Practica 5.- “Tabla de Símbolos”

2.1.4 Directivas, contador de localidades y archivos auxiliares.

OBJETIVO DEL SUBTEMA. Modificar el programa que se diseño en el punto 2.1.3. El programa actualizado además, deberá de validar la sintaxis de las directivas; Generar el contador de localidades; y generar los archivos auxiliares (temporal o de listado; y tabla de símbolos).

Introducción:
El ensamblador puede generar código objeto en un sólo paso o en varios. La principal limitación del ensamblador de un paso es que no puede resolver referencias delanteras (por ejemplo saltos hacia adelante con referencias simbólicas).

Durante la primera fase o primer paso, se abren dos archivos, el de texto que tiene el programa fuente (escrito en símbolos de ensamblador o mnemotécnicos) y el importantísimo archivo que contiene las tablas de códigos de operación del conjunto de instrucciones de la máquina objeto.

Adicionalmente se genera una tabla de símbolos definidos por el usuario, es decir las etiquetas o referencias simbólicas y se inicializa un contador local de instrucciones de acuerdo a las pseudo instrucciones dadas en el programa (ORG, EQU, DB, DW, DS.B DS.W y END). La tabla de símbolos y el contador de localidades sirven para una posterior consulta y en última instancia para completar la traducción de las instrucciones con todo y operando.

El contador de localidades CONTLOC que es una variable empleada como ayuda en la asignación de direcciones, el valor inicial de CONTLOC de ser de 0. La directiva ORG se usa para modificar el valor de esta variable.

Después de procesar cada proposición fuente se suma a CONTLOC la longitud de la instrucción ensamblada o el área de datos generada. De esta forma cada vez que se alcanza una etiqueta en el programa fuente el valor actual de CONTLOC proporciona la dirección asociada a esa etiqueta. El archivo de listado LST contiene el CONTLOC, la línea que se leyó, y el código máquina que genera esa instrucción.

DIRECTIVAS

DB
<etiqueta> DB opr8,opr8,opr8,…,opr8
Define el valor de un byte o de los bytes que sean puestos en una dirección dada. DB asigna el valor de la expresión (opr8) como código maquina en la dirección de memoria indicada por el CONLOC en tamaño de 8 bits. Entonces el contador de localidades se incrementa según el número de operandos de la instrucción.
Esta directiva también acepta cadenas encerradas entre comillas dobles (“) como operando, en cuyo caso cada carácter de la cadena se interpretara como un operando y el numero de caracteres de la cadena será el numero en el que se tiene que incrementar el CONLOC.

Ejemplo:

```
CONLOC    CODMAQ    INSTRUCCIÓN_EN_LENGUAJE_ENSAMBLADOR
4000      0102030A          DB    01,%10,$3,@12
4004      41424344          DB    “ABCD”
4008
```

DC.B
Idéntico al DB.

DW
Define el valor de una palabra o de las palabras que sean puestos en una dirección dada. DW asigna el valor de la expresión al contador de localidades actual. Entonces el contador de localidades se incrementa multiplicando por dos el número de operandos. Ejemplo:

```
CONLOC    CODMAQ              INSTRUCCIÓN_EN_LENGUAJE_ENSAMBLADOR
4000      000100020003000A          DW    01,%10,$3,@12
4008      0041004200430044          DB    “ABCD”
400C
```

DC.W
Idéntico al DW.

DS
Esta directiva se utiliza para reservar espacio en memoria, el valor del operando es el numero de bytes que deberá reservar. Generalmente esta directiva contiene etiquetas ya que se utilizan como nombres de variables. Ejemplo:

```
CONLOC    CODMAQ              INSTRUCCIÓN_EN_LENGUAJE_ENSAMBLADOR
4000                          D1        DS    01
4001                          D2        DS    05
4006                          R         DS    $20
4026
```

```
      TABSIM
D1          4000H
D2          4001H
R           4006H
```

DS.B
Idéntico a DS

DS.W
Muy similar a DS la única diferencia es que ahora el operando indica numero de palabras. El numero de bytes será el valor del operando multiplicado por 2. Ejemplo:

```
CONLOC    CODMAQ              INSTRUCCIÓN_EN_LENGUAJE_ENSAMBLADOR
4000                          V1        DS.W 01
4002                          V2        DS.W  05
400C                          V3        DS.W  $20
```

404C

TABSIM

V1	4000H
V2	4002H
V3	400CH

END
Indica el final del código en lenguaje ensamblador. Es el fin del archivo fuente. Cuando el programa ensamblador encuentra esta directiva el ensamblador deja de buscar instrucciones.

