# Tema 02. Modos de direccionamiento y tipos de datos

## Módulo B. Arquitectura del procesador

Ingeniería Técnica en Informática

Facultad de Informática - Universidad Complutense de Madrid







## Introducción (1/2)

- El funcionamiento de un computador está determinado por las instrucciones que ejecuta.
- El área que estudia las características de ese conjunto de instrucciones se denomina arquitectura del procesador o arquitectura del repertorio de instrucciones y engloba los siguientes aspectos:



## Introducción (2/2)

- Formato de los datos
  - Tipos de datos que puede manipular el computador a nivel de lenguaje máquina
- Registros de la arquitectura
  - Conjunto de registros visibles al programador (de datos, direcciones, estado, PC)
- Modos de direccionamiento
  - Forma de especificar la ubicación de los datos y modos para acceder a ellos
- Repertorio de instrucciones
  - Operaciones que se pueden realizar y sobre qué tipos de datos actúan
- Formato de instrucción
  - Descripción de las diferentes configuraciones de bits que adoptan las instrucciones máquina

## Datos e instrucciones

- Los datos e instrucciones que manipula un programa se almacenan en la memoria
  - Temporalmente se pueden almacenar datos en los registros de la CPU para su manipulación: el acceso a un registro es mucho más rápido que el acceso a memoria
  - Eventualmente pueden solicitarse datos a los dispositivos de E/S

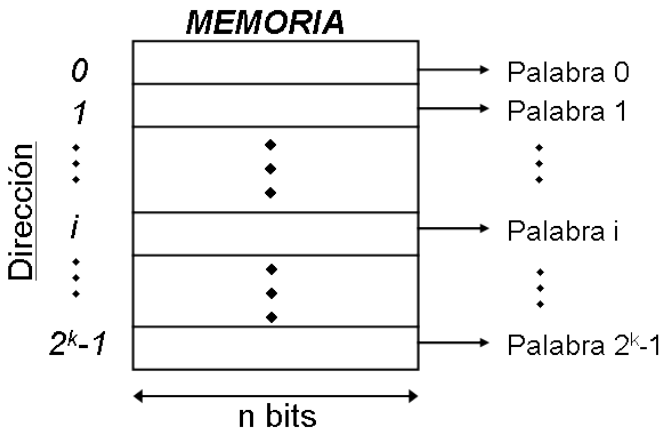


## Organización en palabras (1/4)

- La memoria se organiza en grupos de n bits llamados palabras de memoria
  - El ancho de palabra suele coincidir con el número de bits utilizados para representar números. Los computadores actuales utilizan anchos de palabra entre 16 y 64 bits
- Cada palabra de memoria tiene asignado un identificador llamado dirección de memoria
  - El rango de direcciones de memoria es un número entre 0 y $2^k - 1$
  - Esas $2^k$ direcciones distintas constituyen el espacio de direcciones del computador
  - Para especificar una dirección se necesitan $k$ bits de dirección



## Organización en palabras (2/4)






## Organización en palabras (3/4)

**Example**

Computador de 32 bits con 16 MBytes de memoria

- Ancho de palabra:
- Nº de palabras en memoria:
- Espacio de direcciones:
- Nº de bits de dirección necesarios:

# Organización en palabras (4/4)

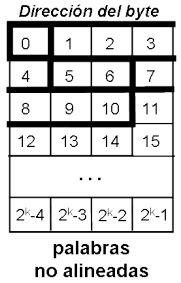
- Sin embargo, es posible que la arquitectura permita una mayor resolución en la información que es individualmente direccionable:
  - Se llama unidad direccionable, a la mínima cantidad de información que tiene una dirección única
  - Es común que en un computador de ancho de palabra 16, 32 ó 64 bits, el tamaño de la unidad direccionable sea de 1 byte (8 bits)
- Cuando el ancho de palabra y tamaño de la unidad direccionable no coinciden aparecen 2 problemas:
  - Alineamiento: cómo relacionar las direcciones de las palabras con las direcciones de las unidades direccionables
  - Ordenamiento: cómo repartir el contenido de una palabra en un conjunto consecutivo de unidades direccionables



# Alternativas de alineamiento (1/2)

## Palabras no alineadas

- Permitir que una palabra tenga cualquier dirección
  - La interfaz de memoria debe secuenciar los accesos (ya que podrán requerirse varios para obtener la información)



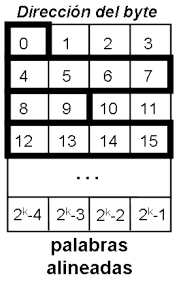

palabras no alineadas



# Alternativas de alineamiento (2/2)

## Palabras alineadas

- Limitar las direcciones de palabra según su tamaño
  - Los bytes en cualquier dirección, las palabras de 16 bits en direcciones pares, las de 32 bits en direcciones múltiplos de 4...
  - Se desperdicia memoria




palabras alineadas



# Alternativas de ordenamiento (1/5)

## Endianness

- Endianness: Formato en el que se almacenan los datos de más de un byte
  - Little-endian: El byte menos significativo se almacena en la dirección baja de memoria
    - Intel x86
  - Big-endian: El byte más significativo se almacena en la dirección baja de memoria
    - Motorola, SPARC(9)
  - Bi-endian: Se permite seleccionar qué ordenamiento utilizar
    - En ocasiones se puede modificar vía software, otras veces viene preconfigurado en la placa base
    - ARM, PowerPC (pero no el PPC970/G5), DEC Alpha, SPARC V9, MIPS, PA-RISC e IA64

# Alternativas de ordenamiento (2/5)

## Ejemplo

**Example**

Queremos almacenar en la dirección base ADDR, el número 12345678 codificado en BCD:

- La memoria es direccionable por bytes
- 1 dígito BCD ocupa 4 bits
- 2 dígitos BCD por byte: 12-34-56-78



