DEVASM-Z80

Entorno de Desarrollo Integrado en Ensamblador para Plataforma ZX-Spectrum basado en PASMO

Las “razas” no existen,

existen solo en las mentes de los racistas.

***Luigi Luca Cavalli-Sforza**, fallecido el pasado mes de agosto de este mismo año 2018, es considerado el principal experto mundial en la diversidad genética de las poblaciones humanas y cuyos estudios concluyeron que la cultura y la lengua son más determinantes que los genes, desmontando por completo el concepto de raza.*

índice de contenidos

GUÍA RÁPIDA *v 1.5* PARA CREAR UN ENTORNO CRUZADO DE DESARROLLO INTEGRADO (IDE) PARA LA PLATAFORMA SINCLAIR ZX-SPECTRUM CON ***ConTEXT***, ***PASMO*** Y ***SPECTACULATOR***

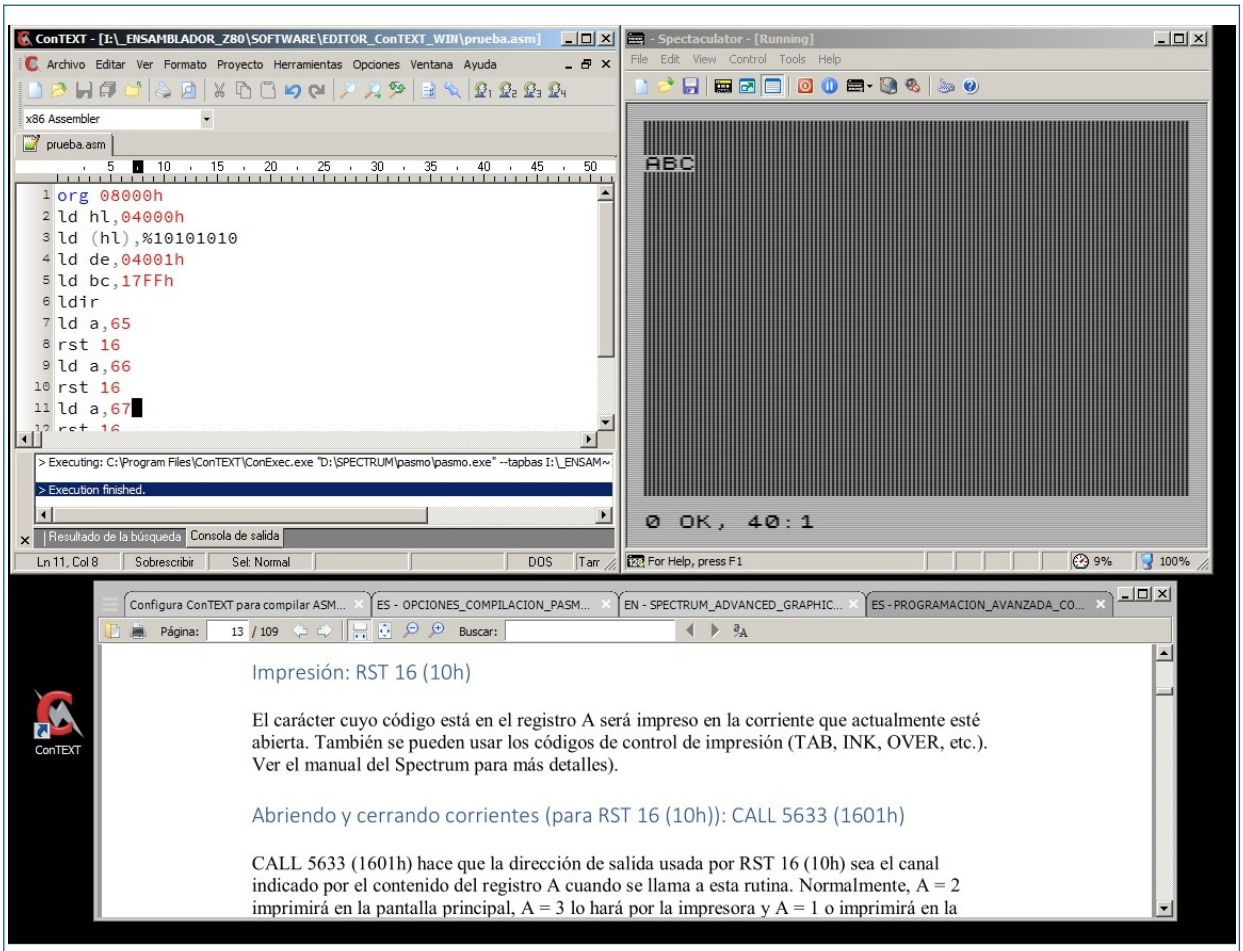
Si algún día de tu vida decides adentrarte en el mágico mundo de la programación en lenguaje ensamblador, puedes optar por varias vías para escribir tu código en ensamblador y traducirlo a código máquina, esta que vamos a detallar es sólo una de ellas.

