**Universidad de Jaén**
**E.U.P. Linares**

**Dpto. Telecomunicaciones**
**Área de Ingeniería Telemática**

# Sebastián García Galán

**Sgalan@ujaen.es**

# TEMA 3: NIVEL DE MÁQUINA CONVENCIONAL

## 3.1 SÍMPLEZ.

MODELO ESTRUCTURAL
MODELO FUNCIONAL
PROBLEMAS

## 3.2 PROGRAMACIÓN DE SÍMPLEZ.

SUMA DE DOS NÚMEROS
CONSTRUCCIÓN DE BUCLES
AUTOMODIFICACIÓN DE PROGRAMAS
INTRODUCCIÓN AL USO DE SUBPROGRAMAS
INTRODUCCIÓN A LAS COMUNICACIONES CON EL EXTERIOR
PROBLEMAS

## 3.3 SÍMPLEZ+I4.

MODELO ESTRUCTURAL
MODELO FUNCIONAL
MODOS DE DIRECCIONAMIENTO
CONVENIOS SIMBÓLICOS
USO DEL REGISTRO DE ÍNDICE COMO CONTADOR
PUNTEROS E ÍNDICES
INTERRUPCIONES
PROBLEMAS

# TEMA 3: NIVEL DE MÁQUINA CONVENCIONAL

## 3.4 DIRECCIONAMIENTO.

- INTRODUCCIÓN
- DIRECCIONAMIENTO DIRECTO
- DIRECCIONAMIENTO INMEDIATO
- DIRECCIONAMIENTO INDEXADO
- DIRECCIONAMIENTO RELATIVO
  - A PROGRAMA
  - A PÁGINA
  - A BASE
  - A SEGMENTO
- DIRECCIONAMIENTO BASADO EN R.P.G.
- MODOS AUTOINCREMENTO Y AUTODECREMENTO

## 3.5 PILAS Y SUBPROGRAMAS.

- MAQUINAS DE PILAS
- SIMULACIÓN DE UNA PILA
- USO DE UNA PILA PARA LOS SUBPROGRAMAS E INTERRUPCIONES

# TEMA 3: NIVEL DE MÁQUINA CONVENCIONAL

## 3.6 REPERTORIOS DE INSTRUCCIONES.

MOVIMIENTO DE DATOS
ARITMÉTICAS LÓGICAS Y DE COMPARACIÓN
DE DESPLAZAMIENTOS
DE TRANSFERENCIA DE CONTROL
DE GOBIERNO

## 3.7 LENGUAJES ENSAMBLADORES.

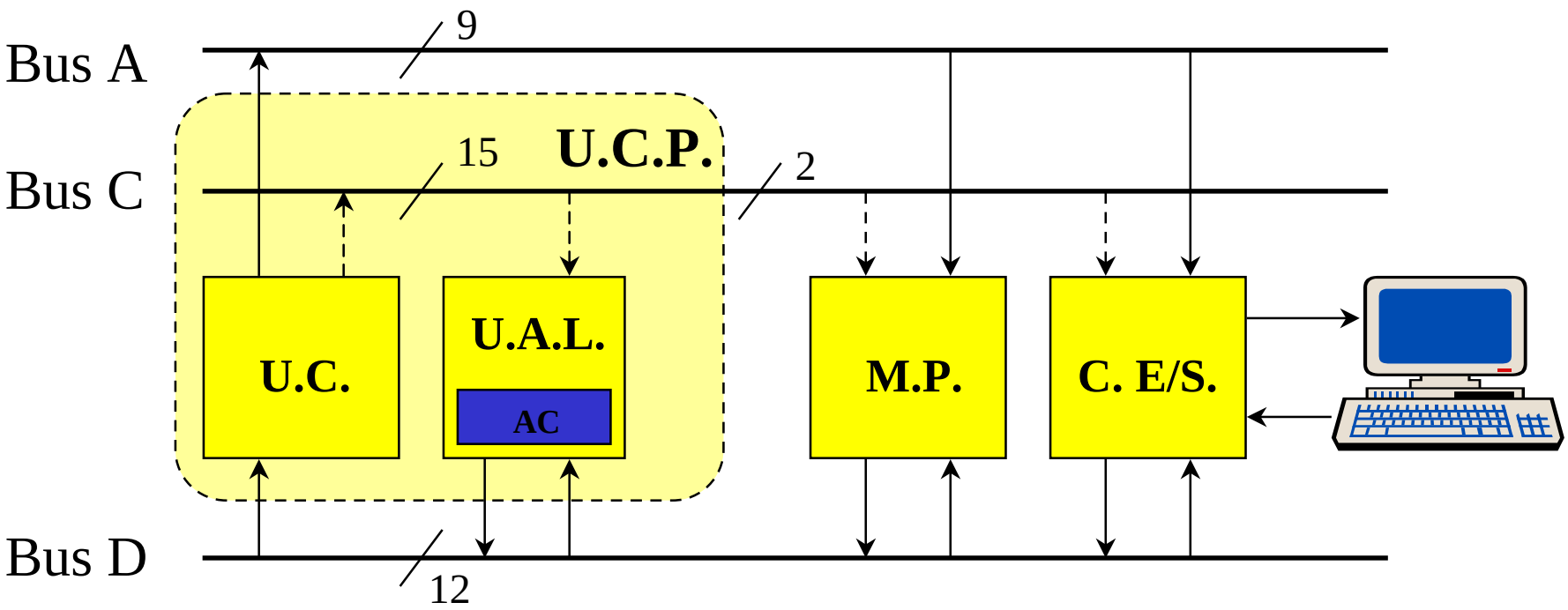
LENGUAJE IEEE 694

## 3.8 ALGORÍTMEZ. PROGRAMACIÓN.

MODELO ESTRUCTURAL
MODELO FUNCIONAL
REPERTORIO DE INSTRUCCIONES
MODOS DE DIRECCIONAMIENTO
INTERRUPCIONES
ENSAMBLADOR DE ALGORÍTMEZ
PROGRAMACIÓN
COMUNICACIONES CON LOS PERIFÉRICOS
CONSULTA Y SERVICIO DE INTERRUPCIONES
CONSULTA Y GESTIÓN DE PRIORIDADES POR HARDWARE

# 3.1 SÍMPLEZ

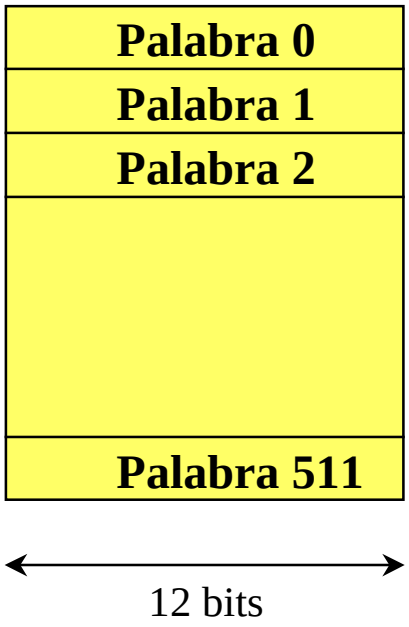
## MODELO ESTRUCTURAL:



# 3.1 SÍMPLEZ

## MODELO ESTRUCTURAL:

MEMORIA PRINCIPAL:



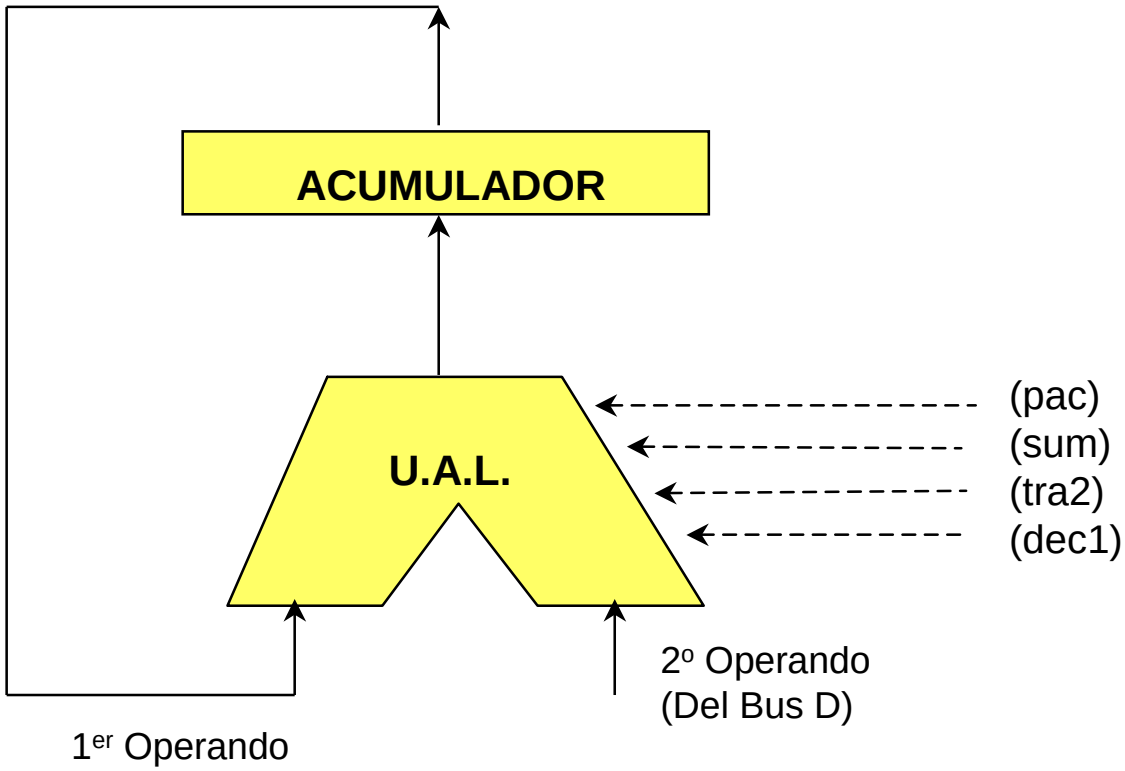

**Longitud de palabra:** Nº de bits que se escriben o leen en una operación.

**Capacidad:** Nº máximo de palabras que caben.

# 3.1 SÍMPLEZ

## MODELO ESTRUCTURAL:

### UNIDAD ARITMÉTICA Y LÓGICA:



# 3.1 SÍMPLEZ

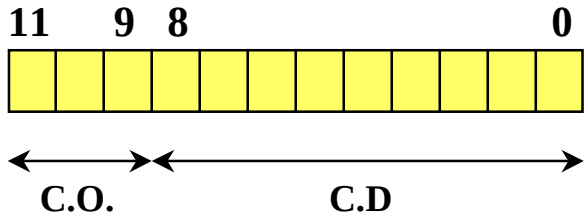
## MODELO FUNCIONAL:

- **REPRESENACIÓN DE LA INFORMACIÓN**

NÚMEROS (0-4095)

CARACTERES (ASCII)

INSTRUCCIONES



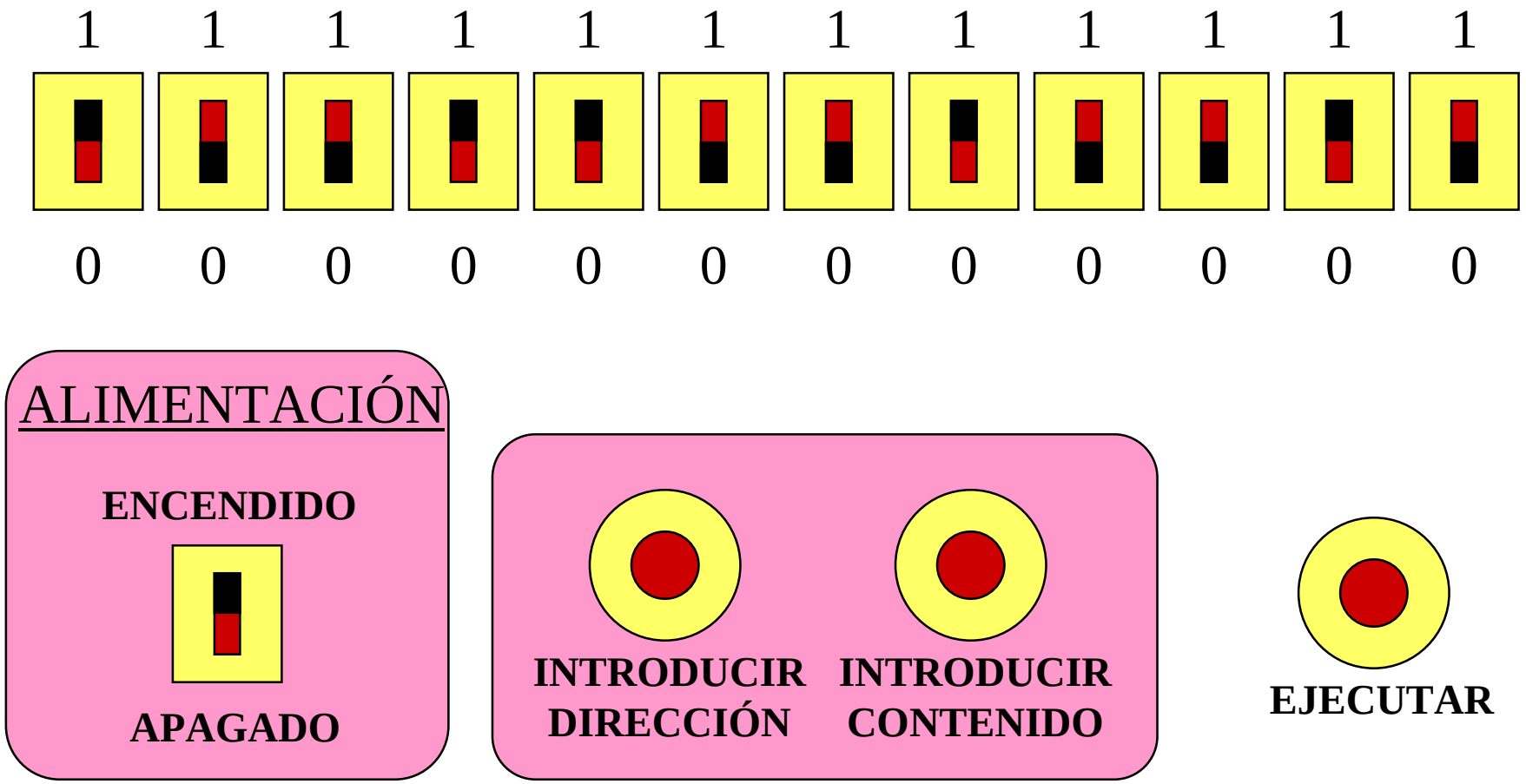
# 3.1 SÍMPLEZ

## MODELO FUNCIONAL:

- **REPERTORIO DE INSTRUCCIONES**

| C.O. (Bin) | C.O. (Oct) | Nemónico |
|---|---|---|
| 000 | 0 | ST |
| 001 | 1 | LD |
| 010 | 2 | ADD |
| 011 | 3 | BR |
| 100 | 4 | BZ |
| 101 | 5 | CLR |
| 110 | 6 | DEC |
| 111 | 7 | HALT |

# 3.1 SÍMPLEZ


1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0
ALIMENTACIÓN
ENCENDIDO
APAGADO
INTRODUCIR DIRECCIÓN
INTRODUCIR CONTENIDO
EJECUTAR

# 3.1 SÍMPLEZ

## PROBLEMA:

En la memoria de Símplez se tiene el siguiente programa:

| Dirección | Contenido(Oct) |
|---|---|
| 0 | 3003 |
| 1 | 0001 |
| 2 | 0000 |
| 3 | 5000 |
| 4 | 2001 |
| 5 | 0002 |
| 6 | 7000 |

Si la U.C. empieza ejecutando la instrucción almacenada en la posición 0, averigüe que hace el programa, y escriba los contenidos de las direcciones 0 a 6 de la M.P. después de la ejecución.

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## PROGRAMA 1: SUMA DE DOS NÚMEROS

Sumar los contenidos de las palabras de direcciones 10 y 11 y llevar el resultado a la posición 12.

| Dirección M.P. (Dec) | Contenido (Oct) | Nemónico |
|---|---|---|
| [0] | 1012 | LD /10 |
| [1] | 2013 | ADD /11 |
| [2] | 0014 | ST /12 |
| [3] | 7000 | HALT |

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## PROGRAMA 2: CONSTRUCCIÓN DE BUCLES

Calcular la suma de los 10 primeros términos de la sucesión de Fibonacci:

$$t_0 = 0; \qquad t_1 = 1; \qquad t_n = t_{n-1} + t_{n-2}; \qquad (n>1)$$

La suma se calcula de la siguiente forma:

$$S_0 = 0; \qquad S_n = t_n + S_{n-1}; \qquad (n>1)$$

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

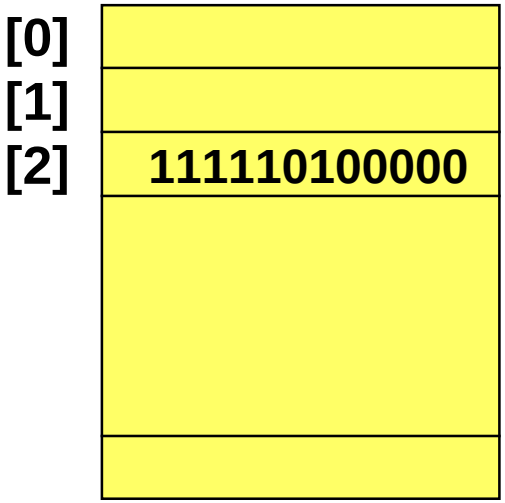
## PROGRAMA 2: CONSTRUCCIÓN DE BUCLES

| Dir. | Cont. | Nemónico | | Dir. | Cont. | Nemónico | |
|---|---|---|---|---|---|---|---|
| [0] | 5000 | CLR | | [13] | 0057 | ST | /47 |
| [1] | 0057 | ST | /47 | [14] | 1061 | LD | /49 |
| [2] | 1063 | LD | /51 | [15] | 0060 | ST | /48 |
| [3] | 0060 | ST | /48 | [16] | 1056 | LD | /46 |
| [4] | 0062 | ST | /50 | [17] | 6000 | DEC | |
| [5] | 1064 | LD | /52 | [18] | 4025 | BZ | /21 |
| [6] | 0056 | ST | /46 | [19] | 0056 | ST | /46 |
| [7] | 1057 | LD | /47 | [20] | 3007 | BR | /7 |
| [8] | 2060 | ADD | /48 | [21] | 7000 | HALT | |
| [9] | 0061 | ST | /49 | ----- | | | |
| [10] | 2062 | ADD | /50 | ----- | | | |
| [11] | 0062 | ST | /50 | [51] | 0001 | | |
| [12] | 1060 | LD | /48 | [52] | 1000 | | |

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## PROGRAMA 3: AUTOMODIFICACIÓN DE PROGRAMAS

* Interpretación de los contenidos de memoria:



