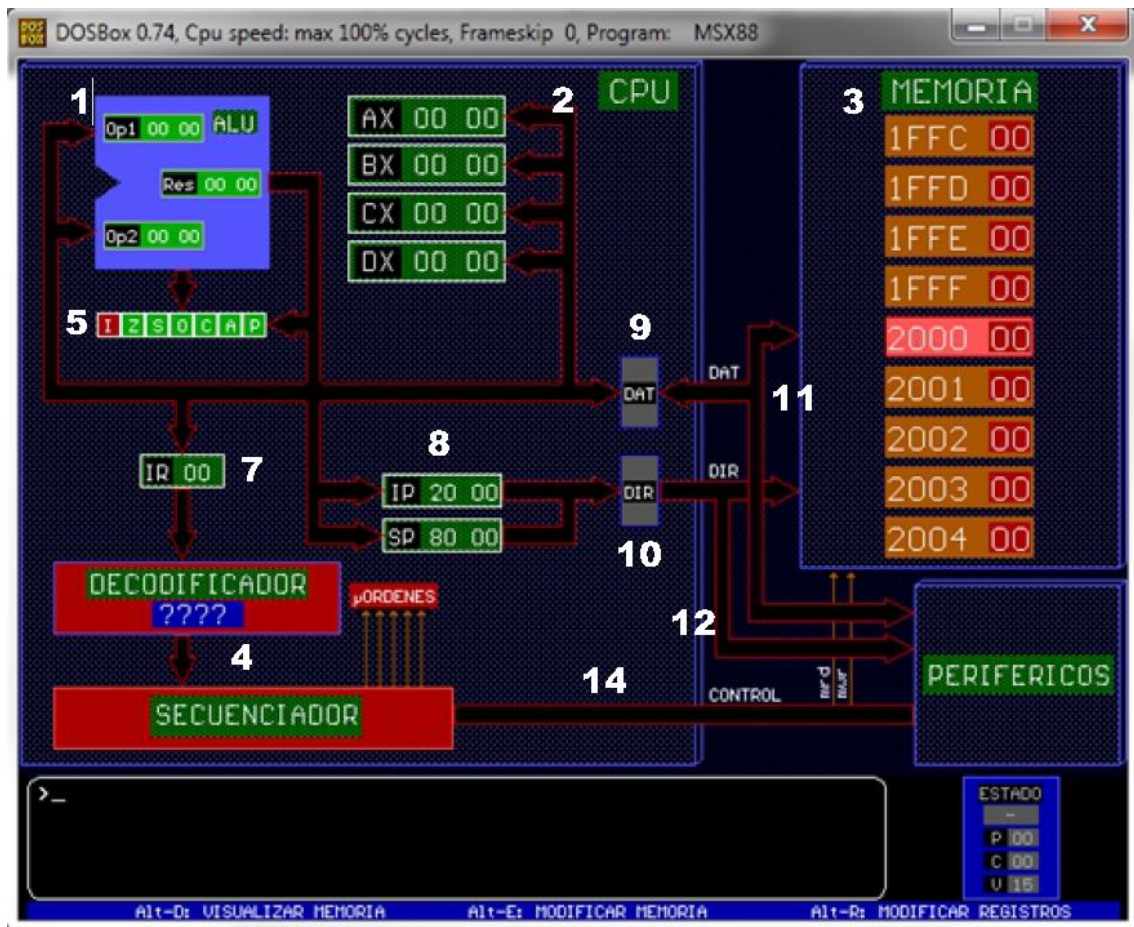


## Ejercicios

1. Describe brevemente cada uno de sus componentes.



- 1- ALU : (Arithmetic Logic Unit), se encarga de realizar las operaciones aritméticas básicas (Suma, Resta, División y Multiplicación) además de realizar algunas operaciones Lógicas (Yes, Or, Not, And – Es decir, si; y, o, no) entre dos números o dos conjuntos de números.
- 2- Registro de uso general: Un bloque de memoria donde se almacena el resultado de las operaciones aritméticas de la ALU.
- 3- Memoria: es donde se guardan los datos en cada dirección.
- 4- Unidad de control: genera caracteres alfanuméricos en el lenguaje máquina en un orden.
- 5- Set de instrucciones: dirige el tráfico de datos.
- 7- Registro de instrucción
- 8- Contador de programa (IP)
- 9- Datos
- 10- Direcciones
- 11- Bus de datos
- 12- Bus de direcciones
- 14- Bus de control

