

•  
•  
•  
•  
•  
•  
•  
•  
•  
•

## Estructura y Tecnología de Computadores

### *Módulo C.* Arquitectura del procesador

---

#### Tema 5. Repertorio de instrucciones y formato de la instrucción máquina

---

**José Manuel Mendías Cuadros**  
*Dpto. Arquitectura de Computadores y Automática*  
*Universidad Complutense de Madrid*

• • • • • • •

2

•

---

## contenidos

---

### 1. Repertorio de instrucciones

Clasificación de las instrucciones. Transferencia de datos. Instrucciones aritméticas. Instrucciones lógicas y de manipulación de bits. Desplazamiento y rotación. Control de flujo. Otras instrucciones

### 2. Aplicaciones del repertorio de instrucciones

Ejecución alternativa. Ejecución iterativa. Subrutinas

### 3. Formato de la instrucción máquina

Lenguaje ensamblador vs. código máquina. Elementos constitutivos de una instrucción máquina. Alternativas de diseño del formato de instrucción. Número de operandos explícitos de una instrucción. Ejemplo de diseño del formato de la instrucción máquina. Ciclo de instrucción

### 4. Arquitecturas CISC y RISC

Arquitecturas CISC. Problemas de las arquitecturas CISC. Características de las arquitecturas RISC. Ventajas y desventajas de las arquitecturas RISC. Computadores CISC y computadores RISC.

### 5. Ejemplos

MC68000. Arquitectura MIPS

estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Clasificación de las instrucciones

- ☒ **Transferencia de datos**, permiten el movimiento entre distintos dispositivos de almacenamiento del computador
- ☒ **Aritméticas**, permiten realizar operaciones de tipo aritmético
- ☒ **Lógicas y de manipulación de bits**, permiten realizar operaciones lógicas, bit a bit, entre los operandos o manipular un bit del operando
- ☒ **Desplazamiento y rotación**, permiten desplazar o rotar un operando a la decha. o la izda. un nº determinado de bits
- ☒ **Control de flujo**, permiten romper la secuencia normal de ejecución
- ☒ **Otras instrucciones**
  - Transformación de datos
  - Manipulación de direcciones
  - Creación de marcos de almacenamiento local
  - Control del sistema
  - E/S

estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Trasferencia de datos

- ☒ Permiten el movimiento entre distintos dispositivos de almacenamiento del computador:
  - Reg → Reg
  - Reg → Mem
  - Reg → Pila
  - Mem → Reg
  - Mem → Mem
  - Mem → Pila
  - Pila → Reg
  - Pila → Mem
- ☒ Necesario especificar:
  - Tipo de movimiento
  - Dirección de operandos fuente y destino
  - Tamaño de datos a mover (byte, palabra, doble palabra, ...)
  - Nº de elementos a mover (para movimientos múltiples)

Instrucción	Operación	Descripción
MOVE    fuente,destino	destino ← fuente	Transfiere palabra de reg. a reg., reg. a mem, mem. a reg o mem. a mem. (fuente= mem. o reg.; destino = mem. o reg.)
LOAD    Ri, dir	Ri ← dir	Transfiere palabra de memoria a registro
STORE   dir, Ri	Ri ← dir	Transfiere palabra de registro a memoria
PUSH    fuente	Pila ← fuente	Transfiere palabra de mem. o reg. a la cabecera de pila
POP    destino	destino ← Pila	Transfiere palabra de cabecera de pila a mem. o reg.
MOVEM   fnte., dest., n	fnte <sub>0</sub> ← dest <sub>0</sub> ..... ..... fnte <sub>n-1</sub> ← dest <sub>n-1</sub>	Transfiere n palabras a partir de una dir. inicial fuente y una dir. inicial destino

estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Instrucciones aritméticas

- ☒ Permiten realizar operaciones de tipo aritmético
- ☒ Necesario especificar:
  - Tipo de operación (suma, resta, multiplicación, división, etc.)
  - Tipo de operandos y de aritmética (con signo, sin signo, entera, punto flotante, BCD, ...)
  - tamaño de datos sobre los que se opera
  - Dirección de operandos fuente y destino (0, 1, 2 ó 3, según el repertorio)

<i><b>Instrucción</b></i>	<i><b>Operación</b></i>	<i><b>Descripción</b></i>
ADD    fnte1,fnte2,dest	$\text{dest} \leftarrow \text{fnt1} + \text{fnt2}$	Suma dos operandos
SUB    fnte1,fnte2,dest	$\text{dest} \leftarrow \text{fnt1} - \text{fnt2}$	Resta dos operandos
MULT   fnte1,fnte2,dest	$\text{dest} \leftarrow \text{fnt1} * \text{fnt2}$	Multiplica dos operandos
DIV    fnte1,fnte2,dest	$\text{dest} \leftarrow \text{fnt1} / \text{fnt2}$	Divide dos operandos
NEG    fnte,dest	$\text{dest} \leftarrow - \text{fnt}$	Cambia de signo al operando
ABS    fnte,dest	$\text{dest} \leftarrow \text{Abs}(\text{fnt})$	Obtiene el valor absoluto del operando
INC    fnte,dest	$\text{dest} \leftarrow \text{fnt} + 1$	Suma 1 al operando
DEC    fnte,dest	$\text{dest} \leftarrow \text{fnt} - 1$	Resta 1 al operando
COMP   fnte1,fnte2	$\text{fnt1} - \text{fnt2}$ Activa Estado	Compara dos operandos y activa los bits de estado según el resultado de la comparación

estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Instrucciones lógicas y de manipulación de bits

- ☒ Permiten realizar operaciones lógicas, bit a bit, entre los operandos o manipular un bit del operando
- ☒ Necesario especificar:
  - Tipo de operación (AND, OR, NOT, Bit Clear, Bit Set, etc)
  - Tamaño de datos sobre los que se opera (byte, palabra, doble palabra, ...)
  - Dirección de operandos fuente y destino (0, 1, 2 ó 3, según el repertorio)
  - El número de bit en las instrucciones de manipulación de bit

<i><b>Instrucción</b></i>	<i><b>Operación</b></i>	<i><b>Descripción</b></i>
AND    fnte1,fnte2,dest	$\text{dest} \leftarrow \text{fnt1} \text{ AND } \text{fnt2}$	Realiza la Y lógica de dos operandos
OR    fnte1,fnte2,dest	$\text{dest} \leftarrow \text{fnt1} \text{ OR } \text{fnt2}$	Realiza la O lógica de dos operandos
NOT    fnte,dest	$\text{dest} \leftarrow \text{NOT } \text{fnt}$	Realiza la negación lógica del operando
XOR    fnte1,fnte2,dest	$\text{dest} \leftarrow \text{fnt1} \text{ XOR } \text{fnt2}$	Realiza la O exclusiva de dos operandos
BCLR   dest, n	$\text{dest}(n) \leftarrow 0$	Pone a cero el bit especificado
BSET   dest, n	$\text{dest}(n) \leftarrow 1$	Pone a uno el bit especificado
BTST   fnte, n	$\text{Estado} \leftarrow \text{fnte}(n)$	Activa estado según valor del bit indicado

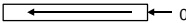
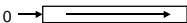
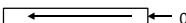
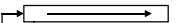
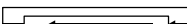
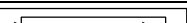
estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Desplazamiento y rotación

- ☒ Permiten desplazar o rotar un operando a la decha. o la izda. un nº determinado de bits
- ☒ Necesario especificar:
  - Tipo de operación (desplazamiento izda. o decha., rotación izda. o decha., )
  - Tamaño de datos sobre los que se opera (byte, palabra, doble palabra, ...)
  - Dirección del operando
  - Nº de bits a desplazar o rotar

<i><b>Instrucción</b></i>	<i><b>Operación</b></i>	<i><b>Descripción</b></i>
LSL    fnte,n		Desplaz. lógico izda. n bits (Logic Shift Left)
LSR    fnte,n		Desplaz. lógico decha. n bits (Logic Shift Right)
ASL    fnte,n		Desplaz. aritm. izda. n bits (Arith. Shift Left)
ASR    fnte,n		Desplaz. aritm. decha. n bits (Arith. Shift Right)
RL    dest, n		Rotación izda. n bits (Rotate Left)
RR    dest, n		Rotación decha. n bits (Rotate Right)

estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Control de flujo

- ☒ Permiten romper la secuencia normal de ejecución y saltar a una determinada dirección especificada en la instrucción o implícita
- ☒ Necesario especificar:
  - Dirección de siguiente instrucción a ejecutar si el salto es explícito

<i><b>Instrucción</b></i>	<i><b>Operación</b></i>	<i><b>Descripción</b></i>
JMP    dir	$PC \leftarrow dir$	Salta (Jump) a la dirección especificada
Bcc    dir	If cc=TRUE then $PC \leftarrow dir$ Else $PC \leftarrow PC + long\_instr$	Bifurca (Branch) a la dirección especificada si la condición es cierta
JSR    dir	Estado, $PC \rightarrow Pila$ ; $PC \leftarrow dir$	Salto a subrutina; guarda en pila PC y estado
RTS	Estado, $PC \leftarrow Pila$	Retorno de subrutina; recupera de pila PC y estado
SKIP    n	$PC \leftarrow PC + n * long\_instr$	Salta el nº de instrucciones especificado
NOP	$PC \leftarrow PC + long\_instr$	No ejecuta ninguna operación, pero la ejecución continúa secuencialmente

estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Control de flujo (cont.)

