

# Técnicas Digitales II

## uProcesadores – 8085

2020

# uProcesador

- Breve repaso teórico

Que es un microprocesador?

Como esta conformado?

Que necesita para su funcionamiento?

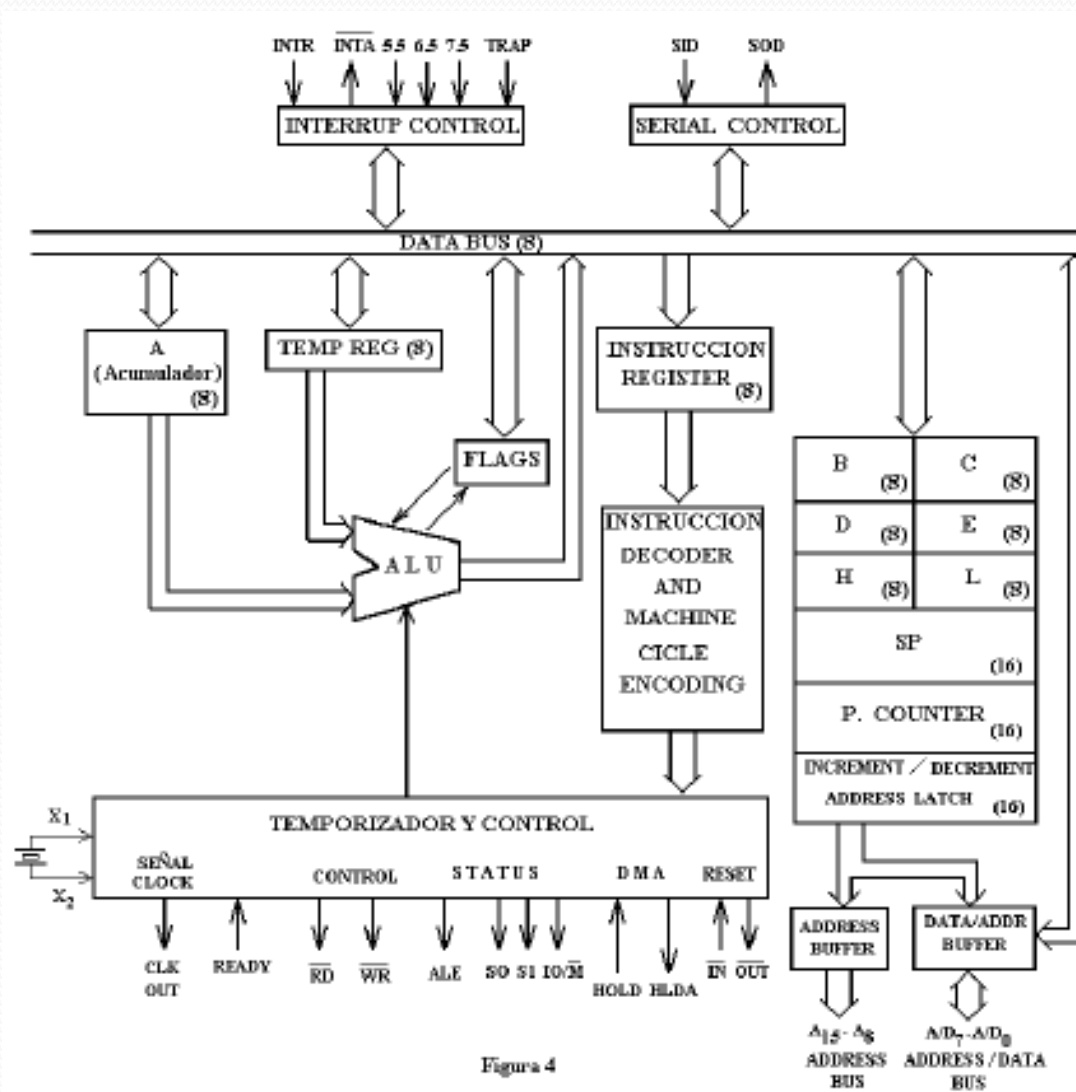



Figura 4


# Assembler

- Usualmente para la programación deben haber empleado C, Pascal, Basic, etc. Todos lenguajes de alto nivel.
- Las instrucciones de alto nivel deben ser traducidas por un compilador a las que sean comprensibles por el uP.
- Cada uP posee su propio Set de Instrucciones para la programación en bajo nivel.
- Un uP solo puede interpretar un reducido numero de instrucciones. Las mismas son operaciones básicas y primitivas, que permiten la realización de conjunto de operaciones mas complejas.
- Un determinado problema es capaz de ser resuelto mediante una multiplicidad de soluciones.

- 
- La cantidad de instrucciones que sean necesarias emplear depende en gran medida del conocimiento del manejo del set de instrucciones como de la creatividad del programador.
  - A pesar de la variedad de instrucciones que posee el set de cada uP, es normal ver la similitud entre las que utilizan las mismas familias, lo que permite usualmente una compatibilidad entre versiones o modelos de hardware.
  - Por otra parte, luego de aprender el mecanismo para la resolución de problemáticas con un set de instrucciones específico, resulta sencillo migrar a otros.
  - El uP que utilizamos es de 8 bits. Nos permite incorporar la lógica de funcionamiento y utilización del lenguaje de bajo nivel.

# Formato de datos

- El decodificador de instrucciones es capaz de interpretar determinada información (códigos de 8bits).
- Los datos que se leen o escriben en los registros son de 8 bits.
- La dirección de la posición de memoria o modulo entrada/salida se compone de 16 bits.
- Toda la información con que trabaja el uP son números binarios, que para mayor comodidad de manejo se expresan en hexadecimal.

- 
- Toda la información que maneja el uP posee una longitud de 8 bits. De ahí la denominación de este tipo de procesador. De esta manera, tendremos procesadores de 8 bits, de 16 bits, de 32 bits, etc.
  - La memoria esta organizada en posiciones de 1 byte cada una.
  - Los uPs de 8 bits sentaron las bases para el desarrollo del avance sobre la tecnología de 16 bits.

# Arquitectura

Registros de trabajo

Acumulador

Registro auxiliar

Registro de estado

Memoria programa/datos (ext)



