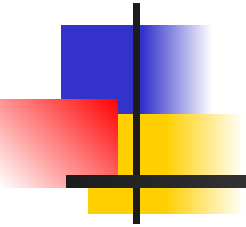


# Organización de Computadoras



## Clase 8



# Temas de Clase

---

- Organización de Registros
- Instrucciones



# Organización de registros

---

- Registros visibles al usuario: son utilizados por el programador.
- Registros de control y estado: son utilizados por la UC para controlar la operación de la CPU (no son visibles por el programador).



# Registros visibles al usuario

---

- Propósito general
- Datos
- Dirección
- Códigos de condición



## Registros visibles al usuario(2)

---

- Pueden ser asignados a una variedad de funciones:
  - ✓ cualquier registro de propósito general puede contener el operando para cualquier código de operación (verdadero propósito)
  - ✓ pueden existir restricciones (ej. registros dedicados a operaciones en PF)



# Registros visibles al usuario(3)

---

- ✓ se pueden utilizar para direccionamiento (ej. indirecto de registro)
- ✓ sólo para datos ó sólo para direcciones
- ✓ los registros de dirección pueden ser asignados para un mdd (ej. reg índice para direccionamiento autoindexado)



# Discusión

---

- ❖ ¿Todos los registros de propósito general ó especializar su uso?
  - ❖ Todos de propósito general: afecta al tamaño de las instrucciones.
  - ❖ Especializados: puede estar implícito en el código de operación a qué registro se refiere (ej. Acumulador). Se ahorran bits. Limitan la flexibilidad del programador.
- ❖ No hay una receta.



# Número de registros

---

- ❖ Afecta al tamaño de la instrucción.
- ❖ Mayor N° de registros, más bits para especificarlos en la instrucción.
- ❖ Pocos registros: más referencias a memoria
- ❖ N° óptimo: entre 8 y 32 reg. Más, no hay gran mejora (aumenta tamaño de la instrucción).
- ❖ 2<sup>do</sup> cuatrimestre: discutimos RISC.





# Longitud de los registros

---

- De direcciones: deben ser capaces de almacenar la dirección más grande.
- De datos: deben estar habilitados para almacenar la mayoría de los tipos de datos.
- Algunas máquinas permiten 2 registros contiguos utilizados como un solo registro para almacenar valores de doble longitud.



# Bits de condición (banderas)

---

- Bits establecidos por la CPU como resultado de operaciones.
- Pueden ser utilizados por las instrucciones de bifurcación condicional.
- Generalmente no son alterados por el programador.



# Registros de control y estado

---

- Empleados para controlar la operación de la CPU. En la mayoría de las máquinas no son visibles al usuario.
- Los 4 esenciales para la ejecución de instrucciones:
  - Contador de programa (PC)
  - Registro de instrucción (IR)
  - Registro de dirección de memoria (MAR)
  - Registro buffer de memoria (MBR)



## Reg. de control y estado (2)

---

- Los 4 reg recién mencionados se emplean para el movimiento de datos entre la cpu y memoria.
- Dentro de la CPU los datos se deben presentar a la ALU para procesamiento, ésta puede acceder al MBR y a los reg visibles por el usuario. Puede haber también reg temporales adicionales para intercambiar datos con el MBR y demás reg visibles.

# Organización de registros CPU PII Intel (principales)

16	8	8	
	AH <i>A</i> X	AL	<b>EAX</b>
	BH <i>B</i> X	BL	<b>EBX</b>
	CH <i>C</i> X	CL	<b>ECX</b>
	DH <i>D</i> X	DL	<b>EDX</b>

De uso general

 No presente en 8086

En el 8086 eran registros de 16 bits AX, BX, CX y DX

# Organización de registros CPU PII Intel (principales)(1)

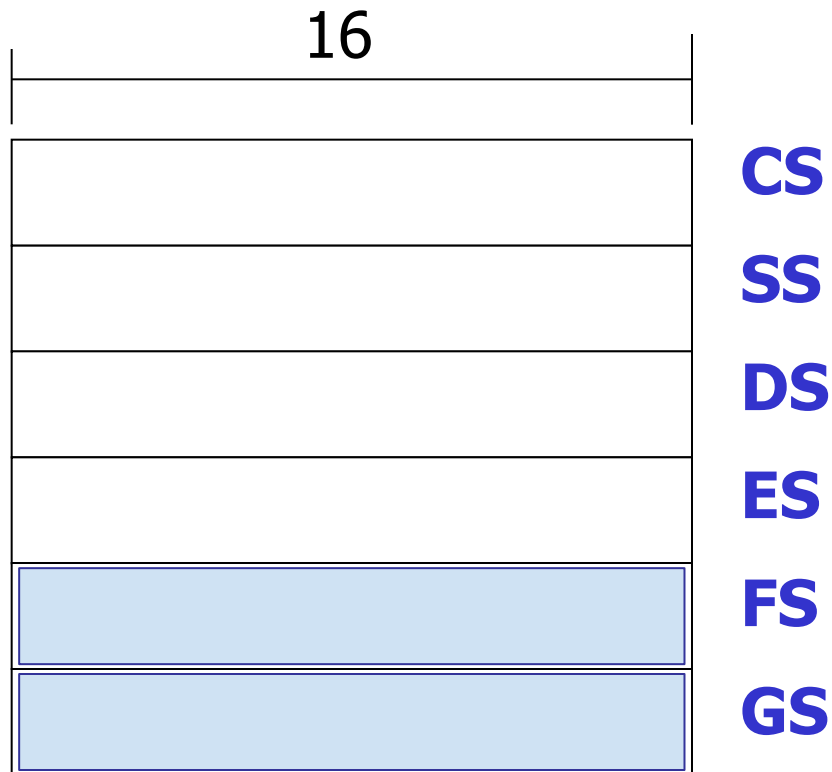
16	16	
	<b>SP</b>	<b>ESP</b>
	<b>BP</b>	<b>EBP</b>
	<b>DI</b>	<b>EDI</b>
	<b>SI</b>	<b>ESI</b>

