2 = 4096Mb$$

- Empresa B (salto superior más cercano 6 años x 2)

$$tm_{EmpB} = 2048Mb_{act} \cdot 2 \cdot 2 = 8192Mb$$

2.1 Rendimiento. Concepto y definiciones

Ley de Amdahl

Concepto

Amdahl

Relación

- Relacionado con el principio de diseño de **favorecer el caso frecuente**
- **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 más rápido **está limitada por la fracción de tiempo en que se puede utilizar** ese modo más rápido

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

Análisis de
rendimiento

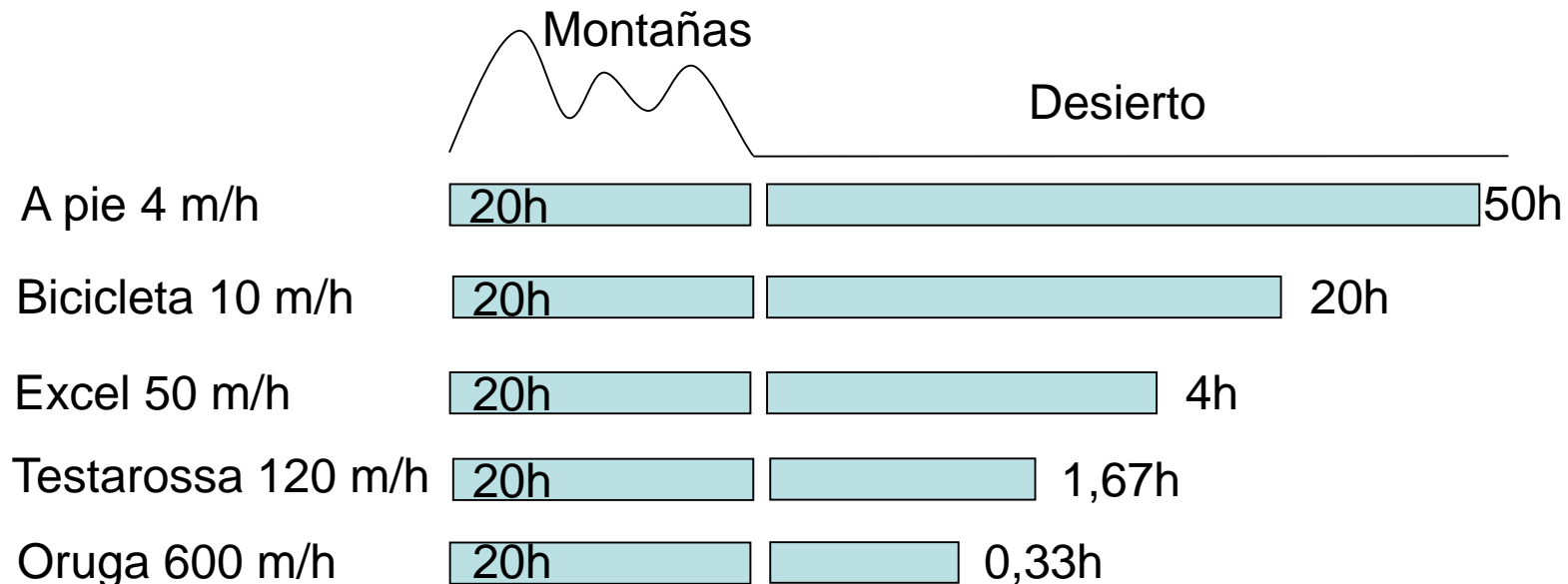
2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- ◆ **Ejemplo:** Se pretende viajar entre dos puntos cuyo trayecto involucra a dos tipos de terreno. El primer tramo es a través de las montañas y el segundo por el desierto. Tenemos varios tipos de vehículos pero las montañas tienen que ser recorridas necesariamente a pie, empleando para ello 20 horas. El segundo tramo de 200 millas puede ser recorrido de cinco formas diferentes:



Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

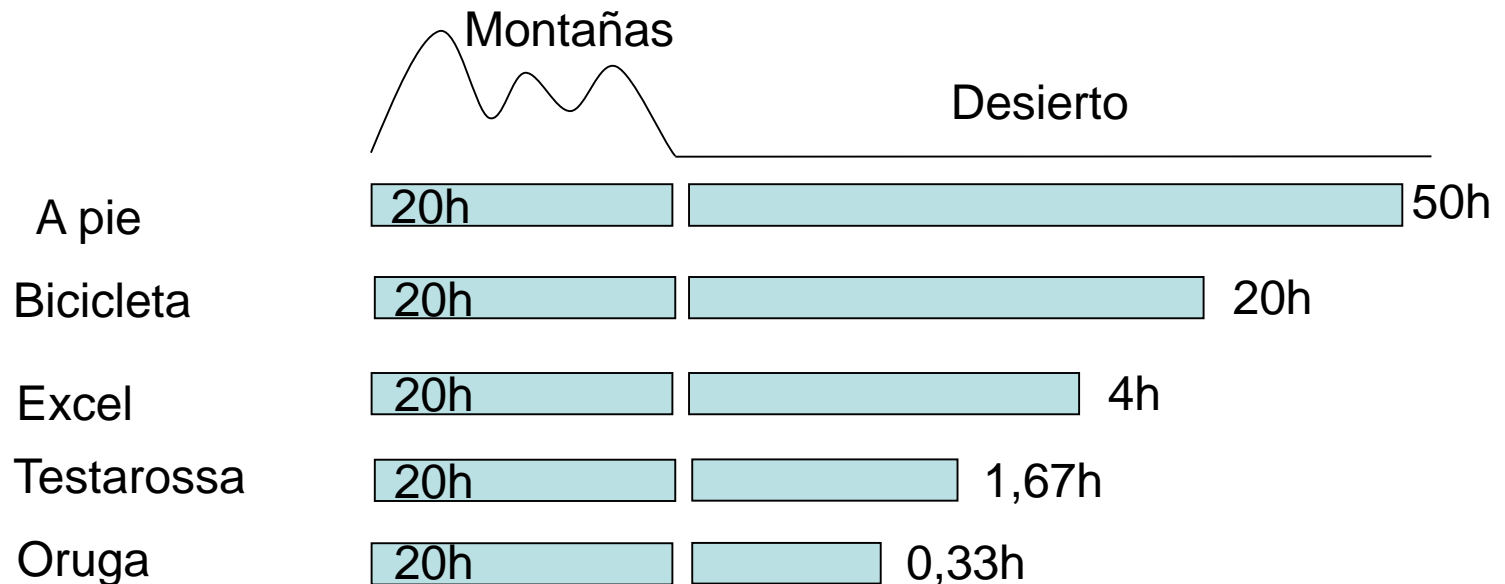
Concepto

Amdahl

Relación

Tomando como referencia el recorrido a pie de la distancia completa

Vehículo segunda parte del viaje	Horas de la segunda parte del viaje	Aceleración en el desierto	Horas del viaje completo	Aceleración en el viaje completo
A pie	50	$1=4/4=50/50$	$70=50+20$	$1,0=70/70$
Bicicleta	20	$2,5=10/4=50/20$	$40=20+20$	$1,8=70/40$
Excel	4	$12,5=50/4=50/4$	$24=20+4$	$2,9=70/24$
Testarossa	1,67	$30=120/4=50/1,67$	$21,67=20+1,67$	$3,2=70/21,67$
Vehículo oruga	0,33	$150=600/4=50/0,33$	$20,33=20+0,33$	$3,4=70/20,33$



Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

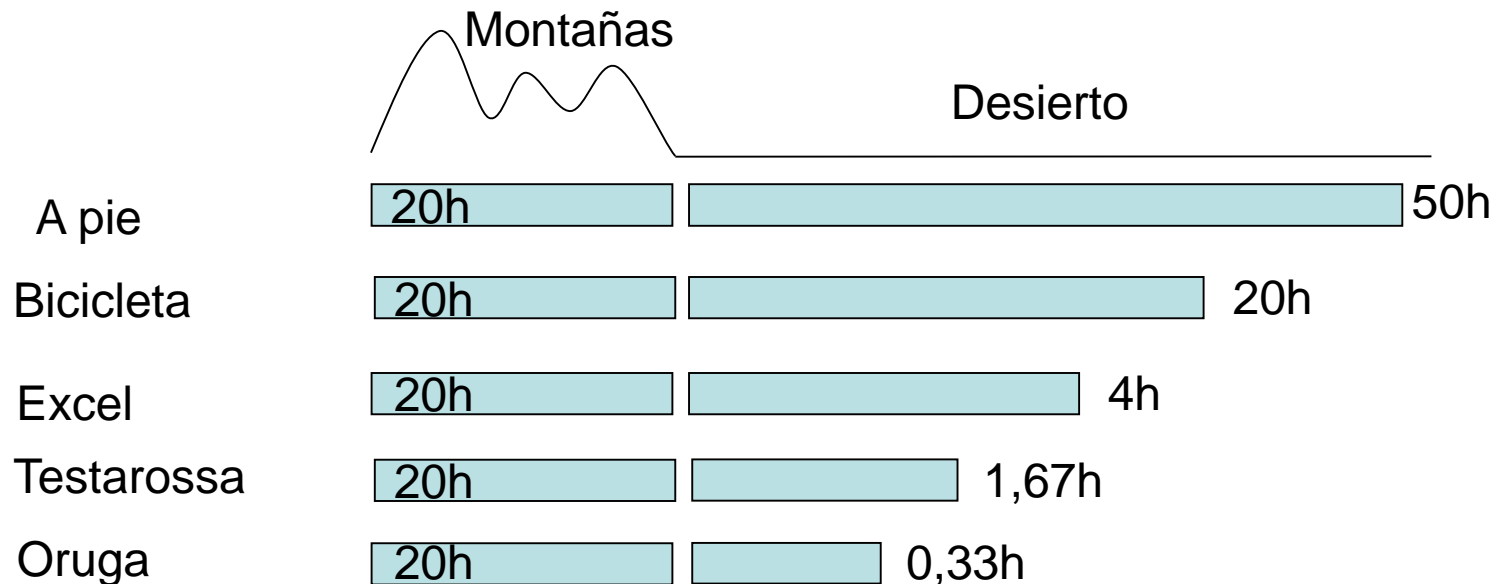
Concepto

Amdahl

Relación

Análisis de
rendimiento

- ❖ **Ley de Amdahl:** aceleración dependiente de dos factores:
- ❖ **Fracción mejorada:** fracción de tiempo de la opción sin la mejora que puede utilizarse para aprovechar la mejora (siempre menor o igual que uno). En el ejemplo anterior 50/70.
- ❖ **Aceleración mejorada:** Aceleración del modo mejorado (mayor que uno). En el ejemplo anterior aceleración en el desierto.



2.1 Rendimiento. Concepto y definiciones

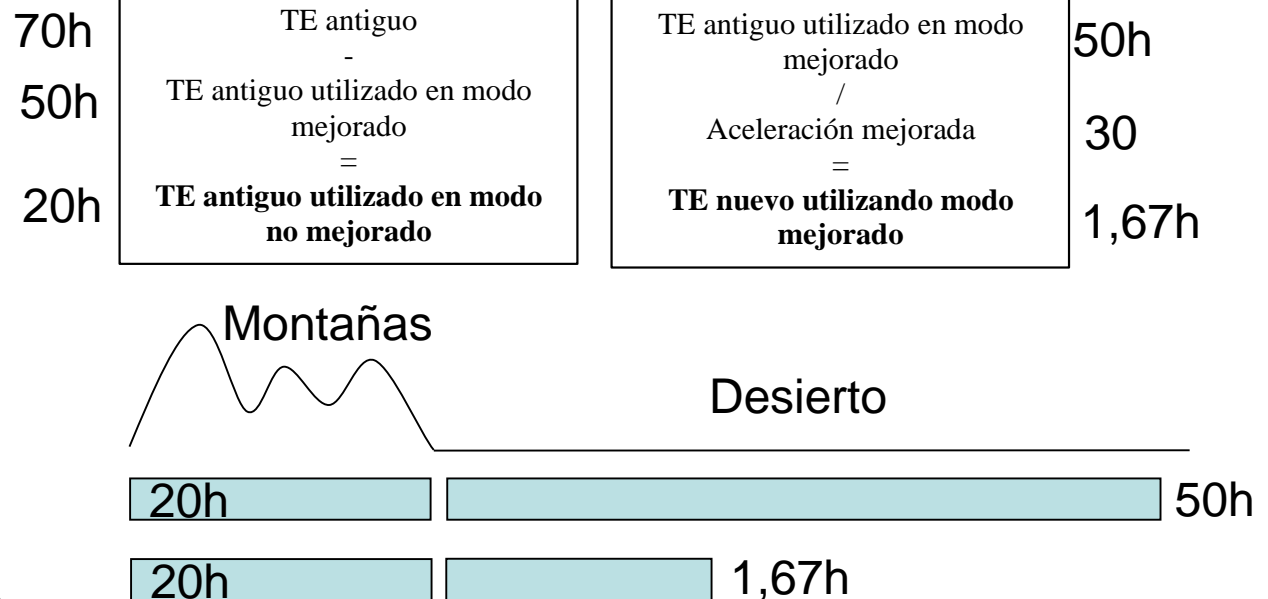
Concepto

Amdahl

Relación

- El tiempo de ejecución nuevo (máquina original con el modo mejorado) (21,67) es el tiempo empleado sin utilizar la parte mejorada (20) mas el tiempo empleado utilizando la parte mejorada (1,67) (tiempo a pie + tiempo en Testarossa)

$$TE_{nuevo} = TE_{antiguo} \left(\underbrace{(1 - Fracción_{mejorada})}_{\text{TE antiguo - TE antiguo utilizado en modo mejorado}} + \underbrace{\frac{Fracción_{mejorada}}{Aceleración_{mejorada}}}_{\text{TE antiguo utilizado en modo mejorado / Aceleración mejorada}} \right)$$



Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

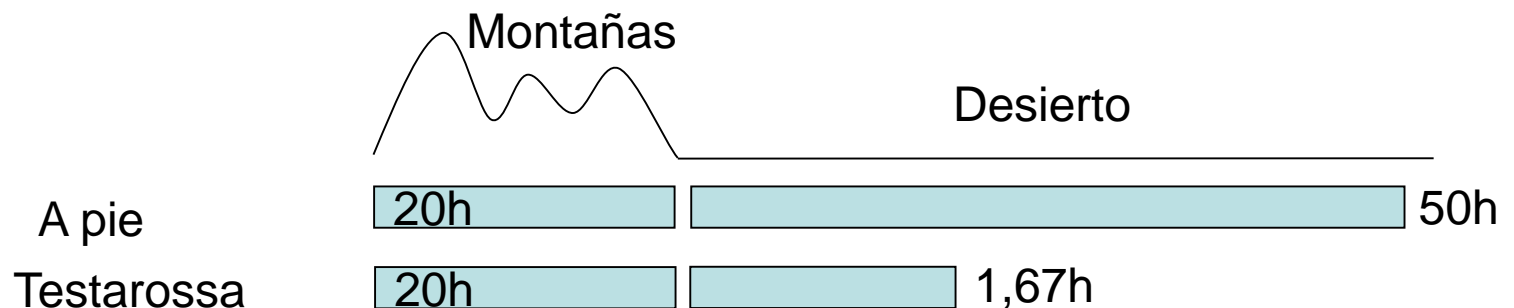
Relación

- El **tiempo de ejecución nuevo** (máquina original con el modo mejorado) (21,67) es el tiempo empleado sin utilizar la parte mejorada (20) mas el tiempo empleado utilizando la parte mejorada (1,67) (tiempo a pie + tiempo en Testarossa)

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

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

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



Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- **Ejemplo:** Suponer que estamos considerando una mejora que corra diez veces más rápida que la máquina original, pero sólo es utilizable el 40% del tiempo. ¿Cual es la aceleración global lograda al incorporar la mejora?

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- **Ejemplo:** Suponer que estamos considerando una mejora que corra diez veces más rápida que la máquina original, pero sólo es utilizable el 40% del tiempo. ¿Cual es la aceleración global lograda al incorporar la mejora?

$$\text{Fracción}_{\text{mejorada}} = 0,4$$

$$\text{Aceleración}_{\text{mejorada}} = 10$$

$$\text{Aceleración}_{\text{global}} = \frac{1}{0,6 + \frac{0,4}{10}} = \frac{1}{0,64} \approx 1,56$$

- **Ejemplo:** Suponer que una cache es 5 veces más rápida que la memoria principal y supongamos que la cache puede ser utilizada el 90% del tiempo ¿Qué aumento de velocidad se logrará al utilizar la cache?

Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- **Ejemplo:** Suponer que estamos considerando una mejora que corra diez veces más rápida que la máquina original, pero sólo es utilizable el 40% del tiempo. ¿Cual es la aceleración global lograda al incorporar la mejora?

$$\text{Fracción}_{\text{mejorada}} = 0,4$$

$$\text{Aceleración}_{\text{mejorada}} = 10$$

$$\text{Aceleración}_{\text{global}} = \frac{1}{0,6 + \frac{0,4}{10}} = \frac{1}{0,64} \approx 1,56$$

- **Ejemplo:** Suponer que una cache es 5 veces más rápida que la memoria principal y supongamos que la cache puede ser utilizada el 90% del tiempo ¿Qué aumento de velocidad se logrará al utilizar la cache?

$$\text{Fracción}_{\text{mejorada}} = 0,9$$

$$\text{Aceleración}_{\text{mejorada}} = 5$$

$$\text{Aceleración}_{\text{global}} = \frac{1}{0,1 + \frac{0,9}{5}} = \frac{1}{0,25} = 3,6$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- **Ejercicio 3:** Para las cuatro siguientes preguntas, supongamos que se está considerando mejorar una máquina añadiéndole un modo vectorial. Cuando se ejecuta un cálculo en modo vectorial, es 20 veces más rápido que en el modo normal de ejecución. Llamamos al porcentaje de tiempo que puede emplearse el modo vectorial porcentaje de vectorización.
- 1. Dibujar un gráfico donde se muestre la aceleración como porcentaje del cálculo realizado en modo vectorial. Rotular el eje y con "aceleración neta" y el eje x con "porcentaje de vectorización".
- 2. ¿Que porcentaje de vectorización se necesita para conseguir una aceleración de 2?
- 3. ¿Que porcentaje de vectorización se necesita para conseguir la mitad de la aceleración máxima alcanzable utilizando el modo vectorial?
- 4. Supongamos que hemos medido el porcentaje de vectorización de programas, obteniendo que es del 70%. El grupo de diseño hardware dice que puede duplicar la velocidad de la parte vectorizada con una inversión significativa de ingeniería adicional. Se desea saber si el equipo de compilación puede incrementar la utilización del modo vectorial como otra aproximación para incrementar el rendimiento. ¿Que incremento en el porcentaje de vectorización (relativo a la utilización actual) se necesitará para obtener la misma ganancia de rendimiento? ¿Que inversión es recomendable?

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

● Ejercicio 3:

- 1. Dibujar un gráfico donde se muestre la aceleración como porcentaje del cálculo realizado en modo vectorial. Rotular el eje y con "aceleración neta" y el eje x con "porcentaje de vectorización".

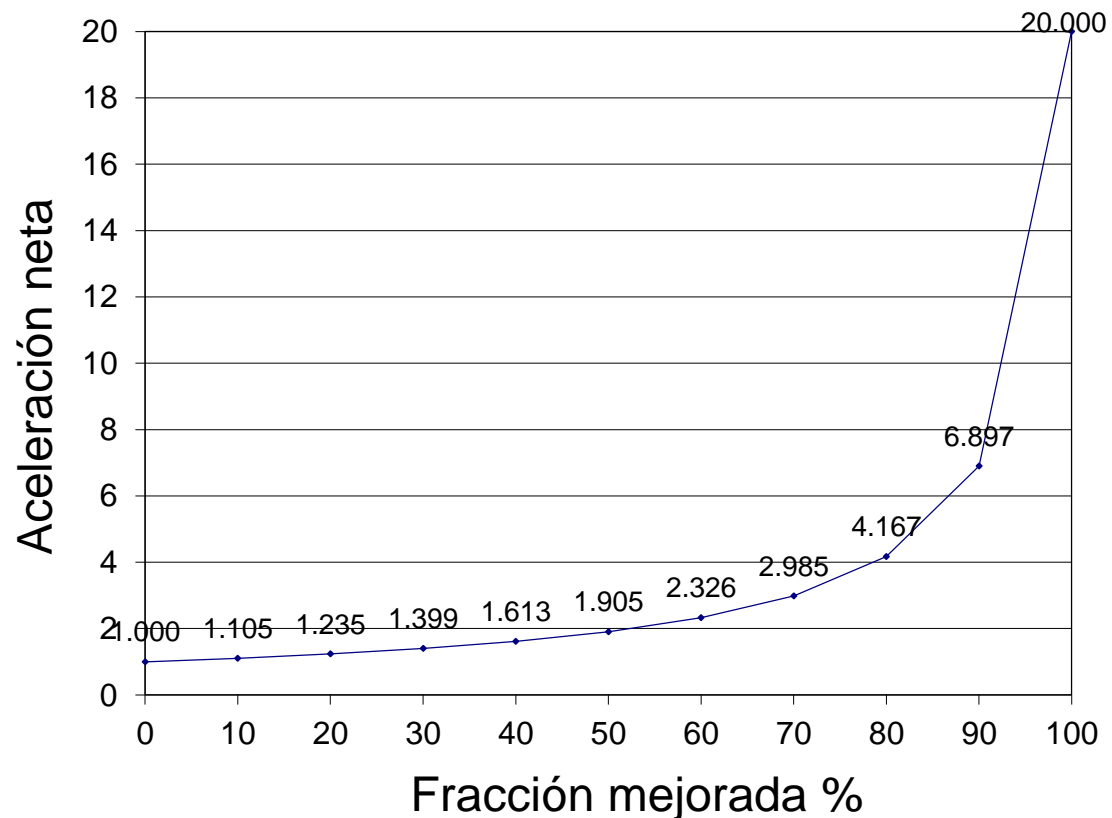
$$0\% \quad a_g = \frac{1}{1 + \frac{0}{20}} = 1$$

$$10\% \quad a_g = \frac{1}{0,9 + \frac{0,1}{20}} = 1,105$$

$$20\% \quad a_g = \frac{1}{0,8 + \frac{0,2}{20}} = 1,235$$

$$90\% \quad a_g = \frac{1}{0,1 + \frac{0,9}{20}} = 6,9$$

$$100\% \quad a_g = \frac{1}{0 + \frac{1}{20}} = 20$$



2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

• Ejercicio 3:

- 2.¿Que porcentaje de vectorización se necesita para conseguir una aceleración de 2?

$$2 = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \Rightarrow f_m = 0,526 = 52,6\%$$

- 3.¿Que porcentaje de vectorización se necesita para conseguir la mitad de la aceleración máxima alcanzable utilizando el modo vectorial?.

$$10 = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \Rightarrow f_m = 0,947 \simeq 95\%$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- 4. Hemos medido el porcentaje de vectorización de programas, obteniendo que es del 70%. El grupo de diseño hardware dice que puede duplicar la velocidad de la parte vectorizada con una inversión significativa de ingeniería adicional. Se desea saber si el equipo de compilación puede incrementar la utilización del modo vectorial como otra aproximación para incrementar el rendimiento. ¿Que incremento en el porcentaje de vectorización (relativo a la utilización actual) se necesitará para obtener la misma ganancia de rendimiento? ¿Que inversión es recomendable?.

$$TE_{\text{ant}} = TE_{\text{vec}} * 20 = TE_{\text{vec.r}} * 40 \rightarrow TE_{\text{vec}} = TE_{\text{vec.r}} * 2$$

- Aceleración global conseguida por el grupo de hardware

$$a_g = \frac{1}{(1 - 0,7) + \frac{0,7}{40}} = 3,15$$

Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- 4. Hemos medido el porcentaje de vectorización de programas, obteniendo que es del 70%. El grupo de diseño hardware dice que puede duplicar la velocidad de la parte vectorizada con una inversión significativa de ingeniería adicional. Se desea saber si el equipo de compilación puede incrementar la utilización del modo vectorial como otra aproximación para incrementar el rendimiento. ¿Que incremento en el porcentaje de vectorización (relativo a la utilización actual) se necesitará para obtener la misma ganancia de rendimiento? ¿Que inversión es recomendable?

$$TE_{ant} = TE_{vec} * 20 = TE_{vec.r} * 40 \rightarrow TE_{vec} = TE_{vec.r} * 2$$

- Incremento del % de vectorización para alcanzar $a_g = 3,15$

$$a_g = 3,15 = \frac{1}{(1 - f_m) + \frac{f_m}{20}} \Rightarrow f_m = 0,719$$

$$\Delta \% vector = |0,7 - 0,719| = 0,019 = 1,9\%$$

Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

- **Ejercicio 4:** El coprocesador de un computador mejora en un factor de 5 el procesamiento de números en coma flotante. El tiempo de ejecución de cierto programa es de 1 minuto con el coprocesador instalado, y de 2,5 minutos sin este.
- a) Calcula el porcentaje del tiempo de ejecución, sin el coprocesador instalado, que el programa realiza operaciones en coma flotante.
- b) Calcula el tiempo de ejecución del programa sin el coprocesador instalado, para realizar las operaciones en coma flotante.
- c) Calcula el tiempo de ejecución del programa con el coprocesador instalado, para realizar las operaciones en coma flotante.
- d) Calcula el tiempo de ejecución del programa para realizar las operaciones enteras.
- e) Comprueba la coherencia del planteamiento sumando los resultados de los apartados c y d, obteniendo el tiempo de ejecución del programa con el coprocesador instalado.





2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- **Ejercicio 4:** El coprocesador de un computador mejora en un factor de 5 el procesamiento de números en coma flotante. El tiempo de ejecución de cierto programa es de 1 minuto con el coprocesador instalado, y de 2,5 minutos sin este.
- a) Calcula el porcentaje del tiempo de ejecución, sin el coprocesador instalado, que el programa realiza operaciones en coma flotante.

	Tarea sin punto flotante	Tarea con punto flotante
Sin copro		 2,5m
Con copro		 1m

$$a_g = \frac{t_{e.ant}}{t_{e.nue}} = \frac{2,5}{1} = \frac{1}{(1 - f_m) + \frac{f_m}{5}} \Rightarrow f_m = 0,75$$

Análisis de
rendimiento

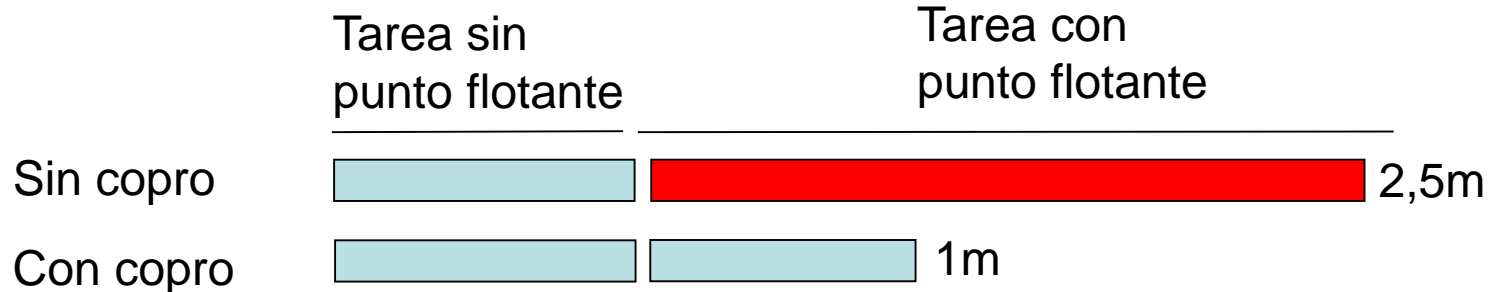
2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- **Ejercicio 4:** El coprocesador de un computador mejora en un factor de 5 el procesamiento de números en coma flotante. El tiempo de ejecución de cierto programa es de 1 minuto con el coprocesador instalado, y de 2,5 minutos sin este.
- b) Calcula el tiempo de ejecución del programa sin el coprocesador instalado, para realizar las operaciones en coma flotante.



$$t_{e.ant} = 2,5$$

$$t_{e.ant.flot} = 2,5 \square 0,75 = 1,875m$$

Análisis de
rendimiento





2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- **Ejercicio 4:** El coprocesador de un computador mejora en un factor de 5 el procesamiento de números en coma flotante. El tiempo de ejecución de cierto programa es de 1 minuto con el coprocesador instalado, y de 2,5 minutos sin este.
- c) Calcula el tiempo de ejecución del programa con el coprocesador instalado, para realizar las operaciones en coma flotante.