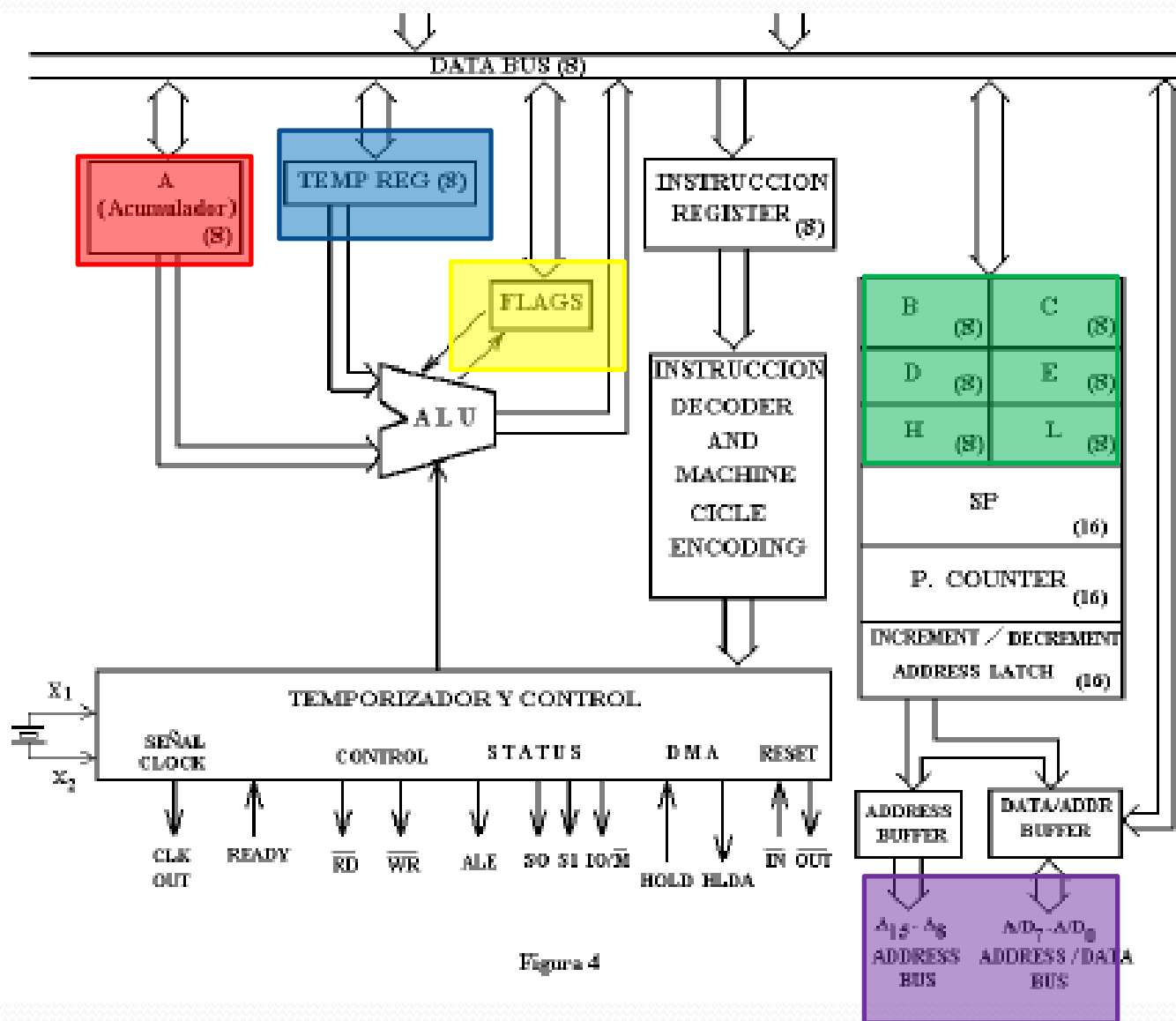


Figura 4

## Registros de trabajo

- Registros de trabajo son registros de 8 bits, B , C , D , E , H , L. Se pueden emplear como registros individuales o de a pares. Cada registro par consta de 16 bits y pueden ser empleados utilizados como puntero para direccionamiento en memoria.

## Acumulador

- Acumulador, es uno de los registros de 8 bits que posee uno de los operandos que manipula la ALU para realizar alguna operación. El resultado de las operaciones de la ALU se deposita en el mismo registro, convirtiendo al registro en fuente y destino.

## Registro auxiliar

- Registro auxiliar es el registro donde se almacena el segundo operando con el que trabaja la ALU al realizar cualquier operación.

## Registro de estado

- Registro de estado : es un registro de 8 bits, cuyo contenido indica si se cumple alguna condición al efectuarse una operación lógica. Los bits que actúan como indicadores o flags de condición son
  - Z (cero): se pone a 1 cuando el resultado de una operación ha sido cero.
  - P (paridad): se pone a 1 cuando el resultado de una operación tiene paridad par, o sea, la cantidad de unos (1) que posee el número es par.