# Alternativas de ordenamiento (3/5)

## ¿Cuál es mejor?

- Little-Endian
  - Instrucciones ensablador diseñadas para obtener números de 1, 2, 4, o más bytes es sencillo: Se comienza por la dirección base y desplazamiento 0, 1, 2, etc.
  - Debido a la relación 1:1 entre desplazamiento en memoria y número de byte (desp. 0 es el byte 0), las rutinas matemáticas de precisión múltiple son sencillas de escribir
- Big-endian
  - Es muy sencillo saber si un número es positivo o negativo (mirando el bit con desplazamiento 0)
  - El dato está almacenado en el sentido en que se imprime



# Alternativas de ordenamiento (4/5)

## Curiosidades

- En *Los Viajes de Gulliver* los habitantes de Lilliput y Blefuscu discuten la manera como comer los huevos, empezando por el lado mas gordo (big-endian) o empezando por el lado más pequeño (little-endian)
- .BMP de Windows, dado que fue desarrollado para una arquitectura Little-endian, insiste en este formato
  - Si desde una máquina 'Big-endian' queremos generar un archivo .BMP, hay que invertir el orden de los bytes
- Formatos Little-endian: BMP (Windows and OS/2 Bitmaps), GIF y QTM (Quicktime Movies)
- Formatos Big-endian: Adobe Photoshop, JPEG, SGI (Silicon Graphics) y WPG (WordPerfect Graphics Metafile)
- Formatos Bi-endian: TIFF (formato codificado en un byte) y Microsoft RIFF (.WAV & .AVI)



# Alternativas de ordenamiento (5/5)

## Curiosidades

**Example**

Detectar si una máquina es little-endian o big-endian

```
#include <stdio.h>
int main(void) {
  int i = 1;
  char* p = (char*)&i;
  if ( p[0] == 1 )
    printf("Little Endian\n");
  else
    printf("Big Endian\n");
  return 0;
}
```

# Contenidos de la memoria (1/6)

- Palabra en memoria: Colección de 0's y 1's
- Es el computador el encargado de interpretar lo que representa en cada momento



# Contenidos de la memoria (2/6)

## Una palabra puede representar...

- Datos
  - Numéricos
    - Enteros (Magnitud y signo, C1, C2, ...)
    - Reales (Punto fijo, Punto flotante, ...)
    - Decimales (BCD, Exceso-3, ...)
  - Booleanos o lógicos
    - TRUE, FALSE
  - Caracteres
    - Letras, dígitos decimales, signos puntuación
    - Representación más común: ASCII
- Instrucciones
  - Código de operación + Información de direccionamiento



# Contenidos de la memoria (3/6)

## Ejemplos: Palabra de 32 bits

**Example**

Un número entero entre $-(2^{31}-1)$ y $(2^{31}-1)$ codificado en magnitud y signo

| $b_{31}$ | $b_{30}$ | ... | ... | $b_1$ | $b_0$ |
|---|---|---|---|---|---|

donde:

- $b_{31}$ es el bit de signo
  - $b_{31} = 0$ para números positivos
  - $b_{31} = 1$ para números negativos
- Magnitud: $b_{30} \times 2^{30} + \ldots + b_1 \times 2^1 + b_0 \times 2^0$



# Contenidos de la memoria (4/6)

## Ejemplos: Palabra de 32 bits

**Example**

32 dígitos booleanos o lógicos

| $b_{31}$ | $b_{30}$ | ... | ... | $b_1$ | $b_0$ |
|---|---|---|---|---|---|

donde:

- $b_k = 0 \Rightarrow FALSE$
- $b_k = 1 \Rightarrow TRUE$

# Contenidos de la memoria (5/6)

## Ejemplos: Palabra de 32 bits

**Example**

Cuatro caracteres ASCII

| 8 bits | 8 bits | 8 bits | 8 bits |
|---|---|---|---|
| Carácter ASCII | Carácter ASCII | Carácter ASCII | Carácter ASCII |



# Contenidos de la memoria (6/6)

## Ejemplos: Palabra de 32 bits

**Example**

Una instrucción máquina

| 8 bits | 24 bits |
|---|---|
| Código de operación | Información de direccionamiento |

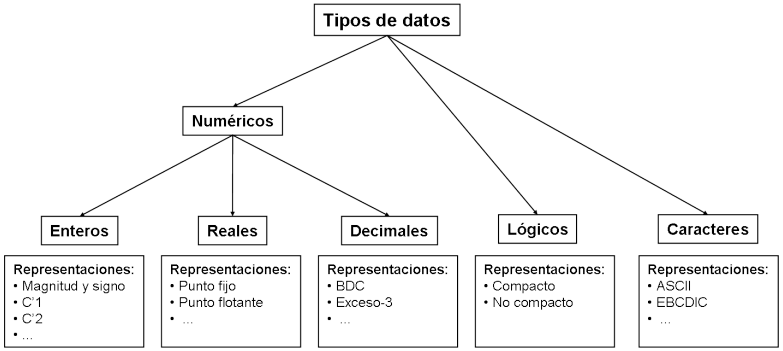


# Caracterización de los tipos de datos (1/2)

- Un tipo de datos es una sucesión de bits caracterizada por dos propiedades:
  - Su dominio que limita el rango y la precisión. Es el conjunto de valores que el dato puede tomar y depende de:
    - El tipo de representación
    - El tamaño o número de bits utilizado para la representación
- Las operaciones que pueden realizarse sobre ese dato
- Se dice que una arquitectura soporta un determinado tipo de datos si
  - Tiene asignada al menos una representación de ese tipo de datos
  - Dispone de un conjunto de operaciones para manipular esa representación



