Introducción a los µprocesadores I

- Descripción: Dispositivo integrado digital, programable y de actuación secuencial.

- ¿ Donde encontramos un µprocesador ?:
  - Encima de la mesa: Unidad Central de Proceso (CPU).
  - Horno microondas.
  - Sistemas de control de automóviles.
  - Teléfono celular.
  - Hasta en Marte!

Introducción a los µprocesadores II

- Verdades sobre un µprocesador:
  - Sólo entiende lenguaje binario. Lenguaje máquina.
    BIT: Unidad mínima de información.
    Palabra: Conjunto de bits que codifican una información.
    Byte: Palabra de 8 bits.
    Nibble: Palabra de 4 bits.
  - Sólo hace lo que le decimos.
  - Sin embargo: Las órdenes están codificadas en binario.
  - Todo lo que un µprocesador sabe está almacenado en memoria o es proporcionado por un dispositivo periférico.

Evolución histórica: ENIAC

- Evolución histórica en los µprocesadores:
  - 1944: J. Presper Eckert y John Mauchly desarrollan ENIAC.
    - Financiada por la armada de EEUU.
    - 80 x 8.5 pies de largo.
    - Programación manual (cables y conmutadores)
    - Datos: Tarjetas perforadas.
    - Tiempo de programación: media hora / un día.
    - 30 Toneladas de peso.
    - 150 Kwatt. De consumo.

Modelo de Von Neumann

- 1944: John Von Neumann entra en el proyecto ENIAC.
  - Objetivo: Mejorar la forma de introducir los programas.
  - Propuesta: Computador de programa almacenado EDVAC (Electronic Discrete Variable Automatic Computer).

Von Neumann y Herman Goldstine

Eckert y Mauchly

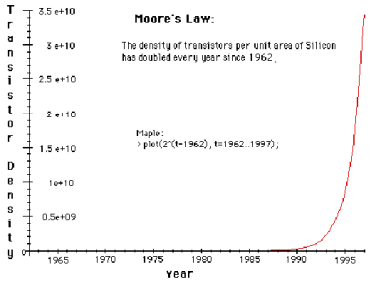
- 1951: UNIVAC I. Primer computador comercial de propósito general. 1 millón de dólares.

Evolución histórica II

- 1947: Bardeen/Brattain/Shockley inventan el transistor.
- 1959: Robert Noyce/Jack Kirby inventan el circuito integrado.
- 1964: IBM introduce el IBM 360.
  - Creó la idea de máquina compatible (todavía en uso hoy en día).
- 1965: Gordon Moore estimó que el número de transistores/chip se duplicaba cada 18 meses.
- 1971: Intel introduce el 4004, primer µprocesador monopastilla.
- 1974: Intel introduce el 8008. Bill Gate estudia segundo curso en la Universidad de Harvard.
- 1977: Steve Jobs y Steve Wozniak sacan el *apple II.*
- 1981: IBM + Intel revolucionan el mundo de los computadores con el 80x86, utilizando el MS-DOS.

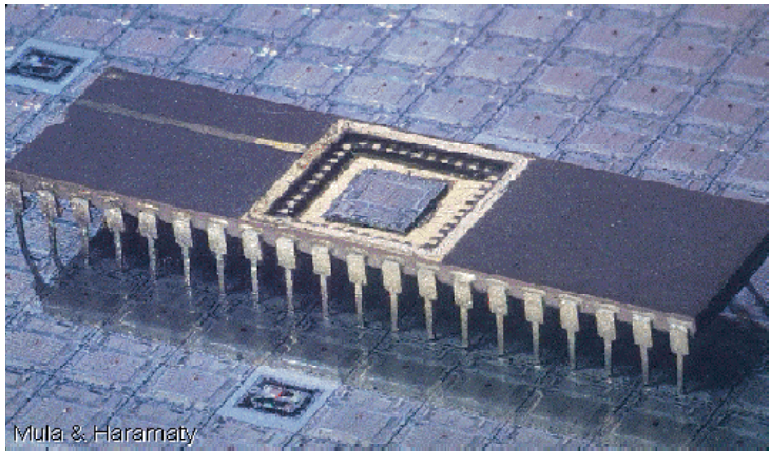
Ley de Moore

- Una interesante tendencia:
  - Las prestaciones son proporcionales al número de transistores.
  - Se ha mantenido durante 20 años.
  - Crecimiento exponencial.



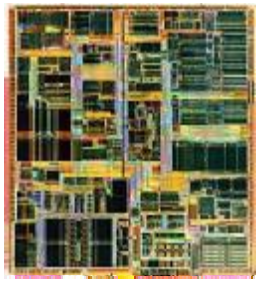
IBM System 360

Aspecto de un µprocesador



Aumento del nivel de Integración

Intel 4004 (1971):
- Número de Transistores: 2.300
- Frecuencia : 0.5 MHz
- Dimensiones: 3mm x 4mm



Intel Pentium II (1997):
- Número de Transistores: ≅ 7.5M
- Frecuencia: 450MHz
- Dimensiones: 12.8cm x 6.3cm

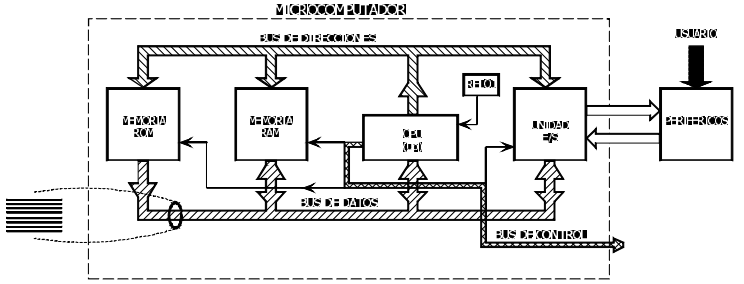
Computadores modernos

Apple II; Características:
- Tipo de almacenamiento:
  Cinta Magnética.
- Memoria interna: Pocos Kbytes.
- Monitor: Alfanumérico Monocromo.

Pentium Pro; Características:
- Tipo de almacenamiento:
  Disco Magnético, FDD,CDROM.
- Memoria interna: 256 Mbytes.
- Monitor: 21" de Alta resolución y bajo consumo.

Sistema basado en µprocesador

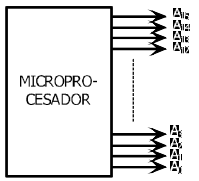
- Sistema µcomputador:

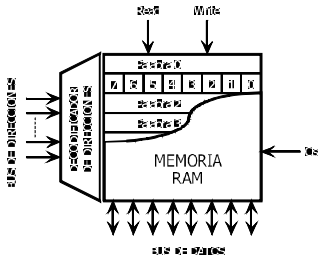


- Sistema de buses organizados. Interconexión de elementos a través de los buses.
- Información binaria transferida a través de los buses.

Sistema de buses organizados I

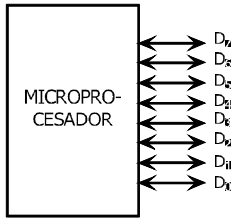
- Bus de direcciones:
  - Contiene la *dirección* (localización del dato en memoria) entre el µprocesador y la memoria. Unidireccional.
  - Ancho típico de 16 bits -> 2^{16} (65.536) posiciones de memoria. Cada número se refiere a una posición de memoria.
  - Cada posición de memoria almacena un byte (8 bits) o una palabra (16 bits).





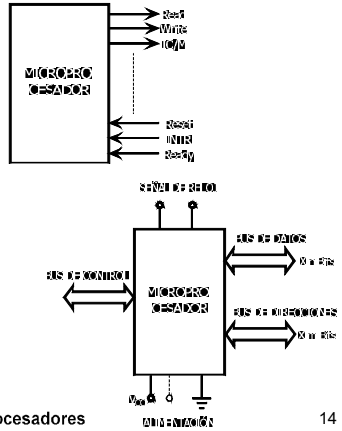
Sistema de buses organizados II

- Bus de datos:
  - Transfiere la información, en binario, entre el µprocesador y otras unidades externas (memoria o E/S). Bidireccional.
  - Tamaño típico: 8 ó 16 bits.
  - Tamaño relacionado con el ancho de palabra en la memoria.
  - Relacionado directamente con las prestaciones del sistema.



Sistema de buses organizados III

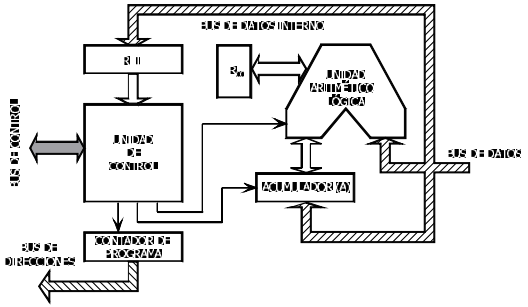
- Bus de Control:
  - Contiene las *señales de control* específicas para controlar y coordinar las operaciones del µprocesador.
  - Ejemplo: Señal Read.
    - Ejecuta un ciclo de lectura.
  - Señal Write.
    - Ejecuta un ciclo de escritura.
  - Señal simple Read/$\overline{\text{Write}}$.
  - Otras señales: Interrupciones, Reset, De reloj, etc…
- Esquema general:
  - Señales de reloj.
  - Alimentación.



Arquitectura Interna I

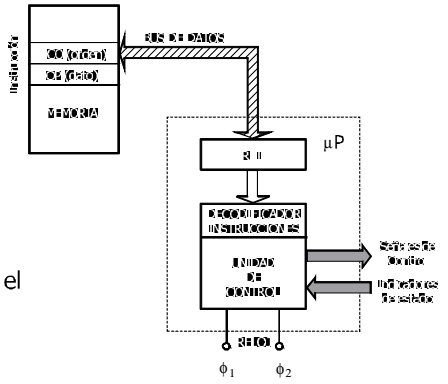
- Ruta de Datos.
  - Transferencia, memorización y procesamiento de la información.
  - Tipos de información: Datos, Instrucciones y Direcciones.
  - Ejemplo:

Instrucción	LDAA 3200	
1º Byte	CO	LDAA
2º Byte	A_H	32
3º Byte	A_L	00



Arquitectura Interna II

- Unidad de Control:
  - Interpreta las instrucciones y genera las señales de control (µórdenes).
  - Pasos a realizar:
    - 1.- Lectura del CO (opcode).
    - 2.- Generación de las señales de control.
  - Gestiona la aceptación de comandos introducidos a través del bus de Control.
  - Otras operaciones: Incrementar el contador de programa.



Secuencia Operativa: Ejemplo

- Desarrollo de una secuencia operativa dentro del µprocesador:
  - Sumar dos números *X* e *Y* dejando el resultado en *Z*.
  - *X*, *Y* y *Z* corresponden en el programa con las siguientes direcciones de memoria:
    - X = 0011H
    - Y = 0012H
    - Z = 000EH
  - Ciclo de *Fetch*.
  - Ciclo de *Ejecución*.

DIRECCIÓN	MEMORIA	
0000	CARGAR	CO 1ª Instrucción
0001	11	Byte bajo de la dirección
0002	00	Byte alto de la dirección
0003	SUMA	CO 2ª Instrucción
0004	12	Dirección del 2º dato
0005	00	
0006	ALMACENA	CO 3ª Instrucción
0007	0E	
0008	00	Dirección del resultado
0009	STOP	CO 4ª Instrucción
000A		
000B		
000C		
000D		
000E	Z	Resultado
000F		
0010		
0011	X	1er Sumando
0012	Y	2º Sumando

Características de los µprocesadores

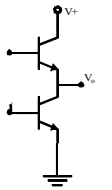
- Características Hardware:

µprocesador	Bus de Datos	Ciclo de Inst.	Direcciones	Fabricante
8080	8 bits	1.5 ns.	64 K	Intel
8085	8 bits	0.8 ns.	64 K	Intel
2650	8 bits	1.5 ns.	32 K	Signetics
Z-80	8 bits	1 ns.	64 K	Zilog
8086	16 bits	0.4 ns.	1 M	Intel
68000	16 bits	0.5 ns.	16 M	Motorola

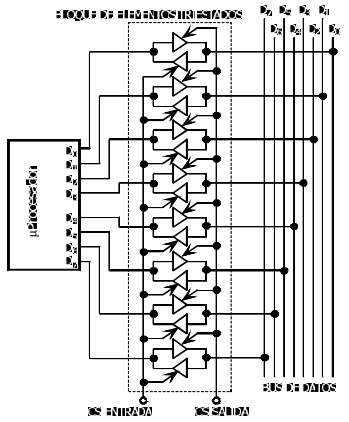
  - Capacidad de Interrupción: Números y niveles de interrupción.
  - Familia de periféricos.
- Características Software:
  - Juego de Instrucciones.
  - Optimización del juego de instrucciones.

El tercer estado

- Estructura de buses compartidos: El tercer estado.
  - Desconexión total de los bloques no implicados en la transferencia.
  - Uso de *buffers* triestado.