Nuestro objetivo es conseguir un entorno de desarrollo rápido que integre las herramientas necesarias para poder escribir el código y ejecutarlo de la forma más cómoda posible. Para ello nos serviremos de un editor gratuito y bastante potente llamado ***ConTEXT***, del compilador ***PASMO*** y del emulador ***SPECTACULATOR 5.3*** (última versión gratuita liberada de este emulador para Windows de ZX-Spectrum). Resumiendo, para montar nuestro IDE solo necesitamos TRES aplicaciones que corren sobre *Microsoft Windows* y que son toltamente gratuitas. Todas ellas están disponibles para su descarga en la web del proyecto ***ZXOPENSOURCE*** al que llegarás fácilmente desde *Google*.

- ***ConText*** Text Editor **http://www.contexteditor.org/**
- ***Pasmo*** Assembler (Pasmo)
- ***Spectaculator*** Emulator **http://www.spectaculator.com/**

Por otro lado, aunque las 3 herramientas citadas funcionan de forma portable, en el caso del editor ***ConTEXT*** es necesario que sea instalado en el sistema operativo destino ya que de lo contrario puede experimentar problemas en la integración del compilador ***PASMO***.

Una vez creadas las asociaciones necesarias entre el editor y las otras dos aplicaciones (compilador y emulador) ya podremos empezar a picar código y compilarlo directamtente mediante la tecla **F9** y ejecutarlo en nuestro emulador mediante la tecla **F10**, todo ello prácticamente de forma inmediata tal y como se muestra en la imagen siguiente.



Resultado final del IDE DEVASM-Z80 una vez creado configurado el editor ConTEXT y conectado con el compilador PASMO y con el emulador SPECTACULATOR. La ventana inferior solo muestra un documento de consulta que nada tiene que ver con el entorno de desarrollo.

Para explicar como crear y configurar el entorno seguiremos los pasos siguientes **al pie de la letra**, aunque si quieres puedes probar con otras configuraciones de compilador y/o emulador. Vamos a ello, verás que no reviste mayor complicación.

1.- Instala en tu sistema operativo *Windows®* el editor ***ConTEXT*** sirviéndote para ello del instalador del mismo. Si te decides a utilizar una versión portable es posible que experimentes problemas con la integración/asociación de aplicaciones externas derivadas debido a las rutas, por lo que recomiendo instalar en el sistema operativo, al menos, el editor ***ConTEXT***. Una vez instalado y antes de empezar a trabajar con él, puedes configurarlo a tu gusto en cuanto a colores, idioma, etc.

2.- Ahora asociaremos los ficheros con extensión ***.ASM*** (que serán los que contengan el código fuente) al editor ***ConTEXT*** para que puedas abrirlos desde el explorador de *Windows®* directamente. Esto es sencillo, haz Clic! con el botón derecho del ratón sobre cualquier fichero con extensión ***.ASM*** y selecciona la opción **ABRIR CON...** Después, en el cuadro de diálogo que muestra *Windows®* deberás indicar la aplicación con la que deseas abrir dicho tipo de fichero, es decir, ***ConTEXT***. Si no aparece en la lista de aplicaciones puedes buscarla mediante la opción/botón **EXAMINAR…** e indicar la ruta completa hasta la aplicación ejecutable ***ConTEXT*** que, normalmente, estará instalada en la carpeta del sistema ***ARCHIVOS DE PROGRAMA*** ó ***PROGRAM FILES***.

3.- Ahora instala el compilador ***PASMO*** en una carpeta de tu disco duro. En realidad, ***PASMO*** es un software portable que no requiere instalación alguna, por lo que tan solo deberás copiar la carpeta que lo contenga en tu disco duro ó PENDRIVE, sí, también podemos trabajar desde una unidad externa. Como normalmente lo descargarás en formato comprimido ***.ZIP***, solo deberás descomprimirlo para poder usarlo.