	Tarea sin punto flotante	Tarea con punto flotante
Sin copro		 2,5m
Con copro		 1m

$$t_{e.ant} = 2,5$$

$$t_{e.nue.flot} = \frac{1,875}{5} = 0,375m$$

Análisis de
rendimiento

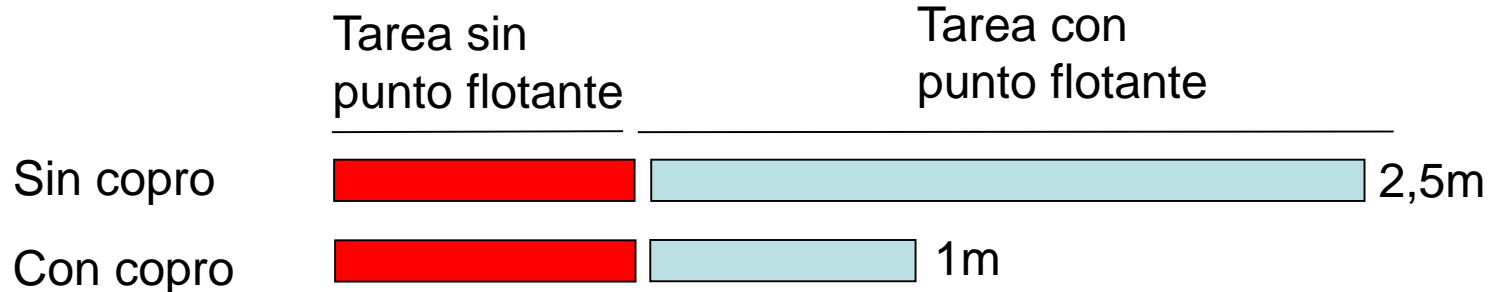
2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- **Ejercicio 4:** El coprocesador de un computador mejora en un factor de 5 el procesamiento de números en coma flotante. El tiempo de ejecución de cierto programa es de 1 minuto con el coprocesador instalado, y de 2,5 minutos sin este.
- d) Calcula el tiempo de ejecución del programa para realizar las operaciones enteras.



$$t_{e.int} = 2,5 - 1,875 = 0,625m$$

Análisis de
rendimiento

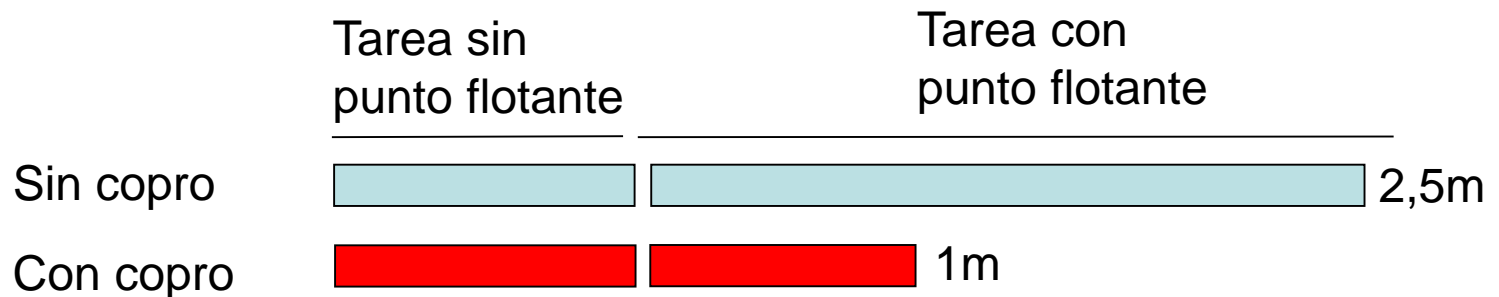
2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

- **Ejercicio 4:** El coprocesador de un computador mejora en un factor de 5 el procesamiento de números en coma flotante. El tiempo de ejecución de cierto programa es de 1 minuto con el coprocesador instalado, y de 2,5 minutos sin este.
- e) Comprueba la coherencia del planteamiento sumando los resultados de los apartados c y d, obteniendo el tiempo de ejecución del programa con el coprocesador instalado.



Análisis de
rendimiento

$$t_{nue} = t_{int} - t_{nue.flo} = 0,375 + 0,625 = 1 m$$

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Relación rendimiento y coste

¿Por qué el cliente se decide por un computador en lugar de otro?

- ❖ Las medidas estándares del rendimiento permiten comparar cuantitativamente a los diseñadores
- ❖ Las comparativas basan sus informes en aspectos relacionados con el rendimiento y coste de compra

2.1 Rendimiento. Concepto y definiciones

Concepto

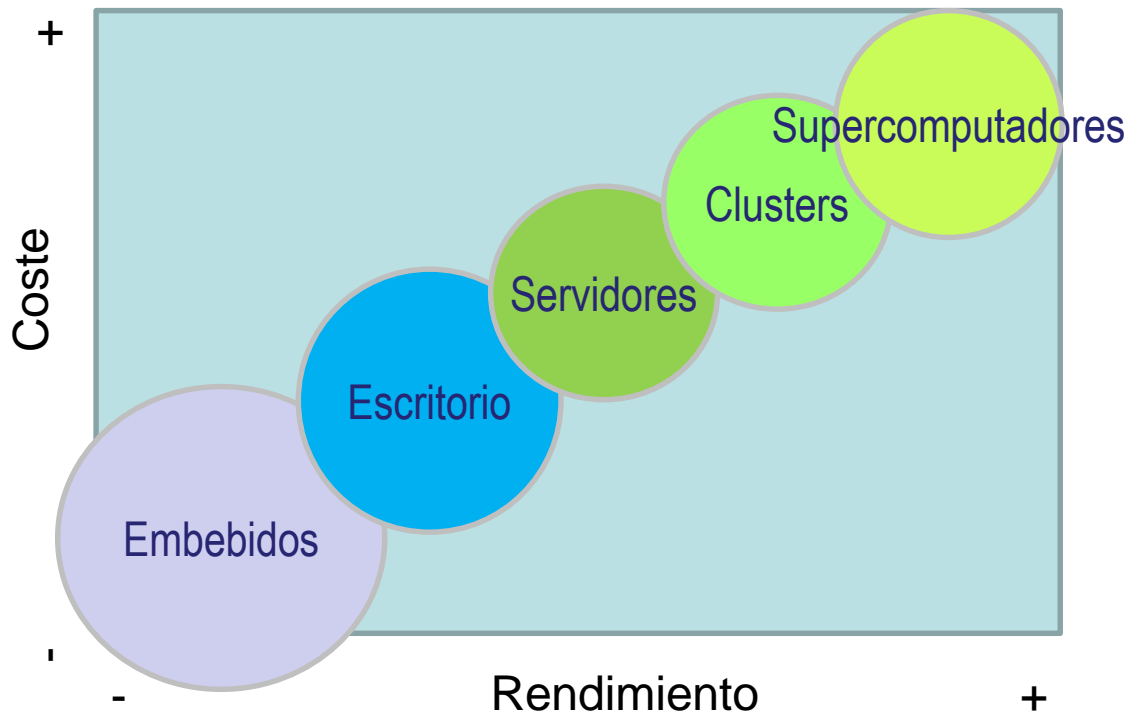
Amdahl

Relación

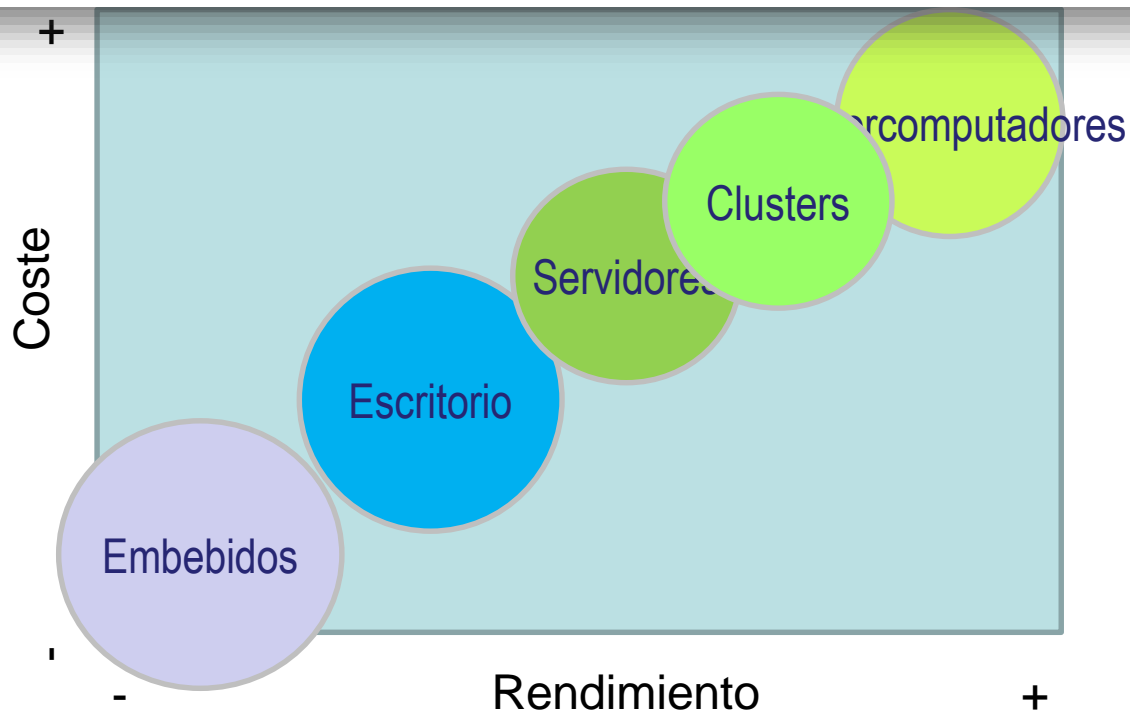
Análisis de
rendimiento

Relación rendimiento y coste

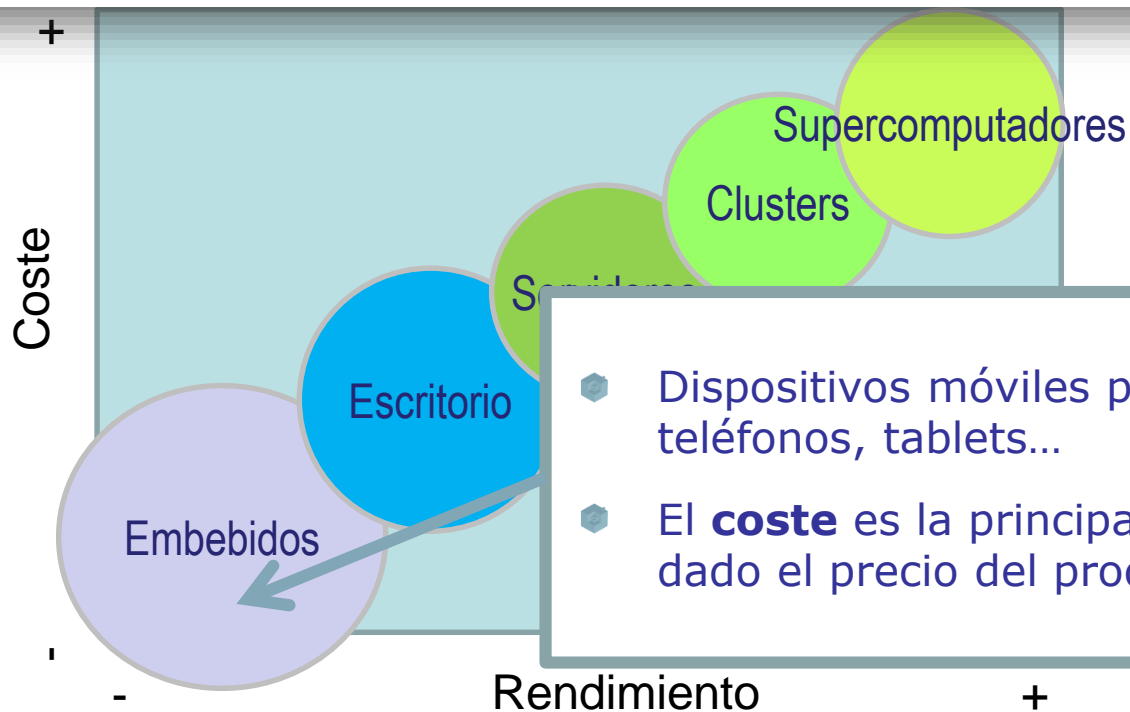
- El diseño de computadores abarca desde el alto coste alto rendimiento hasta el bajo coste bajo rendimiento



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

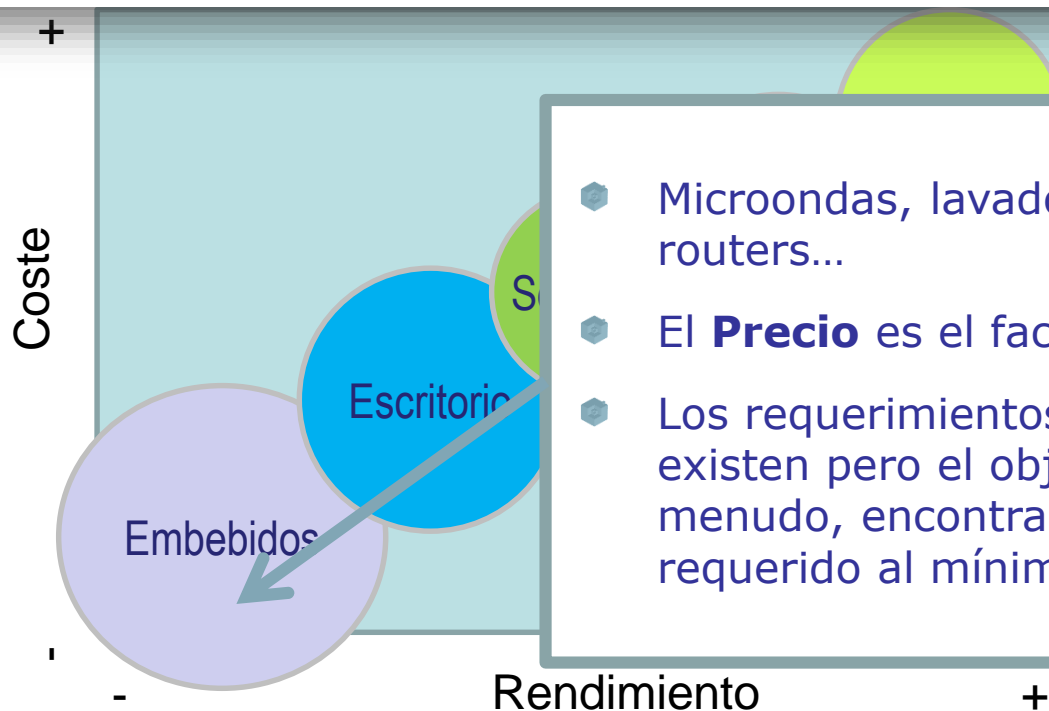


Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	\$100–\$1000	\$300–\$2500	\$5000–\$10,000,000	\$100,000–\$200,000,000	\$10–\$100,000
Price of micro-processor	\$10–\$100	\$50–\$500	\$200–\$2000	\$50–\$250	\$0.01–\$100
Critical system design 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



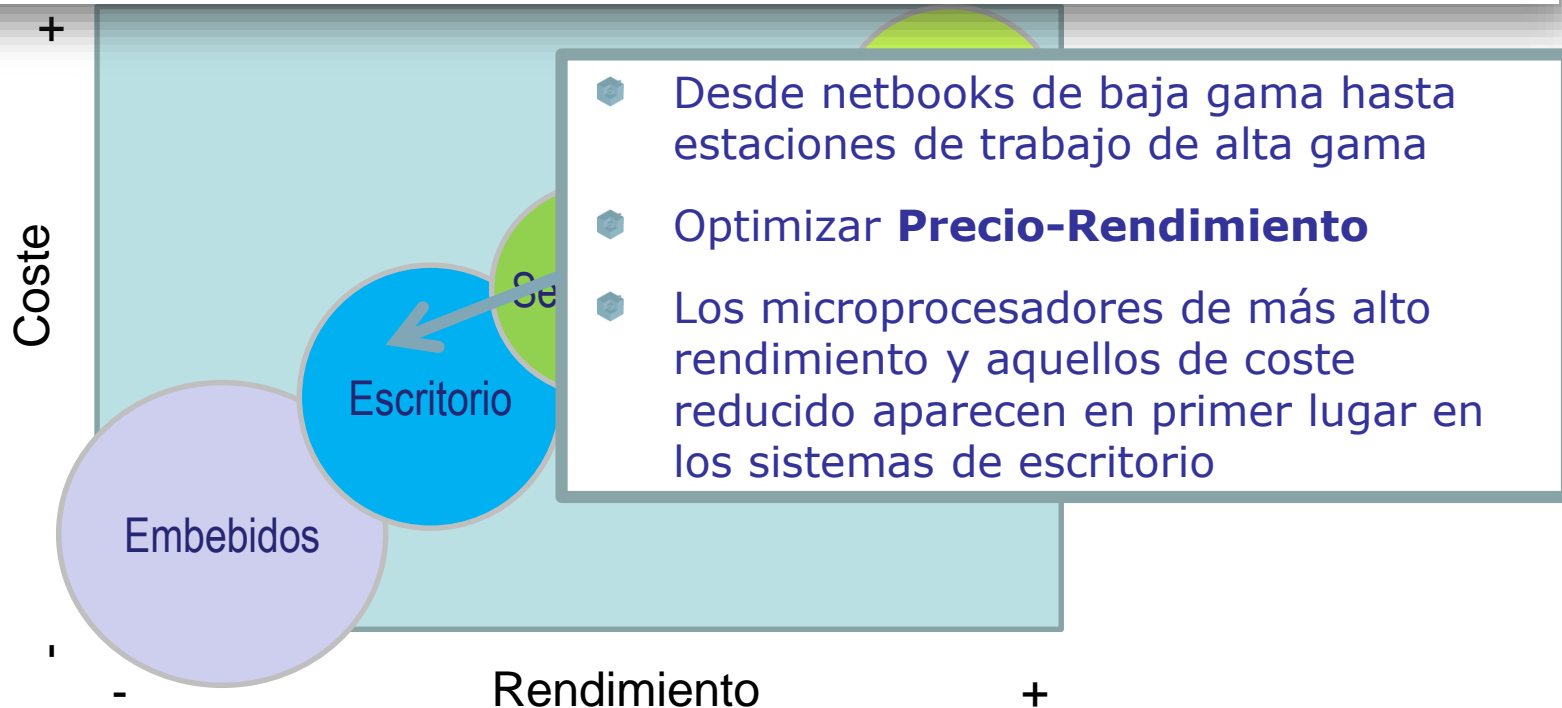
- Dispositivos móviles personales: teléfonos, tablets...
- El **coste** es la principal preocupación dado el precio del producto

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





- Microondas, lavadoras, impresoras, routers...
- El **Precio** es el factor clave
- Los requerimientos de rendimiento existen pero el objetivo principal es, a menudo, encontrar el rendimiento requerido al mínimo precio

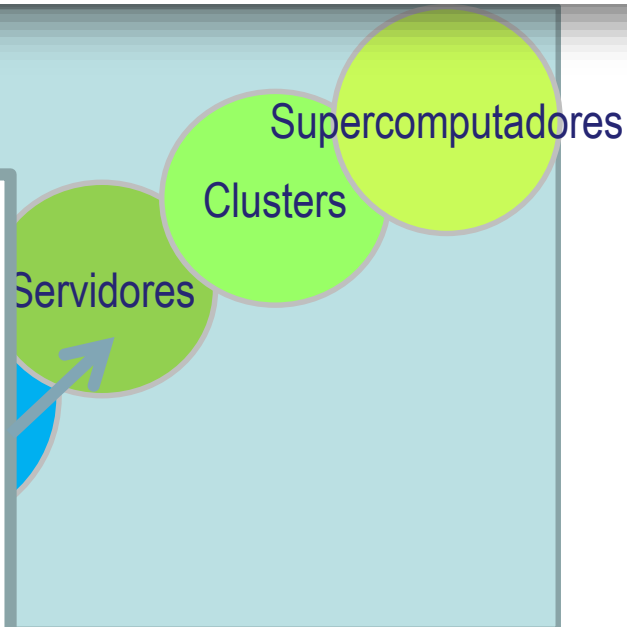
Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	\$100–\$1000	\$300–\$2500	\$5000–\$10,000,000	\$100,000–\$200,000,000	\$10–\$100,000
Price of micro-processor	\$10–\$100	\$50–\$500	\$200–\$2000	\$50–\$250	\$0.01–\$100
Critical system design 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



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

+

-  Diseñados para una **productividad** eficiente
-  Rendimiento global del servidor—en términos de transacciones por minuto o páginas web servidas por segundo—es crucial

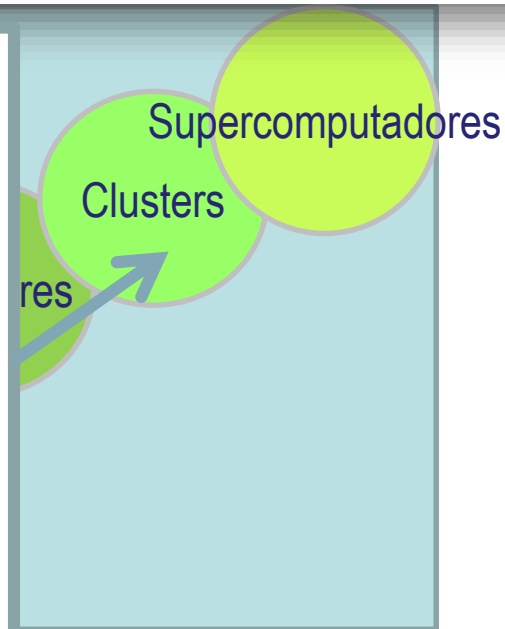


Rendimiento

+

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

- Colecciones de computadores de escritorio o servidores conectados por redes de área local que actúan como un solo computador más grande
- Relacionados con aplicaciones de búsqueda, redes sociales, compartición de video, juegos multijugador...
- **Precio-rendimiento** es crítico



Rendimiento

+

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

- Énfasis en el rendimiento del punto flotante, capaces de ejecutar grandes programas por lotes que pueden correr por semanas
- **Rendimiento** es crítico. El coste es menos importante

Supercomputadores
clusters



Rendimiento

+

2.1 Rendimiento. Concepto y definiciones

Coste

Concepto

Amdahl

Relación

- El coste es un parámetro a tener muy en cuenta al diseñar un nuevo procesador o al modificar uno existente
- Los factores principales que influyen en el coste de un computador son:
 - Curva de aprendizaje:**
 - Costes de manufacturación decrecen a lo largo del tiempo incluso sin mejoras en la tecnología de implementación básica
 - El porcentaje de dispositivos manufacturados que pasan los procedimientos de prueba se incrementa a lo largo del tiempo

Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Coste

- El coste es un parámetro a tener muy en cuenta al diseñar un nuevo procesador o al modificar uno existente
- Los factores principales que influyen en el coste de un computador son:
 - Curva de aprendizaje:**
 - Volumen:** El incremento del volumen afecta al coste en:
 - Reduciendo el tiempo necesario para bajar la curva de aprendizaje
 - Incrementando la eficiencia de compra y manufactura (10% menos por cada doble de volumen)
 - Reduciendo la cantidad del coste de diseño que debe ser amortizado para cada computador

2.1 Rendimiento. Concepto y definiciones

Coste

Concepto

Amdahl

Relación

- El coste es un parámetro a tener muy en cuenta al diseñar un nuevo procesador o al modificar uno existente
- Los factores principales que influyen en el coste de un computador son:
 - Curva de aprendizaje:**
 - Volumen:**
 - Mercado altamente competitivo:**
 - Las DRAMs, discos, monitores, teclados son vendidos por diferentes fabricantes y son esencialmente idénticos
 - La competencia reduce la distancia entre el precio de venta y el coste, pero también reduce el coste porque:
 - Los componentes tienen tanto un gran volumen como una clara definición

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Coste de un circuito integrado (IC)

- Los costes de los ICs se están convirtiendo en una porción cada vez más grande del coste que varía entre computadores
- Los factores que influyen en el coste del silicio son:
 - El número de puertas:** influye en el número de transistores necesarios. Un aumento de estos requiere un área de silicio mayor
 - Conexiones** entre elementos: el número y la longitud
 - Regularidad** del diseño: cuanto más regular sea el diseño, menos área ocupará
- Los procesos de IC están caracterizados por el "feature size":
 - Tamaño mínimo de un transistor o conexión sea en la dimensión X o Y (10 micras en 1971 a 0.007 micras en 2018)

2.1 Rendimiento. Concepto y definiciones

Microprocessor	16-Bit address/ bus, microcoded	32-Bit address/ bus, microcoded	5-Stage pipeline, on-chip I & D caches, FPU	2-Way superscalar, 64-bit bus	Out-of-order 3-way superscalar	Out-of-order superpipelined, on-chip L2 cache	Multicore OOO 4-way on chip L3 cache, Turbo
Product	Intel 80286	Intel 80386	Intel 80486	Intel Pentium	Intel Pentium Pro	Intel Pentium 4	Intel Core i7
Year	1982	1985	1989	1993	1997	2001	2015
Die size (mm ²)	47	43	81	90	308	217	122
Transistors	134,000	275,000	1,200,000	3,100,000	5,500,000	42,000,000	1,750,000,000
Processors/chip	1	1	1	1	1	1	4
Pins	68	132	168	273	387	423	1400
Latency (clocks)	6	5	5	5	10	22	14
Bus width (bits)	16	32	32	64	64	64	196
Clock rate (MHz)	12.5	16	25	66	200	1500	4000
Bandwidth (MIPS)	2	6	25	132	600	4500	64,000
Latency (ns)	320	313	200	76	50	15	4

- Los procesos de IC están caracterizados por el "feature size":
 - Tamaño mínimo de un transistor o conexión sea en la dimensión X o Y (7 micras en 1971 a 0.007 micras en 2018)