Punteros e índices

 No presente en 8086

En el 8086 eran registros de 16 bits SP,BP,DI y SI

# Organización de registros CPU PII Intel (principales)(2)



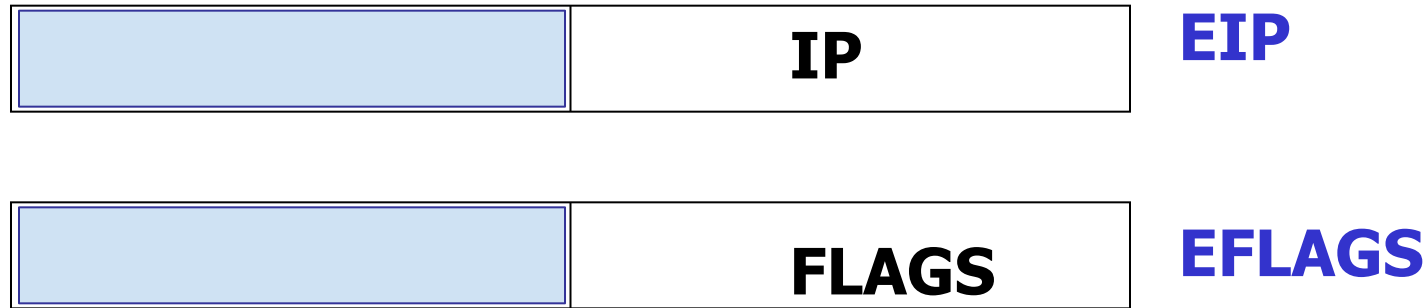
Segmentos




No presente en 8086

En el 8086 había 4 registros de  
segmentos: CS,SS,DS Y ES

# Organización de registros CPU PII Intel (principales)(3)



PC y banderas

 No presente en 8086

En el 8086 eran registros de 16 bits IP y FLAGS





# Organización de registros CPU PII Intel (principales)(4)

---

- AX : acumulador, es el principal en las operaciones aritméticas
- BX : puntero base (dir de memoria)
- CX : contador, interviene en instrucciones de ciclo
- DX : datos, participa en multiplicación y división



# Organización de registros CPU PII Intel (principales)(5)

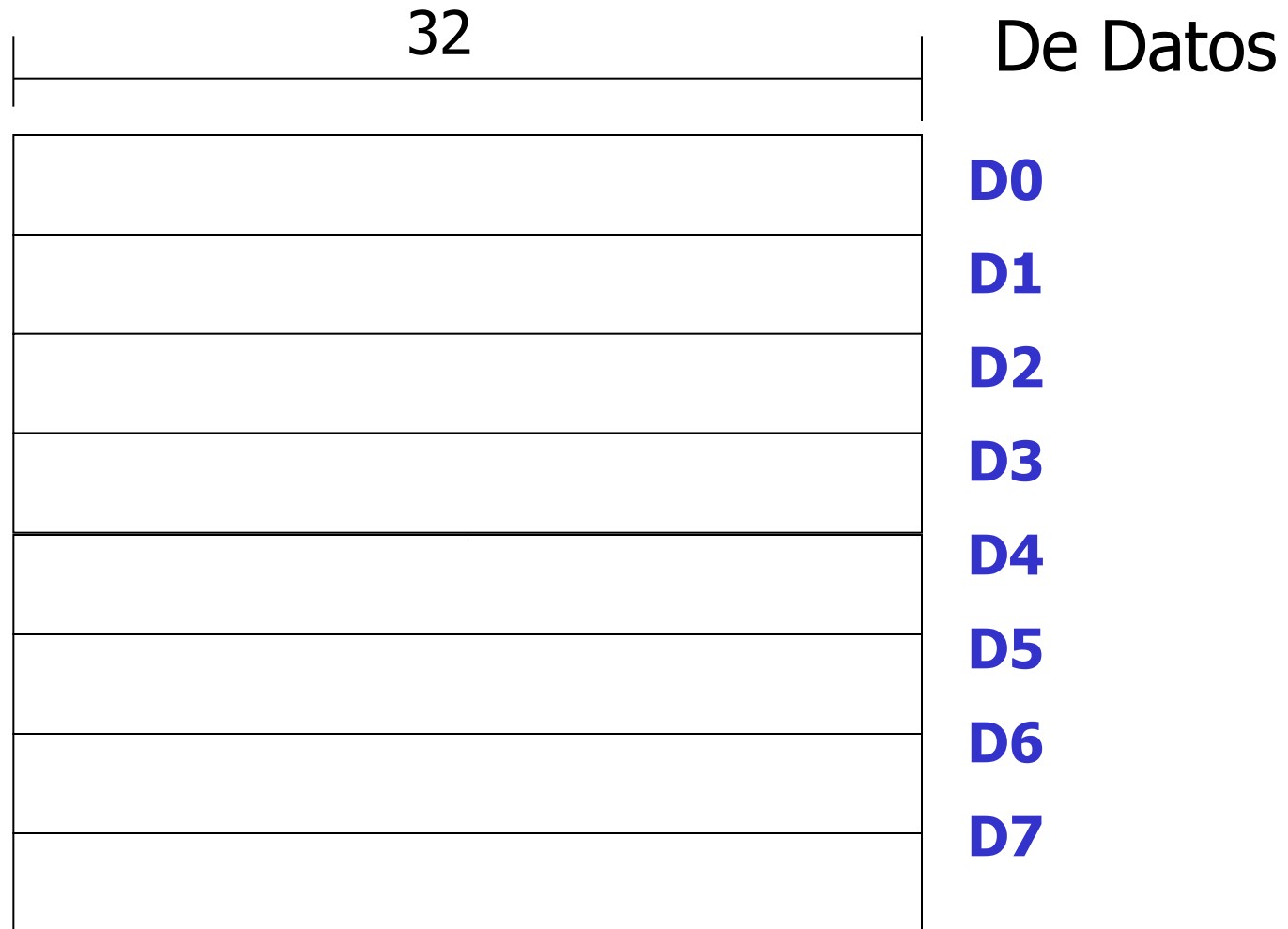
---

- SI y DI : apuntadores que utilizan las instrucciones que recorren arreglos o tablas
- BP y SP : también son apuntadores a memoria, pero a una zona especial: pila ó stack
- E : reg de 32 bits