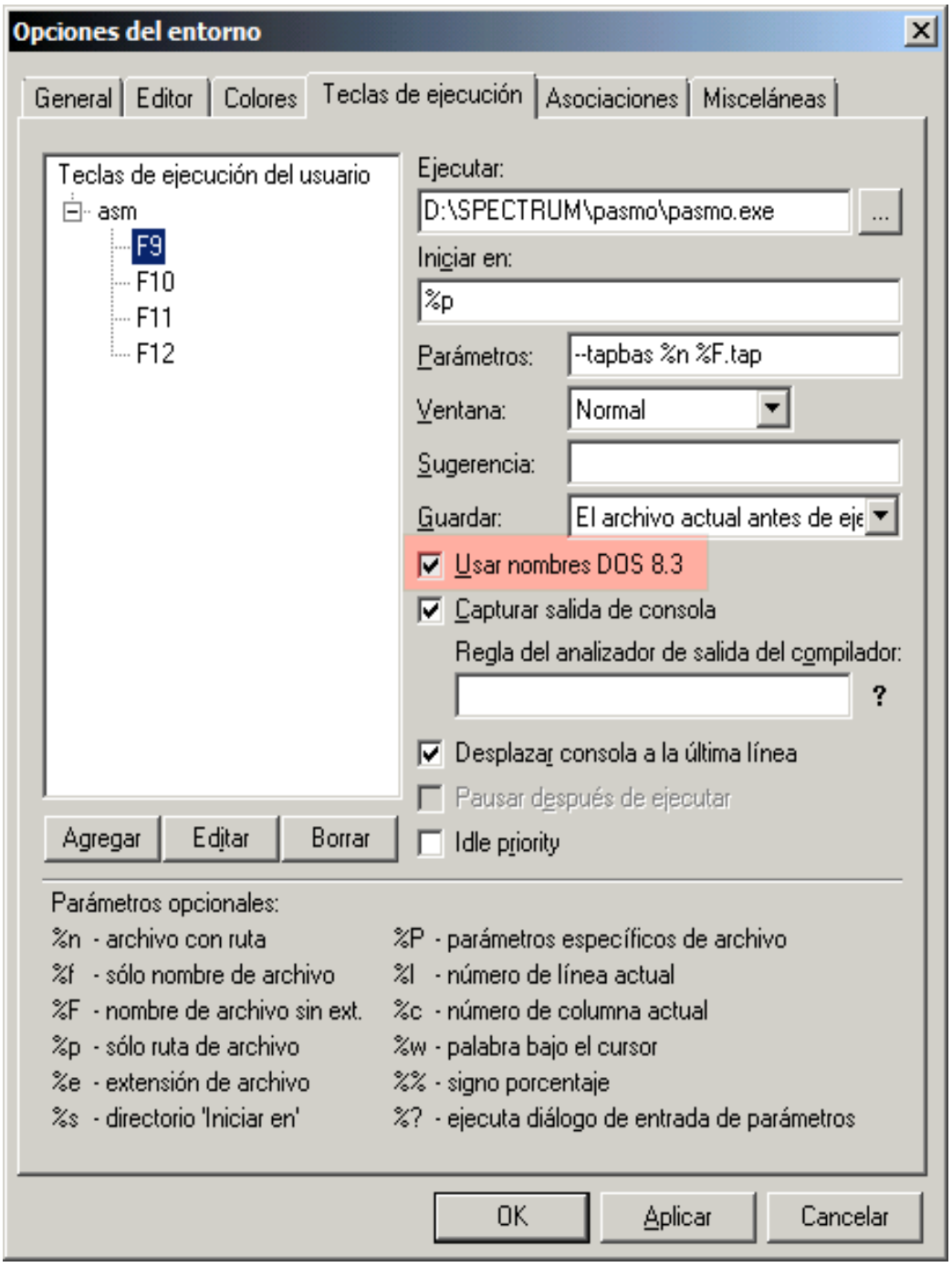
4.- Ahora instala el emulador ***SPECTACULATOR*** en una carpeta de tu disco duro. En realidad, ***SPECTACULATOR*** es un software portable que no requiere instalaciones extra en el sistema, por lo que si lo descargas en formto portable y comprimido en ***.ZIP***, solo deberás descomprimirlo para poder usarlo.

5.- Ahora debes configurar el propio ***ConTEXT*** para editar los archivos **ASM** de la siguiente forma:

1.- Seleccione el menú ***Opciones -> Opciones del entorno***. Se mostrará un cuadro de diálogo.
2.- Seleccione ahora la pestaña ***Teclas de Ejecución.***
3.- Pulsa ***Añadir*** y añade una nueva entrada para archivos con extensión ***.ASM***

Una vez hecho esto, ya estamos preparados para asociar y conectar el editor ***ConTEXT*** con las aplicaciones externas que deseamos, en nuestro caso, el compilador de código ***PASMO*** y el emulador ***SPECTACULATOR***.

6.- Ahora vamos a asociar la tecla **F9** al compilador ***PASMO*** de tal manera que realice la compilación del código fuente que tengamos en el fichero ***.ASM*** que estemos editando. Para ello deberás pulsar/seleccionar la tecla **F9** e introducir los siguientes datos en el cuadro de diálogo de ***ConTEXT,*** asegurándonos que en el cuadro de texto ***Ejecutar*** seleccionamos la ruta completa hasta el compilador ***PASMO*** que ya tenemos en nuestro disco duro.