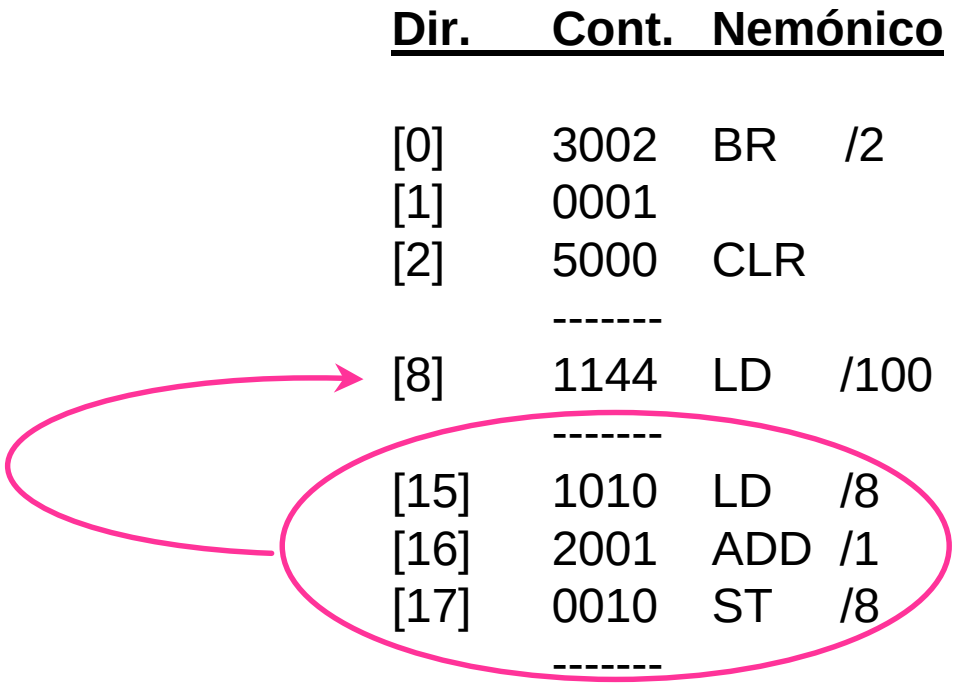

$7640_{O)} = 4000$

HALT

- LD /2
- BR /2

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## PROGRAMA 3: AUTOMODIFICACIÓN DE PROGRAMAS



| Dir. | Cont. | Nemónico | |
|---|---|---|---|
| [0] | 3002 | BR | /2 |
| [1] | 0001 | | |
| [2] | 5000 | CLR | |
| | ------- | | |
| [8] | 1144 | LD | /100 |
| | ------- | | |
| [15] | 1010 | LD | /8 |
| [16] | 2001 | ADD | /1 |
| [17] | 0010 | ST | /8 |
| | ------- | | |
| [25] | 3010 | BR | /8 |
| | ------- | | |
| | ------- | | |

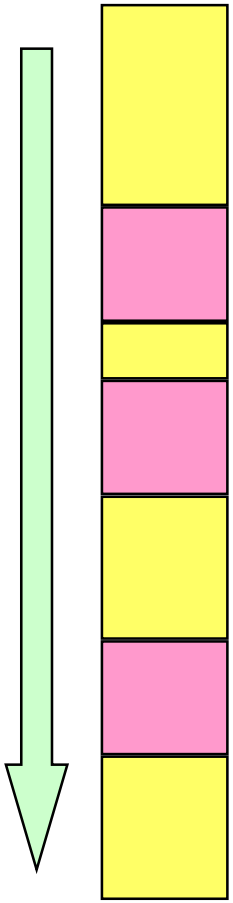
# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## PROGRAMAS PROPUESTOS:

- Sumar los números almacenados en las posiciones 50 - 149 dejando el resultado en la posición 150.
- Intercambiar los contenidos de memoria de las posiciones de memoria 100 - 149 y 200 - 249.

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

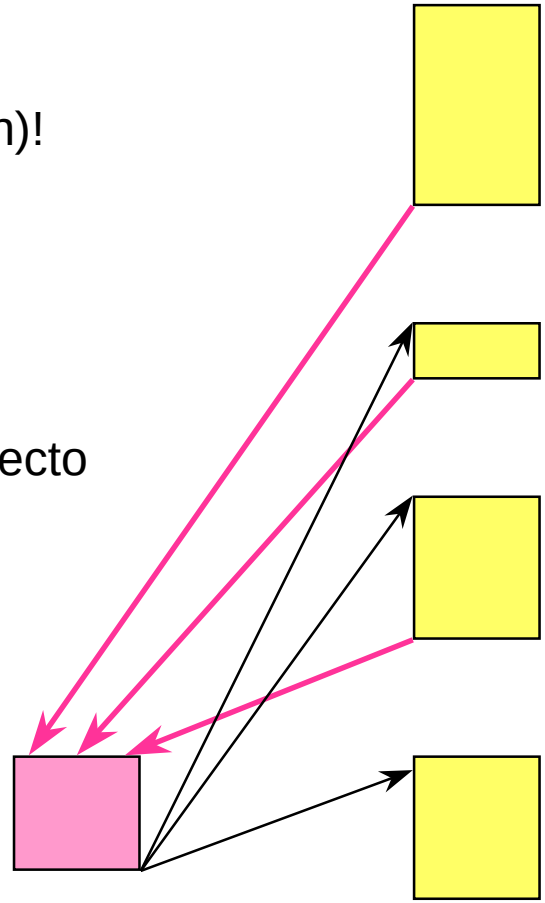
## INTRODUCCIÓN AL USO DE SUBPROGRAMAS



$$\binom{m}{n} = \frac{m!}{n!\,(m-n)!}$$

PROBLEMAS

- Volver al lugar correcto
- Pasar los datos



# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## INTRODUCCIÓN AL USO DE SUBPROGRAMAS

**Programa para restar**

| Dir. | Cont. | Nemónico | |
|---|---|---|---|
| [200] | 3313 | BR | /203 |
| [201] | 0000 | | |
| [202] | 0017 | | |
| [203] | 6000 | DEC | |
| [204] | 0311 | ST | /201 |
| [205] | 1312 | LD | /202 |
| [206] | 6000 | DEC | |
| [207] | 4323 | BZ | /211 |
| [208] | 0312 | ST | /202 |
| [209] | 1311 | LD | /201 |
| [210] | 3313 | BR | /203 |
| [211] | 1311 | LD | /201 |
| [212] | 7000 | HALT | |

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## INTRODUCCIÓN AL USO DE SUBPROGRAMAS

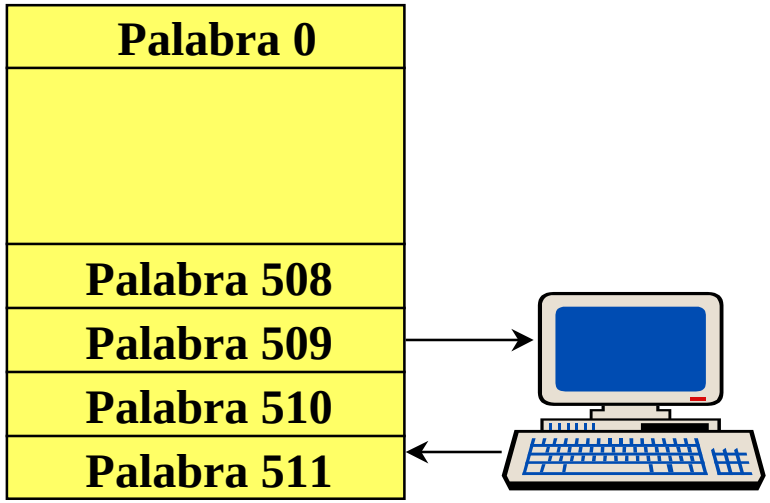
| Dir. | Cont. | Nemónico | |
|---|---|---|---|
| [4] | 3000 | BR | /0 |
| [5] | 0021 | 17 | |
| [6] | 0033 | 27 | |
| [10] | 1063 | LD | /51 |
| [11] | 0312 | ST | /202 |
| [12] | 1004 | LD | /4 |
| [13] | 2005 | ADD | /5 |
| [14] | 0324 | ST | /212 |
| [15] | 1062 | LD | /50 |
| [16] | 3310 | BR | /200 |
| [17] | 0064 | ST | /52 |

| Dir. | Cont. | Nemónico | |
|---|---|---|---|
| [200] | 3313 | BR | /203 |
| [201] | 0000 | | |
| [202] | 0000 | | |
| [203] | 6000 | DEC | |
| [204] | 0311 | ST | /201 |
| [205] | 1312 | LD | /202 |
| [206] | 6000 | DEC | |
| [207] | 4323 | BZ | /211 |
| [208] | 0312 | ST | /202 |
| [209] | 1311 | LD | /201 |
| [210] | 3313 | BR | /203 |
| [211] | 1311 | LD | /201 |
| [212] | 0000 | | |

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## INTRODUCCIÓN A LAS COMUNICACIONES CON EL EXTERIOR

MEMORIA PRINCIPAL:




```
[d]      LD   /508
[d+1]    BZ   /d
[d+2]    LD   /dato
[d+3]    ST   /509

[d]      LD   /510
[d+1]    BZ   /d
[d+2]    LD   /511
[d+3]    ST   /dato
```

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## INTRODUCCIÓN A LAS COMUNICACIONES CON EL EXTERIOR

### Crítica a la espera activa:

El tiempo que tarda la UCP en ejecutar cada instrucción viene determinado por MP y no por la UCP, ya que la MP es más lenta.

**Ciclo de memoria:** tiempo que transcurre desde que se inicia un acceso a la MP hasta que puede iniciarse otro. En Símplez será de 200ns.

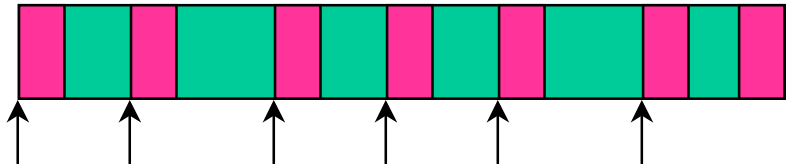
Tasa de transferencia de caracteres: 30 caractéres por segundo.

# 3.2 PROGRAMACIÓN DE SÍMPLEZ

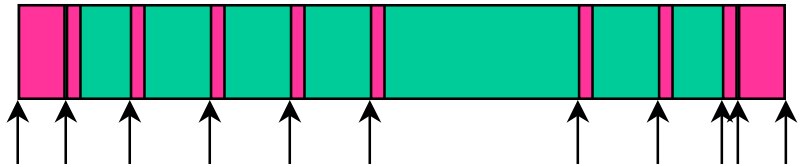
## INTRODUCCIÓN A LAS COMUNICACIONES CON EL EXTERIOR

- Espera activa

- Interrupciones



- Acceso Directo a Memoria



# 3.2 PROGRAMACIÓN DE SÍMPLEZ

**PROBLEMA:**

**Sea un ordenador de palabras de 9 bits. En el formato de instrucción, los tres primeros representan el código de operación, y los restantes la dirección.**

**El repertorio de instrucciones es el siguiente:**

**000: Parar**
**001: Poner a cero el acumulador**
**010: Almacenar el acumulador**
**011: Cargar el acumulador**
**100: Sumar al acumulador**
**101: Salto incondicional**
**110: Escribir por la unidad de salida**
**111: Leer un dato por la unidad de entrada**

**¿ Qué hace el siguiente programa ?**

| | | | |
|---|---|---|---|
| **0:** | **111001000** | **4:** | **100001000** |
| **1:** | **111001001** | **5:** | **010001010** |
| **2:** | **011001001** | **6:** | **110001010** |
| **3:** | **100001000** | **7:** | **000000000** |

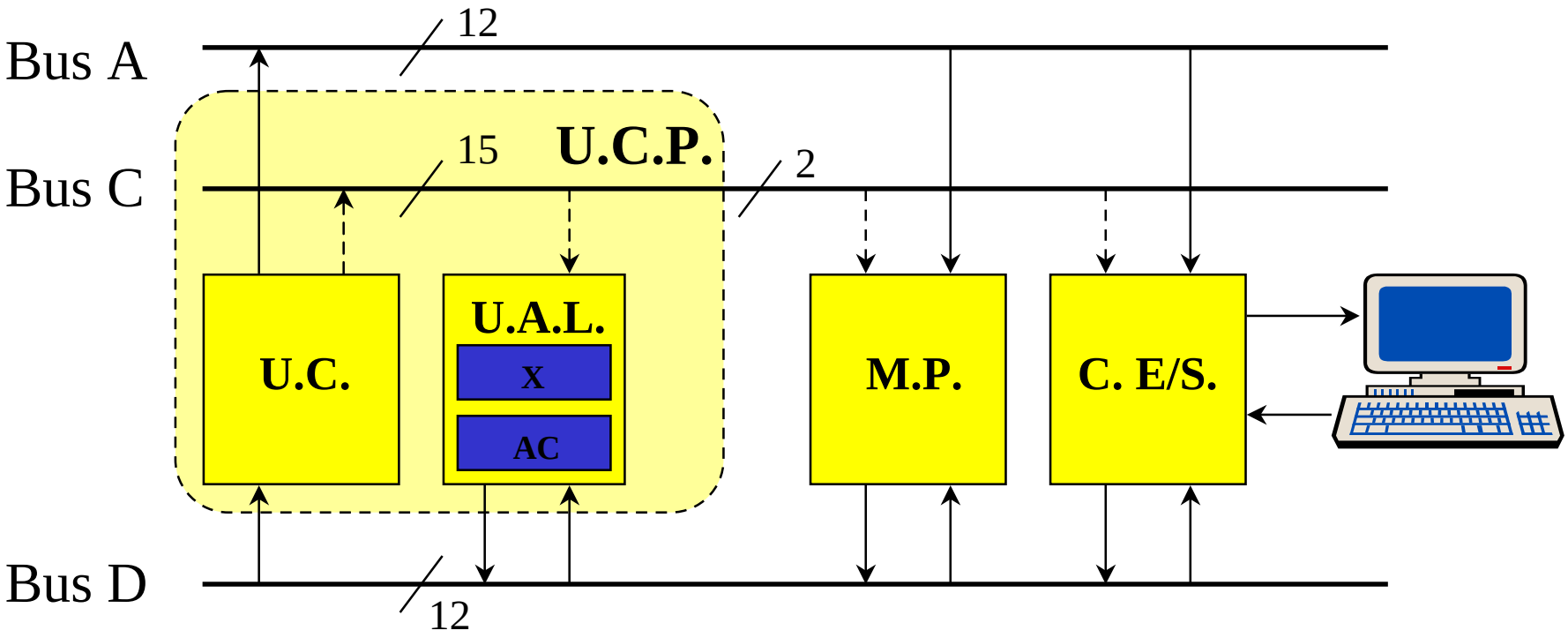
# 3.2 PROGRAMACIÓN DE SÍMPLEZ

## PROBLEMA PROPUESTO:

Escribir un subprograma para escribir por la pantalla 80 caracteres. Se supondrá que el programa que lo llama ha colocado previamente esos 80 caracteres en otras tantas palabras de MP, las de direcciones comprendidas entre 10 y 89.

# 3.3 SÍMPLEZ+i4

## MODELO ESTRUCTURAL:



# 3.3 SÍMPLEZ+i4

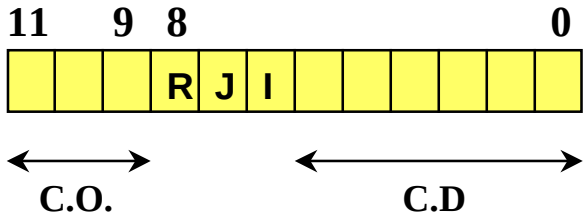
## MODELO FUNCIONAL:

- **REPRESENACIÓN DE LA INFORMACIÓN**

NÚMEROS (0-4095)

CARACTERES (ASCII)

INSTRUCCIONES




**R:** **Registro.**
**J:** **Indexado.**
**I:** **Indirecto.**

# 3.3 SÍMPLEZ+i4

## MODELO FUNCIONAL:

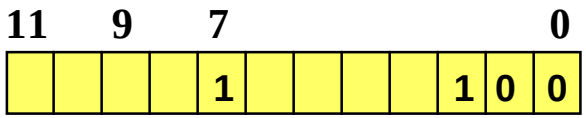
- **REPERTORIO DE INSTRUCCIONES**

| C.O. (Bin) | C.O. (Oct) | Nemónico |
|---|---|---|
| 000 | 0 | ST |
| 001 | 1 | LD |
| 010 | 2 | ADD |
| 011 | 3 | BR |
| 100 | 4 | BZ |
| 101 | 5 | CLR |
| 110 | 6 | DEC |
| 11100 | 7 | HALT |
| 11101 | 7 | EI |
| 11110 | 7 | DI |

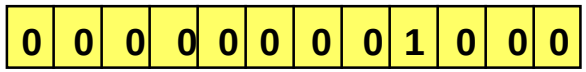
# 3.3 SÍMPLEZ+i4

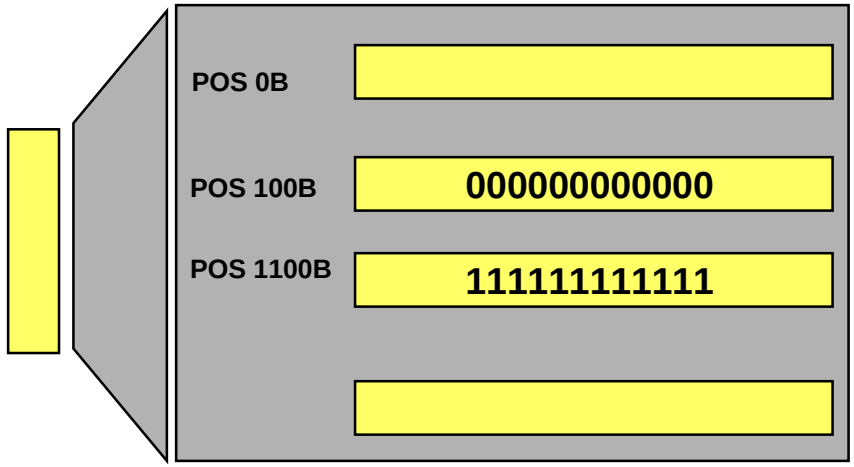
## MODOS DE DIRECCIONAMIENTO:

**INSTRUCCIÓN:**




**REGISTRO X:**




**DIRECCIONAMIENTO DIRECTO:**

DIR. EFEC. = 100B

OPERANDO = 000000000000B

**DIRECCIONAMIENTO INDEXADO:**

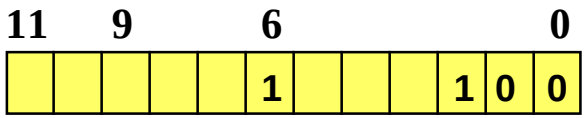
DIR. EFEC. = 1100B

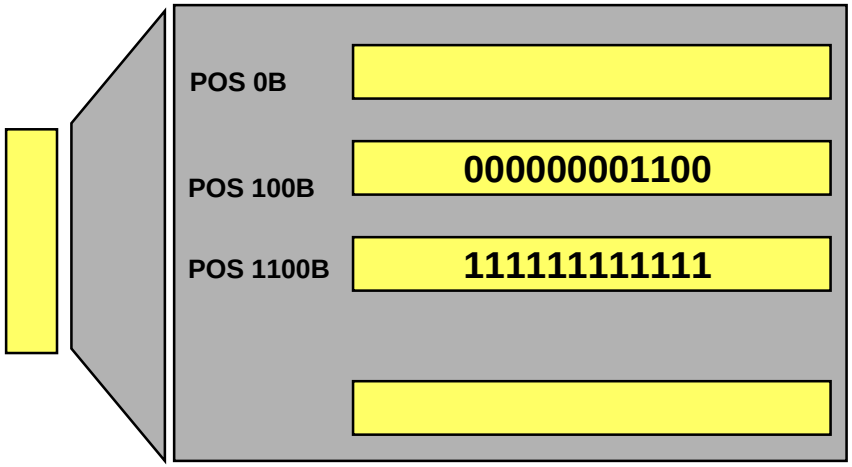
OPERANDO = 111111111111B

# 3.3 SÍMPLEZ+i4

## MODOS DE DIRECCIONAMIENTO:

**INSTRUCCIÓN:**




**DIRECCIONAMIENTO DIRECTO:**

DIR. EFEC. = 100B

OPERANDO = 000000001100B

**DIRECCIONAMIENTO INDIRECTO:**

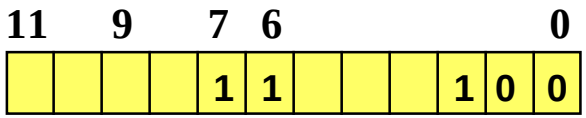
DIR. EFEC. = 1100B

OPERANDO = 111111111111B

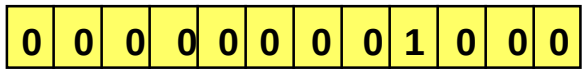
# 3.3 SÍMPLEZ+i4

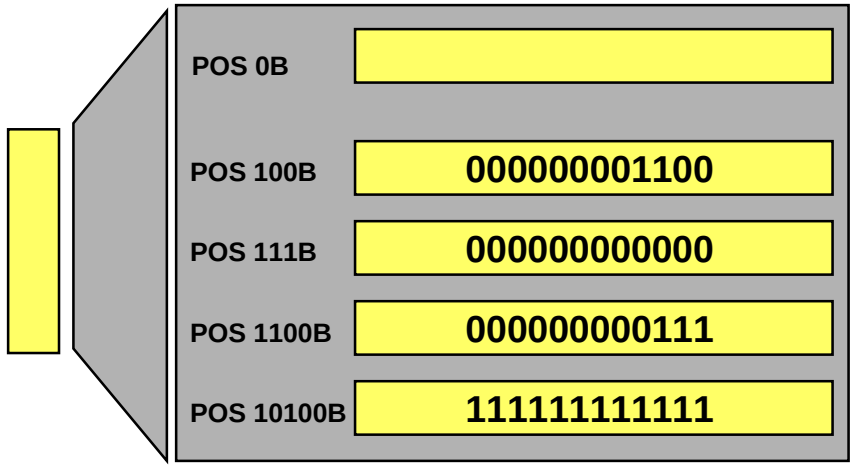
## MODOS DE DIRECCIONAMIENTO:

**INSTRUCCIÓN:**




**REGISTRO X:**




**DIRECCIONAMIENTO INDIR-INDEX:**

DIR. EFEC. = 10100B

OPERANDO = 111111111111B

**DIRECCIONAMIENTO INDEX-INDIR:**

DIR. EFEC. = 111B

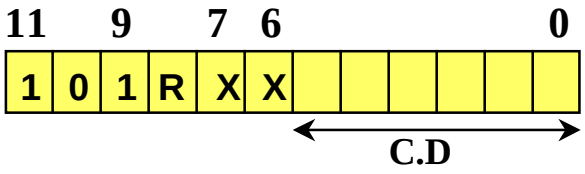
OPERANDO = 000000000000B

# 3.3 SÍMPLEZ+i4

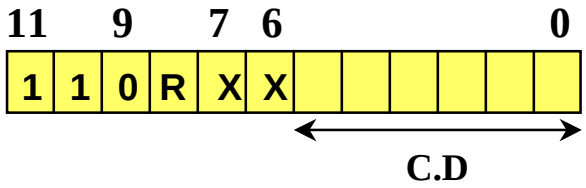
## MODOS DE DIRECCIONAMIENTO:

### DIRECCIONAMIENTO INMEDIATO

**INSTRUCCIÓN: CLR --> LD**




**INSTRUCCIÓN: DEC --> SUB**



## 3.3 SÍMPLEZ+i4

# MODOS DE DIRECCIONAMIENTO:

**RESUMEN**

| Bits 8,7,6 | Modo | Registro |
|---|---|---|
| 000 | Directo | Acumulador |
| 001 | Indirecto | “ |
| 010 | Indexado | “ |
| 011 | Postindexación | “ |
| 100 | Directo | Registro X |
| 101 | Indirecto | “ |
| 110 | Indexado | “ |
| 111 | Postindexación | “ |

**¿Inmediato?**

## 3.3 SÍMPLEZ+i4

# CONVENIOS SIMBÓLICOS:

| | BINARIO | OCTAL | SIMBÓLICO |
|---|---|---|---|
| **INFORMACIÓN SOBRE EL REGISTRO:** | 000000000010 | 0002 | ST .A, /2 |
| | 000100000010 | 0402 | ST .X, /2 |
| | 001100000111 | 1407 | LD .X, /7 |
| **DIRECCIONAMIENTO INDIRECTO:** | 000001000010 | 0102 | ST .A, [/2] |
| | 000101000010 | 0502 | ST .X, [/2] |
| **DIRECCIONAMIENTO INDEXADO:** | 000010000010 | 0202 | ST .A, /2 [.X] |
| | 010010000011 | 2203 | ADD .A, /3 [.X] |
| **DIRECCIONAMIENTO INDIRECTO E INDEXADO:** | 000011000010 | 0302 | ST .A, [/2] [.X] |
| | 010011000011 | 2303 | ADD .A, [/3] [.X] |
| **DIRECCIONAMIENTO INMEDIATO:** | 101000000010 | 5002 | LD .A, #2 |
| | 101100001010 | 5412 | LD .X, #10 |
| | 110000000001 | 6001 | SUB .A, #1 |
| | 110100111111 | 6477 | SUB .X, #63 |

## 3.3 SÍMPLEZ+i4

# USO DEL REGISTRO DE ÍNDICE COMO CONTADOR:

**Calcular la suma de los 10 primeros términos de la sucesión de Fibonacci:**

| Dir. | Cont. | Nemónico | | Dir. | Cont. | Nemónico | |
|---|---|---|---|---|---|---|---|
| [0] | 5000 | LD | .A, #0 | [10] | 0062 | ST | .A, /50 |
| [1] | 0057 | ST | .A, /47 | [11] | 1060 | LD | .A, /48 |
| [2] | 5001 | LD | .A, #1 | [12] | 0057 | ST | .A, /47 |
| [3] | 0060 | ST | .A, /48 | [13] | 1061 | LD | .A, /49 |
| [4] | 0062 | ST | .A, /50 | [14] | 0060 | ST | .A, /48 |
| [5] | 5410 | LD | .X, #8 | [15] | 6401 | SUB | .X, #1 |
| [6] | 1057 | LD | .A, /47 | [16] | 4022 | BZ | /18 |
| [7] | 2060 | ADD | .A, /48 | [17] | 3006 | BR | /6 |
| [8] | 0061 | ST | .A, /49 | [18] | 7000 | HALT | |
| [9] | 2062 | ADD | .A, /50 | | | | |

## 3.3 SÍMPLEZ+i4

# PUNTEROS E ÍNDICES:

**PROBLEMA:** Sumar los números almacenados en las posiciones 50 - 149 dejando el resultado en la posición 150.

**1.- Con direccionamiento indirecto.**

**2.- Con direccionamiento indirecto e indexado.**

# 3.3 SÍMPLEZ+i4

## PUNTEROS E ÍNDICES:

### 1.- Con direccionamiento indirecto.

| Dir. | Cont. | Nemónico |
|---|---|---|
| [0] | 3004 | BR /4 |
| [1] | 0144 | 100 |
| [2] | 0225 | 149 |
| [3] | 0226 | 150 |
| [4] | 1401 | LD .X, /1 |
| [5] | 5000 | LD .A, #0 |
| [6] | 2102 | ADD .A, [/2] |
| [7] | 0103 | ST .A, [/3] |
| [8] | 6401 | SUB .X, #1 |
| [9] | 4017 | BZ /15 |

| Dir. | Cont. | Nemónico |
|---|---|---|
| [10] | 1002 | LD .A, /2 |
| [11] | 6001 | SUB .A, #1 |
| [12] | 0002 | ST .A, /2 |
| [13] | 1103 | LD .A, [/3] |
| [14] | 3006 | BR /6 |
| [15] | 7000 | HALT |

# 3.3 SÍMPLEZ+i4

## PUNTEROS E ÍNDICES:

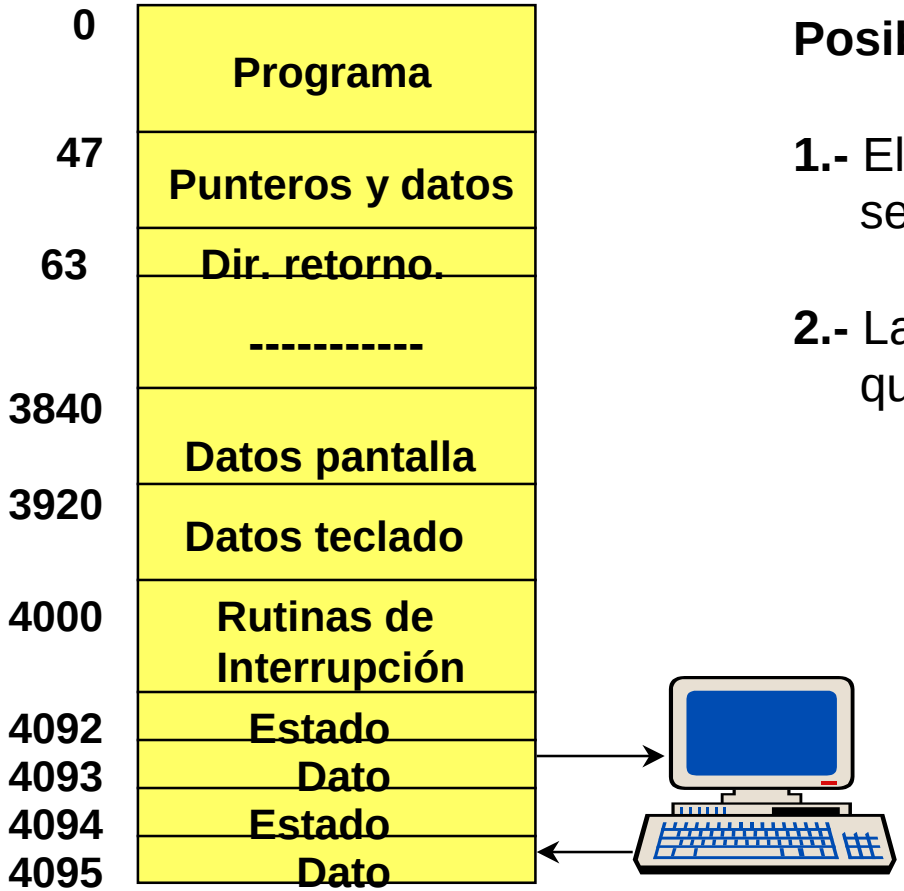
### 2.- Con direccionamiento indirecto e indexado.

| Dir. | Cont. | Nemónico | | Dir. | Cont. | Nemónico | |
|---|---|---|---|---|---|---|---|
| [0] | 3004 | BR | /4 | [10] | 2102 | ADD | .A, [/2] |
| [1] | 0143 | 99 | | [11] | 0103 | ST | .A, [/3] |
| [2] | 0062 | 50 | | [12] | 7000 | HALT | |
| [3] | 0226 | 150 | | | | | |
| [4] | 1401 | LD | .X, /1 | | | | |
| [5] | 5000 | LD | .A, #0 | | | | |
| [6] | 2302 | ADD | .A, [/2][.X] | | | | |
| [7] | 6401 | SUB | .X, #1 | | | | |
| [8] | 4012 | BZ | /10 | | | | |
| [9] | 3006 | BR | /6 | | | | |

## 3.3 SÍMPLEZ+i4

# INTERRUPCIONES:

MEMORIA PRINCIPAL:




**Posibles causas de interrupción:**

**1.-** El teclado interrumpirá cada vez que se pulse una tecla.

**2.-** La pantalla interrumpirá siempre que que acabe de escribir un carácter.

# 3.3 SÍMPLEZ+i4

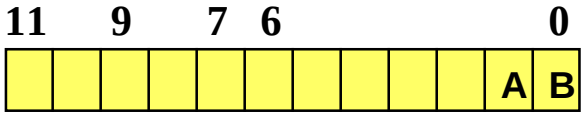
## INTERRUPCIONES:

**Inhibición y permiso de interrupciones:**

**1.-** Instrucciones **EI** y **DI**.

**2.-** Individualmente a cada periférico.

**ESTADO:**




**B: Preparado o no.**
**A: Interrupciones permitidas o no.**

**Iniciar pantalla: A=1; B=1;**
**Iniciar teclado: A=1; B=0;**

## 3.3 SÍMPLEZ+i4

# INTERRUPCIONES:

**Proceso de Interrupción:**

**1.-** Finalizar la instrucción en curso.
Guardar la dirección de retorno.
Guardar contenidos de los registros AC y X.

**2.-** Identificar causa de interrupción y bifurcar a la dirección correspondiente a esa causa.

**3.-** Finalizada Rutina de Servicio, se restauran los contenidos de los registros AC y X y se bifurca a la dirección de retorno.

**Cambio de contexto**
(Inhibición de interrupciones)

## 3.3 SÍMPLEZ+i4

# INTERRUPCIONES:

**DIFERENCIAS CON SUBPROGRAMAS:**

- La bifurcación a subprograma es consecuencia de la ejecución de instrucciones escritas por el programador.
- La UCP debe, automáticamente, al cambiar el contexto, salvaguardar la dirección de retorno. En Símplez+i4 será en la posición de memoria 63.
- La UCP bifurcará a una dirección a partir de la cual se deberá tener cargado un programa que guarde los contenidos de los registros AC y R, investigue la causa de interrupción y bifurque a la RS que corresponda.

## 3.3 SÍMPLEZ+i4

# INTERRUPCIONES. GENERALIZACIÓN.

**CAUSAS DE INTERRUPCIÓN:**

**Externas:** Originadas por causas ajenas a la UCP.

Periférico que avisa de que está dispuesto para una transferencia.
Fallo de la tensión de alimentación.
…

**Internas, o excepciones:** Se producen como consecuencia de acontecimientos dentro de la UCP.

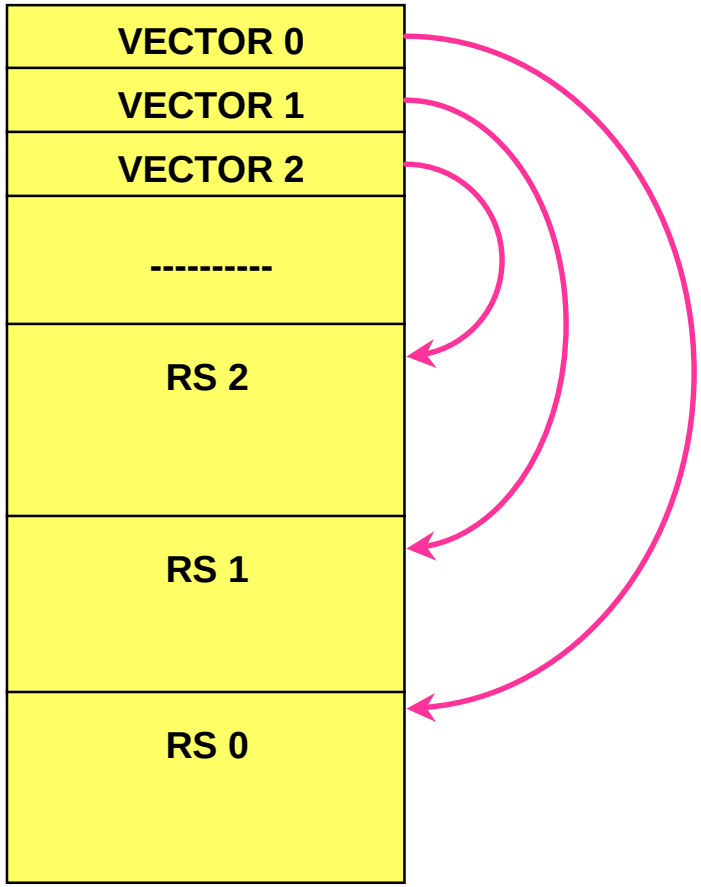
Desbordamiento en la UAL.
Código de operación inexistente.
Interrupción de programa.
…

**CAMBIOS DE CONTEXTO:** REGISTRO DE ESTADO.

## 3.3 SÍMPLEZ+i4

# INTERRUPCIONES. GENERALIZACIÓN.

### VECTORES DE INTERRUPCIÓN



## 3.3 SÍMPLEZ+i4

# INTERRUPCIONES. GENERALIZACIÓN.

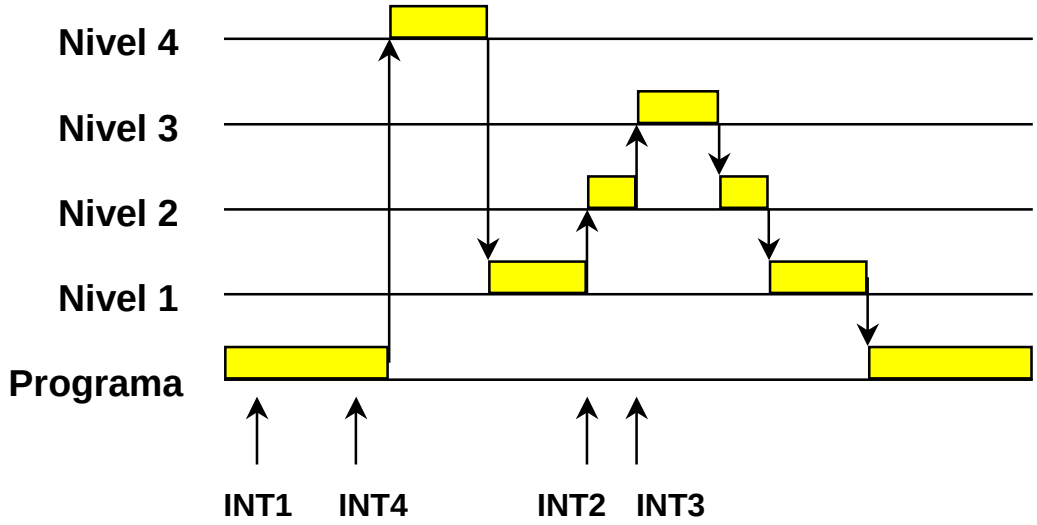
**CONSULTA DE INTERRUPCIONES:**

- CONSULTA POR SOFTWARE.
- INTERRUPCIONES VECTORIZADAS.

## 3.3 SÍMPLEZ+i4

# INTERRUPCIONES. GENERALIZACIÓN.

### ANIDAMIENTO Y GESTIÓN DE PRIORIDADES:



# 3.3 SÍMPLEZ+i4

## PROBLEMAS:

**1.- Se supone que el efecto de la instrucción “EI” es “retardado”, es decir, que si hay una interrupción pendiente no se atiende inmediatamente después de ella, sino de la que sigue. ¿Porqué?**

**2.- Escriba una programa que permita poner a cero una zona de memoria de 64 palabras que empieza por la dirección 100.**

# 3.4 DIRECCIONAMIENTO

## INTRODUCCIÓN:

**OBJETIVOS:**

Facilitar la tarea de programación.

Conseguir programas con menos instrucciones y por lo tanto más eficaces.