2.1 Rendimiento. Concepto y definiciones

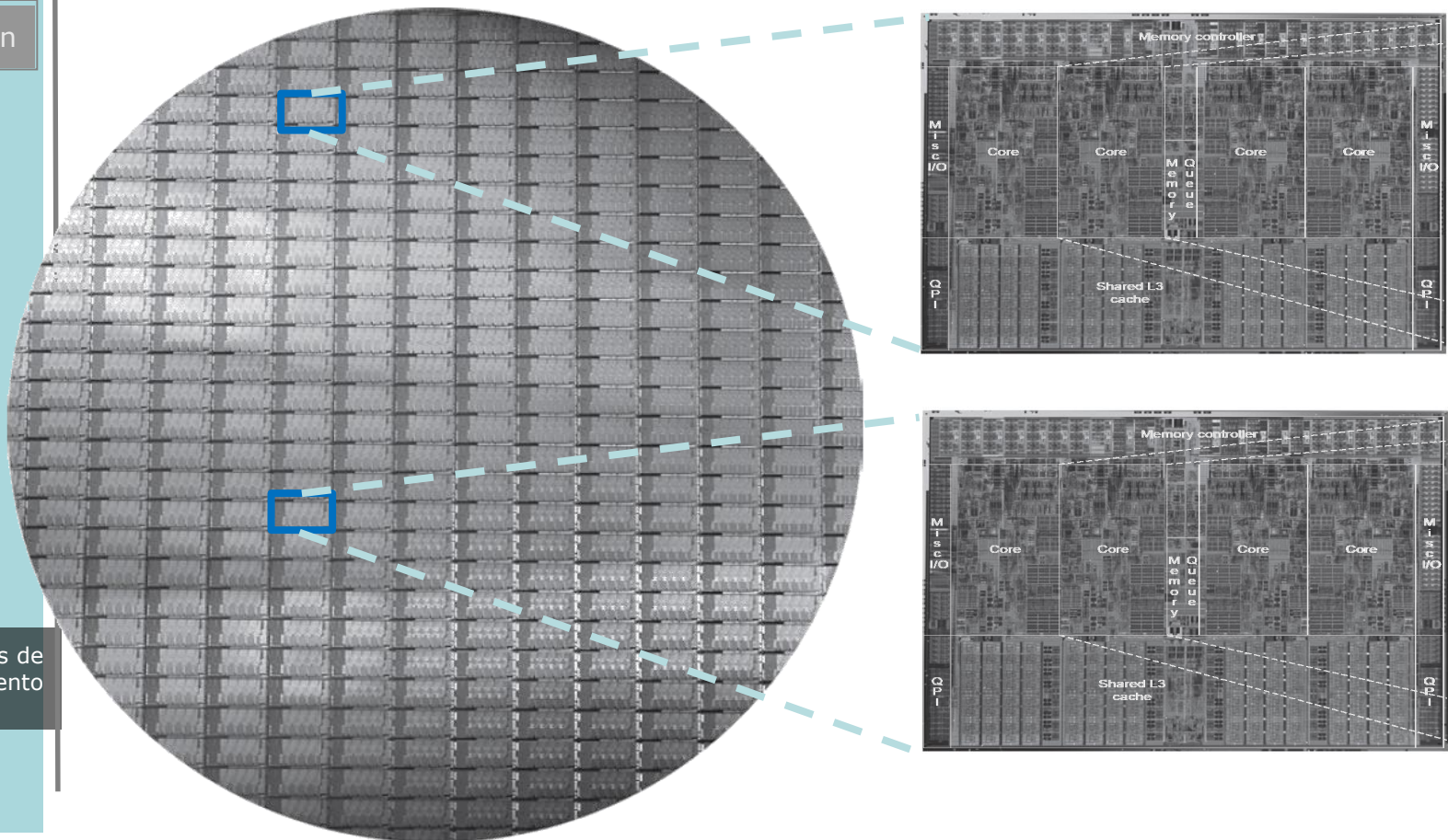
Concepto

Amdahl

Relación

Coste de un circuito integrado (IC)

- El proceso básico de fabricación del silicio no ha cambiado: la oblea es testeada y cortada en dados que son empaquetados



Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

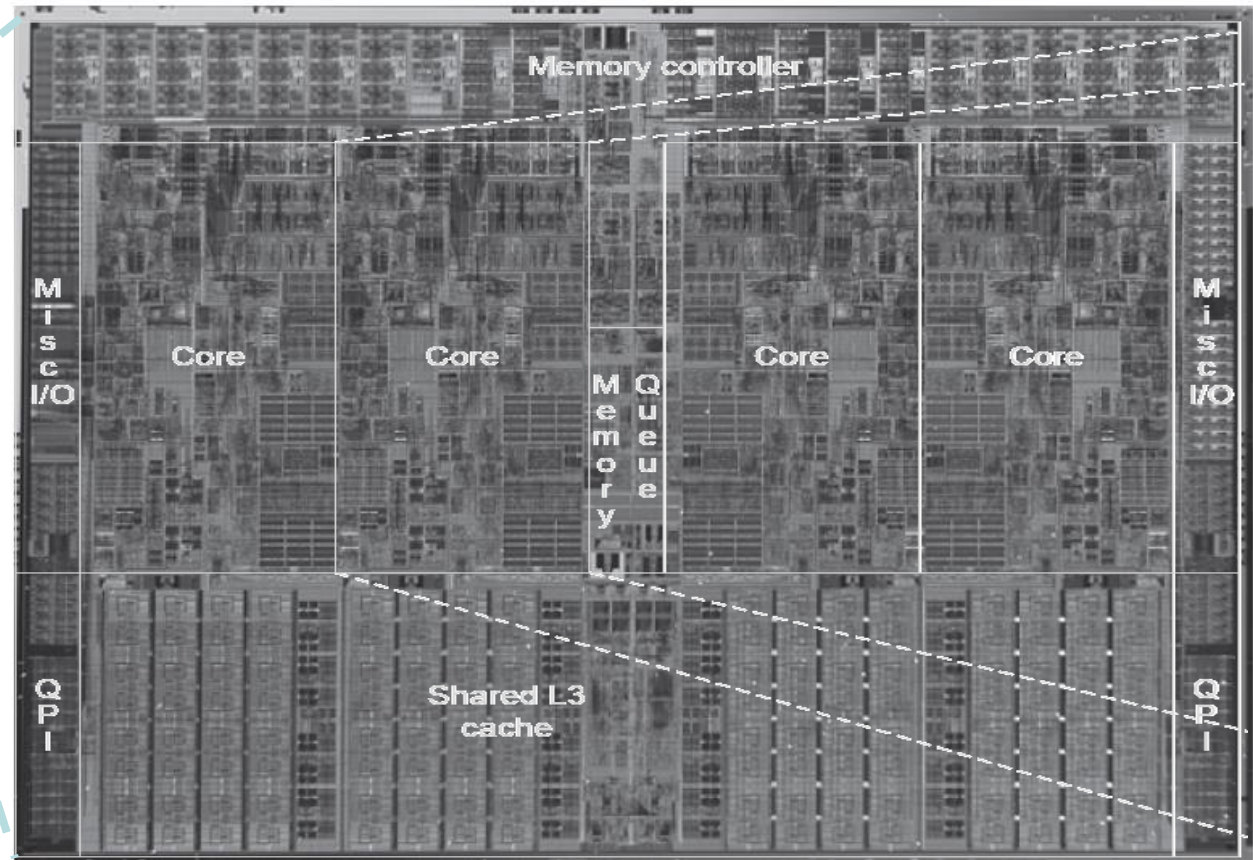
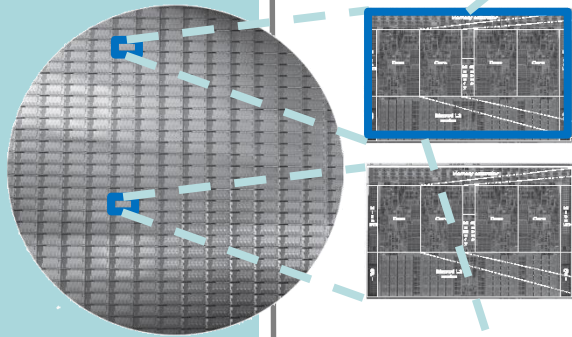
Concepto

Amdahl

Relación

Coste de un circuito integrado (IC)

- El proceso básico de fabricación del silicio no ha cambiado: la oblea es testeada y cortada en dados que son empaquetados



Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Coste de un circuito integrado (IC)

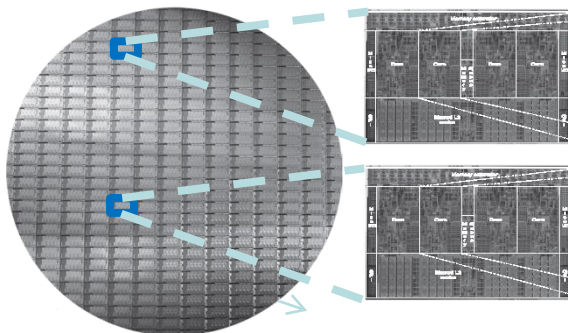
- El proceso básico de fabricación del silicio no ha cambiado: la oblea es testeada y cortada en dados que son empaquetados

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

- El número de dados por oblea es aproximadamente el área de la oblea dividida por el área del dado. De forma más precisa, puede estimarse por:

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2} \times \text{Die area}}$$



- Compensa los dados cerca de la periferia de la oblea. Aproximadamente el número de dados a lo largo del borde

2.1 Rendimiento. Concepto y definiciones

Coste de un circuito integrado (IC)

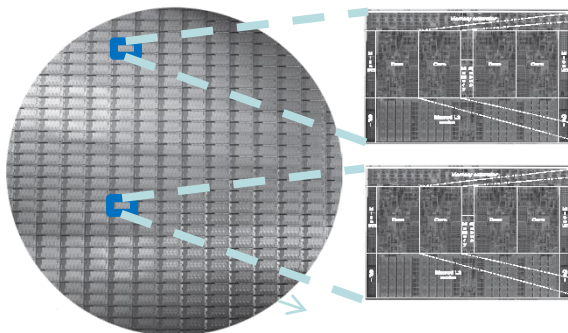
- Este es el numero máximo de dados por oblea:

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2} \times \text{Die area}}$$

- ¿Cual es la fracción de dados no defectuosos en la óblea, o el rendimiento del dado (die yield)? Asumiendo que:
 - Los defectos están distribuidos aleatoriamente
 - el rendimiento es inversamente proporcional a la complejidad del proceso de fabricación

$$\text{Die yield} = \text{Wafer yield} \times 1 / (1 + \text{Defects per unit area} \times \text{Die area})^N$$

- Fórmula Bose–Einstein: modelo empírico teniendo en cuenta el rendimiento en muchas líneas de fabricación



- Defectos por cm² (2017)
 - = 0.012-0.016 (28 nm)
 - = 0.016-0.047 (16 nm)
- N, factor de complejidad del proceso
 - = 7.5-9.5 (28 nm)
 - = 10-14 (16 nm)

Concepto

Amdahl

Relación

Análisis de
rendimiento

2.1 Rendimiento. Concepto y definiciones

Concepto

Amdahl

Relación

Análisis de
rendimiento

Ejercicio

- Intel utiliza obleas de 300mm para construir su Core i7. Estas obleas tienen un coste de 20000€. El dado del Core i7 tiene unas dimensiones 20.7 mm x 10.5 mm. Asumiendo que el proceso de fabricación de Intel tiene una densidad de defectos de 0.023 por cm^2 y que el factor de complejidad del proceso para un tamaño mínimo del transistor de 32 nm es 13.5, calcula el coste del dado. Por simplicidad, se asume que el rendimiento de la oblea es del 100% ¿Cuál sería el coste del dado dentro de 2 años, sabiendo que el factor de complejidad del proceso para la tecnología de 32nm decrece un 5% por año?

2.2. Evaluación del rendimiento

Tema 2 Análisis del rendimiento

Arquitectura de los Computadores

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Tiempo de ejecución

- El tiempo es la medida más fiable del rendimiento
- El tiempo de ejecución de un programa se mide en segundos

El rendimiento

- La relación entre el tiempo y el rendimiento es inversa
- El rendimiento se mide como una frecuencia de eventos por segundo

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

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Tiempo de programa / Tiempo de CPU

- ⚙ **Tiempo de reloj, tiempo de respuesta, tiempo transcurrido:** Latencia para completar una tarea incluyéndolo todo: accesos a disco, accesos a memoria, actividades de entrada salida, gastos del sistema operativo, multiprogramación...
- ⚙ **Rendimiento del sistema:** Este término se utiliza para referenciar el tiempo transcurrido en un sistema no cargado.
- ⚙ **Tiempo de CPU**
 - ⚙ CPU usuario + CPU sistema
 - ⚙ Tiempo en que la CPU está calculando sin incluir tiempos de espera para E/S o para ejecución de otros programas
 - ⚙ **Tiempo de programa:** Este término se refiere al tiempo de CPU del usuario (nos centraremos en rendimiento de la CPU)

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Tiempo de programa / Tiempo de CPU

- **Tiempo de reloj, tiempo de respuesta, tiempo transcurrido:** Latencia para completar una tarea incluyéndolo todo: accesos a disco, accesos a memoria, actividades de entrada salida, gastos del sistema operativo, multiprogramación...
- **Rendimiento del sistema:** Este término se utiliza para referenciar el tiempo transcurrido en un sistema no cargado.

La función *time* de Unix produce una salida de la forma: 90.7u 12.9s 2:39 65%, donde:

- Tiempo de CPU del usuario = 90.7 segundos
- Tiempo de CPU utilizado por el sistema = 12.9 segundos
- Tiempo de CPU = 90.7 seg. + 12.9seg = 103.6
- Tiempo de respuesta = 2 minutos 39 segundos = 159 segundos
- Tiempo de CPU = 65% del tiempo de respuesta = 159 segundos * 0.65 = 103.6
- Tiempo esperando operaciones de E/S y/o el tiempo ejecutando otras tareas 35% del tiempo de respuesta = 159 segundos * 0.35 = 55.6 segundos

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Tiempo de programa / Tiempo de CPU

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

$$\text{Tiempo de CPU} = \frac{\text{Ciclos de reloj de CPU para un programa}}{\text{Duración del ciclo de reloj}}$$

$$\text{Tiempo de CPU} = \frac{\text{Ciclos de reloj de CPU para un programa}}{\text{Frecuencia de reloj}}$$

- No tiene sentido mostrar el tiempo transcurrido en función del ciclo de reloj ya que la latencia de los dispositivos de entrada salida es independiente del ciclo de reloj de la CPU

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

CPI

- Número medio de ciclos de reloj por instrucción. Se expresa en función del número de ciclos de reloj y el número de instrucciones ejecutadas

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

- Podemos expresar el tiempo de CPU en función del CPI:

$$\text{Tiempo de CPU} = \frac{\text{Ciclos de reloj de CPU para un programa}}{\text{Ciclos de reloj de CPU para un programa}} * \text{Duración del ciclo de reloj}$$

$$\text{Tiempo de CPU} = \frac{\text{Recuento de instrucciones}}{\text{Recuento de instrucciones}} * \text{CPI} * \text{Duración del ciclo de reloj}$$

$$\text{Tiempo de CPU} = \text{RI} * \text{CPI} * \text{clk}$$

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

En ocasiones se detalla el CPI por cada tipo de instrucción estática i

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

I_i = instrucciones dinámicas para cada tipo de instrucción estática i .

CPI_i = Número medio de ciclos de reloj para la instrucción tipo i .

Esto nos permite expresar

$$\text{Tiempo de CPU} = \sum_{i=1}^n (CPI_i \cdot I_i) \cdot \text{Duración del ciclo de reloj}$$

$$CPI = \frac{\sum_{i=1}^n (CPI_i \cdot I_i)}{\text{recuento de instrucciones}} = \sum_{i=1}^n (CPI_i \cdot \frac{I_i}{\text{recuento de instrucciones}})$$

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

2.2 Evaluación del rendimiento

Métricas

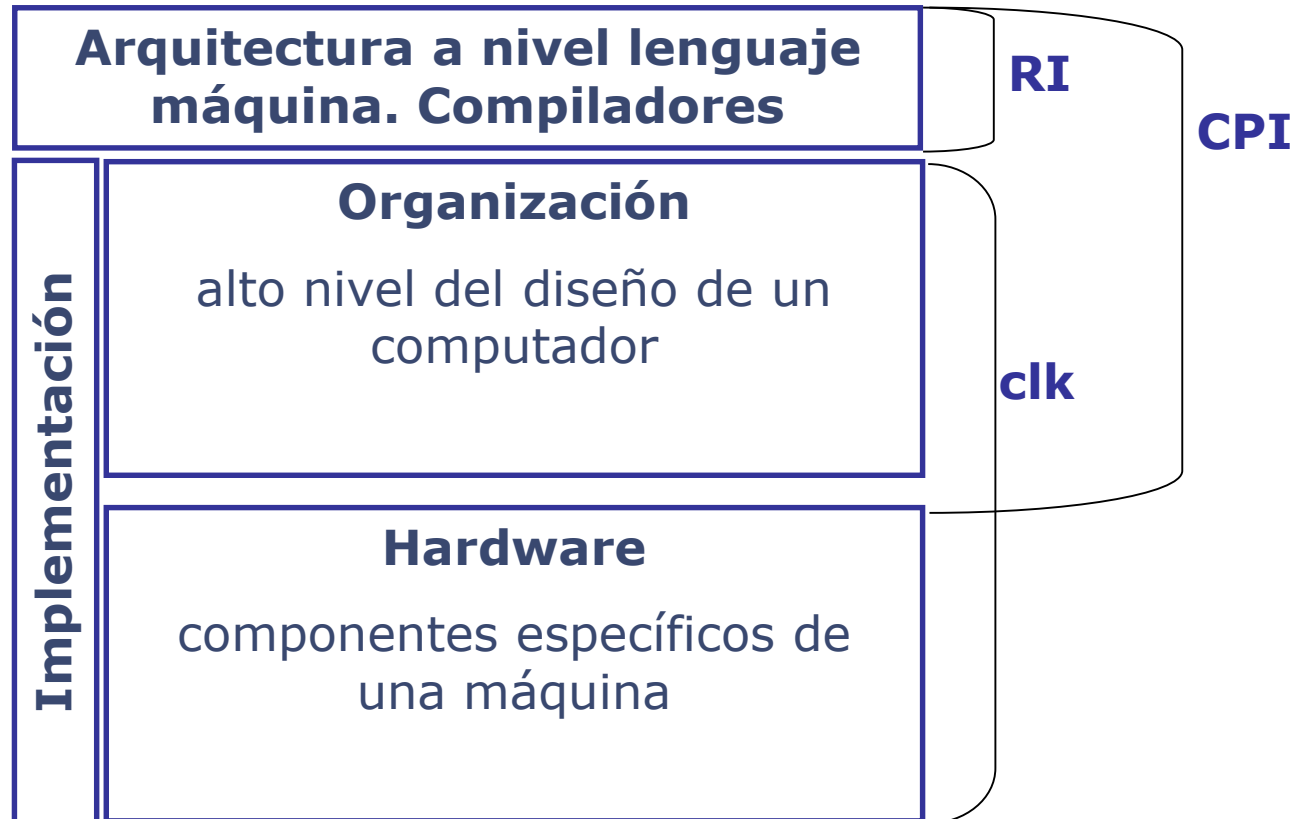
Benchmarks

Formulación

Análisis de
rendimiento

Tres parámetros interdependientes

$$\text{Tiempo de CPU} = \text{RI} * \text{CPI} * \text{clk}$$



2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Ejemplo: Suponer que estamos considerando dos alternativas para una instrucción de salto condicional:

CPU A. Una instrucción de comparación inicializa un código de condición y es seguida por un salto que examina el código de condición.

CPU B. Se incluye una comparación en el salto.

En ambas CPU, la instrucción de salto condicional emplea 2 ciclos de reloj, y las demás instrucciones 1 (en este sencillo ejemplo se están despreciando las pérdidas del sistema de memoria). En la CPU A, el 20% de todas las instrucciones ejecutadas son saltos condicionales; como cada salto necesita una comparación, otro 20% de las instrucciones son comparaciones. Debido a que la CPU A no incluye la comparación en el salto, su ciclo de reloj es un 25% más rápido que el de la CPU B. ¿Qué CPU es más rápida?.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Ejemplo: Después de ver el análisis, un diseñador consideró que, volviendo a trabajar en la organización, la diferencia de las duraciones de los ciclos de reloj podía reducirse, fácilmente, a un 10%. ¿Qué CPU es más rápida ahora?.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Ejemplo: Supongamos que estamos considerando otro cambio en un repertorio de instrucciones. La máquina, inicialmente, sólo tiene instrucciones de carga y de almacenamiento en memoria, y, después, todas las operaciones se realizan en los registros. Tales máquinas se denominan máquinas de carga almacenamiento (load/store). A continuación observamos medidas de la máquina de carga almacenamiento que muestran la frecuencia de instrucciones, denominada mezcla de instrucciones (instruction mix) y número de ciclos de reloj por instrucción.

Operación	Frecuencia	Cuenta de ciclos de reloj
Ops ALU	43%	1
Load	21%	2
store	12%	2
Saltos	24%	2

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Supongamos que el 25% de las operaciones de la unidad aritmético lógica (ALU) utilizan directamente un operando cargado que no se utiliza de nuevo.

Proponemos añadir instrucciones a la ALU que tengan un operando fuente en memoria. Estas nuevas instrucciones de registro memoria emplean dos ciclos de reloj. Supongamos que el repertorio extendido de instrucciones incrementa en 1 el número de ciclos de reloj para los saltos, pero sin afectar a la duración del ciclo de reloj. ¿Mejorará este cambio el rendimiento de la CPU?.

Operación	Frecuencia	Cuenta de ciclos de reloj
Ops ALU	43%	1
Load	21%	2
store	12%	2
Saltos	24%	2

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Alternativas para la medida del rendimiento

- La medida más fiable del rendimiento es el **tiempo de ejecución** de los **programas reales**
- Alternativas al tiempo como medida del rendimiento y a los programas reales como objetos de medida han conducido a errores en el diseño de computadores

MIPS Millones de instrucciones por segundo

$$MIPS = \frac{\text{recuento de instrucciones}}{\text{Tiempo de ejecución} \cdot 10^6}$$

- Los MIPS se muestran como un parámetro intuitivo para reflejar el rendimiento. Máquinas más rápidas tienen MIPS más altos.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Alternativas para la medida del rendimiento

Considerando

$$\text{Re cuenta de instrucciones} = \frac{\text{tiempo de ejecución}}{\text{CPI} \cdot \text{ciclo de reloj}}$$

$$\text{MIPS} = \frac{\text{recuento de instrucciones}}{\text{Tiempo de ejecución} \cdot 10^6}$$

$$\text{MIPS} = \frac{\frac{\text{tiempo de ejecución}}{\text{CPI} \cdot \text{ciclo de reloj}}}{\text{Tiempo de ejecución} \cdot 10^6} = \frac{1}{\text{CPI} \cdot \text{ciclo de reloj} \cdot 10^6}$$

$$\text{MIPS} = \frac{\text{Frecuencia de reloj}}{\text{CPI} \cdot 10^6}$$

2.2 Evaluación del rendimiento

Métricas

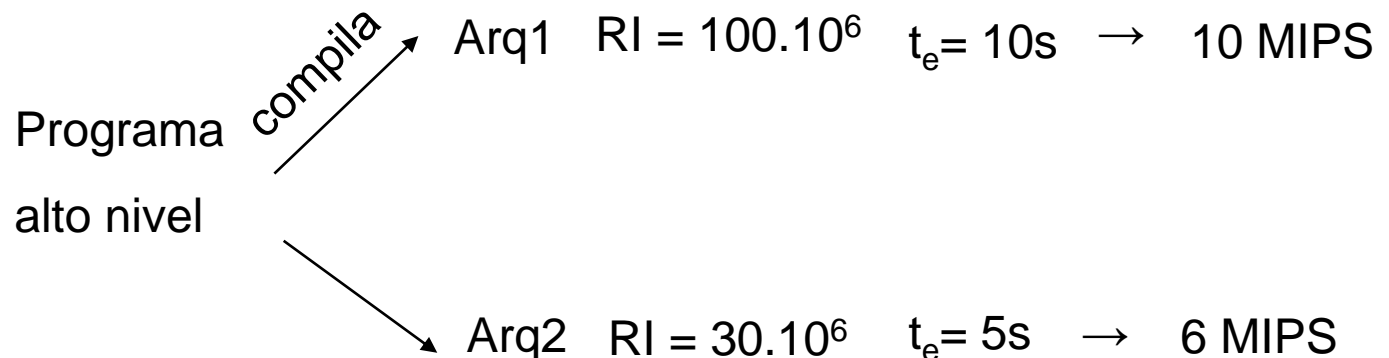
Benchmarks

Formulación

Análisis de
rendimiento

Problemas derivados de la utilización de los MIPS

- ❖ **Dependientes del repertorio de instrucciones.** No es aconsejable comparar los MIPS de computadores con repertorios de instrucciones diferentes.
 - ❖ Reflejan el ritmo de ejecución de instrucciones
 - ❖ No reflejan la efectividad del repertorio RI
 - ❖ Los MIPS pueden variar inversamente al rendimiento.
- ❖ **Dependencia del programa** en el mismo computador.
- ❖ **Ejemplo:**



2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento



Ejemplo: Supongamos que construimos un compilador optimizado para la máquina de carga/almacenamiento descrita en el ejemplo anterior. El compilador descarta el 50% de las instrucciones de la ALU aunque no pueda reducir cargas, almacenamientos ni saltos. Ignorando las prestaciones del sistema y suponiendo una duración del ciclo de reloj de 20 ns (frecuencia de reloj de 50Mhz). ¿Cuál es la frecuencia en MIPS para el código optimizado frente al código sin optimizar? ¿Está el criterio de los MIPS de acuerdo con el del tiempo de ejecución?.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

MIPS relativos y MIPS nativos

- ⚙ **Tiempo referencia**= tiempo de ejecución de un programa en la máquina de referencia
- ⚙ **Tiempo no estimado** = tiempo de ejecución del mismo programa en la máquina que se va a medir
- ⚙ **MIPS** = estimación de los MIPS de la máquina de referencia

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

- ⚙ Los MIPS relativos se apoyan en el tiempo de ejecución
- ⚙ En los años 80 la máquina dominante como referencia era la VAX-11/780, denominada máquina de 1 MIPS.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de rendimiento

FLOPS

FLOPS = Operaciones de punto flotante por segundo

$$MFLOPS = \frac{\text{Número de operaciones en punto flotante de un programa}}{\text{Tiempo de ejecución} \cdot 10^6}$$

$$GFLOPS = \frac{MFLOPS}{10^3}$$

❖ **Problemas** derivados de la utilización de los FLOPS

❖ **Dependencia del repertorio**

❖ Término basado en operaciones con objetivo de poder utilizarlo para comparar diferentes máquinas.

❖ El conjunto de operaciones en punto flotante no es consistente en diferentes máquinas (CRAY-2 no tiene instrucciones de dividir mientras que el motorola 68882 si).

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de rendimiento

FLOPS

FLOPS = Operaciones de punto flotante por segundo

$$MFLOPS = \frac{\text{Número de operaciones en punto flotante de un programa}}{\text{Tiempo de ejecución} \cdot 10^6}$$

$$GFLOPS = \frac{MFLOPS}{10^3}$$

- ❖ **Problemas** derivados de la utilización de los FLOPS
 - ❖ **Dependencia del repertorio**
 - ❖ **Dependencia del programa**
 - ❖ La estimación de los MFLOPS cambia según la mezcla de operaciones rápidas y lentas en punto flotante del programa.
 - ❖ Si el 100% de las operaciones en punto flotante son sumas, la estimación será mayor que si el 100% son divisiones

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

FLOPS Normalizados

- 🔧 Solución a los problemas de FLOPS
- 🔧 Operaciones normalizadas (Livermore Loops)

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

🔧 Programa 1

$4 \cdot 10^6$ sumas

$3 \cdot 10^6$ div

$t_e = 10s$

MFLOPS Nativos

$$\frac{7 \cdot 10^6}{10 \cdot 10^6} = 0,7 MFLOPS$$

MFLOPS Normalizados

$$\frac{16 \cdot 10^6}{10 \cdot 10^6} = 1,6 MFLOPS$$

🔧 Programa 2

$7 \cdot 10^6$ div

$t_e = 17s$

MFLOPS Nativos

$$\frac{7 \cdot 10^6}{17 \cdot 10^6} = 0,4 MFLOPS$$

MFLOPS Normalizados

$$\frac{28 \cdot 10^6}{17 \cdot 10^6} = 1,65 MFLOPS$$

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento



Ejemplo: EL programa SPICE se ejecuta en la DECstation 3100 en 94 segundos. El número de operaciones en punto flotante ejecutadas en ese programa es el de la tabla. ¿Cuántos son los MFLOPS nativos para ese programa? Usando las conversiones ¿Cuántos son los MFLOPS normalizados?.

