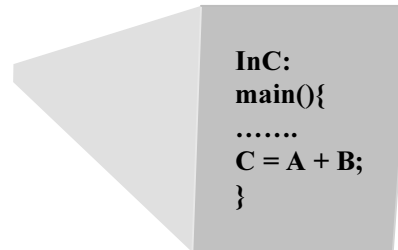


# Introducción al software en los $\mu$ procesadores

## ■ Ejemplo de comandos de la CPU:

### ■ Tarea:

- | Sumar dos números A y B.
- | Suponer los datos A y B almacenados en las posiciones  $1000_2$  y  $1010_2$  respectivamente.
- | El resultado debe almacenarse en C (posición  $1100_2$ ).



```
InC:
main(){
.....
C = A + B;
}
```


## ■ Nota: "C" No es un lenguaje binario!

## ■ ¿ Como lo entiende la máquina ?


- Es *convertido* a binario por dos programas:
  - | El COMPILADOR ("cc") y el ENSAMBLADOR ("as").

## Principales características

## ■ Sabemos que la máquina entiende los siguientes comandos, denominados *instrucciones*.

- |                       |   |                     |
|-----------------------|---|---------------------|
| 00000000 <sub>2</sub> | Sumar el contenido de dos registros   | (Add).              |
| 00000010 <sub>2</sub> | Sumar el contenido del acumulador con el contenido de una posición de memoria | (Add).              |
| 00000101 <sub>2</sub> | Idem pero restando los operandos  | (Subtract)          |
| 00100011 <sub>2</sub> | Cargar un registro desde memoria  | (Load Acumulator).  |
| 00101011 <sub>2</sub> | Almacenar un registro en memoria  | (Store Acumulator). |
- 

Estas codificaciones corresponden a los "OpCodes".



Estas codificaciones corresponden a los Nemónicos.

## ■ Algunas características comunes:

- La mayoría de las instrucciones poseen 1, 2 ó 3 *operandos*.
- Los *registros* son elementos internos de almacenamiento temporal.
- Las instrucciones que trabajan con registros llevan menos tiempo de ejecución.

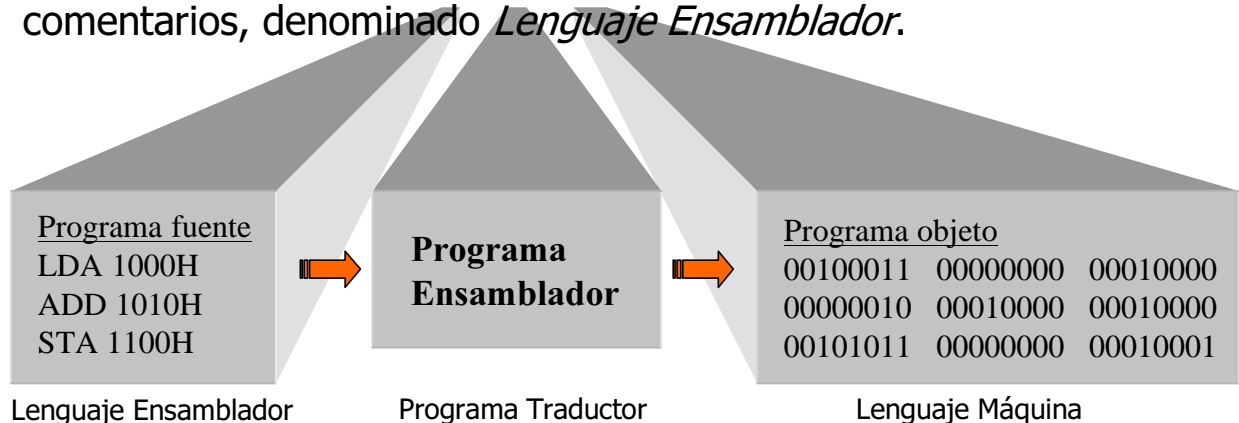
# Ensamblado de un programa

## Instrucciones en Lenguaje Máquina    Comentarios

00100011	00000000	00010000	Cargar el acumulador con el dato (1000H).
00000010	00010000	00010000	Sumar el cont. del acumulador con el dato (1010H).
00101011	00000000	00010001	Almacenar el cont. del acumulador en la pos. 1100H.

