

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

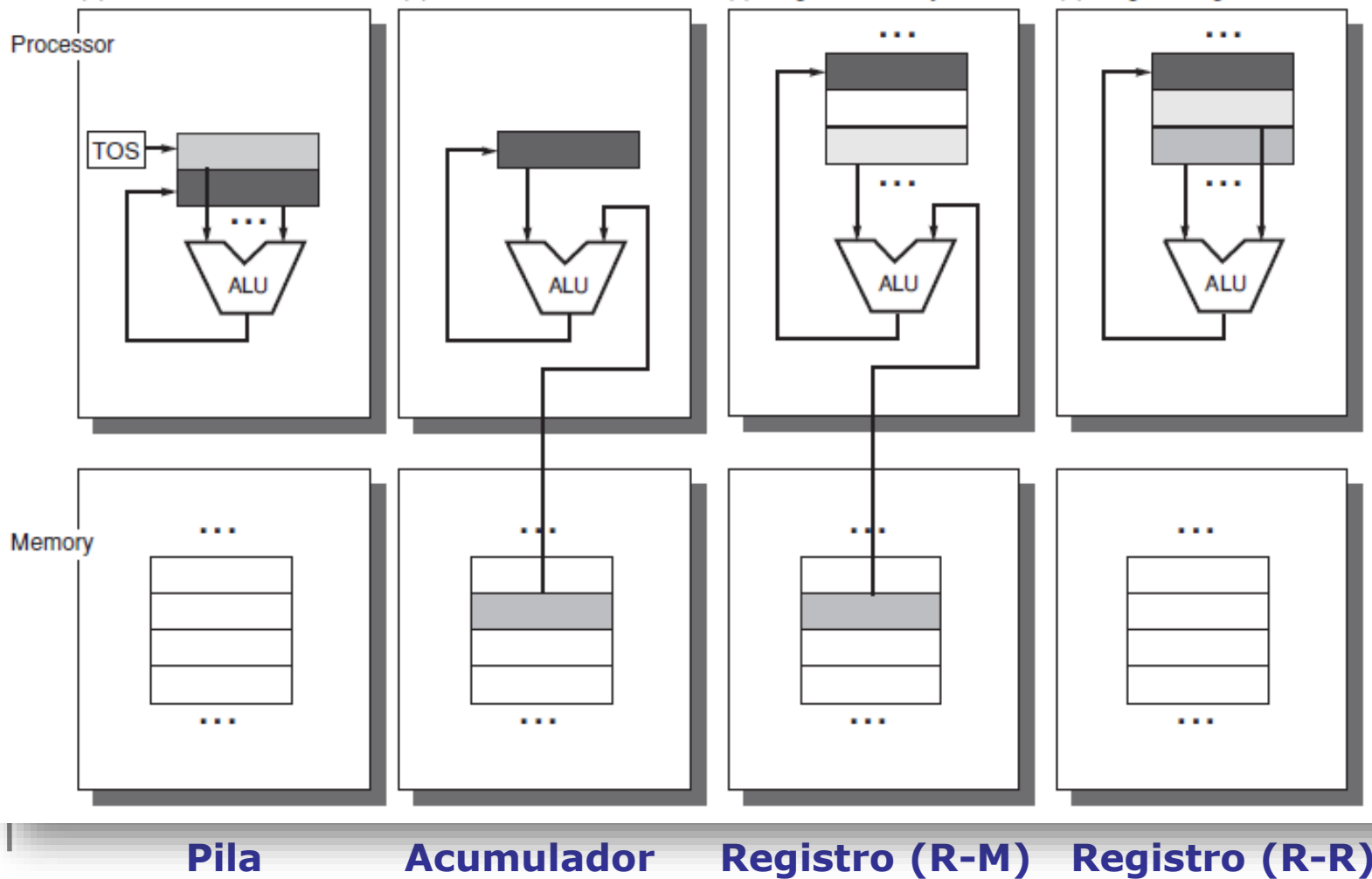
Diseño del
repertorio de
instrucciones

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

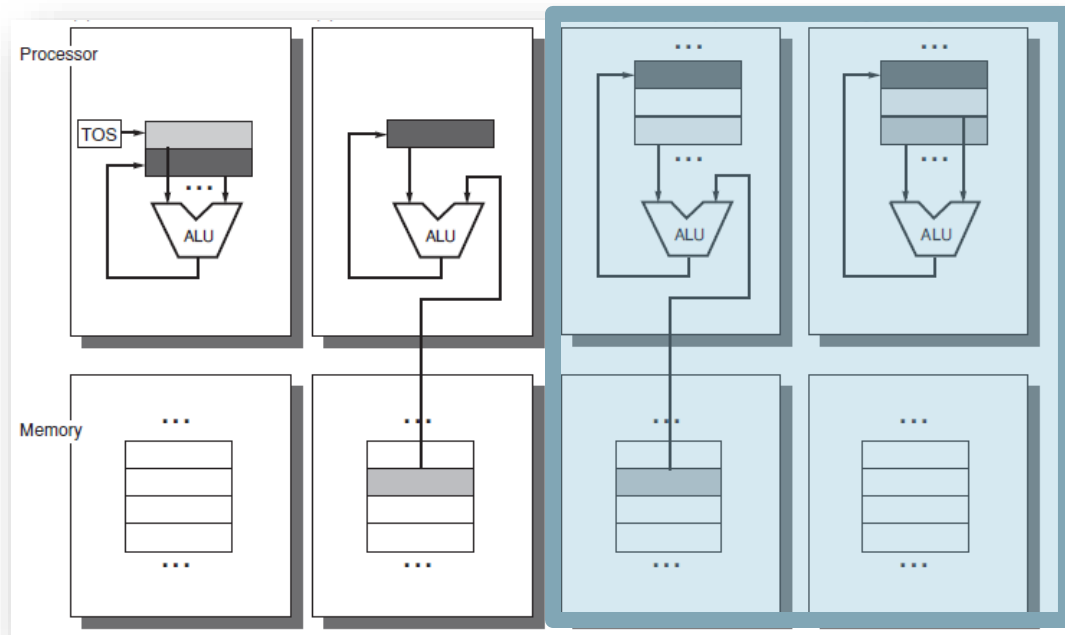
Programación

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

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 (ej. VAX, DLX)
 - **Dos operandos:** Un operando es fuente y destino
 - ADD R1,R2 (ej. VAX, IBM3060, x86)

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

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.

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

3.1.3 Arquitecturas GPR

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.

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

3.2 Características

Tema 3. Diseño del repertorio de instrucciones

3.2.1 Direccionamiento 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 qué extremo de byte comenzar y la discusión **de los Viajes de Gulliver sobre qué extremo del huevo abrir**