# Organización de registros CPU MOTOROLA 68000

---





# Organización de registros CPU MOTOROLA 68000 (2)

Apuntador del stack usuario
Apuntador del stack supervisor

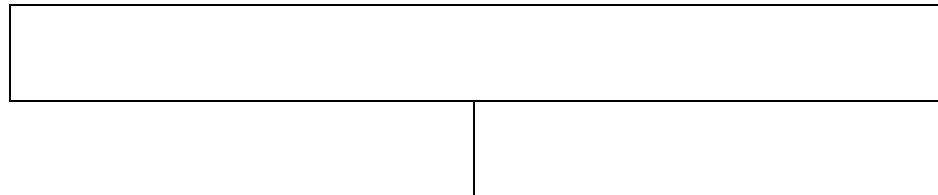
De  
Direcciones

**A0**  
**A1**  
**A2**  
**A3**  
**A4**  
**A5**  
**A6**  
**A7**  
**A7'**



# Organización de registros CPU MOTOROLA 68000 (3)

---



**PC**

**Estado**

- 8 registros de 32 bits de datos
- 9 registros de direcciones
  - 2 stacks: uno para usuario y otro para S.O.



# Instrucciones - Intel

---

- Tienen la forma :

instrucción destino,fuente

- destino y fuente son 2 operandos, donde c/u de ellos está especificado por alguno de los mdd vistos, el otro operando es un registro de la CPU

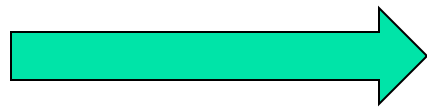


# Instrucciones - Intel (2)

---

## ❖ Llamando :

- mem = especificación de una dirección de memoria
- reg = registro de la CPU
- inm = dato inmediato



Las instrucciones  
tienen la forma



# Instrucciones - Intel (3)

---

- Instrucción mem, reg
- Instrucción reg , mem
- Instrucción reg , reg
- Instrucción reg , inm
- Instrucción mem, inm





# Instrucciones - Intel (4)

---

- El nombre destino y fuente proviene del hecho que si hay un movimiento de datos, es desde la derecha (fuente) hacia la izquierda (destino).
- En una suma hay 2 operandos y el resultado se almacena en el lugar del operando izquierdo (destino).

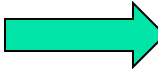


# Instrucciones - Intel 8086

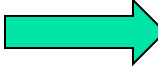
---

Ejemplos:

**ADD AX,BX**        $AX = AX + BX$

**ADD AL,AH**        $AL = AL + AH$

**MOV AL,CH**        $AL = CH$

**SUB AX,BX**        $AX = AX - BX$


❖ Direcccionamiento por registro



# Instrucciones - Intel 8086 (2)

---

Ejemplos:

**ADD AX,35AFh**        $AX = AX + 35AFh$

**ADD AL,15**        $AL = AL + 15$

**MOV AL,3Eh**        $AL = 3Eh$

**SUB AX,1234h**        $AX = AX - 1234h$

❖ **Direccionamiento Inmediato**



# Instrucciones - Intel 8086 (3)

---

Ejemplos:

**ADD AX, [35AFh]**

➡  $AX = AX + \text{contenido direcc. } 35AFh \text{ y } 35B0h$

**ADD AL, DATO**

➡  $AL = AL + \text{contenido variable DATO (8 bits)}$

**MOV NUM1, CH**

➡  $NUM1 = CH$  (la variable de 8 bits NUM1 queda con el mismo contenido que CH)

❖ **Direccionamiento Directo**



# Instrucciones - Intel 8086 (4)

---

Ejemplos:

**ADD AX, [BX]**

→  $AX = AX + \text{dato almacenado en dirección contenida en BX y la que sigue}$

**MOV [BX], AL**

→  $\text{dato en la dirección contenida en BX} = AL$

❖ **Direccionamiento Indirecto por registro**



# Instrucciones - Intel 8086 (5)

---

Ejemplos:

**MOV CX, [BX+SI]**

→ CX = dato almacenado en la direcc. BX+SI y la siguiente

**MOV [BX+DI], AL**

→ dato almacenado en la direcc. BX+DI = AL

❖ Direccionamiento base + índice



# Instrucciones - Intel 8086 (6)

---

Ejemplos:

**MOV AL, [BX+2]**

→ AL=dato almacenado en dir BX+2

**MOV [BX+2Ah], AX**

→ dato almacenado en dir BX+2Ah y la que sigue = AX (16 bits)

❖ Direcccionamiento Relativo por registro



# Instrucciones - Intel 8086 (7)

---

Ejemplos:

**MOV AL, [BX+SI+2]**

→ AL = dato almacenado en la dir BX+SI+2

**MOV [BX+DI+2Ah], AX**

→ dato almacenado en la dir BX+DI+2Ah y la que sigue = AX (16 bits)

❖ Direccionamiento relativo base+índice





# Formatos de instrucción- Criterios de diseño

---

- ✓ ¿Instrucciones cortas ó largas?
- ✓ N° de bits/seg
  - ✓ ancho de banda de la memoria
- ✓ Velocidad procesador/Velocidad memoria
- ✓ Instrucciones más cortas
  - ✓ el procesador “parece” más rápido.



# Formatos de instrucción- Criterios de diseño (2)

---

- ✓ Suficientes bits para expresar todas las operaciones deseadas.
- ✓ La experiencia demuestra dejar bits libres para el futuro.
- ✓ Cantidad de bits de datos.



