UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

FACULTAD DE INGENIERÍA

INGENIERIA EN COMPUTACIÓN

ESTRUCTURA Y PROGRAMACION DE COMPUTADORAS

PROYECTO:

"ABAKOS TZILMIZTLI"

Alumno	Jaime Hernández Vázquez
Miguel Israel Barragán Ocampo	Profesor

Índice.

Objetivo del trabajo.

Una calculadora realizada en un lenguaje de bajo nivel guía al desarrollador, este caso un alumno de licenciatura, hacia un entendimiento más propio de lo que es una computadora y lo que ésta puede realizar.

De manera general, se abordan las operaciones principales de un microprocesador de la familia Intel 80X86 y su coprocesador matemático, y las interrupciones para el uso de dos de los principales dispositivos de entrada y salida.

En adición a lo anterior, el desarrollador de un programa de bajo nivel necesita conocer la representación de información en una computadora, sistemas numéricos (binario, octal, decimal y hexadecimal), los formatos binarios y su organización, así como las posibles operaciones con éstos y el código ASCII en un principio. Posteriormente y no menos importante es el conocimiento sobre la organización del sistema: Sus componentes básicos; buses, memoria, dispositivos de entrada y salida. Su sincronización: reloj, acceso a memoria, estados de espera, memoria caché e incluso más información que puede ser importante en el diseño de software que simplifiquen los algoritmos con las herramientas matemáticas y de hardware disponibles.

En cuanto al procesador, debe conocerse una estructura funcional de éste. Siendo que cada procesador es diferente, la arquitectura puede cambiar y no es necesario conocerla a detalle para los distintos procesadores de la familia 80X86 ya que operan de forma semejante y las instrucciones implementadas no presentan muchos cambios entre éstos. Otro aspecto importante es conocer el efecto de cada instrucción que se utiliza, sus variables de entrada y de salida, ya sean registros o localidades de memoria, deben conocerse las características de los registros y el uso de memoria.

Para obtener una pieza de software, en lenguaje de bajo nivel, es necesario declarar las instrucciones con una sintaxis correcta, que posteriormente el ensamblador puede señalar. El ensamblador utilizado es el Turbo Assembler de Borland, ya que las directivas tienen la sintaxis para éste. Aunque pudiera utilizarse otro ensamblador, es necesario verificar que el código sea compatible. Después un ligador (linker) es necesario para relacionar el código ensamblado con alguna biblioteca de funciones necesaria.

La realización de una biblioteca se lleva a cabo mediante un administrador de bibliotecas (library manager) y funciona de manera semejante al enlazador.

Lo antes mencionado, desde el punto de vista didáctico, es suficiente motivo - de un curso semestral de programación en lenguaje de bajo nivel si se tiene en cuenta que los conocimientos anteriores son de programación básica en lenguaje de alto nivel- para la realización de una calculadora como la que se plantea en adelante.

Introducción.

Abakos Tzilmiztli (*abakos* del latín, que significa ábaco y *tzilmiztli* del náhuatl "puma negro"). Es una calculadora que mediante el almacenamiento de números reales[1] en una pila[2], realiza operaciones aritméticas, trascendentes y de conversión sobre él o los números que se encuentren en el tope de la pila.

Las operaciones se ejecutan tras introducir un comando, que de ser válido ejecuta la operación y el/los elementos involucrados son reemplazados por el resultado, de esa manera puede utilizarse el resultado para una operación posterior.

Además de realizar tales operaciones, la calculadora muestra en pantalla, de existir, todos los elementos de la pila. Una interfaz gráfica contiene los recursos necesarios para el uso de la calculadora, un cuadro de información del programa, uno que proporciona la fecha y hora del sistema, otro que muestra ayuda rápida y otro disponible para visualizar el numero o comando que introducirá el usuario.

El entorno en el cual se ejecuta la calculadora es el shell[3] del sistema operativo DOS, en modo texto.

La descripción básica de implementación de la calculadora es mediante interrupciones del BIOS[4] y de DOS y el uso del coprocesador matemático que opera sobre números en formato flotante y realiza una serie de operaciones especiales que el procesador por sí solo, no.