## Registro de estado (cont)

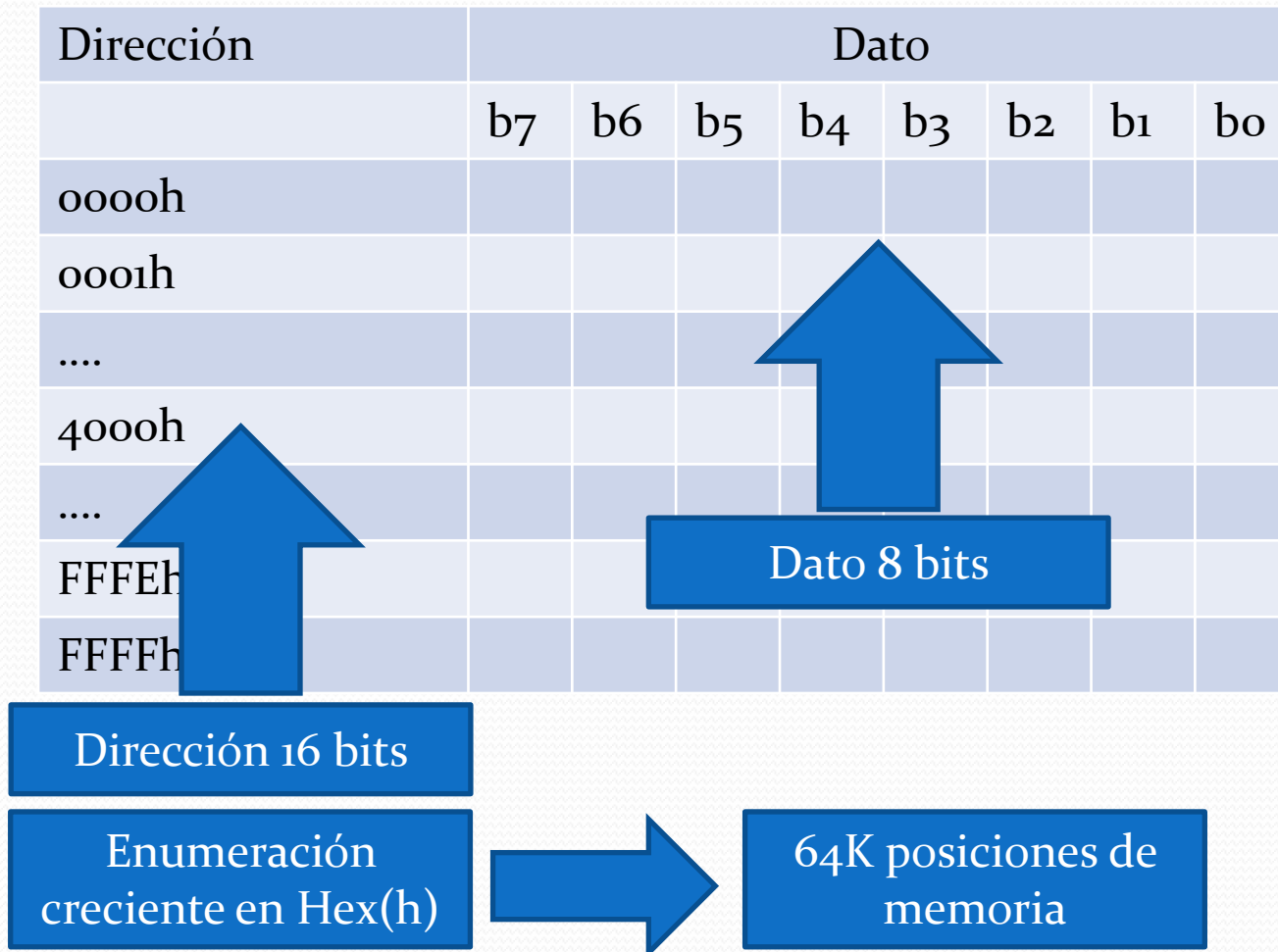
- Cy (acarreo): se pone a 1 cuando se produce acarreo en el bit mas significativo ya sea por una suma o por una resta.
- AC (acarreo auxiliar) acarreo del cuarto bit del numero al realizar una operación.
- S (signo): indica el signo del resultado de una operación aritmética llevada a cabo con números con signo, 1 si es negativo y 0 si es positivo.