# Caracterización de los tipos de datos (2/2)



## Representación de datos numéricos enteros (1/2)

- Magnitud y signo (MS)
  - Simétrico
  - Dos representaciones para el cero:
    - $+0 \rightarrow 000\ldots00$
    - $-0 \rightarrow 100\ldots00$
  - Rango: $-(2^{n-1}-1) \leq x \leq +(2^{n-1}-1)$
- Complemento a 1 (C1)
  - Simétrico
  - Dos representaciones para el cero:
    - $+0 \rightarrow 000\ldots00$
    - $-0 \rightarrow 111\ldots11$
  - Rango: $-(2^{n-1}-1) \leq x \leq +(2^{n-1}-1)$
- Complemento a 2 (C2)
  - No simétrico
  - Una representación para el cero
  - Rango: $-(2^{n-1}) \leq x \leq +(2^{n-1}-1)$



## Representación de datos numéricos enteros (2/2)

**Example**

Ejemplo: Tamaño = 4 bits

| Positivos | | | |
|---|---|---|---|
| $b_3b_2b_1b_0$ | MS | C1 | C2 |
| 0000 | 0 | 0 | 0 |
| 0001 | 1 | 1 | 1 |
| 0010 | 2 | 2 | 2 |
| 0011 | 3 | 3 | 3 |
| 0100 | 4 | 4 | 4 |
| 0101 | 5 | 5 | 5 |
| 0110 | 6 | 6 | 6 |
| 0111 | 7 | 7 | 7 |

| Negativos | | | |
|---|---|---|---|
| $b_3b_2b_1b_0$ | MS | C1 | C2 |
| 1000 | -0 | -7 | -8 |
| 1001 | -1 | -6 | -7 |
| 1010 | -2 | -5 | -6 |
| 1011 | -3 | -4 | -5 |
| 1100 | -4 | -3 | -4 |
| 1101 | -5 | -2 | -3 |
| 1110 | -6 | -1 | -2 |
| 1111 | -7 | -0 | -1 |



## Punto fijo

- Un nº fijo de bits representa la parte entera y otro la decimal
- El punto decimal se coloca entre ambas partes en un lugar fijo
- El rango representable es muy limitado
- Puede utilizar aritmética entera

**Example**

| Signo | Parte entera | Parte decimal |
|---|---|---|
| $b_{31}$ | $b_{30}$ ... ... $b_{10}$ | $b_9$ ... $b_0$ |
| 1 bit | 21 bits | 10 bits |

$$(-1)^{b_{31}} \cdot \left(b_{30} \cdot 2^{20} + b_{29} \cdot 2^{19} + \ldots + b_9 \cdot 2^{-1} + \ldots + b_0 \cdot 2^{-10}\right)$$



## Punto flotante

- Un número N se representan mediante un signo, una mantisa (M) y un exponente (E)
  - $N = \pm M \times B^E$
- La base es implícita y es común para todos los números
  - No necesita almacenarse (normalmente $B = 2$)
- Existen multitud de asignaciones posibles de bits a la mantisa y al exponente
- Requiere de una aritmética propia

**Example**

| Signo | Exponente | Mantisa |
|---|---|---|
| $b_{31}$ | $b_{30}$ ... $b_{23}$ | $b_{22}$ ... ... $b_0$ |
| 1 bit | 8 bits | 23 bits |

$$N = (-1)^{b_{31}} \cdot 1.M \cdot 2^E, E = b_{30} \cdot 2^7 + \ldots + b_{23} \cdot 2^0$$

# Binary-coded decimal (BCD)

- Cada dígito decimal se codifica mediante un grupo de 4 bits, posibles representaciones:
  - BCD: los dígitos decimales se codifican mediante su equivalente binario
  - Exceso-3: los dígitos decimales se codifican sumando 3 a su equivalente binario
  - Otros: Aiken, 5421, etc.
- Ventajas
  - Cada codificación BCD facilita ciertas operaciones aritméticas
  - Se pueden mostrar fácilmente en visualizadores 7-segmentos
  - La BIOS almacena la hora en BCD (¿razones históricas?)
- Desventajas
  - Sólo se utilizan 10 de las 16 posibles combinaciones (4 bits)



# BCD y Exceso-3

| Decimal | BCD | Exceso-3 |
|---|---|---|
| 0 | 0000 | 0011 |
| 1 | 0001 | 0100 |
| 2 | 0010 | 0101 |
| 3 | 0011 | 0110 |
| 4 | 0100 | 0111 |
| 5 | 0101 | 1000 |
| 6 | 0110 | 1001 |
| 7 | 0111 | 1010 |
| 8 | 1000 | 1011 |
| 9 | 1001 | 1100 |



# Empaquetamiento

- Como los computadores almacenan los datos en conjuntos de bytes, hay dos maneras comunes de almacenar los datos BCD
  - Decimal no empaquetado: Un dígito por byte

| X | Dig. 1 | X | Dig. 2 | X | Dig. 3 |
|---|---|---|---|---|---|
| Byte 1 | | Byte 2 | | Byte 3 | |

  - Decimal empaquetado: Dos dígitos por byte

| Dig. 1 | Dig. 2 | Dig. 3 | Dig. 4 | Dig. 5 | Dig. 6 |
|---|---|---|---|---|---|
| Byte 1 | | Byte 2 | | Byte 3 | |



# Representación de caracteres

- Los caracteres son necesarios para representación de la información escrita
  - Letras del alfabeto (a, b, ..., z, A, B, ..., Z, á, é, ..., ü, ...)
  - Signos de puntuación (, ; : . ¿? ¡! -)
  - Caracteres numéricos (0, 1, ..., 9)
- Existen diversas alternativas de codificación (7 u 8 bits por carácter)
  - ASCII: American Standard Code for Information Interchange
  - EBCDIC: Extended Binary Coded Decimal Interchange Code

# Representación de información lógica o booleana

- Para representar información lógica se necesita un único bit
  - TRUE = 1
  - FALSE = 0
- Representación compacta
  - 8 dígitos booleanos por byte
  - Necesidad instrucciones complejas para acceder al dato booleano
- Representación no compacta
  - 1 dígito booleano por byte
  - Instrucciones de acceso sencillas pero se desperdician 7 bits