- ☒ Las instrucciones de bifurcación o salto condicional (Bcc), saltan o no en función de la condición especificada
- ☒ Esta condición se calcula a partir de los **bits de condición** del registro de estado
  - Los bits de condición se activan según el resultado de las instrucciones ejecutadas anteriormente

Condic.	Nombre	Es cierta si
EQ	Igual (Equal)	En una comparación previa los operandos son iguales
NEQ	No igual (No equal)	En una comparación previa los operandos son distintos
GT	Mayor que (Greater Than)	En una comparación previa el primer operando es mayor que el segundo
GE	Mayor o igual (Gr. or Eq.)	En una comparación previa el primer operando es mayor o igual que el segundo
LT	Menor que (Less Than)	En una comparación previa el primer operando es menor que el segundo
LE	Menor o igual (Les.or Eq.)	En una comparación previa el primer operando es menor o igual que el segundo
Z	Cero (Zero)	El resultado de una operación anterior es cero
NZ	No cero (No Zero)	El resultado de una operación anterior es distinto de cero
P	Positivo	El resultado de una operación anterior es positivo
N	Negativo	El resultado de una operación anterior es negativo
C	Acarreo (Carry)	El resultado de una operación anterior ha producido acarreo
NC	No acarreo (No Carry)	El resultado de una operación anterior no ha producido acarreo
V	Desbord. (Overflow)	El resultado de una operación anterior ha producido desbordamiento
NV	No desbord. (No Overfl.)	El resultado de una operación anterior no ha producido desbordamiento
T	Verdad (True)	Siempre cierta
F	Falso (False)	Siempre falsa

estructura y tecnología de computadores

• • • • • • •

## 1. repertorio de instrucciones

### Otras instrucciones

- ☒ En este grupo están englobadas distintas instrucciones
  - **Transformación de datos**
    - Cambian el formato de los datos, por ej. de decimal a binario
  - **Manipulación de direcciones**
    - Permiten calcular la dirección efectiva de un operando y almacenarla en un registro o en pila, p. ej.
      - ✓ LEA fnte, reg (Load effective address: reg ← Dirección de fnte)
      - ✓ PEA fnte (Push Effective Address: Pila ← Dirección de fnte)
  - **Creación de marcos de almacenamiento local**
    - ✓ Permiten reservar espacio en la pila del sistema, p. ej.:
      - ✓ LINK reg, #tam (Pila ← reg, reg ← SP, SP ← SP+tam)
      - ✓ UNLK reg (SP ← reg, reg ← Pila)
  - **Control del sistema**
    - Suelen ser instrucciones privilegiadas que usa el sistema operativo, p. ej.:
      - ✓ RESET, para reiniciar el computador (PC ← valor inicial)
      - ✓ RTE: retorno de excepción o interrupción
  - **E/S**
    - Para entrada y salida de datos entre el computador y dispositivos periféricos, p. ej.:
      - ✓ IN dir\_perif, reg (reg ← perif)
      - ✓ OUT dir\_perif, reg (perif ← reg)

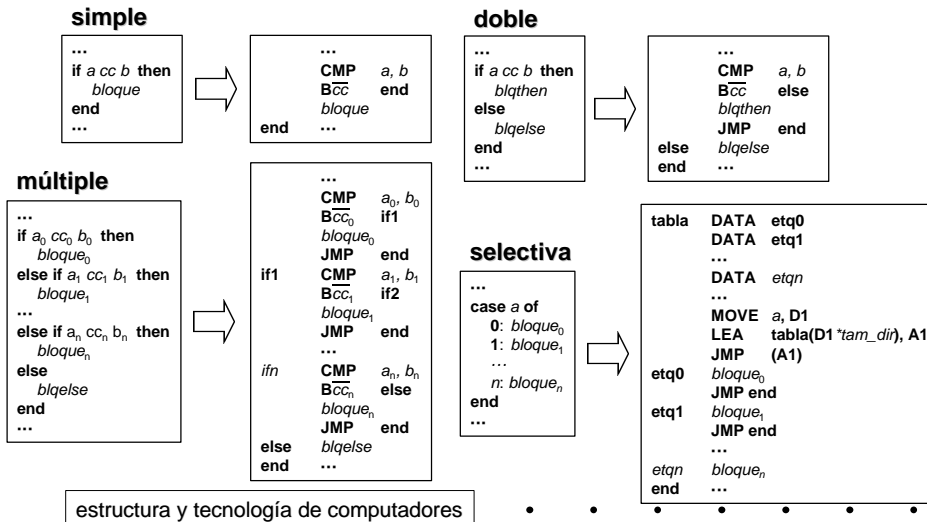
estructura y tecnología de computadores

• • • • • • •

## 2. aplicaciones del repertorio de instrucciones

### Ejecución alternativa

☒ Según el valor de cierta condición o valor, se decide entre varios caminos de ejecución

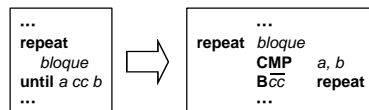


## 2. aplicaciones del repertorio de instrucciones

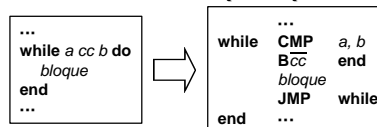
### Ejecución iterativa

☒ Según el valor de cierta condición, se decide repetir la ejecución de cierto bloque de código

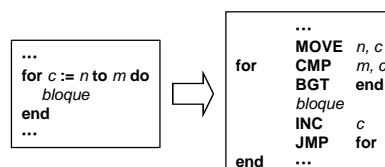
#### con evaluación al final



#### con evaluación al principio



#### con un número definido de iteraciones



## 2. aplicaciones del repertorio de instrucciones

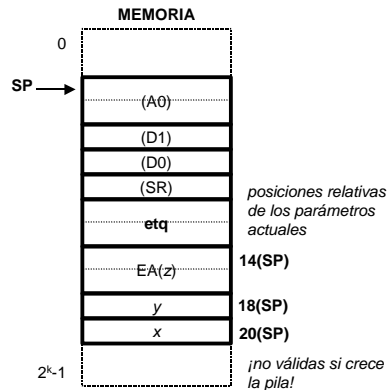
### Subrutinas

```
...
procedure sub( a, b :integer; var c : integer )
begin
  bloque
end
...
sub( x, y, z );
...
```

Suponiendo:

- ✓ una memoria **direccionada por bytes**
- ✓ un tamaño palabra de 16 bits
- ✓ un tamaño de dirección de 32 bits
- ✓ que un **integer** se representa mediante palabra

```
sub  ...
      PUSH  SR      preservación del contexto
      PUSH  D0
      PUSH  D1
      PUSH  A0
      MOVE  14(SP), D0  acceso a parámetros actuales
      MOVE  18(SP), D1
      MOVE  20(SP), A0
      bloque
      POP   A0      restauración del contexto
      POP   D1
      POP   D0
      POP   SR
      RTS          retorno al programa llamante
      ...
      PUSH  x      paso de parámetros actuales
      PUSH  y
      PEA   z
      JSR   sub     llamada a la subrutina
      etq   ADD   #8, SP  liberación del área de parámetros
      ...
```

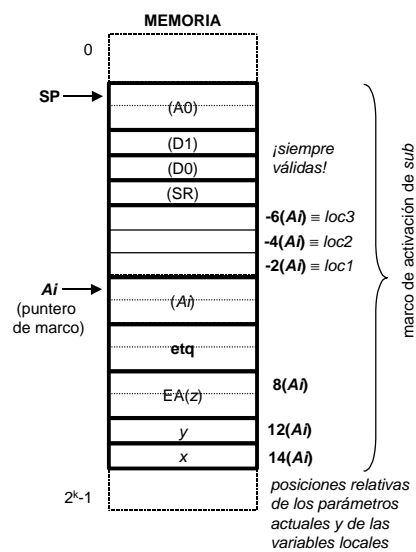


estructura y tecnología de computadores

## 2. aplicaciones del repertorio de instrucciones

```
...
procedure sub( a, b :integer; var c : integer )
var loc1, loc2, loc3 :integer
begin
  bloque
end
...
sub( x, y, z );
...
```

```
sub  ...
      LINK  Ai, #-6  creación del área local de variables
      PUSH  SR      preservación del contexto
      PUSH  D0
      PUSH  D1
      PUSH  A0
      bloque
      POP   A0      restauración del contexto
      POP   D1
      POP   D0
      POP   SR
      UNLK  Ai      destrucción del área local de variables
      RTS          retorno al programa llamante
      ...
      PUSH  x      paso de parámetros actuales
      PUSH  y
      PEA   z
      JSR   sub
      etq   ADD   #8, SP  liberación del área de parámetros
      ...
```