## Estructura del registro de estado

S	Z	X	AC	X	P	X	CY
---	---	---	----	---	---	---	----

(bits marcados con X no definidos)

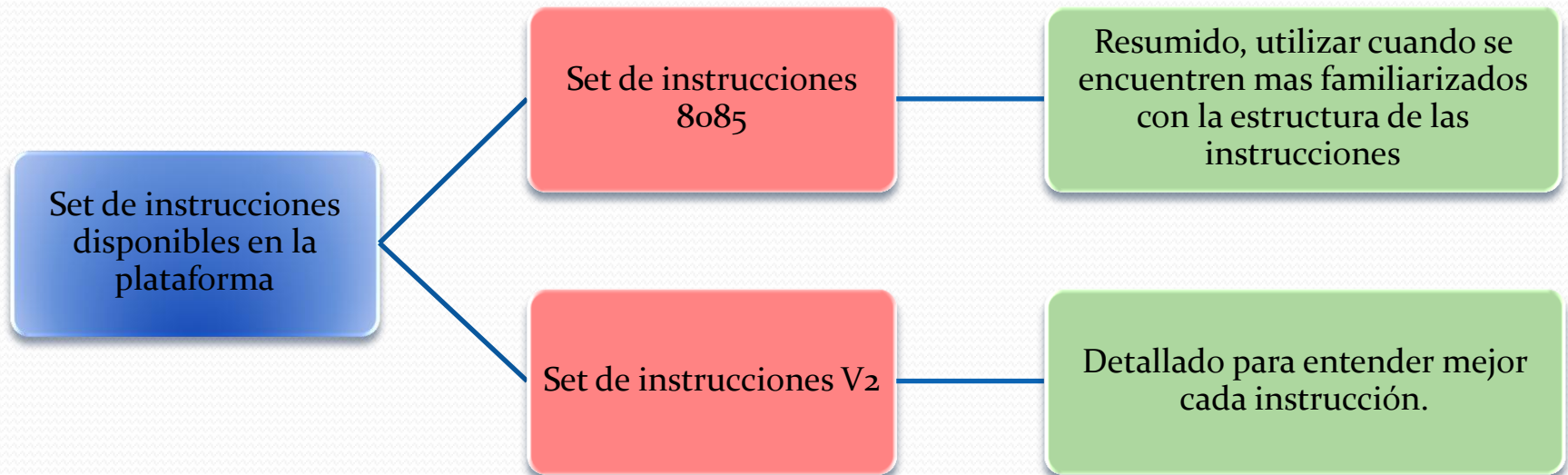
## Memoria programa/datos (ext)



# Set de instrucciones

- Si bien el lenguaje maquina es binario, suele emplearse el hexa de manera que 4 bits son representados por un dígito. Esto se debe a la complejidad de trabajar todo en binario, y para minimizar los errores de escritura.
- Un código escrito en hexa o en binario reviste cierta dificultad para su lectura de manera directa.
- Con el objeto de que la programación resulte entendible para la escritura y lectura, las instrucciones se encuentran representadas por abreviaturas de palabras, indicativas de la operación a la que hacen referencia, denominado Mnemónico.
- Por ejemplo MOV para realizar movimientos entre registros, ADD para la suma, SUB para la resta, etc.

# Instrucciones



# Set de instrucciones 8085

## 2 - INSTRUCCIONES LOGICAS Y ARITMETICAS

### SUMAS O ADICIONES

Sub grupo

Sume el Reg. B al Reg. A  
Sume el Reg. C al Reg. A  
Sume el Reg. D al Reg. A  
Sume el Reg. E al Reg. A  
Sume el Reg. H al Reg. A  
Sume el Reg. L al Reg. A

Descripción

Familia de instrucciones

Código de inst.