# Ejemplo - MC68000

**Example**

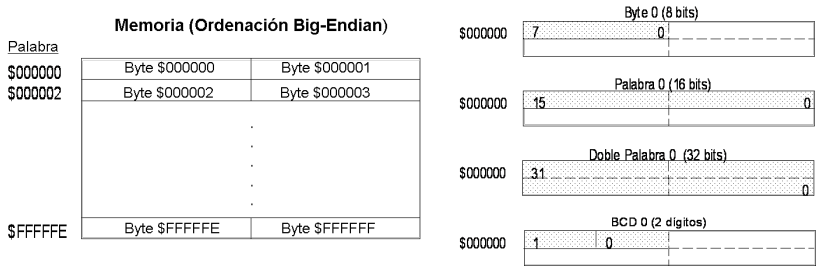
Algunos tipos de datos

- Enteros sin signo o con signo (complemento a 2)
  - Tamaño Byte (8 bits) → se representa como .b (byte)
  - Tamaño Palabra (16 bits) → se representa como .w (word)
  - Tamaño Doble Palabra o Palabra Larga (16 bits) → se representa como .l (long word)
- Decimal BCD empaquetado (dos dígitos BCD por byte)



# Ejemplo - MC68000

Organización de la memoria






# Registros

Dentro del procesador hay un conjunto de registros que constituyen el nivel más alto en la jerarquía de memoria, son de dos tipos:

- Registros visible por el usuario: Permiten al programador en ensamblador minimizar las referencias a memoria principal.
- Registros de control y de estado: Utilizados por la unidad de control para controlar el funcionamiento del procesador o por el sistema operativo.

# Registros visibles por el usuario

- Registros de uso general: Pueden ser asignados por el programador a distintas funciones. Conterner operando, dirección del mismo, etc.
- Registros de datos: Pueden usarse únicamente para contener datos.
- Registros de direcciones: De uso más o menos general, o pueden estar dedicados a modos de direccionamiento concretos
  - Punteros de segmento: Para direccionamiento segmentado, contiene la dirección base del segmento
  - Registros índice: Se usan para direccionamiento indexado
  - Puntero de pila: Apunta a la cabecera de la pila
- Registros de condición: Tambien llamados flags, almacenan información del resultado de operaciones (positivo, cero, negativo, etc.)



# Registros de control y de estado
## Registros de control

La mayoría de estos registros no son visibles por el usuario. Cada familia tiene un conjunto de registros distinto. Una posible generalización es

- Contador de programa, PC: Dirección de la instrucción a captar
- Registro de instrucción, IR: Instrucción captada más recientemente
- Registro de dirección de memoria, MAR: Dirección de una posición de memoria
- Registro intermedio de memoria, MDR: Contiene los datos a escribir en memoria o el dato leído más recientemente



# Registros de control y de estado
## Registro de estado

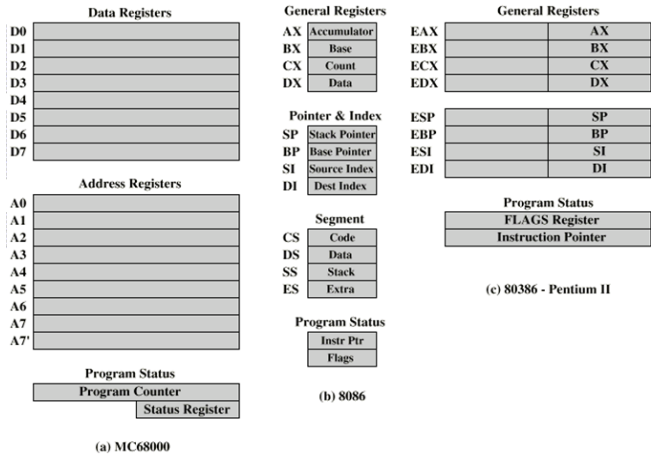
Muchos procesadores incluyen un registro o un conjunto de ellos, conocido como palabra de estado de programa (PSW, program status word), los indicadores más comunes son:

- Signo: Bit de signo resultado de la última operación aritmética
- Cero: Puesto a uno cuando el resultado es 0
- Acarreo: 1 si la operación produce acarreo o adeudo, se usa en operaciones aritméticas multipalabra
- Igual: 1 si el resultado de una operación lógica es la igualdad
- Desbordamiento: Usado para indicar desbordamiento aritmético
- Interrupciones habilitadas/desabilitadas: Para permitir o deshabilitar interrupciones
- Supervisor: Indica si el procesador funciona en modo supervisor o usuario. En modo supervisor se pueden ejecutar instrucciones privilegiadas



# Registros de control y de estado
## Ejemplo

No hay un esquema universalmente aceptado sobre la mejor forma de organizar los registros del procesador.

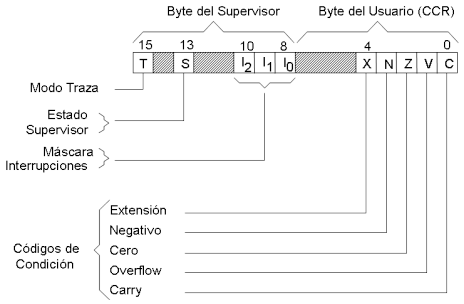


# Registros de control y de estado
## Ejemplo - MC68000

- Registros de datos (D0-D7)
  - Para operaciones con datos
  - Tamaños: .b, .w, .l
- Registros de direcciones (A0-A6)
  - Para operaciones con direcciones
  - Tamaños: .w, .l
- Punteros de pila (A7, A7')
  - Uno para usuario (A7=USP)
  - Otro para supervisor (A7'=SSP)
- Registro de Estado (SR)
  - Parte más significativa: sólo accesible en modo supervisor
  - Parte menos significativa: códigos de condición (CCR)
- Contador de programa (CP)
  - Apunta a la siguiente instrucción a ejecutar

Tema 02. Modos de direccionamiento y tipos de datos
Registros y pila de la arquitectura
Registros