ADDD	25.999.440
SUBD	18.266.439
MULD	33.880.810
DIVD	15.682.333
COMPARED	9.745.930
NEGD	2.617.846
ABSD	2.195.930
CONVERTD	1.581.450
<i>Total</i>	109.970.178

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Programas para evaluar el rendimiento

- ⚙️ Tiempo de ejecución de la carga de trabajo del usuario (workload) (mezcla de programas y órdenes del S.O.)
- ⚙️ **Programas reales.** compiladores de C, software de tratamiento de textos como TeX y herramientas CAD como Spice
- ⚙️ **Núcleos (Kernels).** pequeños fragmentos clave de programas. Livermore Loops y Linpack.
- ⚙️ **Benchmarks reducidos (toys).** 10 y 100 líneas de código. Criba de Eratóstenes, Puzzle y clasificación rápida (quicksort).
- ⚙️ **Benchmarks Sintéticos.** se crean artificialmente intentando simular la frecuencia media de operaciones y operandos de un gran conjunto de programas. Whetstone y Dhrystone.
- ⚙️ Whetstone: instrucciones Algol principios de los años setenta.
- ⚙️ Dhrystone: Originalmente en ADA y más tarde en C y Pascal.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Programas reales frente a otros benchmarks

- Importancia para las empresas de los benchmarks: empleo de recursos para optimizar el funcionamiento de estos pero no de programas reales.
- Ejemplo extremo: empleo de optimizadores de compiladores sensibles a los benchmarks que aplican optimizaciones.
- Si se utilizasen programas reales para evaluar el rendimiento, las mejoras repercutirían en el usuario final.

Razones de la utilización de benchmarks pequeños

- Portabilidad:** En el pasado lenguajes de programación inconsistentes entre máquinas dificultando el transporte
- Fácil simulación:** Cuando se diseña una nueva máquina
- Estandarización:** Los pequeños benchmarks más fácilmente
- En la actualidad la popularidad de los sistemas operativos estándares (UNIX, WINDOWS...) elimina la principal dificultad.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Colecciones de benchmarks

- ⚙ Medir el rendimiento de los procesadores con una variedad de aplicaciones
- ⚙ Ventajas clave: la debilidad de algún benchmark es minimizada por la presencia de otros
- ⚙ Las colecciones de benchmarks formadas por programas que pueden ser núcleos, pero fundamentalmente programas reales

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Colecciones de benchmarks

- **SPEC:** El grupo System Performance Evaluation Cooperative se formó en 1988 con representantes de diversas compañías (Hewlett-Packard, DEC, MIPS, Sun) que llegaron al acuerdo de ejecutar un conjunto de programas y entradas reales.
- SPEC89, SPEC92, SPEC95, SPEC CPU2000, SPEC CPU2006, SPEC CPU2017.
- Desktop Benchmarks
 - SPEC CPU 2006 (CINT, CFP)
 - SPECviewperf (OpenGL Graphics Library)
 - SPECapc (ProEngineer;Solidworks;Unigraphics...)
- BenchMarks Para Servidores
 - SPECrate (productividad)
 - SPECsfs (Rendimiento sistema ficheros de red)
 - SPECweb (Rendimiento servidores web)
- BenchMarks sistemas embebidos

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Colecciones de benchmarks

- ⚙ **Otras colecciones**
- ⚙ **Business Winstone:** Proceso por lotes Netscape, Corel, WordPerfect, Microsoft...
- ⚙ **CC Winstone:** Aplicaciones creación contenido multimedia (Photoshop, Premiere, edición audio...)
- ⚙ **Winbench:** Procesos por lotes miden rendimiento CPU, video, disco,... orientado a medida por subsistemas.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Formulación de los resultados de la evaluación del rendimiento

- **Reproducibilidad:** enumerar todo lo necesario para repetir los experimentos.
- SPICE tarda 94 segundos en una DECstation 3100
- Este enunciado prescinde de aspectos como:
 - Entradas del programa
 - Versión del programa
 - Versión del compilador
 - Nivel de optimización y código compilado
 - Versión del sistema operativo
 - Cantidad de memoria principal
 - Número y tipos de disco
 - Versión de la CPU

Análisis de
rendimiento

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Formulación de los resultados de la evaluación del rendimiento

• **Informe SPEC:**

- **Descripción hardware**
- **Descripción software**
- **Descripción parámetros compilación**
- **Publicación resultados básicos y optimizados**

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Formulación de los resultados de la evaluación del rendimiento

● Informe SPEC:

● Descripción hardware y software

Hardware

CPU Name: Intel Core i7-965 Extreme Edition
CPU Characteristics: Intel Turbo Boost Technology up to 3.46 GHz
CPU MHz: 3200
FPU: Integrated
CPU(s) enabled: 4 cores, 1 chip, 4 cores/chip, 2 threads/core
CPU(s) orderable: 1 chip
Primary Cache: 32 KB I + 32 KB D on chip per core
Secondary Cache: 256 KB I+D on chip per core
L3 Cache: 8 MB I+D on chip per chip
Other Cache: None
Memory: 12 GB (6 x 2GB Samsung M378B5673DZ1-CF8 DDR3-1066 CL7)
Disk Subsystem: 80 GB Intel X-25M SATA Solid-State Drive
Other Hardware: None

Software

Operating System: Windows Vista Ultimate w/ SP1 (64-bit)
Compiler: Intel C++ Compiler Professional 11.0 for IA32
Build 20080930 Package ID: w_cproc_p_11.0.054
Microsoft Visual Studio 2008 (for libraries)
Auto Parallel: Yes
File System: NTFS
System State: Default
Base Pointers: 32-bit
Peak Pointers: 32-bit
Other Software: None
SmartHeap Library Version 8.1 from
<http://www.microquill.com/>

Análisis de
rendimiento

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Formulación de los resultados de la evaluación del rendimiento

Informe SPEC:

- Descripción hardware y software
- Descripción parámetros compilación

Compiler Invocation

C benchmarks:
icl -Qvc9 -Qc99

C++ benchmarks:
icl -Qvc9

Base Optimization Flags

C benchmarks:
-QxSSE4.2 -Qipo -O3 -Qprec-div- -Qopt-prefetch -Qparallel
-Qpar-runtime-control -Qvec-guard-write /F512000000

C++ benchmarks:
-QxSSE4.2 -Qipo -O3 -Qprec-div- -Qopt-prefetch -Qcxx-features
/F512000000 shlw32m.lib -link /FORCE:MULTIPLE

Peak Optimization Flags

C benchmarks:

400.perlbench: -QxSSE4.2(pass 2) -Qprof_gen(pass 1) -Qprof_use(pass 2)
-Qipo -O3 -Qprec-div- -Qansi-alias -Qopt-prefetch
/F512000000 shlw32m.lib -link /FORCE:MULTIPLE

401.bzip2: -QxSSE4.2(pass 2) -Qprof_gen(pass 1) -Qprof_use(pass 2)
-Qipo -O3 -Qprec-div- -Qopt-prefetch -Qansi-alias
/F512000000

403.gcc: -QxSSE4.2(pass 2) -Qprof_gen(pass 1) -Qprof_use(pass 2)
-Qipo -O3 -Qprec-div- /F512000000

429.mcf: -QxSSE4.2 -Qipo -O3 -Qprec-div- -Qopt-prefetch

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Formulación de los resultados de la evaluación del rendimiento

Informe SPEC:

- Descripción hardware y software
- Descripción parámetros compilación
- Publicación resultados básicos y optimizados

Results Table

Benchmark	Base						Peak					
	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio
400.perlbench	426	22.9	426	22.9	<u>426</u>	<u>22.9</u>	305	32.0	305	32.1	<u>305</u>	<u>32.0</u>
401.bzip2	526	18.3	<u>526</u>	<u>18.3</u>	526	18.3	517	18.7	518	18.6	<u>517</u>	<u>18.7</u>
403.gcc	<u>305</u>	<u>26.4</u>	305	26.4	302	26.7	264	30.5	267	30.1	<u>264</u>	<u>30.5</u>
429.mcf	189	48.3	<u>189</u>	<u>48.3</u>	190	48.0	189	48.2	192	47.5	<u>190</u>	<u>48.0</u>
445.gobmk	444	23.6	444	23.6	<u>444</u>	<u>23.6</u>	<u>396</u>	<u>26.5</u>	395	26.5	396	26.5
456.hmmer	496	18.8	495	18.9	<u>495</u>	<u>18.9</u>	392	23.8	<u>392</u>	<u>23.8</u>	392	23.8
458.sjeng	494	24.5	494	24.5	<u>494</u>	<u>24.5</u>	472	25.6	472	25.6	<u>472</u>	<u>25.6</u>
462.libquantum	99.5	208	<u>99.6</u>	<u>208</u>	99.9	207	99.5	208	<u>99.6</u>	<u>208</u>	99.9	207
464.h264ref	599	37.0	599	36.9	<u>599</u>	<u>37.0</u>	538	41.1	<u>539</u>	<u>41.1</u>	539	41.1
471.omnetpp	255	24.5	<u>254</u>	<u>24.6</u>	254	24.6	218	28.7	<u>218</u>	<u>28.7</u>	217	28.7
473.astar	395	17.8	<u>395</u>	<u>17.8</u>	395	17.8	355	19.8	<u>356</u>	<u>19.7</u>	356	19.7
483.xalancbmk	232	29.8	231	29.9	<u>231</u>	<u>29.8</u>	232	29.8	231	29.9	<u>231</u>	<u>29.8</u>

Análisis de
rendimiento

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Formulación de los resultados de la evaluación del rendimiento

	Computador A	Computador B	Computador C
Programa 1	1	10	20
Programa 2	1000	100	20
Tiempo total	1001	110	40

➤ Afirmaciones:

- A es 900% más rápido que B para el programa 1.
- B es 900% más rápido que A para el programa 2.
- A es 1900% más rápido que C para el programa 1.
- C es 4900% más rápido que A para el programa 2.
- B es 100% más rápido que C para el programa 1.
- C es 400% más rápido que B para el programa 2.
- El contraste de las afirmaciones presenta un cuadro confuso.

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Resúmenes del rendimiento

⚙️ Tiempo total de ejecución

	Computador A	Computador B	Computador C
Programa 1	1	10	20
Programa 2	1000	100	20
Tiempo total	1001	110	40

- ⚙️ B es 810% más rápido que A para los programas 1 y 2.
- ⚙️ C es 2400% más rápido que A para los programas 1 y 2.
- ⚙️ C es 175% más rápido que B para los programas 1 y 2.

⚙️ Tiempo medio de ejecución

$$\frac{1}{n} \sum_{i=1}^n \text{Tiempo}_i$$

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

Análisis de
rendimiento

Resúmenes del rendimiento

⬢ Tiempo de ejecución ponderado

- ⬢ Asignar a cada programa un factor de peso w_i que indique la frecuencia relativa del programa en la carga de trabajo.

$$\sum_{i=1}^n w_i \cdot \text{Tiempo}_i$$

- ⬢ w_i = frecuencia del programa iésimo de la carga de trabajo
- ⬢ **Tiempo_i** = tiempo de ejecución del programa i-ésimo

⬢ Media Geométrica

$$\sqrt[n]{\prod_{i=1}^n \text{Tiempo}_i}$$

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

2.2 Evaluación del rendimiento

Métricas

Benchmarks

Formulación

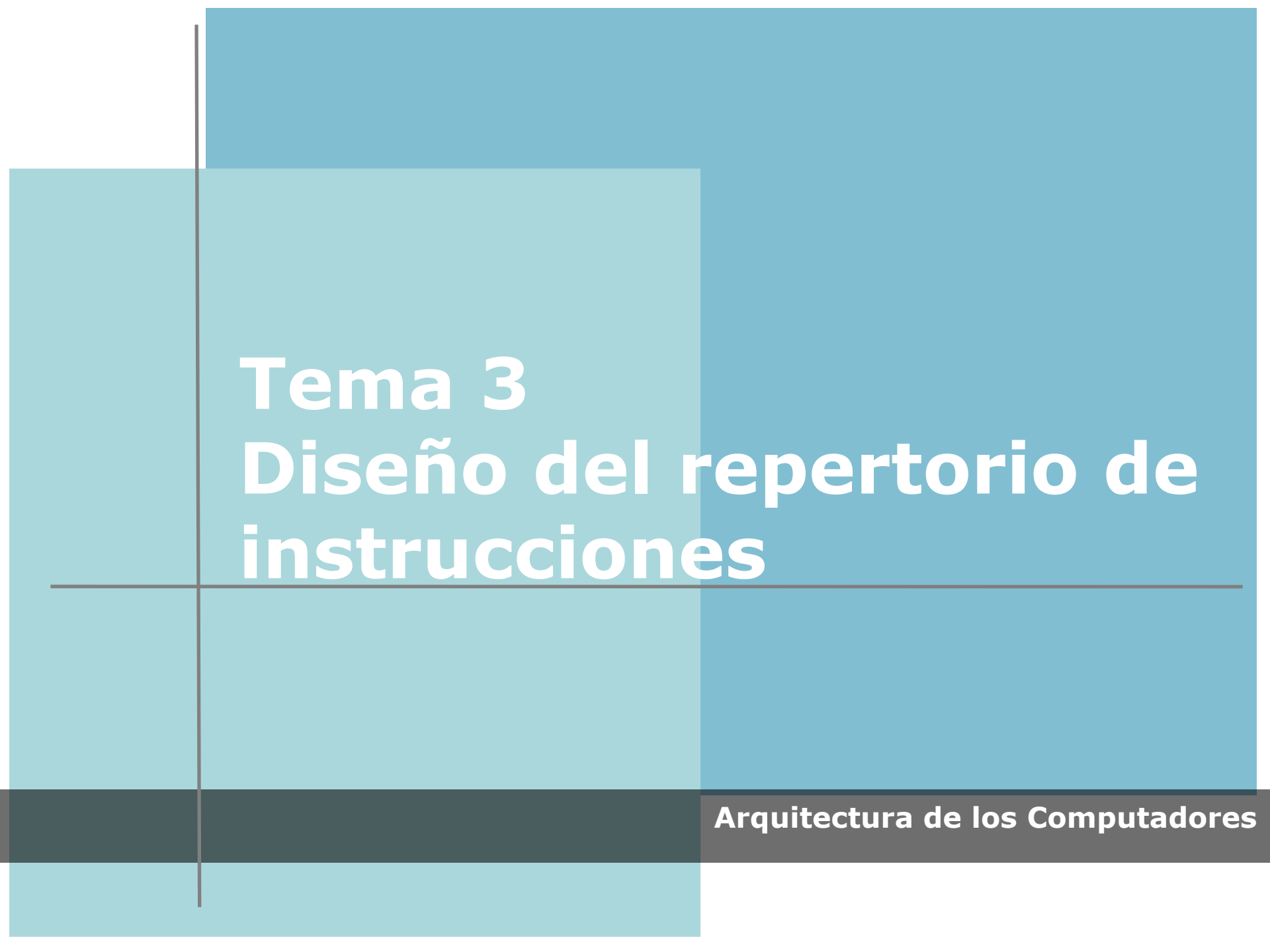
Análisis de
rendimiento

Intel Core i7-965

Benchmark	Base Ref Time	Base Run Time	Base Selected
400.perlbench	9770	425.9	22.9396572
401.bzip2	9650	526.2	18.33903459
403.gcc	8050	305.2	26.37614679
429.mcf	9120	188.8	48.30508475
445.gobmk	10490	443.6	23.64743012
456.hmmer	9330	494.7	18.8599151
458.sjeng	12100	494.2	24.48401457
462.libquantum	20720	99.6	208.0321285
464.h264ref	22130	598.9	36.95107697
471.omnetpp	6250	253.9	24.61599055
473.astar	7020	395.1	17.76765376
483.xalancbmk	6900	231.3	29.83138781

Media Aritmética	10960.83333	371.45	41.67912673	29.5082335
	MA(TR)	MA(TE)	MA(TR/TE)	MA(TR)/MA(TE)

Média Geométrica	10108.21274	334.1591446	30.2496966	30.2496966
	MG(TR)	MG(TE)	MG(TR/TE)	MG(TR)/MG(TE)



Tema 3

Diseño del repertorio de instrucciones

Arquitectura de los Computadores

Tema 3. Diseño del repertorio de instrucciones

Objetivos

- Analizar las arquitecturas desde el nivel de lenguaje máquina, aportando el punto de vista del diseñador de compiladores
- Comprender la influencia que ejercen los lenguajes y los compiladores sobre la arquitectura.
- Reflexionar sobre las ventajas e inconvenientes de los distintos enfoques para abordar el diseño de los repertorios de instrucciones, aportando una taxonomía de éstas.
- Conocer medidas que reflejen el distinto grado de utilización de los repertorios de instrucciones, dependiendo de la aplicación ejecutada.

Tema 3. Diseño del repertorio de instrucciones

Contenido

• 3.1 Introducción

- 3.1.1 Introducción
- 3.1.2 Taxonomía
- 3.1.3 Arquitecturas GPR

• 3.2 Características del repertorio

- 3.2.1 Direccionamiento de la memoria
- 3.2.2 Tipo y tamaño de los operandos
- 3.2.3 Operaciones

• 3.3 Evolución de la programación de los computadores

- 3.3.1 Introducción
- 3.3.2 La arquitectura como objeto del compilador
- 3.3.3 Instrucciones de palabra muy larga (VLIW)

• 3.4 Ejemplos característicos

- 3.4.1 DEC VAX
- 3.4.2 IBM 360/370
- 3.4.3 Intel x86
- 3.4.5 DLX

Tema 3. Diseño del repertorio de instrucciones

Debate inicial

- ¿Por qué no todas las máquinas tienen el mismo repertorio de instrucciones?
- ¿Sería aconsejable esta situación?
- ¿Qué tipo de repertorio es mejor, un repertorio complejo o un repertorio simple?
- ¿En qué influye la forma de programar las máquinas?
- ¿El compilador influye en el rendimiento?

3.1 Introducción

Tema 3. Diseño del repertorio de instrucciones

Arquitectura de computadores

3.1.1 Introducción

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Definición

- **Arquitectura del repertorio de instrucciones (ISA / Instruction Set Architecture):**
 - Se trata de la porción del computador visible por el programador o el diseñador de compiladores

3.1.1 Introducción

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Áreas de aplicación

• Escritorio

- **Énfasis** del rendimiento de los programas con tipos de datos **enteros** y de **punto flotante (FP)**,
- **Escasa preocupación** por el **tamaño** del programa o el **consumo** de energía

• Servidores

- Bases de datos, servidor de archivos, aplicaciones web...
- El rendimiento del **FP** es mucho **menos importante** que el rendimiento para enteros o cadenas de caracteres

• Aplicaciones embebidas

- **Valoran coste y potencia**
- Tamaño del código es importante -> menos memoria -> más barato y menos consumo
- Además, algunas clases de instrucciones (como FP) pueden ser opcionales para reducir costes del chip

3.1.2 Taxonomía de las arquitecturas a nivel ISA

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Almacenamiento de operandos en la CPU

- ◆ GPR, pila, acumulador

◆ Operandos explícitos

- ◆ 0,1,2,3

◆ Posición del operando

- ◆ R-R, R-M, M-M

◆ Operaciones

- ◆ CISC-RISC

◆ Tipo y tamaño de los operandos

- ◆ Enteros, PF, decimales, caracteres, cadenas...

3.1.2 Taxonomía de las arquitecturas a nivel ISA

Introducción

Características

Programación

Ejemplos

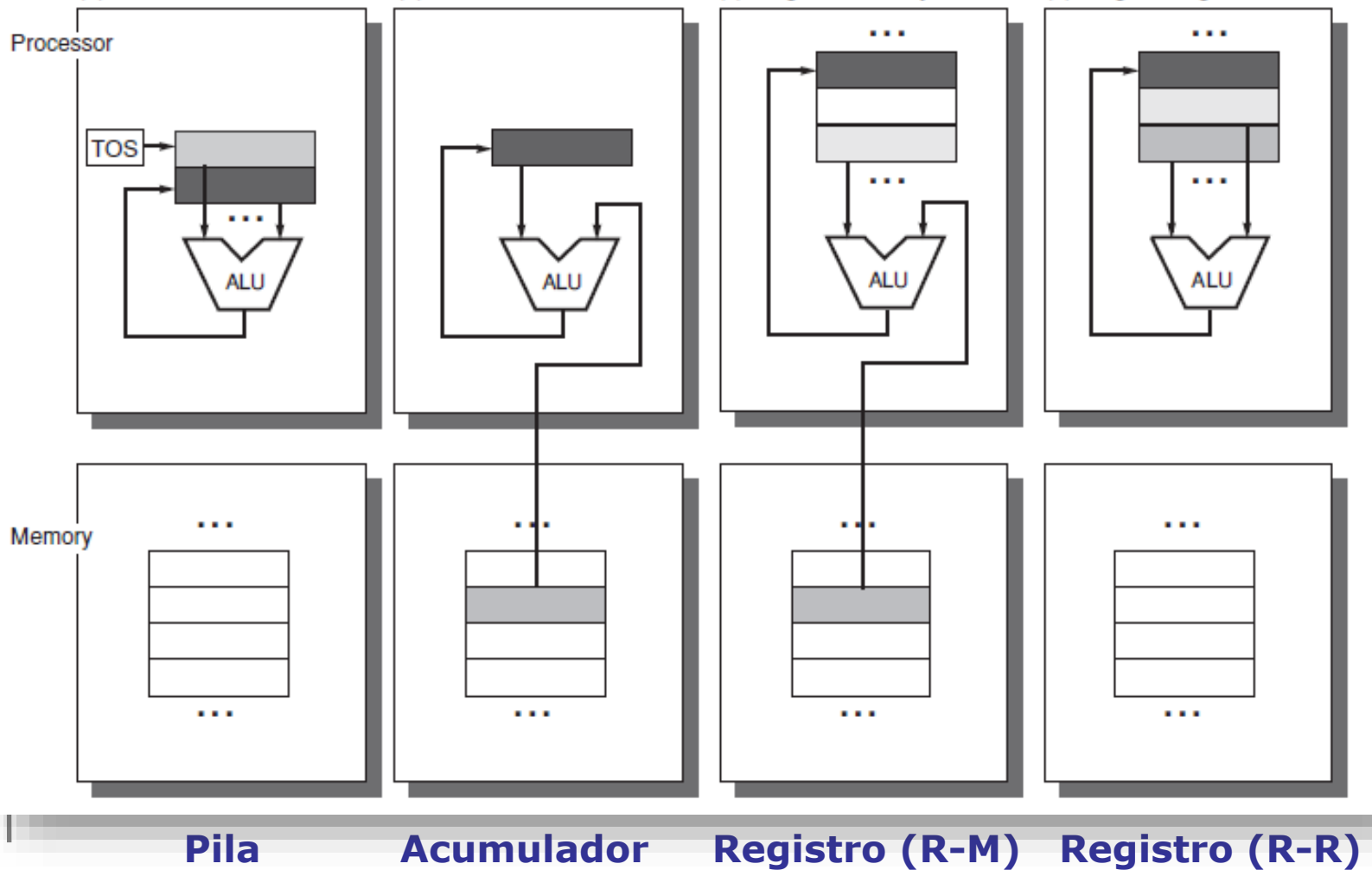
Diseño del
repertorio de
instrucciones

Tipo de almacenamiento interno de la CPU

- El tipo de almacenamiento interno es la **diferenciación más básica**
- Arquitectura de pila:** Los operandos están implícitamente en la cima de la pila
- Arquitectura de acumulador:** Un operando está implícitamente en el acumulador
- Arquitectura de registros de propósito general (GPR):** Tienen sólo operandos explícitos en registros o en posiciones de memoria

3.1.2 Taxonomía de las arquitecturas a nivel ISA

Tipo de almacenamiento interno de la CPU



Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

3.1.2 Taxonomía de las arquitecturas a nivel ISA

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Tipo de almacenamiento interno de la CPU

- **Cuatro ejemplos:** Secuencia de código $C=A+B$ en las cuatro clases de repertorios de instrucciones

Pila	Acumulador	Registro (R-M)	Registro (R-R)
Push A	Load A	Load R1, A	Load R1,A
Push B	Add B	Add R1, B	Load R2,B
Add	Store C	Store C, R1	Add R3,R1,R2
Pop C			Store R3,C

3.1.2 Taxonomía de las arquitecturas a nivel ISA

Introducción

Características

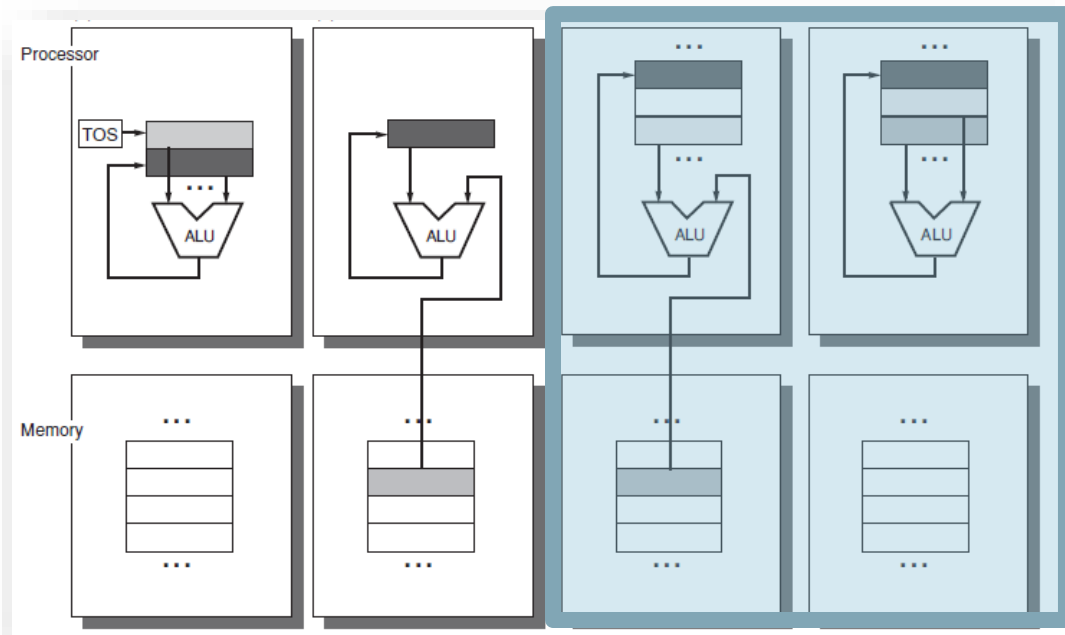
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Tendencia actual GPR

- ◆ **Máquinas** más **antiguas** arquitecturas **pila y acumulador**
- ◆ **A partir de 1980** frecuentemente arquitecturas **GPR**
 - ◆ Los registros tienen acceso más rápido que la memoria
 - ◆ Los registros son más fáciles de utilizar por los compiladores y de manera más efectiva



3.1.3 Arquitecturas GPR

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ **Ventaja: utilización efectiva de registros por el compilador**

- ◆ **Ubicación de variables:** reduce el tráfico de memoria y acelera el programa (los registros son más rápidos que la memoria) (Ej. Bucle del algoritmo de la burbuja)
- ◆ **Evaluar expresiones:** Los registros permiten una ordenación más flexible que las pilas o acumuladores (almacenamiento temporal subexpresiones)
- ◆ **Densidad de código:** Un registro se nombra con menos bits que una posición de memoria
- ◆ **Registros no reservados:** Los escritores de compiladores prefieren que los registros sean no reservados para ubicar las variables de forma más flexible (Ej. Registro EBX en x86)

3.1.3 Arquitecturas GPR

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

- **Número de registros necesario:** depende del uso del compilador reservando registros para:
 - Evaluar expresiones
 - Paso de parámetros
 - Ubicar variables. Según algoritmo de ubicación utilizado

- Appendix A. Pg. 525-531 Hennessy 5th ed.

How many registers are sufficient? The answer, of course, depends on the effectiveness of the compiler. Most compilers reserve some registers for expression evaluation, use some for parameter passing, and allow the remainder to be allocated to hold variables. Modern compiler technology and its ability to effectively use larger numbers of registers has led to an increase in register counts in more recent architectures.