3.2.1 Direccionamiento de la memoria

Introducción

Características

Programación

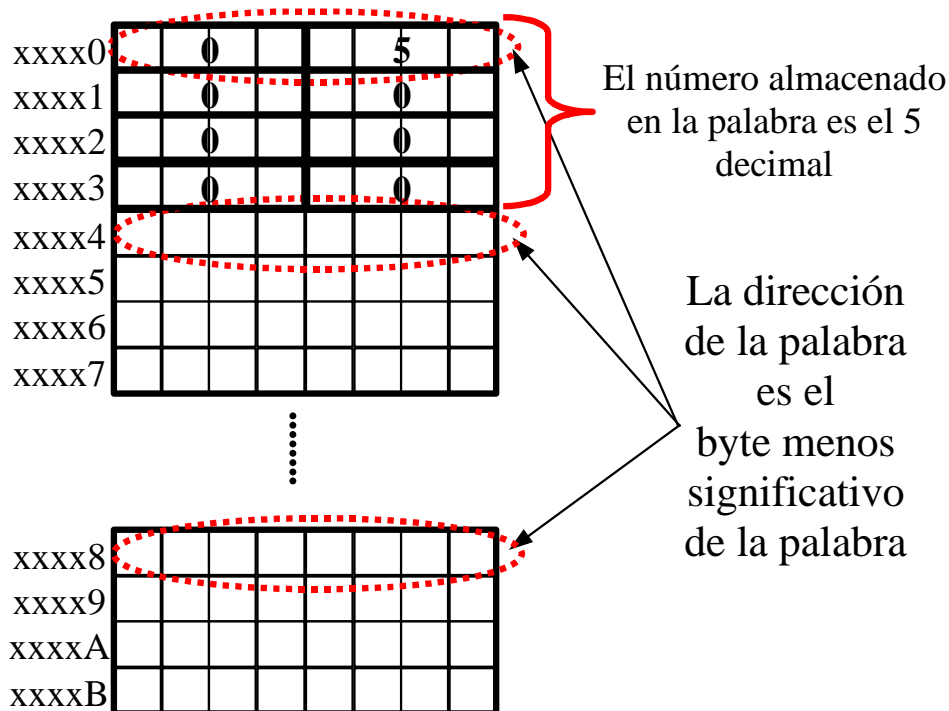
Ejemplos

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

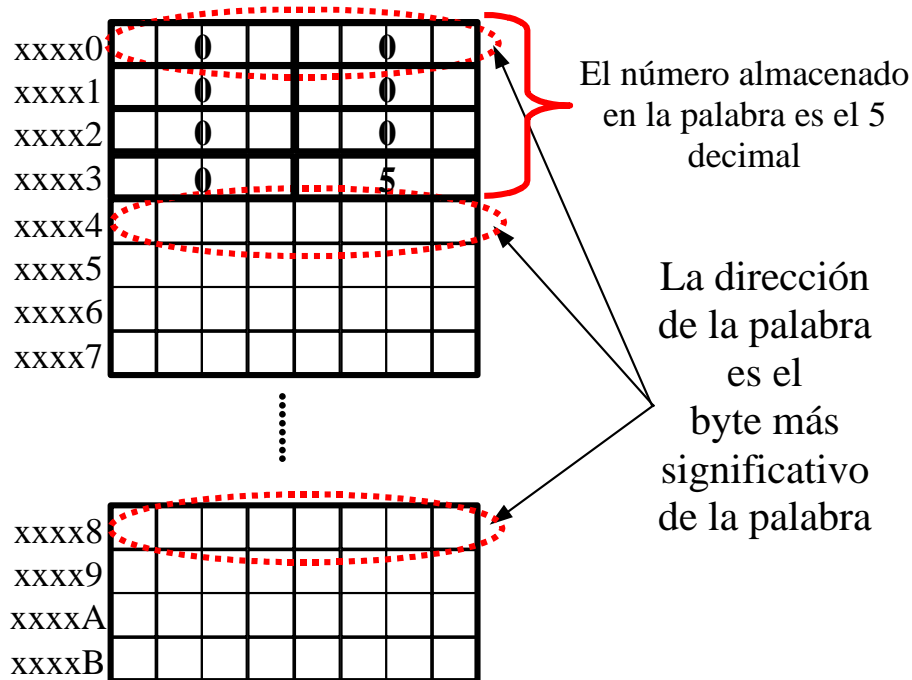
Ejemplos

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

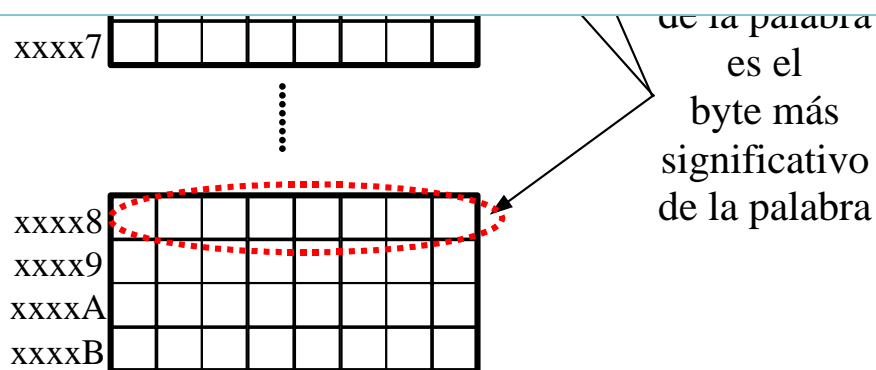
Diseño del
repertorio de
instrucciones

• 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

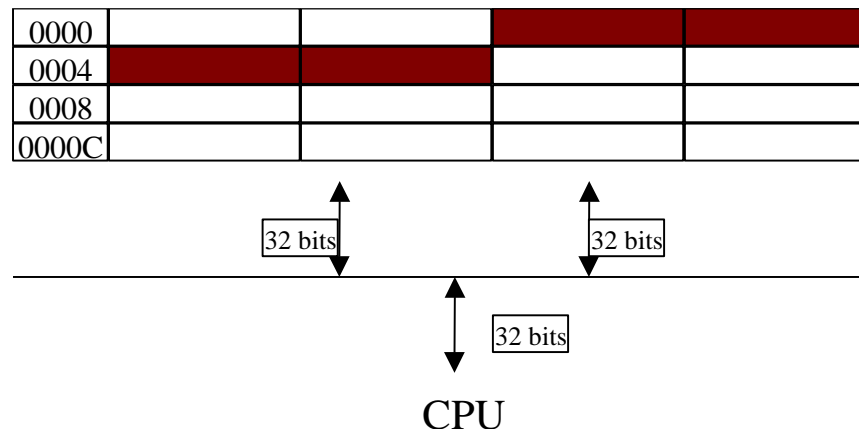
Programación

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:** Qué 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