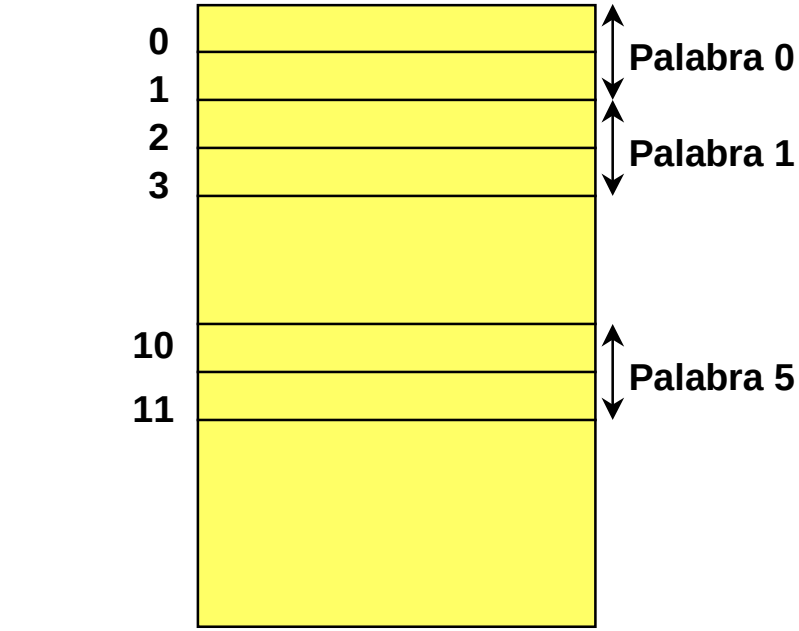
Reducir el espacio ocupado por las instrucciones en la MP, al permitir que el campo CD sea menor de lo que haría falta para direccionar toda la MP.

Facilitar la protección de unos programas frente a otros en un entorno de multiprogramación.

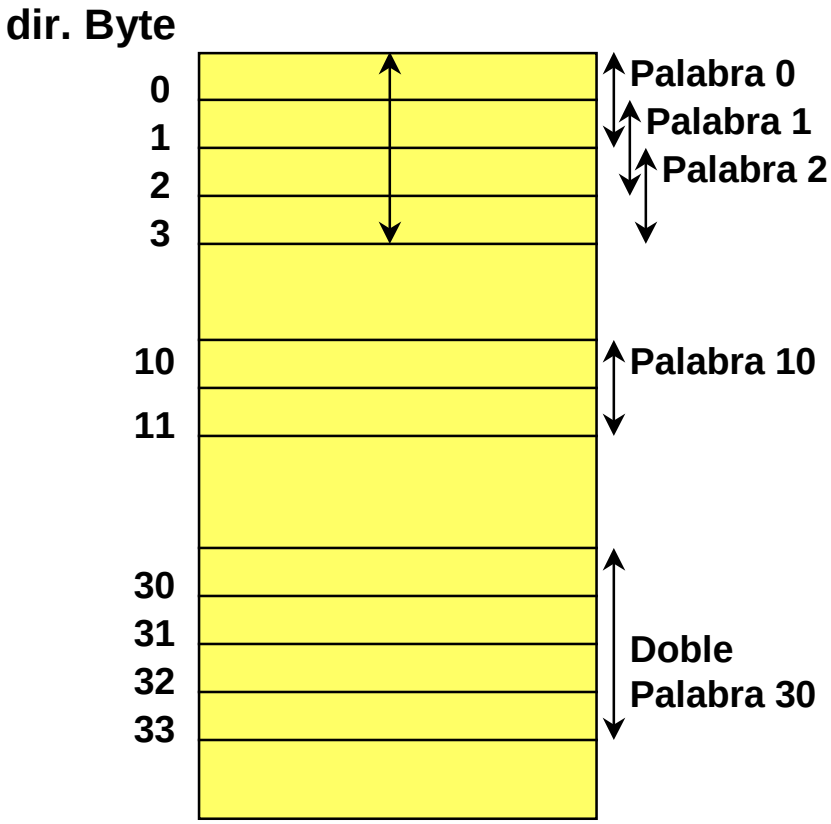
# 3.4 DIRECCIONAMIENTO

## INTRODUCCIÓN:

**DIRECCIONAMIENTO A BYTE Y A PALABRA:**



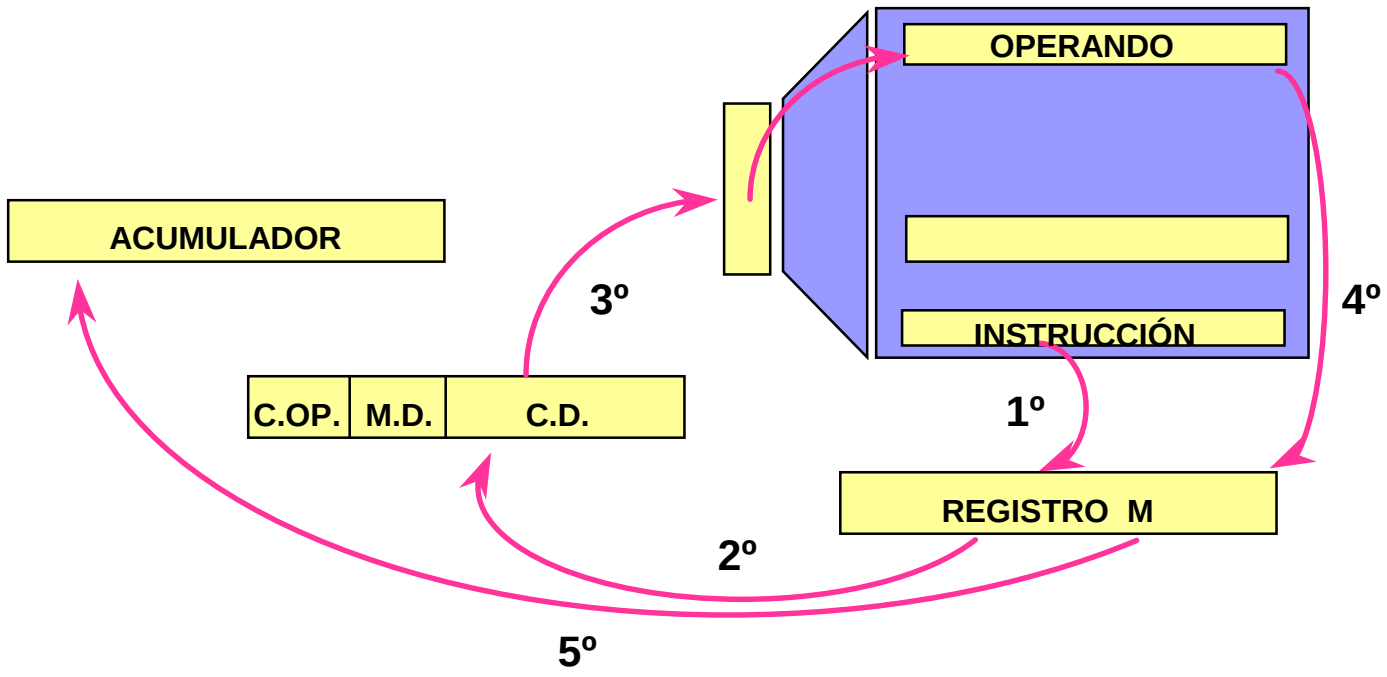

**Palabras con dirección siempre par**




**Palabras y dobles palabras con direcciónes arbitrarias**

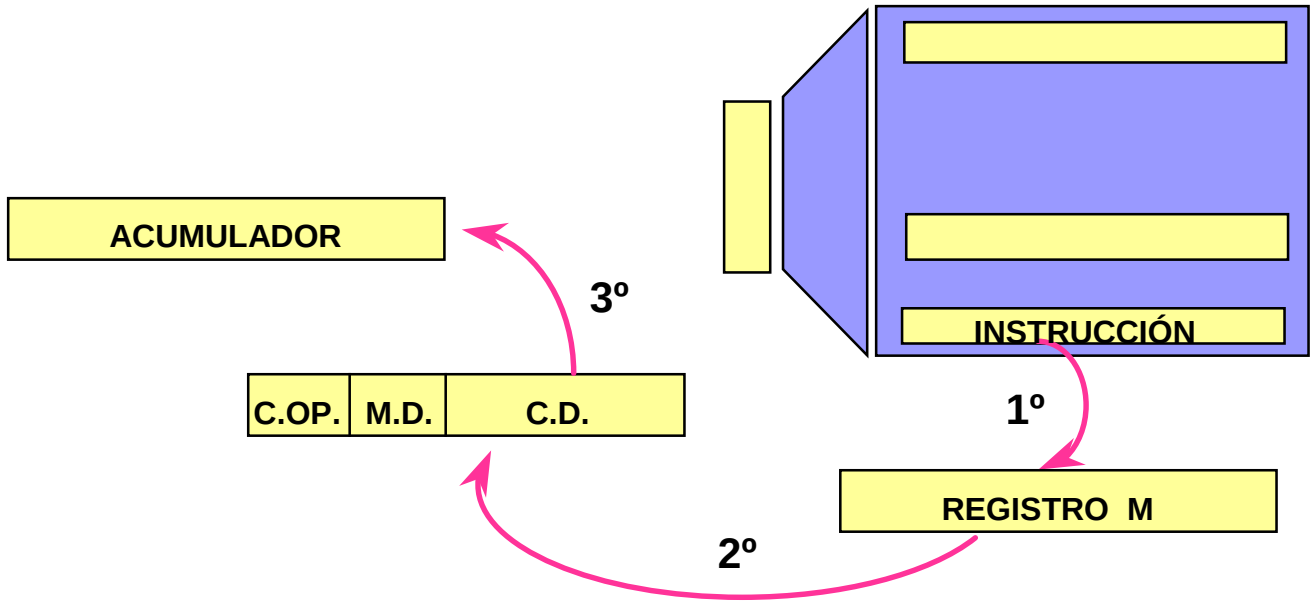
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO DIRECTO:



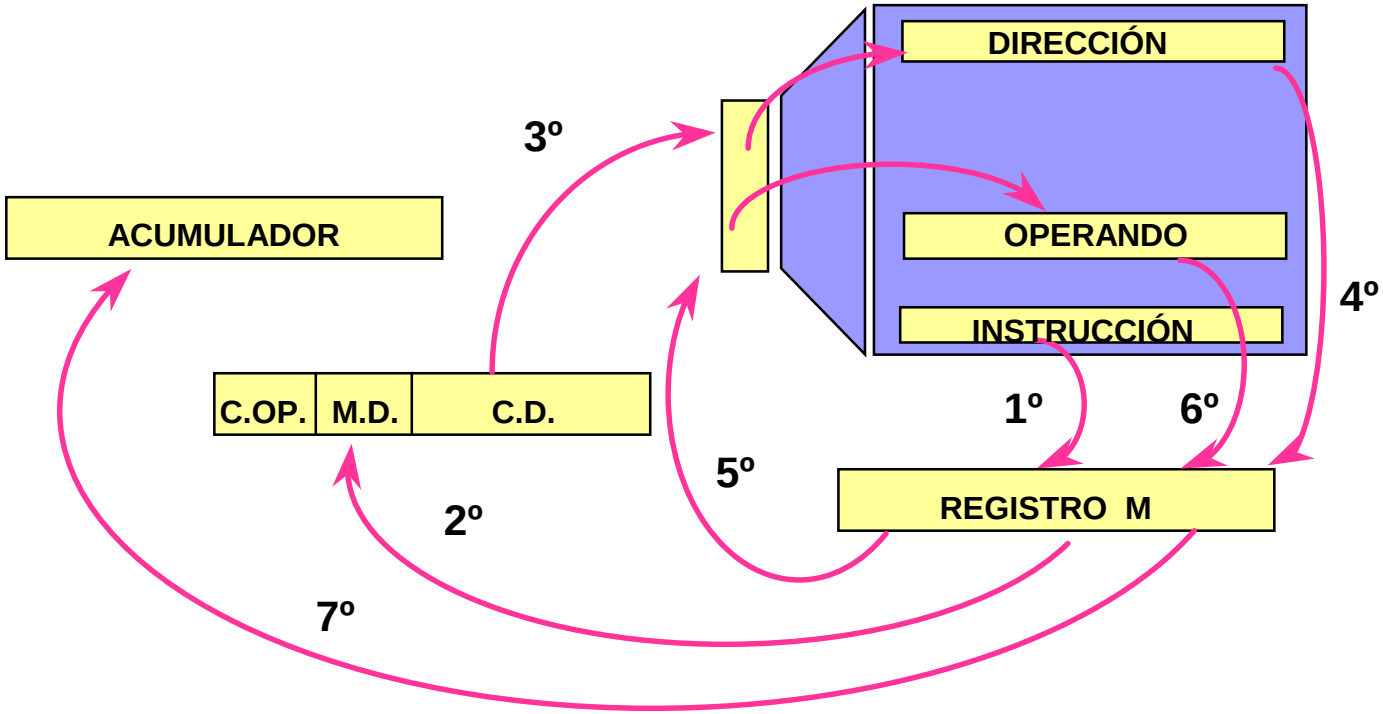
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO INMEDIATO:



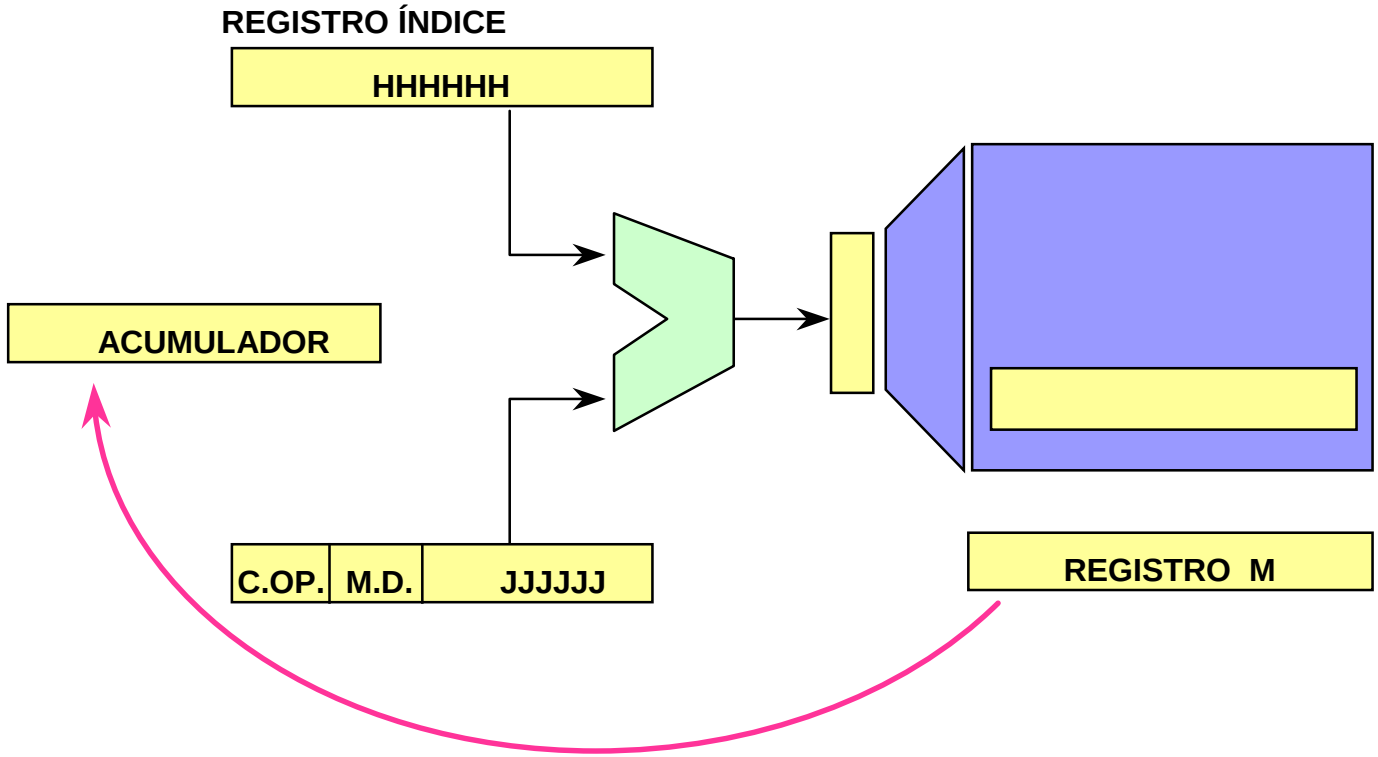
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO INDIRECTO:



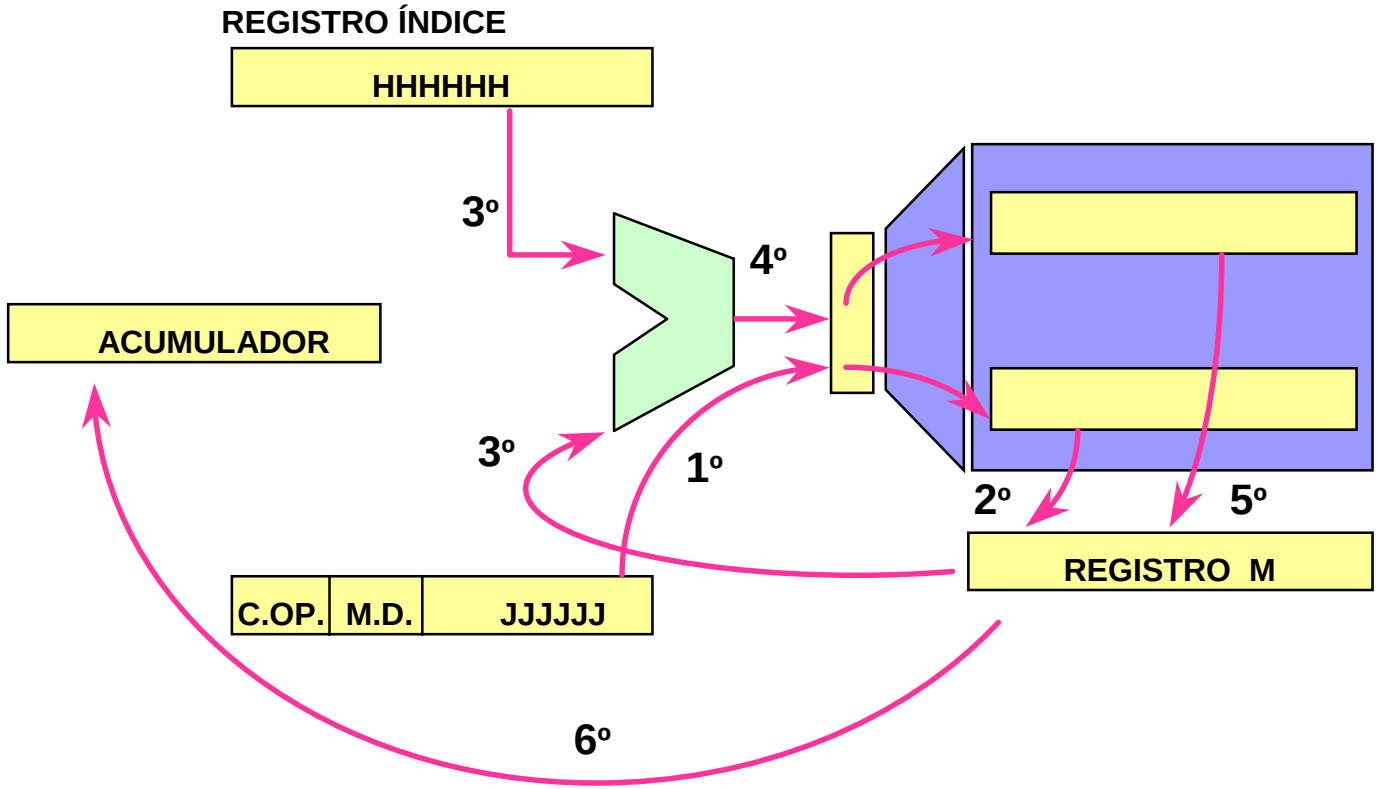
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO INDEXADO:



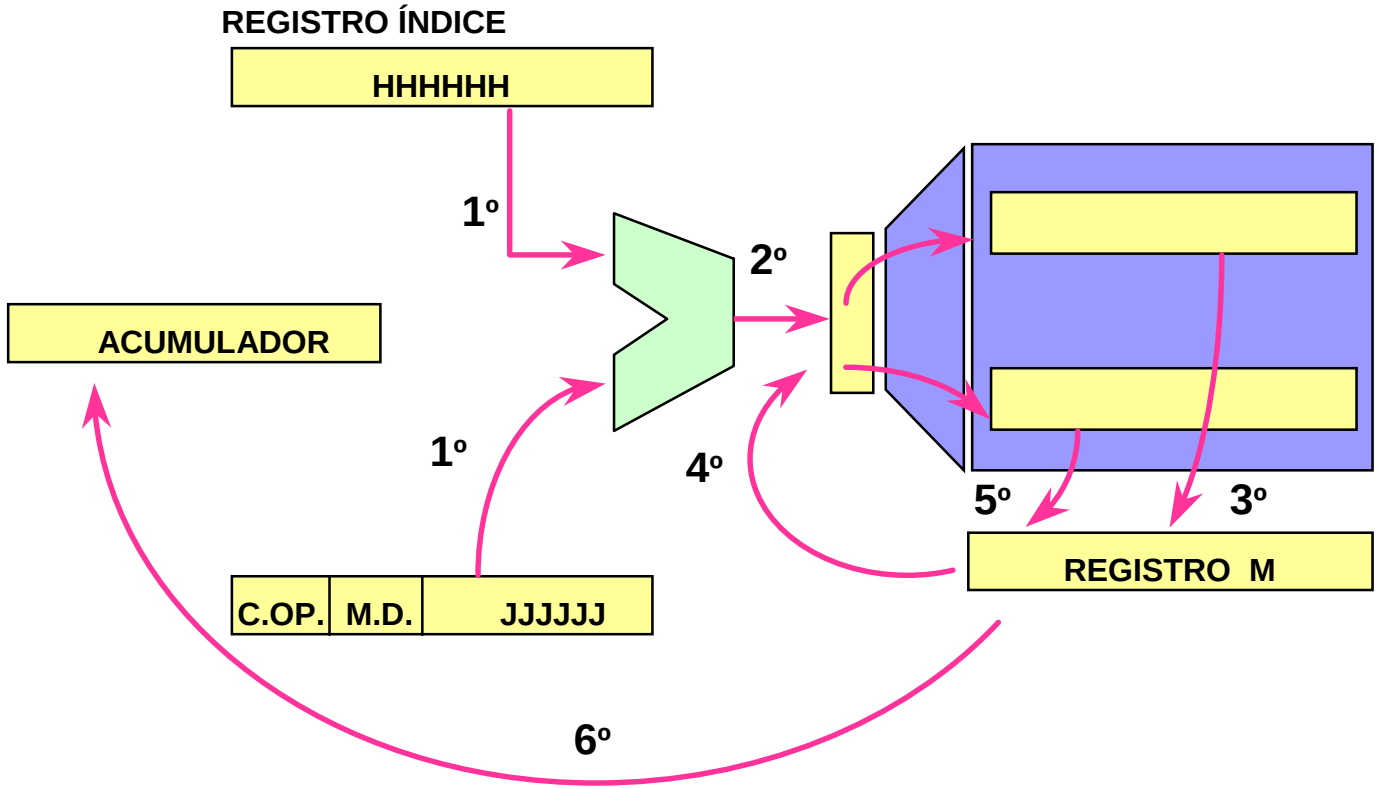
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO POSTINDEXACIÓN:



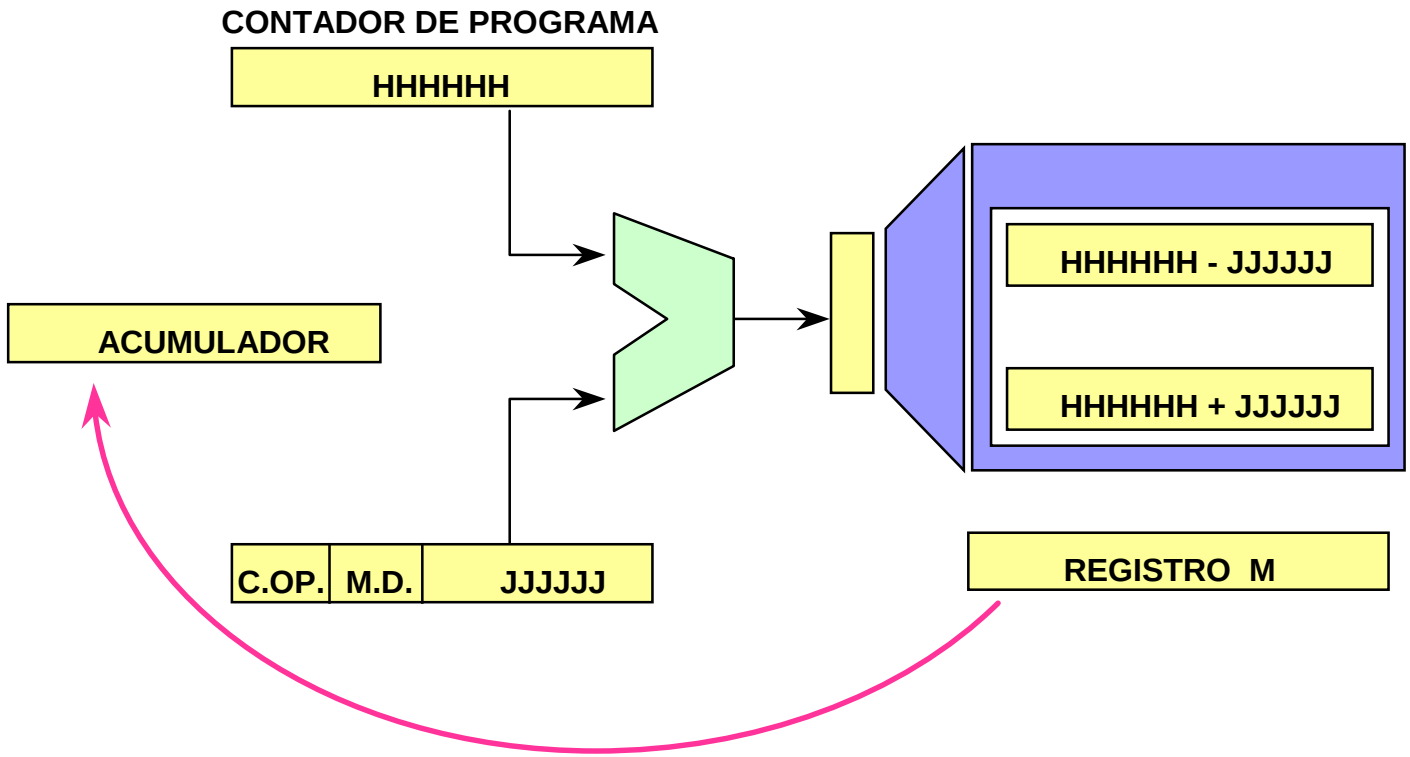
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO PREINDEXACIÓN:



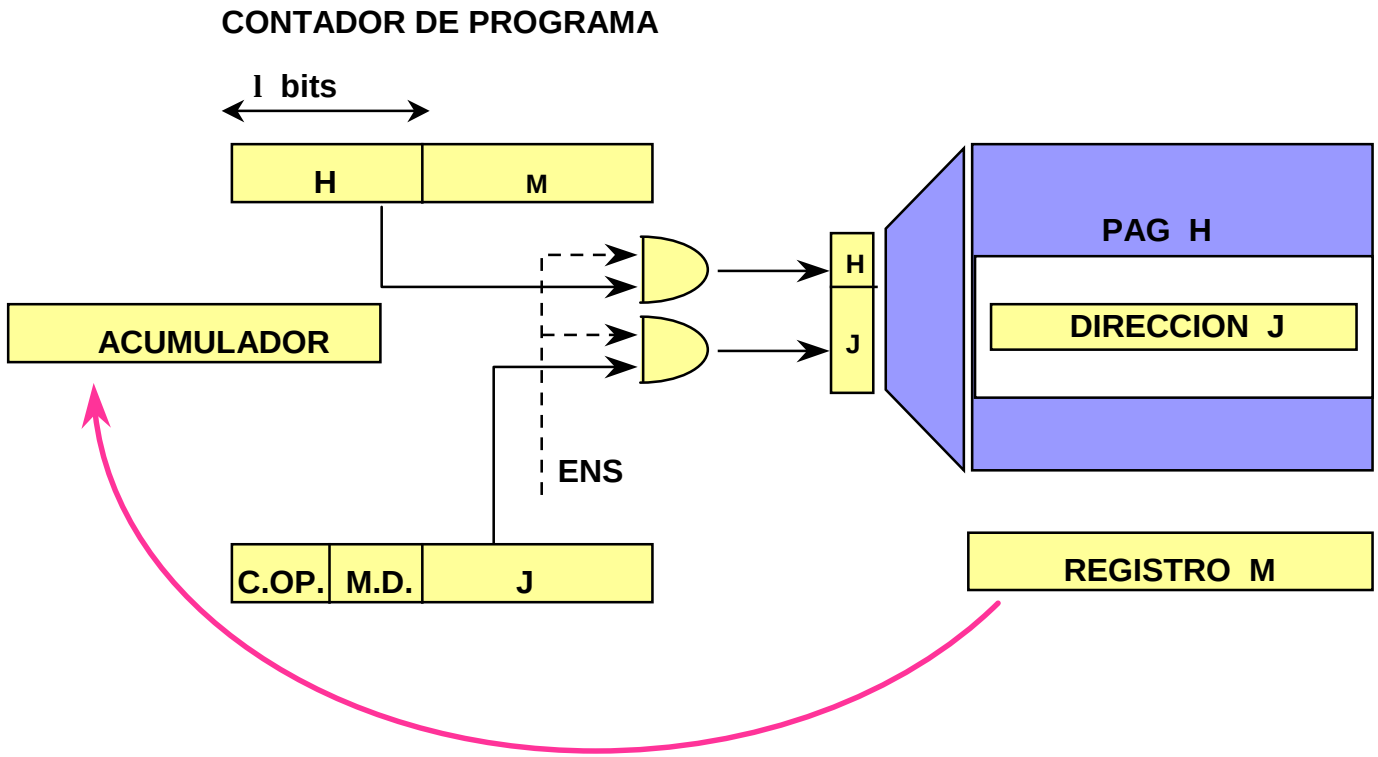
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO RELATIVO A PROGRAMA:



# 3.4 DIRECCIONAMIENTO

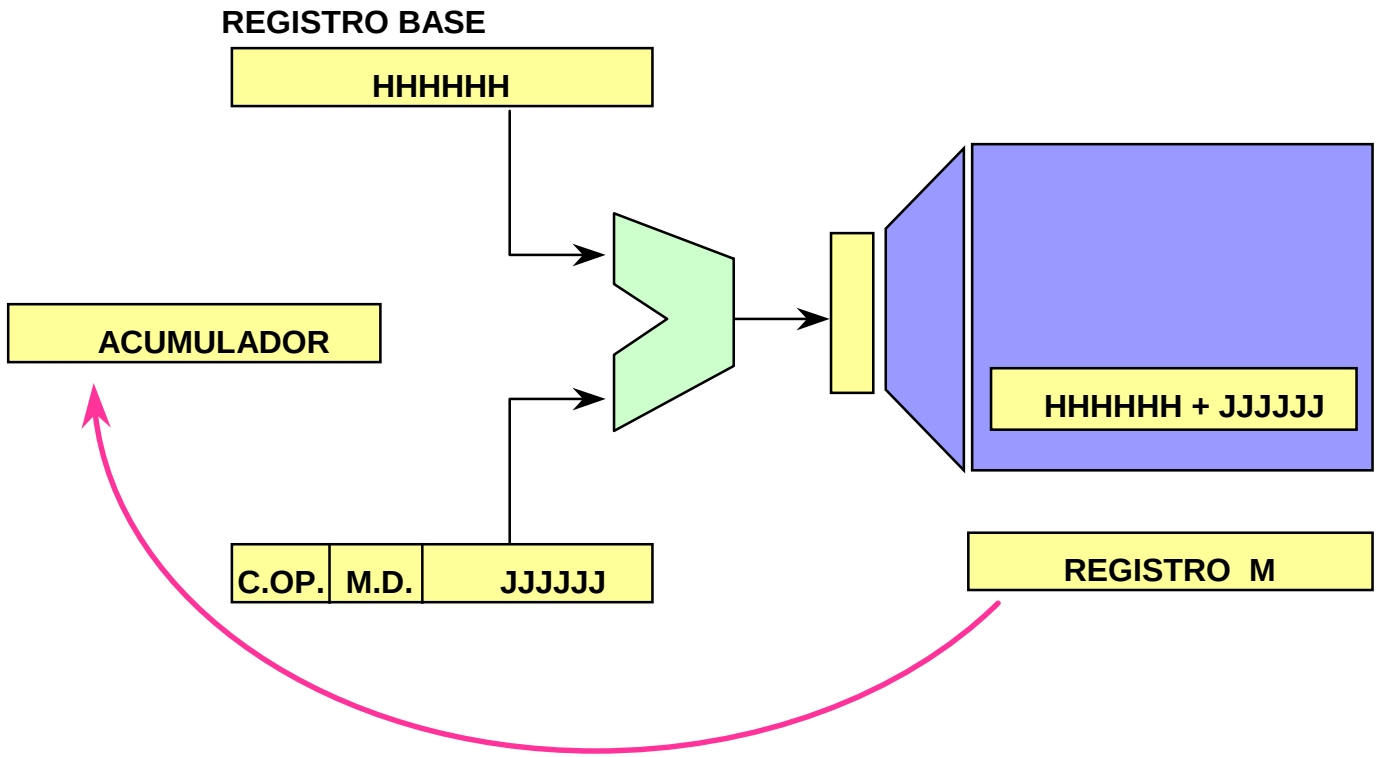
## DIRECCIONAMIENTO RELATIVO A PÁGINA:




**Vecindad entre instrucciones y datos**

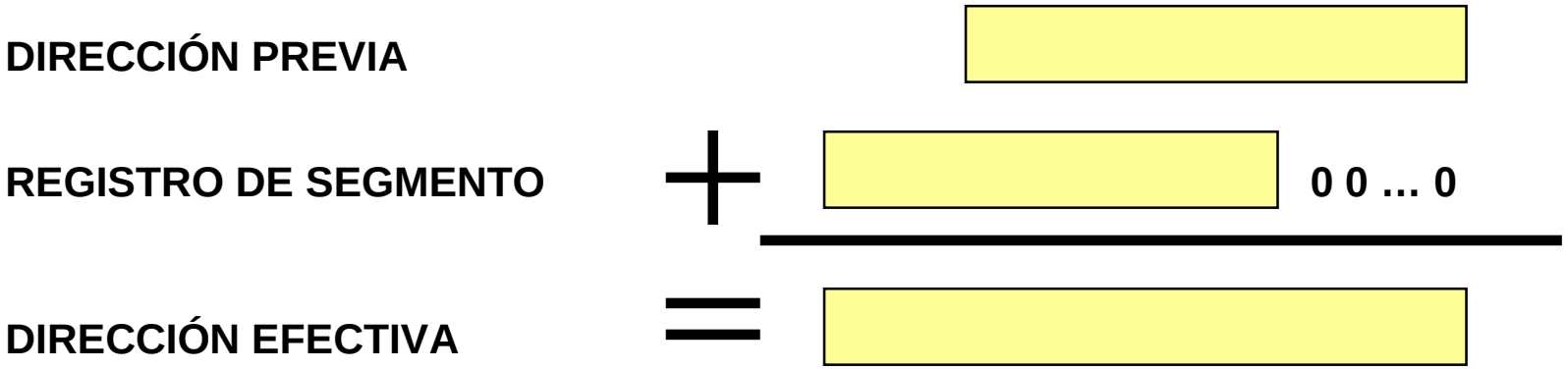
# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO RELATIVO A BASE:



# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO RELATIVO A SEGMENTO:

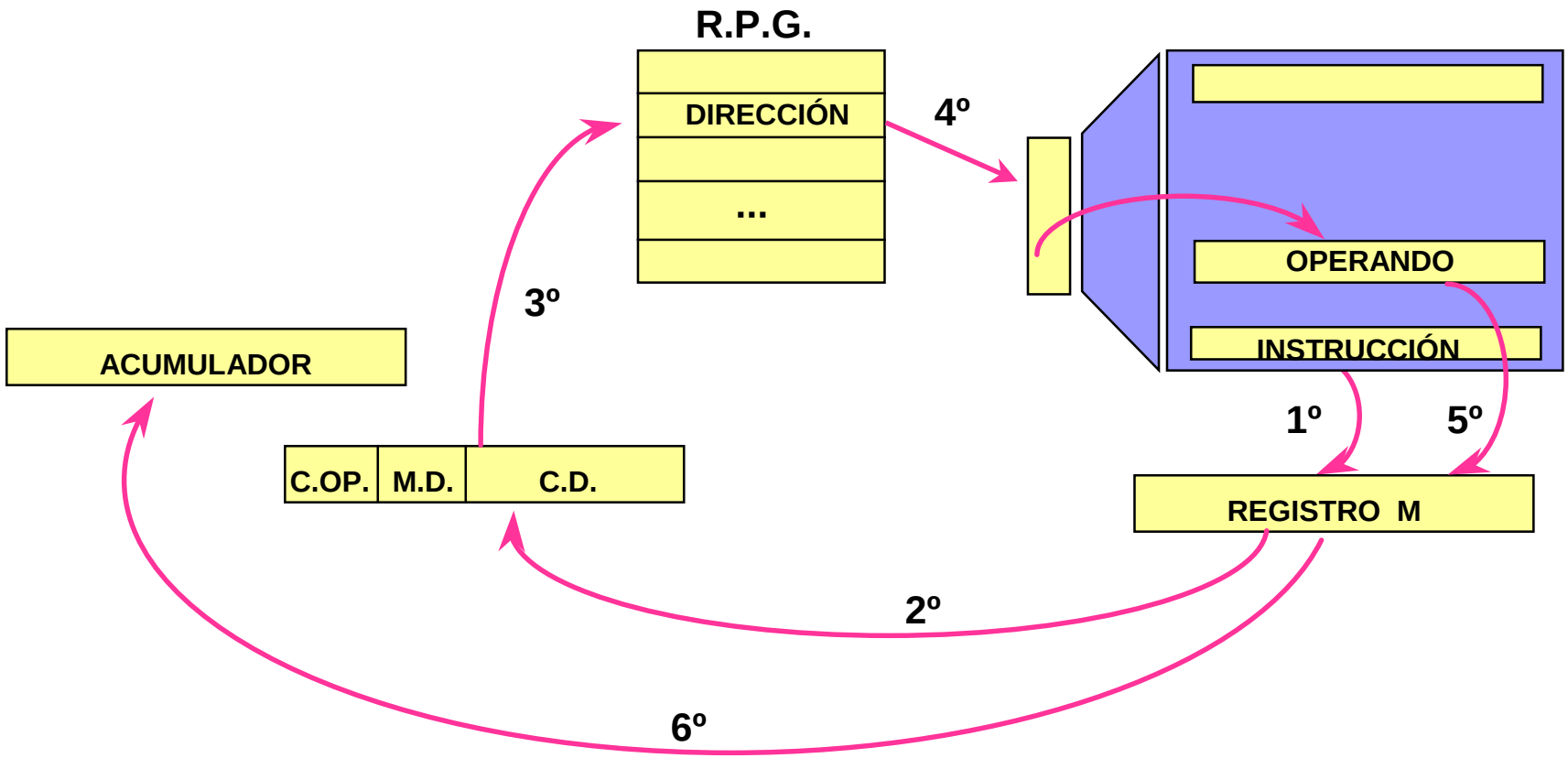



**EJEMPLO:**

CD: Longitud de 16 bits. (64k palabras)
RS: Longitud de 16 bits.
Desplazamiento: 4 bits.
DE: Longitud de 20 bits. (1M palabras)