El programa abakos.asm utiliza procedimientos de la biblioteca liakas.lib (abreviatura del totonaca, *liakasltawakat* que significa libro). Como posteriormente se describe.

[1] Es imposible almacenar números irracionales en una computadora, en este caso los llamados números reales son racionales

[2] Estructura de datos tipo LIFO, en donde se realizan las operaciones conocidas: Push y Pop (introducir y extrer datos de la estructura, respectivamente).

[3] Pieza de software que provee una interfaz al usuario de un sistema operativo y proporciona acceso al kernel (componente central de un sistema operativo)

[4] Sistema básico de entrada/salida, es el administrador de los dispositivos y controla su hardware.

Filosofía de diseño.

Para llegar al objetivo del programa se intentó crear un programa de filosofía minimalista, quizá aplicando el principio KISS, que realizara lo necesario en pocas líneas y que además fuera lo suficientemente robusto como para ser a prueba de algún tipo de errores, esto fue fácilmente logrado en el procedimiento read, que se obtuvo de The Intel Microprocessors, Barry Brey; 1995 ya que si es introducida una cadena de caracteres que no resulten ser un número o un comando, carga cero a la pila. Esto es un buen diseño de software, por el contrario, la poca experiencia del autor hizo que no lograra el cometido en la mayor parte del código como se puede observar más adelante.

Tal vez esto se deba a un paradigma de programación estructurada y orientada a objetos con métodos redundantes - como se puede observar en algún procedimiento – y siendo el utilizado en la calculadora, un lenguaje con un paradigma distinto en donde, a juicio propio, puede complicase o simplificarse enormemente una tarea a realizar. Esto se suma a la falta de experiencia y a los pocos conocimientos de ciencias computacionales especialmente donde entran las matemáticas.

El programa se encuentra separado en tres módulos principales:

- El programa principal abakos.asm
- Los procedimientos en lichanat.asm
- El procedimiento Float2String

Los primeros dos, el autor de éste texto los realizó, mientras que el tercero fue proporcionado por el profesor Miguel Barragán, procedimiento que realiza la conversión de un número en formato flotante a una cadena de caracteres.

Algunos procedimientos fueron imposibles de ser extraídos del programa principal (listados más adelante) porque no fueron lo suficientemente independientes como para incluirse en una biblioteca dado que realizaban saltos hacia el principio del programa – comienza casi todo el proceso de nuevo – o bien, saltos a otros procedimientos o a instrucciones aisladas.

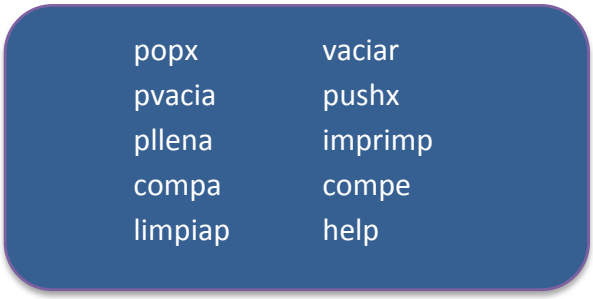
Los procedimientos exitosamente aislados del programa principal están en el archivo lichanat (del totonaca *lichanat*: semilla) que de igual forma que el archivo del programa principal abakos.asm, es ensamblado, pero con la diferencia de que se agrega el objeto (.obj) a una biblioteca, liakas en este caso.

Los procedimientos públicos en el archivo lichanat e incluso el procedimiento Float2String están incluidos en la biblioteca liakas como se muestra en la **Figura 1.**

Otra forma de la programación modular son las macroinstrucciones, que en este caso se utilizan tanto para encapsular funciones - definidas por procedimientos en donde se necesitan uno o más parámetros de entrada, que cambian de valor dentro del mismo código (el nombre de una cadena, por ejemplo) – o para realizar instrucciones que se repiten una cantidad considerable de veces.