# Ejemplo para MSX88

---

- Editar prueba.asm
  - Usar Editor de textos
- Ensamblar prueba.asm
  - Usar Asm88
    - Prueba.o y Prueba.lst
- Enlazar prueba.o
  - Usar Link88
    - Prueba.eje
- Usar MSX88
  - Cargar prueba.eje y ejecutar

```
ORG 2000H
MOV BX,3000H
MOV AX,[BX]
ADD BX, 02H
MOV CX,[BX]
ADD AX,CX
PUSH AX
POP DX
HLT
```

```
ORG 3000h
DB 55h, 33h, 44h, 22h
END
```



# Archivo prueba.lst

---

Dir.	Código máquina	Línea	Código en lenguaje ensamble
------	----------------	-------	-----------------------------

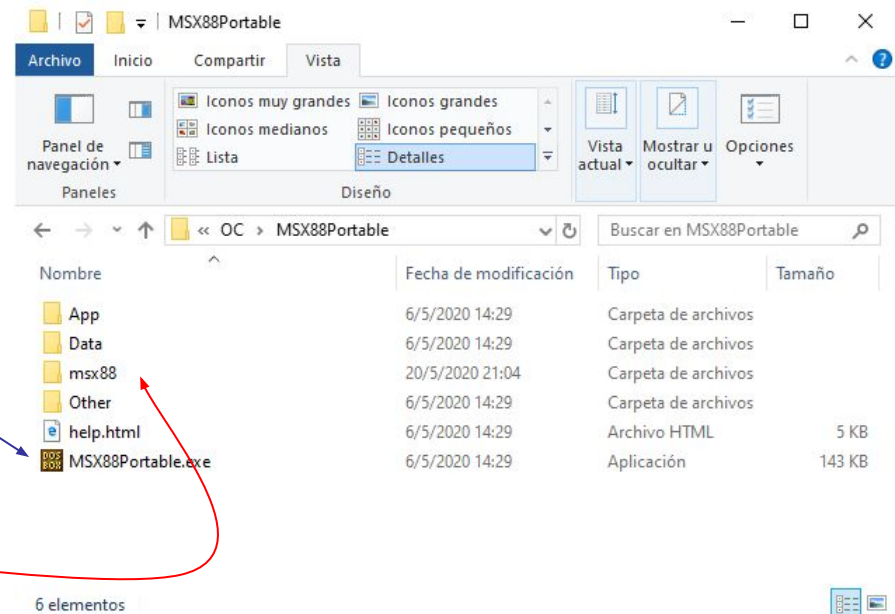
		1	ORG 2000H
2000	BB 00 30	2	MOV BX, 3000H
2003	8B 07	3	MOV AX, [BX]
2005	81 C3 02 00	4	ADD BX, 02H
2009	8B 0F	5	MOV CX, [BX]
200B	03 C1	6	ADD AX, CX
200D	50	7	PUSH AX
200E	5A	8	POP DX
200F	F4	9	HLT
		10	
		11	ORG 3000h
3000	55 33 44 22	12	DB 55h, 33h, 44h, 22h
		13	END

S I M B O L O S:

Nombre:	Tipo:	Valor:
---------	-------	--------

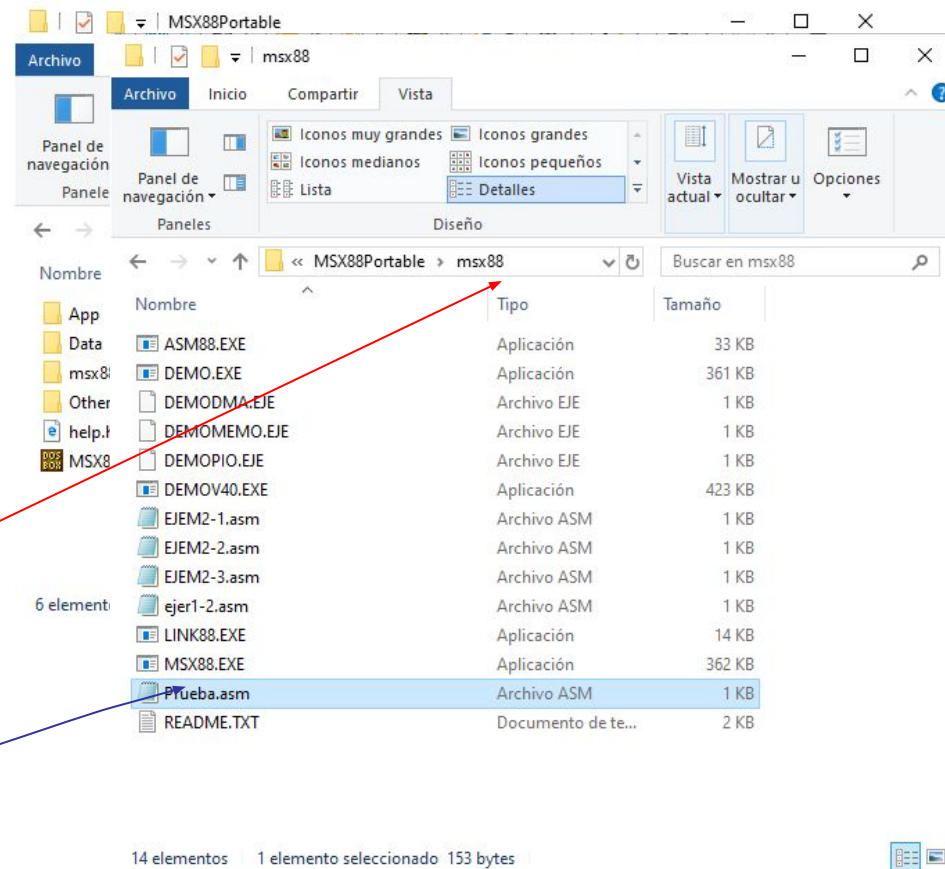
# Guía rápida para el ejemplo MS88

- Descargar MSX88Portable.zip
- Extraer todo en una carpeta  
Esa será la carpeta principal para empezar a trabajar.  
Desde ahí luego ejecutaremos el programa **MSX88Portable.exe** que nos provee el entorno de trabajo para los el simulador, ensamblador y linker (**msx88**, **asm88** y **link88**).
- Dentro de la carpeta principal, está la carpeta **msx88**, que será nuestra carpeta de trabajo. Ahí estarán nuestros archivos (.asm, .lst, .o y .eje)