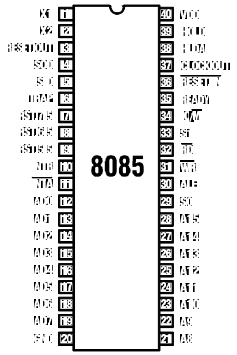


CS	In	Out
1	0	1
1	1	0
0	X	DESCONEXIÓN



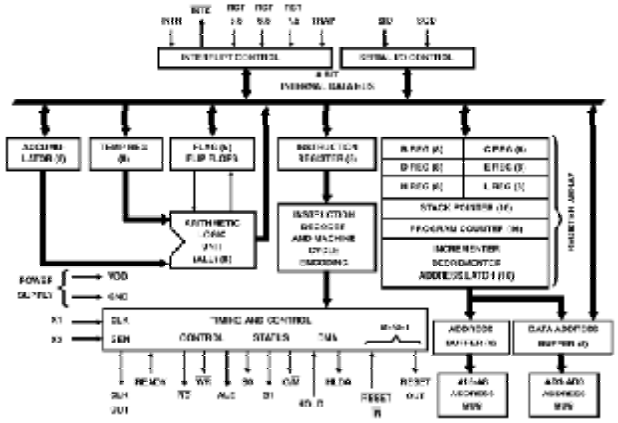
Microprocesador 8085 de Intel

- Introducción:
  - Microprocesador de 8 bits (1973).
  - Reloj oscilador interno. Sólo precisa un oscilador de cuarzo externo.
  - Alimentación única de 5 Voltios.
  - Posee 74 instrucciones.
  - 4 entradas de interrupciones vectorizadas.
  - Encapsulado DIL de 40 pines.



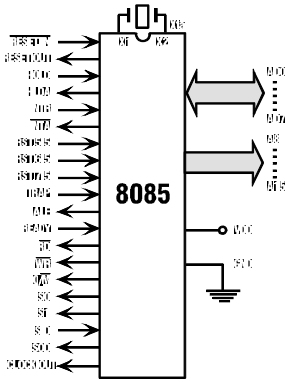
Estructura interna del 8085

- Diagrama de bloques:

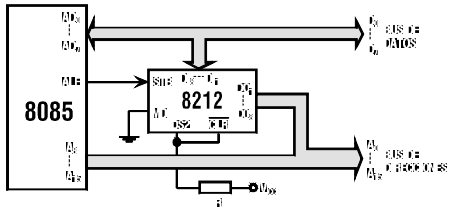


Buses y líneas de control del 8085

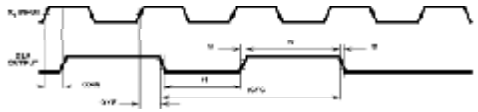
- Arquitectura externa:



- Demultiplexado de datos y direcciones:

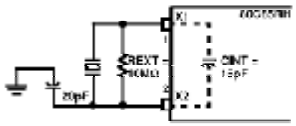


- Señal de reloj:

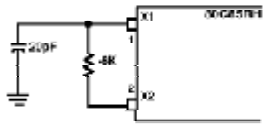


Circuitos de reloj para el 8085

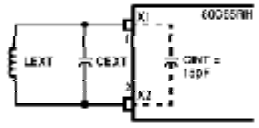
- Los circuitos utilizados pueden ser:



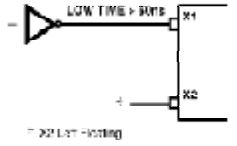
a.) QUARTZ-CRYSTAL CLOCK DRIVER



c.) RC-CIRCUIT CLOCK DRIVER



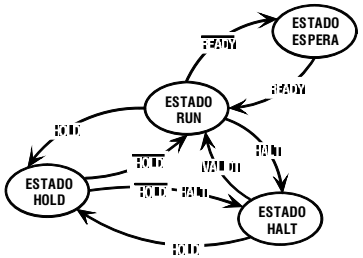
b.) LC-TUNED CIRCUIT CLOCK DRIVER



d.) 0-4MHz INPUT FREQUENCY EXTERNAL CLOCK DRIVER CIRCUIT

Estados y ciclos de máquina del 8085

- Diagrama de estados:



- Ciclos de Máquina:

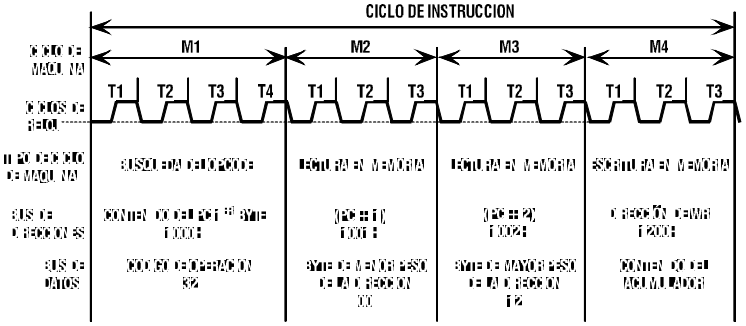
Ciclo de Instrucción → Ciclos de Máquina → Ciclos de Reloj

MACHINE CYCLE			STATUS IO/M	STATUS S1	STATUS S0	CONTROL RD	CONTROL WR	CONTROL INTA
Opcode Fetch (OF)			0	1	1	0	1	1
Memory Read (MR)			0	1	0	0	1	1
Memory Write (MW)			0	0	1	1	0	1
I/O Read	(IOR)		1	1	0	0	1	1
I/O Write	(IOW)		1	0	1	1	0	1
Acknowledge of INTR	(INA)		1	1	1	1	1	0
Bus Idle	(BI)	DAD ADR S	0	1	0	1	1	1
		RST, TRAP	1	1	1	1	1	1
		HALT	TS	0	0	TS	TS	1

Diagrama de tiempos y secuenciamiento

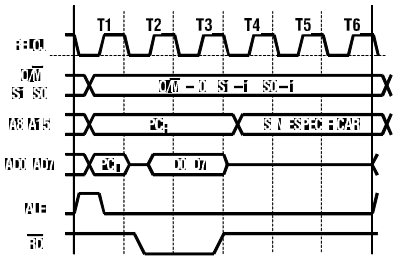
- Ejemplo de ejecución de la instrucción:

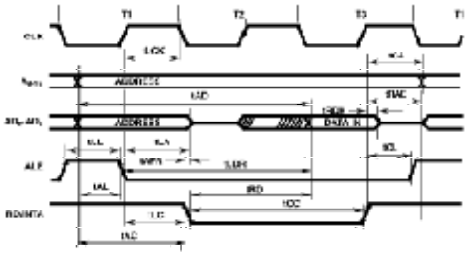
1000H: STA 1200H (almacenada en la posición 1000H)



Ciclos de Fetch y Read

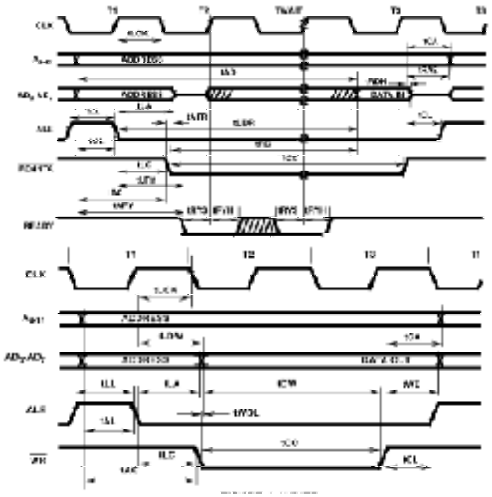
- Ciclo de Fetch:
  - 4 ó 6 ciclos de reloj.
  - T1: Demultiplexación Data/Addr.
  - T3: OpCode al Registro de Instrucción (final del ciclo).
  - T4: Decodificación de la Instrucc.
- Ciclo de Read:
  - 3 ciclos de reloj.
  - T1: Demultiplexación Data/Addr.
  - T3: El dato está disponible en el bus.





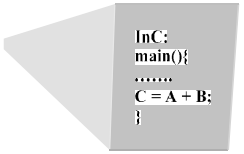
Ciclo de Escritura

- Ciclo de Read (Not Ready):
  - Señal READY a nivel bajo. Indica Memoria o I/O no preparado.
  - El µp inserta ciclos de espera: TWAIT.
- Ciclo de Write:
  - 3 ciclos de reloj.
  - T1: Demultiplexación Data/Addr.
  - T2 y T3: El dato está presente en el bus.



Introducción al software en los µprocesadores

- 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).



- 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_2 Sumar el contenido de dos registros (Add).
  - 00000010_2 Sumar el contenido del acumulador con el contenido de una posición de memoria (Add).
  - 00000101_2 Idem pero restando los operandos (Substract)
  - 00100011_2 Cargar un registro desde memoria (Load Acumulator).
  - 00101011_2 Almacenar un registro en memoria (Store Acumulator).

Estas codificaciones corresponden a los "*OpCodes*".

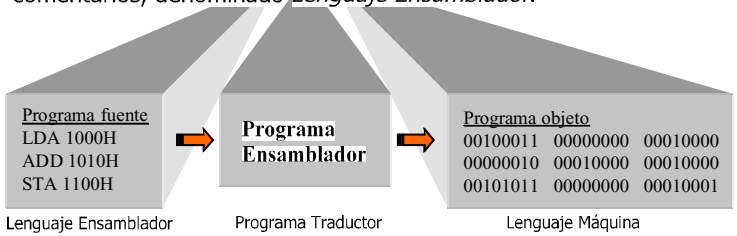
Estas codificaciones corresponden a los Nemónicos.

- 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

Mon May 10 1999 19:51

Defined Symbol Name     Value     References

Pre     CODE            0000
Pre     DATA            0000
  3     xdir        =   044E        8
  1     ydir        =   1100        5
  2     zdir        =   1101        6

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
---------------------------------------------

                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                    org   1000H
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
```

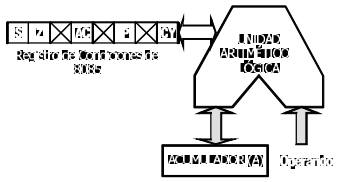
```
; Programa para que ustedes aprendan

          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:

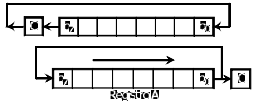
```
ADD B
SBB  M
CPI  55H
```



- **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 3º bit al 4º bit.**

Clasificación de las instrucciones II

- Lógicas: Efectúan operaciones lógicas, tales como AND, OR, XOR, NAND, NEGACIÓN, etc…
  - Ejemplos: ANA D
ORA M
CMA
- 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
- De desplazamiento y rotación: Desplazan o rotan el contenido del acumulador.
  - Ejemplos: RAL
RRC

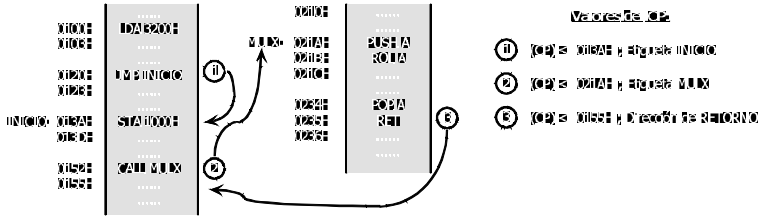


Clasificación de las instrucciones II

- 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
- 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

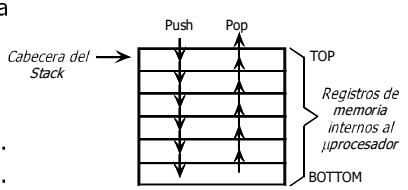
- Instrucciones de salto (o de ruptura de secuencia):
  - El uso de subrutinas es una técnica muy importante en el diseño de software para sistemas con µprocesadores.
  - Ejemplo de programa:



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

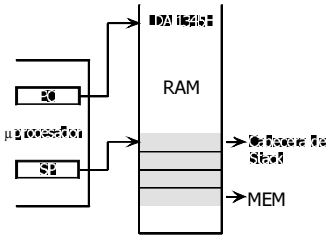
Operaciones sobre la PILA

- 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 µprocesador 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) -> (SP);
        - (SP) -1 -> SP;
    - Instrucción POP R:
      - Transfiere el contenido de la dirección de memoria apuntada por el *Stack Pointer* al registro R.
        - ((SP) +1) -> R;
        - (SP) +1 -> 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.
      - (PCl) -> (SP);
      - (PCh) -> (SP) -1;
      - (SP) -2 -> SP;
      - byte2 -> PCl;
      - byte3 -> PCh;

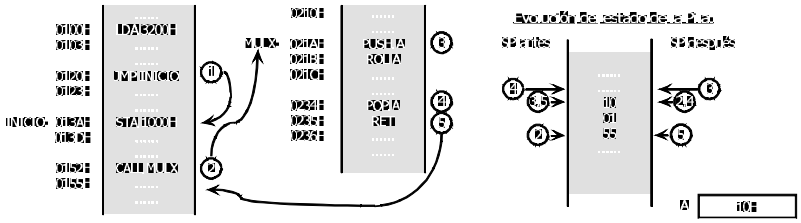
OpCode	Byte 2	Byte 3

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) -> PCl;
      - (SP) +2 -> SP;

Ejemplo de manejo de la PILA

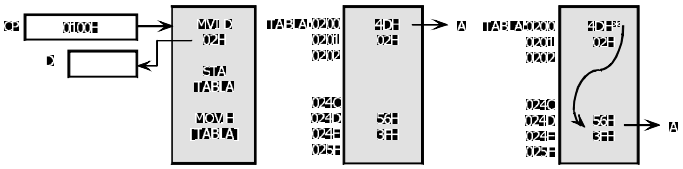
- Ejemplo de manejo de la pila:



- Posibilidad de anidamiento de subrutinas.
- Llamadas y retornos condicionales a los bits del registro de estado.