Los procedimientos privados que utilizan los procedimientos públicos de lichanat.asm están listados a continuación:

Los procedimientos utilizados por abakos.asm son los siguientes:



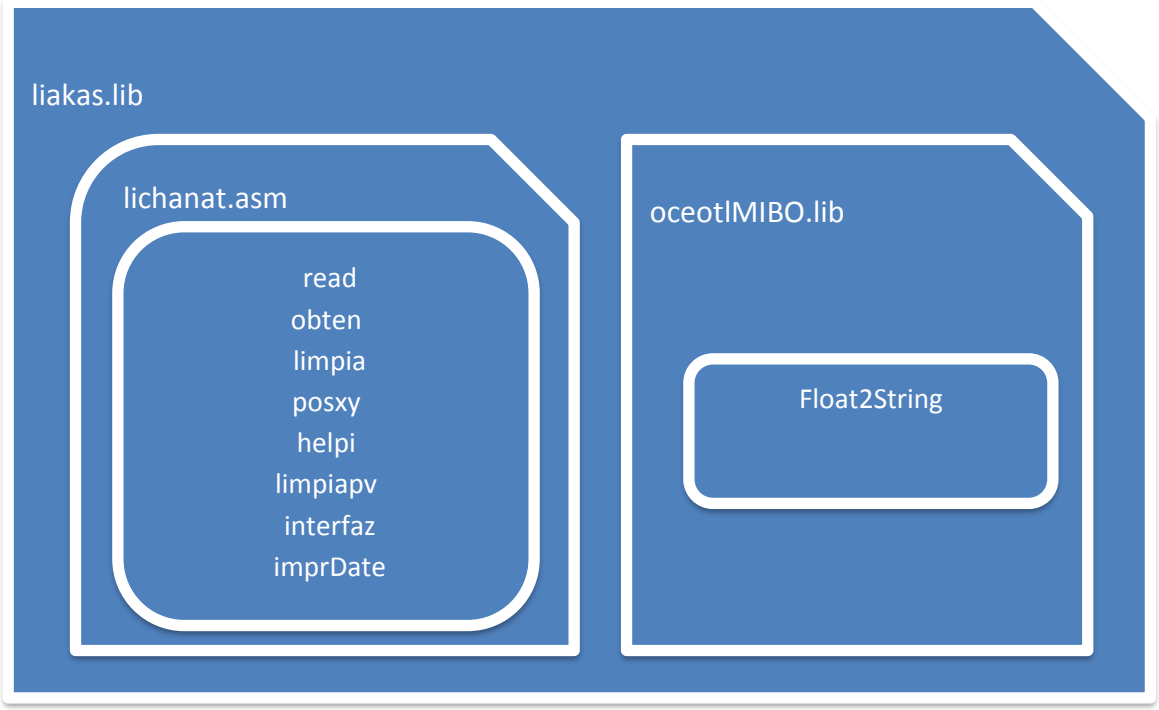


Figura 1. Procesos públicos en los diferentes archivos que forman a la biblioteca liakas.lib

Las macro instrucciones son utilizadas en ambos archivos e incluso pueden ser las mismas tanto en un archivo como en otro, esto es porque no se incluyeron en la biblioteca. Se muestran en la **Figura 2.**

Cabe mencionar que las variables enteras que se coloquen como parámetros son de 16 bits , una palabra (word).

