TEMA 1. SISTEMAS INFORMÁTICOS. ESTRUCTURA FUNCIONAL. ENSAMBLADOR

Fundamentos de Hardware

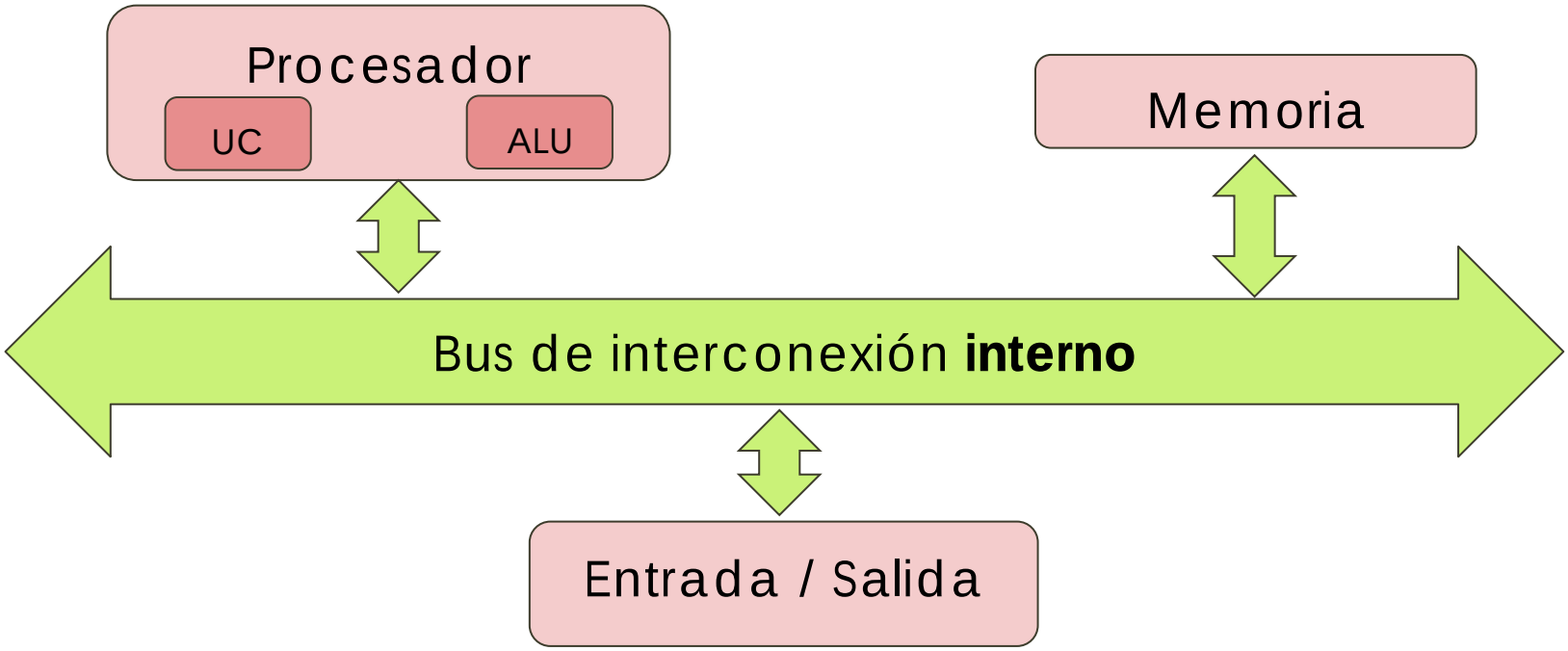
1º ASIR

ÍNDICE

1. INTRODUCCIÓN

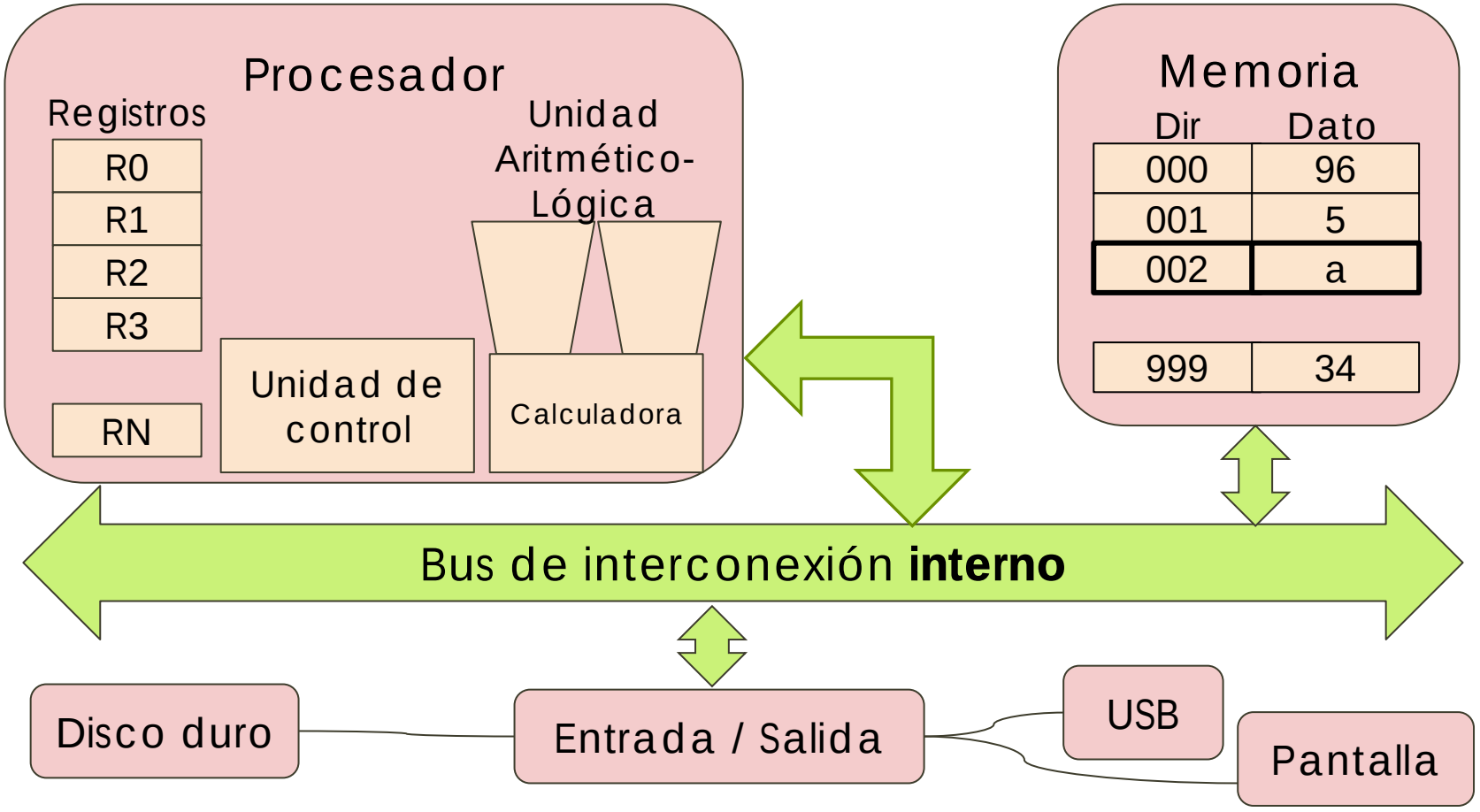
ARQUITECTURA VON NEUMANN

La arquitectura de John von Neumann es la que se utiliza en la mayoría de los ordenadores actuales.



1. INTRODUCCIÓN

ARQUITECTURA VON NEUMANN



1. INTRODUCCIÓN

LENGUAJE ENSAMBLADOR

Lenguaje utilizado para escribir programas informáticos de bajo nivel. Utiliza nemotécnicos para cada instrucción, para los registros, las posiciones de memoria, ...

Ejemplos de instrucciones en lenguaje ensamblador:

MOV destino, origen

LOAD dir_origen, destino
ADD operando1, operando2
PUSH
POP
GOTO
...