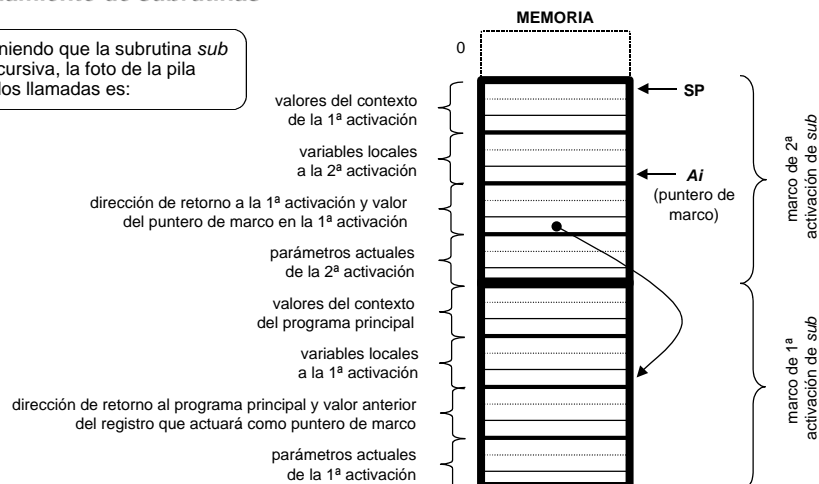


estructura y tecnología de computadores

## 2. aplicaciones del repertorio de instrucciones

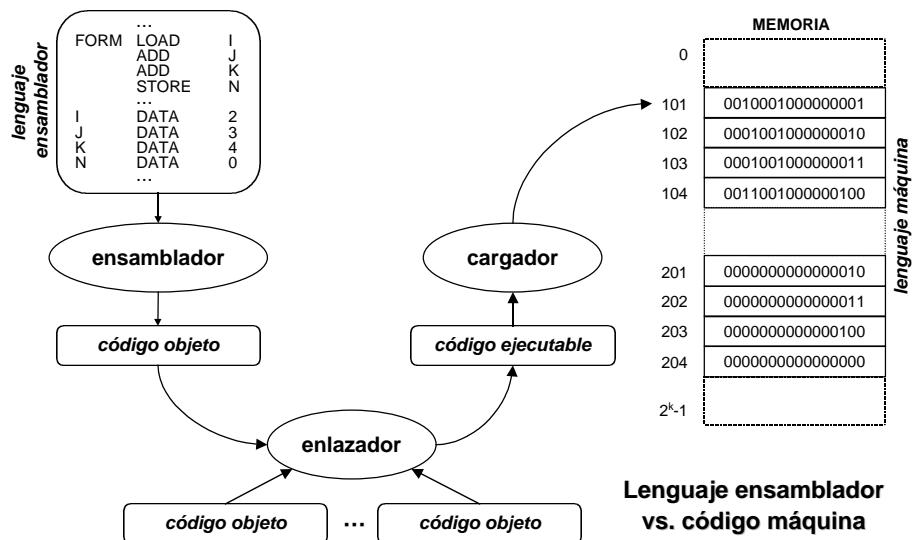
### Anidamiento de subrutinas

Suponiendo que la subrutina *sub* es recursiva, la foto de la pila tras dos llamadas es:



estructura y tecnología de computadores

## 3. formato de la instrucción máquina



estructura y tecnología de computadores



### 3. formato de la instrucción máquina

#### ☒ **Ensamblador (*assembler*):**

- Resuelve las etiquetas de instrucciones y datos
  - ⇒ Relativas a PC para instrucciones
  - ⇒ Relativas a algún registro para datos
- Expande macros y pseudo-instrucciones
- Interpreta las directivas de ensamblaje
- Fija la representación de los datos
- Traduce las instrucciones a código máquina
- Crea el fichero objeto
  - ⇒ Determina cabeceras, segmentos de código y segmentos de datos
  - ⇒ Crea tablas de símbolos: símbolos no resueltos + símbolos visibles
  - ⇒ Añade información para la depuración

#### ☒ **Enlazador (*linker*):**

- Combina varios códigos objeto, resolviendo las referencias cruzadas
- Crea un código ejecutable

#### ☒ **Cargador (*loader*):**

- Lee y carga sobre memoria el código ejecutable
- Inicializa registros, pila y argumentos
- Fija los vectores de excepción
- Salta a la rutina de inicio del programa

estructura y tecnología de computadores

• • • • • • •

### 3. formato de la instrucción máquina

#### Elementos constitutivos de una instrucción máquina

##### ☒ **Código de operación (*opcode*)**

- Especifica el tipo de operación a realizar: *Suma, Resta, Y lógica, Movimiento de datos, etc.*

##### ☒ **Operandos fuente**

- La instrucción puede involucrar uno o más operandos fuente (o ninguno)
- Estos podrán especificarse utilizando distintos modos de direccionamiento

##### ☒ **Operando destino**

- Si la instrucción produce un resultado deberá especificar el operando destino
- Este podrá especificarse utilizando distintos modos de direccionamiento

##### ☒ **Instrucción siguiente**

- En ocasiones es necesario especificar cuál es la siguiente instrucción a ejecutar
- Normalmente la ejecución del programa es secuencial
  - ⇒ La dirección de la siguiente instrucción suele ser implícita:
    - ⇒  $Dir. \text{ instrucción siguiente} = Dir. \text{ instrucción actual} + Longitud \text{ instrucción actual}$
- En las instrucciones de salto sí es necesario especificar la dir. de la siguiente instrucción
  - ⇒ Ruptura de la secuencia normal del programa

**Instrucción máquina:**

opcode	op. fuente 1	• • • •	op. fuente n	op. destino	instruc. siguiente
--------	--------------	---------	--------------	-------------	--------------------

estructura y tecnología de computadores

• • • • • • •

### 3. formato de la instrucción máquina

#### Alternativas de diseño del formato de instrucción

- ☒ A la hora de diseñar el formato de las instruc. máquina de un repertorio, debemos decidir:
  - Cuál será la longitud de las instrucciones
    - ⇒ Todas las instrucciones de igual longitud
    - ⇒ Instrucciones de distinta longitud, según el tipo de operación a realizar
  - De cuántos campos constará la misma (dependiendo del tipo de instrucción)
  - Cuántos bits ocuparán cada uno de esos campos
  - Codificación de cada campo
- ☒ Los principales factores a tener en cuenta para decidir este formato son:
  - **Número de operaciones** distintas presentes en el repertorio
    - ⇒ Aritméticas, lógicas, de control, de movimiento de datos, ...
  - **Número de operandos** de la instrucción (incluyendo operandos fuente y destino)
    - ⇒ Instrucciones sin operandos
    - ⇒ Instrucciones con 1, 2, 3, ... operandos
  - **Modos de direccionamiento** disponibles para cada operando
    - ⇒ Inmediato, absoluto, de registro, indirecto, .....
  - **Tamaño y tipos de datos** soportados
    - ⇒ Bit, byte, palabra, doble palabra, ....
    - ⇒ Caracteres, BCD, magnitud y signo, complemento a 2, punto flotante, ....

estructura y tecnología de computadores

### 3. formato de la instrucción máquina

#### Número de operandos explícitos en una instrucción

- ☒ Una operación típica de un computador (p. ej.  $A = B + C$ ) suele tener tres operandos
  - Dos operandos fuente (B y C) y un operando destino (A)
- ☒ Las principales alternativas que se presentan son:
  - Instrucciones con tres operandos explícitos
  - Instrucciones con dos operandos explícitos
  - Instrucciones con un único operando explícito
  - Instrucciones sin operandos explícitos

#### Instrucciones con tres operandos explícitos

- ☒ Se especifican los tres operandos en la instrucción máquina
  - Cada operando puede soportar distintos modos de direccionamiento
- ☒ Formato flexible, pero ocupa mucho espacio

#### Ejemplo

**ADD A, B, C**       $A \leftarrow B + C$

#### Instrucciones con dos operandos explícitos

- ☒ Dos operandos explícitos y uno implícito
  - Uno de los operandos actúa como fuente y destino
  - Cada operando puede soportar distintos modos de direccionamiento
- ☒ Es menos flexible, pero ocupa menos espacio