## ■ ¿Qué es más fácil de entender, el lenguaje máquina o los comentarios?. Los comentarios!.

- Utilización de un *lenguaje intermedio* que se asemeja a los comentarios, denominado *Lenguaje Ensamblador*.



## Programas ensambladores

### ■ Programa ensamblador:

- Realiza el ensamblado línea a línea del programa fuente editado.
- Guarda una estrecha relación con la estructura de la máquina.
- Todo programa ensamblador está asociado a un lenguaje ensamblador propio.
- Da la posibilidad de utilizar *etiquetas*, *pseudoinstrucciones* y *directivas*, con la finalidad de facilitar la fase de programación.
  - Etiquetas: Es un nombre simbólico que se puede utilizar en sustitución de la dirección que le corresponde.
  - Pseudoinstrucciones: Información para el programa ensamblador que éste entiende, pero que no se traducen por una instrucción de máquina.
  - Directivas: Son órdenes para el ensamblador cuyo vestigio desaparece totalmente en el programa traducido.

# Ensambladores y lenguaje ensamblador

## ■ Ensambladores cruzados:

- Permiten ensamblar código de un microprocesador (ej. 8085) en sistemas basados en micros diferentes (ej. 80x86, Pentium, etc...).
- Dan la posibilidad de obtener un listado del programa ensamblado y una lista de referencias cruzadas.

## ■ Lenguaje ensamblador:

- Es propio de cada máquina.
- No tiene paralelismo con el lenguaje convencional.
- Producen programas objetos más depurados, menos extensos.
- Sintaxis típica de un lenguaje ensamblador:

•	Etiqueta	Código de Operación	Operando	Comentario
---	----------	---------------------	----------	------------

## Aspecto de un programa en ensamblador

## ■ Ejemplo de programa en ensamblador del 8085:

2500 A.D. 8085 Macro Assembler - Version 4.03b

-----

Input Filename : pr1.asm  
Output Filename : pr1.obj

```
1      1100      ydir equ 1100h
2      1101      zdir equ 1101h
3      044E      xdir equ 1102h
4 0100                      org 100h
5 0100 3A 00 11      lda ydir
6 0103 21 01 11      lxi h,zdir
7 0106 86           add m
8 0107 32 4E 04      sta xdir
9 010A                      end
```

Defined Symbol		Name	Value	References
Pre	CODE		0000	
Pre	DATA		0000	
3	xdir	=	044E	8
1	ydir	=	1100	5
2	zdir	=	1101	6

Mon May 10 1999 19:51

Lines Assembled : 9    Assembly Errors : 0

# Aspecto de un programa en ensamblador

## ■ Ejemplo de programa en ensamblador del 8085:

### ■ Más pseudoinstrucciones y uso de etiquetas:

2500 A.D. 8085 Macro Assembler - Version 4.03b

: Programa para que ustedes aprendan

Input Filename : guay.asm  
Output Filename : guay.obj

```
1      003A      si equ 3ah
2      00F0      no equ f0h
3      0041      cc equ 'A'
4 1000
5 1000 3A                db si
6 1001 F0                db no
7 1002                  ds cc+1
8 1044 41                db cc
9 1045 3E F0             mvi a,no
10 1047                  end
```

Mon May 10 1999 19:50

```
ORG      1000H
LONG     EQU      10
JMP      INICIO
TABLA    DS       10
DATO     DB       2A
INICIO   LXI      H,TABLA
          MVI      B, LONG
          MVI      A, DATO
SEGUIR   MOV      A, M
          INX      H
          DCR      B
          JNZ      SEGUIR
          HLT
          END
```

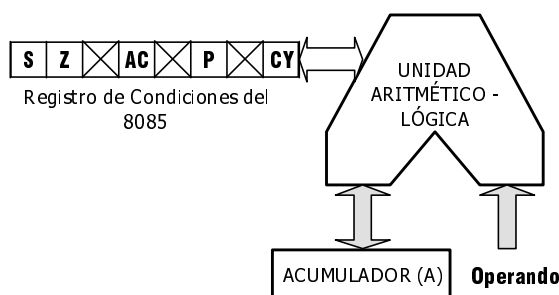
## Clasificación de las instrucciones I

## ■ Tipos de instrucciones más representativos usados en los μprocesadores: (los ejemplos corresponden a los ensambladores del 8085 y 68HC11).