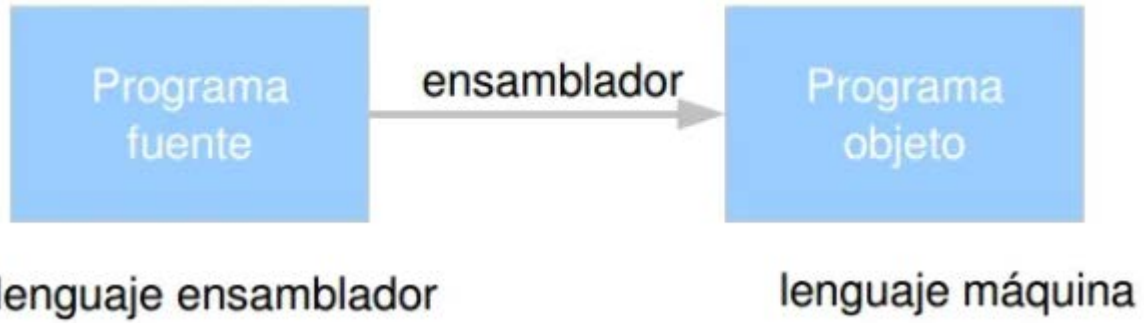
MOV AX, 6
MOV BX, AX
LOAD #0003, CX
ADD BX, CX

1. INTRODUCCIÓN

LENGUAJE ENSAMBLADOR

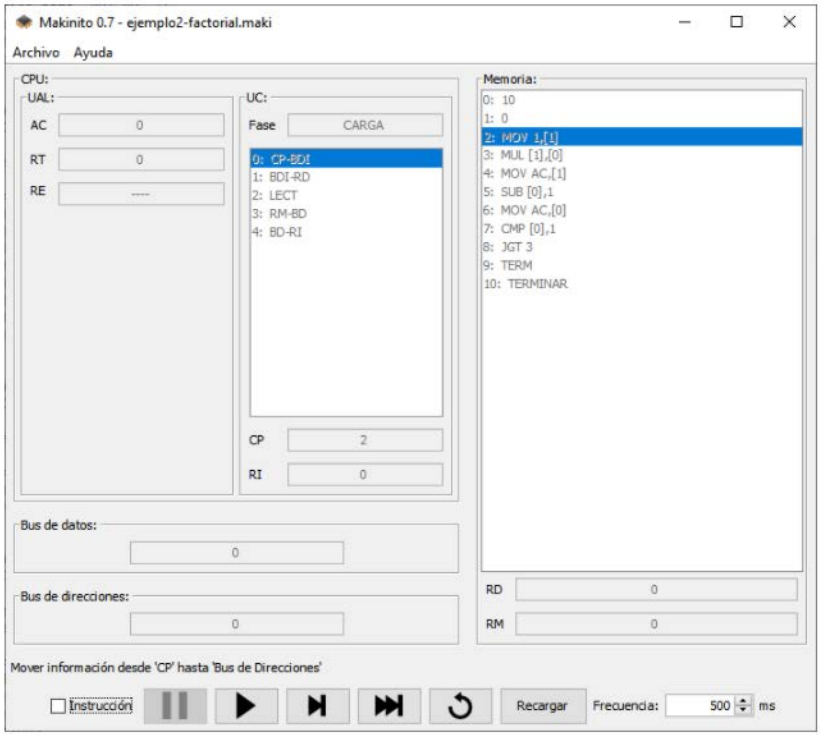
Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente, sino que hay que “traducirlo” a lenguaje máquina.