EQU
Asigna un valor directamente a una etiqueta dentro de la tabla de símbolos (tabsim). Esta directiva debe tener etiqueta y un operando numérico. Ejemplo:

CONLOC	CODMAQ	INSTRUCCIÓN_EN_LENGUAJE_ENSAMBLADOR	
4000		V1	EQU 01
4000		V2	EQU 50
4000		V3	EQU $2000
4040			

TABSIM

V1	0001H
V2	0032H
V3	2000H

ORG
Esta directiva se usa para modificar el valor del contador de localidades (CONLOC), cuando exista esta directiva el CONLOC deberá tomar el valor del operando de la directiva. Ejemplo:

CONLOC	CODMAQ	INSTRUCCIÓN_EN_LENGUAJE_ENSAMBLADOR	
4000		V1	DS.W 01
4002		V2	DS.W 05
400C			ORG $5000
5000		V3	DS.W $20
5040			

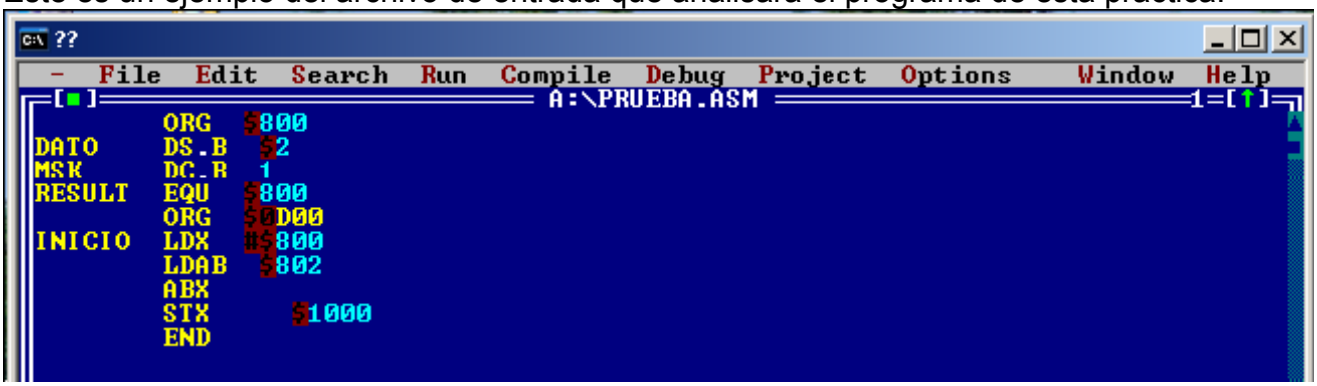
Desarrollo:
Los pasos son casi los mismos pues esta práctica (como las otras) es una modificación de la tres. Así es que lo nuevo que le agregue lo pondré con azul… Después de separar la línea de instrucción en partes, tomo las que me va a servir (instrucción) y el (los) operandos para determinar el numero de bytes que ocupa la intrucción: Aquí agregue validaciones por si encuentra directivas: END; para que solo haga la instrucción actual y termine el ciclo, ORG; para inicializar el contador de localidades(tomo el puntero que tiene la dirección de esta directiva y lo convierto a entero), además tengo un contador de ORG's, EQU; debe tener su etiqueta, abro el archivo tabsim (para tabla de simbolos) y escribo la etiqueta con el valor de la directiva, para estas tres anteriores el aumento del conloc es cero.

después DS.B y DC.B asigno a la longitud el número de operadores, DS.W y DC.W multiplico el conop * 2 y lo asigno a longitud, y para DS.D, DC.D será conop*4 y asigno a longitud.