Mnemónico

Nro de Bytes

Ciclos de maquina

80	ADD B	1	4
81	ADD C	1	4
82	ADD D	1	4
83	ADD E	1	4
84	ADD H	1	4
85	ADD L	1	4



Descripción	Código	Nemotécnico	B*	C*
-------------	--------	-------------	----	----

## 1 - INSTRUCCIONES PARA TRANSFERENCIA DE DATOS

### MOVIMIENTOS AL REGISTRO B

Mueva el contenido del Reg. B al Reg. B	40	MOV B, B	1	4
Mueva el contenido del Reg. C al Reg. B	41	MOV B, C	1	4
Mueva el contenido del Reg. D al Reg. B	42	MOV B, D	1	4
Mueva el contenido del Reg. E al Reg. B	43	MOV B, E	1	4
Mueva el contenido del Reg. H al Reg. B	44	MOV B, H	1	4
Mueva el contenido del Reg. L al Reg. B	45	MOV B, L	1	4
Mueva el contenido de M al Reg. B*	46	MOV B, M	1	7
Mueva el contenido del Reg. A al Reg. B	47	MOV B, A	1	4

# Set de instrucciones V2

Mnemónico y representación

**MOV**      **r1,r2**  
**(r2) ← (r1)**

Descripción

El contenido del registro r2 es transferido al registro r1

Un ciclo de maquina consta de  
3 a 6 estados

Ciclos : 1

Estados : 4

Unidad básica de tiempo, es un  
ciclo de reloj, la mitad de la  
frecuencia del cristal

Direccionamiento : Registro

Modo de direccionamiento

Banderas:

Bits de estado que se modifican

# Conformación de código de operación

o	1	D	D	D	S	S	S
---	---	---	---	---	---	---	---

 DDD: destino    SSS: source o fuente

DDD/SSS	111	000	001	010	011	100	101	110
REGISTRO	A	B	C	D	E	H	L	M

Ejemplo:

Mueva el contenido del Reg. B al Reg. A

78

MOV A, B

MOV A,B

$(A) \leftarrow (B)$

A = 1 1 1

B = 0 0 0

Código de operación:

0	1	1	1	1	0	0	0
---	---	---	---	---	---	---	---

=

78h

# Conformación de instrucciones

**Mnemónico** + registro/dato/dirección + registro/dato/dirección

Mueva el contenido del Reg. B al Reg. C	48	MOV C, B
Mueva data al Registro A	3E	MVI A, data
Cargue los registros B y C con data16	01	LXI B, data16
Cargue el Reg. A con el contenido de addr	3A	LDA addr
Cargue Reg. A con el contenido de la posición de memoria definida por D y E	1A	LDAX D
Rotar el Registro A a la derecha	0F	RRC
Sume data al registro A	C6	ADI data

Registro A : A, acumulador

Sume el Reg. A y el acarreo al Reg. A

8E    ADC A

Registros de trabajo : B , C , D , E , H , L (representado por r , r1 o r2)

Mueva el contenido del Reg. B al Reg. L

68    MOV L, B

Mueva el contenido del Reg. C al Reg. L

69    MOV L, C

Mueva el contenido del Reg. D al Reg. L

6A    MOV L, D

Mueva el contenido del Reg. E al Reg. L

6B    MOV L, E

Mueva el contenido del Reg. H al Reg. L

6C    MOV L, H

Mueva el contenido del Reg. L al Reg. L

6D    MOV L, L

Registro par (rp) : B representa el par BC

D representa el par DE

H representa el par HL

Cargue los registros B y C con data16

01    LXI B, data16

Cargue los registros D y E con data16

11    LXI D, data16

Cargue los registros H y L con data16

21    LXI H, data16

## Memoria direccionada por par de registros HL : M

Mueva data a la memoria\*

35

MVI M, data

Dirección de 16 bits : **addr (dir)**

B010	LDA, 1020	3A
B011		20
B012		10

Dato de 8 bits: **data,8 o dato o byte**

400A	ADI oA	C6
400B		oA



Dato de 16 bits: **Data, 16**

Dirección de 16 bit de una subrutina: LABEL

Primer registro de un registro par (mayor orden): rh

Segundo registro de un registro par (mayor orden): rl

Segundo byte de una instrucción: byte 2

Tercer byte de una instrucción: byte 3

Contenido de un registro : ( ) - paréntesis

Referencia a memoria por contenido de un rp : [ ] - corchete

Las instrucciones del 8085 pueden tener 1, 2 o 3 Bytes.