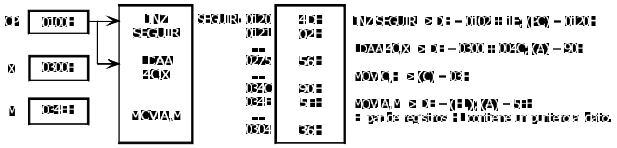
Modos de direccionamiento I

- 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



Introducción al Software del 8085

- **Directivas:**
  - **ORG: (Origen Absoluto)**
    - **Sintaxis:** *ORG <dir>*
    - **Las sentencias posteriores son ubicadas a partir de la dirección especificada.**
    - **Ejemplo:** **ORG 1000H**
  - **END: (Fin del módulo)**
    - **Indica al ensamblador que el programa fuente ha terminado.**
  - **EQU: (Igualar símbolo con valor)**
    - **Sintaxis:** *ETIQUETA EQU VALOR*
    - **Asigna un valor al símbolo del campo etiqueta. Este valor no puede modificarse durante el programa.**
    - **Ejemplo:** **PI EQU 32**

Introducción al Software del 8085

- **Pseudoinstrucciones:**
  - **DB: Definir byte**
    - **Sintaxis:** *ETIQUETA DB VALOR*
    - **Inicializa una posición de memoria con un valor especificado en el campo VALOR.**
    - **Ejemplo:** **DATO DB 10H**
  - **DS: Reservar memoria**
    - **Sintaxis:** *ETIQETA DS VALOR*
    - **Reserva posiciones de memoria sin inicializar. El número de posiciones reservadas es igual a VALOR.**
    - **Ejemplo:** **ZONA DS 10H**
  - **DC: Define una constante**
    - **Sintaxis:** *ETIQUETA DC DATO*
    - **Inicializa una posición de memoria con el valor ASCII de DATO.**
    - **Ejemplo:** **ASCII DC 'ABC'**

Introducción al Software del 8085

- **Modos de direccionamiento I:**
  - **Inmediato:**
    - **El valor del operando está especificado en la propia instrucción.**
    - **Ejemplo: MVI B,02H**
  - **Directo:**
    - **El operando viene especificado por la dirección de memoria donde se halla almacenado. La instrucción contiene la dirección de memoria.**
    - **Ejemplo: LDA 1800H**
  - **Por registro:**
    - **La instrucción contiene el registro o pareja de registros donde se halla el operando.**
    - **Ejemplo: ADD B**

Introducción al Software del 8085

- **Modos de direccionamiento II:**
  - **Por par de registros indirecto:**
    - **1.- La dirección del operando viene determinada por una pareja de registros codificados en la instrucción (B-C, D-E).**
      - **Ejemplo: LDAX B**
    - **2.- Existe un caso particular cuando la pareja de registros es H-L. En este caso se utiliza el nemónico "M" para referenciar dicho direccionamiento.**
      - **Ejemplo: MOV A,M**

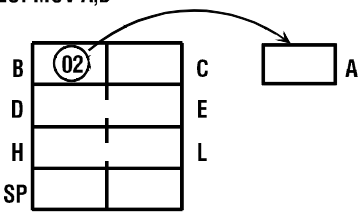
Introducción al Software del 8085

- Clasificación de las instrucciones:
  - Instrucciones de manipulación y transferencia.
  - Instrucciones aritméticas y lógicas.
  - Instrucciones de ruptura de secuencia.
    - Saltos condicionales e incondicionales.
    - Instrucciones de llamada y retorno de subrutinas.
  - Instrucciones de control:
    - Manejo de la pila.
    - De entrada y salida.
    - Interrupciones.

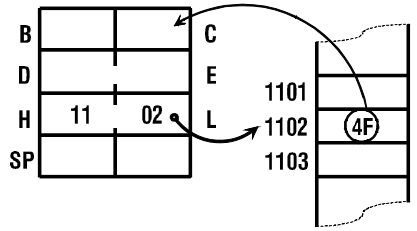
Instrucciones de transferencia I

	(r2) → r1	MOV r,M	[(H)(L)] → r
Formato:	0 1 D D D S S S	Formato:	0 1 D D D 1 1 0
Direccionam.:	Por registro	Direccionam.:	Par de reg. Indirecto
Señalizadores:	Ninguno	Señalizadores:	Ninguno
Ciclos M.: 1	Estados: 4	Ciclos M.: 2	Estados: 7

EJEMPLO: MOV A,B



EJEMPLO: MOV C,M



Instrucciones de transferencia II

(r) → (H)(L)

Formato: 0 1 1 1 0 S S S

Direccionam.: Par de reg. Indirecto

Señalizadores: Ninguno

Ciclos M.: 2 Estados: 7

MVI r,byte **2º Byte → r**

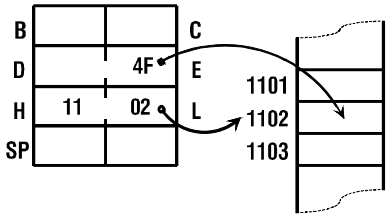
Formato: 0 0 D D D 1 1 0 | Operando

Direccionam.: Inmediato

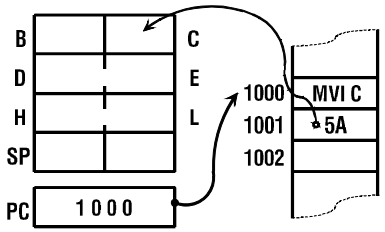
Señalizadores: Ninguno

Ciclos M.: 2 Estados: 7

EJEMPLO: MOV M,E



EJEMPLO: MVI C,5AH



Instrucciones de transferencia III

2º Byte → (H)(L)

Formato: 0 0 1 1 0 1 1 0 | Operando

Direccionam.: Inmediato/P. Reg. Indirec.

Señalizadores: Ninguno

Ciclos M.: 3 Estados: 10

LXI rp,dble **2º byte → (r_a) 3ºr byte → (r_b)**

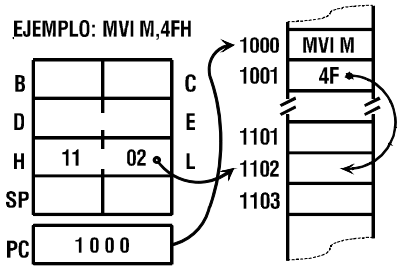
Formato: 0 1 1 1 0 S S S | 2º Byte | 3ºr Byte

Direccionam.: Inmediato

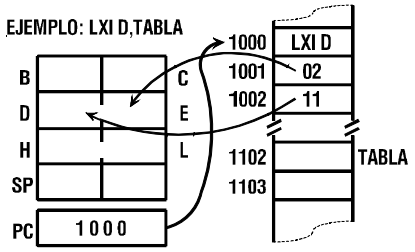
Señalizadores: Ninguno

Ciclos M.: 3 Estados: 10

EJEMPLO: MVI M,4FH



EJEMPLO: LXI D,TABLA



Instrucciones de transferencia IV

[(3ºr Byte)(2ºByte)] → A

Formato: 0 0 1 1 1 0 1 0 | 2º Byte | 3ºr Byte

Direccionam.: Directo

Señalizadores: Ninguno

Ciclos M.: 4 Estados: 13

STA addr **(A) → (3ºr Byte)(2º Byte)**

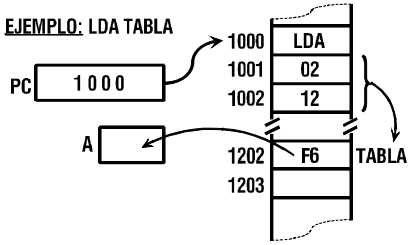
Formato: 0 0 1 1 0 0 1 0 | 2º Byte | 3ºr Byte

Direccionam.: Directo

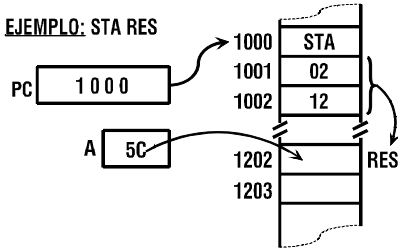
Señalizadores: Ninguno

Ciclos M.: 4 Estados: 13

EJEMPLO: LDA TABLA



EJEMPLO: STA RES



Instrucciones de transferencia V

LHLD addr **[(3ºr Byte)(2ºByte)] → L [(3ºr Byte)(2ºByte)+1] → H**

Formato: 0 0 1 0 1 0 1 0 | 2º Byte | 3ºr Byte

Direccionam.: Directo

Señalizadores: Ninguno

Ciclos M.: 5 Estados: 16

SHLD addr **(L) → (3ºr Byte)(2º Byte) (H) → (3ºr Byte)(2º Byte)+1**

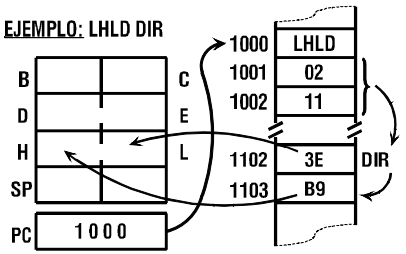
Formato: 0 0 1 0 0 0 1 0 | 2º Byte | 3ºr Byte

Direccionam.: Directo

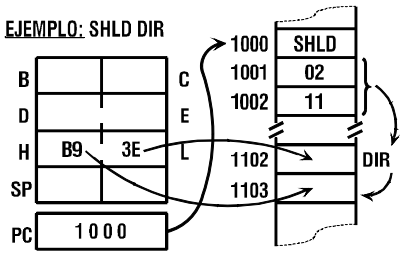
Señalizadores: Ninguno

Ciclos M.: 5 Estados: 16

EJEMPLO: LHLD DIR



EJEMPLO: SHLD DIR



Instrucciones de transferencia VI

LDAX rp [(rp)] → A

Formato: 0 0 R P 1 0 1 0

Direccionam.: Par de reg. Indirecto

Señalizadores: Ninguno

Ciclos M.: 2 Estados: 7

STAX rp (A) → (rp)

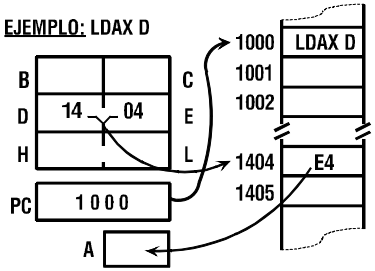
Formato: 0 0 R P 0 0 1 0

Direccionam.: Par de reg. Indirecto

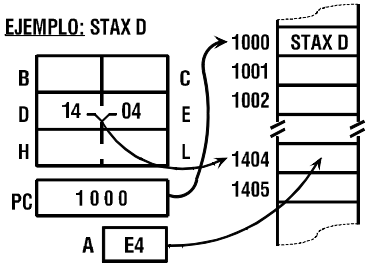
Señalizadores: Ninguno

Ciclos M.: 2 Estados: 7

EJEMPLO: LDAX D



EJEMPLO: STAX D



Instrucciones de transferencia VI

XCHG (H) ↔ (D), (L) ↔ (E)

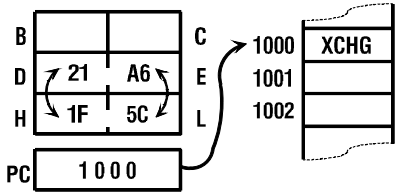
Formato: 1 1 1 0 1 0 1 1

Direccionam.: Por registro

Señalizadores: Ninguno

Ciclos M.: 1 Estados: 4

EJEMPLO: XCHG



Instrucciones aritméticas I

ADD r (A) + (R) → A, (A) - (R) → A

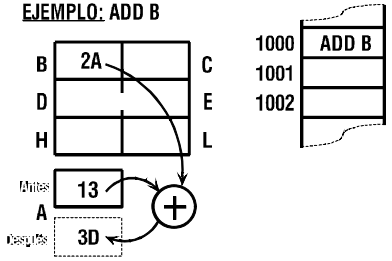
ADC r (A) + (R) + C → A, (A) - (R) - C → A

Direccionam.: Por registro

Señalizadores: Z S AC P CY

Ciclos M.: 1 Estados: 4

EJEMPLO: ADD B



ADI byte (A) + 2º byte → A, (A) – 2º byte → A

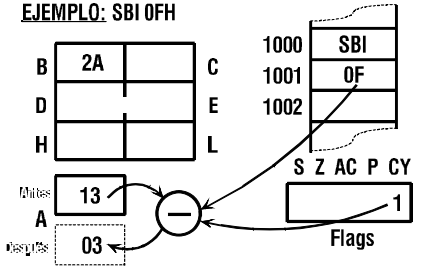
ACI byte (A) + 2º byte + C → A, (A) – 2º byte - C → A

Direccionam.: Inmediato

Señalizadores: Z S AC P CY

Ciclos M.: 2 Estados: 7

EJEMPLO: SBI 0FH



Instrucciones aritméticas II

ADD / SUB M (A) + [(H)(L)] → A, (A) – [(H)(L)] → A

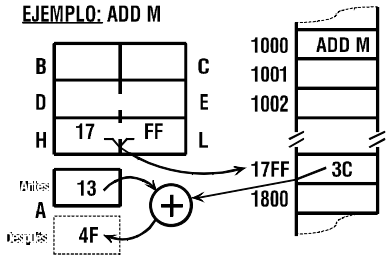
ADC / SBB M (A) + [(H)(L)] + C → A, (A) – [(H)(L)] - C → A