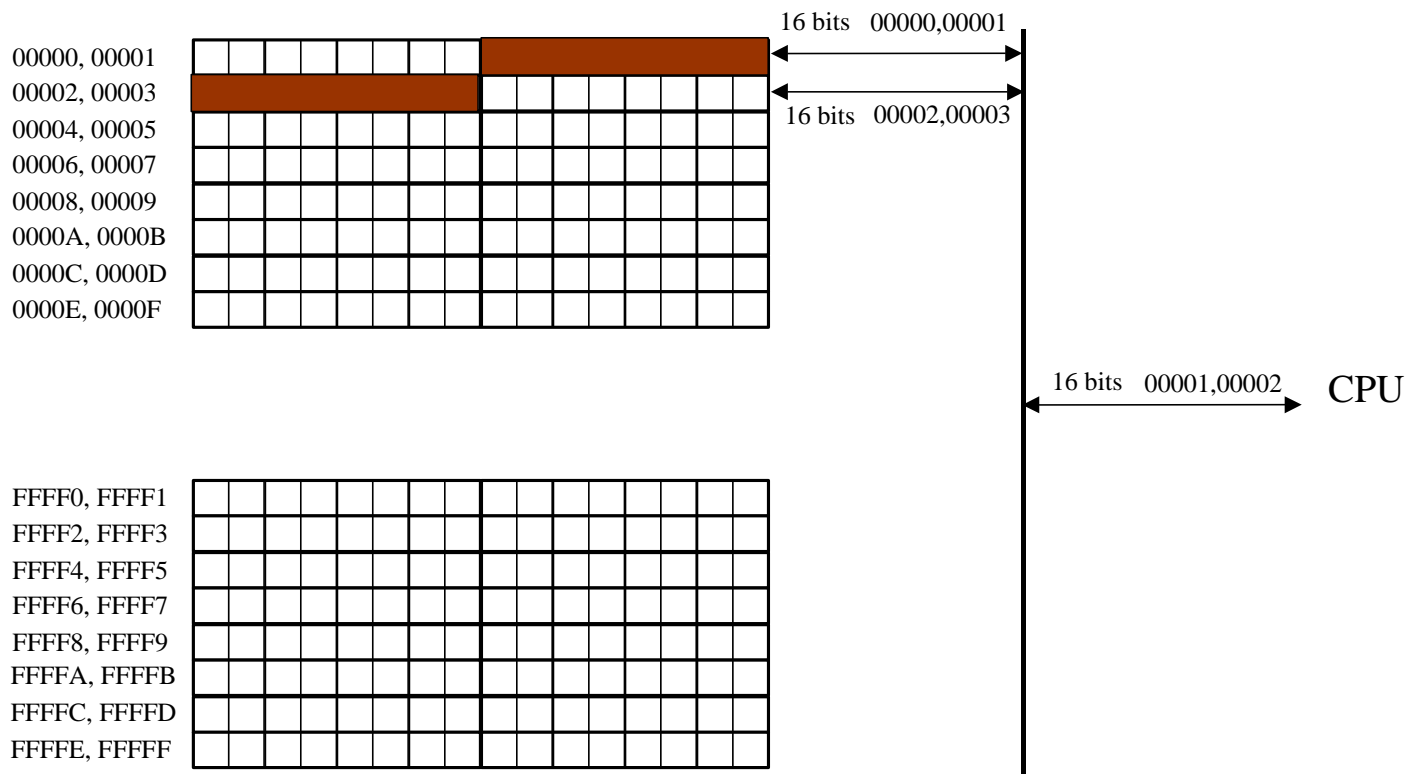
Programación

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**

- ❖ 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

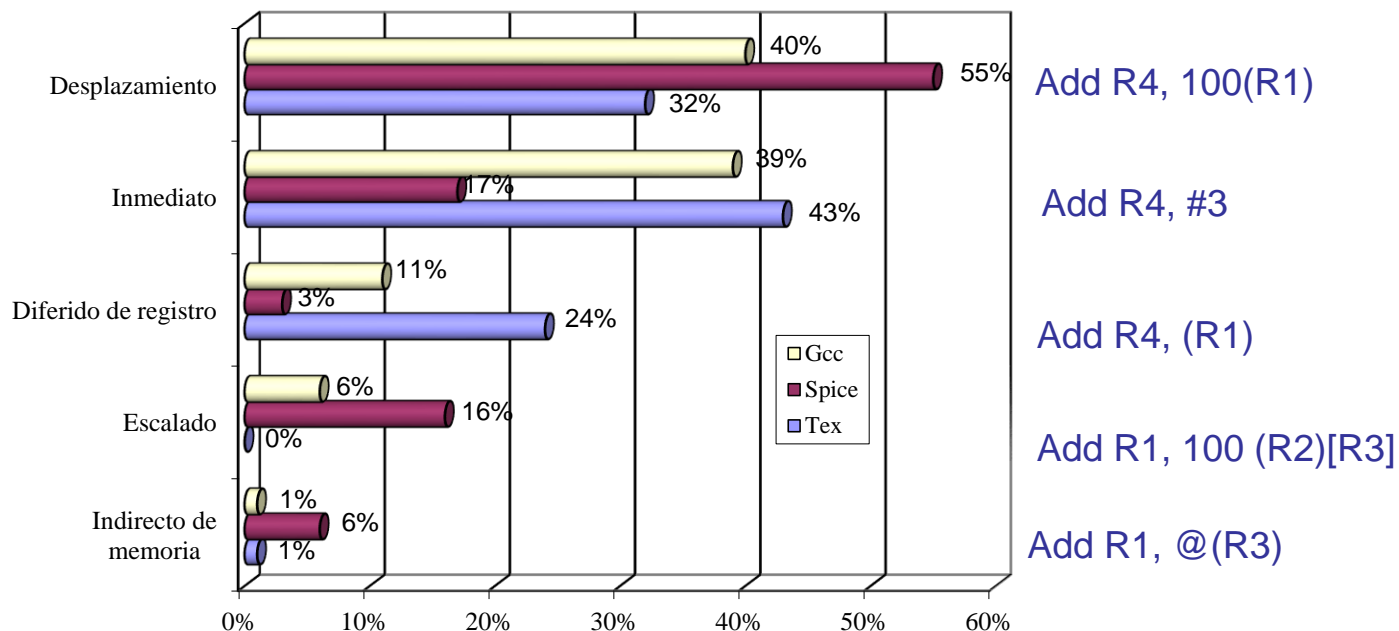
Programación

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

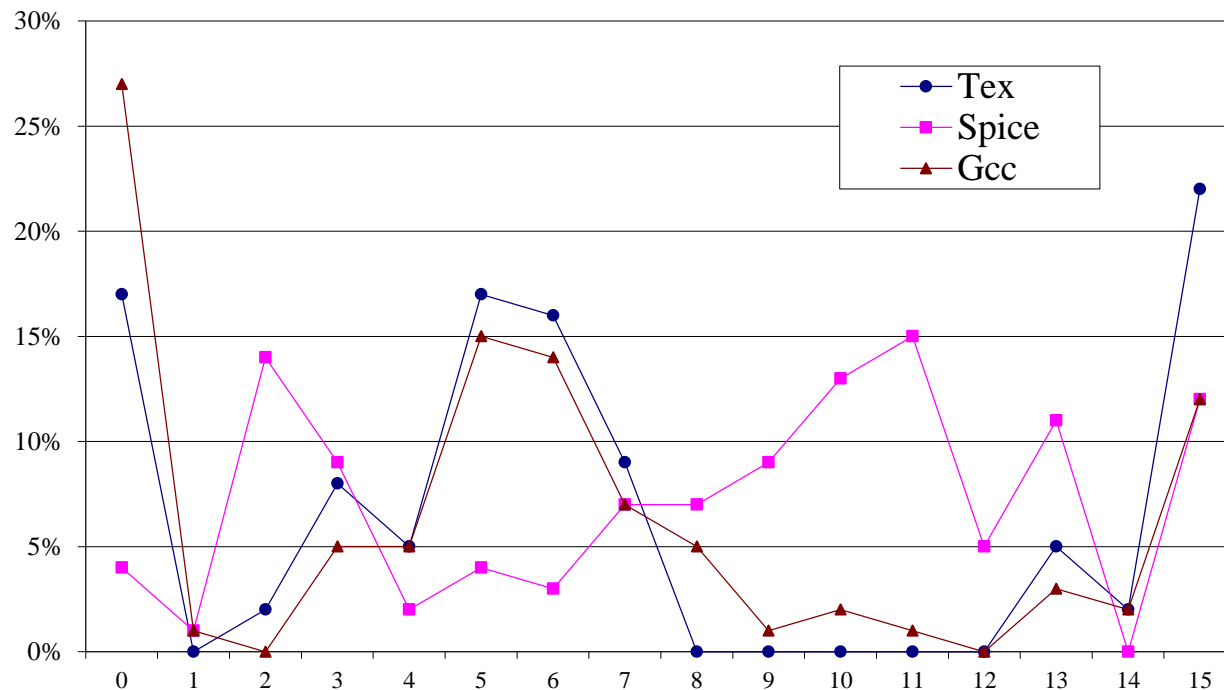
Programación

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

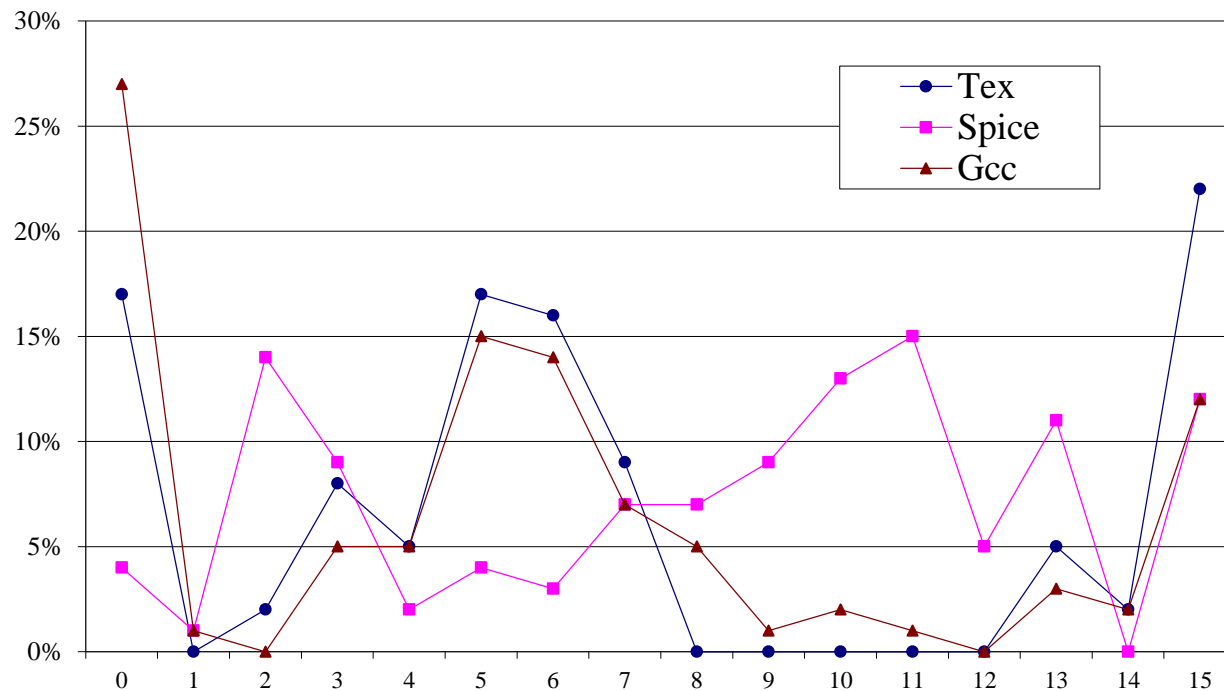
Programación