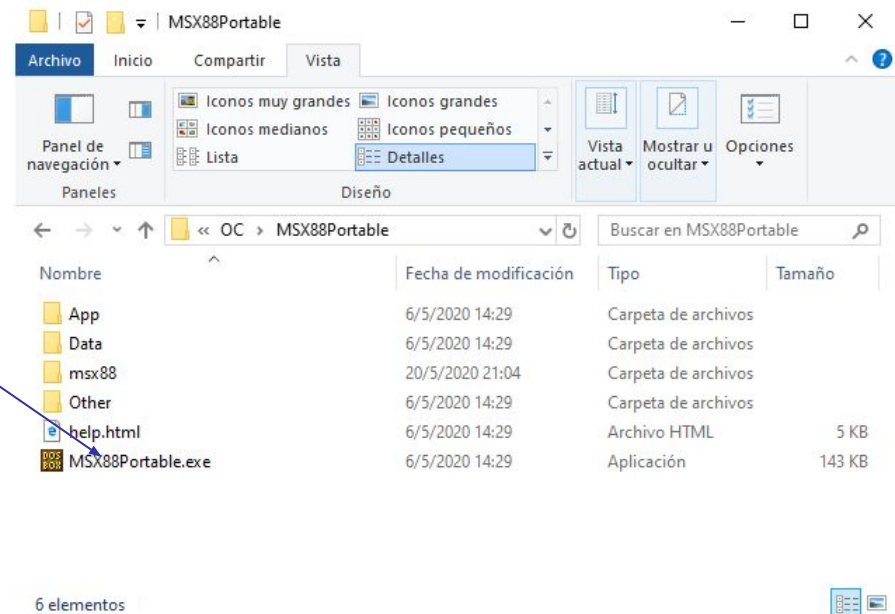
# Guía rápida para el ejemplo MS88

- Descargar MSX88Portable.zip
- Extraer todo en una carpeta  
Esa será la carpeta principal para empezar a trabajar.  
Desde ahí luego ejecutaremos el programa MSX88Portable.exe que nos provee el entorno de trabajo para los el simulador, ensamblador y linker (msx88, asm88 y link88).
- Dentro de la carpeta principal, está la carpeta msx88, que será nuestra carpeta de trabajo. Ahí estarán nuestros archivos (.asm, .lst, .o y .eje)
- Dentro de la carpeta **msx88**, creamos y editamos **prueba.asm** con el block de notas.



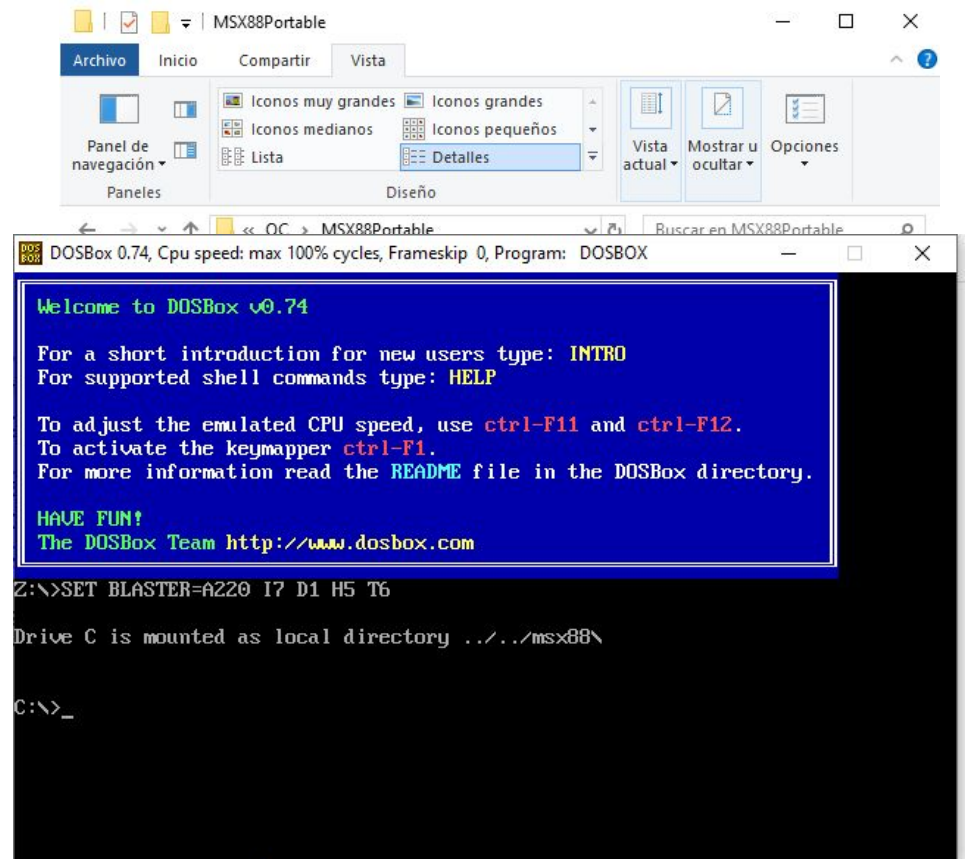
# Guía rápida para el ejemplo MS88

- Ahora desde la carpeta principal, podremos ejecutar el programa **MSX88Portable.exe** que nos provee el entorno de trabajo para el simulador, ensamblador y linker (msx88, asm88 y link88).