Direccionam.: Par de reg. Indirecto

Señalizadores: Z S AC P CY

Ciclos M.: 2 Estados: 7

EJEMPLO: ADD M



DAA Ajuste Decimal

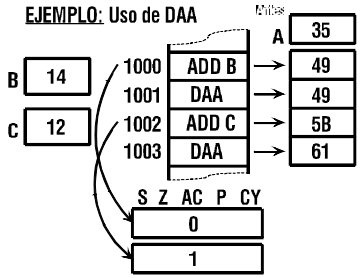
Formato: 0 0 1 0 0 1 1 1

Direccionam.: Implícito

Señalizadores: Z S AC P CY

Ciclos M.: 1 Estados: 4

EJEMPLO: Uso de DAA



Instrucciones aritméticas III

[redacted] (R) + 1 → R

[redacted] (R) -- 1 → R

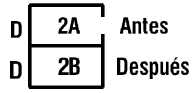
Direccionam.: Por registro
Señalizadores: Z S AC P
Ciclos M.: 1 Estados: 4

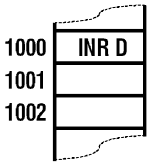
[redacted] [(H)(L)] + 1 → (H)(L)

[redacted] [(H)(L)] – 1 → (H)(L)

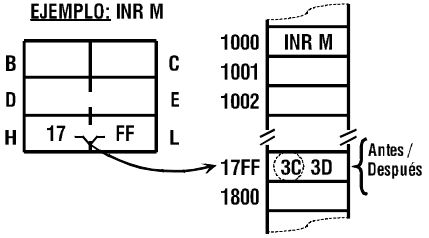
Direccionam.: Par reg. Indirecto
Señalizadores: Z S AC P
Ciclos M.: 3 Estados: 10

EJEMPLO: INR D





EJEMPLO: INR M



Instrucciones aritméticas IV

INX rp (rp) + 1 → rp

DCX rp (rp) – 1 → rp

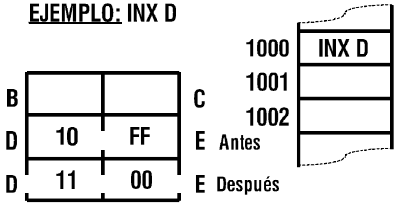
Direccionam.: Por registro
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 6

DAD rp (HL) + (rp) → HL

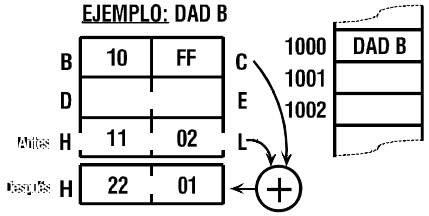
Formato: 0 0 R P 1 0 0 1

Direccionam.: Por registro
Señalizadores: C
Ciclos M.: 3 Estados: 10

EJEMPLO: INX D



EJEMPLO: DAD B



Instrucciones lógicas I

[redacted] (A) • (R) → A
(A) ⊕ (R) → A

[redacted] (A) + (R) → A

Direccionam.: Por registro
Señalizadores: Z S P; AC=1 CY=0
Ciclos M.: 1 Estados: 4

[redacted] (A) • [(H)(L)] → A
(A) ⊕ [(H)(L)] → A

[redacted] (A) + [(H)(L)] → A

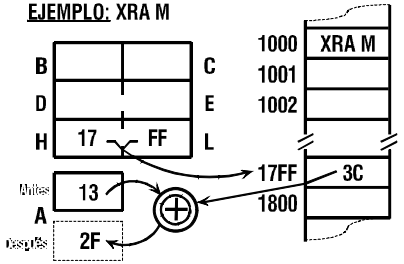
Direccionam.: Par reg. Indirecto
Señalizadores: Z S P; AC=1 CY=0
Ciclos M.: 2 Estados: 7

[redacted] (A) • byte → A
(A) ⊕ byte → A

[redacted] (A) + byte → A

Direccionam.: Inmediato
Señalizadores: Z S P; AC=1 CY=0
Ciclos M.: 2 Estados: 7

EJEMPLO: XRA M



Instrucciones lógicas II

CMP r (A) – (R)

Direccionam.: Por registro
Ciclos M.: 1 Estados: 4

CPI byte (A) – 2º byte

Direccionam.: Inmediato
Ciclos M.: 2 Estados: 7

CMP M (A) – [(H)(L)]

Ciclos M.: 2 Estados: 7
Direccionam.: Par reg. Indirecto

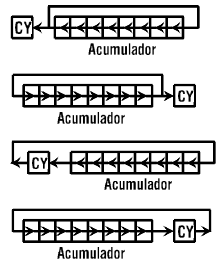
Señalizadores: Z S P AC CY

RLC

RRC

RAL

RAR



Longitud: 1 Byte (Opcode)
Direccionam.: Implícito
Señalizadores: CY
Ciclos M.: 1 Estados: 4

Instrucciones lógicas III

CMA $(\overline{A}) \rightarrow A$

Formato: 0 0 0 1 1 1 1 1

Direccionam.: Implícito

Señalizadores: Ninguno

Ciclos M.: 1 Estados: 4

CMC $\overline{C} \rightarrow C$

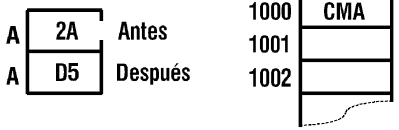
STC $1 \rightarrow C$

Señalizadores: CY

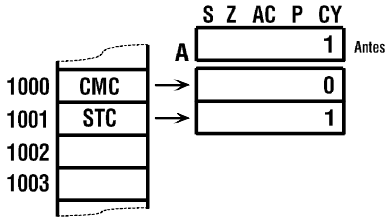
Direccionam.: Implícito

Ciclos M.: 1 Estados: 4

EJEMPLO: CMA



EJEMPLO: Uso de CMC y STC



Instrucciones de ruptura de secuencia

- Hay tres clases de instrucciones para la ruptura de la secuencia de un programa:
  - *De salto* : Este tipo de instrucciones saltan a cualquier instrucción del programa.
  - *De llamada a subrutina* : Son instrucciones de salto que guardan la dirección de retorno (dirección de la siguiente instrucción) en pila.
  - *De retorno* : Originan un salto a la instrucción cuya dirección está almacenada en la pila.
- Todas estas instrucciones pueden ser:
  - *Incondicionales* : Siempre se produce el salto o la bifurcación.
  - *Condicionales* : El salto se produce si se cumple la condición especificada en la instrucción. Las posibles condiciones están resumidas en la tabla adjunta.

CCC	Símbolo	Condición
0 0 0	NZ	NO ZERO (Z = 0)
0 0 1	Z	ZERO (Z = 1)
0 1 0	NC	NO ACARREO (CY = 0)
0 1 1	C	ACARREO (CY = 1)
1 0 0	PO	PARIDAD IMPAR (P = 0)
1 0 1	PE	PARIDAD PAR (P = 1)
1 1 0	P	POSITIVO (S = 0)
1 1 1	M	NEGATIVO (S = 1)

Instrucciones de ruptura de secuencia I

JMP $(3^{er}$ byte$)(2^{o}$ byte$) \rightarrow PC$

Formato: 11000011 | Addr(2)

Direccionam.: Inmediato

Señalizadores: Ninguno

Ciclos M.: 3 Estados: 10

Jccc Addr Si CCC $(3^{er}$ byte$)(2^{o}$ byte$) \rightarrow PC$ Si no $(PC) + 2 \rightarrow PC$

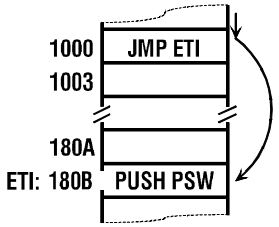
Formato: 11CCC010 | Addr(2)

Direccionam.: Inmediato

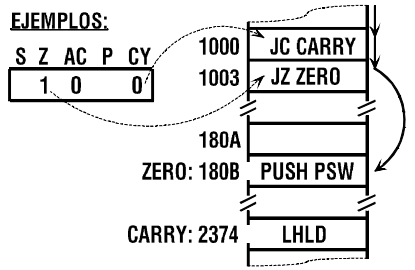
Señalizadores: Ninguno

Ciclos M.: 2/3 Estados: 7/10

EJEMPLO: JMP ETI



EJEMPLOS:



Instrucciones de ruptura de secuencia II

CALL $(PC_H) \rightarrow (SP) - 1$
$(PC_L) \rightarrow (SP) - 2$
$(SP) - 2 \rightarrow SP$
$(3^{er}$ byte$)(2^{o}$ byte$) \rightarrow PC$

Formato: 11001101 | Addr (2)

Direccionam.: Inmediato

Señalizadores: Ninguno

Ciclos M.: 5 Estados: 18

Cccc Addr Si CCC Idem CALL Si no $(PC) + 2 \rightarrow PC$

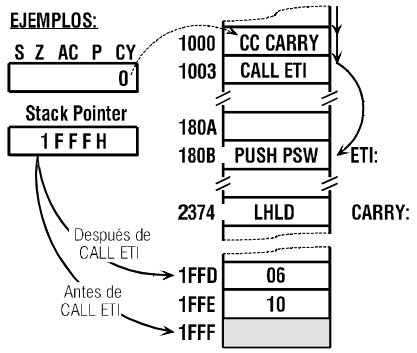
Formato: 11CCC100 | Addr(2)

Direccionam.: Inmediato

Señalizadores: Ninguno

Ciclos M.: 2/5 Estados: 9/18

EJEMPLOS:



Instrucciones de ruptura de secuencia III

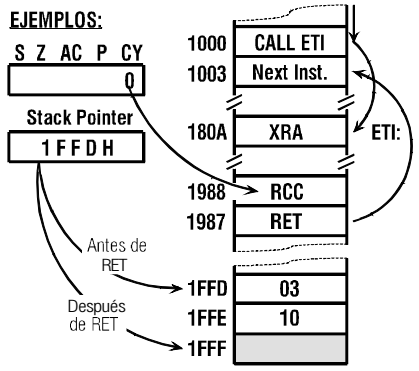
[ ] $[(SP)] \rightarrow PC_L$, $[(SP)+1] \rightarrow PC_H$, $(SP) - 2 \rightarrow SP$

Formato: 1 1 0 0 1 0 0 1
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 3 Estados: 10

Rccc Si CCC Idem RET, Si no $(PC) + 2 \rightarrow PC$

Formato: 1 1 C C C 0 0 0
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 1/3 Estados: 6/12

EJEMPLOS:

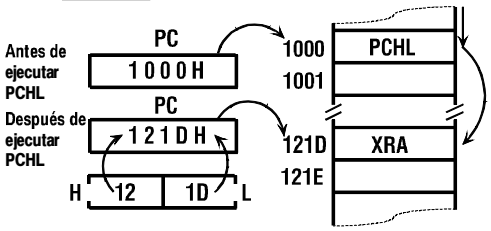


Instrucciones de ruptura de secuencia IV

PCHL $(H) \rightarrow PC_H$; $(L) \rightarrow PC_L$

Formato: 1 1 1 0 1 0 0 1
Direccionam.: Implícito / Por registro
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 6

EJEMPLO: PCHL



Instrucciones de manejo de la Pila e I/O I

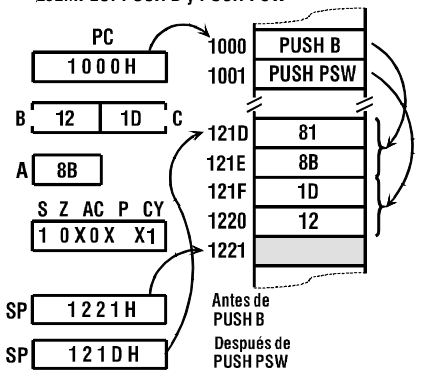
[ ] $(R_H) \rightarrow (SP) - 1$, $(R_L) \rightarrow (SP) - 2$, $(SP) - 2 \rightarrow SP$

Formato: 1 1 R P 0 1 0 1
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 3 Estados: 12

PUSH PSW $(A) \rightarrow (SP) - 1$, $(Rcc) \rightarrow (SP) - 2$, $(SP) - 2 \rightarrow SP$

Formato: 1 1 1 1 0 1 0 1
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 3 Estados: 12

EJEMPLO: PUSH B y PUSH PSW



Instrucciones de manejo de la Pila e I/O II

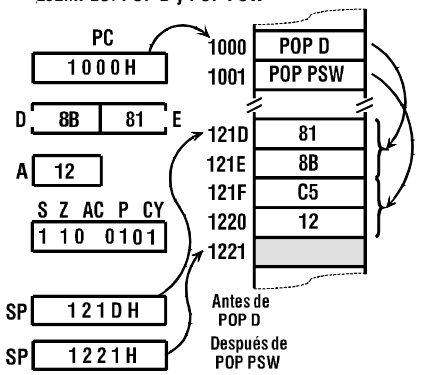
POP rp $[(SP)] \rightarrow R_L$, $[(SP) + 1] \rightarrow R_H$, $(SP) + 2 \rightarrow SP$

Formato: 1 1 R P 0 0 0 1
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 3 Estados: 10

POP PSW $[(SP)] \rightarrow Rcc$, $[(SP) + 1] \rightarrow A$, $(SP) + 2 \rightarrow SP$

Formato: 1 1 1 1 0 0 0 1
Direccionam.: Implícito
Señalizadores: Todos
Ciclos M.: 3 Estados: 10

EJEMPLO: POP D y POP PSW



Instrucciones de manejo de la Pila e I/O III

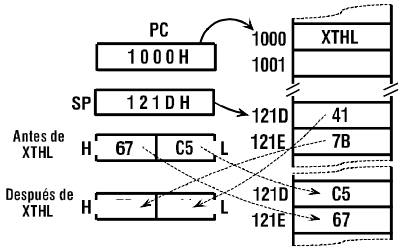
XTHL — [(SP)] ← → L, [(SP) +1] ← → H

Formato: 11100011
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 5 Estados: 16