# Registros de control y de estado
## Ejemplo - MC68000




- C ('Carry' o acarreo): Activo si arrastre de suma o de resta
- V ('oVerflow' o desbordam.): Activo si resultado operación no se puede expresar en C2 dentro del tamaño seleccionado
- Z ('Zero' o cero): Activo si resultado es cero
- N (Negativo): Activo si resultado es negativo
- X (eXtensión): Activo si arrastre decimal en operaciones BCD

Tema 02. Modos de direccionamiento y tipos de datos
Registros y pila de la arquitectura
Registros

# Registros de control y de estado
## Ejemplo - 8086

- Registros generales
  - AX: Registro acumulador, operaciones de E/S y la mayor parte de la aritmética
  - BX: Registro base, puede ser un índice para direccionamiento indexado, también es común emplear el BX para cálculos
  - CX: Registro contador, puede contener un valor para controlar el número de veces que un ciclo se repite o un valor para desplazamiento de bits.
  - DX: Registro de datos, operaciones de E/S, y operaciones de multiplicación y división con cifras grandes que suponen al DX y AX trabajando juntos
- Registros índice
  - Los registros SI y DI están disponibles para direccionamiento indexado y para sumas y restas

Tema 02. Modos de direccionamiento y tipos de datos
Registros y pila de la arquitectura
Registros

# Registros de control y de estado
## Ejemplo - 8086

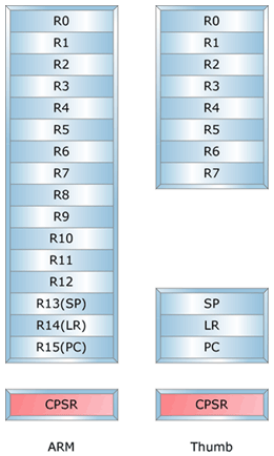
- Registros apuntadores
  - Los registros SP (apuntador de pila) y BP (apuntador base) permiten al sistema el acceso a los datos en el segmento de la pila
- Registros de segmento
  - Definen áreas de 64 KB dentro del espacio de direcciones de 1 MB del 8086. Estas áreas pueden solaparse total o parcialmente. No es posible acceder a una posición de memoria no definida por algún segmento
- Registro apuntador de instrucciones
  - El registro IP de 16 bits contiene el desplazamiento de dirección de la siguiente instrucción que se ejecuta

# Registros de control y de estado
## Ejemplo - 8086

- Registro de estado
  - Los habituales (OF (overflow), IF (interrupción), SF (signo), ZF (cero), CF (acarreo)), todos de un bit
  - DF (dirección): Controla la selección de incremento o decremento de los registros SI o DI en las operaciones con cadenas de caracteres (1=decremento automático; 0=incremento)
  - TF (traza): Permite la operación del procesador en modo de depuración (paso a paso)
  - AF (acarreo auxiliar): Contiene un acarreo externo del bit 3 en un dato de 8 bits, para aritmética especializada
  - PF (paridad): Indica paridad par (1) o impar (0) en una operación de datos de ocho bits



# Registros de control y de estado
## Ejemplo - ARM




- 15 registros de propósito general (R0-R14)
- R13: Puntero de pila
- R14: Retorno de una función
- R15: Contador de programa
- CPSR: Registro de estado
- Thumb: Thumb instruction set



# Registros de control y de estado
## Ejemplo - ARM

- Registro de estado
  - Los habituales (N (Negative), Z (Zero), C (Carry), V (oVerflow))
  - I (IRQ): Activa las interrupciones
  - F (FIQ): Activa las interrupciones en modo rápido
  - T (Thumb): Repertorio reducido de instrucciones
  - modo: Indica el modo de ejecución: USR (modo usuario), FIQ (modo IRQ rápida), IRQ (modo IRQ), SVC (modo supervisor), ABT (modo cancelar), UND (modo no definido)




# Pila del sistema

- Es una zona de memoria para almacenar datos y/o instrucciones de forma temporal
- El acceso a la pila es restringido
  - Sólo se puede añadir o eliminar un elemento al final de la pila, llamado cabecera de la pila
  - También se llama memoria LIFO (Last-In-First-Out, o último que entra primero que sale)
  - El puntero de pila (SP) es un registro que siempre apunta a la cabecera de la pila
  - Se llaman base de la pila y límite de la pila a las direcciones tope por ambos lados de la pila
- Las operaciones básicas sobre una pila son
  - PUSH: para poner un nuevo elemento en la cabecera de la pila
  - POP: para eliminar un elemento de la cabecera de la pila
- Generalmente la pila siempre crece en orden decreciente de direcciones de memoria

# Ejemplo de funcionamiento

**Example**

```
...
PUSH 123
POP X
POP X
```

| | Memoria | |
|---|---|---|
| | | |
| | ... | |
| | | Límite pila |
| | | |
| | | |
| SP → | 123 | $X = 123$ |
| SP → | 91 | $X = 91$ |
| SP → | 4 | |
| | 74 | |
| | 23 | |
| | 320 | Base pila |
| | ... | |



# Definición

- Forma de especificar la ubicación de los datos y modos para acceder a ellos
- Los datos que maneja una instrucción máquina pueden estar ubicados en:
  - En la propia instrucción: El operando está contenido en un campo de la propia instrucción máquina
  - En un registro de la CPU: Los registros de la CPU se pueden utilizar para almacenar temporalmente los datos
  - En la memoria del computador: En este caso será necesario especificar de algún modo la dirección de memoria dónde se halla el operando: dirección efectiva (EA, Effective Address) del operando



# Modos de direccionamiento simples

- Inmediato
- Directo a registro
- Directo a memoria (absoluto)
- Indirecto con registro
- Indirecto con pila
- Indirecto con memoria
- Indirecto con desplazamiento