- MIPS 32 Int 32 FP
- <https://en.wikipedia.org/wiki/AVX-512>
- [AMD Ryzen](#) 160 RFP y 168 RInt

3.1.3 Arquitecturas GPR

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Clasificación de las arquitecturas GPR

- **Número de operandos** de instrucciones ALU
- **Número de operandos** que se pueden direccionar **en memoria** en instrucciones ALU. (0..3)

3.1.3 Arquitecturas GPR

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Clasificación de las arquitecturas GPR

- **Número de operandos** de instrucciones ALU
 - **Tres operandos:** Un resultado y dos fuentes
 - ADD R1,R2,R3
 - **Dos operandos:** Un operando es fuente y destino
 - ADD R1,R2

3.1.3 Arquitecturas GPR

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Clasificación de las arquitecturas GPR

- **Número de operandos** de instrucciones ALU
- **Número de operandos** que se pueden direccionar **en memoria** en instrucciones ALU. (0..3)
 - **Registro-registro (carga almacenamiento):** Sin referencia a memoria para instrucciones ALU. **Solo** registros de la CPU
 - ADD R1,R2
 - **Registro-memoria:** Se permite un solo operando referenciando la memoria.
 - ADD R1,MEM
 - **Memoria-memoria:** Se permite más de un operando referenciando la memoria. (2 o 3)
 - ADD MEM1,MEM2

3.1.3 Arquitecturas GPR

Introducción




Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Ventajas y desventajas de las arquitecturas GPR

Tipo	Ventajas	Desventajas
R-R	<ul style="list-style-type: none">•Codificación simple, instrucciones de longitud fija.	<ul style="list-style-type: none">•Mayor recuento de instrucciones que las arquitecturas con referencias a memoria.
<div> Impacto sobre el compilador y la implementación.</div> <div> Número de instrucciones</div> <div> Codificación de instrucciones</div>		
M-M	<ul style="list-style-type: none">•No se emplean registros para temporales.•Código más compacto.	<ul style="list-style-type: none">•Gran variación en el tamaño de las instrucciones.•Gran variación en el trabajo por instrucción.•Los accesos a memoria crean cuellos de botella en memoria.

3.1.3 Arquitecturas GPR

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Ventajas y desventajas de las arquitecturas GPR

Tipo	Ventajas	Desventajas
R-R	<ul style="list-style-type: none">•Codificación simple, instrucciones de longitud fija.•Las instrucciones emplean números de ciclos similares para ejecutarse.	<ul style="list-style-type: none">•Mayor recuento de instrucciones que las arquitecturas con referencias a memoria.
R-M	<ul style="list-style-type: none">•Los datos pueden ser accedidos sin cargarlos primero.	<ul style="list-style-type: none">•Se destruye un operando fuente.•Codificar un número de registro y una dirección de memoria en cada instrucción puede restringir el número de registros.•Los ciclos de instrucción varían según los operandos
M-M	<ul style="list-style-type: none">•No se emplean registros para temporales.•Código más compacto.	<ul style="list-style-type: none">•Gran variación en el tamaño de las instrucciones.•Gran variación en el trabajo por instrucción.•Los accesos a memoria crean cuellos de botella en memoria.

3.2 Características

Tema 3. Diseño del repertorio de instrucciones

Arquitectura de computadores

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

❖ **3.2 Características del repertorio**

- ❖ 3.2.1 Direccionamiento de la memoria
- ❖ 3.2.2 Tipo y tamaño de los operandos
- ❖ 3.2.3 Operaciones

3.2.1 Direcccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

- Arquitecturas direccionan por bytes y proporcionan acceso a bytes (8 bits), medias palabras (16 bits), palabras (32 bits) y dobles palabras (64 bits)

• a. Ordenación de los bytes

- Dos convenios para ordenar los bytes de una palabra: **"Little Endian" y "Big Endian"**
- Estos términos provienen de un famoso **artículo de Cohen[1981]** que establece una **analogía** entre la discusión sobre por que extremo de byte comenzar y la discusión **de los Viajes de Gulliver sobre que extremo del huevo abrir**

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

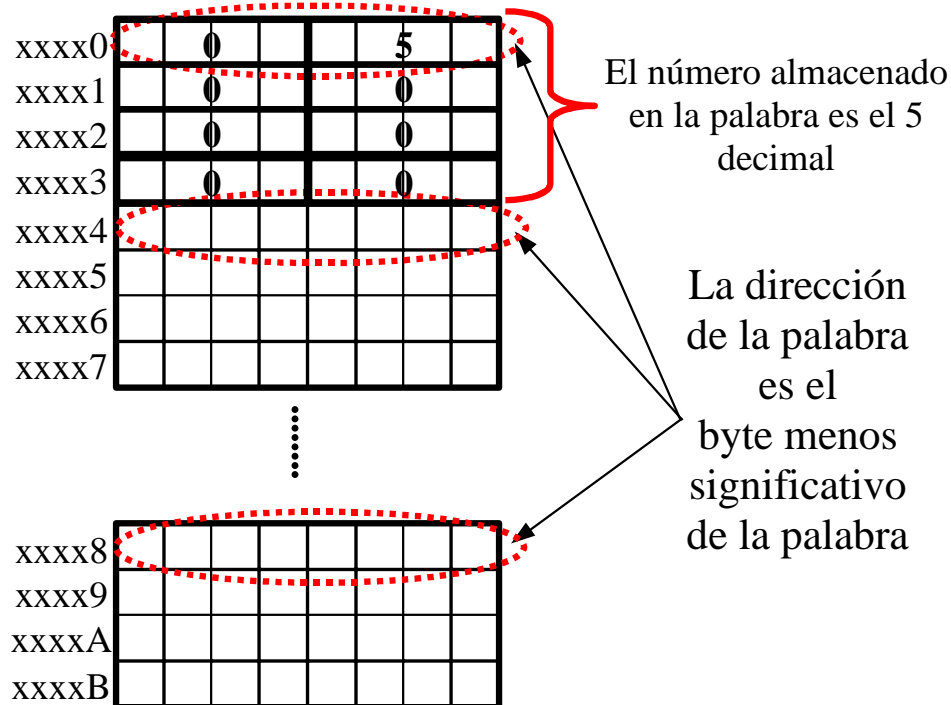
Ejemplos

Diseño del
repertorio de
instrucciones

• a. Ordenación de los bytes

• La ordenación **Little endian** (extremo pequeño)

- La **dirección** de un dato es la del **byte menos significativo**
- DEC PDP11, VAX y 80x86 siguen el modelo Little endian



3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

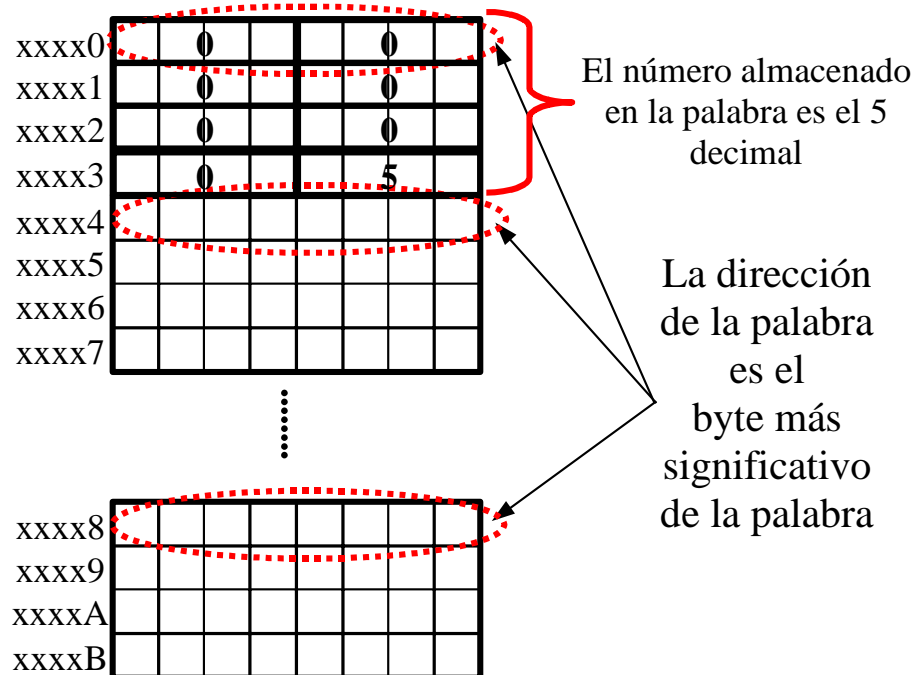
Ejemplos

Diseño del
repertorio de
instrucciones

■ a. Ordenación de los bytes

■ La ordenación **Big endian** (extremo grande)

- La **dirección** de un dato es la del **byte más significativo**
- IBM 360/370, los Motorola 680x0 siguen el modelo Big endian



3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

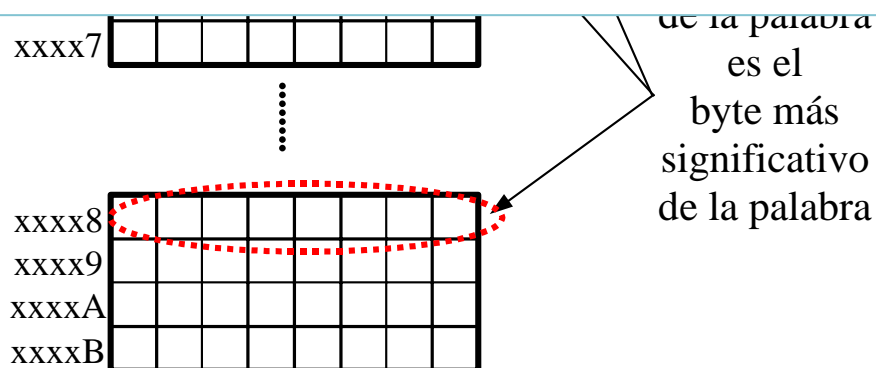
Diseño del
repertorio de
instrucciones

• a. Ordenación de los bytes

• La ordenación **Big endian** (extremo grande)

• La **dirección** de un dato es la del **byte más significativo**

• **La ordenación de los bytes puede ser problema cuando se intercambian datos entre máquinas con diferentes ordenaciones**



3.2.1 Direccionamiento de la memoria

Introducción

Características

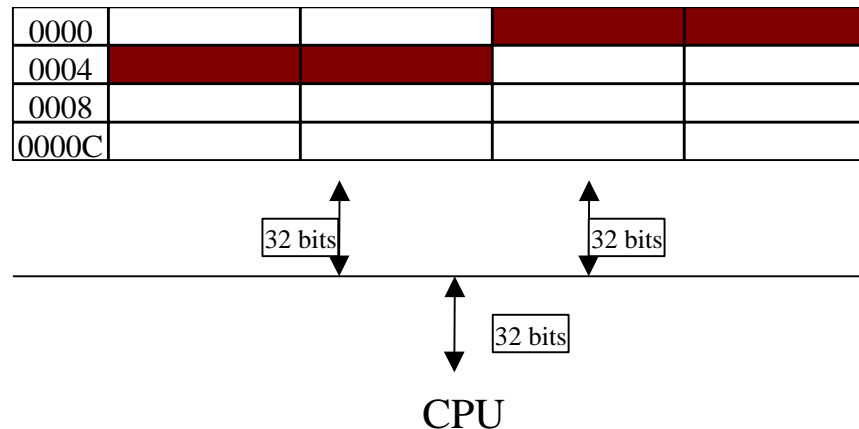
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Alineamiento de los accesos a los objetos de memoria

- Un acceso a un objeto mayor de un byte en la **dirección A** y en una memoria de tamaño **n** bytes (**ancho** de palabra) en su **bus de datos**, esta **alineado**, si la dirección **$A \bmod n = 0$**
- El **acceso no alineado** a los datos **puede empeorar el tiempo** de ejecución del programa debido a la necesidad de realizar varios accesos a memoria para completar un acceso.
- Ejemplo:** Que ocurre en un sistema con un bus de datos de 32 bits al acceder a una palabra no alineada.



3.2.1 Direccionamiento de la memoria

Introducción

Características

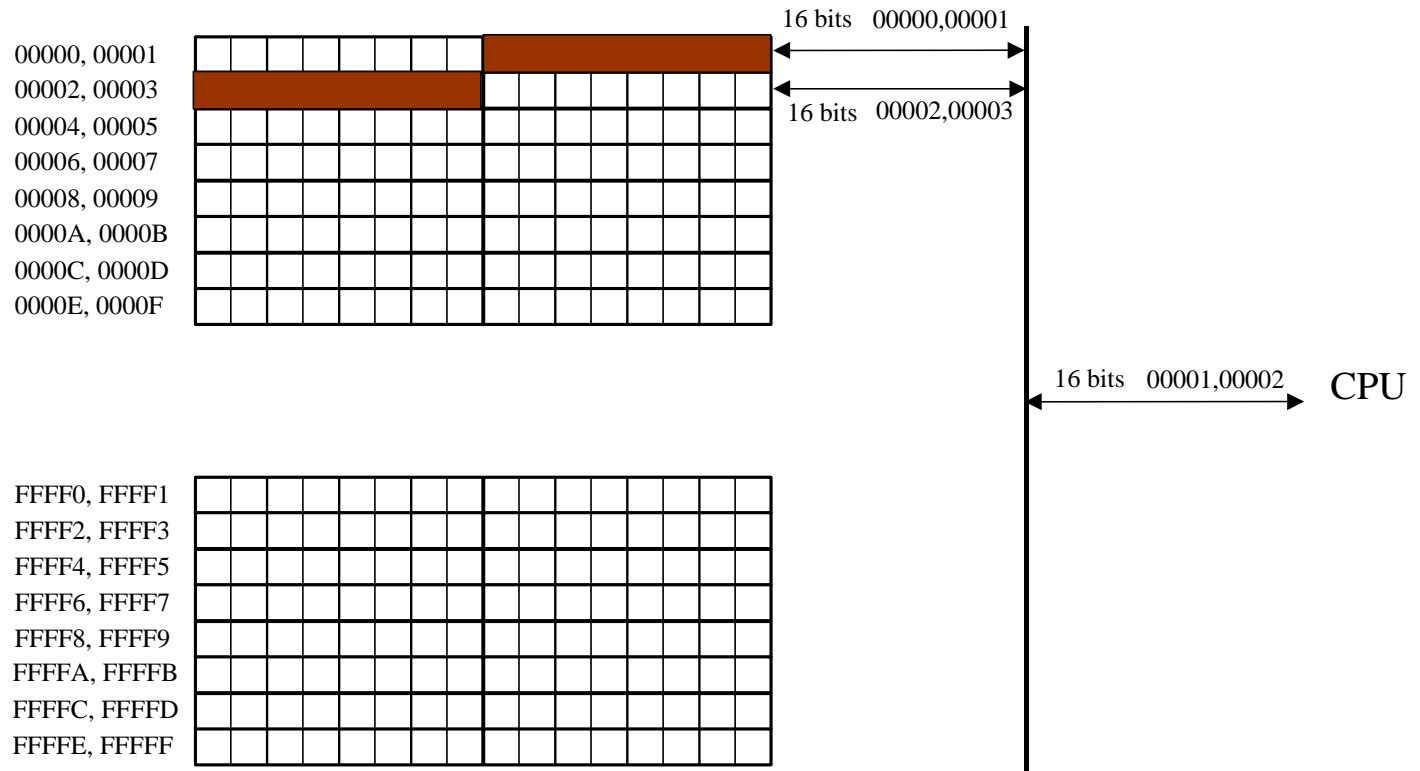
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Alineamiento de los accesos a los objetos de memoria

- **Ejemplo:** Que ocurre en el 80x86 cuando se realiza un acceso a una palabra no alineada (sistema con un bus de 16 bits a memoria)



3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

❖ b. Modos de direccionamiento

- ❖ **Modos de direccionamiento:** forma en que las arquitecturas especifican la dirección de un objeto
- ❖ En las arquitecturas GPR **un modo de direccionamiento puede especificar:**
 - ❖ **Constante, registros o posiciones de memoria**
- ❖ En caso de ser una posición de memoria, la dirección real especificada por el modo de direccionamiento se denomina **dirección efectiva**.

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

b. Modos de direccionamiento

-  Los nombres de los modos de direccionamiento de la tabla pueden diferir entre arquitecturas

Modo de direccionamiento	Ejemplo	Significado	Cuando se usa
Registro	Add R4, R3	$R4 \leftarrow R4 + R3$	Cuando un valor está en un registro
Inmediato o literal	Add R4, #3	$R4 \leftarrow R4 + 3$	Para constantes. En algunas máquinas, literal e inmediato son dos modos diferentes de direccionamiento
Desplazamiento	Add R4, 100(R1)	$R4 \leftarrow R4 + M[100 + R1]$	Acceso a variables locales
Registro diferido o indirecto	Add R4, (R1)	$R4 \leftarrow R4 + M[R1]$	Acceso utilizando un puntero o una dirección calculada
Indexado	Add R3, (R1+R2)	$R3 \leftarrow R3 + M[R1 + R2]$	A veces útil en direccionamiento de arrays- R1 base del array; R2 índice.
Directo o absoluto	Add R1, (1001)	$R1 \leftarrow R1 + M[1001]$	A veces útil para acceder a datos estáticos; la constante que especifica la dirección puede necesitar ser grande

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

b. Modos de direccionamiento

-  Los nombres de los modos de direccionamiento de la tabla pueden diferir entre arquitecturas

Modo de direccionamiento	Ejemplo	Significado	Cuando se usa
Indirecto o diferido de memoria	Add R1, @(R3)	$R1 \leftarrow R1 + M[M[R3]]$	Si R3 es la dirección de un puntero p, entonces el modo obtiene *p
Autoincremento	Add R1, (R2)+	$R1 \leftarrow R1 + M[R2]$ $R2 \leftarrow R2 + d$	Util para recorridos de arrays en un bucle. R2 apunta al principio del array; cada referencia incrementa R2 en el tamaño de un elemento, d.
Autodecremento	Add R1, -(R2)	$R2 \leftarrow R2 - d$ $R1 \leftarrow R1 + M[R2]$	El mismo uso que autoincremento. Autoincremento/decremento también puede utilizarse para realizar una pila mediante introducir y sacar (push y pop)
Escalado o índice	Add R1, 100 (R2)[R3]	$R1 \leftarrow R1 + M[100 + R2 + R3 * d]$	Usado para acceder a arrays por índice. Puede aplicarse a cualquier modo de direccionamiento indexado en algunas máquinas.

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

b. Modos de direccionamiento (ejemplos)

Dir.	Valor	Registros
1000	1012	R1=1012
1004	4	R2=1004
1008	8	R3=2
1012	1004	d = 4 (palabras de 32 bits)
1016	1020	
1020	16	
1024	0	

	Mod dir	efectiva	→	valor
Desp	8(R1):	1020	→	16
Autoinc	(R1)+:	1012	→	1004 (después R1 sería igual a 1016)
Indir	@(R1):	1004	→	4
Esca	4(R2)[R3]:	1016	→	1020

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

❖ **b. Modos de direccionamiento**

- ❖ Los **modos de direccionamiento reducen el RI** pero **complican la implementación** pudiendo incrementar el CPI medio
- ❖ El arquitecto de computadores debe **elegir** que **modos de direccionamiento** incluir **en base a estudios de frecuencia de utilización**

3.2.1 Direccionamiento de la memoria

Introducción

Características

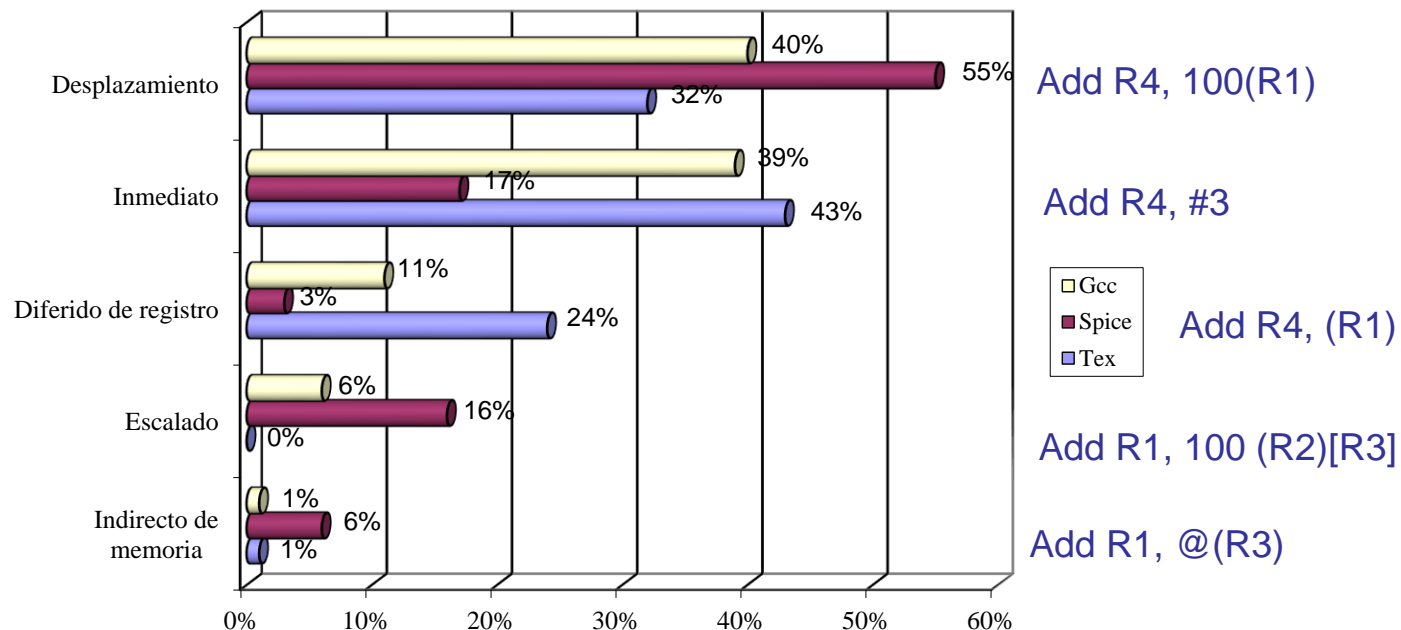
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

■ b. Modos de direccionamiento

- Frecuencia de utilización de los modos de direccionamiento. SPEC (gcc, spice, Tex) en el VAX. Medidas independientes de arquitectura
- El **direccionamiento inmediato y desplazamiento dominan** la utilización de los modos de direccionamiento



3.2.1 Direccionamiento de la memoria

Introducción

Características

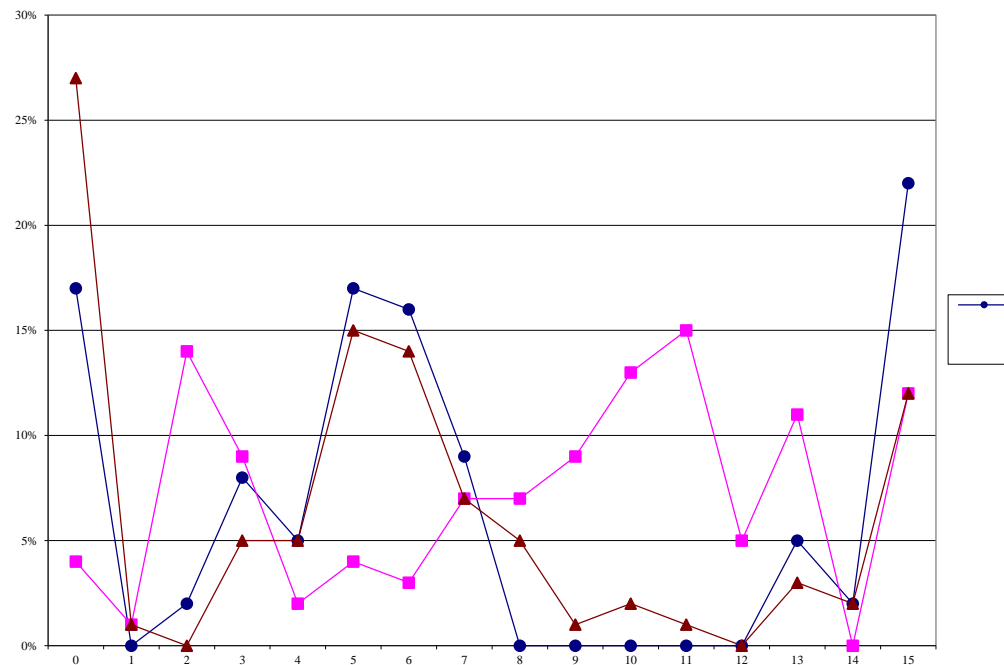
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Modo de direccionamiento desplazamiento

- ¿Cuál es el **rango más frecuente de desplazamientos** en este modo de direccionamiento?
- La respuesta indicará que tamaño soportar** (afecta a la longitud de la instrucción)



3.2.1 Direccionamiento de la memoria

Introducción

Características

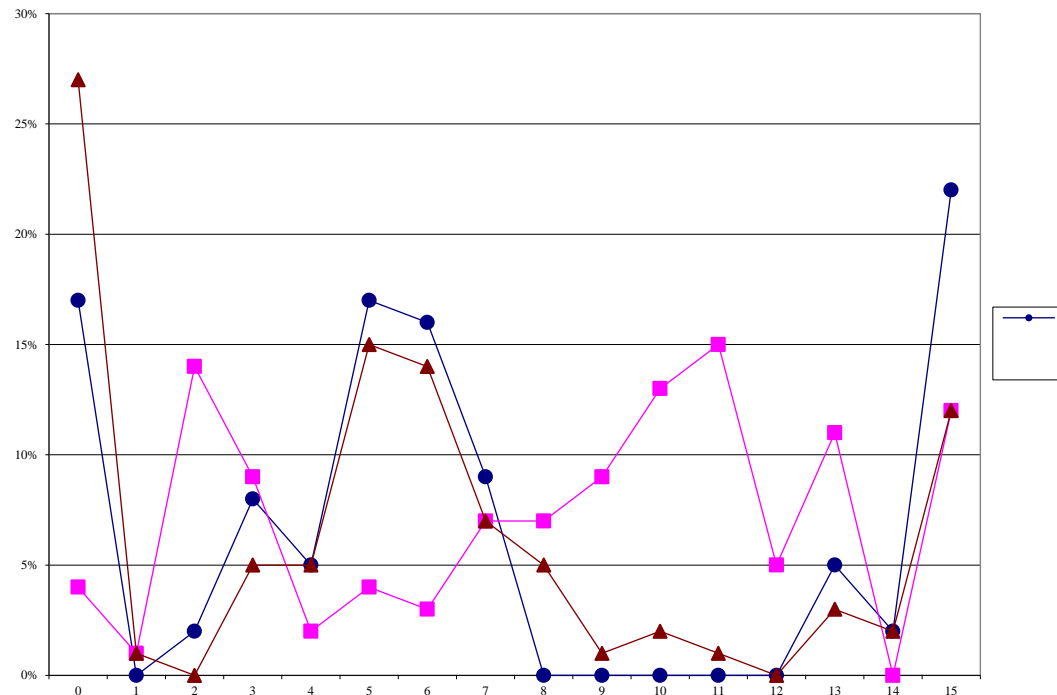
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

❖ c. Modo de direccionamiento desplazamiento

- ❖ Los **desplazamientos** están **ampliamente distribuidos**
- ❖ eje x **\log_2 desplazamiento (tamaño campo desplazamiento)**
- ❖ VAX (8,16,32); IBM360 (12) ; DLX (16); 80x86 (8,16).



3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

❖ d. Modo de direccionamiento literal o inmediato

❖ Los **inmediatos se utilizan** frecuentemente en:

- ❖ **Operaciones aritméticas** (ADD R1,R2,#3)
- ❖ **Comparaciones** (principalmente para saltos) (CMP R1,#0)
- ❖ **Transferencias** para poner una constante en un registro. (MOV R1,#1)
 - ❖ **Constantes** escritas en el código que tienden a ser pequeñas
 - ❖ Constantes de **direcciones** que pueden ser grandes

❖ Dos cuestiones:

- ❖ ¿**Que operaciones** necesitan **soportar inmediatos**?
- ❖ ¿**Qué rango** de valores es necesario para los inmediatos?