# Guía rápida para el ejemplo MS88

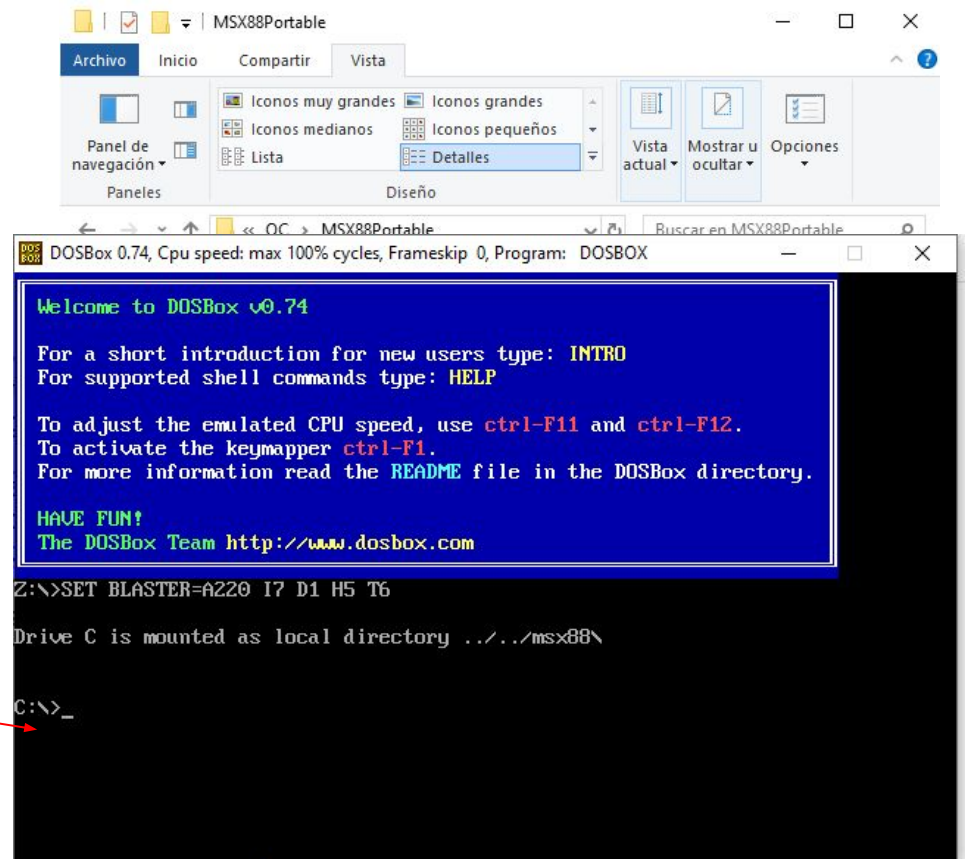
- Ahora desde la carpeta principal, podremos ejecutar el programa **MSX88Portable.exe** que nos provee el entorno de trabajo para el simulador, ensamblador y linker (msx88, asm88 y link88).
- Se abrirán dos ventanas,
  - “DOSBox status windows” que podemos minimizar (no cerrar!)
  - “DOSBox 0.74” que es la ventana de trabajo.





# Guía rápida para el ejemplo MS88

- Ahora desde la carpeta principal, podremos ejecutar el programa MSX88Portable.exe que nos provee el entorno de trabajo para el simulador, ensamblador y linker (msx88, asm88 y link88).
- Se abrirán dos ventanas,
  - “DOSBox status windows” que podemos minimizar (no cerrar!)
  - “DOSBox 0.74” que es la ventana de trabajo.
- En “DOSBox 0.74” podremos escribir **msx88** para correr el simulador, o **asm88** o **link88** para el ensamblador o el linker.



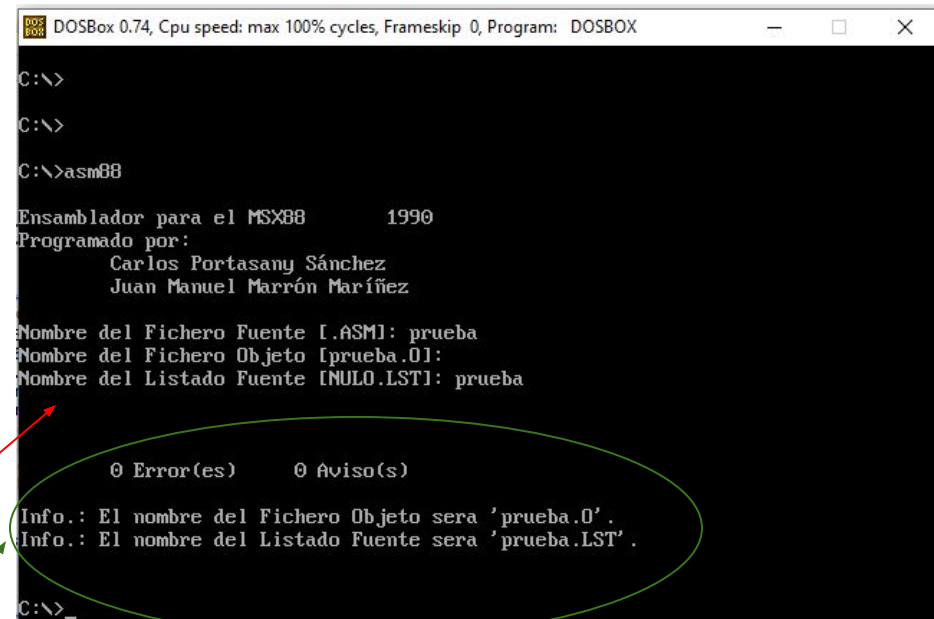
# Guía rápida para el ejemplo MS88

- Ahora desde la carpeta principal, podremos ejecutar el programa MSX88Portable.exe que nos provee el entorno de trabajo para el simulador, ensamblador y linker (msx88, asm88 y link88).
- Se abrirán dos ventanas,
  - “DOSBox status windows” que podemos minimizar (no cerrar!)
  - “DOSBox 0.74” que es la ventana de trabajo.
- En “DOSBox 0.74” podremos escribir **msx88** para correr el simulador, o **asm88** o **link88** para el ensamblador o el linker.