El programa encargado de este proceso se denomina ensamblador



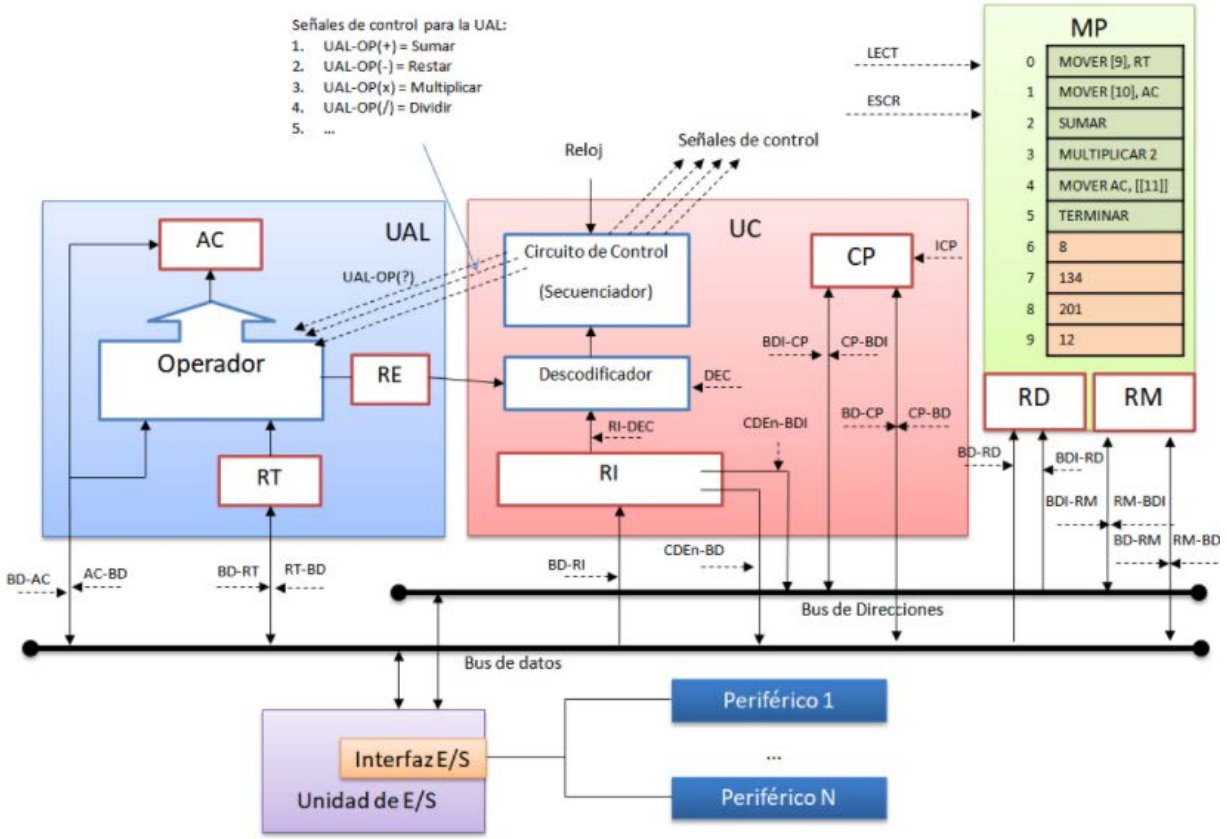
2. MAKINITO

Programa que simula el funcionamiento de una máquina con arquitectura de Von Neumann.



2. MAKINITO

La arquitectura simulada por Makinito es la siguiente:



2. MAKINITO

REGISTROS

ALU

AC: Acumulador
RE: Registro de Estado
RT: Registro Temporal

UC

CP: Contador de programa
RI: Registro de Instrucción

Memoria

RD: Registro de direcciones
RM: Registro de memoria

2. MAKINITO

JUEGO DE INSTRUCCIONES

INSTRUCCIÓN	DESCRIPCIÓN
MOV x, y	Copia el valor de **x** en **y**
ADD [x [, y]]	Suma los valores **x** e **y** y guarda el resultado en **AC**
SUB [x [, y]]	Resta los valores **x** e **y** y guarda el resultado en **AC**
MUL [x [, y]]	Multiplica los valores **x** e **y** y guarda el resultado en **AC**
DIV [x [, y]]	Divide los valores **x** e **y** y guarda el resultado en **AC**
CMP [x [, y]]	Compara los valores **x** e **y** y modifica el registro de estados

2. MAKINITO

JUEGO DE INSTRUCCIONES

INSTRUCCIÓN	DESCRIPCIÓN
JMP d	Salta a la dirección de memoria **d**
JE d	Salta a la dirección de memoria **d** si **x=y**
JLE d	Salta a la dirección de memoria **d** si **x<=y**
JGE d	Salta a la dirección de memoria **d** si **x>=y**
JLT d	Salta a la dirección de memoria **d** si **x<y**
JGT d	Salta a la dirección de memoria **d** si **x>y**
JNE d	Salta a la dirección de memoria **d** si **x≠y**
TERM	Detiene la máquina

2. MAKINITO

JUEGO DE INSTRUCCIONES

x e **y** pueden ser valores inmediatos, directos, indirectos o registros

x, [, y]: si no se especifica el 2º operando se usa por defecto el registro AC.

[x, y]: Si no se especifica ningún operando se usan por defecto los registros AC y RT respectivamente.