#### Ejemplo

**ADD B, C**       $B \leftarrow B + C$

estructura y tecnología de computadores

### 3. formato de la instrucción máquina

#### Número de operandos explícitos en una instrucción (cont.)

##### Instrucciones con un único operando explícito

- ☒ Se especifican un sólo operando en la instrucción máquina
  - Uno de los operandos y el resultado se almacenan en un registro especial del procesador: el **acumulador**
  - El operando puede soportar distintos modos de direccionamiento
- ☒ Formato muy reducido, pero poco flexible

##### Ejemplo

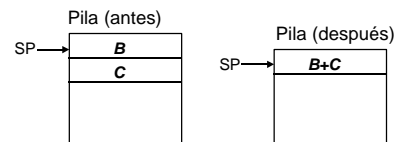
**ADD B**      *Acum.  $\leftarrow$  B + Acum.*

##### Instrucciones sin operandos explícitos

- ☒ Tanto los operandos como el resultado se almacenan en la pila del sistema
- ☒ Ocupa muy poco espacio
- ☒ Requiere varias instrucciones previas de acceso a la pila

##### Ejemplo

**ADD**      *(SP)  $\leftarrow$  (SP) + (SP-1)*



estructura y tecnología de computadores

### 3. formato de la instrucción máquina

#### Número de operandos explícitos en una instrucción: EJEMPLO

- ☒ Escribir los programas simbólicos para la siguiente operación:  

$$Y = (A-B)/(C+D \cdot E)$$
 en un computador de 0, 1, 2 ó 3 direcciones, utilizando las instrucciones que sean necesarias y sin sobrescribir el contenido de ninguno de los operandos fuente

##### 3 operandos

SUB	Y, A, B	(Y $\leftarrow$ A-B)
MULT	T, D, E	(T $\leftarrow$ D * E)
ADD	T, T, C	(T $\leftarrow$ T + C)
DIV	Y, Y, T	(Y $\leftarrow$ Y / T)

##### 2 operandos

MOVE	Y, A	(Y $\leftarrow$ A)
SUB	Y, B	(Y $\leftarrow$ Y - B)
MOVE	T, D	(T $\leftarrow$ D)
MULT	T, E	(T $\leftarrow$ T * E)
ADD	T, C	(T $\leftarrow$ T + C)
DIV	Y, T	(Y $\leftarrow$ Y / T)

##### 1 operando

LOAD	D	(Ac $\leftarrow$ D)
MULT	E	(Ac $\leftarrow$ Ac * E)
ADD	C	(Ac $\leftarrow$ Ac + C)
STORE	Y	(T $\leftarrow$ Ac)
LOAD	A	(Ac $\leftarrow$ A)
SUB	B	(Ac $\leftarrow$ Ac - B)
DIV	Y	(Ac $\leftarrow$ Ac / Y)
STORE	Y	(Y $\leftarrow$ Ac)

##### 0 operandos

PUSH	E
PUSH	D
MULT	
PUSH	C
ADD	
PUSH	B
PUSH	A
SUB	
DIV	
POP	Y

estructura y tecnología de computadores

### 3. formato de la instrucción máquina

#### Ejemplo de diseño del formato de la instrucción máquina

- ☒ Diseñar el formato de las instrucciones de un computador con 8 registros de propósito general que permita codificar en una instrucción de 32 bits lo siguiente:
- 12 instrucciones de 3 operandos
    - Dos operandos que permiten direccionamiento absoluto e indirecto de memoria
    - Un operando que permite direccionamiento de registro en indirecto de registro
  - 150 instrucciones de 2 operandos
    - Un operando que permite direccionamiento absoluto e indirecto de memoria
    - Un operando que permite direccionamiento de registro en indirecto de registro
  - 30 instrucciones de 0 operandos
- ☒ **Nota:** El campo dirección de los operandos con direccionamiento absoluto e indirecto de memoria será de 11 bits.

#### Solución

- ☒ Los operandos que permiten direccionamiento absoluto e indirecto de memoria requieren:
- 1 bit para especificar el modo de direccionamiento ( $m=0 \rightarrow$  absoluto;  $m=1 \rightarrow$  indirecto)
  - 11 bits para especificar la dirección
- ☒ Los operandos que permiten direccionamiento de registro en indirecto de registro requieren:
- 1 bit para especificar el modo de direccionamiento ( $m=0 \rightarrow$  de registro;  $m=1 \rightarrow$  indirecto de reg.)
  - 3 bits para especificar el número de registro (ya que tenemos 8 registros)

estructura y tecnología de computadores

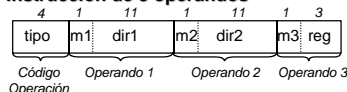
• • • • • • •

### 3. formato de la instrucción máquina

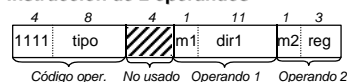
#### Solución (Cont.)

- ☒ Para las 12 instrucciones de tres operandos necesitamos:
- 4 bits para especificar el tipo de operación: del 0 (0000) al 11 (1011)
  - 12 bits para especificar el primer y segundo operando respectivamente
  - 4 bits para especificar el tercer operando
- ☒ Para las 150 instrucciones de dos operandos necesitamos:
- 8 bits para especificar el tipo de operación: del 0 (0000000) al 149 (1010101)
  - 12 bits para especificar el primer operando
  - 4 bits para especificar el tercer operando
- ☒ Para las 30 instrucciones sin operandos necesitamos:
- 5 bits para especificar el tipo de operación: del 0 (00000) al 29 (11101)

#### Instrucción de 3 operandos



#### Instrucción de 2 operandos



#### Instrucción sin operandos



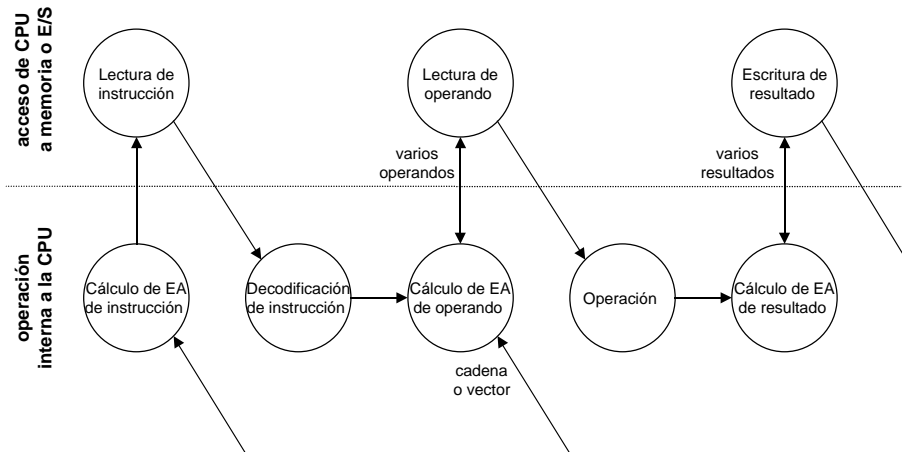
#### Nomenclatura

**mk:** modo de direccionamiento del operando k  
**dirk:** dirección del operando k  
**reg:** número de registro  
**tipo:** tipo de instrucción a ejecutar

estructura y tecnología de computadores

• • • • • • •

### 3. formato de la instrucción máquina



Ciclo de instrucción

estructura y tecnología de computadores

### 4. arquitecturas CISC y RISC

#### Arquitecturas CISC (Computador de conjunto de instrucciones complejo)

- ☒ Desde los años 60 (en que aparece la unidad de control microprogramada) hasta principio de los años 80 la tendencia ha sido **incrementar la complejidad de la CPU**
  - Gran número de instrucciones en el repertorio e **instrucciones complejas**
  - Uso de gran número de modos de **direccionamiento complejos**
- ☒ Esta tendencia se debe a varias razones:
  - Aparecen lenguajes de programación de alto nivel cada vez más sofisticados
    - ⇒ Diseño de instrucciones complejas que permitan implementar instrucciones de alto nivel directamente o con un pequeño número de instrucciones máquina
    - ⇒ Una instrucción de alto nivel compleja implementable directamente en hardware, mejora la eficiencia del programa ya que se ejecuta mucho más rápidamente
  - Gracias al uso de unidad de control microprogramada, se pueden implementar estas instrucciones máquinas complejas de forma simple, sin demasiado coste para el diseñador
  - Facilidad para mantener la compatibilidad hacia abajo con los miembros de la misma familia
    - ⇒ Añadir nuevas instrucciones al repertorio, manteniendo las antiguas