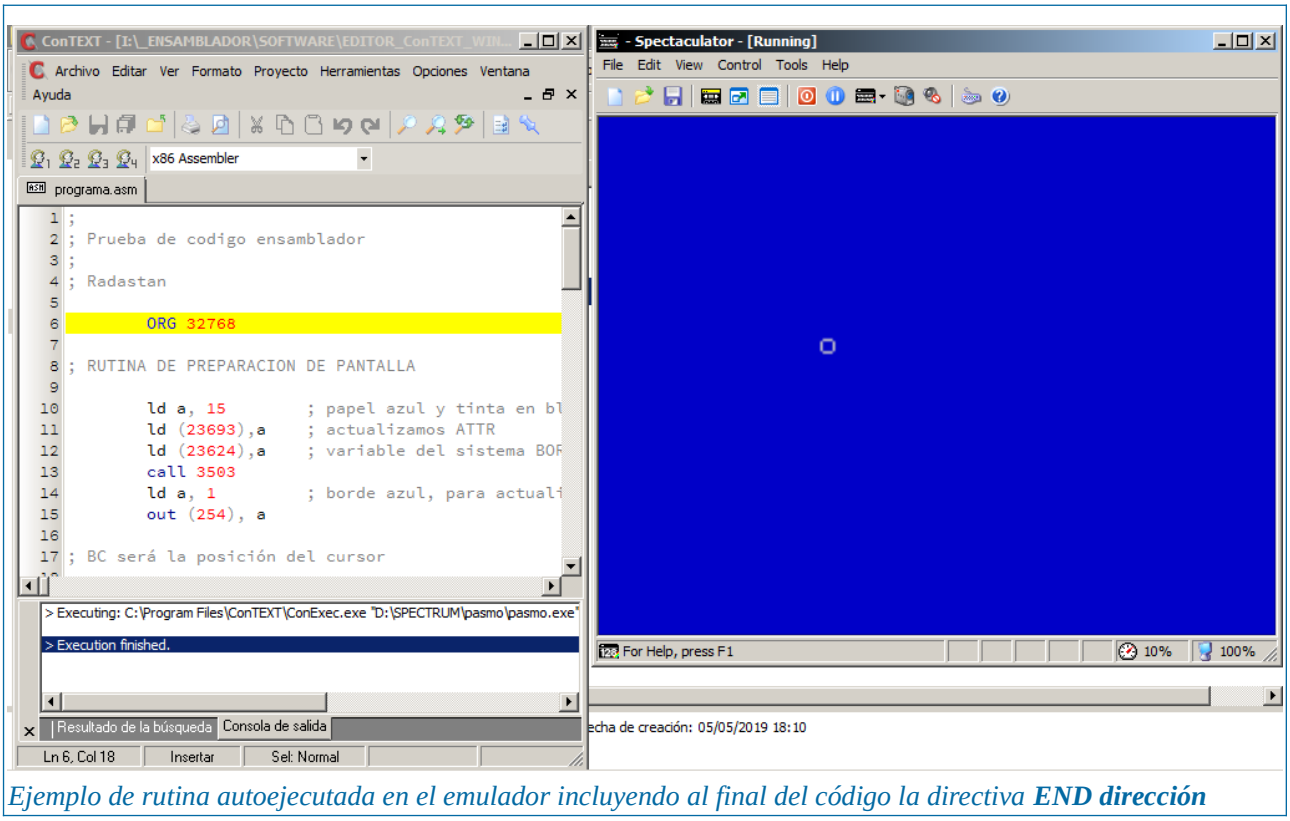
> **IMAGEN ANTERIOR:** En la línea de parámetros introducimos los operadores deseados para el compilador, en nuestro caso **`--tapbas %n %F.tap`** (respeta las minúsculas).
>
> Al pasar estos parámetros al compilador conseguimos que se genere un fichero con extensión ***.TAP*** y el correspondiente cargardor BASIC para poder ejecutarlo directamente. (consulta ***Anexo-2*** para más detalles).
>
> Recuerda que si activas ***USAR NOMBRES DOS 8.3*** en esta ventana correspodiente a la tecla **F9** y que lanza el compilador ***PASMO***, deberás hacer lo mismo en la ventana de configuración del emulador que corresponde a la tecla **F10**.

Si nuestro código compilado no se autoejecuta en el emulador al llamar a la tecla **F10**, lo que nos obligaría a ejecutar la rutina manualmente mediante un comando ***RANDOMIZE USR dirección***, podemos solucionar esto concluyendo el código fuente con la directiva ***END dirección***, de manera que el código de nuestro programa quede comprendido entre las directivas ***ORG*** y ***END***