3.2.1 Direccionamiento de la memoria

Introducción

Características

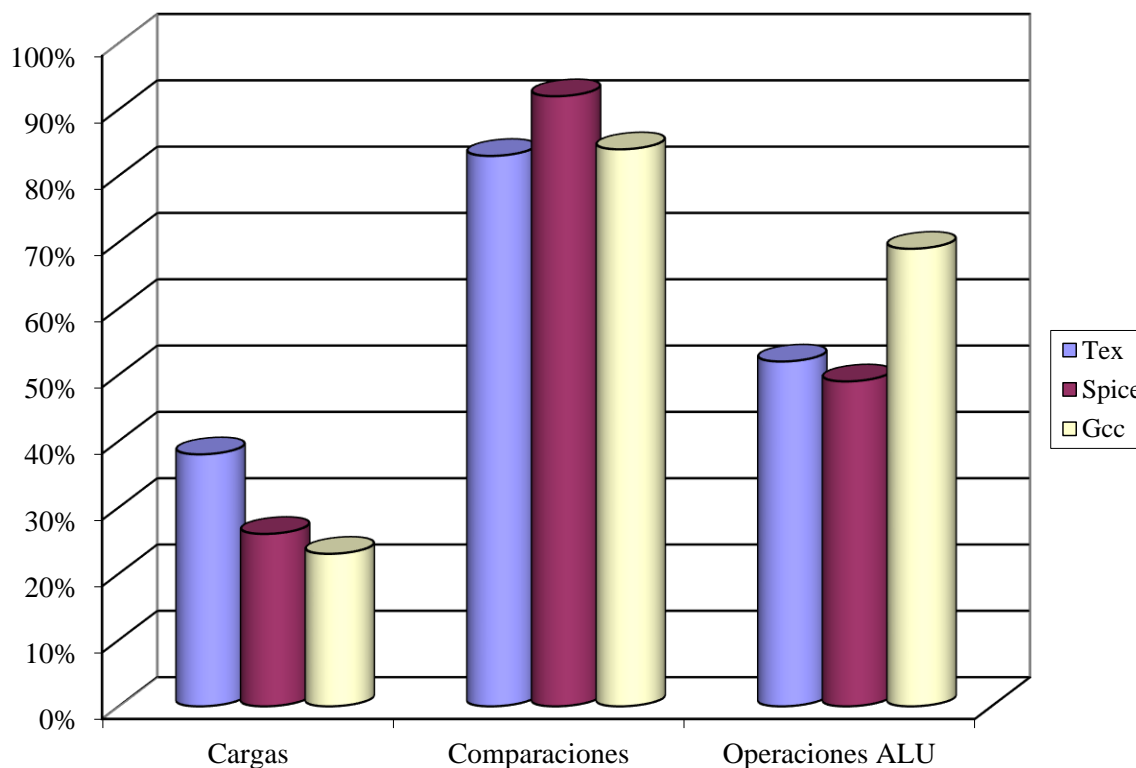
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

d. Modo de direccionamiento literal o inmediato

¿Que operaciones necesitan soportar inmediatos?



3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• d. Modo de direccionamiento literal o inmediato

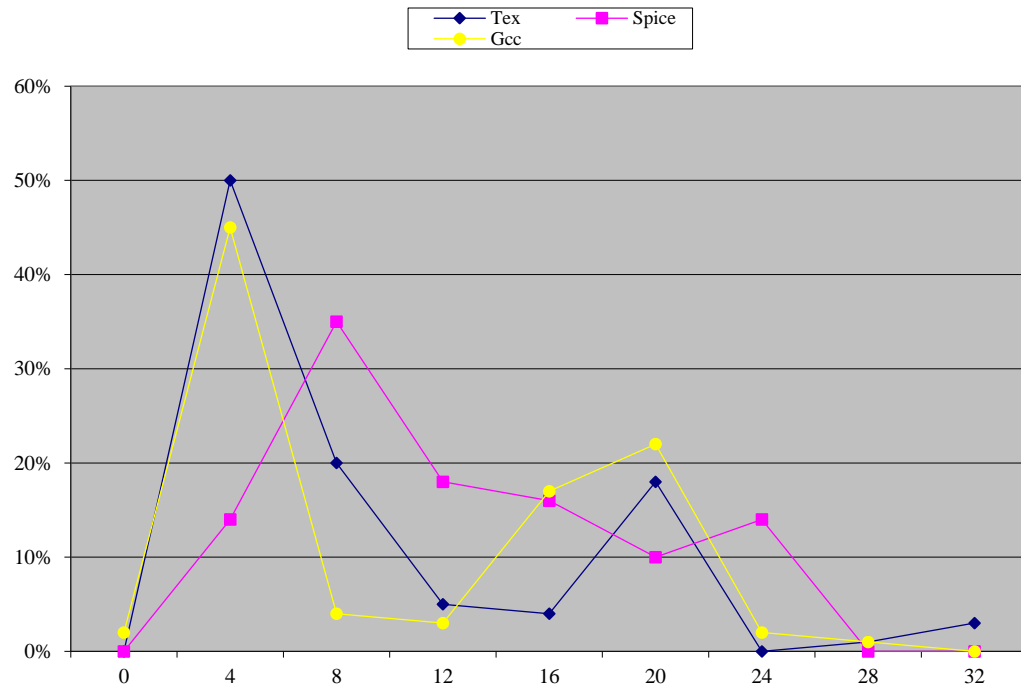
- ¿Qué **rango** de valores es necesario para los inmediatos?
- El **tamaño** de los inmediatos **afecta a la longitud de la instrucción**
- Distribución de valores inmediatos: Los **inmediatos pequeños** son los **más utilizados**, aunque se usan inmediatos grandes en el cálculo de direcciones.

• **VAX (8,16,32)**

• **IBM360 (8)**

• **DLX (16)**

• **80x86 (8,16)**



3.2.1 Direcccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Resumen modos: Appendix A. Pg. 536 Hennessy 5th ed.

1.- Because of their popularity, we would expect a new architecture to support **at least** the following addressing modes: **displacement, immediate, and register indirect**. They represent 75% to 99%

2.- We would expect the **size of the address for displacement mode** to be at least **12 to 16 bits**, since these sizes would capture 75% to 99% of the displacements

3.- We would expect the **size of the immediate** field to be at least **8 to 16 bits**.

Diseño del
repertorio de
instrucciones

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

❖ e. Codificación de los modos de direccionamiento

- ❖ **Codificación incluida en el código de operación:** Para un pequeño número de combinaciones modo de direccionamiento/código de operación, el modo de direccionamiento puede codificarse en el código de operación
- ❖ **Especificador de direcciones separado para cada operando:** En muchas ocasiones se necesita este especificador para indicar el modo de direccionamiento que esta usando cada operando

❖ El arquitecto debe equilibrar

- ❖ El **interés** de **disponer** del **mayor número posible de registros y modos de direccionamiento**
- ❖ El **impacto** del **tamaño de los campos de los registros y de los modos de direccionamiento en el tamaño medio de la instrucción**
- ❖ El **interés** de **tener instrucciones codificadas en longitudes fáciles de manejar** e implementar

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

e. Codificación de los modos de direccionamiento

a) Variable (VAX, Intel 80x86)

Operación y nº de operandos	Especificador de dirección 1	Campo de dirección 1
--------------------------------	---------------------------------	-------------------------

.....

Especificador de dirección 1	Campo de dirección 1
---------------------------------	-------------------------

b) Fijo (Alpha, ARM, MIPS, PowerPC, SPARC)

Operación	Campo de dirección 1	Campo de dirección 2	Campo de dirección 3
-----------	-------------------------	-------------------------	-------------------------

c) Híbrido (IBM 360,370)

Operación	Especificador de dirección	Campo de dirección
-----------	-------------------------------	-----------------------

Operación	Especificador de dirección	Campo de dirección 1	Campo de dirección 2
-----------	-------------------------------	-------------------------	-------------------------

- **Variable:** cualquier modo de direccionamiento con cualquier operador.
- Interesante con número alto de modos de direccionamiento y operaciones. Consigue menor RI pero las instrucciones individuales varían en talla y cantidad de trabajo. Ejemplo VAX.

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

e. Codificación de los modos de direccionamiento

a) Variable (VAX, Intel 80x86)

Operación y nº de operandos	Especificador de dirección 1	Campo de dirección 1
--------------------------------	---------------------------------	-------------------------

.....

Especificador de dirección 1	Campo de dirección 1
---------------------------------	-------------------------

b) Fijo (Alpha, ARM, MIPS, PowerPC, SPARC)

Operación	Campo de dirección 1	Campo de dirección 2	Campo de dirección 3
-----------	-------------------------	-------------------------	-------------------------

c) Híbrido (IBM 360,370)

Operación	Especificador de dirección	Campo de dirección
-----------	-------------------------------	-----------------------

Operación	Especificador de dirección	Campo de dirección 1	Campo de dirección 2
-----------	-------------------------------	-------------------------	-------------------------

- **Fija:** Combina la operación y el modo de direccionamiento en el código de operación.
- Tamaño único para todas las instrucciones. Interesante con número reducido de modos de direccionamiento y operaciones. Fáciles de decodificar e implementar pero conducen a RI altos. Ejemplo MIPS.

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

e. Codificación de los modos de direccionamiento

a) Variable (VAX, Intel 80x86)

Operación y nº de operandos	Especificador de dirección 1	Campo de dirección 1
--------------------------------	---------------------------------	-------------------------

.....

Especificador de dirección n	Campo de dirección n
---------------------------------	-------------------------

b) Fijo (Alpha, ARM, MIPS, PowerPC, SPARC)

Operación	Campo de dirección 1	Campo de dirección 2	Campo de dirección 3
-----------	-------------------------	-------------------------	-------------------------

c) Híbrido (IBM 360,370)

Operación	Especificador de dirección	Campo de dirección
-----------	-------------------------------	-----------------------

Operación	Especificador de dirección	Campo de dirección 1	Campo de dirección 2
-----------	-------------------------------	-------------------------	-------------------------

- **Híbrida:** Esta alternativa reduce la variabilidad en talla y trabajo proporcionando varias longitudes de instrucción.
- Es una alternativa intermedia que persigue las ventajas de las anteriores: reducir recuento de instrucciones y formato sencillo de fácil implementación. Ejemplo IBM 360.

3.2.2 Tipo y tamaño de los operandos

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Forma de designar el tipo de operando. Dos alternativas:

◆ **En el código de operación**

- ◆ El tipo de operando se expresa en el código de operación. Es el método utilizado con más frecuencia

◆ **Datos identificados o autodefinidos**

- ◆ El dato se anota con identificadores que especifican el tipo de cada operando y que son interpretados por el hardware
- ◆ Son extremadamente raras. Arquitecturas de Burroughs. Symbolics para implementaciones LISP

3.2.2 Tipo y tamaño de los operandos

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Tamaños más comunes de los operandos:

- **Byte** (8 bits)
- **Media palabra** (16 bits)
- **Palabra** (32 bits)
- **Doble palabra** (64 bits)

3.2.2 Tipo y tamaño de los operandos

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Codificaciones más comunes de los operandos:

◆ Caracteres

- ◆ **EBCDIC**: usado en las arquitecturas de grandes computadores IBM
- ◆ **ASCII**: (128 ASCII estandar y 256 ASCII extendido). Muy difundido
- ◆ **16-bit Unicode**: usado en Java → gana popularidad

◆ **Enteros**: Representación en **complemento a 2** muy difundida

◆ **Punto flotante: 754 de IEEE** (estándar más difundido)

- ◆ **Precisión simple**: 32 bits (1+8+23 signo, exponente, mantisa).
- ◆ **Precisión doble**: 64 bits (1+11+52 signo, exponente, mantisa).
- ◆ **Precisión simple extendida**
- ◆ **Precisión doble extendida**
- ◆ **Formatos extendidos**: para evitar errores y desbordamientos en operaciones intermedias aumentando el número de bits de mantisa y exponente. Dependen de implementaciones

3.2.2 Tipo y tamaño de los operandos

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Codificaciones más comunes de los operandos:

- **Cadenas de caracteres:** Algunas arquitecturas soportan operaciones sobre **cadenas de caracteres ASCII** (comparaciones, desplazamientos...)
- **Decimales:** Algunas arquitecturas soportan un formato denominado habitualmente **decimal empaquetado (BCD)**. Se utilizan 4 bits para codificar los valores 0-9, y en cada byte se empaquetan dos dígitos decimales

For business applications, some architectures support a decimal format, usually called *packed decimal* or *binary-coded decimal*—4 bits are used to encode the values 0 to 9, and 2 decimal digits are packed into each byte. Numeric character strings are sometimes called *unpacked decimal*, and operations—called *packing* and *unpacking*—are usually provided for converting back and forth between them.

One reason to use decimal operands is to get results that exactly match decimal numbers, as some decimal fractions do not have an exact representation in binary. For example, 0.10_{10} is a simple fraction in decimal, but in binary it requires an infinite set of repeating digits: $0.0001100110011..._2$. Thus, calculations that are exact in decimal can be close but inexact in binary, which can be a problem for financial transactions. (See Appendix J to learn more about precise arithmetic.)

3.2.2 Tipo y tamaño de los operandos

Introducción

Características

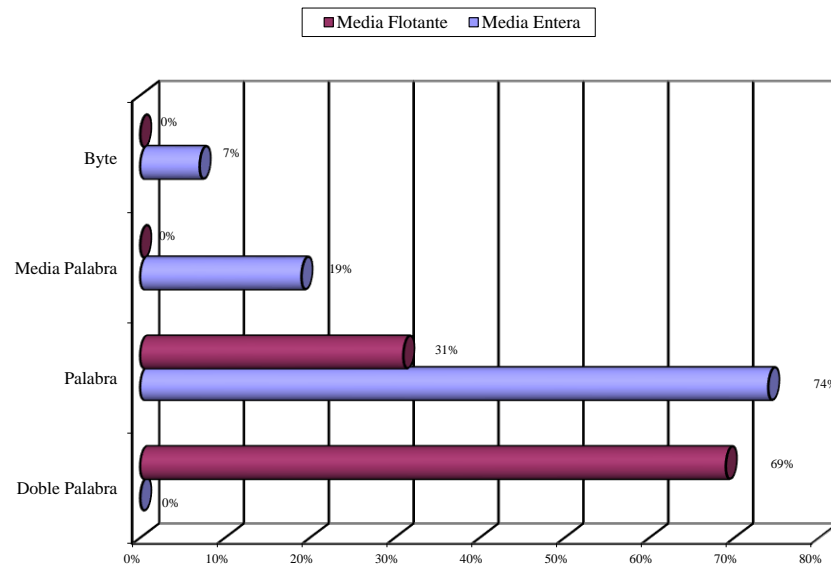
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Distribución de los accesos a los datos por tamaños:

- Los accesos a los tipos principales de datos (palabra y doble palabra) dominan claramente
- Predominio de operandos enteros de 32 bits y operandos en coma flotante de 64 bits (IEEE 754).



3.2.3 Operaciones

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

a. Tipos de operaciones

Tipo de operación	Ejemplo
Aritmético y lógico	Operaciones lógicas y aritméticas enteras: suma, and, resta, or...
Transferencias de datos	Cargas y almacenamientos
Control	Salto, bifurcación, llamada y retorno de procedimiento, traps
Sistema	Llamada al sistema operativo, instrucciones de gestión de memoria virtual
Punto flotante	Operaciones de punto flotante: suma, multiplicación
Decimal	Suma, multiplicación decimal, conversiones de decimal a caracteres
Cadenas	Transferencia, comparación de cadenas, búsqueda de cadenas
Gráficos	Operaciones sobre pixels, operaciones de compresión descompresión

- Tres primeras categorías. Todas las máquinas proporcionan un repertorio completo de este tipo de operaciones
- Funciones del sistema: El soporte varía entre arquitecturas
- Punto flotante: frecuente incluso en repertorios reducidos

3.2.3 Operaciones

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

a. Tipos de operaciones

Tipo de operación	Ejemplo
Aritmético y lógico	Operaciones lógicas y aritméticas enteras: suma, and, resta, or...
Transferencias de datos	Cargas y almacenamientos
Control	Salto, bifurcación, llamada y retorno de procedimiento, traps
Sistema	Llamada al sistema operativo, instrucciones de gestión de memoria virtual
Punto flotante	Operaciones de punto flotante: suma, multiplicación
Decimal	Suma, multiplicación decimal, conversiones de decimal a caracteres
Cadenas	Transferencia, comparación de cadenas, búsqueda de cadenas
Gráficos	Operaciones sobre pixels, operaciones de compresión descompresión

- ⚙️ Tres últimas categorías pueden no estar presentes en algunas arquitecturas. Las arquitecturas de repertorio extenso CISC pueden contener un amplio repertorio en estas categorías

3.2.3 Operaciones

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Regla de comportamiento común a todas las arquitecturas

- Las instrucciones utilizadas más extensamente de un conjunto de instrucciones son las operaciones simples

	Instrucciones 80x86	Promedio
1	Load	22%
2	Salto condicional	20%
3	Comparación	16%
4	Store	12%
5	Add	8%
6	And	6%
7	Sub	5%
8	Move reg-reg	4%
9	Call	1%
10	Return	1%
	Total	96%

- Ejemplo:** 10 instrucciones simples del 80x86 que contabilizan el 96% de las instrucciones ejecutadas. El diseñador debe esforzarse en hacer rápidas estas instrucciones.

3.2.3 Operaciones

Introducción

Características

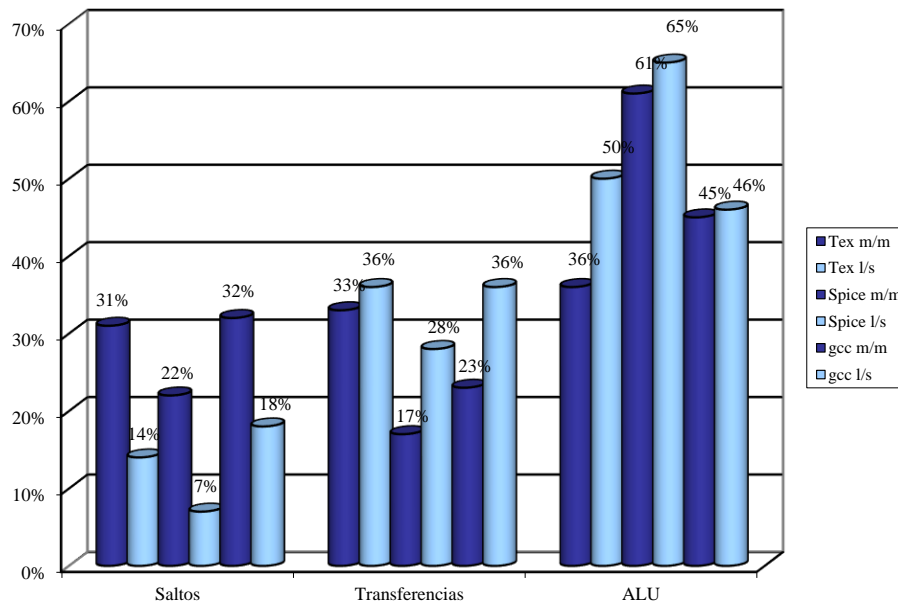
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

b. Rep. Inst. M-M vs R-R (carga/almacenamiento, I/s)

- ◆ Frecuencias para una arquitectura I/s (MIPS) y una M-M (VAX)
 - ◆ Referencias a memoria
 - ◆ Operaciones de la ALU
 - ◆ Instrucciones de flujo de control (saltos y bifurcaciones)



- ◆ Máquina R-R mayor porcentaje de movimientos de datos
- ◆ Frecuencia relativa más baja para saltos en R-R

3.2.3 Operaciones

Introducción

Características

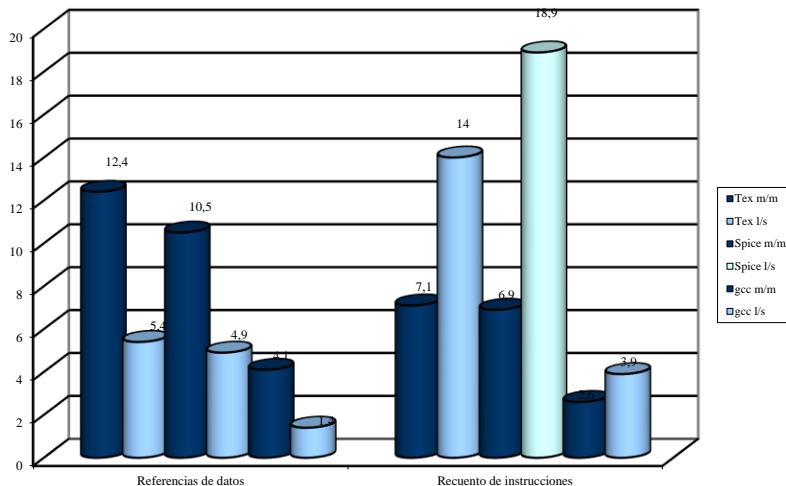
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

b. Rep. Inst. M-M vs R-R (carga/almacenamiento, I/s)

- Recuento absoluto de instrucciones ejecutadas
- Referencias a datos en memoria (cargas, almacenamientos, ALU mem)



- Máquina I/s requiere más instrucciones
- Podríamos deducir: de los RI y las frecuencias de operaciones de transferencia que el número de referencias a datos en I/s es mayor
- Datos indican lo contrario (más referencias a datos en M-M que I/s)
- En M-M referencias a datos no sólo con operaciones de transferencia sino con las ALU

3.2.3 Operaciones

Introducción

Características

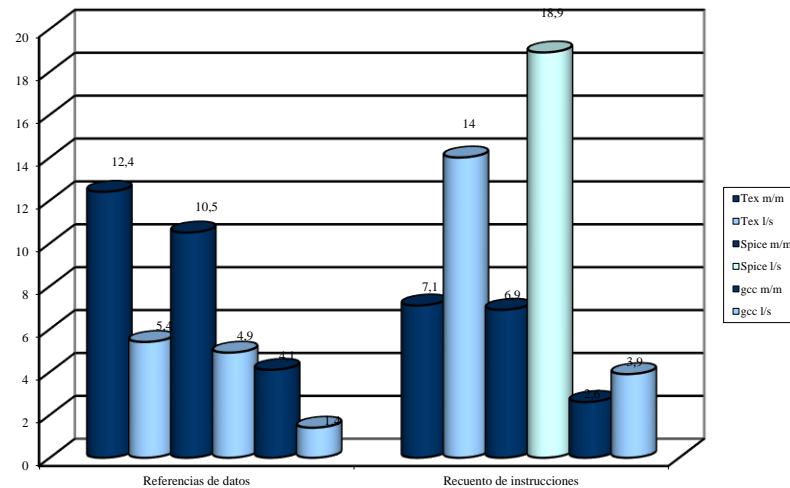
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

b. Rep. Inst. M-M vs R-R (carga/almacenamiento, I/s)

- Recuento absoluto de instrucciones ejecutadas
- Referencias a datos en memoria (cargas, almacenamientos, ALU mem)



- Máquina I/s requiere más instrucciones
- Podríamos deducir: de los RI y las frecuencias de operaciones de transferencia que el número de referencias a datos en I/s es mayor

- Diferencia las referencias a datos consecuencia de mejores posibilidades de ubicación de registros de I/s
- Diferencias entre las referencias a datos de mem-mem y I/s equilibra la diferencia entre referencias a instrucciones

3.2.3 Operaciones

Introducción

Características

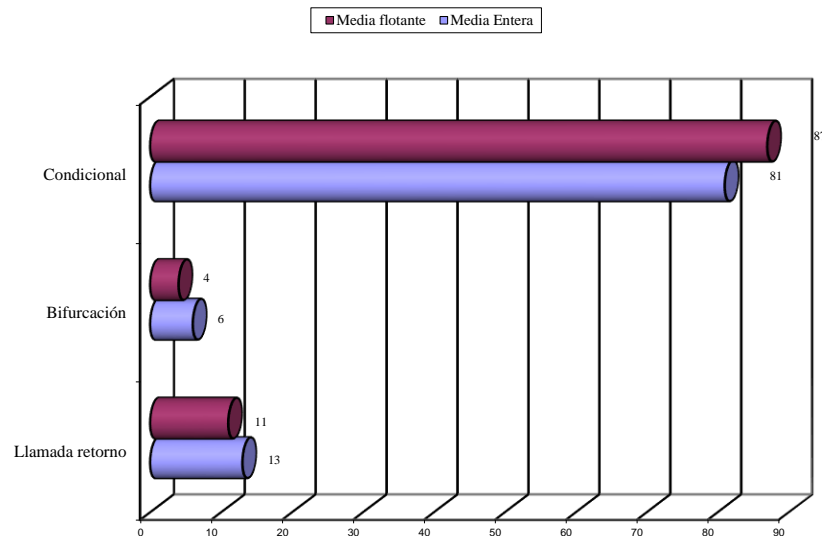
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Instrucciones de control

- Cuatro tipos de cambios del flujo de control
 - Saltos condicionales
 - Bifurcaciones incondicionales
 - Llamadas a procedimientos
 - Retornos de procedimiento
- Frecuencia de instrucciones de flujo de control para máquina I/s



- Los saltos condicionales son los que más se utilizan

3.2.3 Operaciones

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Instrucciones de control

• Formas de especificar el destino del salto

- **Explícitamente** (lo más frecuente) excepción (retorno de procedimiento)
- JE ET, JMP ET, CALL ET, RET

• Saltos relativos al PC

- Dirección especificada mediante desplazamiento sumado al PC
- Normalmente la posición destino del salto es cercana a la actual (pocos bits)

• Saltos no relativos al PC

- Para saltos a direcciones concretas de destino no conocido en tiempo de compilación. Necesario especificarlo dinámicamente.
- Se puede nombrar un registro que contenga la dirección del destino
- Alternativamente, se puede utilizar cualquier modo de direccionamiento

3.2.3 Operaciones

Introducción

Características

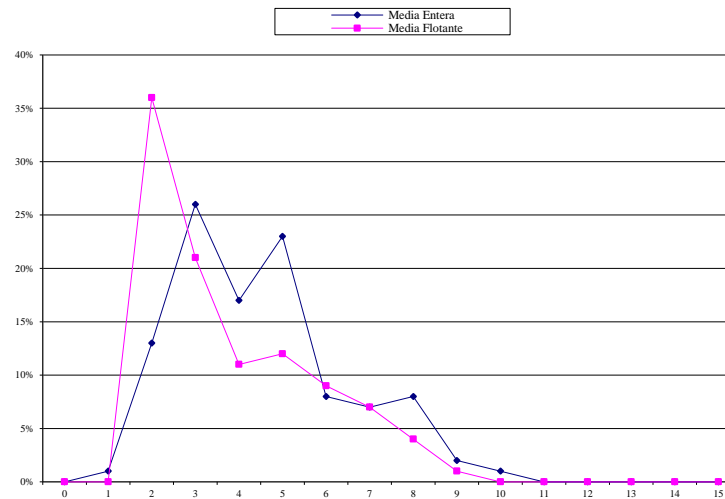
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Instrucciones de control

- Diseñador debe conocer magnitud de los desplazamientos para ver como afecta a la longitud y codificación de la instrucción
- Observamos distancias de los saltos relativos al PC. Número de instrucciones entre el destino y la instrucción de salto



- Saltos más frecuentes en programas enteros tienen entre 3 y 5 bits (25%)
- En programas en punto flotante 2 bits son los más frecuentes
- Los campos de desplazamientos cortos son suficientes 8 bits cubren el 93%

3.2.3 Operaciones

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Instrucciones de control

- **Formas de especificar la condición del salto**

- **Código de condición**

- Los saltos examinan bits especiales inicializados por las operaciones de la ALU

- **Ejemplo:**

- SUB R1, R2, R3; $R1 = R2 - R3$
 - CMP R1, #0; Si $R1 = 0$ el indicador $z = 1$
 - BEQ eti; Salta si $z = 1$

- **Ventaja:** Las comparaciones pueden eliminarse en algún caso.