# Guía rápida para el ejemplo MS88

- En "DOSBox 0.74" podremos escribir msx88 para correr el simulador, o asm88 o link88 para el ensamblador o el linker.
- Corremos asm88 para ensamblar nuestro fuente
  - en la opción .asm ingresamos **prueba**
  - en la opción .o simplemente apretar enter
  - escribamos **prueba** cuando nos dé la opción de generar .lst, sino no genera nada.
  - Ya tenemos generados prueba.o y prueba.lst



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: DOSBOX
C:\>
C:\>
C:\>asm88

Ensamblador para el MSX88      1990
Programado por:
    Carlos Portasany Sánchez
    Juan Manuel Marrón Maríñez

Nombre del Fichero Fuente [.ASM]: prueba
Nombre del Fichero Objeto [prueba.O]:
Nombre del Listado Fuente [NULO.LST]: prueba

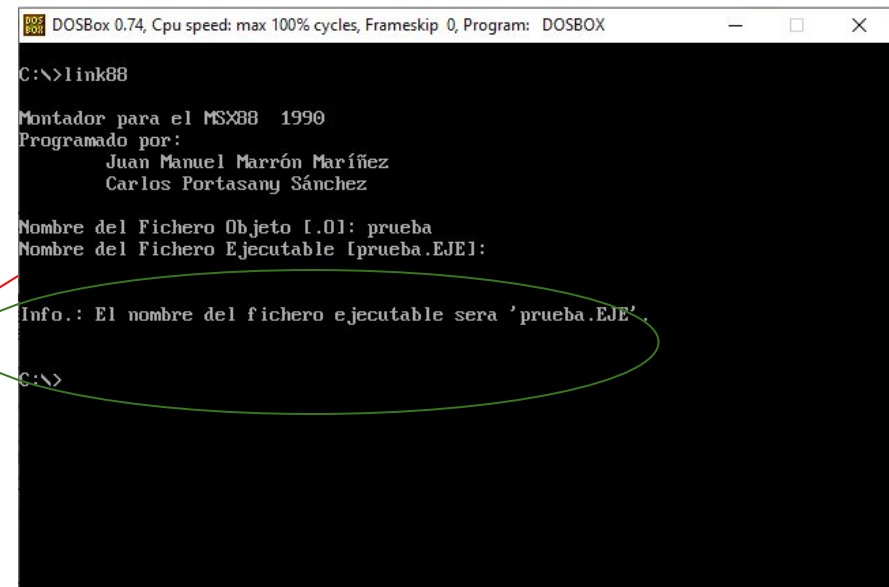
      0 Error(es)      0 Aviso(s)
Info.: El nombre del Fichero Objeto sera 'prueba.o'.
Info.: El nombre del Listado Fuente sera 'prueba.LST'.
C:\>
```

- 



# Guía rápida para el ejemplo MS88

- En "DOSBox 0.74" podremos escribir `msx88` para correr el simulador, o `asm88` o `link88` para el ensamblador o el linker.
- Corremos `link88` para enlazar nuestro archivo `prueba.o`
  - en la opción `.o` escribir **prueba** y apretar enter
  - en la opción `.exe` simplemente apretar enter.
  - Ya tenemos generado `prueba.exe`



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: DOSBOX
C:\>link88

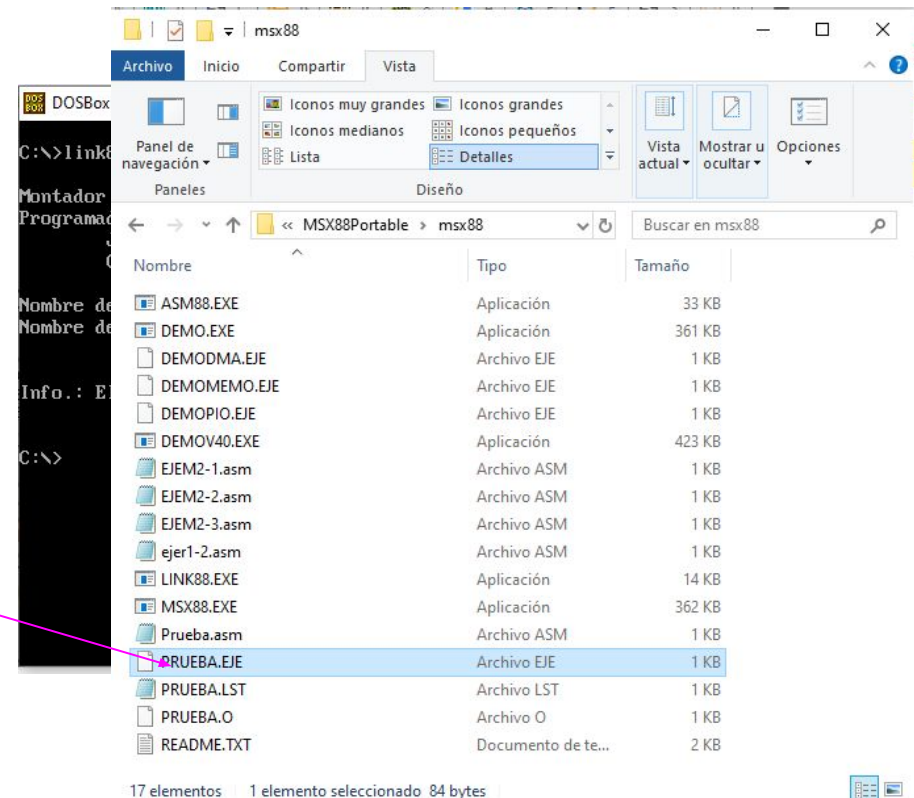
Montador para el MSX88 1990
Programado por:
    Juan Manuel Marrón Maríñez
    Carlos Portasany Sánchez

Nombre del Fichero Objeto [.O]: prueba
Nombre del Fichero Ejecutable [prueba.EJE]:

Info.: El nombre del fichero ejecutable sera 'prueba.EJE'.
C:\>
```