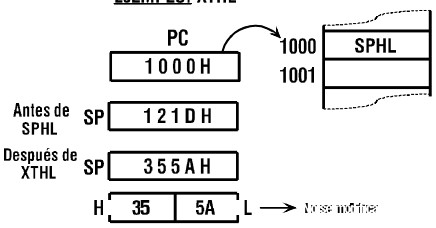
SPHL — (H) → SP$_H$, (L) → SP$_L$

Formato: 11111001
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 6

EJEMPLO: XTHL



EJEMPLO: XTHL



Instrucciones de manejo de la Pila e I/O IV

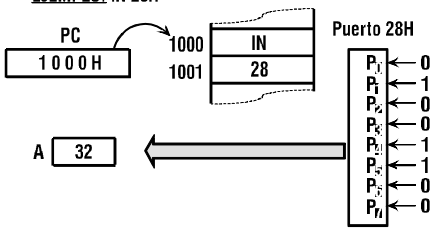
IN Port — (Puerto) → A

Formato: 11011011 | Port(1)
Direccionam.: Directo
Señalizadores: Ninguno
Ciclos M.: 3 Estados: 10

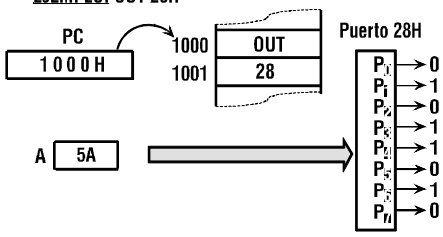
OUT Port — (A) → Puerto

Formato: 11010011 | Port(1)
Direccionam.: Directo
Señalizadores: Ninguno
Ciclos M.: 3 Estados: 10

EJEMPLO: IN 28H



EJEMPLO: OUT 28H



Instrucciones de control I

EI — Habilita las interrupciones

Formato: 11111011
Direccionam.: Ninguno
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 4

DI — Deshabilita las Interrupciones

Formato: 11111011
Direccionam.: Ninguno
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 4

NOP — No Operación

Formato: 00000000
Direccionam.: Ninguno
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 4

HLT — Para el microprocesador

Formato: 01110110
Direccionam.: Ninguno
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 5

Instrucciones de control I

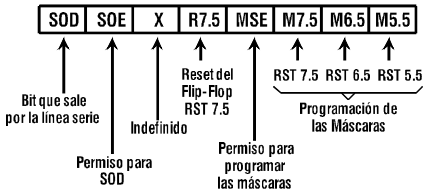
SIM — Posiciona las máscaras de interrupción.

Formato: 00110000
Direccionam.: Ninguno
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 4

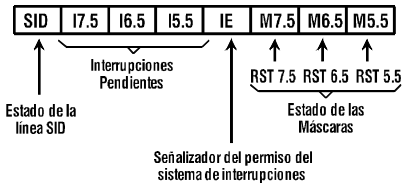
RIM — Lectura de las máscaras de interrupción.

Formato: 00100000
Direccionam.: Ninguno
Señalizadores: Ninguno
Ciclos M.: 1 Estados: 4

Significado del contenido del acumulador antes de SIM



Contenido del acumulador tras ejecutar la instrucción RIM



Introducción a las E/S del sistema

- Interfaz entre dispositivos periféricos y el µprocesador.
  - Diferencia en velocidad de funcionamiento (tasa de transferencia).
  - Diferencia en las unidades de información.
  - Diferencia en los modos de funcionamiento.

- Dispositivos mas comunes:
  - Adaptador para comunicación paralelo PPI o PIA.
  - Adaptador para comunicación serie USART.
  - Controlador de interrupciones PIC.
  - Controlador de tiempos TIMER.
  - Controlador de acceso directo a memoria DMA.

Manipulación de periféricos

- Procedimientos para la manipulación de los periféricos:

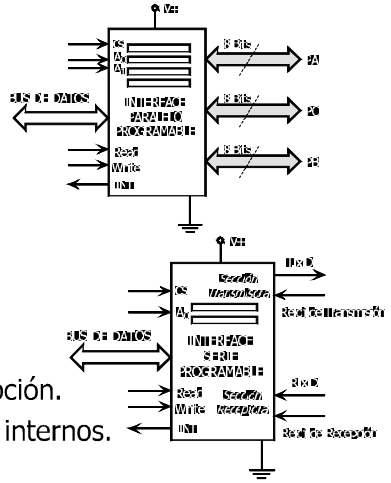
  - *Por software* : Proceso de *consultas sucesivas*. Bajo rendimiento de la CPU. Proceso síncrono. Consulta por *Polling*.

  - *Mediante interrupciones* : El dispositivo provoca una interrupción a la CPU. Posibilidad de asignar *prioridades* y *enmascaramiento*. Esta gestión la realiza el controlador de interrupciones (PIC).

  - *Acceso Directo a Memoria* : Utilizando la señal HOLD. Transferencia directa entre memoria y periférico (para grandes bloques de información). El µprocesador se desconecta de los buses.
    - Ejemplo: Disco duro < > Memoria Principal

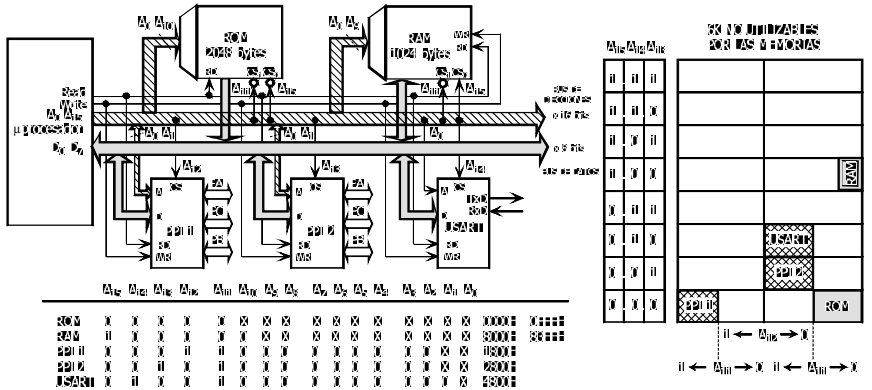
Esquema General de un dispositivo de E/S

- Adaptador de E/S paralelo:
  - - Resuelve la adaptación del sistema a periféricos que trabajan con información en paralelo.
  - - Dispositivo programable.
- Adaptador de E/S serie:
  - - Serialización del dato a transmitir.
  - - Paralelización del dato recibido.
  - - Gestión de errores en la transmisión del dato.
- Disponen de peticiones de interrupción.
- Entradas de selección de registros internos.



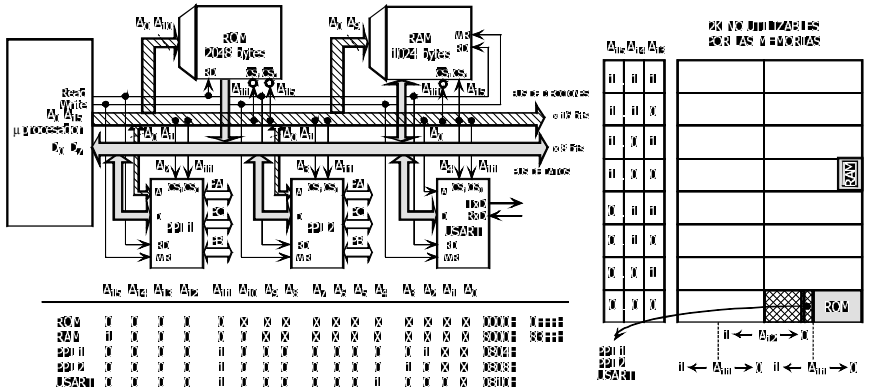
I. Ejemplo de interconexión

- Interconexión con el µprocesador: selección lineal.
  - Ejemplo: Sistema con 2K de ROM, 1K de RAM, 2 PPI, 1 USART.



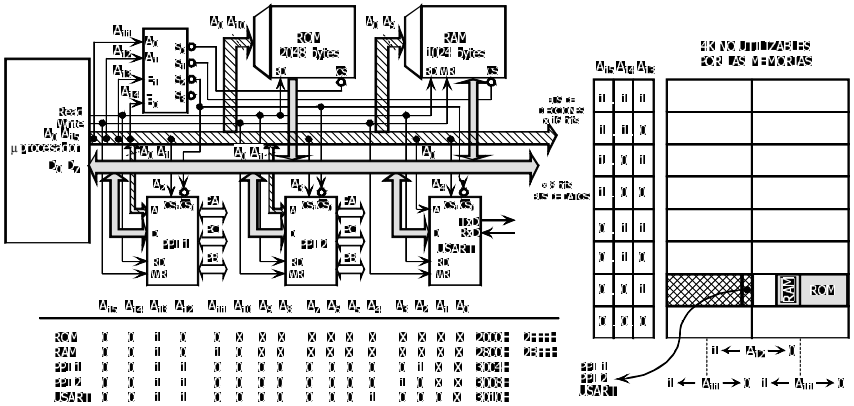
II. Ejemplo de interconexión

- Optimización del mapa de memoria (selección lineal).



III. Ejemplo de interconexión

- Optimización del mapa de memoria (selección por decodificación).

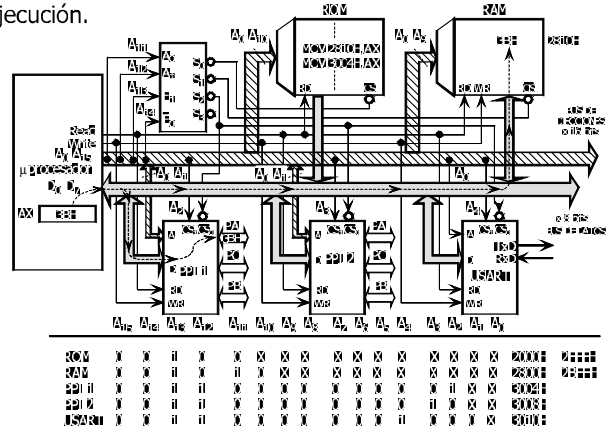


Introducción a las E/S del sistema

- Estructura de E/S por instrucciones de memoria:
  - Las transferencias de E/S periféricas se efectúan por instrucciones de memoria. Las señales *Read* y *Write* sirven tanto para las memorias como para los interfaces de E/S.
- Estructura de E/S por instrucciones de E/S:
  - Las transferencias son efectuadas por instrucciones específicas de los periféricos.
  - En esta estructura, el mapeado de una memoria y de un interface de E/S, pueden coincidir (señales de *Read* y *Write* diferentes para las memorias y para los interfaces de E/S).
  - El espacio de memoria está totalmente disponible para la memoria. En este caso, la cantidad máxima posible de memoria, coincide con el espacio de memoria del sistema.

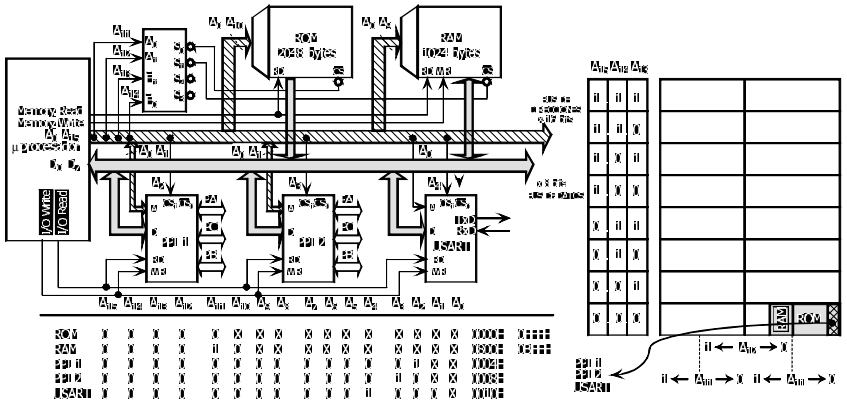
Introducción a las E/S del sistema

- Estructura de E/S por instrucciones de memoria:
  - Ejemplo de ejecución.



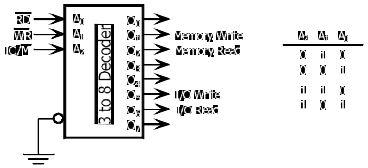
Introducción a las E/S del sistema

- Estructura de E/S por instrucciones de E/S:



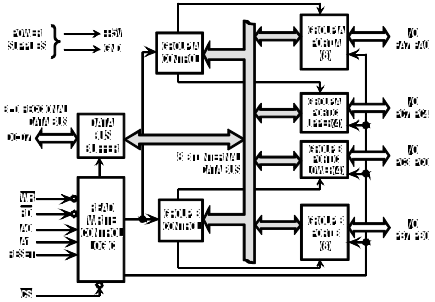
Introducción a las E/S del sistema

- Generación de las señales Memory Read, Memory Write, I/O Read e I/O Write:
  - Las genera directamente el µprocesador (ejemplo, µp. Intel 8080).
  - El µprocesador genera solamente Read y Write (tanto para Memoria como para E/S):
    - En este caso el µprocesador genera una señal (IO/$\overline{M}$ en el caso de Intel) indicando si el acceso es a Memoria o a E/S.
    - A.- Uso de la señal IO/$\overline{M}$ para la selección de las memorias e interfaces de E/S.
    - B.- Uso de la señal IO/$\overline{M}$ para genera las señales de Read y Write para memoria y E/S.