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

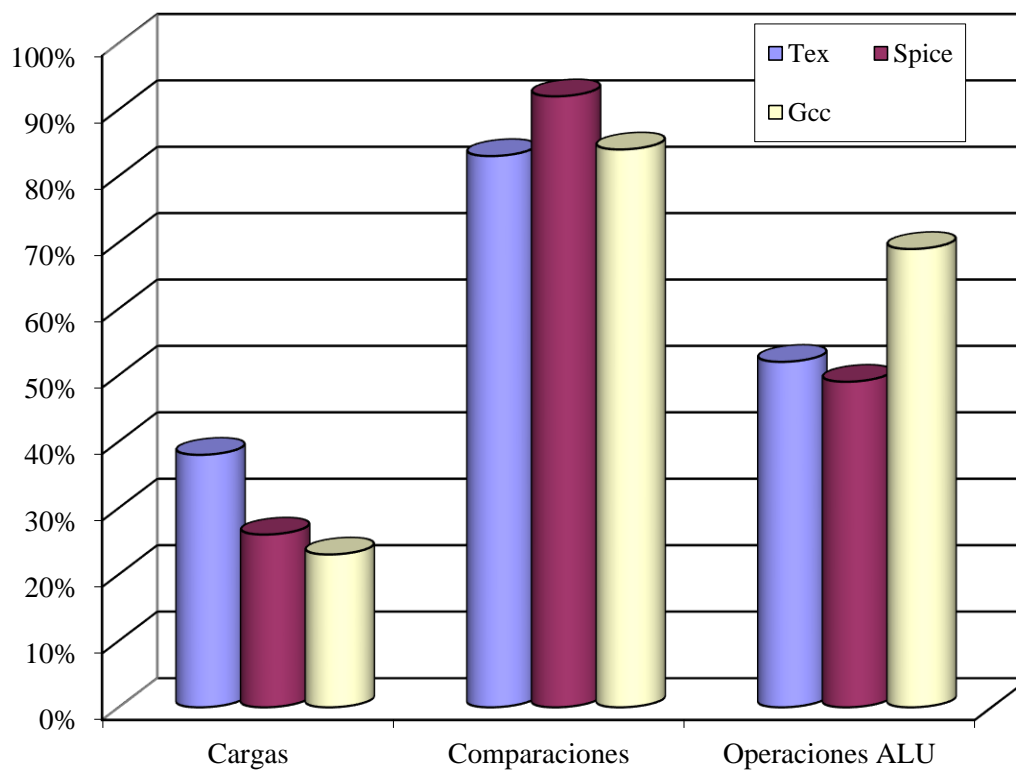
Programación

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

• **VAX (8,16,32)**

• **IBM360 (8)**

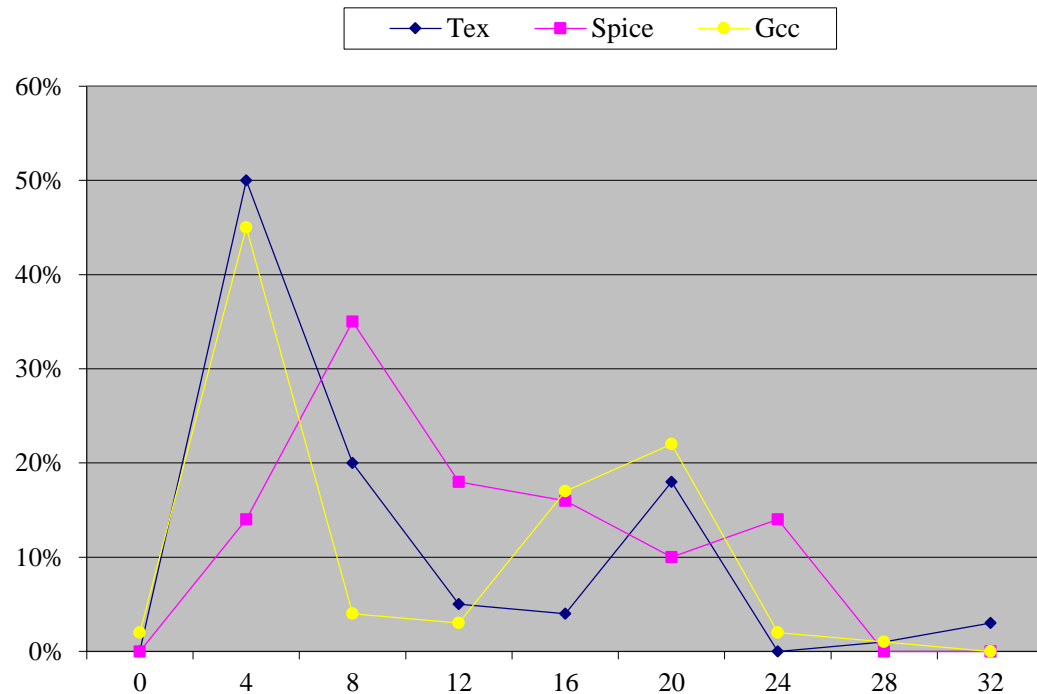
• **DLX (16)**

• **80x86 (8,16)**

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.



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 operación:** 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