## 2. Definición de variables

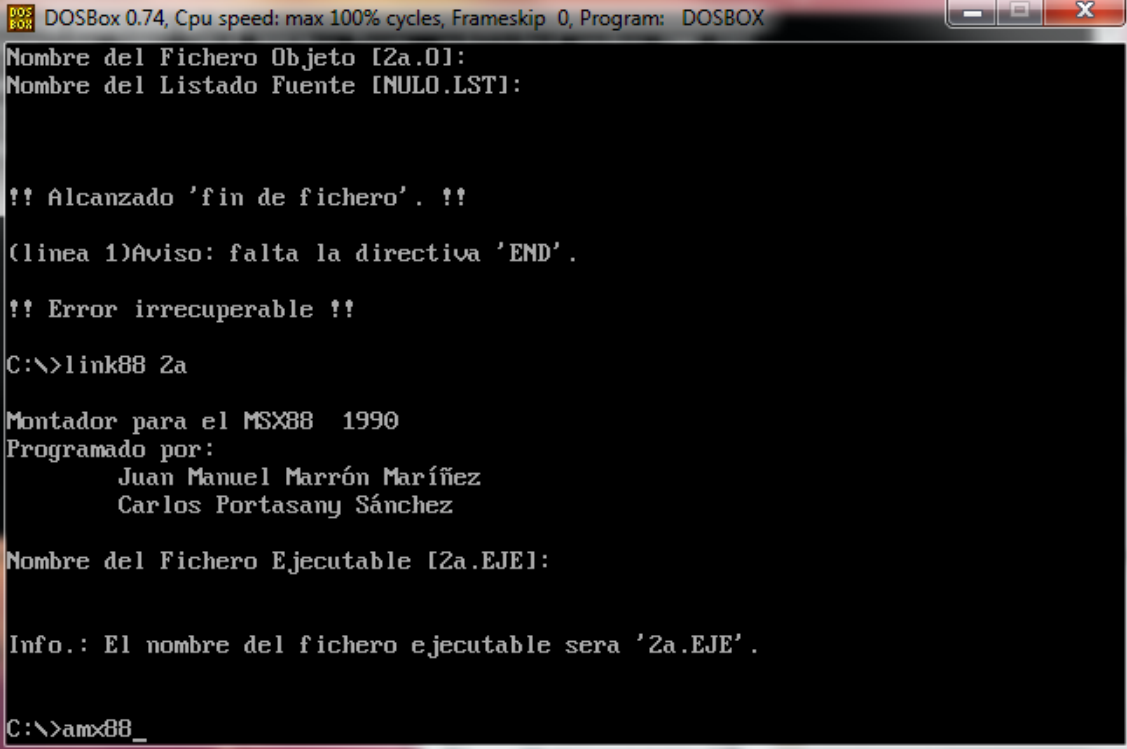
- Define una variable tipo Byte (8 bits) llamada var1 con un valor inicial numérico 50.

*Variable DB valor.*

Para declarar la variable en el fichero que acaba en .asm, pongo Var1 dB 50.

```
1  ORG 1000h
2  var1 DB 50
3
4  END
5
```

Lo guardo en la carpeta msx88, en el programa se introduce como comando "asm 2a" que es el nombre del fichero que he creado y en las dos siguientes fases le doy "enter" luego lo linkeo y me genera el fichero.eje .



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: DOSBOX
Nombre del Fichero Objeto [2a.O1:
Nombre del Listado Fuente [NULO.LST]:

?? Alcanzado 'fin de fichero'. ??
(linea 1)Aviso: falta la directiva 'END'.
?? Error irreparable ??
C:\>link88 2a