# Guía rápida para el ejemplo MS88

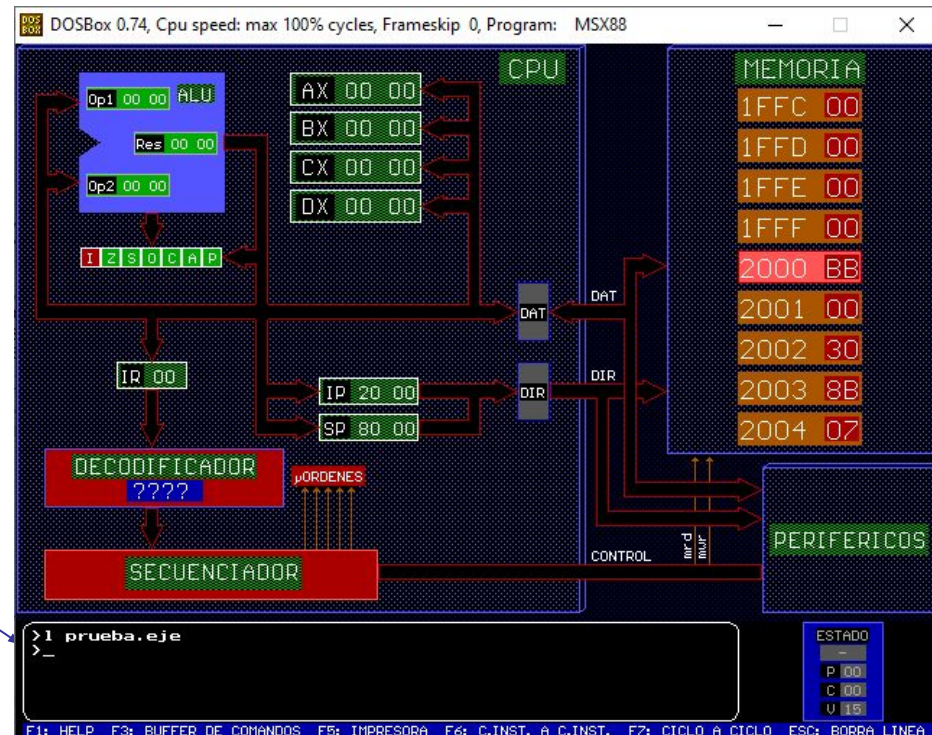
- En "DOSBox 0.74" podremos escribir msx88 para correr el simulador, o asm88 o link88 para el ensamblador o el linker.
- Corremos link88 para enlazar nuestro archivo prueba.o
  - en la opción .o escribir **prueba** y apretar enter
  - en la opción .eje simplemente apretar enter.
  - Ya tenemos generado prueba.eje





# Guía rápida para el ejemplo MS88

- En "DOSBox 0.74" podremos escribir msx88 para correr el simulador, o asm88 o link88 para el ensamblador o el linker.
- Corremos el simulador msx88
  - con el comando l cargamos el prueba.eje

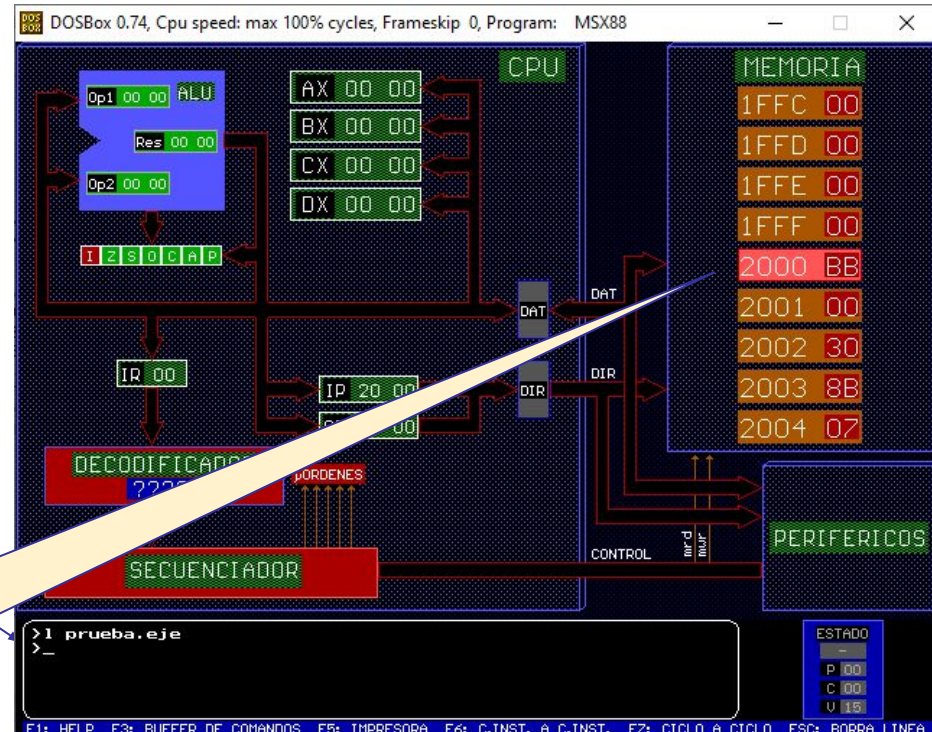


# Guía rápida para el ejemplo MS88

- En "DOSBox 0.74" podremos escribir msx88 para correr el simulador, o asm88 o link88 para el ensamblador o el linker.
- Corremos el simulador msx88
  - con el comando l cargamos el prueba.eje

Puede observarse como se cargó en memoria el programa prueba.eje

Ya se puede empezar la ejecución por ej con F6







## mas información ...

---

- Organización de los registros
  - Capítulo 11 apartado 11.2. Stallings, W., 5º Ed.
- Formatos de instrucciones
  - Capítulo 10 apartado 10.3.y 10.4 Stallings, W., 5º Ed.
- Links de interés
  - [http://www.intel.com/museum/online/hist\\_micro/hof/index.htm](http://www.intel.com/museum/online/hist_micro/hof/index.htm)
- Simulador MSX88
  - En Descargas de página web de cátedra