estructura y tecnología de computadores

## 4. arquitecturas CISC y RISC

### Problemas de las arquitecturas CISC

- ☒ Los estudios de ejecución de programas en arquitecturas CISC revelan que
  - La mayor parte de las instrucciones complejas apenas si se utilizan en programas reales
  - Lo mismo ocurre con modos de direccionamiento complejos
- ☒ Dificultad para diseñar compiladores eficientes
  - Al aumentar la complejidad del repertorio se hace cada vez más difícil diseñar compiladores que aprovechen realmente esta gran variedad y versatilidad de instrucciones máquina
- ☒ Las instrucciones ocupan mucho espacio en memoria
  - Al existir un gran número de instrucciones y de modos de direccionamiento se necesita mucho espacio para codificar las instrucciones máquina y se tarda más tiempo en leerlas de memoria
  - El tiempo de acceso a memoria para captar instrucciones (*Fetch*) resulta cada vez más crítico conforme evolucionan los computadores
- ☒ Debido a estos problemas a principio de los 80 surge la arquitectura **RISC (Computador de conjunto de instrucciones reducido)**
  - **Idea básica:** hacer rápidos y eficientes los casos más comunes a costa de reducir la velocidad en los casos menos comunes

estructura y tecnología de computadores

• • • • • • •

## 4. arquitecturas CISC y RISC

### Características de las arquitecturas RISC

- ☒ **Repertorio de instrucciones simple y ortogonal**
  - Sólo se dispone de instrucciones máquina básicas
  - Repertorio ortogonal: no existen varias instrucciones distintas para realizar la misma operación
- ☒ **Modos de direccionamiento sencillos**
  - Cada operando sólo soporta unos pocos modos de direccionamiento simples
- ☒ **Formatos de instrucción uniformes**
  - Número reducido de formatos de instrucciones distintos
  - Todas las instrucciones de longitud similar
- ☒ **Tipos de datos simples**
  - Las instrucciones sólo soportan unos pocos tipos de datos básicos distintos
- ☒ **Arquitectura de tipo registro-registro**
  - Sólo las instrucciones LOAD (Mem → Reg) y STORE (Reg → Mem) hacen referencia a memoria
  - El resto de instrucciones (aritméticas, lógicas, etc.) son de tipo registro-registro,
    - Tanto los operandos fuente como destino son registros de la arquitectura
  - Las arquitecturas RISC suelen caracterizarse por disponer de un gran número de registros de propósito general

estructura y tecnología de computadores

• • • • • • •

## 4. arquitecturas CISC y RISC

### Ventajas de las arquitecturas RISC

- ☒ Su estructura es mucho más simple
  - Es más rápido y puede trabajar a mayor velocidad
  - Su diseño es más sencillo y barato
- ☒ Instrucciones más cortas
  - Tardan menos tiempo en leerse de memoria
- ☒ Mayor simplicidad para generar código máquina
  - Los compiladores son más eficientes y más sencillos de diseñar

### Desventajas de las arquitecturas RISC

- ☒ El compilador genera un mayor número de instrucciones máquina
  - Se necesitan varias instrucciones para ejecutar las instrucciones de alto nivel
  - Mayor número de instrucciones ↔ Instrucciones más rápidas
- ☒ Más difícil mantener la compatibilidad entre computadores de la misma familia
  - Para diseñar un repertorio simple no se pueden "heredar" todas las instrucciones

estructura y tecnología de computadores

• • • • • • •

## 4. arquitecturas CISC y RISC

### Computadores CISC

- ☒ **VAX 11** (desaparecido)
- ☒ **IBM Mainframes** (serie 360, 370 y descendientes)
- ☒ **Intel 80x86** (i8086, i80286, i80386, i80486, Pentium, Pentium II, Pentium III)
- ☒ **Motorola MC68xxx** (MC68000, MC68010, MC68020, MC68030, MC68040, MC68060 )

### Computadores RISC

- ☒ **Motorola MC88000** (desaparecido)
- ☒ **MIPS Rxxxx** (R2000, R3000, R4000, R5000, R8000, R10000, R12000)
- ☒ **Sun SPARC** (SPARC, Super SPARC 2, Ultra SPARC I, Ultra SPARC II, Ultra SPARC III)
- ☒ **HP PA-RISC** (7100, 7200, 7300, 8000, 8200, 8500)
- ☒ **PowerPC** (601, 602, 603, 604, 620, 630, Power3)
- ☒ **DEC Alpha** (21064, 21064a, 21066a, 21164, 21164a, 21264)
- ☒ **Intel 80860, Intel 80960**

estructura y tecnología de computadores

• • • • • • •

## 5. ejemplos: MC68000

### Características del MC68000

- ☒ Procesador CISC de 16 bits
  - Aprox. 90 instrucciones máquina
  - 12 modos de direccionamiento
  - 9 formatos de instrucción distintos y con tamaños de una a cinco palabras
  - Ancho del bus de datos: 16 bits
  - Tamaño mínimo direccionable: 1 byte
  - Ancho del bus de direcciones: 24 bits ( $2^{24}$  bytes = 16 Mbytes de memoria direccionables)
- ☒ Modos de funcionamiento
  - **Modo usuario**
    - Modo normal de ejecución de programas
    - No tiene acceso a determinadas instrucciones privilegiadas
    - No tiene acceso a determinados registros del supervisor
  - **Modo supervisor**
    - Modo de ejecución del Sistema Operativo y durante el tratamiento de interrupciones
    - Tiene acceso a todas las instrucciones privilegiadas y a registros del supervisor

estructura y tecnología de computadores

• • • • • • •

## 5. ejemplos: MC68000

### Tipos de datos y organización de la memoria

- ☒ **Bit**
- ☒ **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 (32 bits) → se representa como *.l (long word)*
- ☒ **Decimal BCD empaquetado** (dos dígitos BCD por byte)

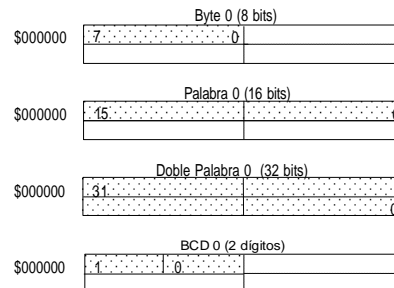
#### Memoria (Ordenación Big-Endian)

Palabra

\$000000  
\$000002

Byte \$000000	Byte \$000001
Byte \$000002	Byte \$000003
	.
	.
	.
	.
Byte \$FFFFFF	Byte \$FFFFFF

\$FFFFFF



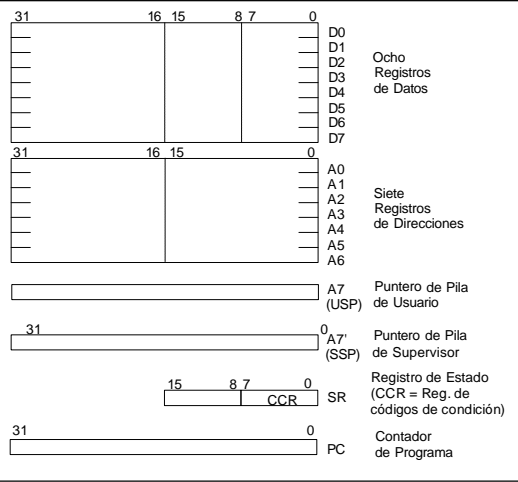
estructura y tecnología de computadores

• • • • • • •



5. ejemplos: MC68000

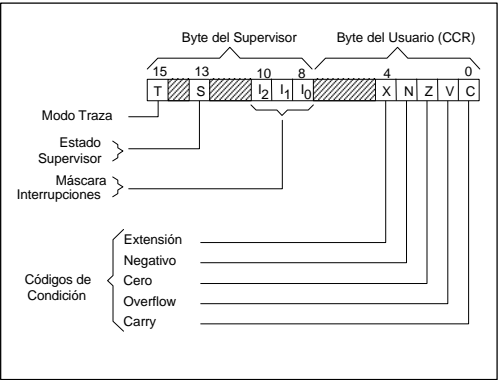
Registros de la arquitectura



- ☒ **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

5. ejemplos: MC68000

Registro de estado (SR)



Códigos de condición

- C ("Carry" o acarreo)**  
Se activa cuando se produce un arrastre de suma o de resta
- V ("oVerflow" o desbordam.)**  
Se activa cuando el resultado de una operación no se puede expresar en C'2 dentro del tamaño seleccionado
- Z ("Zero" o cero)**  
Se activa cuando el resultado de una operación es cero
- N (Negativo)**  
Se activa cuando el resultado de una operación es negativo
- X (eXtensión)**  
Se activa cuando hay arrastre decimal en operaciones BCD