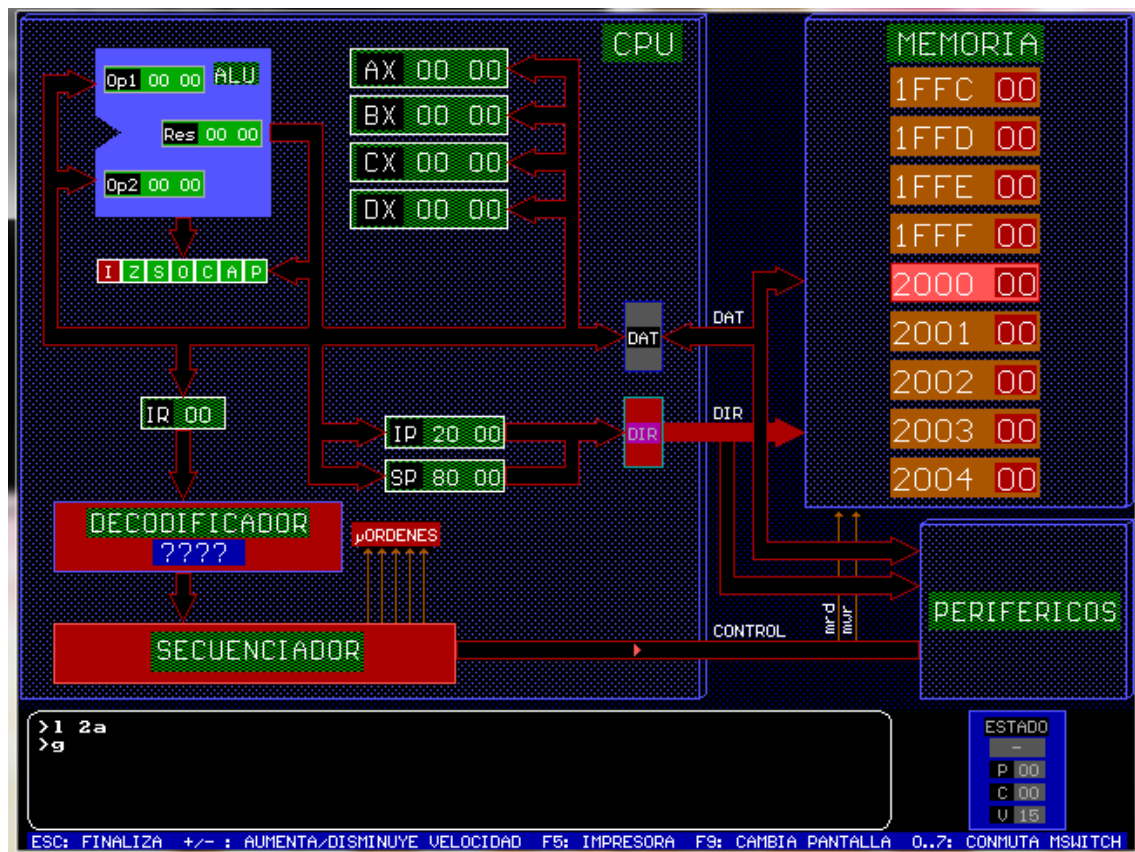
Montador para el MSX88 1990
Programado por:
    Juan Manuel Marrón Maríñez
    Carlos Portasany Sánchez

Nombre del Fichero Ejecutable [2a.EJE]:

Info.: El nombre del fichero ejecutable sera '2a.EJE'.