Interfaz Paralelo Programable 82C55A

- Estructura del PPI:



- Dispone de 3 puertos de E/S paralelos denominados A, B y C, con salidas latcheadas.
- El puerto C está divido en dos grupos de 4, que pueden trabajar de forma combinada con las puertas A y B en determinados modos.
- Internamente tiene 3 registros de datos y un registro de control para configurar el funcionamiento del dispositivo.
- Admite tres modos diferentes de funcionamiento, denominados Modo 0, Modo 1 y Modo 2.

82C55A: Tabla de verdad

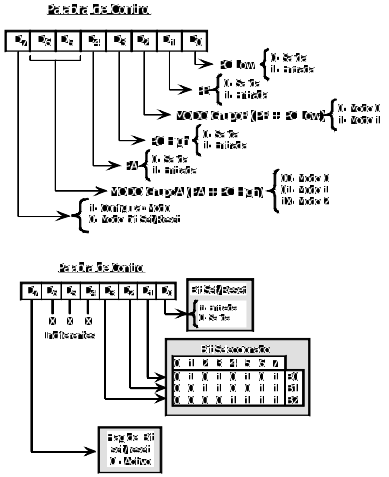
- Sentido de las transferencias de información entre el bus de datos del sistema y los registros internos del PPI:

A_{ii}	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	FUNCIÓN
0	0	0	1	0	Puerta A -> Bus de Datos
0	1	0	1	0	Puerta B -> Bus de Datos
1	0	0	1	0	Puerta C -> Bus de Datos
0	0	1	0	0	Bus de Datos -> Puerta A
0	1	1	0	0	Bus de Datos -> Puerta B
1	0	1	0	0	Bus de Datos -> Puerta C
1	1	0	1	0	ILEGAL
1	1	1	0	0	Bus de Datos -> Reg. Control
X	X	X	X	1	Dispositivo en Triestado

- Las líneas A_0 y A_{ii} determinan la dirección de los puertos.
  - Ejemplo: Dirección de base -> 3000H
    - PA: 3000; PB: 3001; PC: 3002; Reg. Control: 3003H

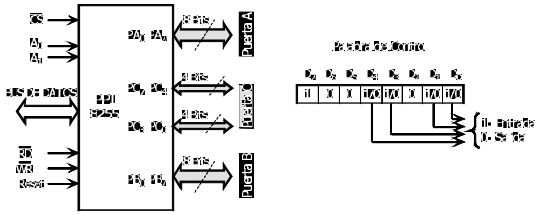
PPI 82C55A: Programación

- Programación del dispositivo:
  - Registro de Control: Configura la dirección y modo de funcionamiento de cada una de las puertas.
  - Modo bit Set/Reset: Permite sacar 1 ó 0 por cualquiera de las líneas del puerto C.



PPI: Funcionamiento en Modo 0

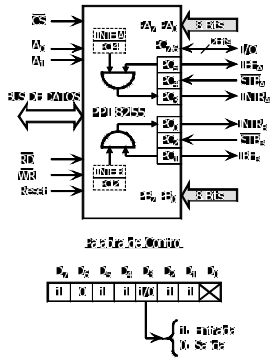
- Modos de funcionamiento:
  - Modo 0: Corresponde a la forma más básica de funcionamiento.
    - Cada puerta se configura como entrada o salida en el registro de control.
    - Salidas latcheadas y entradas no latcheadas.
    - No hay señales de diálogo con los periféricos.



PPI: Funcionamiento en Modo 1 (Entrada)

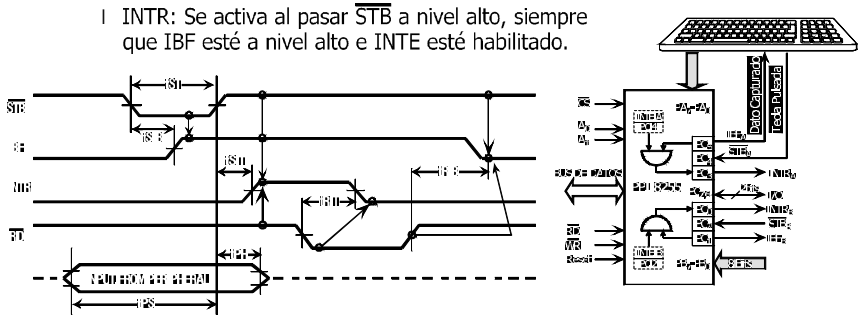
- Modo 1: Transferencia con *Handshaking*. Las Puertas A y B son controladas por las líneas de la puerta C (3 líneas por puerta).

- **Modo de Entrada:**
  - Puerta B: Controlada por los bits PC_0 a PC_3.
  - Puerta A: Controlada por los bits PC_3 a PC_5.
  - PC_6 y PC_7, pueden configurarse como líneas convencionales de E/S.
  - Señales del protocolo de entrada: IBF, STB e INTR.
  - Habilitación de las interrupciones a través de los biestables internos INTEA (PC_4) e INTEB (PC_2).



PPI: Protocolo de entrada en Modo 1

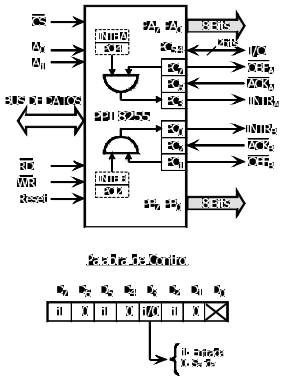
- Modo 1: Protocolo de comunicación con *handshaking* para la entrada.
  - $\overline{STB}$: Un nivel bajo en esta entrada da la señal de memorizar el dato que se encuentra en la entrada.
  - IBF: Un nivel alto, indica que el dato ha sido memorizado en los latches de entrada.
  - INTR: Se activa al pasar $\overline{STB}$ a nivel alto, siempre que IBF esté a nivel alto e INTE esté habilitado.



PPI: Funcionamiento en Modo 1 (Salida)

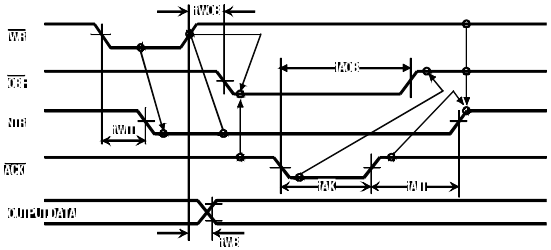
- Modo 1: Transferencia con *Handshaking*. Las Puertas A y B son controladas por las líneas de la puerta C (3 líneas por puerta).

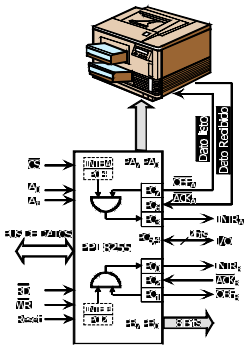
- **Modo de Salida:**
  - Puerta B: Controlada por los bits PC_0 a PC_3.
  - Puerta A: Controlada por los bits PC_3 , PC_6 y PC_7.
  - PC_4 y PC_5, pueden configurarse como líneas convencionales de E/S.
  - Señales del protocolo de salida: OBF, ACK e INTR.
  - Habilitación de las interrupciones a través de los biestables internos INTEA (PC_6) e INTEB (PC_2).



PPI: Protocolo de salida en Modo 1

- Modo 1: Protocolo de comunicación con *handshaking* para la salida.
  - $\overline{OBF}$: Esta señal se activa a nivel bajo cuando queda lleno el buffer de salida de la puerta.
  - $\overline{ACK}$: Un nivel bajo, señala el reconocimiento de la salida cuando el periférico lee la puerta.
  - INTR: Se activa a nivel alto con la señal de reconocimiento, siempre que $\overline{OBF}$ esté a nivel alto e INTE esté habilitado.

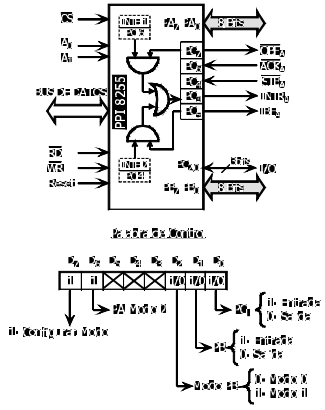




PPI: Funcionamiento en Modo 2

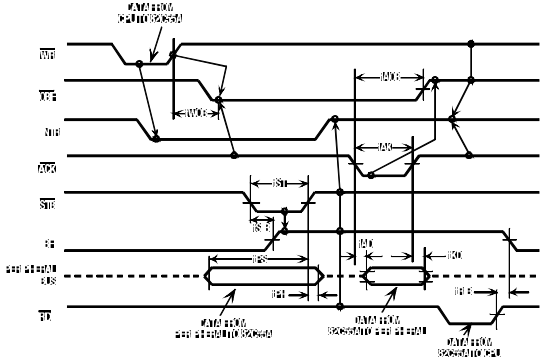
- Modo 2: Transferencia bidireccional con *Handshaking*.

- **Características:**
  - Solamente puede programarse para la Puerta A.
  - Puerta A: Controlada por los bits PC_3 a PC_7.
  - Puerta B: Puede funcionar en Modo 0 ó Modo 1 (Controlada por PC_0 a PC_2).
  - Señales del protocolo: IBF,STB, OBF, ACK e INTR.
  - Habilitación de la interrupción a través de los biestables internos INTE 1 (PC_6) e INTE 2 (PC_4).



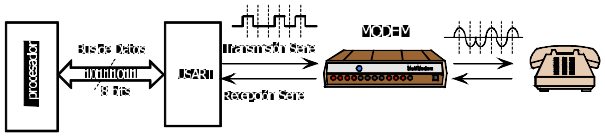
PPI: Protocolo en Modo 2

- Modo 2: Protocolo de comunicación bidireccional con *handshaking*.
  - INTR: La petición de interrupción se activa tanto por el protocolo de entrada como por el protocolo de salida. Estas peticiones pueden deshabilitarse por separado (biestable INTE 1 e INTE 2).



Introducción a la E/S serie

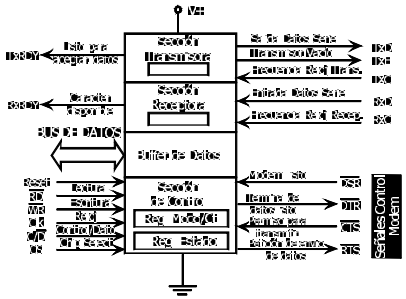
- Generalidades sobre la transmisión serie:



  - Necesidad de un dispositivo adaptador entre los periféricos serie y el µprocesador: el USART.
    - Necesidad de señales de reloj para la transmisión y recepción de datos.
    - Comunicaciones a largas distancias. Utilización del MODEM. Señales analógicas.
      - Modulación en Amplitud, frecuencia, o fase.
    - Modos de transmisión síncrono y asíncrono.

Diagrama de bloques de una USART

- Esquema general:



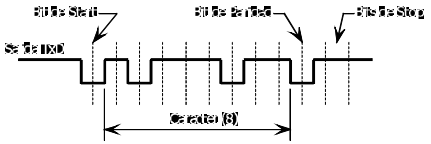
- **Sección Transmisora:**
  - Envía los datos en serie por TxD.
  - TxC recibe una señal cuya frecuencia sincroniza el reloj de transmisión.
  - Comunica al µprocesador la posibilidad de enviar otro carácter por TxRDY.
- **Sección Receptora:**
  - Recibe datos serie por RxD.
  - Sincroniza la frecuencia del reloj receptor a través de RxC.
  - Comunica al µprocesador la presencia de un carácter usando TxRDY.
- **Sección de Control:**
  - Gestiona la comunicación con el µprocesador y el MODEM.

Funcionamiento de una USART

- 1.- Configuración del modo de funcionamiento:
  - Frecuencia, en baudios, para la transmisión serie.
  - Anchura del carácter a transmitir.
  - Número de bits de STOP.
  - Selección del tipo de operación: síncrona o asíncrona.
  - Existencia del bit de paridad y tipo.
- 2. La señal TxRDY pasa a nivel alto indicando disponibilidad para la transmisión. La señal TxD está a nivel alto en ausencia de información.
  - La transmisión es posible siempre que TxEN esté a nivel alto y se active la señal CTS (permiso para transimitir).
- 3. La señal RxRDY pasa a nivel alto indicando que se ha recibido un dato.

Tipos de Transmisiones

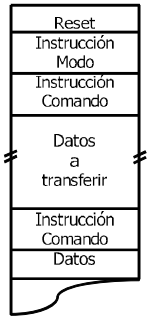
- Transmisión Asíncrona:
  - Señal de transmisión a nivel alto en ausencia de información.
  - Utilización de bits de Start y Stop (comienzo y fin de la transmisión).
  - El carácter es codificado en 5, 6 ó más bits.
  - Bit de paridad: Control de la transmisión.
    - Ejemplo de transmisión: Dato 10111011



- Transmisión Síncrona:
  - No incluye los bits de Start y Stop.
  - Transmisión por bloques de N caracteres.
  - Uso de caracteres de sincronización.