5. ejemplos: MC68000

Modos de direccionamiento

Modo de direccionamiento	Cálculo de EA	Sintaxis
<b>Direccionamiento directo de registro</b>		
Directo de registro de datos	EA = Dn	Dn
Directo de registro de direcciones	EA = An	An
<b>Direccionamiento indirecto de registro</b>		
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 <sub>16</sub>	d <sub>16</sub> (An)
Indirecto de registro indexado con desplazamiento	EA = (An) + (Ri) + d <sub>8</sub>	d <sub>8</sub> (An,Ri.X)
<b>Direccionamiento absoluto</b>		
Absoluto corto	EA = XXXX	XXXX
Absoluto largo	EA = XXXXXX	XXXXXX
<b>Direccionamiento relativo al contador de programa</b>		
Relativo al PC con desplazamiento	EA = (PC) + d <sub>16</sub>	d <sub>16</sub> (PC)
Relativo al PC indexado con desplazamiento	EA = (PC) + (Ri) + d <sub>8</sub>	d <sub>8</sub> (PC,Ri.X)
<b>Direccionamiento inmediato</b>	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<sub>8</sub> = Desplazamiento de 8 bits  
d<sub>16</sub> = Desplazamiento de 16 bits  
PC = Contador de Programa

5. ejemplos: MC68000

Formato de las instrucciones

- ☒ Características
  - Instrucciones de 0, 1 y 2 operandos
  - 9 formatos distintos
  - Longitud entre 1 y 5 palabras
- ☒ Contenidos de una instrucción máquina
  - **Palabra operación** (OW, siempre presente)
    - Primera palabra de la instrucción
      - ✓ Código de operación
      - ✓ Modos de direccionamiento y tamaño de los operandos
  - **Palabras de extensión** (EW, de 0 a 4)
    - Información adicional de los operandos
      - ✓ Valor inmediato
      - ✓ Desplazamiento
- ☒ Codificación modos de direccionamiento
  - Dos campos: **Modo** y **Registro** (6 bits)

Formato genérico de la instrucción máquina

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
OW: Palabra operación( opcode, tamaño, EA)
EW para operando inmediato (si existe, una o dos palabras)
EW para dirección efectiva fuente (si existe, una o dos palabras)
EW para dirección efectiva destino (si existe, una o dos palabras)

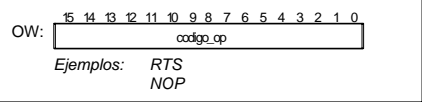
- Instrucciones de 0 operandos**  
**Sintaxis:** OP
- Instrucciones de 1 operando**  
**Sintaxis:** OP destino  
**Operación:** destino → OP (destino)
- Instrucciones de 2 operando**  
**Sintaxis:** OP fuente, destino  
**Operación:** destino → (fuente) OP (destino)

5. ejemplos: MC68000

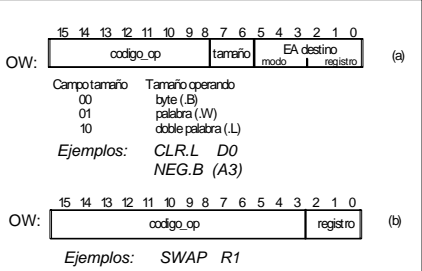
Formato de las instrucciones (cont.)

Codificación de los modos de direccionamiento	EA	
	modo	Registro
Directo de reg. de datos	000	Nº de reg. (Dn)
Directo de reg. de direcciones	001	Nº de reg. (An)
Indirecto de reg.	010	Nº de reg. (An)
Indirecto de reg. con postincrement.	011	Nº de reg. (An)
Indirecto de reg. con predecrem.	100	Nº de reg. (An)
Indirecto de reg. con desplaz.	101	Nº de reg. (An)
Indirecto de reg. Index. con desplaz.	110	Nº de reg. (An)
Absoluto corto	111	000
Absoluto largo	111	001
Relativo al PC con desplaz.	111	010
Relativo al PC index. con desplaz.	111	011
Inmediato	111	100

Instrucciones sin operandos (1 formato)



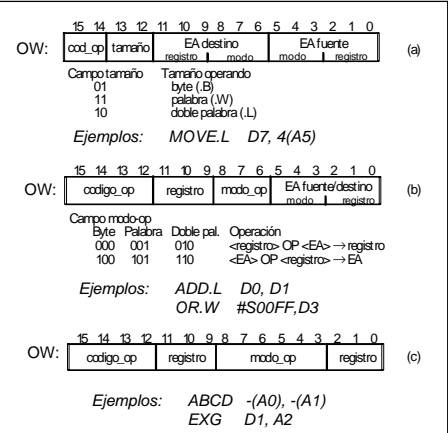
Instrucciones con un operando (2 formatos)



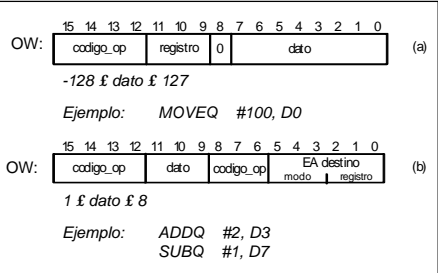
5. ejemplos: MC68000

Formato de las instrucciones (cont.)

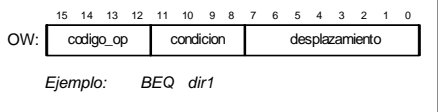
Instrucciones con 2 operandos (3 formatos)



Instrucciones de dos operandos con operando inmediato corto (2 formatos)



Instrucciones de bifurcación condicional (Bcc, 1 formato)



5. ejemplos: MC68000

Repertorio de instrucciones

Notación utilizada

CÓDIGOS DE CONDICIÓN:

*	Afectado.
-	No afectado.
0	Puesto a 0 (clear).
1	Puesto a 1 (set).
U	No definido.

Modo de direccionamiento	Categorías de EA							
	ea	dea	mea	cea	aea	adea	amea	acea
Dn	X	X			X	X		
An	X				X			
(An)	X	X	X	X	X	X	X	X
(An)+	X	X	X		X	X	X	
-(An)	X	X	X		X	X	X	
d <sub>16</sub> (An)	X	X	X	X	X	X	X	X
d <sub>16</sub> (An,Ri.X)	X	X	X	X	X	X	X	X
XXXX	X	X	X	X	X	X	X	X
XXXXXX	X	X	X	X	X	X	X	X
d <sub>16</sub> (PC)	X	X	X	X				
d <sub>16</sub> (PC,Ri.X)	X	X	X	X				
#XXXXXXXX	X	X	X					

estructura y tecnología de computadores

• • • • • • •

5. ejemplos: MC68000

Repertorio de instrucciones

MOVIMIENTO DE DATOS				
Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
EXG	EXG Rm,Rn	L	Rm←Rn	-----
LEA	LEA <cea>,An	L	dirección fuente→An	-----
LINK	LINK An,<d16>	No	An→-(SP); SP→An SP+d16→SP	-----
MOVE	MOVE.t <ea>,<adea>	L,W,B	(fuente)→destino	- * * 0 0
MOVE from SR	MOVE SR,<adea>	W	SR→destino	-----
MOVE to CCR	MOVE <adea>,CCR	W	(fuente)→CCR	*****
MOVE to SR	MOVE <adea>,SR	W	IF S=1 SR→destino ELSE EXCEPCIÓN	*****
MOVE USP	MOVE USP,An	L	IF S=1 USP→An ELSE EXCEPCIÓN	*****
MOVE USP	MOVE An,USP	L	IF S=1 An→USP ELSE EXCEPCIÓN	*****
MOVEA	MOVEA.t <ea>,An	L,W	(fuente)→destino	-----
MOVEM	MOVEM.t <lista_reg>,<cea>	L,W	registros→destino	-----
MOVEM	MOVEM.t <lista_reg>,-(An)	L,W	(fuente)→registros	-----
MOVEM	MOVEM.t <cea>,<lista_reg>	L,W	(fuente)→registros	-----
MOVEP	MOVEP.t d(Am),Dn	L,W	(fuente)→destino	-----
MOVEP	MOVEP.t d(Am),Dn	L,W	(fuente)→destino	-----
MOVEQ	MOVEQ #<d8>,Dn	L	dato (8 bits)→destino	- * * 0 0
PEA	PEA <cea>	L	dirección→-(SP)	-----
SWAP	SWAP Dn	W	Dn[31:16]↔Dn[15:0]	- * * 0 0
UNLK	UNLK An	No	An→SP; (SP)←An	-----