3.2.2 Tipo y tamaño de los operandos

Introducción

Características

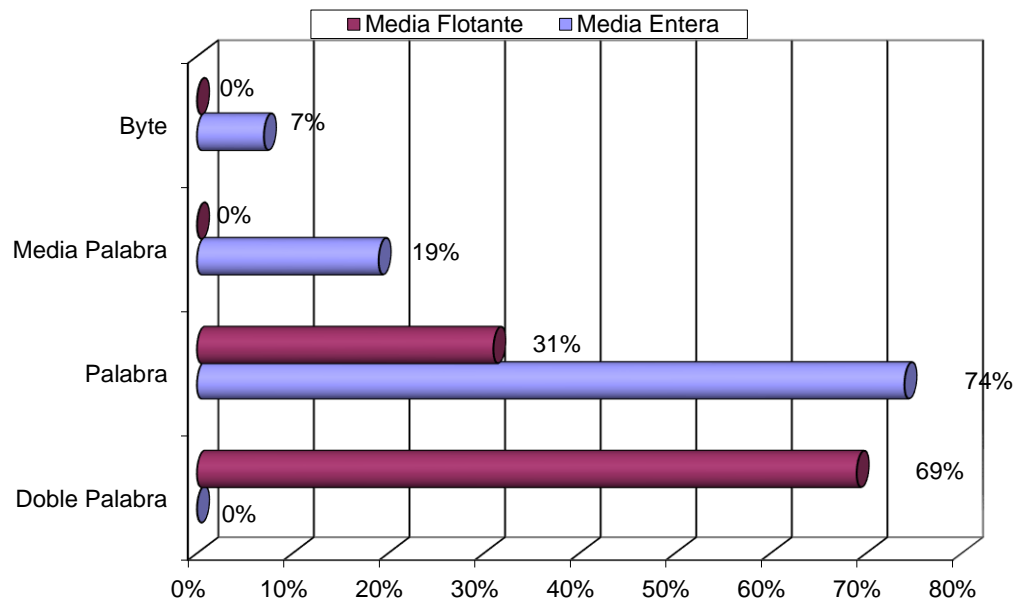
Programación

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

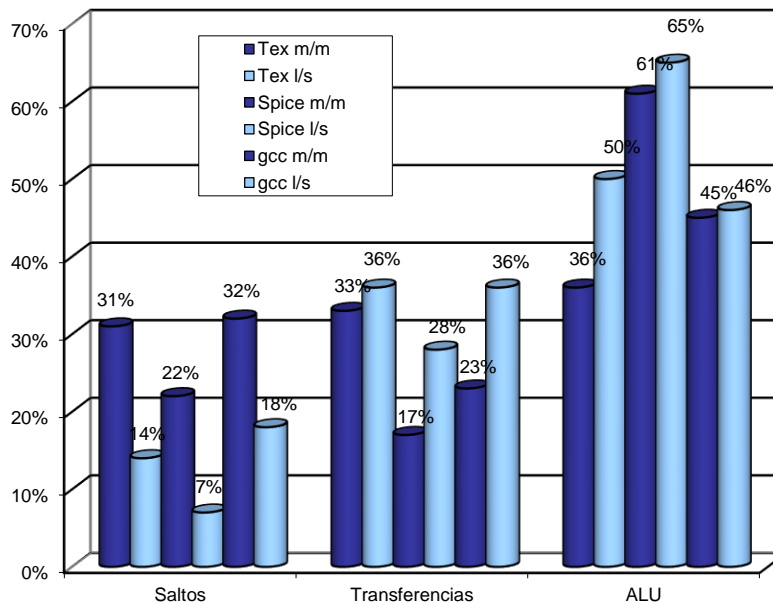
Programación

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

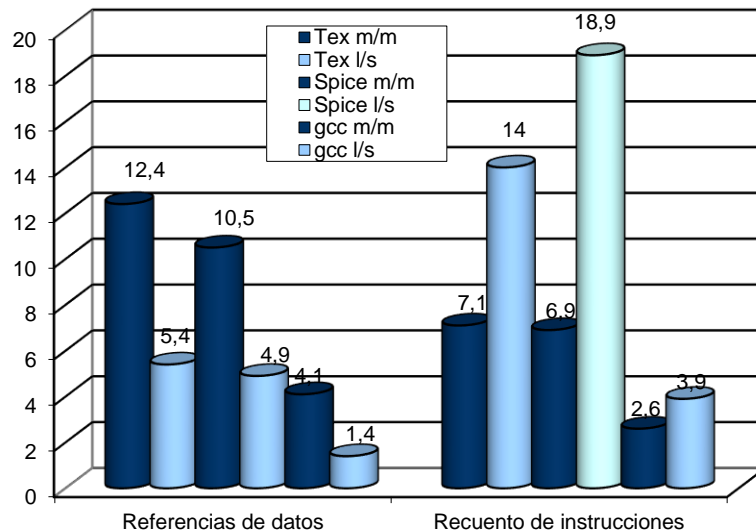
Programación

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

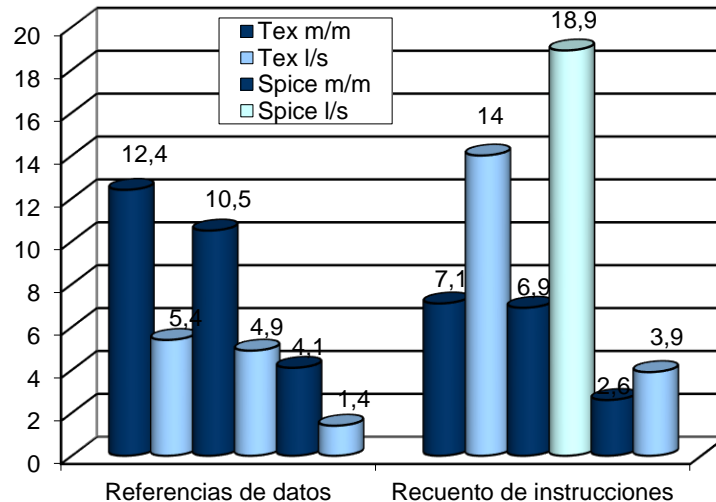
Programación

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

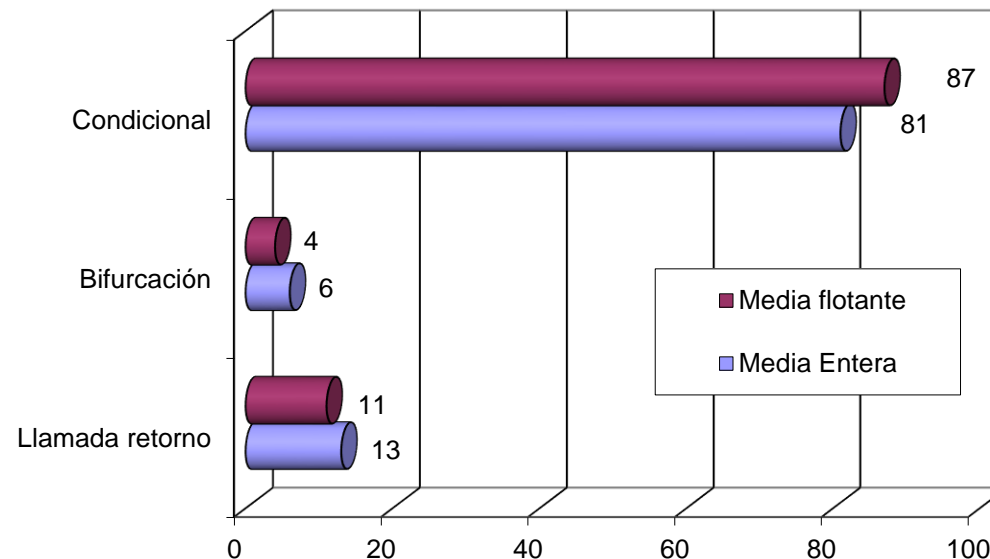
Programación

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

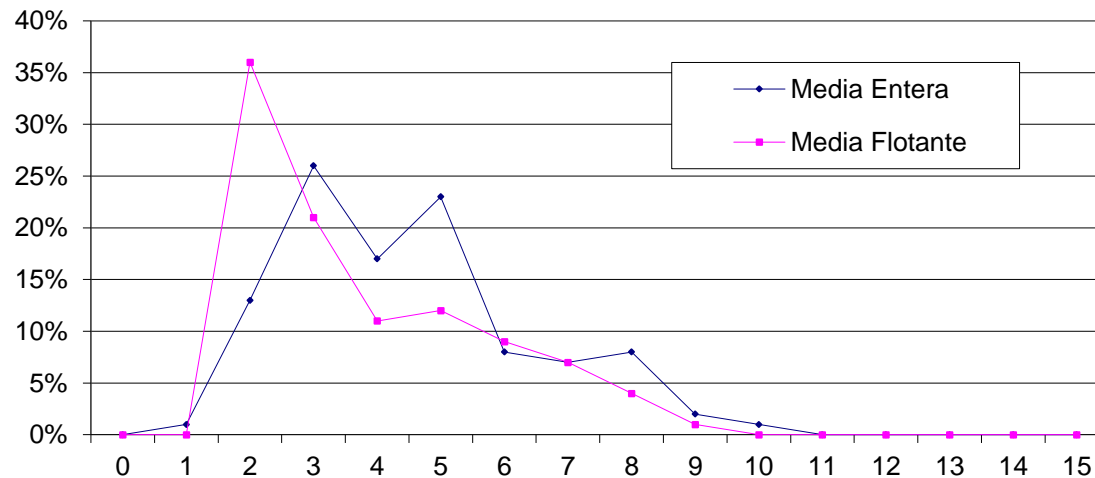
Programación

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<>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

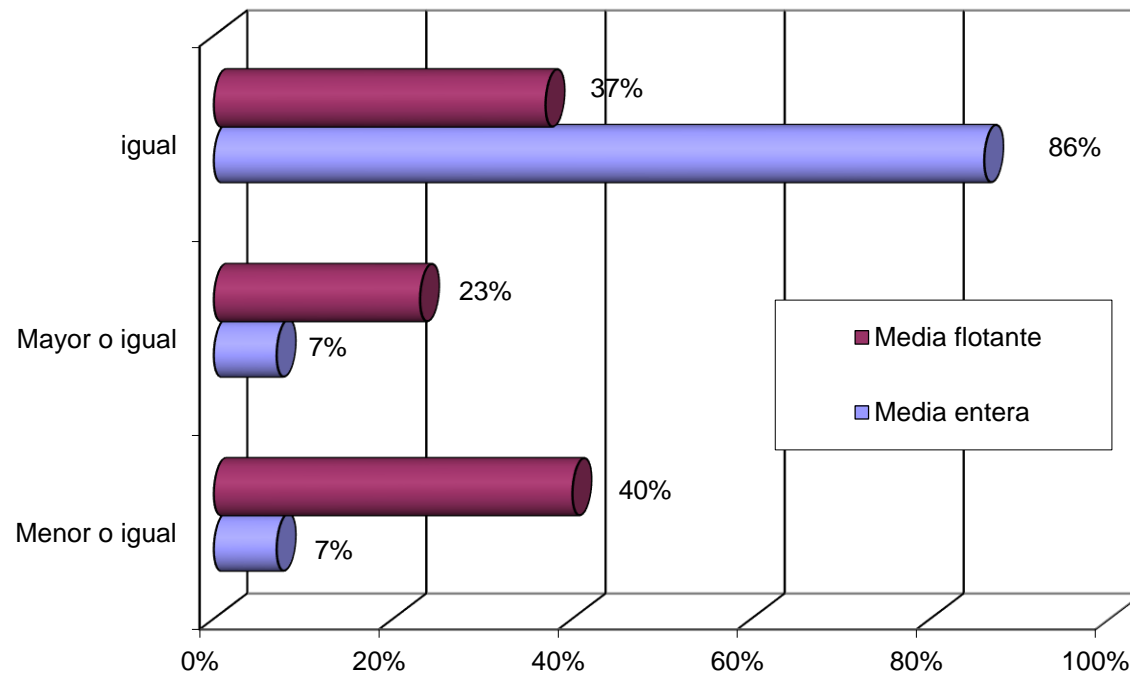
Programación

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

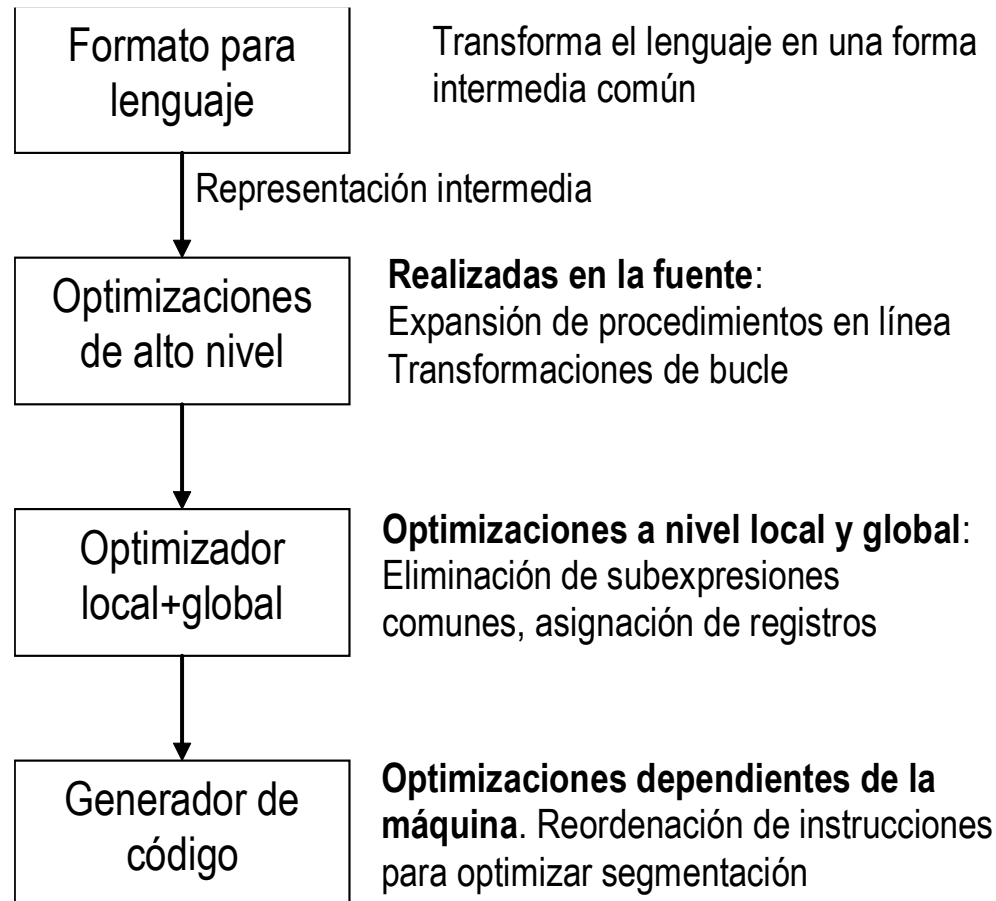
Programación

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
 - 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 proporciona 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