Programación de la USART 8251 de Intel

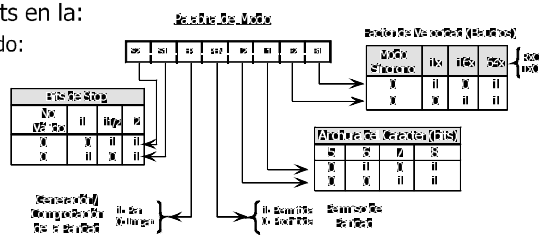
- Configuración de la USART por programa:
  - Se realiza escribiendo dos palabras de control sobre el registro de control, denominadas *instrucción de modo* e *instrucción de comando*.
  - La instrucción de Modo va siempre después de un Reset del dispositivo.
    - Reset Externo
    - Reset Interno (a través de la instrucción de Comando).
  - Después de una instrucción de Modo, le sigue una instrucción de Comando.
  - La instrucción de Comando puede insertarse en cualquier momento.



Disposición de la secuencia de instrucciones que inicializa el USART.

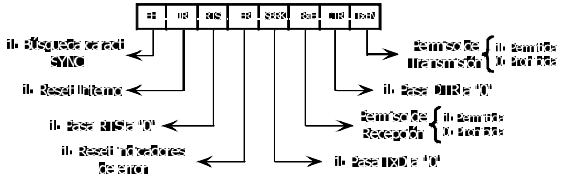
Instrucciones de Modo y Comando

- Disposición de los bits en la:
  - Instrucción de Modo:



Palabra de Comando

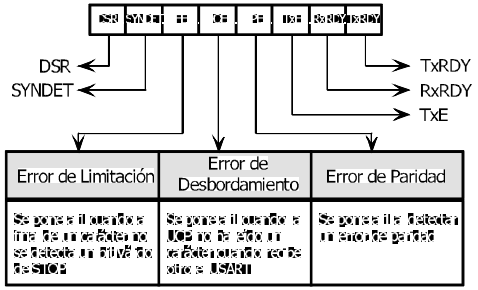
  - Instrucción de Comando:



Palabra de estado del 8251 de Intel

- Se obtiene mediante la lectura en el Registro de Control:
  - Señala las condiciones de errores en la transferencia de información.
  - Permite la comprobación de la existencia de un dato a leer y la posibilidad de transmitir, señales TxRDY y RxRDY, a través de una exploración por programa.

Palabra de Estado

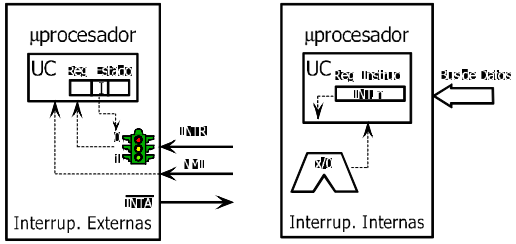


Introducción a las Interrupciones

- Manejo de las interrupciones en el sistema: Generalidades.
  - Una interrupción es una llamada a una subrutina.
  - La petición de interrupción es asíncrona.
  - Ante un interrupción el sistema:
    - Termina de ejecutar la instrucción en curso.
    - Salva en la pila la dirección de retorno y el registro de estado.
    - Carga en el PC la dirección de comienzo de una rutina, denominada *Rutina de Servicio de la Interrupción* y la ejecuta (cada interrupción está asociada a una RSI).
    - Ejecuta la RSI, y termina con un IRET (Retorno de Interrupción), que además de recuperar de la pila la dirección de retorno, recupera el registro de estado.
  - Cada interrupción está asociada a un número, denominado *Vector de Interrupción*.

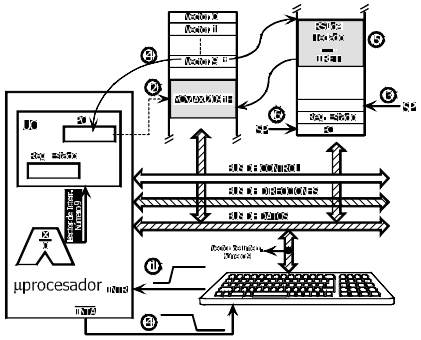
Clasificación de las Interrupciones

- Clasificación de las interrupciones:
  - Internas: Se producen dentro del µprocesador y pueden ser:
    - *Automáticas*: Desbordamiento por división, Overflow, Ejecución paso a paso.
    - *Provocadas por software*: Se producen al ejecutarse una instrucción del tipo "**INT n**".
  - **Externas**: Se piden desde fuera de la CPU y se clasifican en:
    - *Enmascarables*.
    - *No Enmascarables*.



Vectorización de las interrupciones

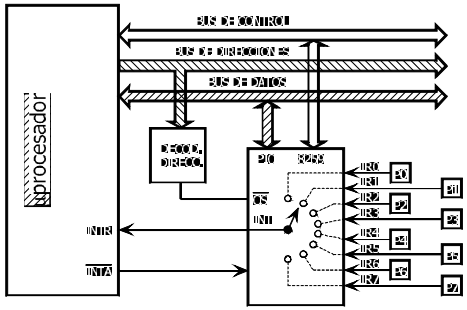
- Las interrupciones pueden a su vez clasificarse en:
  - *Vectorizadas*: Están asignadas por el sistema a un vector de interrupción. Todas las interrupciones internas están vectorizadas.
  - *No Vectorizadas*: Mandan el vector de interrupción al pedir la interrupción.



- 1.- Petición de Interrupción.
- 2.- Terminar la instrucción en curso.
- 3.- Guardar en la pila la dirección de retorno y el registro de estado.
- 4.- Reconocimiento de la interrupción. El contenido del vector de interrupción se carga en el contador de programa.
- 5.- Se ejecuta la Rutina de Servicio de la Interrupción.
- 6.- Recuperar de la pila la dirección de retorno (instrucción IRET).

Introducción al PIC (8259) de Intel

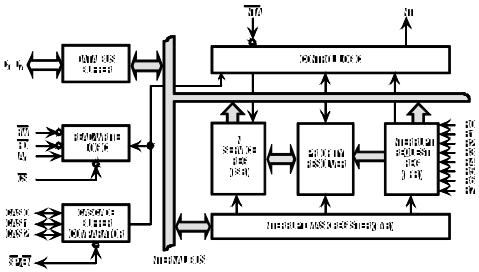
- Controlador de Interrupciones, PIC 8259: Funcionamiento.



- Recibe las peticiones de interrupción de los dispositivos periféricos.
- Determina las prioridades de las diferentes peticiones.
- Posibilita el enmascaramiento individual de las interrupciones.
- Realiza la petición de interrupción al µprocesador.
- Deposita el vector de interrupción en el bus de datos del sistema.

Diagrama de bloques del 8259

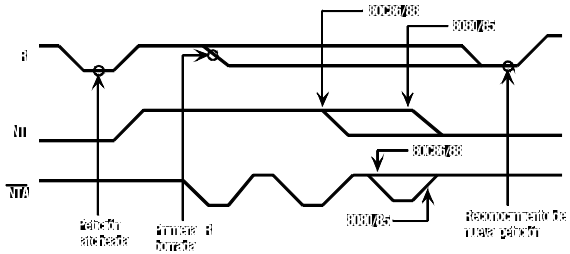
- Estructura interna del PIC:



- IRR: Registra los periféricos que solicitan servicio de interrupción.
- IMR: Registro de 8 bits que contiene las máscaras de interrupciones. Un bit a "0" habilita la interrupción asociada a ese bit.
- ISR: Regitra la petición de interrupción que está siendo atendida en ese momento.
- Priority Resolver: es una lógica intermedia que resuelve la petición más prioritaria que no esté enmascarada.
- Modos de Funcionamiento: Aislado o en cascada con estructura de Maestro -- Esclavo. Señal SP/EN.
- Cascade Buffer Comparator: Registra los códigos de los PIC esclavos cuando existe una estructura en cascada.

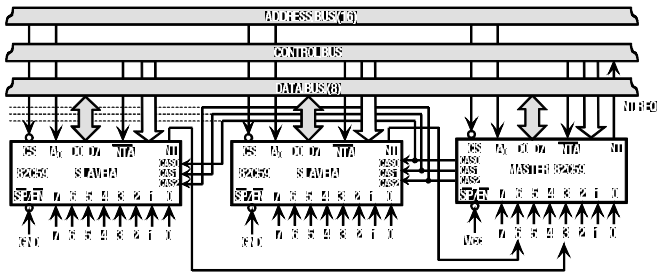
Protocolo de comunicación del PIC

- Modo de funcionamiento aislado:
  - IR: Debe permanecer a nivel alto hasta el primer flanco de bajada de la señal de reconocimiento $\overline{INTA}$.
  - INT: Se activa al recibirse un IR no enmascarado. Desaparece en el segundo flanco de subida de la señal de reconocimiento.
  - $\overline{INTA}$: Con el primer ciclo el 8259 activa el bit de mayor prioridad que ha requerido interrupción. También resetea el bit correspondiente en el registro IRR. Con el segundo ciclo el 8259 deposita el vector de interrupción en el bus de datos.



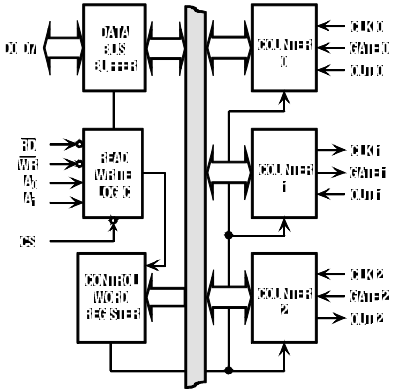
Conexionado del PIC en cascada

- Modo de funcionamiento en cascada:
  - El 8259 puede funcionar como maestro (*master*) controlando un máximo de 8 esclavos (*slaves*) 8259, manejando hasta 64 posibles interrupciones.
  - Las peticiones de interrupción de los *slaves* son dirigidas a las entradas de interrupción del *master*.
  - El *master* controla los *slaves* a través de las líneas CAS0-2 que actúan como *chip-select* de éstos durante los ciclos de $\overline{INTA}$.



Controlador programable de tiempos. Timer

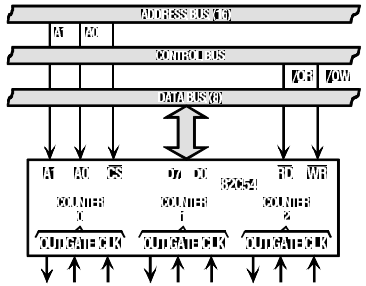
- Diagrama de Bloques del 82C53/82C54 de Intel:



- Permite generar tiempos de retardo precisos, mediante control por programa.
- Comunicación con la CPU a través de interrupciones, mediante las salidas OUT0-3.
- Características:
  - Tres contadores de 16 bits, totalmente independientes, seleccionados con las señales A_0 y A_1.
  - Entradas de Reloj independientes para cada contador, CLK0-3.
  - Contaje en binario o en BCD.
  - Velocidad de contaje: 4MHz, (12MHz para el 82C54).
  - Dispone de 6 modos de funcionamiento.

Programación del Timer 82C53/4

- Programación del 8253:
  - Los contadores se programan escribiendo primero la Palabra de Control y, posteriormente, un contaje inicial.
  - La palabra de control se escribe sobre el Registro de Control (A_0=1 y A_1=1).
  - El contaje inicial se escribe sobre los contadores. Selección con A_0 y A_1. Este contaje depende del modo de escritura seleccionado en la Palabra de Control.
- Conexionado del dispositivo



Resumen de operaciones $\overline{RD}$/$\overline{WR}$:

CS	RD	WR	A1	A0	Operation
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (Three State)
1	X	X	X	X	No-Operation (Three State)
0	1	X	X	X	No-Operation (Three State)

Palabra de control del 82C53

- Formato de la Palabra de Control:

A1,A0=11; CS=0; RD=1; WR=0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC	Select Counter
SC0	SC1	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command

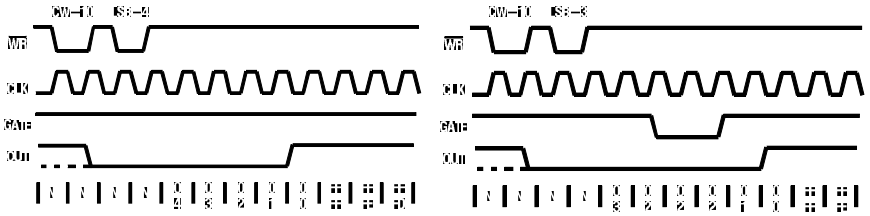
RW	Read/Write
SC0	SC1	
0	0	Counter Latch Command
0	1	RD/WR least significant byte only
1	0	RD/WR most significant byte only
1	1	RD/WR least significant byte first, then most significant byte

BCD	Binary Code Decimal
0	Binary Counter 16-Bit
0	Binary Coded Decimal (BCD) Counter (4 Decades)

M	Mode	
M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

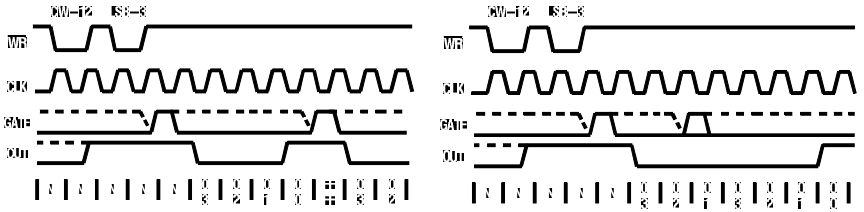
82C53: Funcionamiento en Modo 0

- Modo 0: *Interrupción al final del contaje*.
  - Se utiliza para contar eventos.
  - La salida pasa a nivel bajo cuando se programa. Al final del contaje, la salida pasa a nivel alto.
  - El contaje comienza después de iniciar el contador.
  - La señal GATE detiene el contaje.