estructura y tecnología de computadores

• • • • • • •

## 5. ejemplos: MC68000

### ARITMÉTICA ENTERA

Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
ADD	ADD.l <ea>, Dn ADD.l Dn, <amea>	L, W, B	(fuente)+(destino)→destino	*****
ADDA	ADDA.l <ea>, An	L, W	(fuente)+(destino)→destino	-----
ADDI	ADDI.l # <dato>, <adea>	L, W, B	Dato+(destino)→destino	*****
ADDQ	ADDQ.l # <d3>, <aea>	L, W, B	d3+(destino)→destino	*****
ADDX	ADDX.l Dm, Dn o ADDX.l -(Am), -(An)	L, W, B	(fuente)+(destino)+X→destino	*****
CLR	CLR.l <adea>	L, W, B	0→destino	-0100
CMP	CMP.l <ea>, Dn	L, W, B	(destino)-(fuente) activa CCR	*****
CMPA	CMPA.l <ea>, An	L, W	(destino)-(fuente) activa CCR	*****
CMPI	CMPI.l # <dato>, <adea>	L, W, B	(destino)-Dato; activa CCR	*****
CMPM	CMPM.l (Am)+ (An)	L, W, B	(destino)-(fuente) activa CCR	*****
DIVS	DIVS <dea>, Dn	W	(destino)/(fuente)→destino; Cociente→destino[15:0]; Resto→destino[31:16]	-----0
DIVU	DIVU <dea>, Dn	W	(destino)/(fuente)→destino; Cociente→destino[15:0]; Resto→destino[31:16]	-----0
EXT	EXT.l Dn	L, W	Dn(Extend en signo)→Dn	-----0
MULS	MULS <dea>, Dn	W	(fuente)*(destino)→destino	-----0
MULU	MULU <dea>, Dn	W	(fuente)*(destino)→destino	-----0
NEG	NEG.l <adea>	L, W, B	0-(destino)→destino	*****
NEGX	NEGX.l <adea>	L, W, B	0-(destino)-X→destino	*****
SUB	SUB.l <ea>, Dn SUB.l Dn, <amea>	L, W, B	(destino)-(fuente)→destino	*****
SUBA	SUBA.l <ea>, An	L, W	(destino)-(fuente)→destino	-----
SUBI	SUBI.l # <dato>, <adea>	L, W, B	(destino)-Dato→destino	*****
SUBQ	SUBQ.l # <d3>, <aea>	L, W, B	(destino)-d3→destino	*****
SUBX	SUBX.l Dm, Dn SUBX.l -(Am), -(An)	L, W, B	(destino)-(fuente)-X→destino	*****
TAS	TAS <adea>	B	Test (destino)→CCR; 1→destino[bit 7]	-----0
TST	TST <adea>	L, W, B	Test (destino)→CCR	-----0

estructura y tecnología de computadores

## 5. ejemplos: MC68000

### OPERACIONES LÓGICAS

Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
AND	AND.l <dea>, Dn AND.l Dn, <amea>	L, W, B	(fuente)<AND>(destino)→destino	-----0
ANDI	ANDI.l # <dato>, <adea>	L, W, B	dato<AND>(destino)→destino	-----0
ANDI to CCR	ANDI # <d8>, CCR	B	d8<AND>CCR →CCR	*****
ANDI to SR*	ANDI # <d16>, SR	W	IF S=1 d16 <AND>SR→SR ELSE EXCEPCIÓN	*****
EOR	EOR.l Dn, <adea>	L, W, B	(fuente)<XOR>(destino)→destino	-----0
EORI	EORI.l # <dato>, <adea>	L, W, B	dato<XOR>(destino)→destino	-----0
EORI to CCR	EORI # <d8>, CCR	B	d8<XOR>CCR →CCR	*****
EORI to SR*	EORI # <d16>, SR	W	IF S=1 d16<XOR>SR→SR ELSE EXCEPCIÓN	*****
NOT	NOT.l <adea>	L, W, B	<NOT>(destino)→destino	-----0
OR	OR.l <dea>, Dn OR.l Dn, <amea>	L, W, B	(fuente)<OR>(destino)→destino	-----0
ORI	ORI.l # <dato>, <adea>	L, W, B	dato<OR>(destino)→destino	-----0
ORI to CCR	ORI # <d8>, CCR	B	d8<OR>CCR →CCR	*****
ORI to SR*	ORI # <d16>, SR	W	IF S=1 d16<OR>SR→SR ELSE EXCEPCIÓN	*****

### DESPLAZAMIENTOS Y ROTACIONES

Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
ASL/ASR	ASL/ASR Dm, Dn ASL/ASR # <d3>, Dn	L, W, B	(destino) desplaz. aritmét. de <num> bits→destino	*****
ASL/ASR	ASL/ASR <amea>	W (contador desplaz.=1)	(destino) desplaz. aritmét. de 1 bit →destino	*****
LSL/LSR	Igual que ASL/ASR	Idem	(destino) desplaz. lógico de <num> (o 1)bits→destino	***0*
ROL/ROR	Igual que ASL/ASR	Idem	(destino) rotado <num> (o 1)bits→destino	***0*
ROXL/ROXR	Igual que ASL/ASR	Idem	(destino) rotado (con exten.) <num> (o 1)bits→destino	***0*

estructura y tecnología de computadores

5. ejemplos: MC68000

MANIPULACIÓN DE BITS				
Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
BCHG	BCHG Dm,Dn o BCHG #<d8>,Dn	L (bit mod 32)	<NOT>(destino[bit])→flag Z; <NOT>(destino[bit])→destino[bit]	--*--
BCHG	BCHG Dm,<amea> o BCHG #<d8>,<amea>	B (bit mod 8)	Igual	--*--
BCLR	Igual que BCHG		<NOT>(destino[bit])→flag Z; 0→destino[bit]	--*--
BSET	Igual que BCHG		<NOT>(destino[bit])→flag Z; 1→destino[bit]	--*--
BTST	BTST Dm,Dn o BTST #<d8>,Dn	L (bit mod 32)	<NOT>(destino[bit])→flag Z	--*--
BTST	BTST Dm,<mea> o BTST #<d8>,<mea>	B (bit mod 8)	Igual	--*--

OPERACIONES DE CONTROL DE PROGRAMA				
Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
Bcc	Bcc <etiqueta>	Desplaz. 16 bits	IF cc True PC+despl.→PC	-----
Bcc.S	Bcc.S <etiqueta>	Desplaz. 8 bits	IF cc True PC+despl.→PC	-----
BRA	BRA <etiqueta>	Despl. 8 o 16 bits	PC+desp.→PC	-----
BSR	BSR <etiqueta>	Despl. 8 o 16 bits	PC→(SP); PC+desp.→PC	-----
DBcc	DBccDm, <etiqueta>	Desplaz. 16 bits	IF cc False Dm-1 → Dm IF Dm ≠-1 PC+desp.→PC ELSE PC+2→PC (instr. sig.)	-----
JMP	JMP <cea>	No	destino→PC	-----
JSR	JSR <cea>	No	PC→(SP); destino→PC	-----
Scc	Scc <adea>	B	IF cc True \$FF→destino ELSE \$00→destino	-----
RTR	RTR	No	(SP)+→CCR; (SP)+→PC	*****
RTS	RTS	No	(SP)+→PC	-----

5. ejemplos: MC68000

OPERACIONES EN BCD				
Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
ABCD	ABCD Dm,Dn ABCD -(Am),-(An)	B	(fuente <sub>10</sub> )+(destino <sub>10</sub> )→X→destino <sub>10</sub>	*U*U*
NBCD	NBCD<adea>	B	0-(destino <sub>10</sub> )→X→destino <sub>10</sub>	*U*U*
SBCD	SBCD Dm,Dn SBCD -(Am),-(An)	B	(fuente <sub>10</sub> )-(destino <sub>10</sub> )→X→destino <sub>10</sub>	*U*U*

OPERACIONES DE CONTROL DEL SISTEMA				
Mnemotécnico	Sintaxis	Tamaño	Función	X N Z V C
RESET*	RESET	No	IF S=1 activa RESET ELSE EXCEPCIÓN	-----
RTE*	RTE	No	IF S=1 (SP)+→SP;(SP)+→PC ELSE EXCEPCIÓN	*****
STOP*	STOP #<d16>	No	IF S=1 d16→SR;PC+4→PC; Pausa hasta excepcion ELSE EXCEPCIÓN	*****
CHK	CHK <dea>,Dn	W	IF Dn<0 OR Dn>(fuente) EXCEPCIÓN	-*UUU
ILLEGAL	ILLEGAL	No	PC→-(SSP);SR→-(SSP) Vector #4→PC	-----
TRAP	TRAP #<d4>	No	PC→-(SSP);SR→-(SSP) Vector #4→PC	-----
TRAPV	TRAPV	No	IF V=1 EXCEPCIÓN	-----
NOP	NOP	No	PC+2→PC	-----