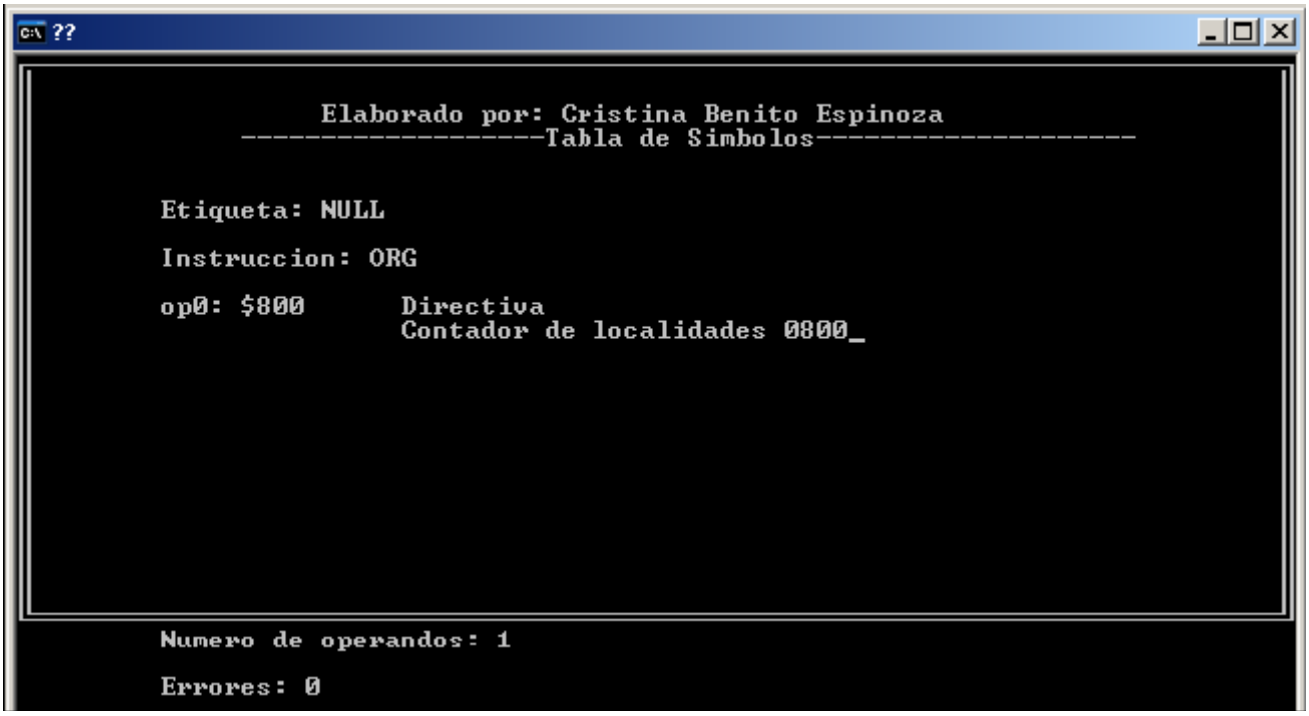
 Si no es una directiva entra al ciclo normal de identificación de instrucción:
 Coloco un contador de punteros y dependiendo del numero me voy a un caso (0,1,2,3)
 Antes que nada es preciso explicar que hice una función especial para la conversión de bases a entero, para poder checar los rangos, en esta función mando como parámetro el número y dependiendo de la base $, %, @ u otra (casi siempre checo los signos que tiene directamente accediendo a x numero de carácter de puntero a operando), convierto con strtol() y lo regreso como entero.
 En el caso 1: primero checo si el operando es una etiqueta (con el primer carácter del puntero a operando), si lo es asigno a una variable el modo de direccionamiento, que en este casi sería relativo, y valido para algunas instrucciones que son de 16 (lbcs, lbcc, ect). Otro punto es el #, si existe en el operando mando lo que le sigue a la función especial de conversión y me lo regresa en decimal para chocarle el rango y dependiendo de este le pongo el modo de direccionamiento a la variable (IMM).
 En el caso de 2 operandos: casi las mismas comprobaciones del caso uno, pero ahora de manera simultánea con los dos operandos; por ejemplo checo que el primer operando sea digito, o que inicie con $ mientras que el segundo sea xysp y mando el primer operando a la función de conversión para determinar el rango y asignar el modo de direccionamiento (IDX, IDX1, IDX2), Otra verificación que realizo es para el [IDX2], compruebo que tenga al inicio del primer operador [y al final del segundo] y lo de adentro es lo mismo que los de arriba. También valide para el tipo de instrucciones movb y movw pero creo que estas no son necesarias por el momento.
 En el caso 3: validé para direccionamientos directo o extendido como opr8, msk8, rel8…
 Termino esta parte mandando a la función de búsqueda de instrucción el modo de direccionamiento al que pertenece esta instrucción, la instrucción misma y toda la línea que fue analizada…. Pero…
 Si fue una de las directivas comienzo a escribir en el archivo lst, toda la instrucción , el conloc , el código maquina (Con varias comparaciones).
 Si no fue una directiva sigue el ciclo normal de identificación de instrucción…Ya dentro de la función de búsqueda abro el archivo de instrucciones que tengo en memoria auxiliar para solo leerlo, pero esta vez lo abro de distinta manera fopen.
 Hago un ciclo mientras no sea fin de archivo y leo TODA la línea del archivo, buscando la instrucción dentro de la línea si la encuentra entra a una comparación mas que es para verificar el modo de direccionamiento, si son iguales; hago un pequeño ciclo en la línea leída con strtol para obtener la longitud, si es igual asigno el valor de la longitud a una variable, y cambio dos banderas que tengo (una es para decirme que en la línea si está la instrucción y el modo de direccionamiento y la otra para marcar que existe, esta ultima se queda activa desde que encuentra lo que busca, la otra esta cambiando con cada ciclo), si no es igual y la segunda bandera está activa; interrumpo el ciclo. Dentro de esta función escribo en lst, conloc, código maquina, algunos espacios y después incremento el conloc.
 El ciclo se repite mientras no se encuentre la instrucción o para imprimir el formato que puede tomar la instrucción con el modo de direccionamiento especificado.
 Si la segunda bandera nunca fue activada y ya es fin de archivo; imprimo que no fue encontrada.
 Cierro el archivo de instrucciones.
 Regreso la longitud y la mando mostrar en el main().