Instrucción de 1 Byte

$(A) \leftarrow (A) + (B)$

400A	ADD B	80
------	-------	----

Instrucción de 2 Bytes

$(A) \leftarrow (A) + (\text{byte } 2)$

400A	ADI dato	C6
400B		dato

Instrucción de 3 Bytes

$[(\text{byte}_3)(\text{byte}_2)] \leftarrow (A)$

400A	STA, adres	32
400B		byte2
400C		byte3



Para la realización del código, emplear la denominación «h» para referenciar a números expresados en hexadecimal, «d» para decimal, «b» para binario.

Ejemplo

400A	SUI 10	D6
------	--------	----

El 10 podría ser tomado como binario, como hex, como decimal, cambiando el significado del programa.

En el simulador, se toma como hex. Pero para los prácticos o parcial debe ser aclarado en que base se encuentran.

400A	SUI 10d	D6
------	---------	----

400A	SUI 10h	D6
------	---------	----

400A	SUI 10b	D6
------	---------	----

# Representación grafica de instrucción

Usualmente empleada para diagrama de flujo.

El resultado de las operaciones o destino van a la izquierda, y la fuente u origen a la derecha. Por ejemplo.

Movimiento de contenido de un registro a otro

$$(r1) \leftarrow (r2)$$

$$(\text{destino}) \leftarrow (\text{fuente})$$

Suma del Acumulador con un valor almacenado en Memoria

$$(A) \leftarrow (A) + [(H)(L)]$$

$(\text{cont. Reg A}) \leftarrow (\text{cont Reg A}) + [\text{valor almacenado en la posición de memoria cuya dirección esta indicada por H-L}]$

# Herramientas para la programación


```
graph TD; A[Herramientas para la programación] --> B[Diagrama de flujo]; A --> C[Código en lenguaje assembler];
```


Diagrama de  
flujo

Código en  
lenguaje  
assembler

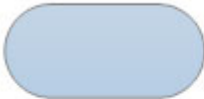

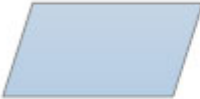
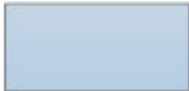