- **Inconveniente:** Problemas en máquinas segmentadas derivados de la posible utilización simultánea de z desde varias instrucciones.

3.2.3 Operaciones

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Instrucciones de control

- **Formas de especificar la condición del salto**

- **Registro de condición**

- Los saltos examinan registros arbitrarios con el resultado de una comparación

- **Ejemplo:**

- SUB R1, R2, R3; $R1 = R2 - R3$
 - SEQ R10, R1, #0; Si $R1 = 0$ se actualiza R10 con un 1
 - BNEZ R10,eti; Salta si $R10 \neq 0$

- **Ventaja:** Independencia entre la operación y el registro implicado

- **Inconveniente:** Se consume un registro

3.2.3 Operaciones

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Instrucciones de control

• Formas de especificar la condición del salto

• Comparación y salto

- La comparación es parte del salto, permitiendo saltar con una sola instrucción, si bien puede ser demasiado trabajo por instrucción

• Ejemplo:

- SUB R1, R2, R3; $R1 = R2 - R3$
- C&B R1, #0, eti; Si $R1 = 0$ salta a etiqueta.

• **Ventaja:** Reducción del recuento de instrucciones

- **Inconveniente:** Puede ser demasiado trabajo para una instrucción, aumentando el CPI o el clk

3.2.3 Operaciones

Introducción

Características

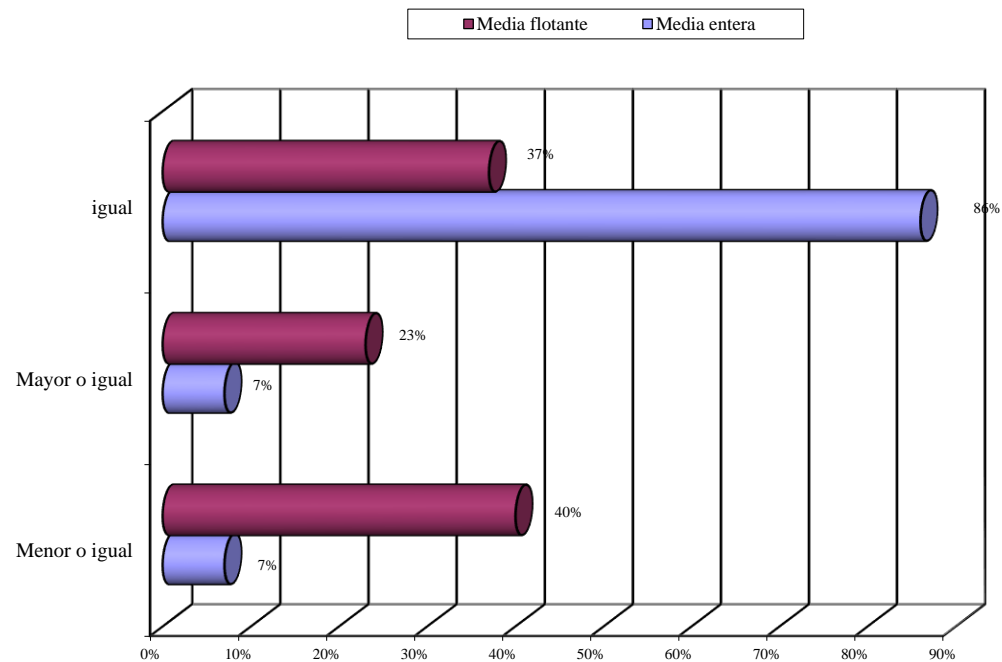
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

c. Instrucciones de control

- La mayor parte de las comparaciones son test de igualdad desigualdad y un gran número son comparaciones con 0 (aproximadamente un 50% son test de igualdad con 0)



3.3 Evolución computadores

Tema 3. Diseño del repertorio de instrucciones

Arquitectura de computadores

3.3.1 Introducción

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

- Inicialmente, las decisiones de diseño de la arquitectura se realizaban para facilitar la programación en lenguaje ensamblador (CISC)
- La aparición de los RISC (y por la madurez de los compiladores) lleva a que los compiladores deban realizar las operaciones eficientemente
- Actualmente, la mayor parte de la programación se realiza en lenguajes de alto nivel para computadores de escritorio, servidores y clusters
 - La mayoría de instrucciones ejecutadas son salida de un compilador
 - La arquitectura a nivel lenguaje máquina es un **objeto del compilador**
 - Decisiones de diseño afectan a la **calidad** del **código** que puede ser generado por un compilador y la **complejidad** de **construir** un buen **compilador**

3.3.2 Arquitectura como objeto del compilador

Introducción

Características

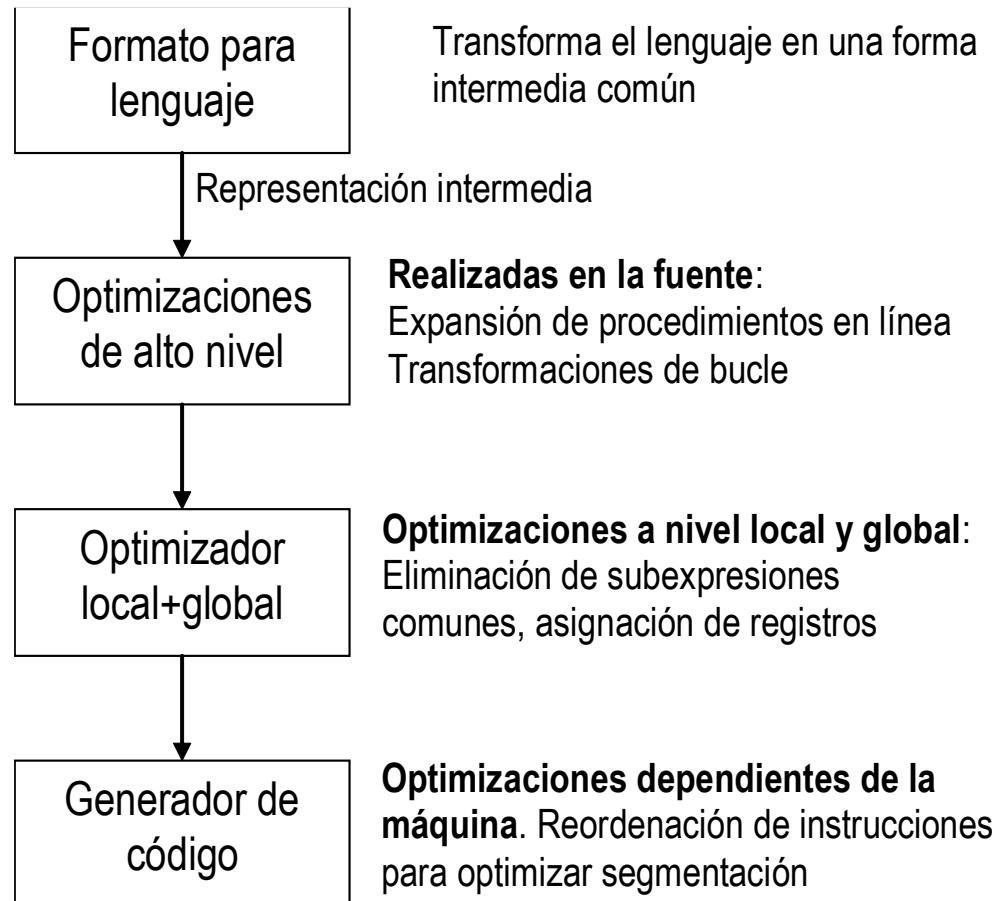
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Estructura de compiladores

- ◆ Pasos de los compiladores para transformar representaciones de alto nivel en representaciones de bajo nivel



3.3.2 Arquitectura como objeto del compilador

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Asignación de registros

- **¿Cuántos registros se necesitan para ubicar las variables?**
 - Óptima ubicación de variables en registros depende del número de registros de propósito general y de estrategia de ubicación
 - Ubicación de registros influye en aceleración del código (acceso a registros frente a acceso a memoria) como en mejorar optimizaciones del compilador (eliminación subexpresiones comunes)
- **Coloreado de grafos:** Algoritmo de ubicación de variables en registros
 - **Problema NP-Completo** pero hay **técnicas heurísticas** que funcionan bien, a partir de 16 reg.
 - El funcionamiento mejora con **al menos 16 registros** (preferiblemente más) de propósito general, para ubicación de variables enteras y análogamente para variables de punto flotante.
- **Ejemplo MIPS** 32 enteros y 32 para trabajo en punto flotante

3.3.2 Arquitectura como objeto del compilador

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

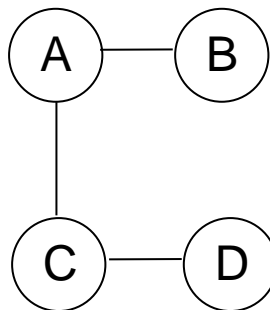
Coloreado de grafos

- Programa: Grafo cuyos nodos son las variables y cuyos arcos muestran el solapamiento en su utilización
- Colorear grafo utilizando número de colores igual al de registros disponibles
- Dos nodos adyacentes no pueden usar el mismo color

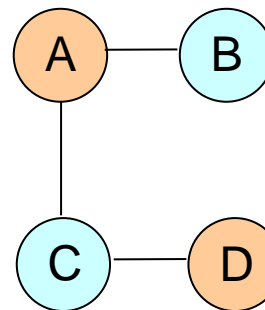
Programa

A=
B=
...
...B...
C=
...A...
D=...
...D...
...C...

Grafo



Grafo Coloreado



Programa registr

R1=
R2=
...
...R2...
R2=
...R1...
R1=...
...R1...
...R2...

3.3.2 Arquitectura como objeto del compilador

Introducción

Características

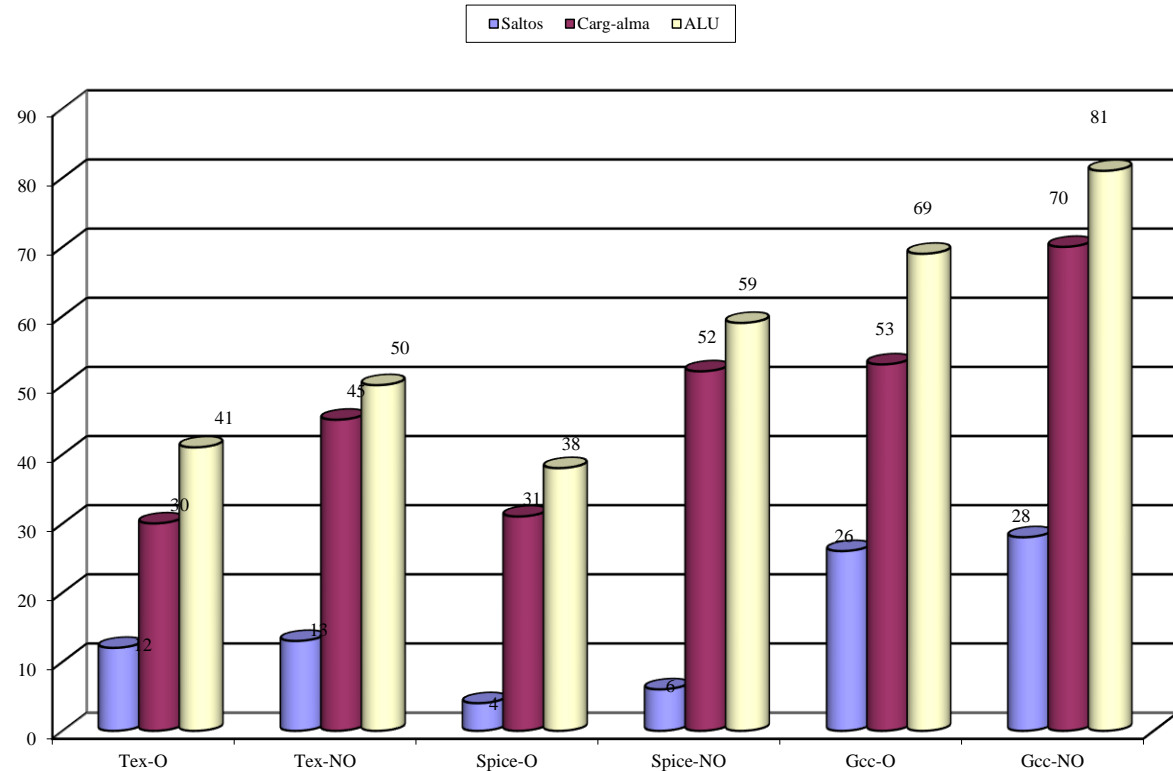
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Influencia de la optimización en la mezcla de instrucciones

- Efecto inmediato de optimización es reducción del RI
- Las estructuras de control son las más difíciles de reducir



3.3.2 Arquitectura como objeto del compilador

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Propiedades que ayudan al diseñador de compiladores

• Ortogonalidad

- Tres componentes principales de un repertorio de instrucciones, operaciones, tipos de datos y modos de direccionamiento deben ser independientes

• Proporcionar primitivas y no soluciones

- Intentos de soportar lenguajes de alto nivel no han tenido éxito

• Proporcionar información de las secuencias alternativas de código de rendimiento óptimo

- Tarea del escritor de compiladores: imaginar secuencias de instrucciones óptimas para cada segmento de código
- El número de instrucciones o el tamaño del código no son representativas

3.3.3 VLIW

Introducción

Características

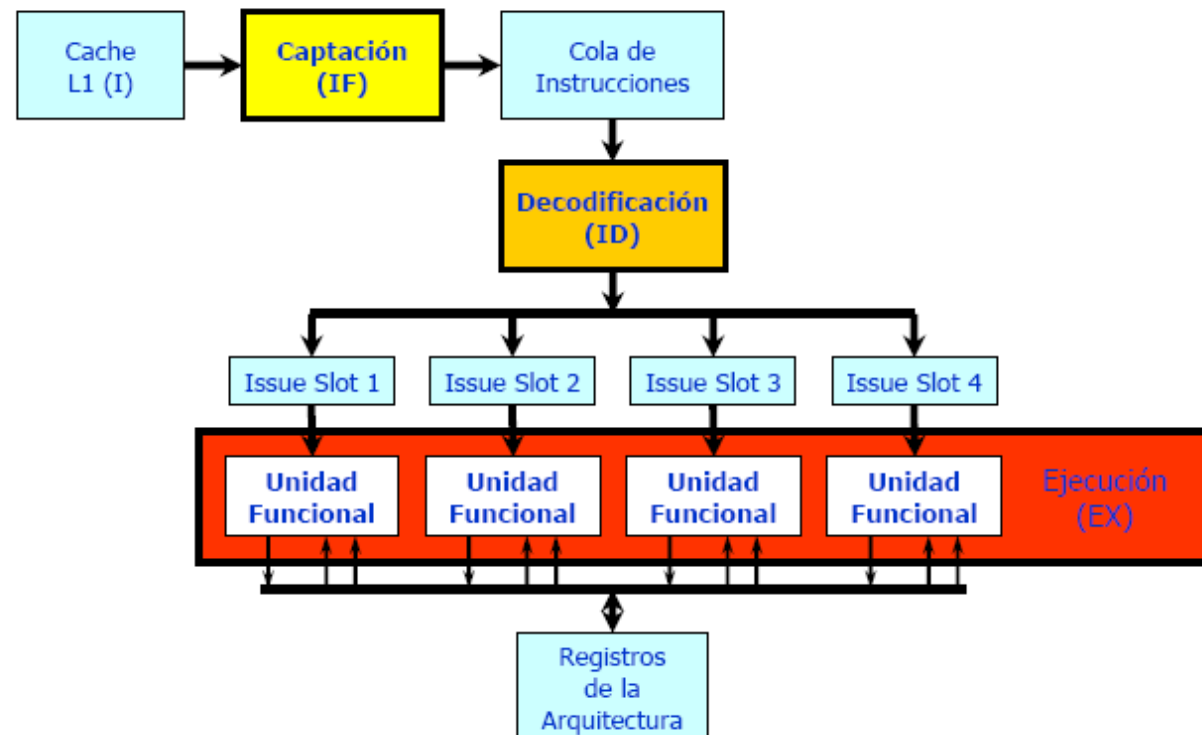
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Evolución de la tecnología de computadores ha permitido:

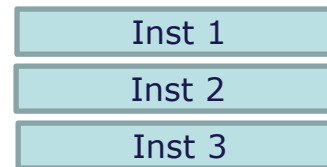
- Disponer de varias unidades de ejecución dentro del mismo procesador
- Los computadores actuales pueden ejecutar varias operaciones simultáneamente en esas unidades de ejecución



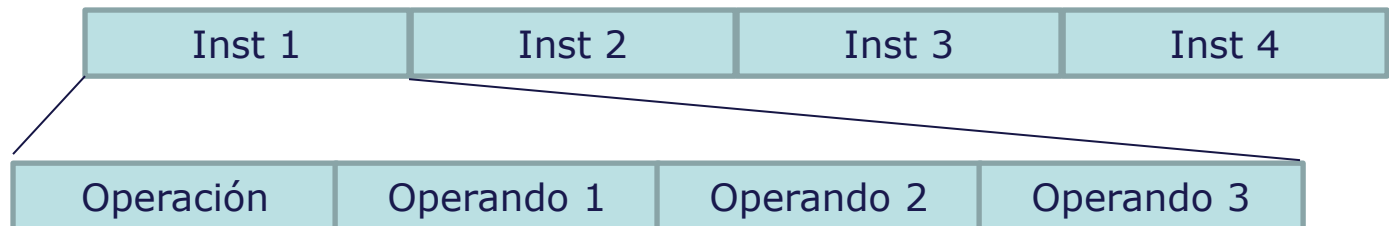
3.3.3 VLIW

Superescalares vs VLIW

- En los procesadores **superescalares**, la organización es la encargada de descubrir el paralelismo que permita aprovechar las instrucciones que se van captando de memoria



- En los procesadores **Very Large Instruction Word (VLIW)**, el paralelismo es implícito en las instrucciones (Intel Itanium-2)
 - Cada instrucción incluye las operaciones que se realizan **simultáneamente**.



Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

3.3.3 VLIW

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Procesamiento VLIW

- La arquitectura VLIW utiliza **varias unidades funcionales independientes**
- En lugar de enviar varias instrucciones independientes a las unidades funcionales, empaqueta varias instrucciones en una única instrucción (112-128 bits)
- La **decisión** de qué instrucciones se deben ejecutar simultáneamente corresponde al **compilador**
- Las ventajas aumentan a medida que se pretenden emitir más instrucciones por ciclo

3.3.3 VLIW

El papel del compilador

◆ Planificación estática (VLIW)

- ◆ Más esfuerzo del compilador: renombrado de registros, reorganizaciones de código,... para mejorar el uso de los recursos disponibles
- ◆ El compilador construye **paquetes de instrucciones sin dependencias**, de forma que el procesador no necesita comprobarlas explícitamente

◆ Planificación dinámica (Superescalar)

- ◆ Menos asistencia del compilador pero más coste hardware (organización) aunque facilita la portabilidad de código entre la misma familia de procesadores

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

3.4 Ejemplos característicos

Tema 3. Diseño del repertorio de instrucciones

Arquitectura de computadores

Ejemplos característicos

Introducción

Características

Programación

Ejemplos

- **VAX** de DEC (ha durado 10 años, década de los 80, y cientos de miles de unidades).
- **IBM 360** (ha durado 25 años décadas 70 y 80, y cientos de miles de unidades).
- **Intel 8086** Es el computador de propósito general más popular del mundo. Decada de los 80 y 90.
- **DLX** Maquina genérica de carga almacenamiento muy popular desde finales de los 80

Diseño del
repertorio de
instrucciones

3.4.1 DEC VAX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Objetivos

- ◆ Facilitar la tarea de escritura de compiladores y sistemas operativos proporcionando una **arquitectura altamente ortogonales**
- ◆ Las **demás arquitecturas** que estudiaremos son **subconjuntos del VAX** en términos de instrucciones y modos de direccionamiento
- ◆ El **VAX** es una **máquina de registros de propósito general**

3.4.1 DEC VAX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Memoria

- ◆ move transfiere datos entre dos posiciones direccionables cualesquiera:
- ◆ Cargas: **reg-mem**
- ◆ Almacenamientos: **mem-reg**
- ◆ Transferencias r-r: **reg-reg**
- ◆ Transferencias m-m: **mem-mem**

3.4.1 DEC VAX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• Tipos de datos

- Inicial del tipo de dato utilizada para completar un nombre de código de operación. Ejemplo mov transfiere un operando del tipo de dato indicado
- **MOVB, MOVW, MOVL, MOVQ, MOVO, MOVF, MOVG, MOVD, ...**

Bits	Tipo de dato	Nuestro nombre	Nombre de DEC
8	Entero	Byte	Byte (B)
16	Entero	Media palabra	Palabra (W)
32	Entero	Palabra	Palabra larga (L)
64	Entero	Doble palabra	Cuad palabra (Q)
128	Entero	Cuad palabra	Octa-palabra (O)
32	Punto flotante	Simple precisión	F_flotante (F)
64	Punto flotante	Doble precisión	D_flotante, G_flotante (D,G)
128	Punto flotante	Huge (Enorme)	H-flotante (H)
4n	Decimal	Empaquetado	Empaquetado (P)
8n	Cadena numérica	Desempaquetado	Cadenas numéricas (S)
8n	Cadenas de caracteres	Caracter	Caracter (C)

3.4.1 DEC VAX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• Modos de direccionamiento

- Una **instrucción VAX** de tres operandos puede incluir **desde 0 a tres referencias a memoria**, cada una de las cuales puede utilizar cualquier modo de direccionamiento

Modo de direccionamiento	Sintaxis
Literal	#valor
Inmediato	#valor
Registro	R_n
Registro diferido	(R_n)
Desplazamiento de byte/palabra/largo	Desplazamiento (R_n)
Desplazamiento diferido de byte/palabra/largo	@Desplazamiento (R_n)
Escalado (indexado)	Modo base $[R_x]$
Autoincremento	$(R_n)+$
Autodecremento	$-(R_n)$
Autoincremento diferido	$@(R_n)+$

3.4.1 DEC VAX

Introducción

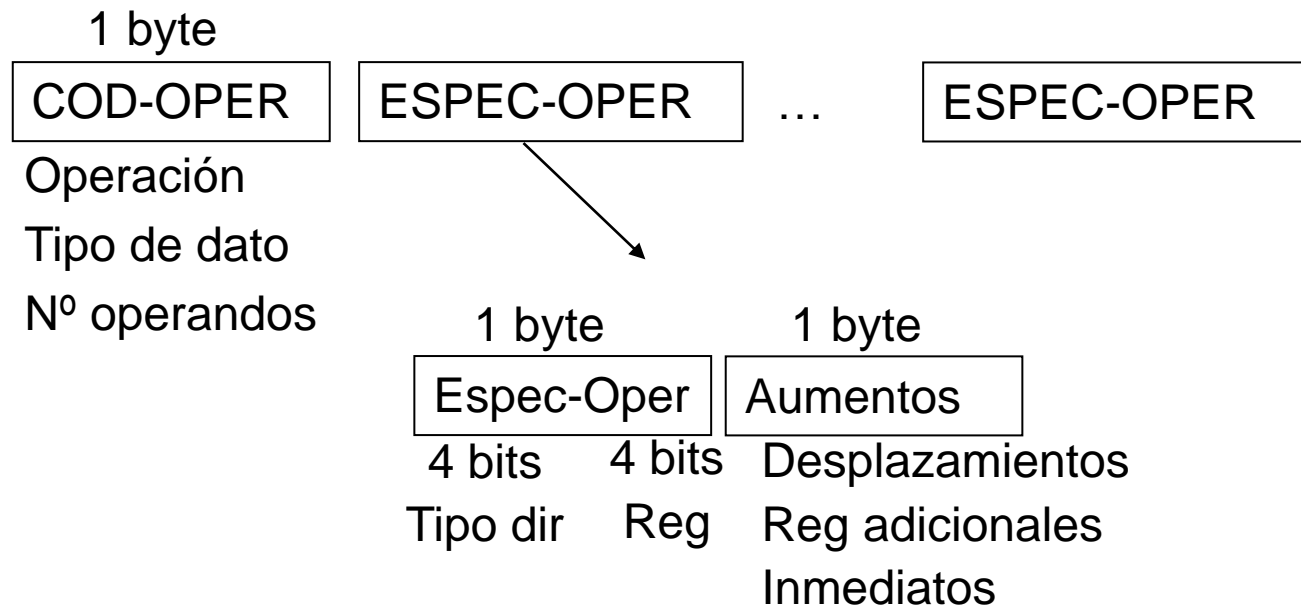
Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• Codificación (Variable)



3.4.1 DEC VAX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Codificación (Variable)

Modo de direccionamiento	Sintaxis	Longitud en bytes
Literal	#valor	1 byte
Inmediato	#valor	1 + longitud del inmediato
Registro	R_n	1
Registro diferido	(R_n)	1
Desplazamiento de byte/palabra/largo	Desplazamiento (R_n)	1 + longitud del desplazamiento
Desplazamiento de byte/palabra/largo	@Desplazamiento (R_n)	1 + longitud del desplazamiento
Escalado (indexado)	Modo base [R_x]	1 + longitud del modo de direccionamiento base
Autoincremento	$(R_n)+$	1
Autodecremento	$-(R_n)$	1
Autoincremento diferido	@(R_n)+	1

3.4.1 DEC VAX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• Codificación (Variable)

• Ejemplo

$$\begin{array}{ccccccc} \bullet & \text{ADDL3} & \text{R1}, & 737(\text{R2}), & \#456 \\ & \uparrow & \uparrow & \uparrow & \uparrow \\ & 1 & + & 1 & + & (1+2) & + & (1+4) \end{array}$$

• Operaciones del VAX (CISC)

- Transferencias de datos
- Aritmética lógica
- Control
- Procedimiento
- Carácter decimal de campo de bits
- Punto flotante
- Sistema
- Otras

3.4.2 IBM 360/370

Introducción

Características

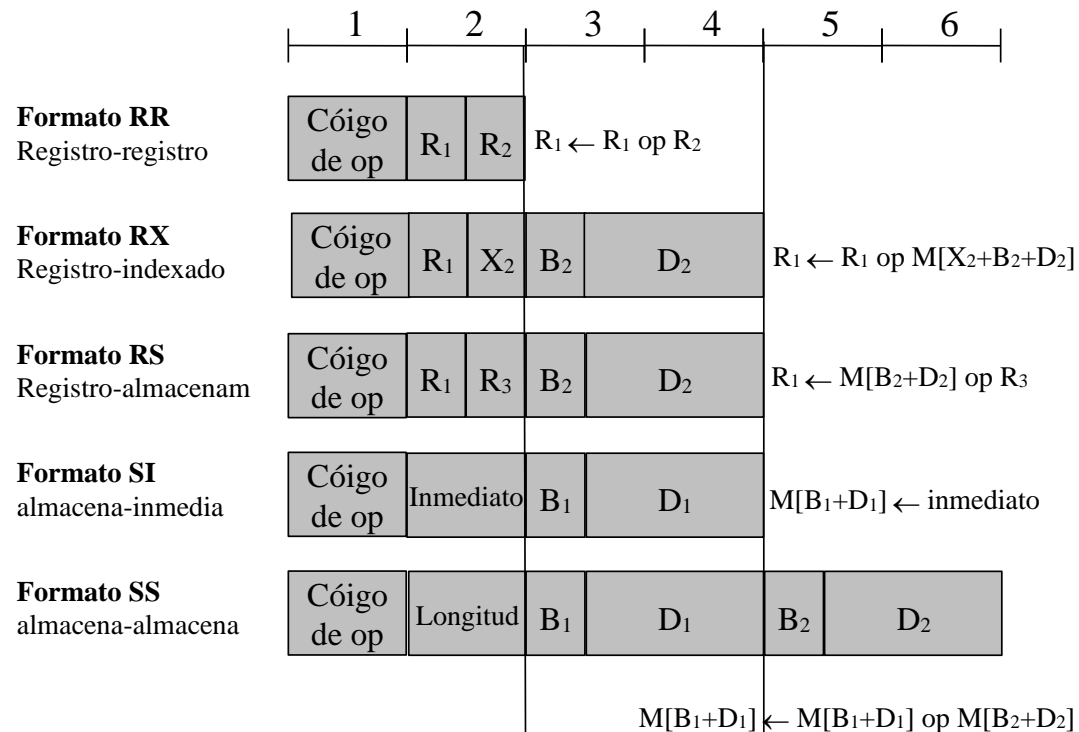
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Objetivos