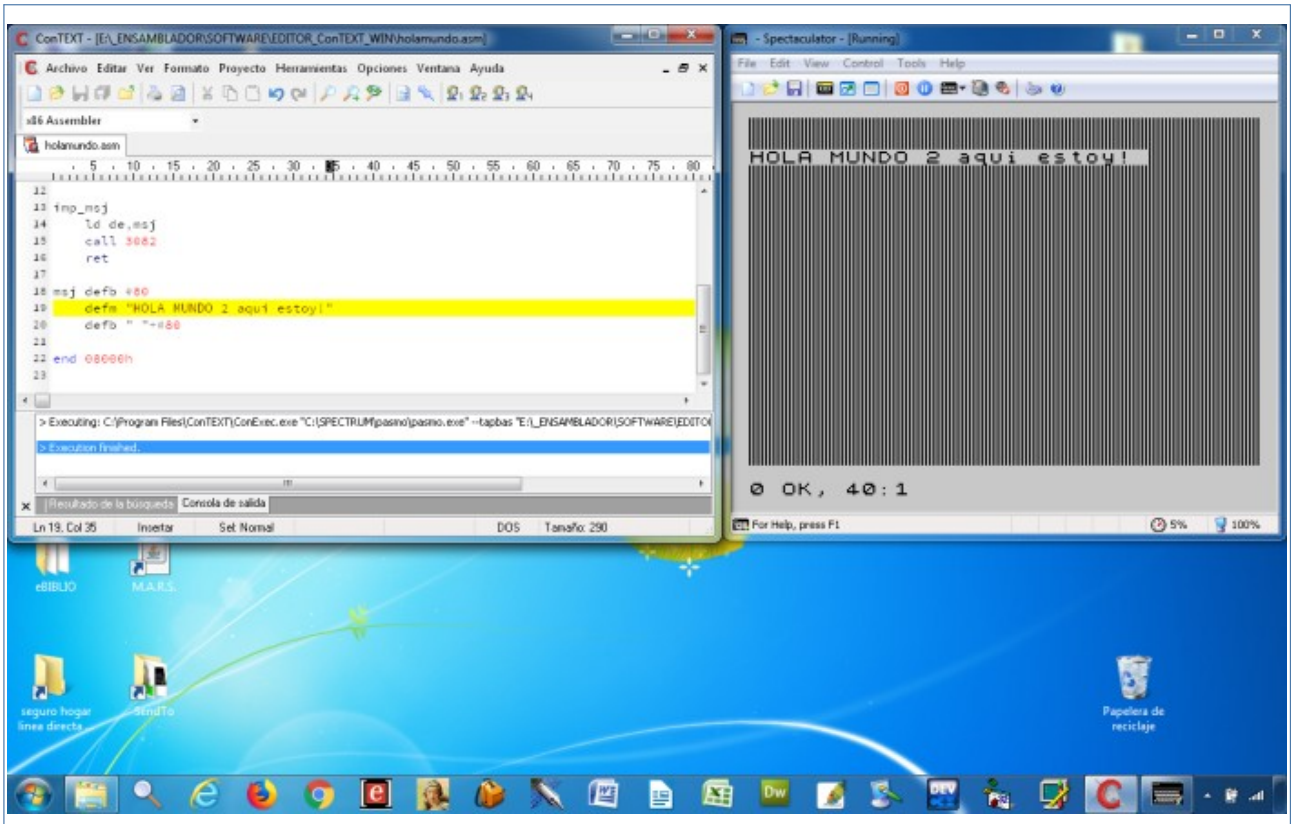
```
ORG 30000
    ... código fuente ...
END 30000
```

Ten presente que estas directivas son soportadas por ***PASMO*** pero podrían no funcionar si utilizas otro compilador distinto.

En última instancia, si continúas experimentando algún problema en este sentido, puede que se deba a alguna particularidad de tu emulador, ten presente que no hay dos emuladores iguales, por lo que te recomiendo que revises la configuración de éste a fondo y sobre todo lo que se refiere a la carga de archivos ***.TAP*** ó, si no consigues los efectos deseados, probar con otro emulador.