- Aritméticas: Todas aquellas instrucciones que ejecutan operaciones aritméticas. Suma, Resta, Incremento, Decremento, etc ... Estas instrucciones afectan a los señalizadores del registro de estado. Los más importantes son: C, AC, Z, S, P, O.

### ■ Ejemplos:

```
ADD B
SBB M
CPI 55H
```



- **S: Flag de Signo.** Corresponde con el bit más significativo del resultado.
- **Z: Flag de Zero.** Se activa si el resultado es cero.
- **P: Flag de Paridad.** Se activa si el resultado da un número par de unos.
- **CY: Flag de Acarreo.** Se activa si se produce un acarreo en una suma o un préstamo en una resta.
- **AC: Flag de Acarreo Auxiliar.** Se emplea en representación BCD. Se activa si se produce un acarreo del 3<sup>er</sup> bit al 4<sup>o</sup> bit.

# Clasificación de las instrucciones II

I Lógicas: Efectúan operaciones lógicas, tales como AND, OR, XOR, NAND, NEGACIÓN, etc...

– Ejemplos: ANA D  
ORA M  
CMA

I De transferencia: de datos entre la CPU, Memoria y/o dispositivos de E/S.

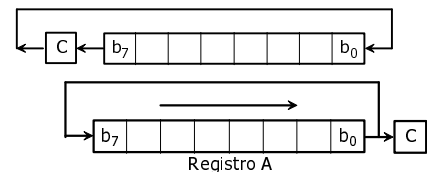
– Ejemplos: MOV A,H                      LXI H,23FAH  
LHLD 2200H                      STA 14FDH

• Dentro de este grupo se encuentran las instrucciones de E/S:

– Ejemplos: IN 23H  
OUT 48H

I De desplazamiento y rotación: Desplazan o rotan el contenido del acumulador.

– Ejemplos: RAL  
RRC



# Clasificación de las instrucciones II

I De salto: Afectan a la ejecución secuencial de las instrucciones del programa. Incluyen fundamentalmente instrucciones de salto y bifurcación, y de llamadas y retornos a subrutinas. Pueden ser condicionales o no.

– Ejemplos: JMP ETIQUETA  
CALL PROC1  
RET

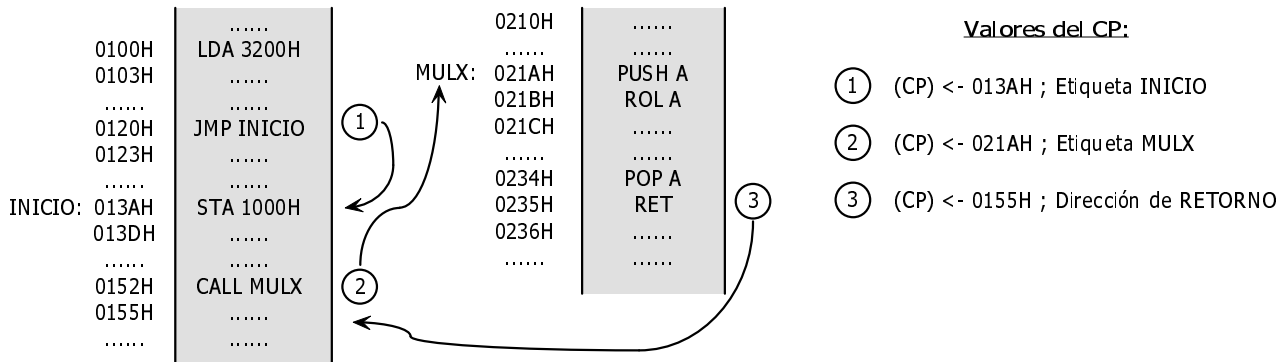
I De control: Engloban las instrucciones de petición de interrupción, para sincronización con elementos externos (coprocesador matemático), de activación de los señalizadores (flags). En general es el grupo más particular de la máquina.