# 3.4 DIRECCIONAMIENTO

## DIRECCIONAMIENTO BASADO EN R.P.G.:



# 3.4 DIRECCIONAMIENTO

## MODOS AUTOINCREMENTO Y AUTODECREMENTO:

**AUTOPREINCREMENTO.**

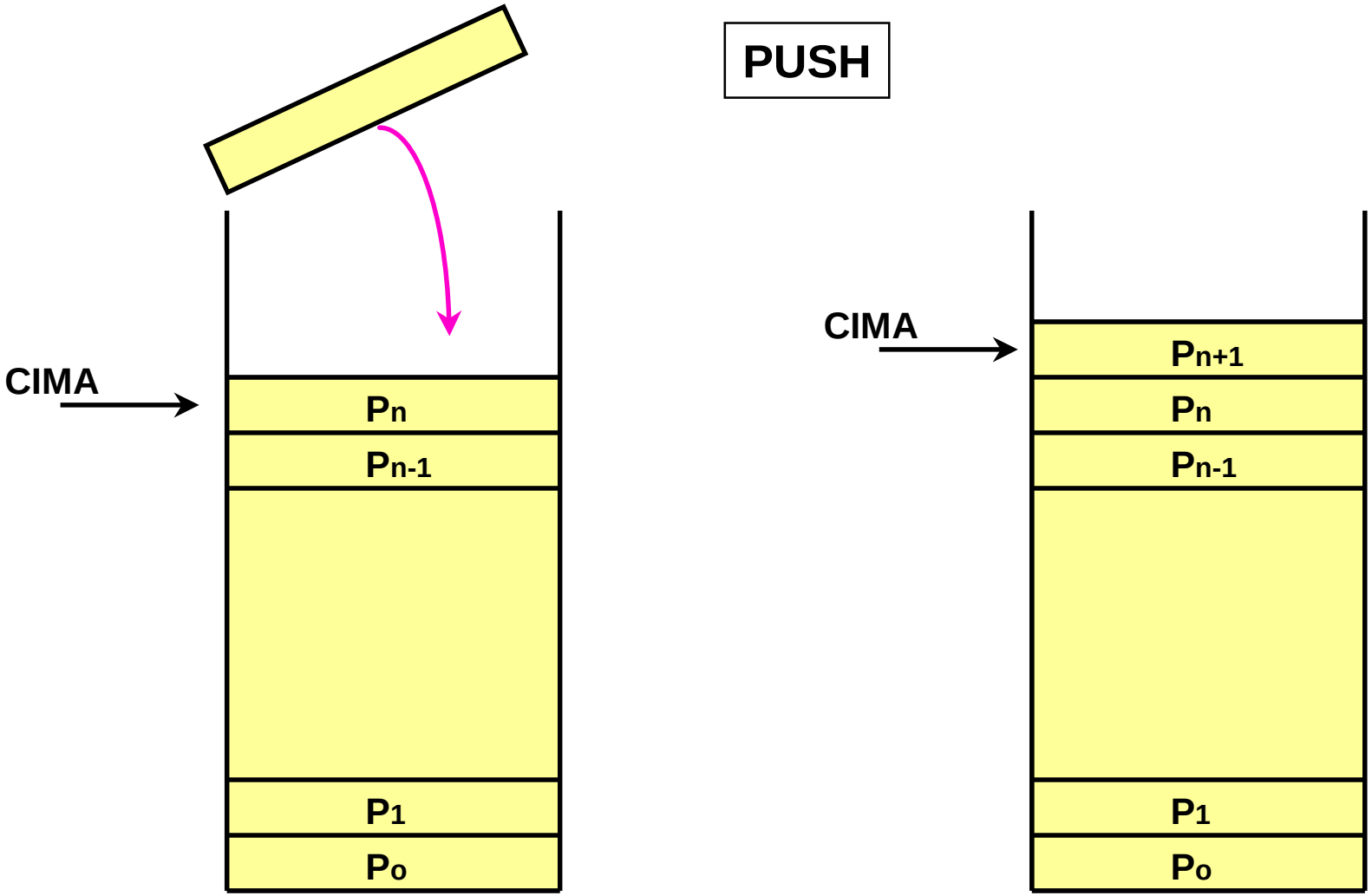
**AUTOPOSTINCREMENTO.**

**AUTOPREDECREMENTO.**

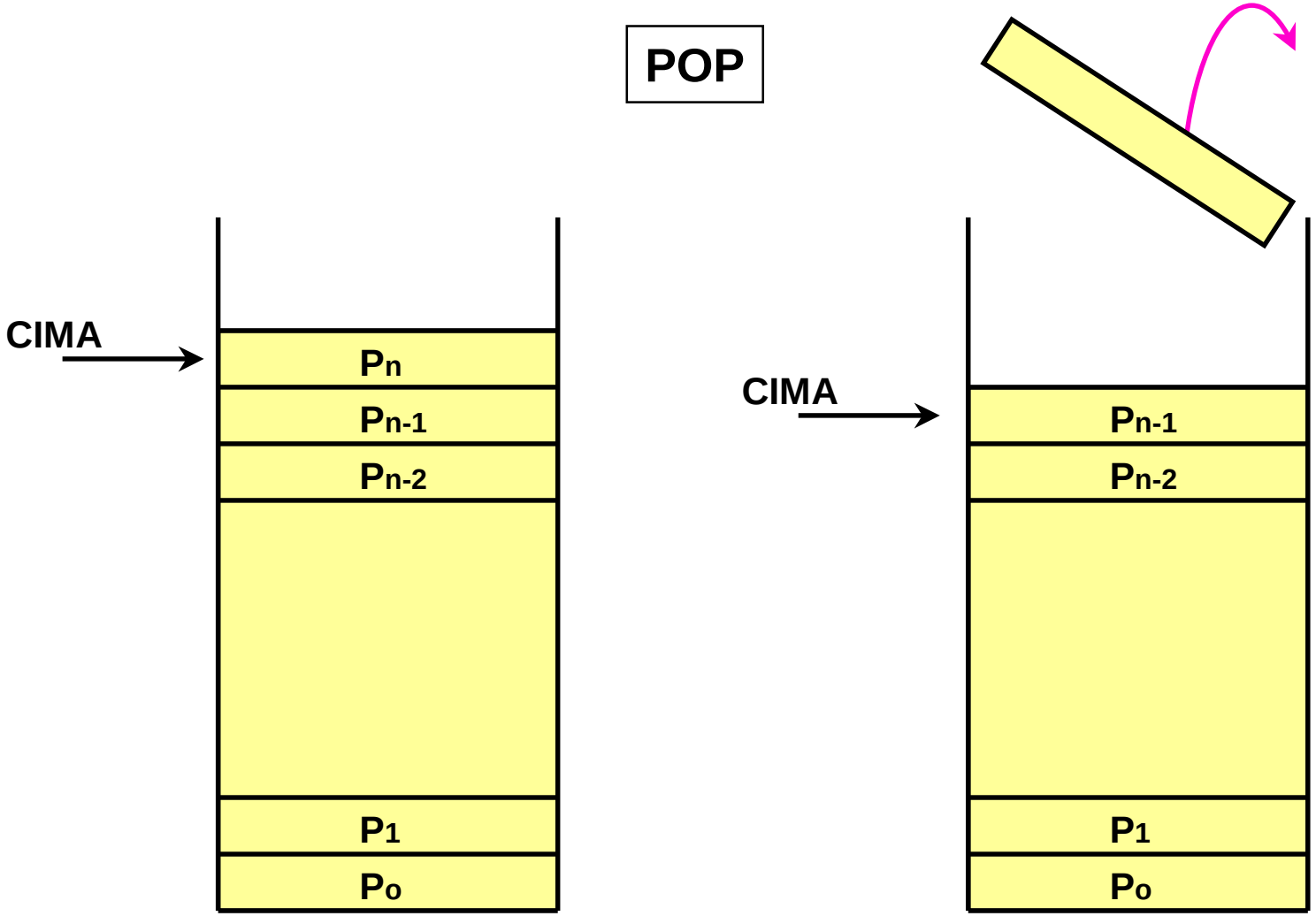
**AUTOPOSTDECREMENTO.**

- **ADD .B**
- **ADD .W**
- **ADD .L**

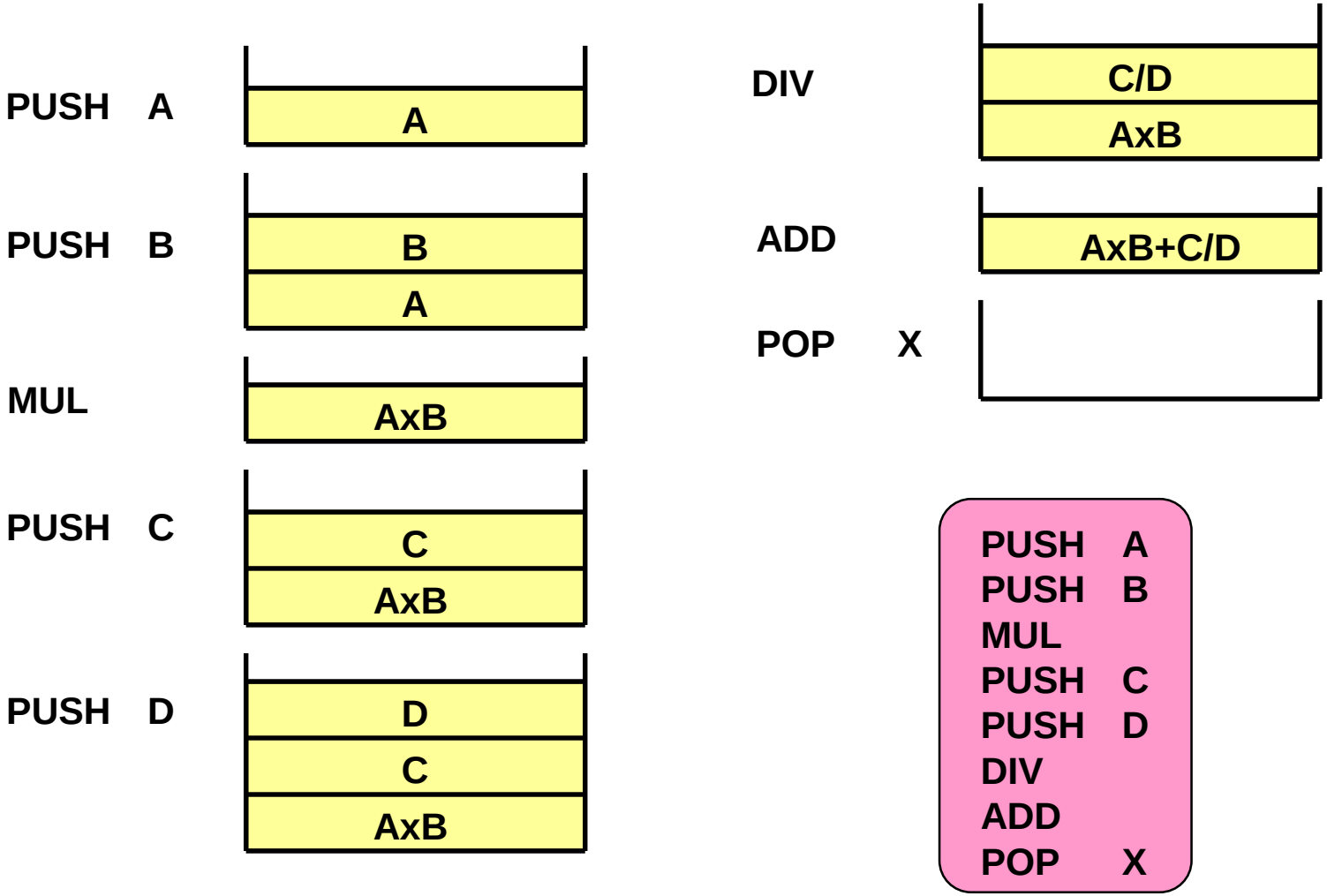
# 3.5 PILAS Y SUBPROGRAMAS



# 3.5 PILAS Y SUBPROGRAMAS



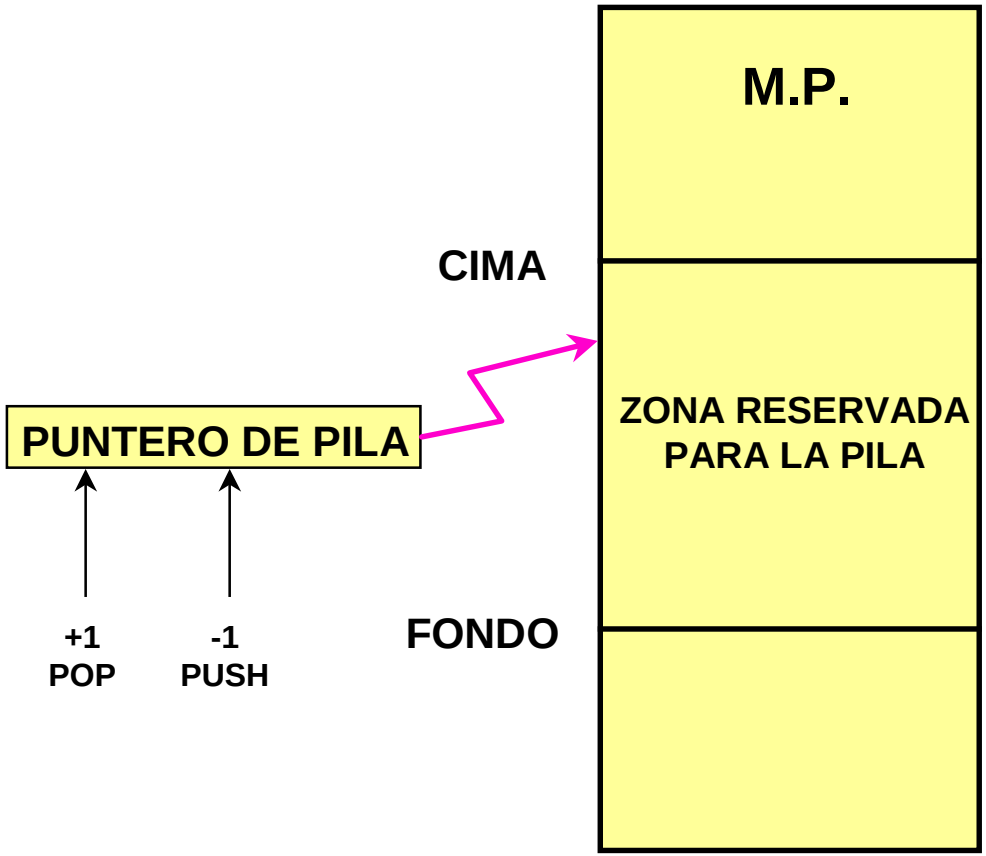
# 3.5 PILAS Y SUBPROGRAMAS




```
PUSH A
PUSH B
MUL
PUSH C
PUSH D
DIV
ADD
POP X
```

# 3.5 PILAS Y SUBPROGRAMAS

## SIMULACIÓN DE UNA PILA :



# 3.5 PILAS Y SUBPROGRAMAS

## USO DE UNA PILA PARA LOS SUBPROGRAMAS E INTERRUPCIONES:

- **Instrucciones PUSH Y POP transparentes al programador.**
- **El anidamiento de subprogramas e interrupciones no plantea ningún problema.**
- **La pila puede utilizarse para transmitir argumentos.**

# 3.6 REPERTORIOS DE INSTRUCCIONES

**DE MOVIMIENTOS DE DATOS** **( LD, ST, MOVE, MOV, PUSH, POP, IN, OUT )**

**ARITMÉTICAS, LÓGICAS Y DE COMPARACIÓN( ADD, SUB, MUL, DIV, INC, NOT, AND, OR, XOR, CLR, SET, CMP )**

**DE DESPLAZAMIENTOS** **( SHR, SHL, ROR, ROL, RORC, ROLC, SHRA, SHLA )**

**DE TRANSFERENCIA DE CONTROL** **( BR, CALL, RET, BZ, BNZ, BN, BNN, CALLZ, CALLNZ, RETZ, RETI, SKIP, SKIPZ, NOP )**

**DE GOBIERNO** **( HALT, EI, DI )**

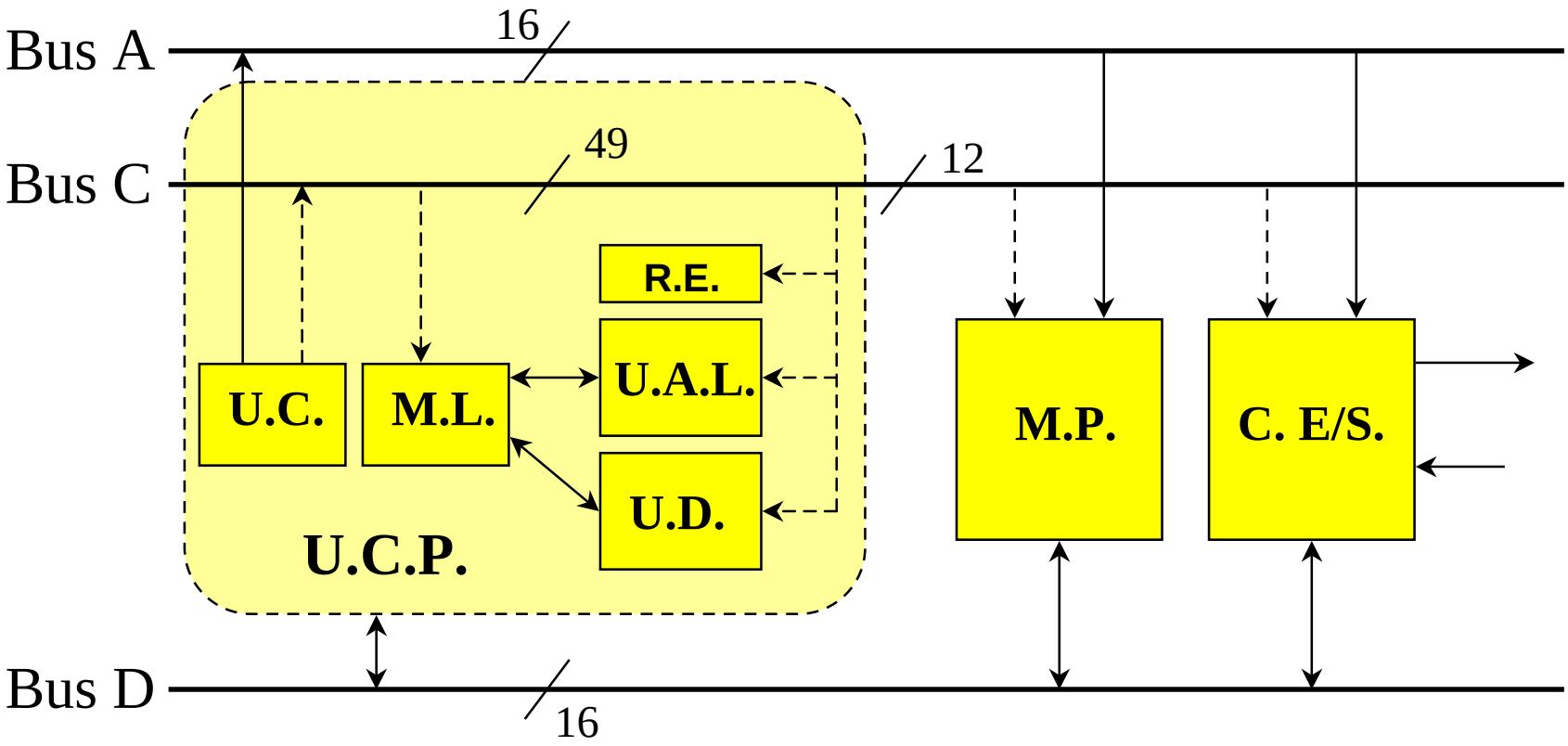
# 3.7 LENGUAJES ENSAMBLADORES

## LENGUAJE IEEE 694 :

| MODO | SÍMBOLO | EJEMPLO |
|---|---|---|
| Absoluto (directo) | Prefijo “/” | /dir |
| Página 0 | Prefijo “!” | !dir |
| Indirecto | Corchetes | [dir] |
| Relativo a CP | Prefijo “$” | $dir |
| Inmediato (literal) | Prefijo “#” | #valor |
| Indexado | Corchetes | dir[.R] |
| Preindexado | Corchetes | [dir[.R]] |
| Postindexado | Corchetes | [dir][.R] |
| Registro | Prefijo “.” | .R |
| Autopreincremento | Prefijo “++” | ++dir |
| Autopostincremento | Sufijo “++” | dir++ |
| Autopredecremento | Prefijo “--” | --dir |
| Autopostdecremento | Sufijo “--” | dir-- |