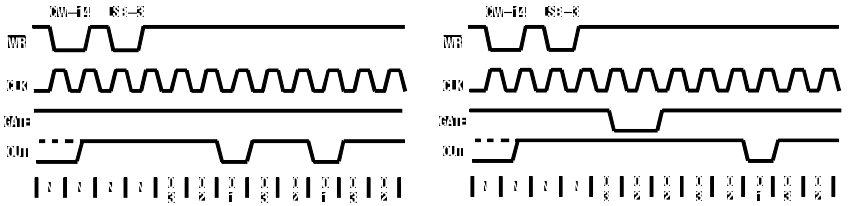
82C53: Funcionamiento en Modo 1

- Modo 1: *Monoestable Programable.*
  - La salida pasa a nivel alto cuando se programa, y pasa a nivel bajo cuando comienza el contaje.
  - El contaje comienza después del TRIGGER de la señal GATE, y la salida pasa a nivel alto al final del contaje.
  - Un TRIGGER: al final del contaje lo inicializa, poniendo la salida a nivel bajo; durante el contaje lo inicializa , manteniendo la salida a nivel bajo.



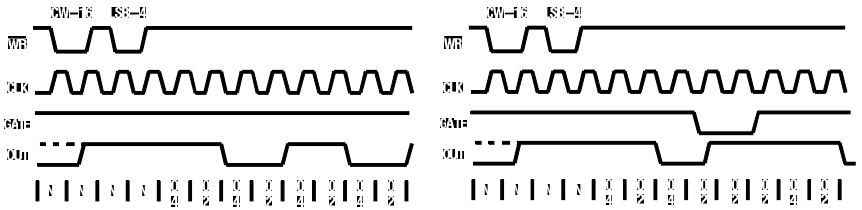
82C53: Funcionamiento en Modo 2

- Modo 2: *Divisor de Frecuencia.*
  - Funciona como un contador de módulo programable.
  - La salida pasa a nivel alto cuando se programa. Una vez programado, pasa a nivel bajo durante el último periodo de CLK.
  - Este modo es periódico; la secuencia se repite indefinidamente.
  - Un TRIGGER en la señal GATE inicializa el contaje.



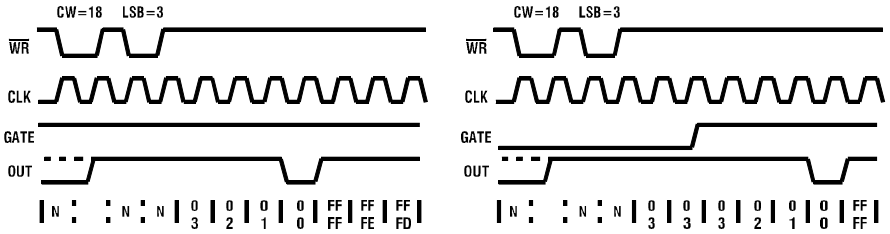
82C53: Funcionamiento en Modo 3

- Modo 3: *Generador de Onda Cuadrada.*
  - Similar al modo 2, excepto el ciclo de la salida OUT.
  - La salida a pasa a nivel alto cuando se programa. La salida pasa a nivel bajo durante la segunda mitad del contaje.
  - Para módulos impares la salida permanece a nivel alto un periodo más de la señal CLK.



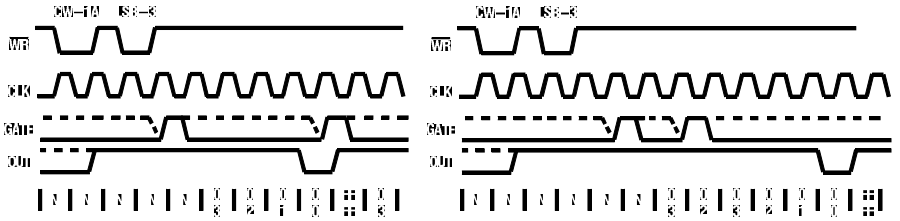
82C53: Funcionamiento en Modo 4

- Modo 4: *Retardo Activado por Software.*
  - La salida pasa a nivel alto cuando se programa. Una vez programado, pasa a nivel bajo al final del contaje, durante un periodo de CLK.
  - El contaje se detiene cuando la entrada GATE pasa a nivel bajo.
  - El contaje se dispara al escribir siempre un nuevo dato.



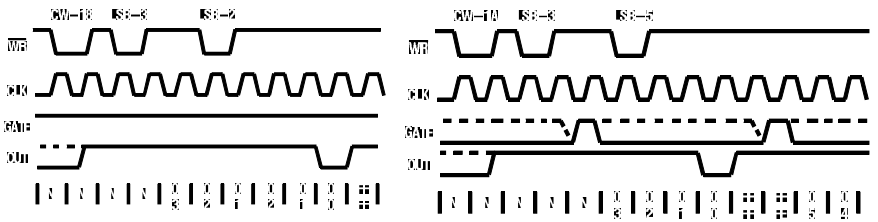
82C53: Funcionamiento en Modo 5

- Modo 5: *Retardo Activado por Hardware.*
  - La salida pasa a nivel alto cuando se programa. Una vez programado, pasa a nivel bajo al final del contaje, durante un periodo de CLK.
  - El contaje se dispara por un flanco de subida de la señal GATE.
  - Un TRIGGER en la señal GATE inicializa el contaje.



8253: Modo4 vs Modo 5

- Modo 4 y 5:
  - Modo 4: El contaje se inicializa siempre al escribir un nuevo dato. El contaje se inicializa por *software*.
  - Modo 5: El contaje se inicializa siempre al recibir un TRIGGER por la señal GATE. La escritura de un nuevo valor no afecta el contaje hasta después del TRIGGER. El contaje se inicializa por *hardware*.



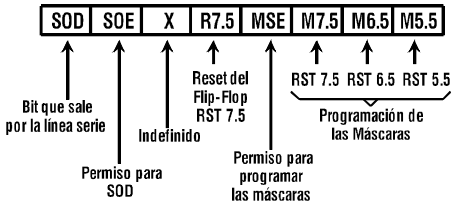
Introducción a las interrupciones en el 8085

- Tiene 5 interrupciones:
  - TRAP, RST 5.5, RST 6.5, RST 7.5 e INTR
- Instrucciones asociadas:
  - EI : Habilitación de las interrupciones (*Enable Interrupt*).
  - DI : Deshabilitación de las interrupciones (*Disable Interrupt*).
  - SIM : Colocar las máscaras de interrupción.
  - RIM : Lectura de las máscaras de interrupción.
- Uso de las interrupciones:
  - TRAP : Es una interrupción *no enmascarable*.
  - INTR : Para poder utilizarla hay que habilitar las interrupciones (EI). INTR no tiene máscara de interrupción asociada.
  - RST 5.5, RST 6.5 y RST 7.5 : Además de habilitar las interrupciones, hay que colocar las máscaras de interrupciones, instrucción SIM.

Manejo de las interrupciones I.

- Instrucción EI: Habilita las interrupciones INTR, RST 5.5, 6.5 y 7.5
- Instrucción DI: Deshabilita las interrpciones anterirores.
- Instrucción SIM: Coloca las máscaras de interrupción.

Significado del contenido del acumulador antes de SIM

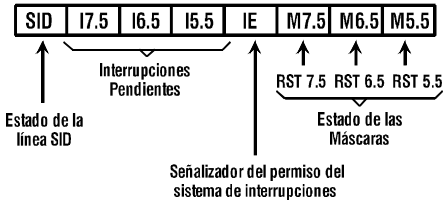


- **M5.5;M6.5;M7.5 : Máscaras de interrupción.**
  - 1 : No se permite la interrupción.
  - 0 : Interrupción habilitada.
- **MSE : Habilita la escritura de las máscaras de interrupción. Si está a cero, los bits MX.X no tienen efecto.**
- **R7.5 : A uno provoca un reset del biestable interno asociado a RST 7.5.**
- **SOE : Habilitación de salida serie.**
- **SOD : Bit que sale por la línea serie.**

Manejo de las interrupciones II.

- Instrucción RIM: Lectura de las máscaras de interrupción. Carga en PA_7 el valor de la entrada serie SID.

Contenido del acumulador tras ejecutar la instrucción RIM



- **M5.5;M6.5;M7.5 : Estados de las máscaras de interrupción.**
- **IE : Habilitación de las interrupciones.**
  - 0 : Deshabilitadas.
  - 1 : Habilitadas.
- **I7.5, I6.5, I5.5 : Interrupciones pendientes.**
- **SID : Contenido de la entrada serie.**

Ejemplos de programación

- Ejemplo 1:
  - Se desea permitir las interrupciones RST5.5 y 7.5, y deshabilitar la 6.5.

```
EI          ; Habilitar las interrup.
MVI  A,0AH  ; Poner máscara en A.
SIM         ; Colocar las máscaras.
```

- Ejemplo 2:
  - Se desea resetear el flip-flop interno R7.5.

```
MVI  A,10H  ; Poner bit 4 a uno.
SIM         ; Resetear el flip-flop.
```

- Ejemplo 3:
  - Se desea deshabilitar la interrupción RST 5.5 y habilitar la RST 6.5 y RST 7.5 dentro de la rutina *TEST*.

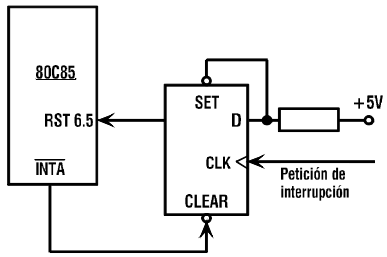
```
TEST:
DI          ; Deshabilitar interrupciones.
MVI  A,09H  ; Máscara de RST 5.5 a 1.
SIM         ; Coloco las máscaras
EI          ; Habilito interrupciones.
```

Nota: Un RESET pone las máscaras de interrupción a 1 y desahabilita las interrupciones.

Descripción de las interrupciones

- TRAP:
  - Sensible al flanco y al nivel y debe mantenerse hasta ser reconocida.
  - ¡ No enmascarable !.
- RST 5.5 y RST 6.5:
  - Sensibles a nivel alto.
  - Enmascarables por máscara y con la instrucción DI.
- RST 7.5:
  - Sensible al flanco.
  - Posee un flip-flop interno (R7.5) que es borrado al atenderla.
  - Enmascarable por máscara y por DI.
- INTR:
  - Interrupción sólo enmascarable por software (instrucción DI).
  - Sensible al nivel.

Reconocimiento por flanco de las interrupciones RST 6.5, RST 5.5 o INTR



Vectorización y prioridades

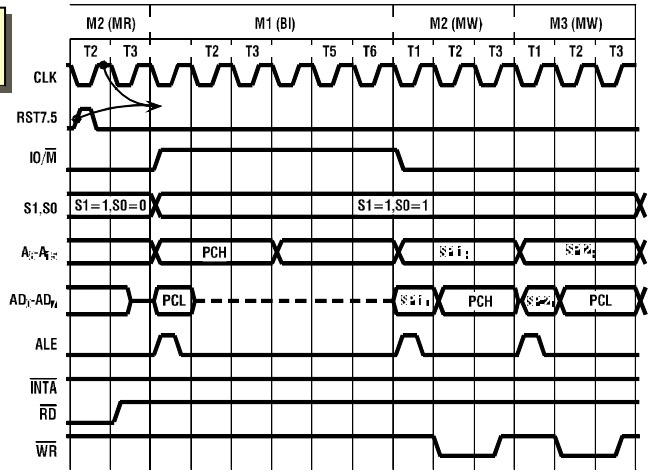
- Tabla de vectorización y prioridades:

Nombre	Prioridad	Dirección De salto	Activación
TRAP	1	24H	Flanco ascendente y nivel hasta ser muestreada.
RST 7.5	2	3CH	Flanco ascendente.
RST 6.5	3	34H	Nivel alto hasta el muestreo.
RST 5.5	4	2CH	Nivel alto hasta el muestreo.
INTR	5	No vectorizada	Nivel alto hasta el muestreo.

- Nota: Todas las interrupciones son muestreadas en el penúltimo flanco ascendente de CLK en el último ciclo de máquina de la instrucción donde se ha producido la petición de interrupción.

Reconocimiento de la interrupción RST 7.5

• Cronograma para el reconocimiento de la interrupción RST 7.5



Reconocimiento de la interrupción INTR

- Reconocimiento de la interrupción con $\overline{INTA}$.
- Instrucciones de RESTART: RTS n

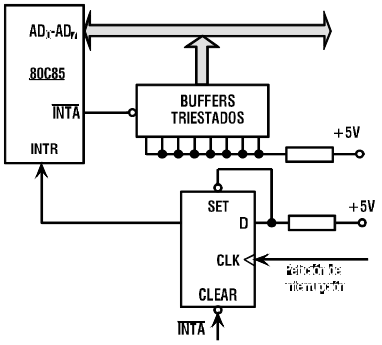
RST n

$(PC_7) \rightarrow [(SP)-1]$
$(PC_0) \rightarrow [(SP)-2]$
$(SP) - 2 \rightarrow SP$
$8*n \rightarrow PC$

Formato: 1 1 N N N 1 1 1
Direccionam.: Implícito
Señalizadores: Ninguno
Ciclos M.: 3 **Estados:** 12

RST 0 →	0000	RST 4 →	0020
RST 1 →	0008	RST 5 →	0028
RST 2 →	0010	RST 6 →	0030
RST 3 →	0018	RST 7 →	0038

Instrucción INTR con un RESTART 7



Interrupción INTR con respuesta tipo CALL