– Ejemplos: (8085)                      PUSH PSW                      EI  
   HLT                                DI  
   XTHL                            NOP  
   RIM                              SIM

# Instrucciones de salto

## ■ Instrucciones de salto (o de ruptura de secuencia):

- El uso de subrutinas es una técnica muy importante en el diseño de software para sistemas con  $\mu$ procesadores.
- Ejemplo de programa:

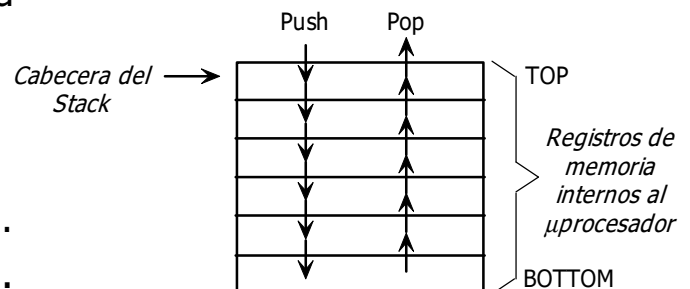


- ¿ De dónde sale la dirección de retorno en el paso 3 ? Uso de la PILA!!

# Operaciones sobre la PILA

## ■ Tratamiento de la PILA (*Stack*):

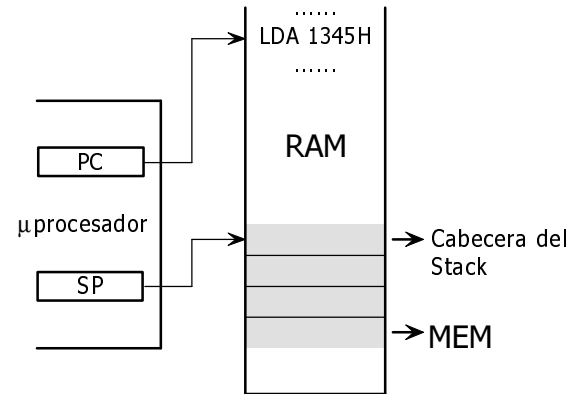
- Definición: Es un *buffer* de memoria, constituido por registros en cascada, tipo LIFO (Last-Input First-Output).
- El único registro accesible en un momento dado, para leer o escribir, es el *Puntero del Stack* (Cabecera o Cima del Stack).
- Operaciones sobre la pila:
  - Escritura (*Push*): Contenido de cada uno de los registros es desplazado a la siguiente posición más baja.
  - Lectura (*Pop*): Contenido de cada uno de los registros es desplazado hacia la siguiente posición más alta.
- Stack de profundidad fija y limitada.



# Implementación de una PILA en memoria

## ■ Implementación del Stack en memoria RAM externa:

- Necesidad de un Puntero de Pila (*Stack Pointer*) interno al  $\mu$ procesador para direccionar la Cabecera de la Pila.
- Longitud del Stack variable.
- Escritura (*Push*): Decremento del SP.
- Lectura (*Pop*): Incremento del SP.
  - Éstas dos últimas operaciones son gestionadas por la Unidad de Control.
- Inicialización del área de memoria para el Stack:
  - LDS MEM ;(Load Stack Pointer).
  - MEM corresponderá con una posición alta de la memoria RAM.



## Uso de la PILA: Almacenamiento de datos

### ■ Uso de la Pila:

- Área de almacenamiento temporal de datos.
  - Instrucción PUSH R:
    - Transfiere el contenido del registro R a la posición de memoria apuntada por el *Stack Pointer*.
      - $(R) \rightarrow (SP)$ ;
      - $(SP) - 1 \rightarrow SP$ ;
  - Instrucción POP R:
    - Transfiere el contenido de la dirección de memoria apuntada por el *Stack Pointer* al registro R.
      - $((SP) + 1) \rightarrow R$ ;
      - $(SP) + 1 \rightarrow SP$ ;

# Uso de la PILA: Llamadas a subrutinas

## ■ Área de almacenamiento temporal de las direcciones de retorno en el uso de las subrutinas.

### ■ Instrucción CALL SUBR:

- Guarda en la pila la dirección de retorno de la subrutina (operación *Push* del PC). Carga el PC con los bytes 2 y 3 de la instrucción.
  - (PC) -> (SP);
  - (PCh) -> (SP) -1;
  - (SP) -2 -> SP;
  - byte2 -> PC;
  - byte3 -> PCh;