2. MAKINITO

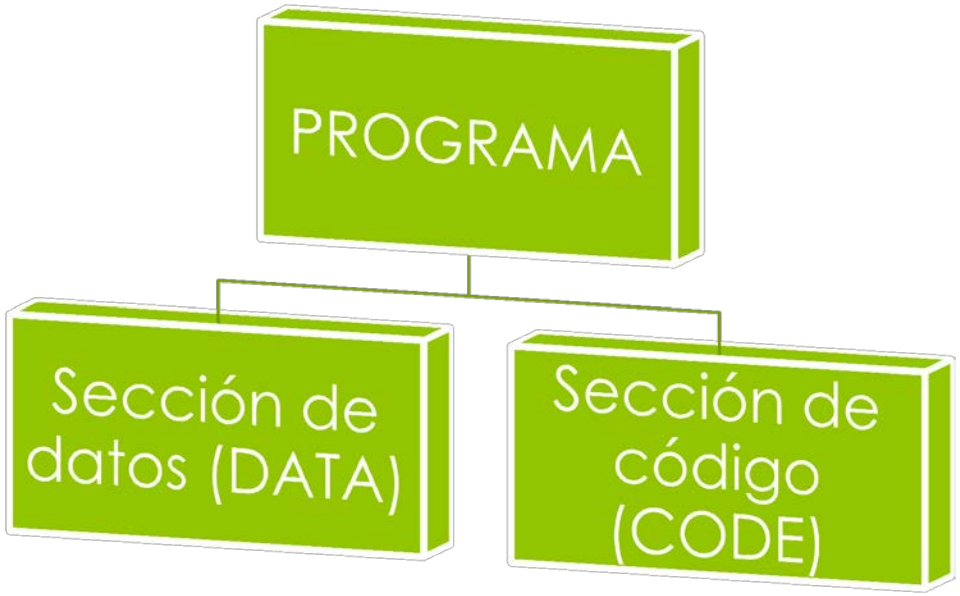
MODOS DE DIRECCIONAMIENTO

MODO	DESCRIPCIÓN	EJEMPLOS
Inmediato	El número es el dato	13 (dec), Dh (hex), 1101b (bin)
Directo	El número entre corchetes es la dirección de memoria del dato	[13]
Indirecto	El número entre dobles corchetes es la dirección de memoria de la dirección de memoria del dato	[[13]]
Registro	El dato está almacenado en un registro	RT, AC, RI, CP, ...

2. MAKINITO

PROGRAMA EN LENGUAJE ENSAMBLADOR

Los programas se componen de dos secciones:



2. MAKINITO

PROGRAMA – SECCIÓN DE DATOS

En esta sección se declaran las variables.

```
BEGIN-DATA
        ...
END-DATA
```

Cada declaración de variable se especifica con el nombre y el valor

```
NOMBRE=VALOR
```

El valor puede ser decimal (23), hexadecimal (3Ah) o binario (1010b). Es posible usar ? como valor si no queremos inicializar la variable al comienzo del programa.

2. MAKINITO

PROGRAMA – SECCIÓN DE DATOS

Es necesario que todo lo que pongamos en la sección de datos esté indentado con espacios en blanco o tabulaciones.

Ejemplo:

```
BEGIN-DATA
        NUM1=10
        NUM2=Ah
        NUM3=1010b
        RES=?
END-DATA
```

2. MAKINITO

PROGRAMA – SECCIÓN DE CÓDIGO

En esta sección están las instrucciones del programa. Su sintaxis es la siguiente:

```
[ETIQUETA:]        MNEMOTÉCNICO [OPERANDO1[,OPERANDO2]]
```

Las etiquetas son necesarias para los saltos.

El mnemotécnico corresponde a cada una de las instrucciones del juego de instrucciones (MOV, ADD, ...)

Los operandos son los parámetros que pasamos a la instrucción. Se pueden utilizar los distintos modos de direccionamiento (inmediato, directo, ...)

2. MAKINITO

PROGRAMA – COMENTARIOS

Se pueden añadir comentarios poniendo un ; (punto y coma al principio de la línea

```
; esto es un comentario
```