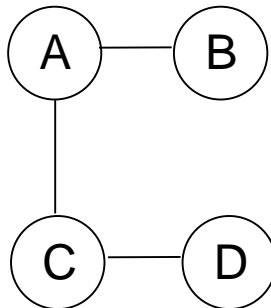
Coloreado de grafos

- 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

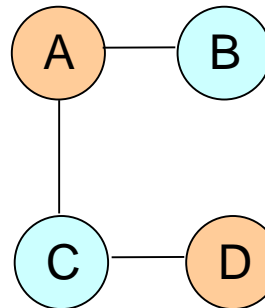
Programa

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

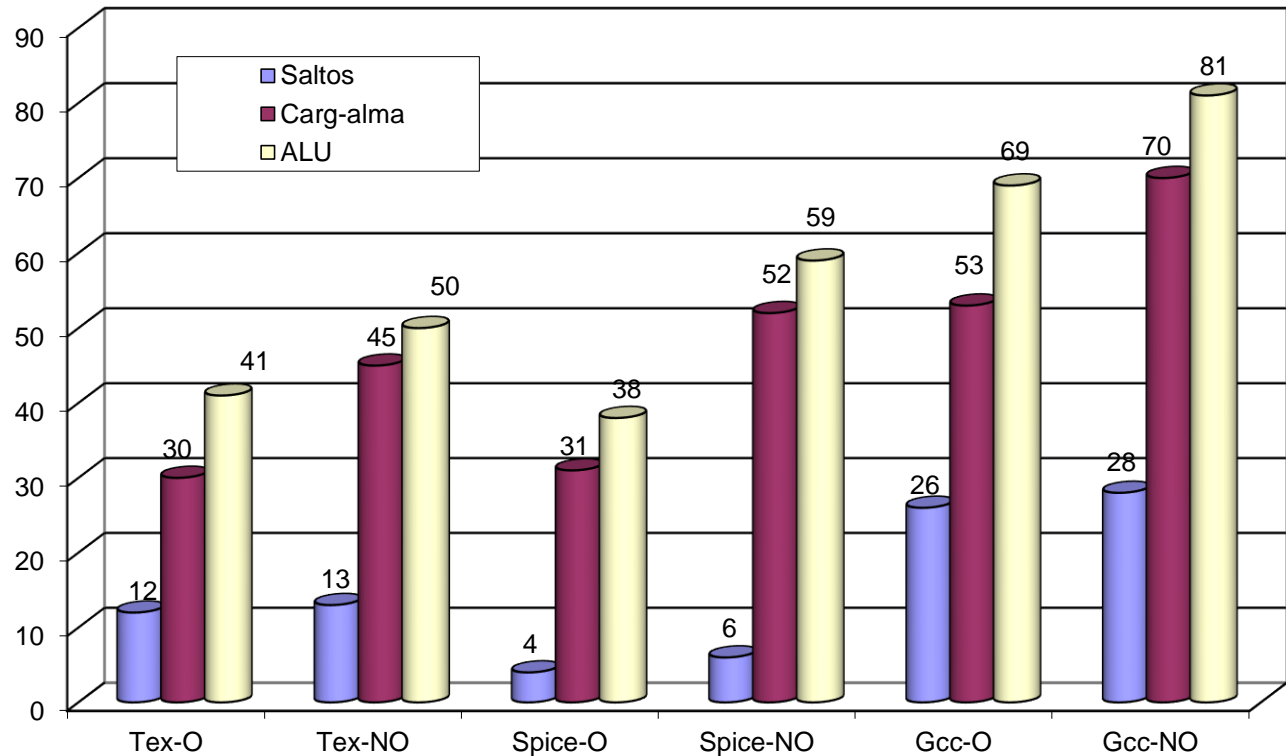
Programación

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

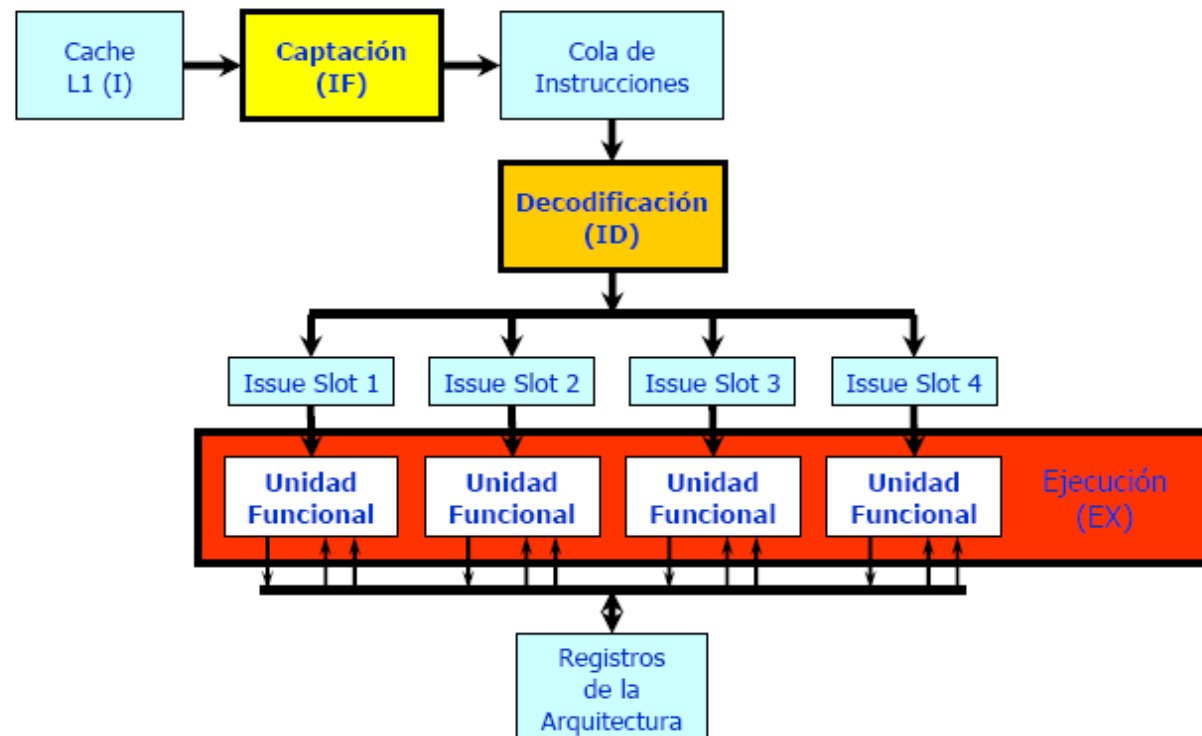
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

Evolución de la tecnología de computadores ha permitido:

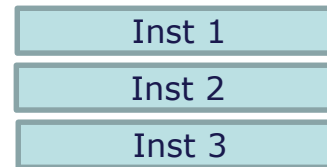
- Disponer de varias unidades de ejecución dentro del mismo procesador
- Los computadores actuales pueden ejecutar varias operaciones simultáneamente en esas unidades de ejecución