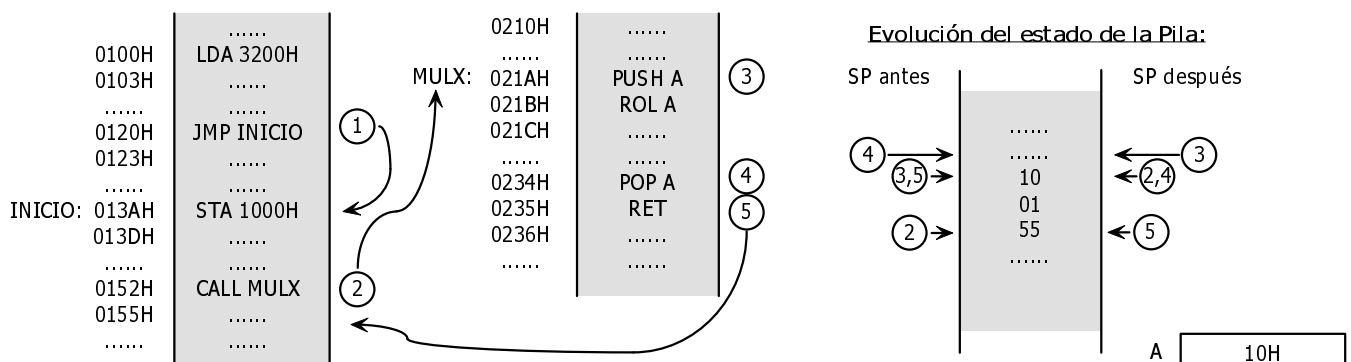
Siempre que el convenio de almacenamiento sea *Little-Endian*.

### ■ Instrucción RET:

- Transfiere el control a la instrucción que sigue al CALL, recuperando la dirección de retorno desde la pila.
  - ((SP) +1) -> PCh;
  - ((SP) +2) -> PC;
  - (SP) +2 -> SP;

## Ejemplo de manejo de la PILA

### ■ Ejemplo de manejo de la pila:



### ■ Posibilidad de anidamiento de subrutinas.

### ■ Llamadas y retornos condicionales a los bits del registro de estado.



# Modos de direccionamiento I

## ■ Modos de direccionamiento más comunes:

- Las instrucciones disponen de diferentes formas para localizar los datos con los que debe operar. Estas variantes reciben el nombre de *Modos de Direccionamiento*.

- Inmediato: El dato está contenido en la instrucción.

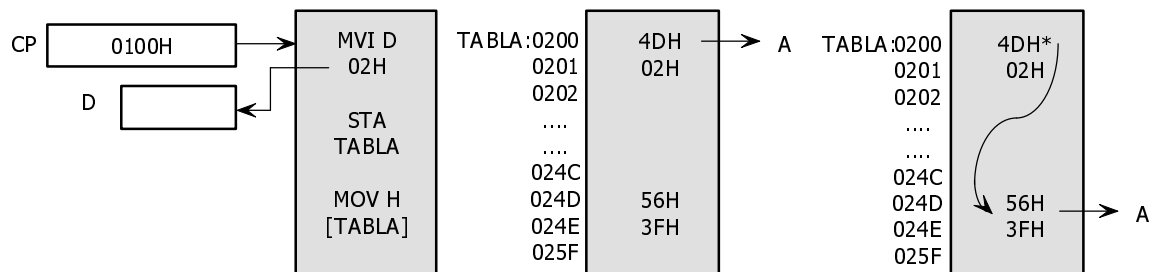
- Ejemplo: MVI D,02H

- Directo: La instrucción contiene la dirección del dato.

- Ejemplo: LDA TABLA

- Indirecto: La instrucción contiene una dirección de un puntero al dato.

- Ejemplo: MOV H,[TABLA] (No está implem. en el 8085).



# Modos de direccionamiento II

- Relativo: La instrucción contiene un *desplazamiento* (offset), que sumado al contador de programa da la *dirección efectiva*.

- Ejemplo: JNZ SEGUIR (No está implem. en el 8085).

- Indexado: La instrucción contiene un valor, que sumado al contenido de un registro índice de la CPU, da la *dirección efectiva*.

- Ejemplo: LDAA \$4C,X (Implementado en el motorola 68hc11).

- Por Registro:

- Directo* : El dato está contenido en un registro.

- Ejemplo: MOV D,C

- Indirecto* : El registro, o par de registros, contiene un puntero.

- Ejemplo: MOV A,M