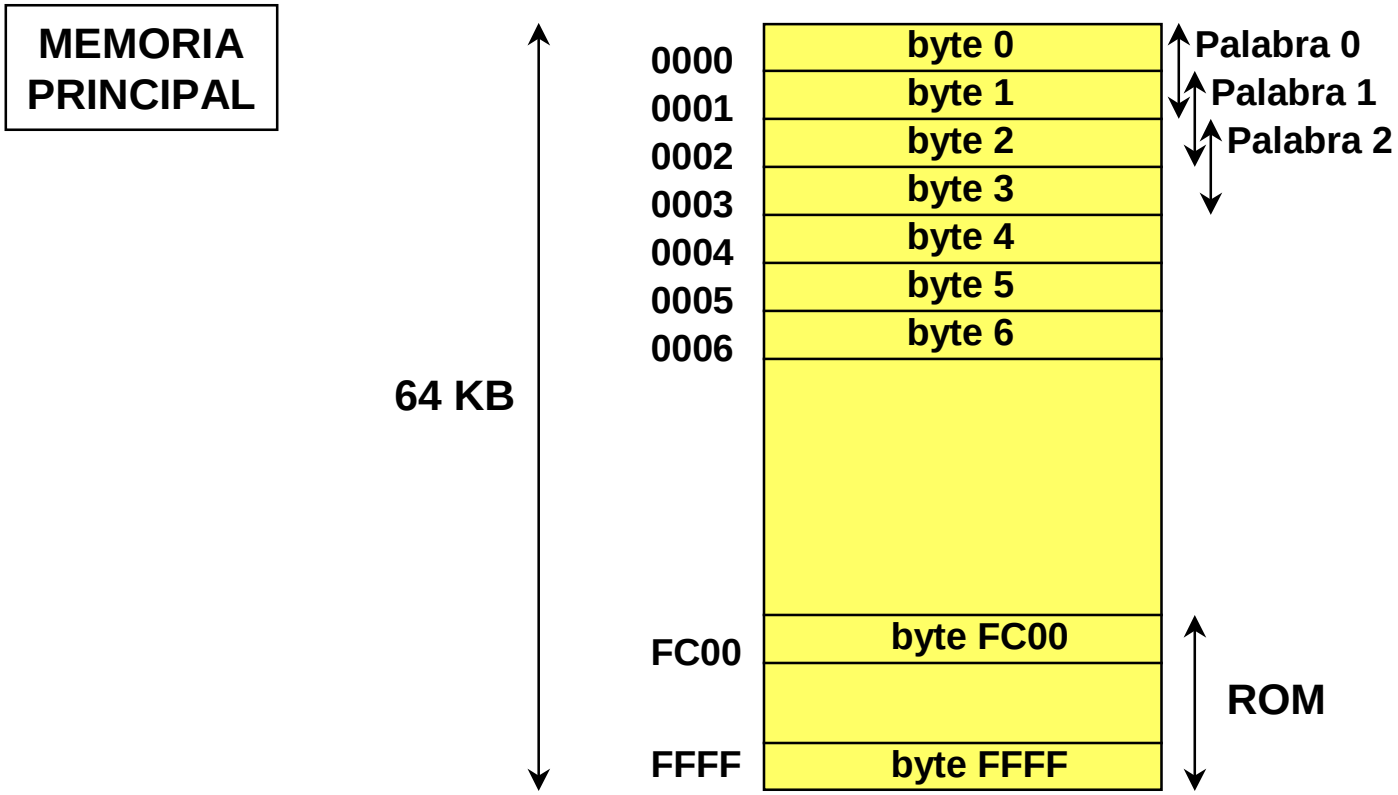
# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODELO ESTRUCTURAL :



# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODELO ESTRUCTURAL :



# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODELO ESTRUCTURAL :

**UNIDAD ARITMÉTICA Y LÓGICA**

- **La longitud de los operandos y del resultado puede ser 8 o 16 bits.**
- **El número de operaciones es mayor, 16.**
- **Los 7 bits menos significativos del registro de estado, son indicadores de cuyo valor, 0 o 1 depende del resultado de la operación realizada.**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODELO ESTRUCTURAL :

**REGISTRO DE ESTADO**

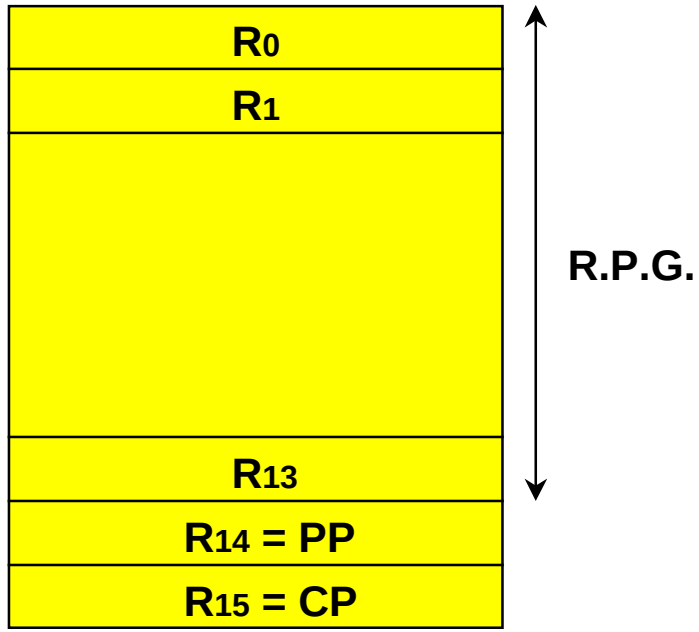
| | | | | | | | RAS | PIN | H3 | H7 | H11 | C | N | V | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Z:** Cero.
**V:** Desbordamiento.
**N:** Negativo.
**C:** Acarreo (del bit de peso 15).
**H11:** Acarreo del bit de peso 11 en aritmética BCD.
**H7:** Acarreo del bit de peso 7 en aritmética BCD.
**H3:** Acarreo del bit de peso 3 en aritmética BCD.
**PIN:** Permitir o inhibir interrupciones.
**RAS:** Modo de rastreo.

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODELO ESTRUCTURAL :

**MEMORIA LOCAL**



# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODELO FUNCIONAL :

- **Los operandos pueden ser de 8 o 16 bits.**

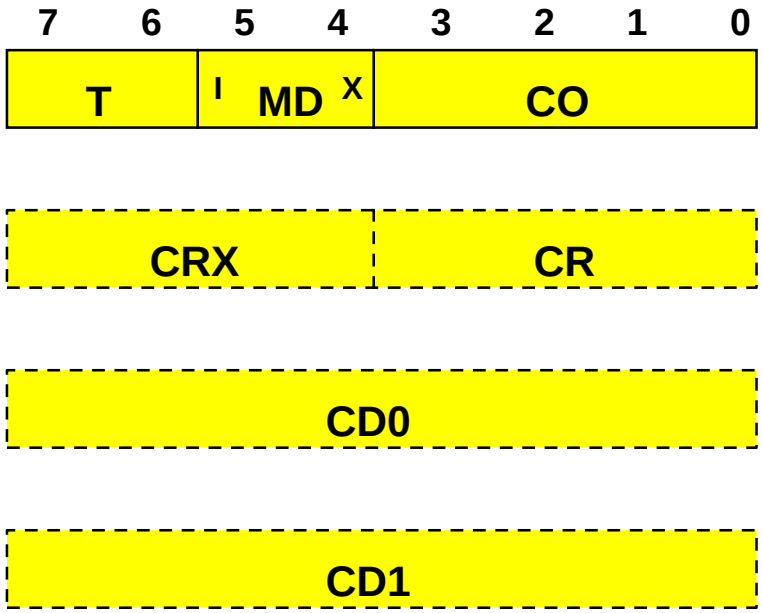
- **Las instrucciones tienen entre uno y cuatro bytes**

- **REPRESENACIÓN DE LA INFORMACIÓN**

| | |
|---|---|
| NÚMEROS | 8 o 16 bits en complemento a 2<br>( -32768 a 32767)<br>Extremista menor. |
| CARACTERES | (ASCII)<br>Extremista menor. |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODELO FUNCIONAL :




**FORMATO DE INSTRUCCIÓN**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## REPERTORIO DE INSTRUCCIONES :

**TIPOS DE INSTRUCCIONES**

| T | TIPO | LONG. | COMENTARIOS |
|---|---|---|---|
| 00 | INHERENTE | 1 BYTE | No direcciona ML ni MP. MD=XX |
| 01 | REGISTRO | 2 BYTES | Direcciona ML( con campo CR). MD=XX |
| 10 | REG. - MEM. | 2,3,4 BYTES | Longitud depende de MD. |
| 11 | MEMORIA | 2,3,4 BYTES | Longitud depende de MD. |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## REPERTORIO DE INSTRUCCIONES :

| BINARIO | HEX | NEMÓNICO | SIGNIFICADO |
|---|---|---|---|
| 00000000 | 00 | CLRC | Pone a cero el indicador C. 0 –> C |
| 00000001 | 01 | CLRV | Pone a cero el indicador V. 0 –> V |
| 00000010 | 02 | EI | Permite interrupciones. |
| 00000011 | 03 | DI | Inhibe interrupciones. |
| 00000100 | 04 | BRK | Genera interrupción de programa. |
| 00000101 | 05 | BRKV | Genera interrupción de programa si V=1. |
| 00000110 | 06 | NOP | No operación. |
| 00000111 | 07 | WAIT | Espera interrupción. |
| 00001000 | 08 | HALT | Detiene la ejecución. |
| 00001001 | 09 | RET | Retorno de subprograma. ((PP)) –> CP<br>(PP) + 2 –> PP |
| 00001010 | 0A | RETI | Retorno de interrupción. ((PP)) –> RE<br>(PP) + 2 –> PP<br>((PP)) –> CP<br>(PP) + 2 –> PP |
| 00001011 | 0B | PUSH .E | (PP) - 2 –> PP<br>(RE) –> PP |
| 00001100 | 0C | POP .E | ((PP)) –> RE<br>(PP) + 2 –> PP |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## REPERTORIO DE INSTRUCCIONES :

| BINARIO | HEX | NEMÓNICO | SIGNIFICADO |
|---|---|---|---|
| 01000000 | 40 | SHR | Desplazamiento a la derecha. |
| 01000001 | 41 | SHL | Desplazamiento a la izquierda. |
| 01000010 | 42 | ROR | Rotación a la derecha. |
| 01000011 | 43 | ROL | Rotación a la izquierda. |
| 01000100 | 44 | RORC | Rotación con C a la derecha. |
| 01000101 | 45 | ROLC | Rotación con C a la izquierda. |
| 01000110 | 46 | SHRA | Desplazamiento aritmético a la derecha. |
| 01000111 | 47 | SHLA | Desplazamiento aritmético a la izquierda. |
| 01001000 | 48 | IN | ((RX)) –> BR; (RX) = Dir. puerto e/s. |
| 01001001 | 49 | OUT | (R) –> B(RX); (RX) = Dir. puerto e/s. |
| 01001010 | 4A | PUSH | (PP) - 2 –> PP<br>(R) –> PP |
| 01001011 | 4B | POP | ((PP)) –> R<br>(PP) + 2 –> PP |
| 01001100 | 4C | CLR | 0 –> R; 0 –> C; 0 –> V; 0 –> N; 1 –> Z |
| 01001101 | 4D | NOT | Complemento a 1 de (R) –> R |
| 01001110 | 4E | NEG | Complemento a 2 de (R) –> R |
| 01001111 | 4F | ADJ | Ajuste decimal de (R) |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## REPERTORIO DE INSTRUCCIONES :

| BINARIO | HEX | NEMÓNICO | SIGNIFICADO |
|---|---|---|---|
| 10??0000 | ?0 | ADD | (R) + (DE) –> R |
| 10??0001 | ?1 | ADD .B | (R) + B(DE) –> BR |
| 10??0010 | ?2 | ADDC | (R) + (DE) + (C) –> R |
| 10??0011 | ?3 | SUB | (R) - (DE) –> R |
| 10??0100 | ?4 | SUB .B | (R) – B(DE) –> BR |
| 10??0101 | ?5 | SUBC | (R) - (DE) - (C) –> R |
| 10??0110 | ?6 | CMP | Pone valores en C, V, N, Z según (R) - (DE) |
| 10??0111 | ?7 | CMP .B | Pone valores en C, V, N, Z según (R) - B(DE) |
| 10??1000 | ?8 | AND | (R) and (DE) –> R |
| 10??1001 | ?9 | OR | (R) or (DE) –> R |
| 10??1010 | ?A | LD | (DE) –> R |
| 10??1011 | ?B | LD .B | (DE) –> BR |
| 10??1100 | ?C | ST | (R) –> (DE) |
| 10??1101 | ?D | ST .B | (R) –> B(DE) |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## REPERTORIO DE INSTRUCCIONES :

| BINARIO | HEX | NEMÓNICO | SIGNIFICADO |
|---|---|---|---|
| 11??0000 | ?0 | BC | Bifurcación a DE si (C) = 1 |
| 11??0001 | ?1 | BNC | Bifurcación a DE si (C) = 0 |
| 11??0010 | ?2 | BV | Bifurcación a DE si (V) = 1 |
| 11??0011 | ?3 | BNV | Bifurcación a DE si (V) = 0 |
| 11??0100 | ?4 | BN | Bifurcación a DE si (N) = 1 |
| 11??0101 | ?5 | BNN | Bifurcación a DE si (N) = 0 |
| 11??0110 | ?6 | BZ | Bifurcación a DE si (Z) = 1 |
| 11??0111 | ?7 | BNZ | Bifurcación a DE si (Z) = 0 |
| 11??1000 | ?8 | BR | Bifurcación incondicional a DE: DE –> CP |
| 11??1001 | ?9 | CALL | (PP) - 2 –> PP |
| | | | (CP) –> (PP) |
| | | | (DE) –> CP |
| 11??1010 | ?A | LD .E | (DE) –> RE |
| 11??1011 | ?B | ST .E | (RE) –> DE |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## MODOS DE DIRECCIONAMIENTO :

| MD | NOMBRE | EFECTO | RESULTADO CON (CRX) = 15 |
|---|---|---|---|
| 00 | AUTOINCREMENTO | (DE) = (RX)<br>(RX) + 2 –> RX<br>ó (RX) + 1 –> RX | Inmediato con CD de 1 o 2 bytes |
| 01 | INDEXADO | DE = (CD) + (RX)<br>CD: 1 Byte con signo. | Relativo a CP |
| 10 | AUTOINCREMENTO INDIRECTO | (DE) = ((RX))<br>(RX) + 2 –> RX | Directo con CD de 2 bytes |
| 11 | INDEXADO INDIRECTO | DE = ((CD) + (RX))<br>CD: 1 Byte con signo. | Relativo a CP e indirecto |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## INTERRUPCIONES :

- **Introduce en la pila los contenidos de CP y de RE, actualizando debidamente PP.**
- **Inhibe las interrupciones (poniendo a cero PIN).**
- **Recoge del bus A la dirección del vector de interrupción.**
- **Bifurca a la dirección contenida en el vector de interrupción.**

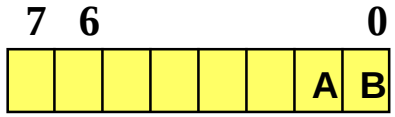
# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## INTERRUPCIONES :

### CAUSAS DE INTERRUPCIÓN

- **Pueden interrumpir hasta 128 controladores de periféricos.**

**ESTADO:**




**B: Preparado o no.**
**A: Interrupciones permitidas o no.**

**Iniciar pantalla: A=1; B=1;**
**Iniciar teclado: A=1; B=0;**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## INTERRUPCIONES :

**CAUSAS DE INTERRUPCIÓN**

- **Causas internas:**

  **BK:** **Originada por la instrucción BRK (“Llamada al supervisor”)**

  **BKV:** **Originada por la instrucción BRKV (“Cuando (V=1”)**

  **RS:** **Originada por el indicador RAS (“modo de rastreo”)**

**Estas interrupciones no se pueden inhibir individualmente.**

- **Entrada de interrupción no enmascarable.**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## INTERRUPCIONES :

**VECTORES DE INTERRUPCIÓN**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## INTERRUPCIONES :

**ESQUEMAS DE INTERRUPCIÓN:**

- **UCP:**
  - **Guarda en la pila los contenidos de CP y RE.**
  - **Pone el indicador PIN a “0” y pasa a investigar la causa de interrupción.**
  - **Mira la entrada de interrupción no enmascarable.**
  - **Explora las tres causas internas (programa, desbordamiento y rastreo).**
  - **Genera una señal de reconocimiento de la interrupción y el HW externo deposita en el Bus A una dirección de vector.**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## INTERRUPCIONES :

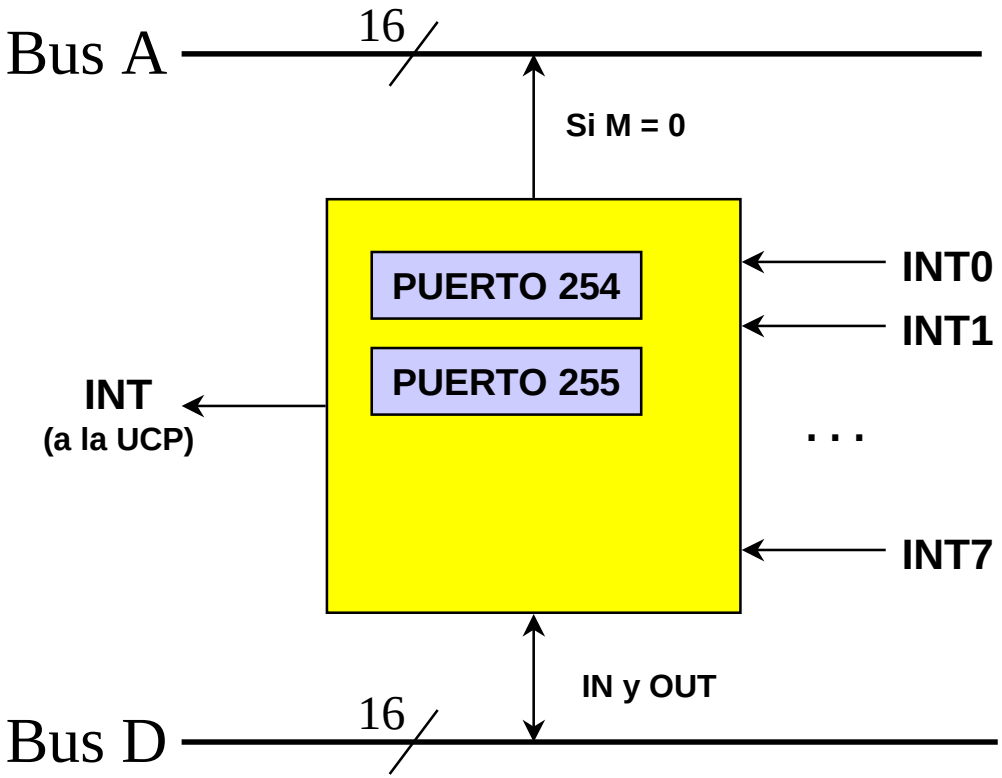
**ESQUEMAS DE INTERRUPCIÓN:**

- **Dependiendo del HW externo:**
  - **El Hw no identifica al periférico (identificación mediante SW).**
  - **Consulta por HW (periféricos en cadena).**
  - **Controlador de interrupciones externo (gestiona prioridades para permitir anidamiento).**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## INTERRUPCIONES :