O al final de una línea

```
ADD 1, 3 ; esto es otro comentario
```

2. MAKINITO

PROGRAMA – EJEMPLO

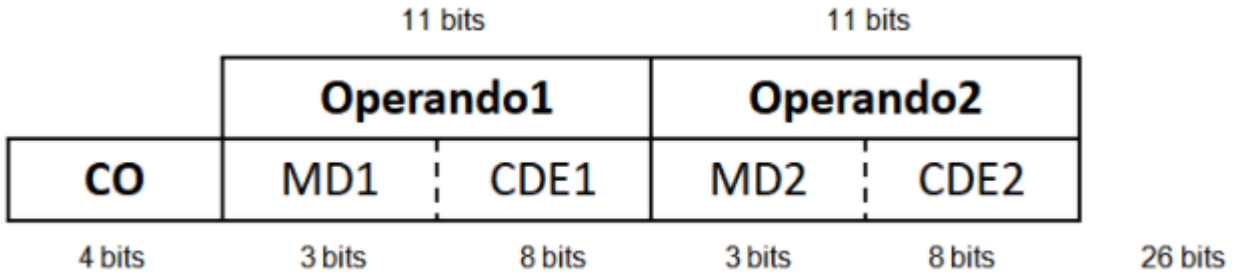
```
; Programa que calcula el factorial de NUM y lo guarda en FACT
; Segmento de datos
BEGIN-DATA
        NUM=10        ; número a calcular el factorial
        FACT=?        ; donde se guarda el resultado
END-DATA

; Segmento de código
BEGIN-CODE
        MOV 1,[FACT]
BUCLE:  MUL [FACT],[NUM]
        MOV AC,[FACT]
        SUB [NUM],1
        MOV AC,[NUM]
        CMP [NUM],1
        JGT BUCLE
        TERM          ; se detiene la máquina
END-CODE
```

2. MAKINITO

FORMATO DE LAS INSTRUCCIONES

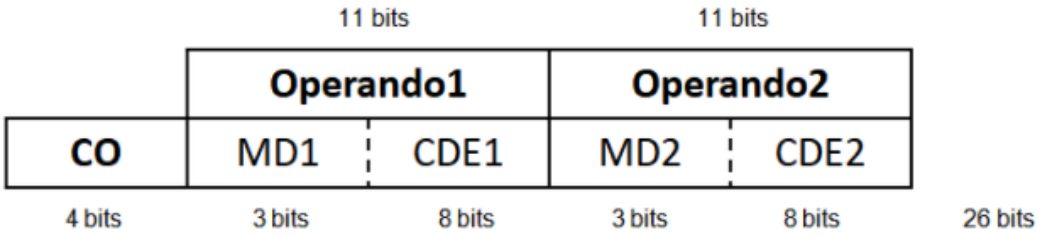
El formato de las instrucciones en código máquina para la arquitectura Makinito es:



CO: Código de operación
MD1: modo de direccionamiento del primer operando
CDE1: dirección efectiva del primer operando
MD2: modo de direccionamiento del segundo operando
CDE2: dirección efectiva del segundo operando

2. MAKINITO

FORMATO DE LAS INSTRUCCIONES



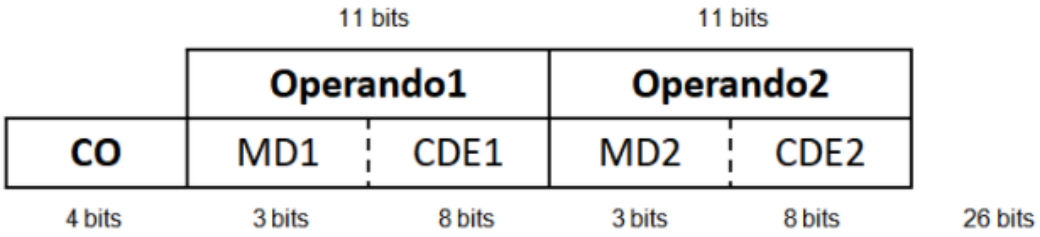
CO	Valor	Binario
MOV	0	0000
ADD	1	0001
SUB	2	0010
MUL	3	0011
DIV	4	0100
…	…	…

MD	Valor	Binario
Sin usar	0	000
Inmediato	1	001
Directo	2	010
Indirecto	3	011
Registro	4	100

Registro	Valor	Binario
AC	0	00000000
RT	1	00000001
RE	2	00000010
RI	3	00000011
CP	4	00000100

2. MAKINITO

FORMATO DE LAS INSTRUCCIONES



Ejemplo de codificación de instrucciones en código máquina:

Instrucción	CO	MD1	CDE1	MD2	CDE2	Código máquina
MOV 13, AC	0000	001	00001101	100	00000000	00000010000110110000000000
ADD [7]	0001	010	00000111	000	00000000	00010100000011100000000000