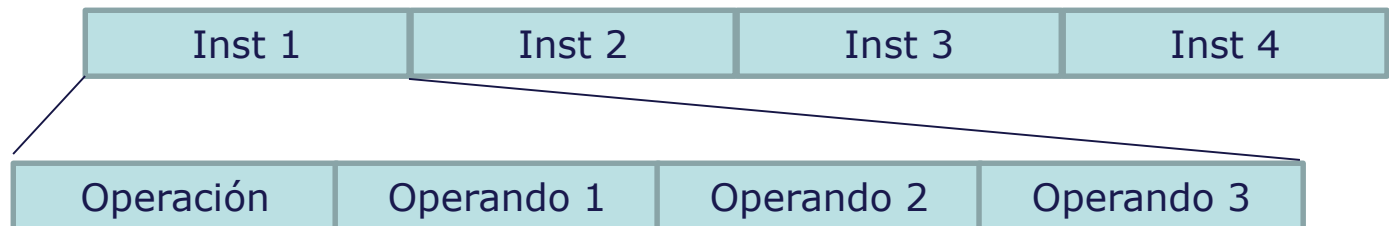
3.3.3 VLIW

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

Introducción

Características

Programación

Ejemplos

Diseño del
repertorio de
instrucciones

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

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

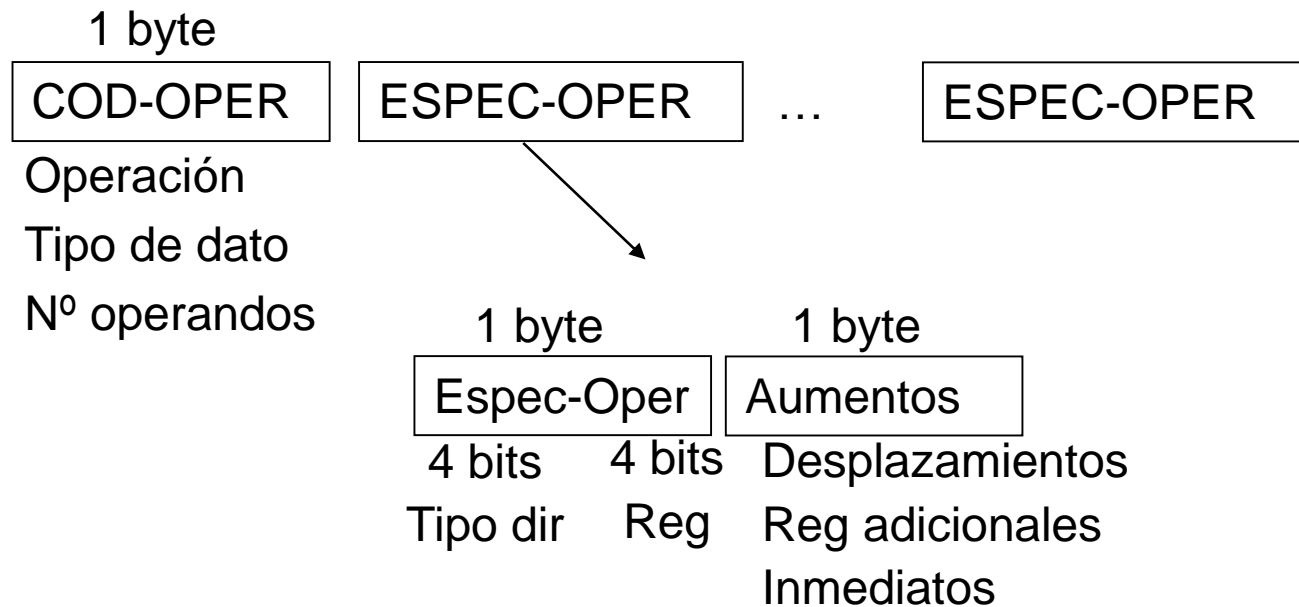
Características

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

• $\text{ADDL3 R1, 737(R2), \#456}$

$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$

$1 + 1 + (1+2) + (1+4)$

• 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

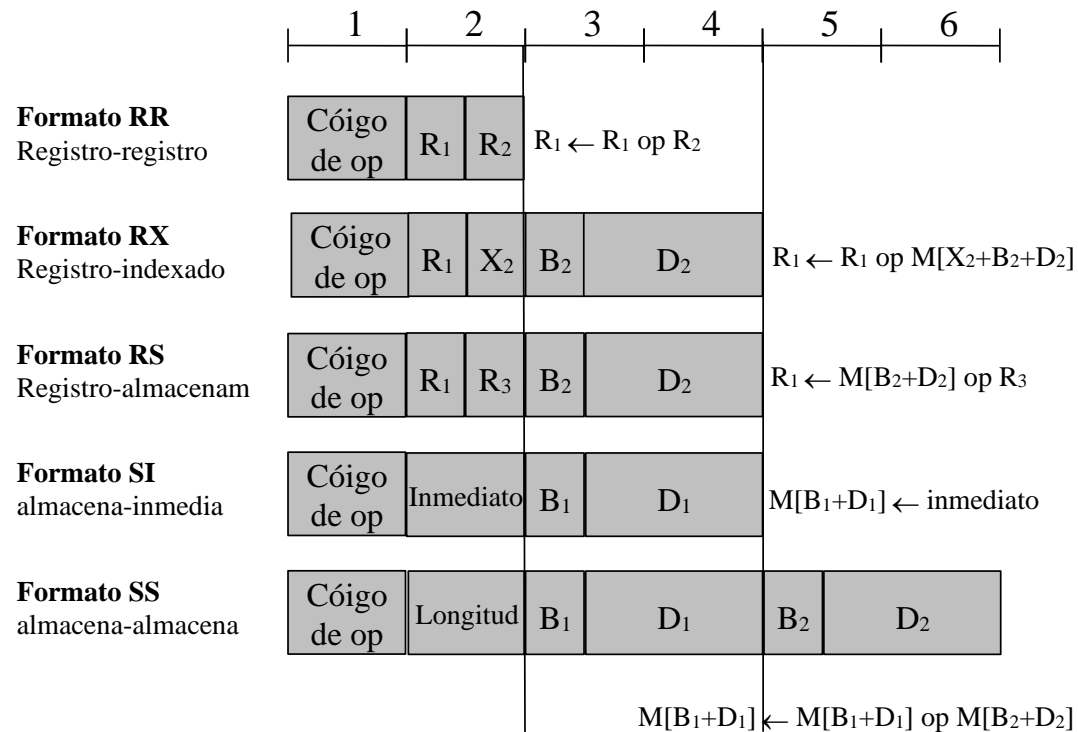
Programación

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

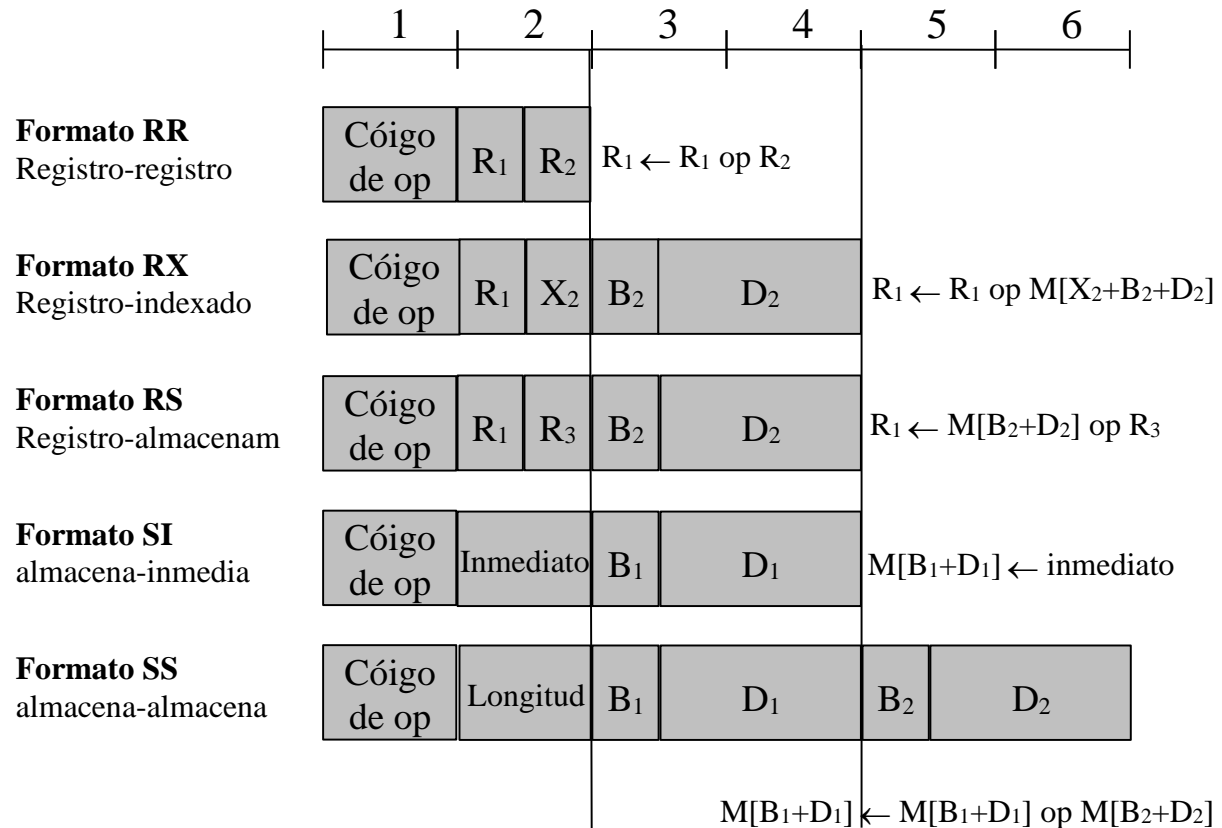
Características

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

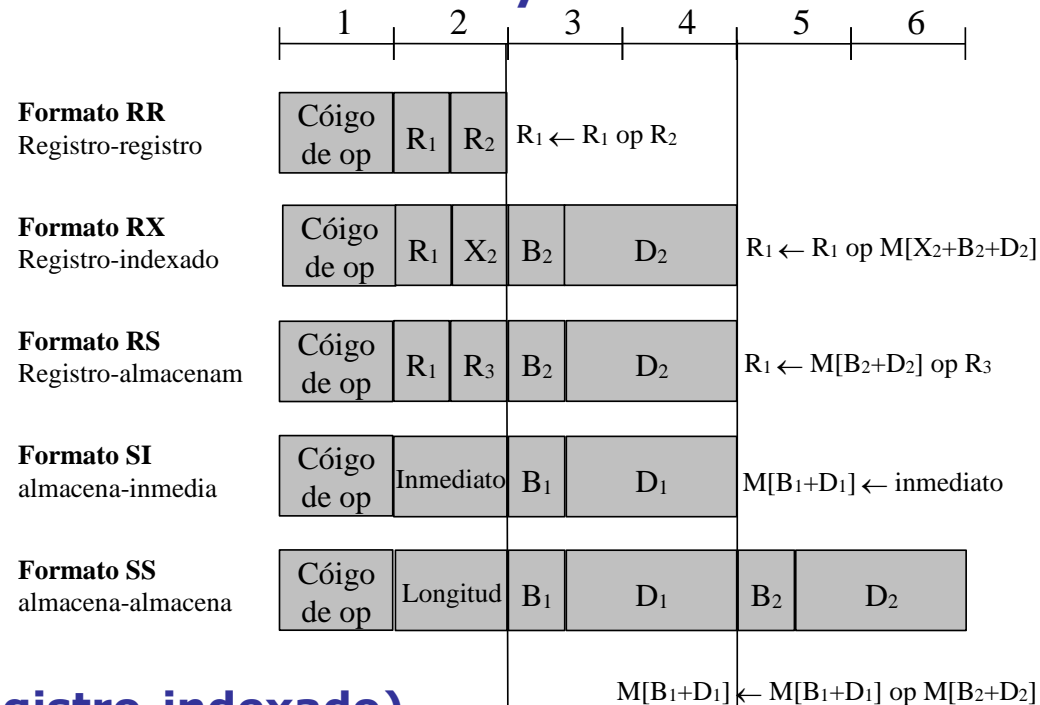
Características

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

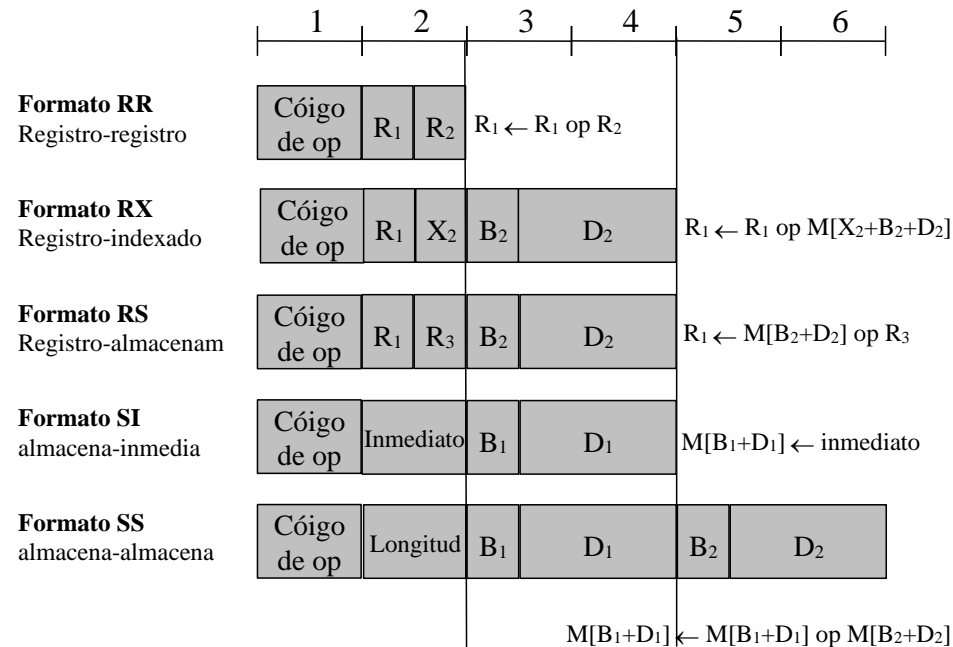
Características

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

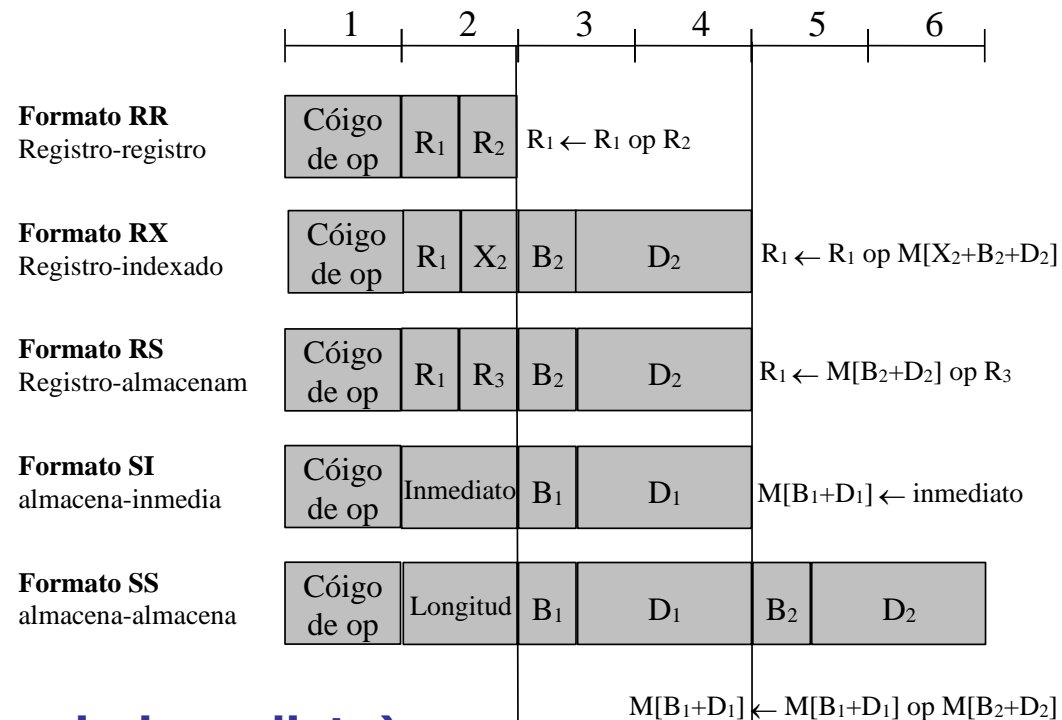
Características

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