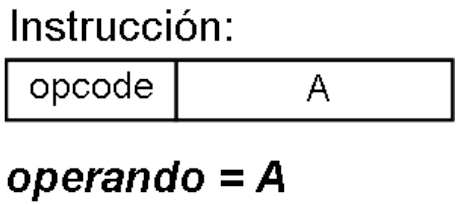
**Notación usada**

Ri: Registro nº i de la CPU
(Ri): Contenido del registro Ri
(X): Contenido de la posición de memoria con dirección X
EA: Dirección efectiva de un operando
SP: Puntero de pila (Stack Pointer)



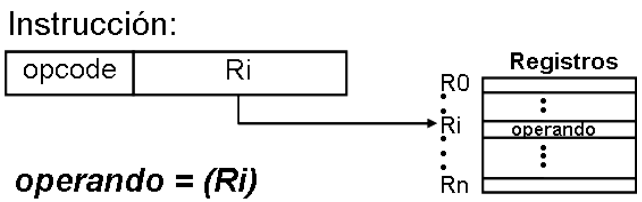
# Inmediato

- El operando está contenido en un campo de la propia instrucción máquina
- Sintaxis: `opcode #A`



## Directo a registro

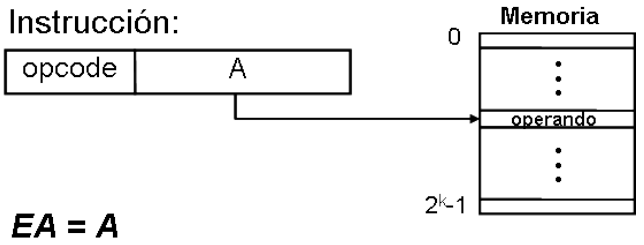
- El operando está contenido en un registro de la CPU
- Sintaxis: `opcode Ri`






## Directo a memoria (absoluto)

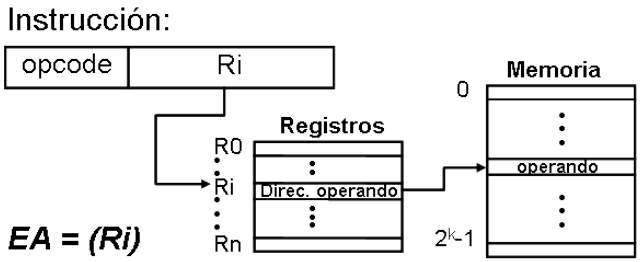
- La dirección efectiva del operando está especificada en la instrucción
- Sintaxis: `opcode A`






## Indirecto con registro

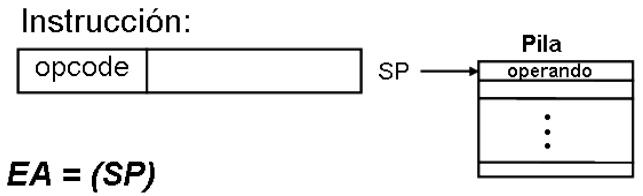
- La dirección efectiva del operando está almacenada en el registro especificado en la instrucción
- Sintaxis: `opcode (Ri)`





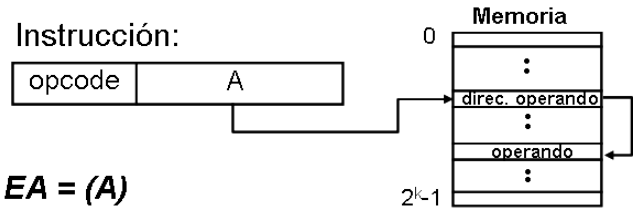

## Indirecto con pila

- El operando está almacenado en la cabecera de la pila del computador (apuntada por SP)
- Sintaxis: `opcode (SP)`



## Indirecto con memoria

- La dirección efectiva del operando está almacenada en la dirección de memoria especificada en la instrucción
- Sintaxis: `opcode (A)`




***EA = (A)***



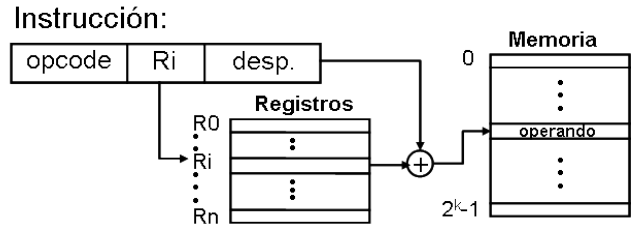
## Indirectos con desplazamiento

- Son un conjunto de direccionamientos en los que la EA del operando se calcula sumando dos cantidades: base (dirección efectiva de memoria) y desplazamiento
  - El lugar donde se ubiquen dichos elementos determina el nombre del direccionamiento



## Indirectos con desplazamiento
### Registro-base

- La dirección efectiva del operando se calcula sumando el campo desplazamiento al contenido del registro especificado
- Sintaxis: `opcode desp(Ri)`

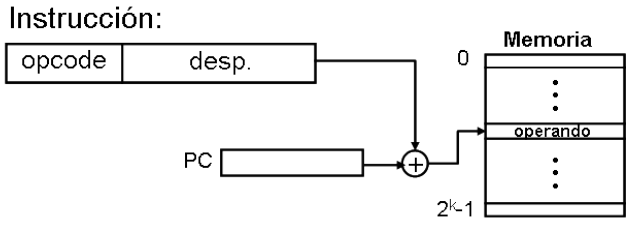



***EA = (Ri)+desp***



## Indirectos con desplazamiento
### Relativo

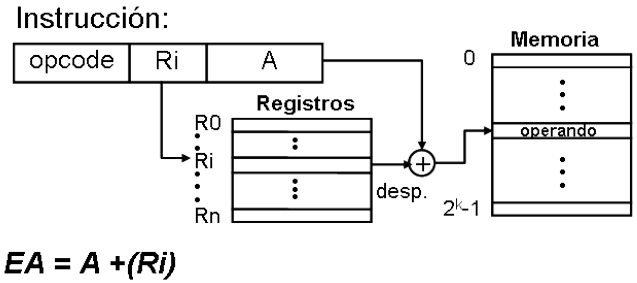
- Es un direccionamiento registro-base que utiliza implícitamente el contador de programa
- Sintaxis: `opcode desp(PC)`