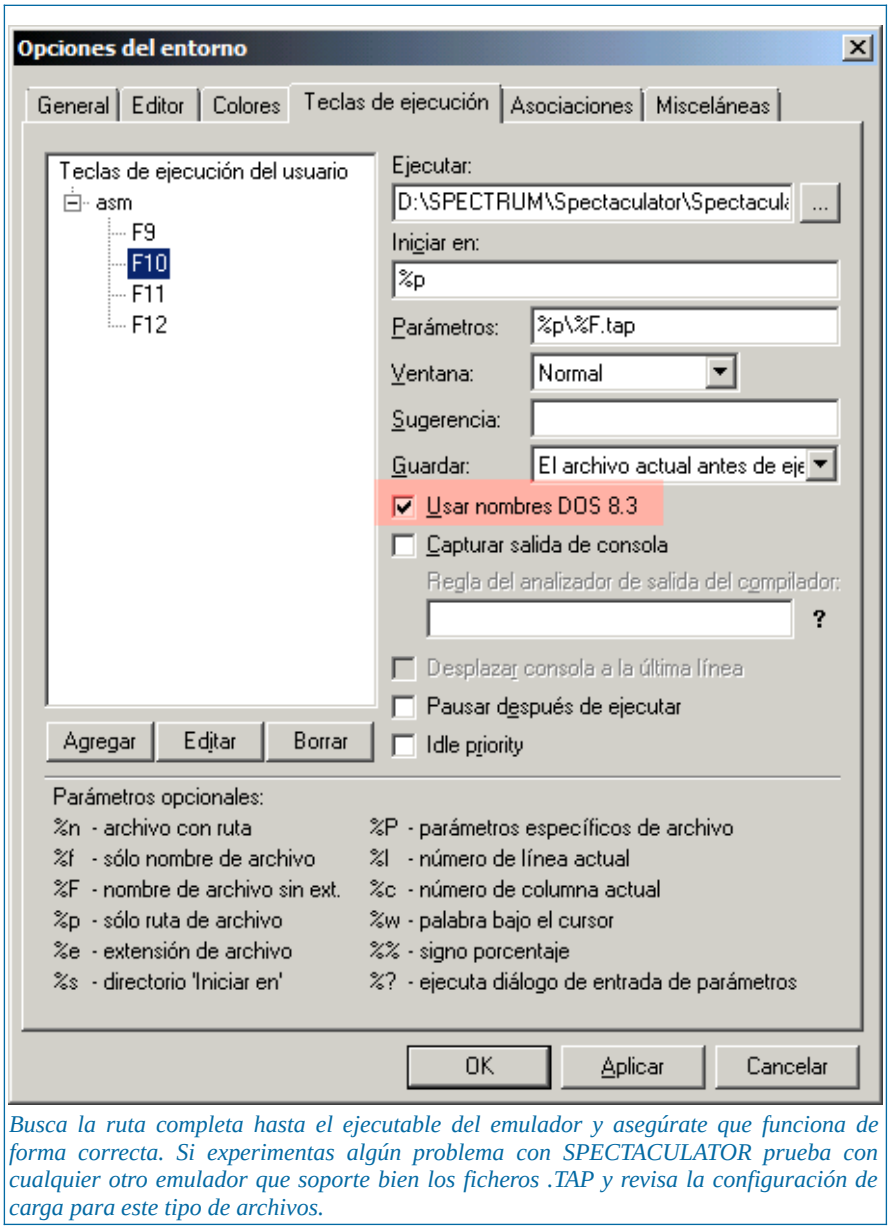


*Ejemplo de rutina autoejecutada en el emulador incluyendo al final del código la directiva **END dirección***



Ejemplo de integración del entorno DEVASM-Z80 en funcionamiento ejecutando el clásico mensaje.

7.- Completado este paso anterior, ya solo nos queda proceder con la asociación de la tecla **F10** al emulador que se encargará de correr el código una vez compilado, en nuestro caso, *SPECTACULATOR*, que aunque puede presentar ciertos fallos en determinadas versiones del sistema *Windows®*, creo que resulta bastante solvente en la mayoría de las circunstancias. Para ello debes pulsar la opción **F10** y volver a cumplimentar los campos indicados en la tal y como se muestra en la captura siguiente:



Busca la ruta completa hasta el ejecutable del emulador y asegúrate que funciona de forma correcta. Si experimentas algún problema con SPECTACULATOR prueba con cualquier otro emulador que soporte bien los ficheros .TAP y revisa la configuración de carga para este tipo de archivos.

Bueno, pues ya podemos empezar a programar y editar código ensamblador creando un fichero con extensión ***.ASM***. Para compilar nuestro código pulsaremos **F9** y para ver el resultado en el emulador, si el proceso de compilado se ha completado sin errores, **F10**

En la carpeta del editor ConTEXT encontrarás algún fichero fuente ASM que podrás utilizar de prueba durante la configuración de tu entorno. También puedes tomar código de cualquier otro lado, siempre que esté escrito en lenguaje ensamblador del Z80 y para la plataforma ZX-Spectrum, eso sí, es imprescindible que tengas presente las directivas incluidas en el código fuente, ya que el uso de directivas/pseudoinstrucciones no soportadas por ***PASMO*** dará errores en la compilación del código pues no todos los programas ensambladores soportan las mismas.

En este sentido, para otorgar de una mayor funcionalidad al documento, he decidido incluir las reglas de codificación del compilador ***PASMO*** en las siguientes páginas.

ANEXOS INCLUIDOS

Anexo 1.

REGLAS DE CODIFICACIÓN PARA PASMO

Anexo 2.

OPCIONES DE COMPILACIÓN EN PASMO

Reglas de codificación para *PASMO*

Normas para las instrucciones:

☑ Pondremos una sola instrucción de ensamblador por línea.

☑ Como existen diferencias entre los "fin de línea" entre *Linux* y *Windows*, es recomendable que los programas se ensamblen con ***PASMO*** en la misma plataforma de S.O. en que se han escrito.

Si ***PASMO*** intenta compilar en *Windows* un programa ASM escrito en un editor de texto de *Linux* (con retornos de carro de Linux) es posible que obtengamos errores de ensamblado (aunque no es seguro).

Si os ocurre al compilar algún ejemplo de código escrito en *Linux* y usáis *Windows*, lo mejor es abrir el fichero .ASM con *Notepad* y grabarlo de nuevo (lo cual lo salvará con formato de retornos de carro de *Windows*).