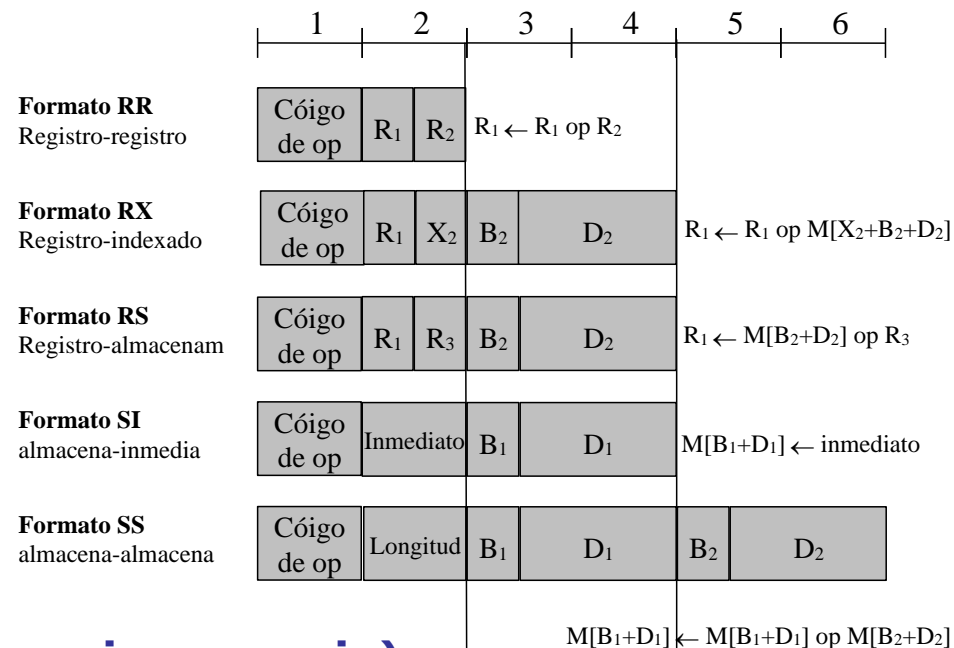
Características

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,i9 (2018):** de 4 a 18 cores

3.4.3 Intel 8086

Introducción

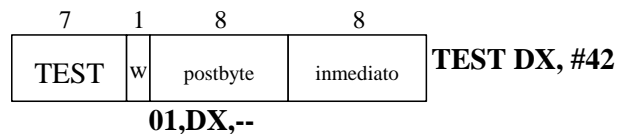
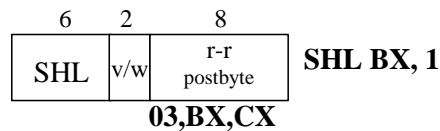
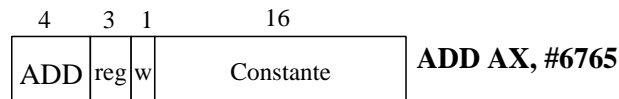
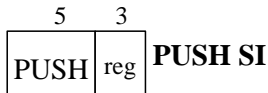
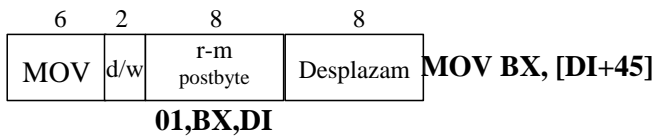
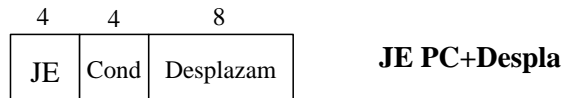
Características

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

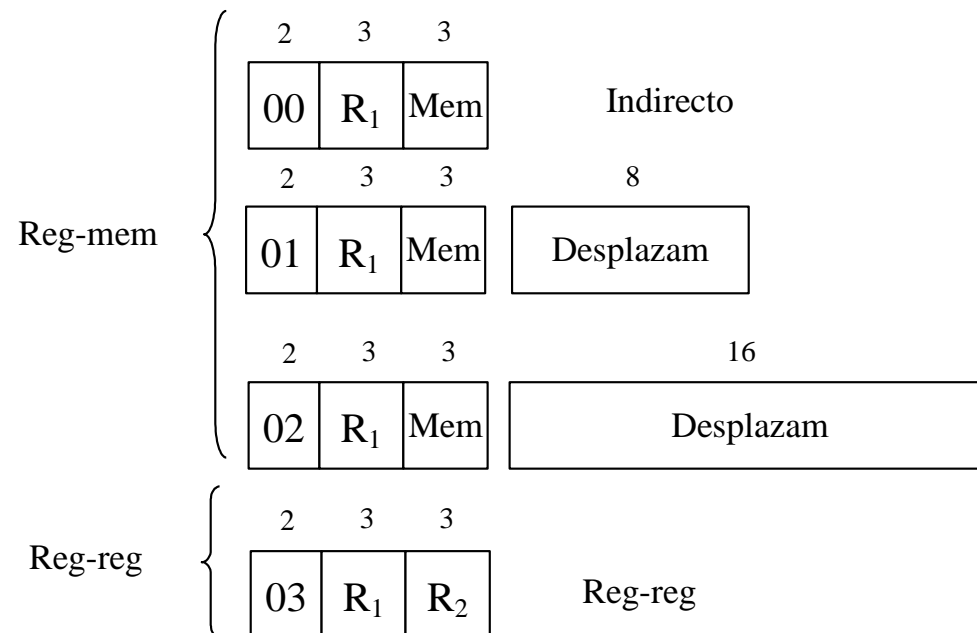
Ejemplos

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 humanos**
- ◆ AMD 29K, DECstation 3100, HP 850, IBM 801, Intel i860, MIPS 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	Multiplifica 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

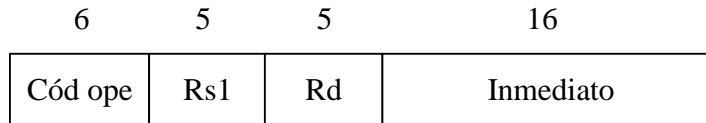
Programación

Ejemplos

Diseño del
repertorio de
instrucciones

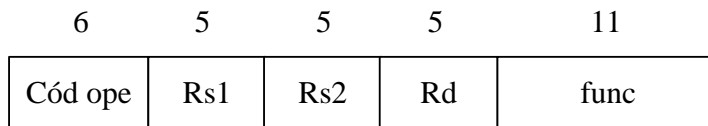
Codificación de las instrucciones

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

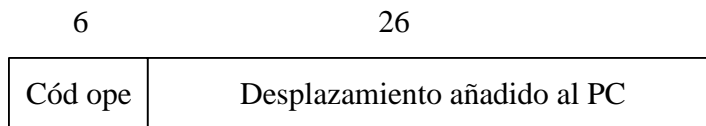
Instrucción tipo R



- 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