### CONTROLADOR DE INTERRUPCIONES:





# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## ENSAMBLADOR DE ALGORÍTMEZ :

**CÓDIGOS DE OPERACIÓN:**

**Pueden ir acompañados de .B o .W, para indicar que la operación es sobre byte o sobre palabra. “ST.B”.**

**REGISTROS:**

**Se indican con un punto seguido del número de registro, excepto el registro RE que se indica con “.E” en las instrucciones POP, PUSH LD y ST.**

**ORDEN DE LOS OPERANDOS:**

**1º se escribe el símbolo del registro y 2º la representación simbólica de la dirección de MP donde está el segundo operando.**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## ENSAMBLADOR DE ALGORÍTMEZ :

**MODOS DE DIRECCIONAMIENTO:**

| | | |
|---|---|---|
| **AUTOINCREMENTO:** | **LD.B** | **.0, [.3++]** |
| **INDEXADO:** | **LD.B** | **.0, /ETI[.3]** |
| **AUTOINCREMENTO INDIRECTO:** | **LD.B** | **.0, [[.3++]]** |
| **INDEXADO INDIRECTO:** | **LD.B** | **.0, [/ETI[.3]]** |
| **INMEDIATO:** | **LD.B** | **.0, #-125** |
| **RELATIVO A PROGRAMA:** | **LD.B** | **.0, $ETI** |
| | **LD.B** | **.0, ETI** |
| **DIRECTO:** | **LD.B** | **.0, /ETI** |
| **RELATIVO A PROGRAMA E INDIRECTO:** | **LD.B** | **.0, [$ETI]** |
| | **LD.B** | **.0, [ETI]** |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## ENSAMBLADOR DE ALGORÍTMEZ :

**CONSTANTES NUMÉRICAS:**

| | |
|---|---|
| **BINARIO:** | **B’** |
| **OCTAL:** | **Q’** |
| **DECIMAL:** | **D’** |
| **HEXADECIMAL:** | **H’** |

**CONSTANTES ALFANUMÉRICAS:**

**Se expresan escribiendo la cadena entre comillas. 1 byte por carácter.**
**Extremista menor.**

**ETIQUETAS:**

**De 1 a 8 caracteres empezando por una letra. Debe haber un espacio en blanco entre la etiqueta y la instrucción o pseudoinstrucción.**
**Si no hay etiqueta, la primera columna de la línea debe estar en blanco.**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## ENSAMBLADOR DE ALGORÍTMEZ :

**DIRECTIVAS:**

**ORG:** Debe ir seguida de una cte. numérica: 0 <= cte. <= 65535
**EQU:** Va acompañada de una etiqueta y de una cte.

```
        ORG     20
DIRE    EQU     100
PROG    ADD     .9, /DIRE       ; ADD   .9, /100
        ADD     .9, #DIRE       ; ADD   .9, #100
        ADD     .9, DIRE        ; ADD   .9, 100
```

**END:** Siempre al final del programa.

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## ENSAMBLADOR DE ALGORÍTMEZ :

**PSEUDOINSTRUCCIONES:**

**RES.B:** **Reserva bytes en memoria.**

```
        ORG     101
ZONA    RES.B   50
```

**RES:** **Reserva palabras en memoria.**

```
        ORG     101
ZONA    RES     100
```

**DATA.B:** **Se traduce como una secuencia de bytes en memoria.**

**DATA:** **Se traduce como una secuencia de palabras en memoria.**

```
        ORG      0
        DATA.B   H'F,-5,"C"
C128    DATA     128
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## PROBLEMA:

**TRADUCIR AL LENGUAJE MÁQUINA:**

```
        ORG     50
        BR      PRG
DOS     EQU     2
ZONAD   EQU     100
FIND    EQU     114
CLV     DATA    0
DRES    RES     1
PRG     LD      .0, #ZONAD
        LD      .1, CLV
BUCLE   CMP     .1, [.0++]
        BNZ     SIGUE
        SUB     .0, #DOS
        ST      .0, DRES
        HALT
SIGUE   CMP     .0, #FIND
        BNZ     BUCLE
        HALT
        END
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## PROGRAMACIÓN:

**Suma de 50 números, almacenados entre las posiciones 50 y 149, dejando el resultado en la posición 150.**

```
            ORG     0
            BR      /PRINC
C           DATA    50
CC          DATA    150
PRINC       LD      .0, /C
            CLR     .1
BUCLE       ADD     .1, [.0++]
            CMP     .0, /CC
            BNZ     /BUCLE
            ST      .1, /150
            HALT
            END
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## PROGRAMACIÓN:

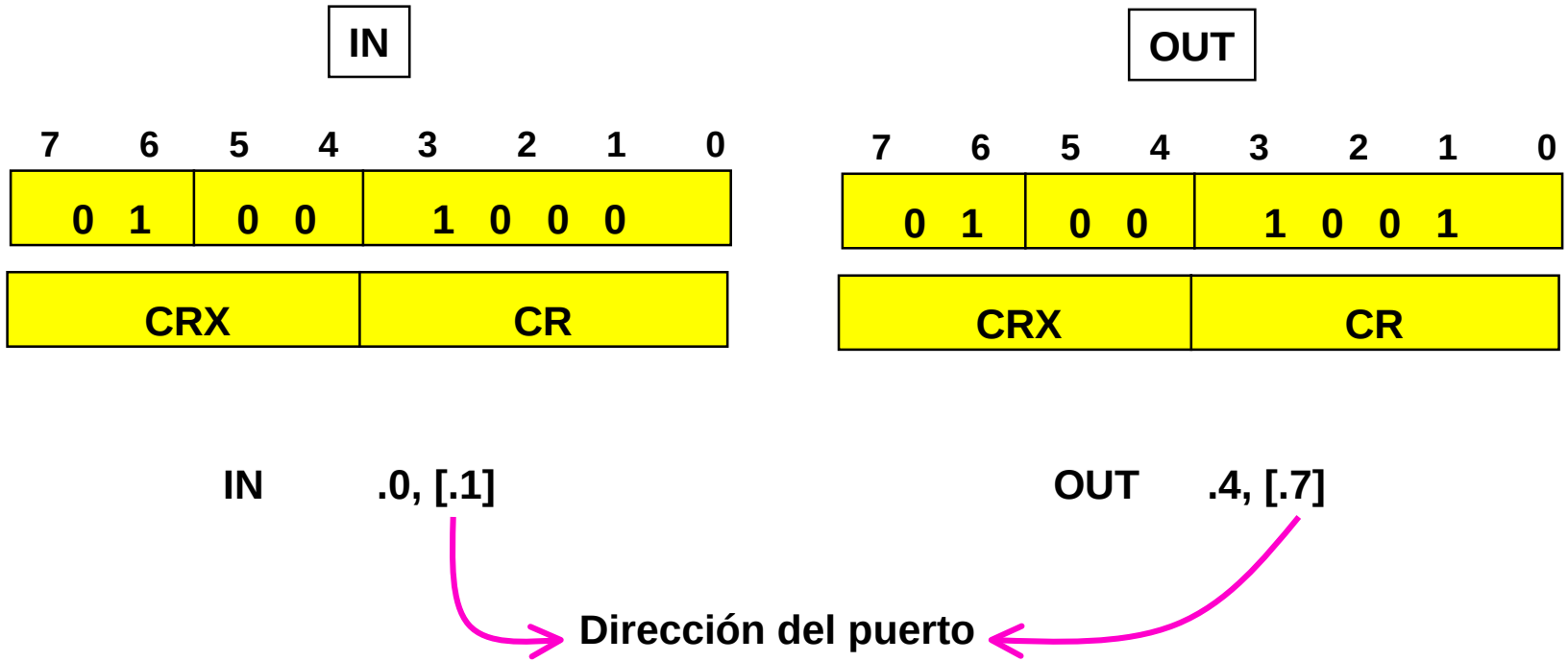
**Intercambio de las zonas de memoria A(100-149) y B(200-249)**

```
                ORG     0
CIEN            EQU     100
DOSC            EQU     200
CCINC           EQU     150
                LD      .0, #CIEN
                LD      .1, #DOSC
BUCLE           LD.B    .2, /0[.0]
                LD.B    .3, /0[.1]
                ST.B    .2, [.1++]
                ST.B    .3, [.0++]
                CMP     .0, #CCINC
                BNZ     BUCLE
                HALT
                END
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

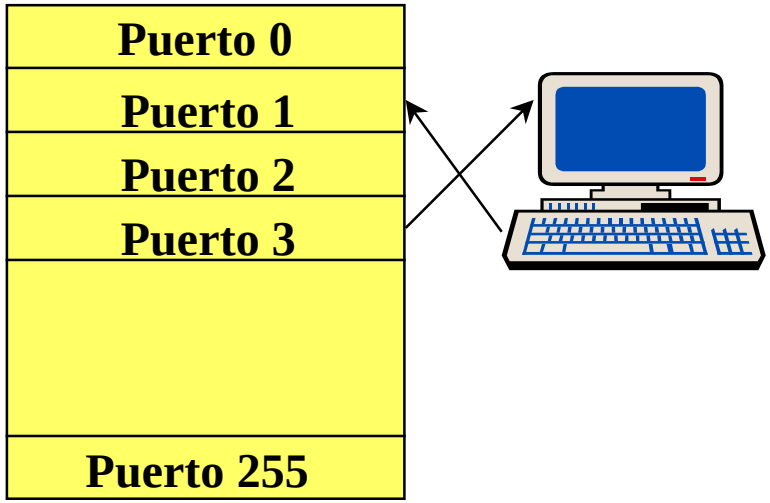
## COMUNICACIONES CON LOS PERIFÉRICOS:

### INSTRUCCIONES DE ENTRADA SALIDA



# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## COMUNICACIONES CON LOS PERIFÉRICOS:




**INICIACIÓN DE LOS PERIFÉRICOS**

```
LD.B    .0, #0
LD.B    .1, #1
LD.B    .2, #2
LD.B    .3, #3

CLR     .4
OUT     .4, [.0]
LD.B    .4, #1
OUT     .4, [.2]
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## COMUNICACIONES CON LOS PERIFÉRICOS:

### LECTURA Y ESCRITURA DE CARACTERES AISLADOS

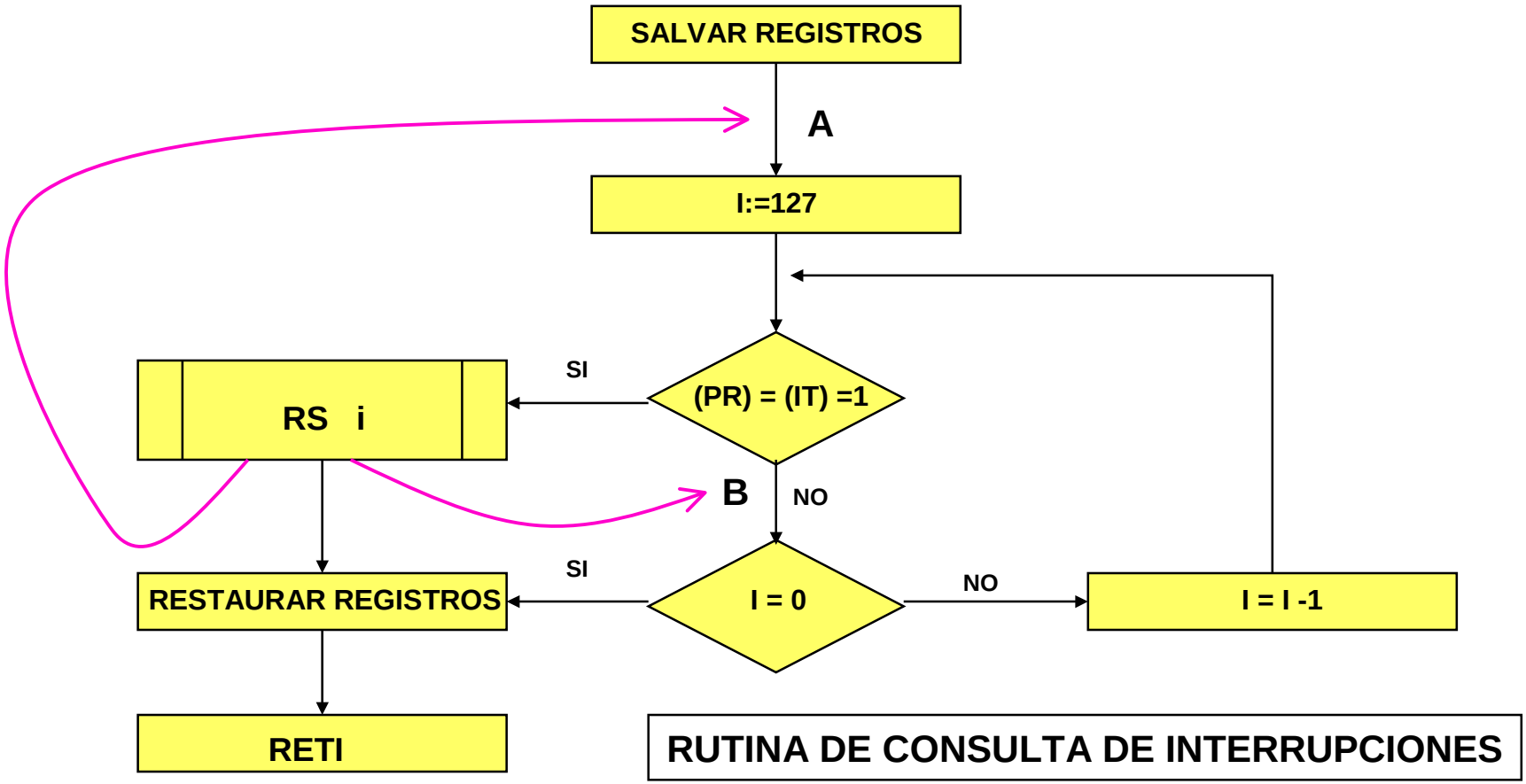
```
            ORG     H'F030
LEECAR      IN      .4, [.0]
            AND     .4, #1
            BZ      LEECAR
            IN      .5, [.1]
            CALL    ESCCAR
            RET
ESCCAR      IN      .4, [.2]
            AND     .4, #1
            BZ      ESCCAR
            OUT     .5, [.3]
            RET
            END
```

```
CALL    LEECAR

CALL    ESCCAR
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## CONSULTA Y SERVICIO DE INTERRUPCIONES:



# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## CONSULTA Y SERVICIO DE INTERRUPCIONES:

**TABLA DE PERIFÉRICOS**

| Dir. MP | +0 | +1 | +2 | +3 | … | +19 |
|---|---|---|---|---|---|---|
| [1024] | FE | 0x | xx | xx | … | xx |
| [1044] | 10 | 8x | xx | xx | … | xx |
| [1064] | 0E | 1x | xx | xx | … | xx |
| [1084] | 00 | 9x | xx | xx | … | xx |
| [1104] | FF | xx | xx | xx | … | xx |

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## CONSULTA Y SERVICIO DE INTERRUPCIONES:

**RUTINA DE CONSULTA DE INTERRUPCIONES**

```
TBPRF   EQU   1024
MSKOP   EQU   H'0080
MSKIT   EQU   H'0003
MSKFIN  EQU   H'FF
        ORG   266
        PUSH  .0
        PUSH  .1
        PUSH  .2
        LD    .0, #TBPRF
        CLR   .1
BUCLE   LD.B  .1, [.0++]
        CMP.B .1, #MSKFIN
        BZ    FIN
        LD.B  .2, [.0++]
        AND   .2, MSKOP
        BZ    SIGUE
        IN    .2, [.1]
        NOT   .2
        AND   .2, #MSKIT
        BNZ   SIGUE
        CALL  [[.1++]]
FIN     POP   .2
        POP   .1
        POP   .0
        RETI
SIGUE   ADD   .0, #18
        BR    BUCLE
        END
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## CONSULTA Y SERVICIO DE INTERRUPCIONES:

**PROGRAMA QUE USA LA RUTINA DE SERVICIO DE INTERRUPCIONES**

```
                ---
MENS1           DATA    “Estoy con cálculo 1”,H’0D
MENS2           DATA    “Estoy con cálculo 2”,H’0D
                ---
                CLR     .0
                ST.B    .0, /337
                LD      .0, #MENS1
                ST      .0, /338
                LD.B    .1, #2
                IN      .0, [.1]
                OR      .0, #3
                OUT     .0, [.1]
                EI
                ---
```

```
                ---
ESPERA          LD.B    .0, /337
                BZ      ESPERA
                CLR     .0
                ST.B    .0, /337
                LD      .0, #MENS2
                ST      .0, /338
                LD.B    .1, #2
                IN      .0, [.1]
                OR      .0, #2
                OUT     .0, [.1]
                ---
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## CONSULTA Y SERVICIO DE INTERRUPCIONES:

**RUTINA DE SERVICIO DE INTERRUPCIONES**

```
            ORG     337
SMENS       RES.B   1
PUNT        RES     1
            LD      .0, PUNT
            LD.B    .1, #3
            LD.B    .2, [.0++]
            OUT     .2, [.1]
            CMP.B   .3, #H'0D
            BZ      FIN
            ST      .0, PUNT
            RET
FIN         LD.B    .1, #2
            IN      .2, [.1]
            AND     .2, #1
            BZ      FIN
            LD.B    .2, #H'0A
            OUT     .2, [.1]
            LD.B    .1, #2
            IN      .2, [.1]
            AND     .2, #H'FD
            OUT     .2, [.1]
            LD.B    .0, #1
            ST.B    .0, SMENS
            RET
            END
```

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## CONSULTA Y SERVICIO DE INTERRUPCIONES:

**ANIDAMIENTO DE INTERRUPCIONES**

**ACCIONES DE LA RUTINA DE SERVICIO:**

- **Guardar los registros de trabajo de la RS.**
- **Inhibir las interrupciones de prioridad inferior o igual.**
- **Ejecutar la instrucción EI.**
- **Comenzar ejecución.**
- **Ejecutar DI, restaurar registros, permitir interrupciones de los periféricos con menos prioridad, y volver a la RCI que devolverá el control al programa interrumpido.**

**(****)**

# 3.8 ALGORÍTMEZ. PROGRAMACIÓN

## CONSULTA Y GESTIÓN DE PRIORIDADES POR HW:

- **CONSULTA POR HW**

- **CONTROLADOR DE INTERRUPCIONES EXTERNO**