***EA = (PC)+desp***

# Indirectos con desplazamiento
## Indexado

- La dirección efectiva del operando se calcula sumando el campo desplazamiento al contenido del registro especificado
- Sintaxis: `opcode desp(Ri)`

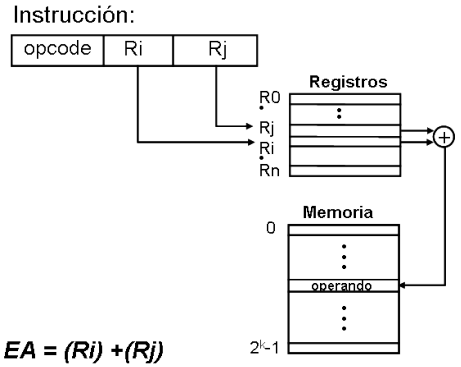



***EA = A +(Ri)***



# Indirectos con desplazamiento
## Registro-base indexado

- La dirección efectiva del operando se calcula sumando el contenido de los dos registros especificados
- Sintaxis: `opcode (Ri,Rj)`




***EA = (Ri) +(Rj)***



# Comparación

| Modo | Algoritmo | Ventaja/Desventaja |
|---|---|---|
| Inmediato | `operando = A` | No referencia a memoria<br>Operando de magnitud limitada |
| Directo mem. | `EA = A` | Sencillo<br>Espacio de direcciones limitado |
| Directo reg. | `operando = Ri` | No referencia a memoria<br>Número limitado de registros |
| Indirecto mem. | `EA = (A)` | Espacio de direcciones grande<br>Referencias a memoria múltiples |
| Indirecto reg. | `EA = (Ri)` | Espacio de direcciones grande<br>Referencia extra a memoria |
| Indirecto desp. | `EA = base + desp.` | Flexibilidad<br>Complejidad |
| Indirecto pila | `EA = (SP)` | 'No referencia a memoria'<br>No siempre disponible |



# Modos de direccionamiento complejos

| Modo | Sintaxis | Dir. efectiva |
|---|---|---|
| Indirecto con registro postincrementado | `Opcode(Ri)+` | `EA=(Ri)`<br>`Ri=Ri+1` |
| Indirecto con registro postdecrementado | `Opcode(Ri)-` | `EA=(Ri)`<br>`Ri=Ri-1` |
| Indirecto con registro preincrementado | `Opcode+(Ri)` | `Ri=Ri+1`<br>`EA=(Ri)` |
| Indirecto con registro predecrementado | `Opcode-(Ri)` | `Ri=Ri-1`<br>`EA=(Ri)` |
| Registro-base indexado con desplazamiento | `Opcode desp(Ri,Rj)` | `EA=(Ri)+(Rj)+desp` |
| Indirecto con registro escalado | `Opcode(Ri×S)` | `EA=(Ri)×S` |
| Registro-base escalado | `Opcode desp(Ri×S)` | `EA=(Ri)×S+desp` |
| Registro-base indexado y escalado con desp. | `Opcode desp(Rj,Ri×S)` | `EA=(Rj)+(Ri)×S+desp` |

# Modos más utilizados

- Inmediato
  - Operaciones con costantes
- Directo con registro
  - Variables locales de procedimientos no recursivos
- Indirecto con registro
  - Variables referenciadas por punteros
- Absoluto
  - Direcciones del sistema
- Relativo
  - Variables globales
- Indexado
  - Acceso a vectores, matrices y cadenas
- Autoincremento/Autodecremento
  - Desapilar/Apilar parámetros de procedimientos
  - Recorrido de vectores y cadenas



# Modos de direccionamiento
## Ejemplo: MC68000

**Modos de direccionamiento**

| Modo de direccionamiento | Cálculo de EA | Sintaxis |
|---|---|---|
| **Direccionamiento directo de registro** | | |
| Directo de registro de datos | EA = Dn | Dn |
| Directo de registro de direcciones | EA = An | An |
| **Direccionamiento indirecto de registro** | | |
| Indirecto de registro | EA = (An) | (An) |
| Indirecto de registro con postincremento (*) | EA = (An); An + N → An | (An)+ |
| Indirecto de registro con predecremento (*) | An + N → An; EA = (An) | -(An) |
| Indirecto de registro con desplazamiento | EA = (An) + $d_{16}$ | $d_{16}$(An) |
| Indirecto de registro indexado con desplazamiento | EA = (An) + (Ri) + $d_8$ | $d_8$(An,Ri.X) |
| **Direccionamiento absoluto** | | |
| Absoluto corto | EA = XXXX | XXXX |
| Absoluto largo | EA = XXXXXX | XXXXXX |
| **Direccionamiento relativo al contador de programa** | | |
| Relativo al PC con desplazamiento | EA = (PC) + $d_{16}$ | $d_{16}$(PC) |
| Relativo al PC indexado con desplazamiento | EA = (PC) + (Ri) + $d_8$ | $d_8$(PC,Ri.X) |
| **Direccionamiento inmediato** | Operando = XXXXXXXX | #XXXXXXXX |

***NOTAS***
EA = Dirección Efectiva
Dn = Registro de datos
An = Registro de direcciones
N = 1 si .B, 2 si .W, 4 si .L
Ri = Registro índice (de datos o direcc.)
$d_8$ = Desplazamiento de 8 bits
$d_{16}$ = Desplazamiento de 16 bits
PC = Contador de Programa



# Modos de direccionamiento
## Ejemplo: ARM

- Cálculo de la dirección efectiva de un operando en ARM
  - Contenido de un registro base `Rn` ± un sesgo
  - Magnitud del sesgo (offset)
    - Un valor inmediato (12 bits menos significativos)
    - Contenido de un tercer registro (`Rm`)