 Checo que la información haya sido guardada de manera correcta en los archivos
 Fin de la práctica.

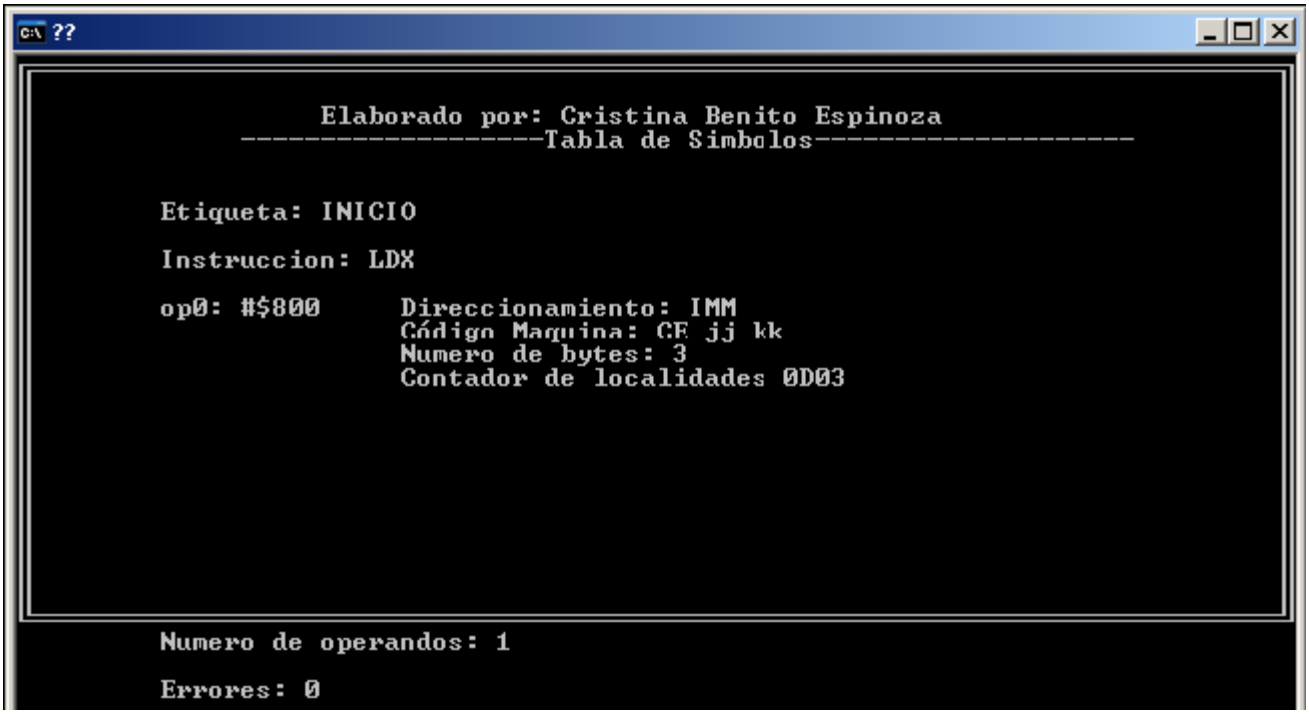
Este es un ejemplo del archivo de entrada que analisara el programa de esta práctica:



Ejemplo de la Salida del programa: Solo se muestra la salida que da el análisis de una línea
de una directiva y otra línea de una instrucción del archivo ASM que se muestra arriba.
Directiva:



Instrucción:



Cuando se analiza línea a línea el ASM, se obtienen las salidas que arriba se mostraron y también los siguientes archivos.

Ensamblador desarrollado por: Valentín Martínez López

```
0000                    ORG   $800
0800          DATO      DS.B  $2
0802 01       MSK       DC.B  1
0803          RESULT    EQU   $800
0803                    ORG   $0D00
0D00 CE jj kk INICIO    LDX   #$800
0D03 F6 hh ll           LDAB  $802
0D06 1A E5              ABX
0D08 7E hh ll           STX   $1000
0D0B                    END
```

Este archivo tiene extensión LST.

Y en seguida la tabla de símbolos correspondiente.

```
DATO      0800H
MSK       0802H
RESULT    0800H
INICIO    0D00H
```