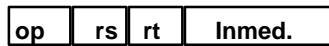
## 5. ejemplos: arquitectura MIPS

### Modos de direccionamiento

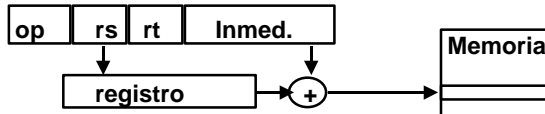
☒ **Directo de registro**



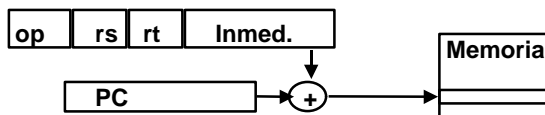
☒ **Inmediato**



☒ **Indirecto con desplazamiento**



☒ **Indirecto con desplazamiento relativo a PC**



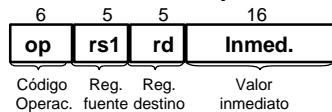
estructura y tecnología de computadores

• • • • •

## 5. ejemplos: arquitectura MIPS

### Formato de la instrucción máquina

☒ **Instrucción de tipo I**



➤ **Instrucciones de LOAD/STORE**

Ej.: *LW \$1, 30(\$2)*      $R1 \leftarrow \text{Mem}(30+(R2))$   
*SF 60(\$5), \$10*      $\text{Mem}(60+(R5)) \leftarrow (R10)$

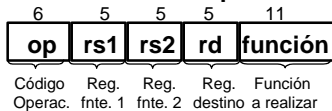
➤ **Instruc. con operando inmediato**

Ej.: *ADDI \$7, \$9, #30*      $R7 \leftarrow (R9) + 30$

➤ **Instruc. de bifurcación condicional**

Ej.: *BNE \$5, \$6, #0x0C*     if  $(R5 \neq R6)$  then  $PC \leftarrow 0x0C$

☒ **Instrucción de tipo R**



➤ **Instruc. aritméticas, lógicas y desplazamiento de tipo de registro-registro**

Ej.: *SUB \$1, \$2, \$3*      $R1 \leftarrow (R2) - (R3)$   
*SLL \$1, \$2, \$3*      $R1 \leftarrow (R2) \ll (R3)$

➤ **Campo función: Especifica el tipo de operación a realizar**  
 Suma, Resta, Mult. Div., And, Or, Desplaz., etc.

☒ **Instrucción de tipo J**



➤ **Instrucciones de salto**

Ej.: *J dir*      $PC \leftarrow \text{dir}$

estructura y tecnología de computadores

• • • • •



## 5. ejemplos: arquitectura MIPS

### Repertorio de instrucciones

#### Instrucción de movimiento de datos

Instrucción	Significado	Ejemplo	Operación
LB	Load byte	LB \$2, 40(\$3)	$R2[24-31] \leftarrow \text{Mem}(40+(R3))_8; R1[0-23] \leftarrow (\text{signo}(\text{Mem}(40+(R3))))^{24}$
LBU	Load byte unsigned	LBU \$2, 40(\$3)	$R2[24-31] \leftarrow \text{Mem}(40+(R3))_8; R1[0-23] \leftarrow (0)^{24}$
LH	Load half word	LH \$2, 40(\$3)	$R2[16-31] \leftarrow \text{Mem}(40+(R3))_{16}; R1[0-15] \leftarrow (\text{signo}(\text{Mem}(40+(R3))))^{16}$
LHU	Load half word unsigned	LHU \$2, 40(\$3)	$R2[16-31] \leftarrow \text{Mem}(40+(R3))_{16}; R1[0-15] \leftarrow (0)^{16}$
LW	Load word	LW \$2, 40(\$3)	$R2[0-31] \leftarrow \text{Mem}(40+(R3))_{32}$
LF	Load Float	LF \$2, 40(\$3)	$F2[0-31] \leftarrow \text{Mem}(40+(R3))_{32}$
LD	Load double float	LD \$2, 40(\$3)	$F2[0-31], F3[0-31] \leftarrow \text{Mem}(40+(R3))_{64}$
SB	Store byte	SB 40(\$3), \$2	$\text{Mem}(40+(R3))_8 \leftarrow R2[24-31]$
SH	Store half word	SB 40(\$3), \$2	$\text{Mem}(40+(R3))_{16} \leftarrow R2[16-31]$
SW	Store word	SB 40(\$3), \$2	$\text{Mem}(40+(R3))_{32} \leftarrow R2[0-31]$
SF	Store float	SB 40(\$3), \$2	$\text{Mem}(40+(R3))_{32} \leftarrow F2[0-31]$
SD	Store double float	SB 40(\$3), \$2	$\text{Mem}(40+(R3))_{64} \leftarrow F2[0-31], F3[0-31]$

estructura y tecnología de computadores

## 5. ejemplos: arquitectura MIPS

### Repertorio de instrucciones

#### Instrucción de aritmética entera, lógicas y desplazamiento

Instrucción	Significado	Ejemplo	Operación
ADD/ADDU	Suma con/sin signo	ADD \$1,\$2,\$3	$R1 \leftarrow (R2) + (R3)$
ADDI/ADDIU	Suma inmediato con/sin signo	ADDI \$1,\$2,#5	$R1 \leftarrow (R2) + 5$
SUB/SUBU	Resta con/sin signo	SUB \$1,\$2,\$3	$R1 \leftarrow (R2) - (R3)$
SUBI/SUBIU	Resta inmediato con/sin signo	SUBI \$1,\$2,#5	$R1 \leftarrow (R2) - 5$
MULT/MULTU	Multipliación con/sin signo	MULT \$1,\$2,\$3	$R1 \leftarrow (R2) * (R3)$
DIV/DIVU	División con/sin signo	DIV \$1,\$2,\$3	$R1[0-15] \leftarrow \text{Cociente } ((R2) / (R3))$ $R1[16-31] \leftarrow \text{Resto } ((R2) \text{ mod } (R3))$
AND	Y lógica	AND \$1,\$2,\$3	$R1 \leftarrow (R2) \text{ AND } (R3)$
ANDI	Y lógica con inmediato	ANDI \$1,\$2,#5	$R1 \leftarrow (R2) \text{ AND } 5$
OR	O lógica	OR \$1,\$2,\$3	$R1 \leftarrow (R2) \text{ OR } (R3)$
ORI	O lógica con inmediato	ORI \$1,\$2,#5	$R1 \leftarrow (R2) \text{ OR } 5$
XOR	O exclusiva	XOR \$1,\$2,\$3	$R1 \leftarrow (R2) \text{ XOR } (R3)$
XORI	O exclusiva con inmediato	XORI \$1,\$2,#5	$R1 \leftarrow (R2) \text{ XOR } 5$
SLL/SRL	Desplaz. lógico Izda./decha.	SLL \$1,\$2,\$3	$R1 \leftarrow (R2) \ll (R3)$
SLLI/SRLI	Desp. Lóg. Izda./decha. con inmed.	SLLI \$1,\$2,#4	$R1 \leftarrow (R2) \ll 4$
SRA	Desplaz. aritmético derecha	SRA \$1,\$2,\$3	$R1 \leftarrow (R2) \gg (R3) \text{ (mantiene signo)}$
SRAI	Desplaz. aritm. Dcha. con inmed.	SRAI \$1,\$2,#4	$R1 \leftarrow (R2) \gg 4 \text{ (mantiene signo)}$
Scc *	Set condicional	SLT \$1,\$2,\$3	if $(R2) < (R3)$ then $R1 \leftarrow 1$ , else $R1 \leftarrow 0$
Sccl *	Set condicional con inmediato	SEQ! \$1,\$2,#0	if $(R2) = 0$ then $R1 \leftarrow 1$ , else $R1 \leftarrow 0$

(\*) cc = LT, GT, LE, GE, EQ, NE

estructura y tecnología de computadores

## 5. ejemplos: arquitectura MIPS

### Repertorio de instrucciones

#### Instrucción de control de flujo

Instrucción	Significado	Ejemplo	Operación
J	Salto (jump)	J dir	$PC \leftarrow \text{dir}$
JR	Salto con registro	JR \$2	$PC \leftarrow (R2)$
BEQZ	Bifurcación condic. si igual a cero	BEQZ dir, \$4	if $(R4) = 0$ then $PC \leftarrow (R4)$
BNEQZ	Bifurcación condic. si distinto de cero	BNEQZ dir, \$4	if $(R4) \neq 0$ then $PC \leftarrow (R4)$
JAL	Salto y link	JAL dir	$R31 \leftarrow (PC) + 4$ ; $PC \leftarrow \text{dir}$
JALR	Salto y link con registro	JALR dir, \$2	$R31 \leftarrow (PC) + 4$ ; $PC \leftarrow (R2)$
TRAP	Provoca una excepción		
RFE	Retorno de excepción		

#### Instrucción de aritmética en punto flotante

Instrucción	Significado
ADDF/ADDD	Suma en punto flotante simple/doble precisión
SUBF/SUBD	Resta en punto flotante simple/doble precisión
MULTF/MULTD	Multipliación en punto flotante simple/doble precisión
DIVF/DIVD	División en punto flotante simple/doble precisión
CVTI2F/CVTI2D	Convierte entero a real simple precisión / doble precisión
CVTF2I/CVTF2D	Convierte real simple precisión a entero / real doble precisión
CVTD2I/CVTD2F	Convierte real doble precisión a entero / real simple precisión