El fichero "regrabado" con *Notepad* podrá ser ensamblado en *Windows* sin problemas.

☑ Además de una instrucción, en una misma línea podremos añadir etiquetas (para referenciar a dicha línea, algo que veremos posteriormente) y comentarios (con ';').

Normas para los valores numéricos:

☑ Todos los valores numéricos se considerarán, por defecto, escritos en decimal.

☑ Para introducir valores números en hexadecimal los precederemos del carácter "**$**", y para escribir valores numéricos en binario lo haremos mediante el carácter "**%**".

☑ Podremos también especificar la base del literal poniéndoles como prefijo las cadena **&H** ó **0x** (para hexadecimal) ó **&O** (para octal).

☑ Podemos especificar también los números mediante sufijos usando una **"H"** para hexadecimal, **"D"** para decimal, **"B"** para binario u **"O"** para octal (tanto mayúsculas como minúsculas).

Normas para cadenas de texto:

☑ Podemos separar las cadenas de texto mediante comillas simples o dobles.

☑ El texto encerrado entre comillas simples no recibe ninguna interpretación, excepto si se encuentran 2 comillas simples consecutivas, que sirven para introducir una comilla simple en la cadena.

☑ El texto encerrado entre comillas dobles permite introducir caracteres especiales al estilo de *C/C++* como **\n**, **\r** ó **\t** (nueva línea, retorno de carro, tabulador…).

☑ El texto encerrado entre comillas dobles también admite **\xNN** para introducir el carácter correspondiente a un número hexadecimal **NN**.

☑ Una cadena de texto de longitud 1 (un carácter) puede usarse como una constante (valor *ASCII* del carácter) en expresiones como, por ejemplo, **'C'+10h**.

Normas para los nombres de ficheros:

☑ Si vemos que nuestro programa se hace muy largo y por lo tanto incómodo para editarlo, podemos partir el fichero en varios ficheros e incluirlos mediante directivas **INCLUDE** (para incluir ficheros ASM) ó **INCBIN** (para incluir código máquina ya compilado). Al especificar nombres de ficheros, deberán estar entre dobles comillas o simples comillas.

Normas para los identificadores:

☑ Los identificadores son los nombres usados para etiquetas y también los símbolos definidos mediante **EQU** y **DEFL**.

☑ Podemos utilizar cualquier cadena de texto, excepto los nombres de las palabras reservadas de ensamblador.

Normas para las etiquetas:

☑ Una etiqueta es un identificador de texto que ponemos poner al principio de cualquier línea de nuestro programa, por ejemplo: "**bucle:**"

☑ Podemos añadir el tradicional sufijo "**:**" a las etiquetas, pero también es posible no incluirlo si queremos compatibilidad con otros ensambladores que no lo soporten (por si queremos ensamblar nuestro programa con otro ensamblador que no sea ***PASMO***).

☑ Para ***PASMO***, cualquier referencia a una etiqueta a lo largo del programa se convierte en una referencia a la posición de memoria de la instrucción o dato siguiente a donde hemos colocado la etiqueta. Podemos utilizar así etiquetas para hacer referencia a nuestros gráficos, variables, datos, funciones, lugares a donde saltar, etc.

Directivas:

☑ Tenemos a nuestra disposición una serie de directivas para facilitarnos la programación, como **DEFB** ó **DB** para introducir datos en crudo en nuestro programa, **ORG** para indicar una dirección de inicio de ensamblado, **END** para finalizar el programa e indicar una dirección de autoejecución, IF/ELSE/ENDIF en tiempo de compilación, **INCLUDE** e **INCBIN**, **MACRO** y **REPT**.

☑ La directiva **END** permite indicar un parámetro numérico (**END XXXX**) que "**pasmo --tapbas**" toma para añadir al listado *BASIC* de arranque el **RANDOMIZE USR XXXX** correspondiente. De esta forma, podemos hacer que nuestros programas arranquen en su posición correcta sin que el usuario tenga que teclear el "**RANDOMIZE USR DIRECCION_INICIO**".

☑ Una de las directivas más importantes es **ORG**, que indica la posición origen donde almacenar el código que la sigue. Podemos utilizar diferentes directivas **ORG** en un mismo programa. Los datos o el código que siguen a una directiva **ORG** son ensamblados a partir de la dirección que indica éste.

☑ Iremos viendo el significado de las directivas conforme las vayamos usando, pero es aconsejable consultar el manual de ***PASMO*** para conocer más sobre ellas.

Operadores:

☑ Podemos utilizar los operadores típicos **+**, **-**, *****, **/**, así como otros operadores de desplazamiento de bits como **>>** y **<<**.

☑ Tenemos disponibles operadores de comparación como **EQ**, **NE**, **LT**, **LE**, **GT**, **GE** o los clásicos **=** , **!=** , **<** , **>** , **<=** , **>=** .