# Diagrama de flujo

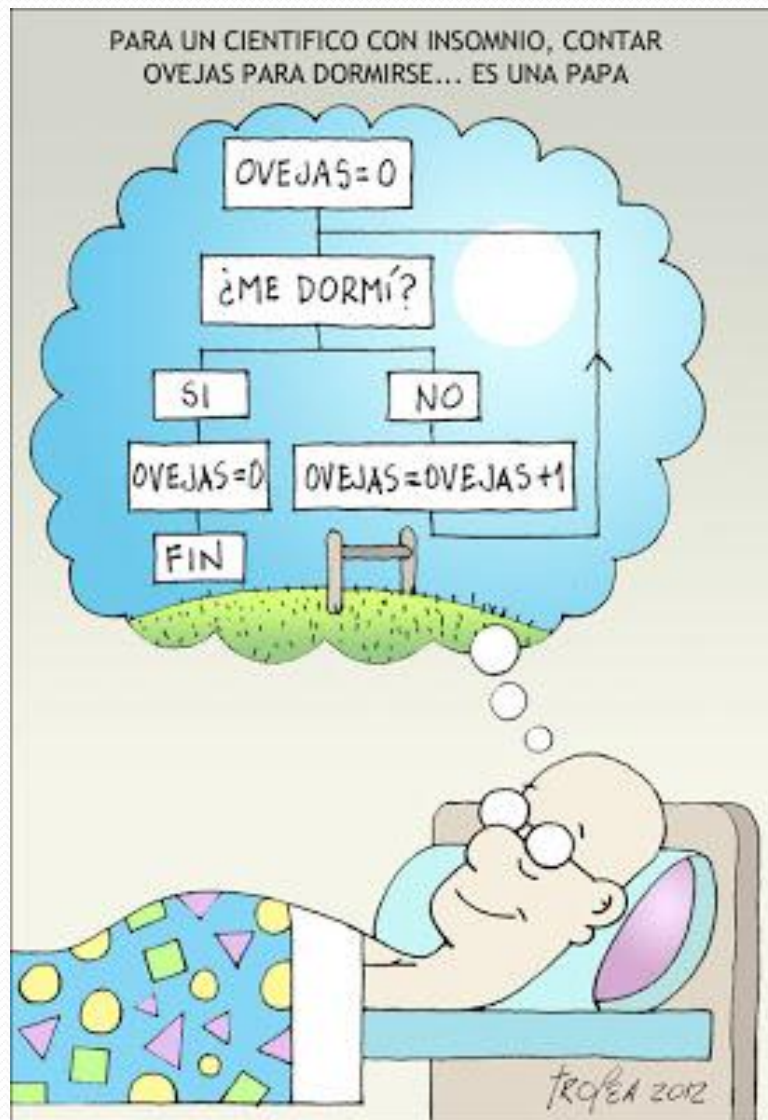
- Que es?
  - Representación grafica del algoritmo o proceso.
  - Lenguaje secuencial, estructura lineal.
  - Estimula el pensamiento analítico.
  - Sentencias en diagrama traducibles a código ( de manera directa)

- 
- Facilita la obtención de una visión transparente del código.
  - Permite definir los límites del mismo.
  - Proporciona un método de comunicación más eficaz.
  - Referencia para establecer mecanismos de control.
  - Mejora tiempos.
  - Dar una estructura a las ideas para poder desarrollar un programa.

- 
- Pensar un problema en un nivel mayor para poder llevarlo a código.
  - Definir una situación problemática.
  - Planificar los pasos lógicos para su resolución mediante código.
  - Modelar un proceso
  - Facilita la rápida comprensión de cada etapa o función, y su relación con las demás.
  - Permite la fácil detección de errores e inconsistencias al alcanzar una visión general del sistema

# Diagrama de flujo – Simbología básica

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Linea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso





# Código en lenguaje assembler

- Código realizado en lenguaje de bajo nivel, estructurado acorde a la representación secuencial para la resolución de las situaciones problemática.
- Contempla el código de operación empleado para la carga directa a memoria, para su implementación.
- Cuadro conteniendo las siguientes columnas: Label, Posición de memoria, Mnemónico y código de operación.

Label	Dirección	Instrucción Mnemónico	OpCode
	8000	LDA, oAoo	3A (código instr.)
	8001		oo (parte baja dir)
	8002		oA (parte alta dir)
	8003	LXI H, oAo1	21
	8004		o1
	8005		oA
	8006	ADD M	86
	8007	STA, oAo2	32
	8008		o2
	8009		oA
	800A	HLT	76

- Label. Es el rótulo utilizado para denotar la dirección a la cual se realiza un salto (condicional o incondicional).
- Dirección. El programa debe iniciar en algún segmento de memoria, definido por el programador. El contador de programa(PC) será iniciado en este punto.
- Mnemónico. Es el programa desarrollado empleando el set de instrucciones.
- Código OP. Cada instrucción posee un código único. El numero de bytes de memoria ocupados dependerá del tipo de instrucción (instrucciones de 1, 2 o 3 bytes).