- Máquina de propósito general con muchos tipos de datos y facilidades para los sistemas operativos
- Compatibilidad del lenguaje máquina



3.4.2 IBM 360/370

Introducción

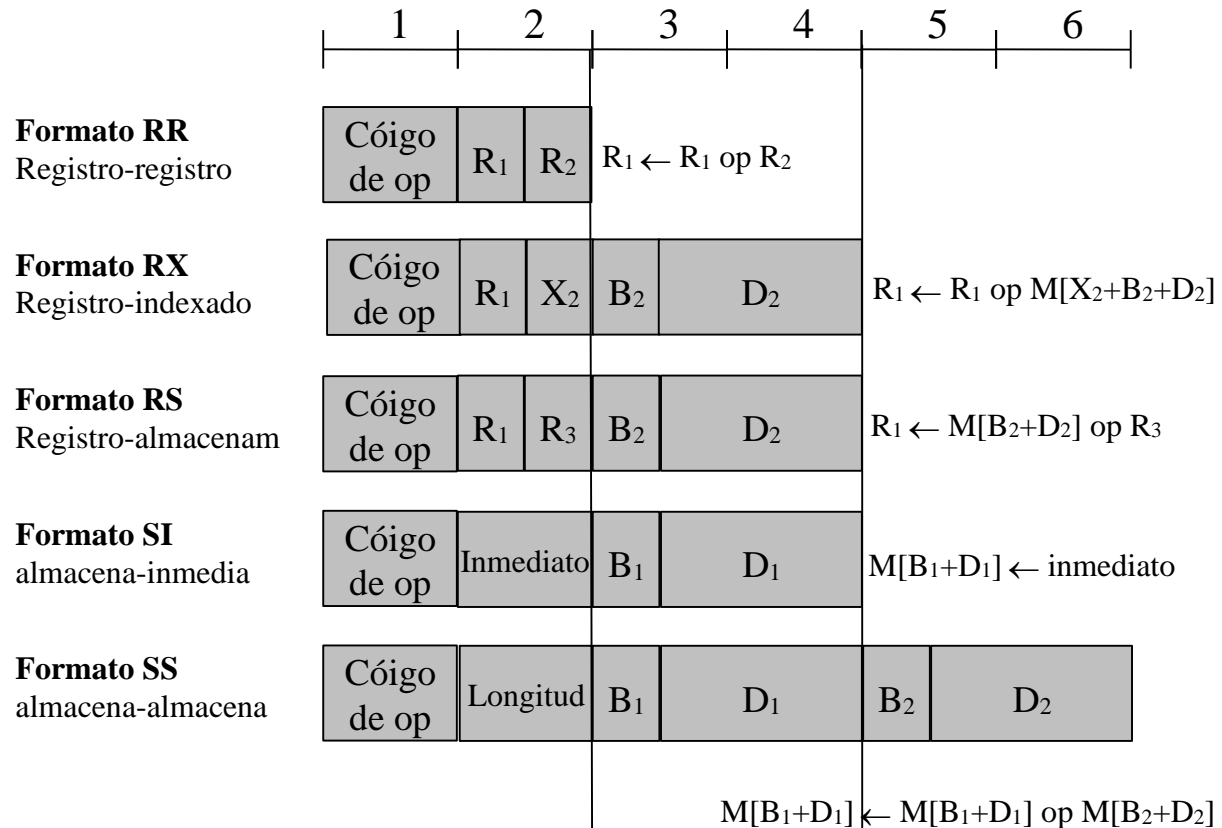
Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Modos de direccionamiento y formatos de instrucción



- ◆ **RR (Registro-registro).** Ambos operandos son el contenido de los registros. El primer operando fuente es también destino

3.4.2 IBM 360/370

Introducción

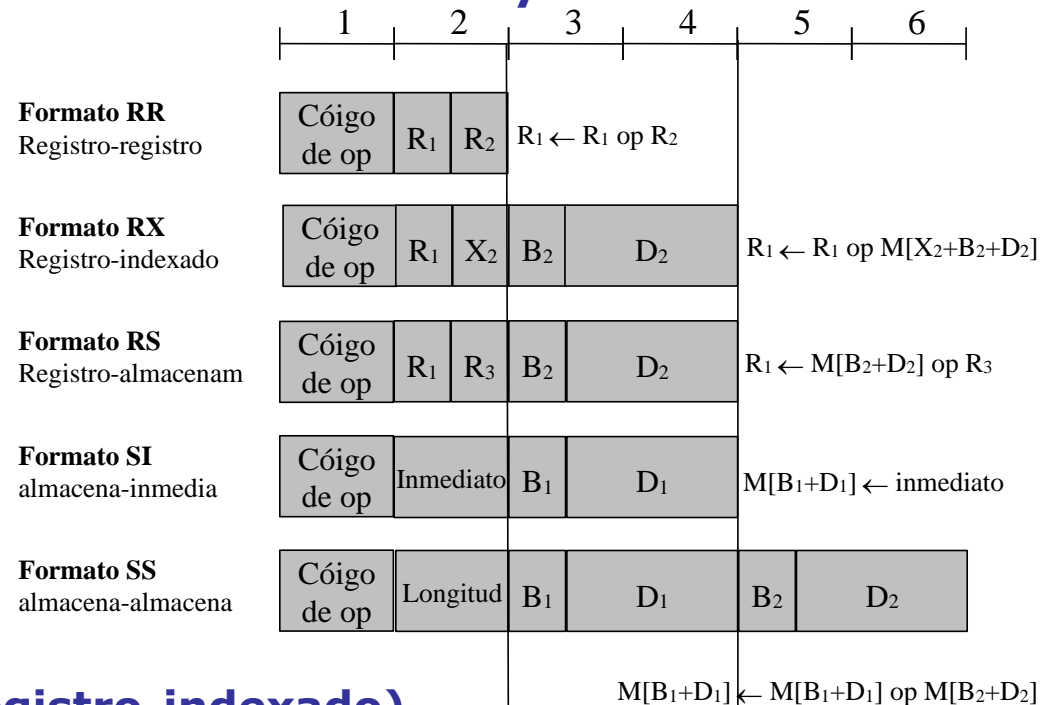
Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Modos de direccionamiento y formatos de instrucción



◆ RX (Registro-indexado)

- ◆ Primer operando (fuente y destino) es un registro
 - ◆ Segundo operando posición de memoria
- D2 desplazamiento de 12 bits
B2 contenido del registro B2
X2 contenido del registro X2

3.4.2 IBM 360/370

Introducción

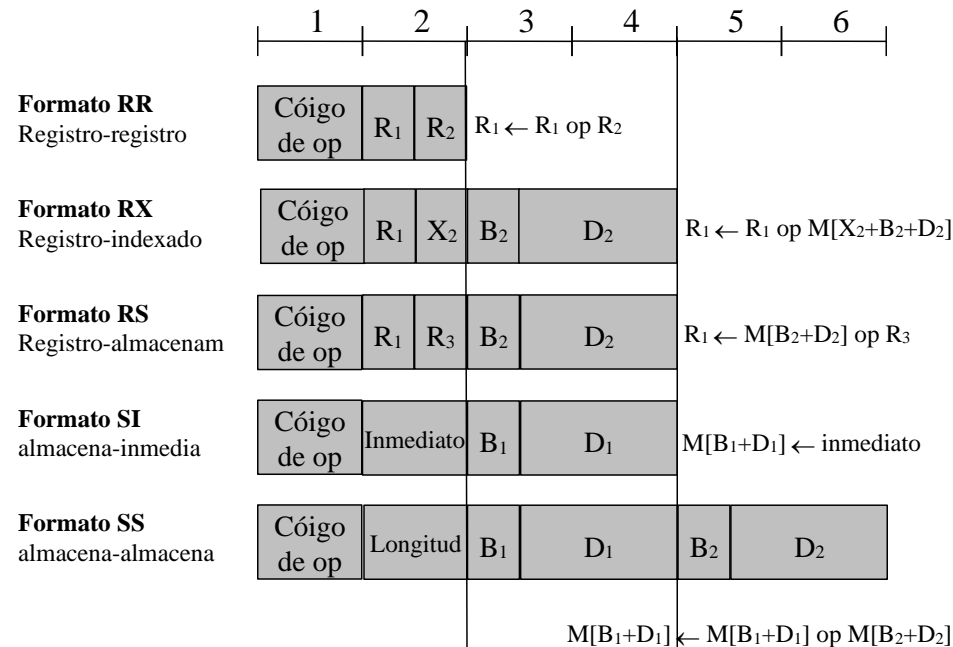
Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Modos de direccionamiento y formatos de instrucción



◆ RS (Registro-memoria)

- ◆ Primer operando es el registro destino
 - ◆ Tercer operando registro como segunda fuente
 - ◆ Segundo operando posición de memoria
- D2: campo desplazamiento de 12 bits
B2: contenido del registro B2

3.4.2 IBM 360/370

Introducción

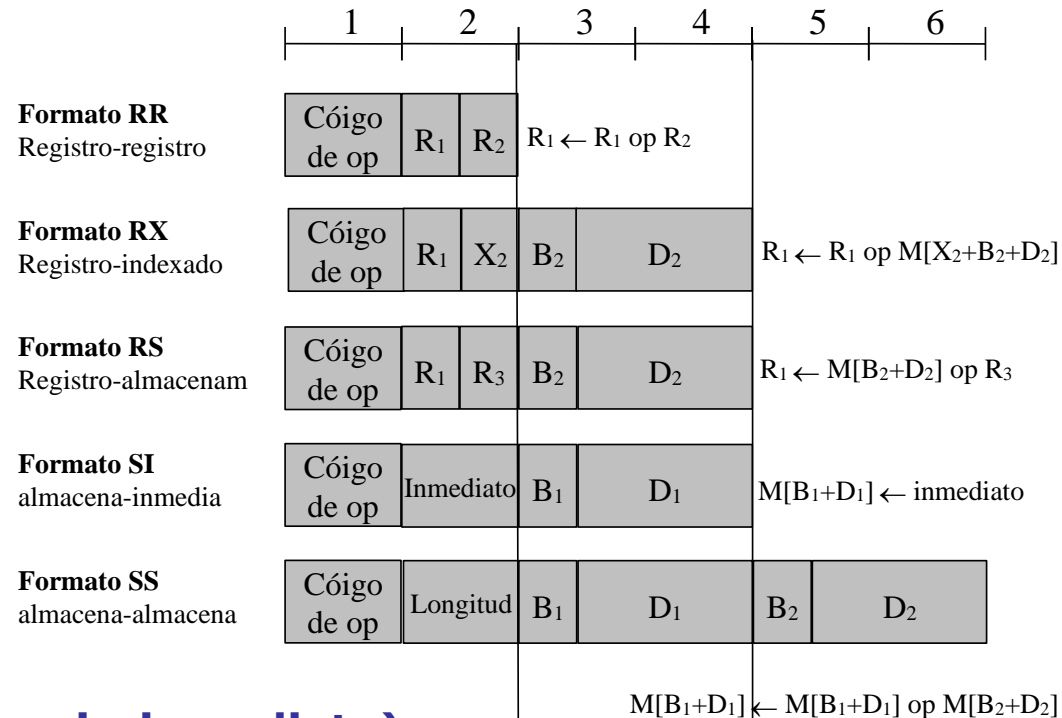
Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Modos de direccionamiento y formatos de instrucción



◆ SI (memoria-inmediato)

- ◆ El destino es un operando de memoria dado por la suma de
- ◆ B1: contenido del registro B1
- ◆ D1: valor del desplazamiento D1.
- ◆ Segundo operando, un campo inmediato de 8 bits es la fuente

3.4.2 IBM 360/370

Introducción

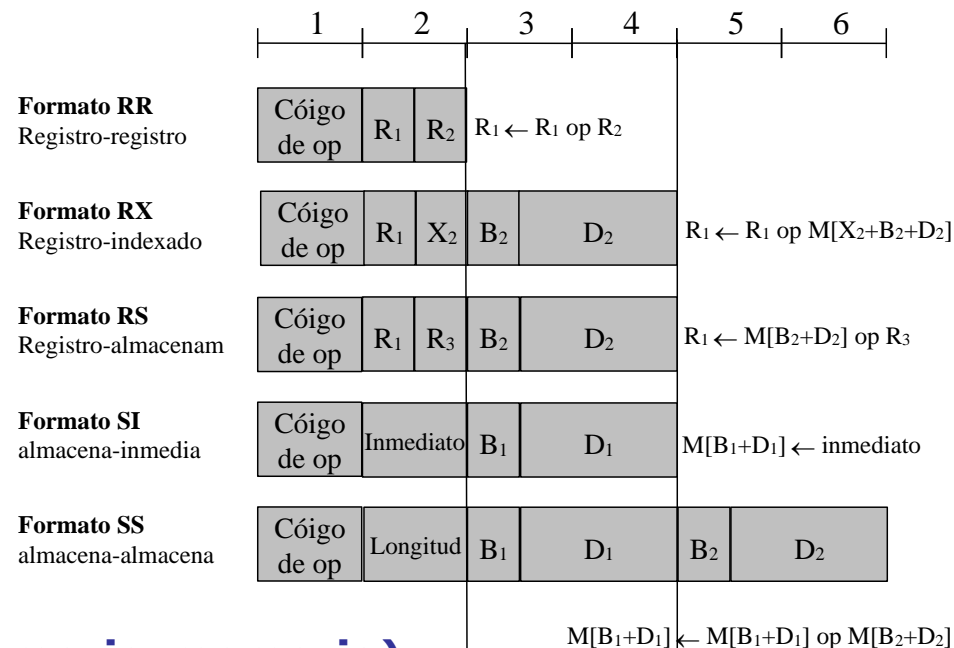
Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Modos de direccionamiento y formatos de instrucción



◆ SS (memoria-memoria)

- ◆ Las direcciones de los dos operandos de memoria son la suma del contenido de un registro base B_i y un desplazamiento D_i
- ◆ El primer operando es fuente y destino

3.4.2 IBM 360/370

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• **Operaciones del 360/370 (CISC)**

- **Control**
- **Aritmético, lógica**
- **Transferencia de datos**
- **Punto flotante**
- **Cadena decimal**

3.4.3 Intel 8086

Introducción

Características

Programación

Ejemplos

- La arquitectura 8086 extensión del 8088 (máquina acumulador)
- El 8086 amplió el banco de registros
- Arquitectura de 16 bits
- Memoria segmentada
- Los diseñadores lograron un espacio de direcciones de 20 bits mediante la segmentación de la memoria

Diseño del
repertorio de
instrucciones

3.4.3 Intel 8086

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• La familia x86

- **Los 80186, 80286, 80386, 80486, Pentium (Pro, II, III, M, 4), Core 2, Core i3/i5/i7** son extensiones compatibles del 8086
- **El 80186** extendió el repertorio original. Sistema de 16 bits.
- **El 80286** amplió el espacio de direcciones a 24 bits. Multitarea y memoria virtual
- **El 80386** (1985) verdadera máquina de 32 bits (registros de 32 bits) (espacio de direcciones de 32 bits). Nuevo conjunto de modos de direccionamiento y de operaciones

3.4.3 Intel 8086

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ La familia x86

- ◆ **El 80486** (1989) más instrucciones. Incremento del rendimiento. Segmentación del cauce (5 etapas) y cache más sofisticada
- ◆ **Pentium.** Introducción de técnicas superescalares, varias instrucciones en paralelo. (varias unidades de ejecución)
- ◆ **Pentium Pro (1995):** Profundiza sobre técnicas superescalares
- ◆ **Pentium II:** tecnología MMX (procesamiento eficiente de video audio y gráficos). Se utilizan registros de la pila del coprocesador

3.4.3 Intel 8086

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ La familia x86

- ◆ **Arquitectura del Pentium II (similar a la del Pentium Pro) consta de una envoltura CISC con un núcleo RISC**
 1. El procesador capta instrucciones de memoria
 2. Cada inst se traduce en varias inst RISC tamaño fijo (microops)
 3. El procesador ejecuta las microops con organización superescalar
 4. Datos escriben en BR en orden establecido por programa
- ◆ **Pentium III:** Instrucciones adicionales en punto flotante para procesamiento eficiente de gráficos 3D. SSE (Streaming SIMD Extension) 8 nuevos registros de 128 bits
- ◆ **Pentium 4.** Supersegmentada de 20 etapas. Duplica ALUs (2 unidades enteras). Nuevas instrucciones SSE

3.4.3 Intel 8086

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ La familia x86

- ◆ **Core 2 (2006):** Arquitectura de 64 bits. Duo: 2 cores
- ◆ **i3,i5,i7 (2008):** Quad-core

3.4.3 Intel 8086

Introducción

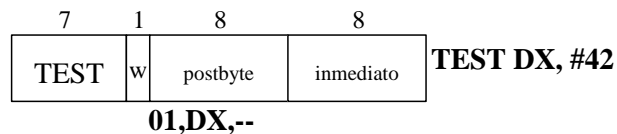
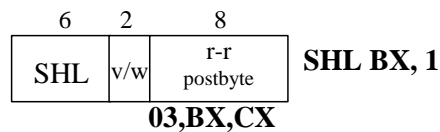
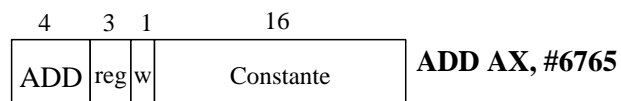
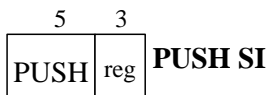
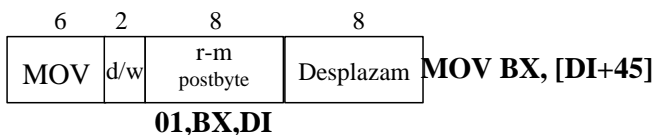
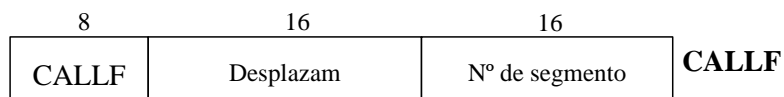
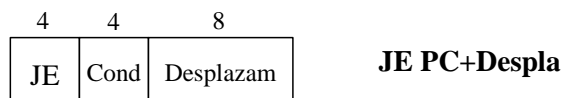
Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Formatos de instrucción



3.4.3 Intel 8086

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Formatos de instrucción

Código (8 bits)	Post-byte(8 bits)	Des	Val
	mod(2b) reg(3b) rm(3b)		

- ◆ **Código:** 1er byte, es el único que existe siempre, el resto pueden aparecer o no.
- ◆ **Post-byte:** Refleja los operandos de la instrucción
 - 1er operando:** mediante mod y rm. Puede ser un registro o una posición de memoria. mod=tipo de direccionamiento. rm=registro de direccionamiento.
 - 2º operando** mediante reg: Debe ser un registro.
- ◆ **Des:** componente desplazamiento de una dirección de memoria. 1 o 2 bytes
- ◆ **Val:** valor inmediato. 1 o 2 bytes

3.4.3 Intel 8086

Introducción

Características

Programación

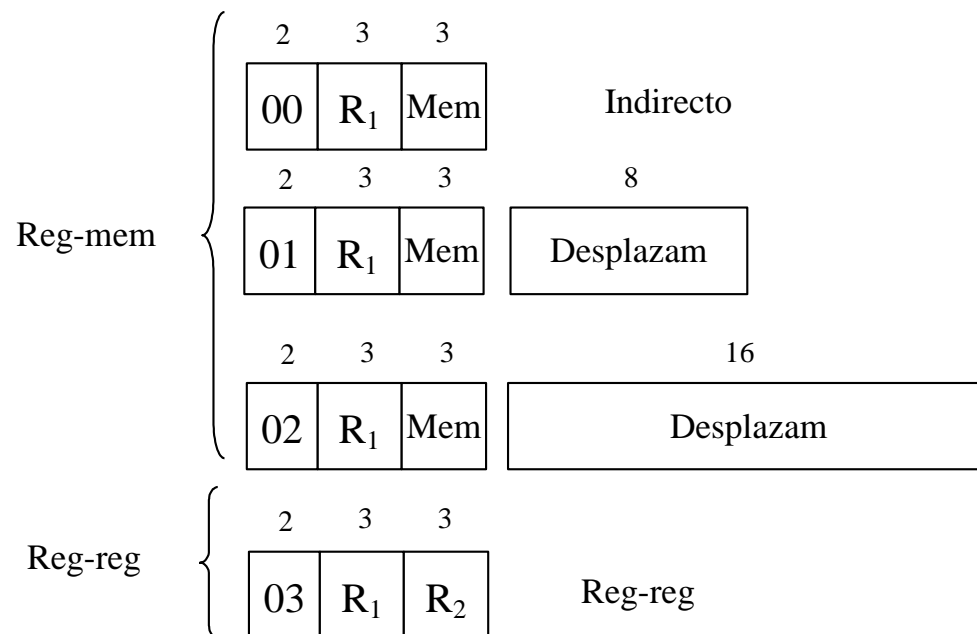
Ejemplos

Diseño del
repertorio de
instrucciones

Formatos de instrucción

Código (8 bits)	Post-byte(8 bits)	Des	Val
	mod(2b) reg(3b) rm(3b)		

Hay cuatro codificaciones posibles para el postbyte:



3.4.4 DLX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ La arquitectura DLX

- ◆ **DLX es una sencilla arquitectura de carga almacenamiento.**
- ◆ **Nombre promedio varias máquinas próximas a DLX en romanos**
- ◆ AMD 29K, DECstation 3100, HP 850, IBM 801, Intel i860, MPS M/102^a, MIPS M/1000, Motorola 88K, RISC I, SGI 4D/60, SPARCstation-1, Sun-4/110, Sun-4/260.
- ◆ **La arquitectura DLX se escogió basándose en las observaciones sobre las primitivas más frecuentes utilizadas en los programas**
- ◆ **Las funciones más sofisticadas se implementaban a nivel software con múltiples instrucciones**

3.4.4 DLX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

• La arquitectura DLX

- **DLX hace énfasis en:**
- **Un sencillo repertorio de instrucciones de carga almacenamiento**
- **Diseño de segmentación eficiente (pipelining)**
- **Un repertorio de instrucciones fácilmente decodificables**
- **Eficiencia como objeto del compilador**

3.4.4 DLX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Características

◆ Los registros

- ◆ La arquitectura tiene 32 registros de propósito general GPR de 32 bits; el valor de R0 siempre es 0.
- ◆ Registros de punto flotante (FPR), se pueden utilizar como 32 registros de simple precisión (32 bits), o como parejas de doble precisión F0 , F2,, F28 , F30 .
- ◆ Registros especiales para acceder a la información sobre el estado, que se pueden transferir a y desde registros enteros (ej. Registro de estado de punto flotante)

3.4.4 DLX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

◆ Características

◆ La memoria

- ◆ La memoria es direccionable por bytes en el modo <<Big Endian>> con una dirección de 32 bits.
- ◆ Todas las referencias a memoria se realizan a través de cargas o almacenamientos entre memoria y los GPR o FPR.
- ◆ Los accesos que involucran a los GPR pueden realizarse a un byte, a media palabra y a una palabra.
- ◆ Los accesos que involucran a los FPR pueden realizarse a palabras en simple o doble precisión.
- ◆ Los accesos a memoria deben estar alineados
- ◆ Todas las instrucciones son de 32 bits y deben estar alineadas

3.4.4 DLX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Operaciones

Tipo de instrucción. Cód de oper	Significado de la instrucción
Transferencia de datos	Transfieren datos entre registros y memoria, o entre registros enteros y FP o registros especiales.
LB, LBU, SB	Carga byte , carga byte sin signo, almacena byte
LH, LHU, SH	Carga med pal , carga med pal sin signo, almacena med pal
LW, SW	Carga palabra , almacena palabra
LF, LD, SF,SD	Carga punto flotante SP, carga punto flotante DP, almacena punto flotante SP, almacena punto flotante DP
MOVI2S, MOVS2I	Transfiere desde/ a GPR a/ desde un registro especial
MOVF, MOVD	Copia un registro de punto flotante a un par en DP
MOVFP2I, MOVI2FP	Transfiere 32 bits desde/a registros FP a/ desde registros enteros
Aritmético-lógicas	Operaciones sobre datos enteros o lógicos en GPR.
ADD, ADDI, ADDU, ADDUI	Suma , suma inmediato (todos los inmediatos son de 16 bits)
SUB, SUBI, SUBU, SUBUI	Resta , resta inmediato con y sin signo
MULT, MULTU, DIV, DIVU	Multiplifica y divide , con signo y sin signo, los operandos deben estar en registros de punto flotante
AND, ANDI	And , and inmediato
OR, ORI, XOR, XORI	Or , or inmediato, or exclusiva, or exclusiva inmediata
LHI	Carga inmediato superior, carga la mitad superior de registro con inmediato
SLL, SRL, SRA, SLLI, SRLI, SRAI	Desplazamientos , lógicos dere izqu, aritméticos derecha

3.4.4 DLX

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Operaciones

Tipo de instrucción. Cód de oper	Significado de la instrucción
Control	Salto y bifurcaciones condicionales; relativos al PC o mediante registros.
BEQZ, BNEZ	Salto GPR igual/no igual a cero, despla 16 bits
BFPT, BFPF	Test de bit de comparación reg estado FP y salto, despla 16
J, JR	Bifurcaciones: desplazamiento de 26 bits
JAL, JALR	Bifurcación y enlace
TRAP	Transfiere a S.O. a una dirección vectorizada
RFE	Volver a código de usuario desde una excepción
Punto flotante	Operaciones en punto flotante en formatos DP y SP
ADDD, ADDF	Suma números DP, SP
SUBD, SUBF	Resta números DP, SP
MULTD, MULTF	Multiplca punto flotante DP, SP
DIVD, DIVF	Divide punto flotante DP, SP
CVTF2D, CVTF2I, CVTD2F, CVTD2I, CVTI2F, CVTI2D	Convierte instrucciones
____D, ____F	Compara DP, SP

3.4.4 DLX

Introducción

Características

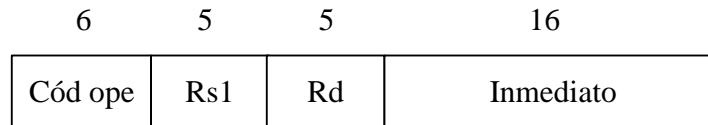
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

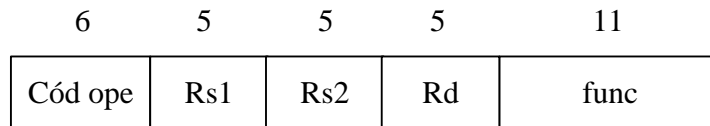
Codificación de las instrucciones

Instrucción tipo I



- Cargas y almacenamientos (byte, media palabra, palabra)
- ALUs con operandos inmediatos
- Instrucciones de salto condicional (BEQZ, BNEZ)
Rs1 registro implicado Rd no se utiliza
- Saltos a registro
Rd=0; Inmediato=0; Rs1=destino

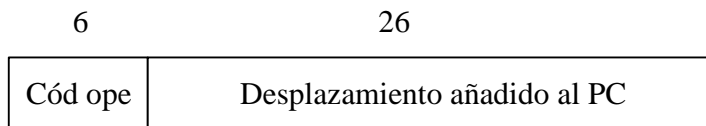
Instrucción tipo R



- Aritméticas y lógicas entre registros

Rs1= fuente1
Rs2= fuente2
Rd= Registro destino
Fun.= operación del flujo de datos

Instrucción tipo J



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