☑ Existen también operadores lógicos como **AND**, **OR**, **NOT**, o sus variantes **&**, **|**, **!** .

☑ Los operadores sólo tienen aplicación en tiempo de ensamblado, es decir, no podemos multiplicar o dividir en tiempo real en nuestro programa usando ***** ó **/**.

Estos operadores están pensados para que podamos poner expresiones como **((32*10)+12)**, en lugar del valor numérico del resultado, por ejemplo.

OPCIONES DE COMPILACIÓN EN *PASMO*

Pasmo, ensamblador Z80 cruzado multiplataforma.

(C) 2004-2005 Julián Albo

Utilización y distribución permitida bajo la licencia GPL.

Para descargar actualizaciones o obtener más información:

http://www.arrakis.es/~ninsesabe/pasmo/

Para compilar (en sistemas *Linux*):

```
./configure
make
```

Para instalar (en sistemas *Linux*):

```
make install
```

Para compilar con otras opciones:

```
./configure --help
```

Documentación:

Además del presente documento, la disponible en el fichero incluido en el paquete o en el sitio web de Pasmo. Ver también los ficheros **.asm** de ejemplo incluidos en el paquete de los fuentes.

Para ensamblar/compilar***** *(Linux/Windows)*:

```
pasmo [ opciones ] fichero.asm fichero.bin [ fichero.simbolos [fichero.publicos] ]
```

***** Podemos lanzar el ensamblado directamente desde la propia carpeta del ejecutable de ***PASMO*** o desde cualquier otra si se añade la ruta del compilador ***PASMO*** a la variable **PATH** del sistema.

Opciones (respete el uso de las minúsculas;):

-d	☞ Mostrar **información de depuración** durante el ensamblado.
-1	☞ Mostrar **información de depuración** durante ensamblado, también en el primer paso.
-v	☞ Verboso. Muestra **información de progreso** del ensamblado.
-I	☞Añade directorio a lista de directorios en los que buscar ficheros para **INCLUDE/INCBIN**.
--bin	☞ Generar el fichero objeto en **binario puro sin cabecera**.
--hex	☞ Generar el fichero objeto en formato ***Intel* HEX.**
--prl	☞ Generar el fichero objeto en formato **PRL**. Adecuado para **RSX** de ***CP/M Plus***.
--cmd	☞ Generar el fichero objeto en formato **CMD de *CP/M 86*.**
--plus3dos	☞ Generar el fichero objeto con cabecera ***PLUS3DOS* (Spectrum disco)**.
--tap	☞ Generar un fichero **.tap** para emuladores de ***Spectrum* (imagen de cinta)**.
--tzx	☞ Generar un fichero .tzx para emuladores de ***Spectrum* (imagen de cinta)**.
--cdt	☞ Generar un fichero **.cdt** para emuladores de ***Amstrad CPC* (imagen de cinta)**.
--tapbas	☞ Igual que que la opción **--tap** pero añadiendo un **cargador Basic**.
--tzxbas	☞ Igual que que la opción **--tzx** pero añadiendo un **cargador Basic.**
--cdtbas	☞ Igual que que la opción **--cdt** pero añadiendo un **cargador Basic**.
--amsdos	☞ Generar el fichero objeto con cabecera ***Amsdos* (*Amstrad CPC* disco).**
--msx	☞ Generar el fichero objeto con cabecera para usarse con **BLOAD** en ***MSX Basic***.
--public	☞ El listado de símbolos incluirá sólo los declarados **PUBLIC.**
--name	☞ **Nombre para la cabecera** en los formatos que lo usan (si no se especifica se usa el nombre del fichero objeto).
--err	☞ Dirige mensajes de error a salida **estándar** en vez de salida error (excepto errores en opciones).
--nocase	☞ Hace que los **identificadores** no distingan **mayúsculas** de **minúsculas**.
--alocal	☞ **Modo autolocal**: las etiquetas que comienzan por '_' son locales y su ámbito termina en la sigiente etiqueta no local o en la siguiente directiva **PROC**, **LOCAL** ó **MACRO**.
-B	
--bracket	☞ **Modo sólo corchetes**: los paréntesis quedan reservados para expresiones.
-E	
--equ	☞ Predefine una **etiqueta**.
-8	
--w8080	☞ Mostrar **warnin**g cuando se usan instrucciones del **Z80** que no existen en el **8080**.
--86	☞ Generar código **8086**.
-	☞ **Fin de opciones**, todo lo que siga se considera nombre de fichero aunque comience por **-**

Si no hay ninguna opción de formato de objeto se asume **-bin**.

La información de depuración va a la ***salida estándar****, los errores a la* ***salida de error****.*

https://calentamientoglobalacelerado.net/zxopensource/

El ser humano es la única especie del reino animal (inteligente?) que establece agravios y prejuicios en función de factores fenotípicos como por ejemplo, el color de la piel.

PRIMER TEST DE CONCIENCIA SOBRE RAZAS HUMANAS