# Modos de direccionamiento
## Ejemplo: ARM

| Denominación | Sintax. ensamb. | Resultado |
|---|---|---|
| Con sesgo inmediato: | | |
| Pre-indexado | [Rn, #sesgo] | EA = [Rn] + sesgo |
| Pre-indexado con post-escritura | [Rn, #sesgo]! | EA = [Rn] + sesgo; Rn ← + sesgo |
| Post-indexado | [Rn], #sesgo | EA = [Rn]; Rn ← + sesgo |
| Con sesgo en Rm: | | |
| Pre-indexado | [Rn, ± Rm, desp] | EA = [Rn] ± Rm desplazado |
| Pre-indexado con post-escritura | [Rn, ± Rm, desp]! | EA = [Rn] ± Rm desplazado Rn ← ± Rm desplazado |
| Post-indexado | [Rn], ± Rm, desp | EA = [Rn]; Rn ← ± Rm desplazado |
| Relativo: | | |
| Pre-indexado con sesgo inmediato | Localización | EA = Localización = [PC] + sesgo |

# Marcos de activación

- En C existen 2 tipos de variables
  - globales: accesibles por cualquier función y asignadas estáticamente a memoria al arrancar el programa
  - locales: accesibles por una única función y asignadas dinámicamente a memoria cada vez que se llama a la función
- Los compiladores de C crean código de tal manera que cada vez que una función es llamada, se reserva en la pila del sistema una zona para las variables locales de la función. Dicha zona se llama marco de activación



# Marcos de activación
## Ejemplo: Distribución de marcos

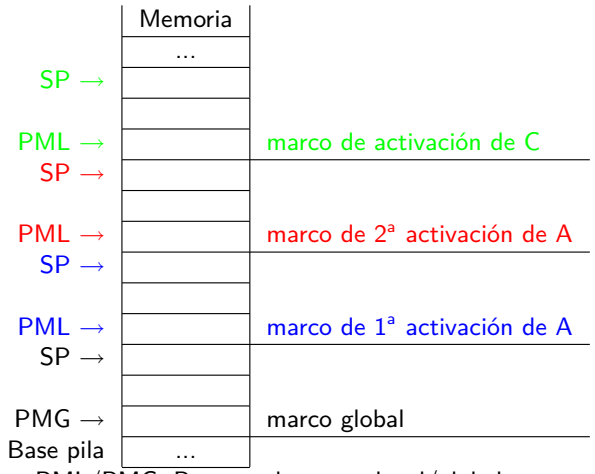
- Supóngase que se tienen tres funciones en C
  - main: Función principal
  - Función A
  - Función B
  - Función C
- Una posible secuencia de ejecución sería
  - **main** → **A** → **A** → **C**
- Veamos como se activan los marcos en memoria



# Marcos de activación
## Ejemplo: Distribución de marcos

**Example**

```
...
main
A
A
C
```




PML/PMG: Puntero de marco local/global

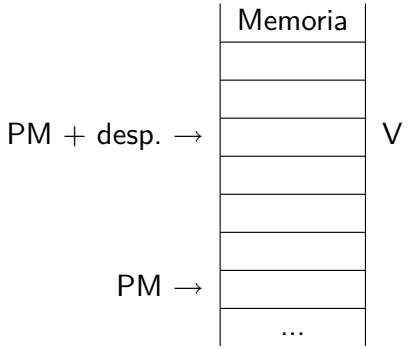


# Marcos de activación
## Ejemplo: Acceso a las variables atómicas

- ¿Qué conocemos?
  - El tamaño y distribución de las variables dentro de los marcos: ya que dependen de las declaraciones que haya hecho el programador
  - La dirección dinámica de los punteros a marcos (que varía de una ejecución a otra)
- ¿cómo calcular las direcciones de las variables?
  - globales: (puntero de marco global) + desplazamiento
  - locales: (puntero de marco local) + desplazamiento

Puede utilizarse el direccionamiento registro-base, en donde el registro base deberá contener la dirección de comienzo del marco (actuando como puntero de marco).

# Marcos de activación
## Ejemplo: Acceso a las variables atómicas





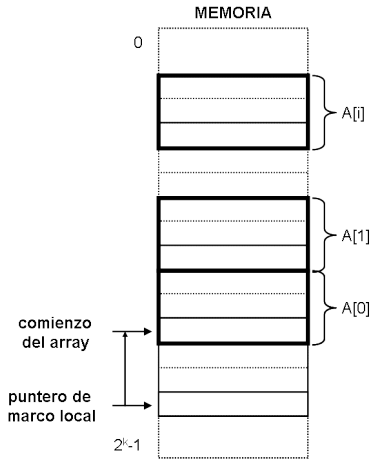

# Marcos de activación
## Ejemplo: Acceso a los elementos de un array

- ¿qué suponemos?
  - El primer índice del array es 0
  - Los elementos del array tiene tamaño n bytes
- ¿qué conocemos?
  - El array comienza en (puntero de marco) + desp.
- ¿cómo calcular la dirección del elemento i?
  - (puntero de marco) + desp. + i × n

Debe utilizarse el direccionamiento registro-base indexado y escalado con desplazamiento, en donde el registro base deberá contener la dirección de comienzo del marco, el registro índice el índice del elemento que se desea acceder y el factor de escala ser igual a n



# Marcos de activación
## Ejemplo: Acceso a los elementos de un array






# Primitivas de pila

- Las operaciones básicas sobre una pila (PUSH y POP) pueden realizarse utilizando los modos de direccionamiento autoindexados:
  - Si la pila crece en orden decreciente de direcciones de memoria
    - la operación PUSH se realiza utilizando direccionamiento indirecto con registro predecrementado
    - la operación POP se realiza utilizando direccionamiento indirecto con registro postincrementado
  - Si la pila crece en orden creciente de direcciones de memoria
    - la operación PUSH se realiza utilizando direccionamiento indirecto con registro preincrementado
    - la operación POP se realiza utilizando direccionamiento indirecto con registro postdecrementado