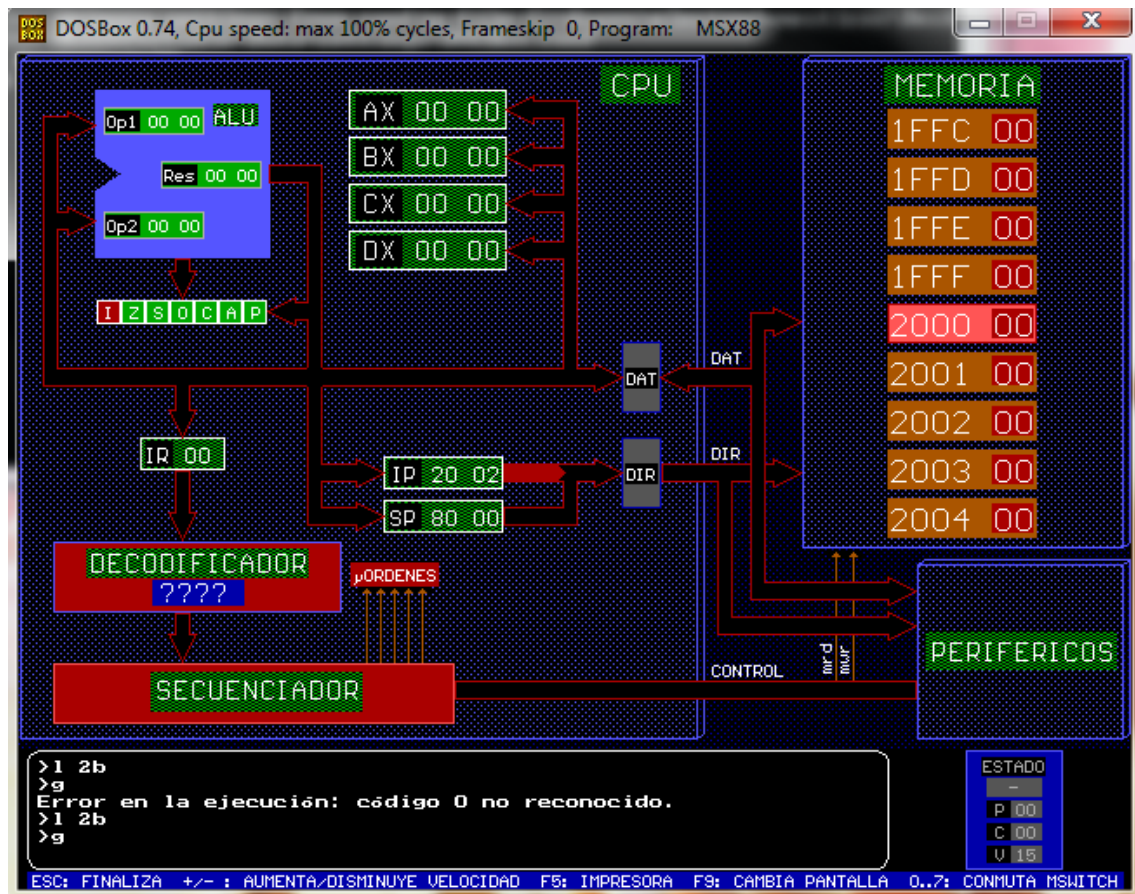
C:\>amx88_
```

Lo ejecuto en el programa msx88 y me queda esta captura.



b. Defina una variable tipo Word (16 bits) llamada var2 sin un valor inicial.  
Variable DW valor

Al igual que el apartado anterior creo un archivo.asm cambiando en el contenido la DB por DW y el valor se deja en blanco, hago los mismos pasos explicados anteriormente y la captura de pantalla final me queda esta.



3. ¿Cuál es la directiva que permite cambiar la dirección a partir de la cual se asignarán a las variables e instrucciones? Pon ejemplos.

Es la directiva **ORG**. Esta directiva dice al ensamblador a partir de qué posición de memoria de programa se situarán las siguientes instrucciones. Rutinas de comienzo, subrutinas de interrupción y otros programas deben comenzar en locaciones de memoria fijados por la estructura del microprocesador.

La directiva **ORG** hace al compilador colocar el código que le sigue en una nueva dirección de memoria (la salida del compilador no solo coloca los códigos de operación sino también las direcciones de cada instrucción del programa). Usualmente se la utiliza para: reset, programas de servicios de interrupción, programa principal, subrutinas.

Ejemplos:

- Inicia el programa en la posición cero:  
`ORG 0x00`
- Inicia el programa en la posición 0000h y luego pasa a la 0005h para no utilizar la posición del vector de interrupción (0004 h)

`ORG 0x00 ; El programa comienza en la dirección 0 y`

```
GOTO inicio ; salta a la dirección 5 para sobrepasar
ORG 0x05 ; el vector de interrupción, situado en la posición 4
Inicio xxx...
```

- 4.Cuál es la instrucción que permite realizar movimientos de datos, entre registros o de un registro a la memoria? Pon ejemplos.

La instrucción es MOV. Esta es una instrucción en el lenguaje ensamblador de la mayoría de procesadores, cuyo propósito es la transferencia de datos entre registros de procesador o registro y memoria.

Adicionalmente MOV también permite el uso de datos absolutos, como por ejemplo mover el número 10 a un registro del procesador.

```
Ejemplo: MOV AX, 0006h
MOV BX, AX
MOV AX, 4C00h
INT 21H
```

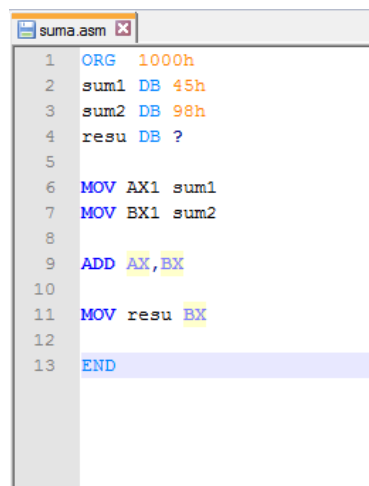
5. Si queremos sumar dos valores, 45 y 98...

- a. ¿Qué instrucción debo utilizar para realizar la suma?

ADD

- b. Define variables que contengan los valores y realiza la suma.

En la siguiente imagen lo muestro



```
suma.asm
1  ORG 1000h
2  sum1 DB 45h
3  sum2 DB 98h
4  resu DB ?
5
6  MOV AX1 sum1
7  MOV BX1 sum2
8
9  ADD AX,BX
10
11 MOV resu BX
12
13 END
```

- c. ¿En cuál de las variables se guarda el resultado?

En la variable resultado

- d. ¿Qué registros cambian durante la operación?

BX y AX

6. Si queremos resolver la resta  $98 - 45$ :

a. ¿Qué instrucción debo utilizar para realizar la resta?

SUB

b. Define variables que contengan los valores y realiza la resta.

En la siguiente variable la muestro

```
1  ORG 1000h
2  dato1 DW 98
3  dato2 DW 45
4  resultado DW ?
5
6  ORG 2000h
7  MOV AX, dato1
8  ADD AX, dato2
9  MOV resultado, AX
10 END
```

c. En cuál de las variables se guarda el resultado?

En AX

Nota: ninguno de los dos códigos expuestos anteriormente me han sido ejecutables. La suma no abre el fichero porque no lo reconoce, la resta no da error tan pronto pero acaba dándolo también. Desconozco el motivo del fallo.

7. Incrementa la variable var1 del tipo byte definida con el valor 0.

```
Var1, AL 0
DEC Var1
```

8. Describe que realiza el siguiente código:

**ORG 2000h** **ORG** es el espacio de memoria donde se va a trabajar, la dirección.

**MOV AX, 20** designa el valor de AX (variable de 16 bit) en 20.

**MOV BX, 2** designa el valor de BX en 2

**Lazo:**

**ADD BX, AX:** se suma BX y AX

**DEC AX se:** decremento AX

**JNZ Lazo:** JNZ vuelve a lazo mientras que el valor último calculado sea 0, es decir 20 veces hasta que de 0 y salga del bucle.

**Fin:** JNZ llega al fin cuando es 0

**JMP:** Fin es otro tipo de salto

**END:** final del código

a. ¿Al pasar a la etiqueta Fin que se produce y porque se realiza de dicha manera? Como se podría subsanar este inconveniente?.



Cuando llega a fin se ejecuta un salto incondicional a la etiqueta fin, con lo cual se produce un bucle. Se produce porque JRP.

Eliminando por completo la etiqueta fin.

b. ¿Qué tipo de salto es JNZ?

JNZ es un salto condicional al valor de 0, en este caso si el valor no es 0.

9. ¿Cuál es la diferencia entre **BX** y **[BX]**?

Cuando BX va entre corchetes quiere decir que va a ser opcional ponerlo o no.

10. Analice y explique que realiza el siguiente código:

**MOV AL, 0:** asigna el valor a AL en 0.

**MOV CL, 10:** asigna el valor a CL en 10.

**Iterar: CMP CL, 1:** CMP es comparar, iterar es una etiqueta entre comillas, por tanto la función es comparar CL con 1.

**JZ Fin:** dice si el último valor calculado es 0 saltaría a la etiqueta fin, lo que provoca que con la instrucción HLT detendría la ejecución del procesador.

**ADD AL, AL:** se suma AL y AL.

**DEC CL:** se decrementa CL.

**JMP Iterar:** Salto incondicional a Iterar, produciendo un bucle. El bucle termina cuando CL valga 0.

**Fin: HLT:** se acaba la función del código.