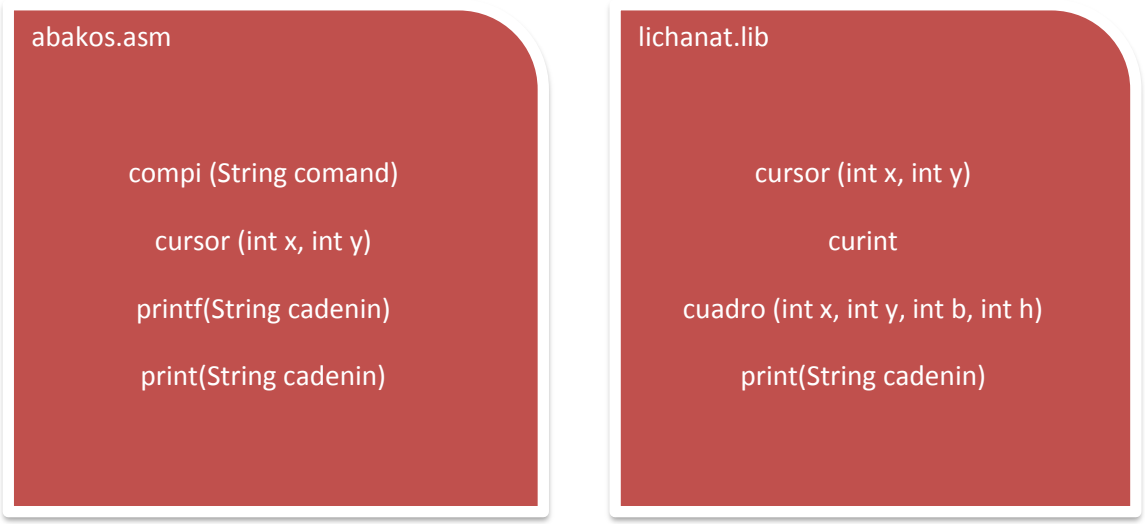


Figura 2. Macroinstrucciones que se encuentran en los archivos que forman el programa

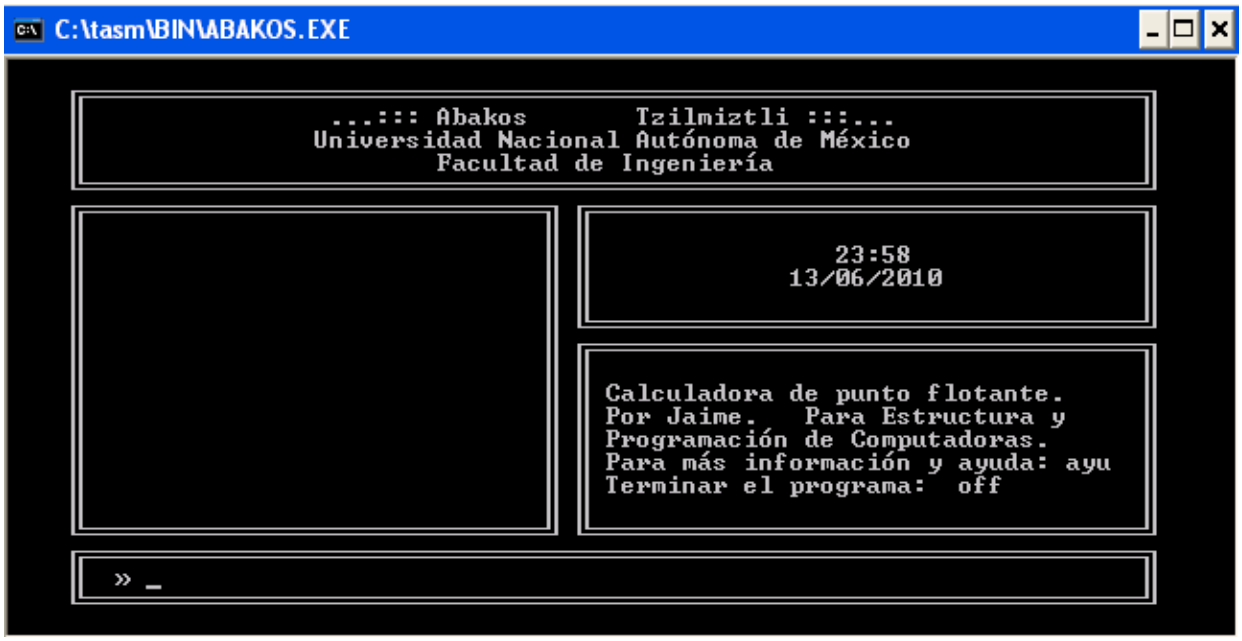
Se utilizó una estructura para definir la variable pila, que contiene 12 variables de formato flotante con extensión de 80 bits (10 bytes) y un índice de la posición del tope de pila. Esta estructura es de tipo LIFO y como es análoga a la pila que utiliza el procesador (implementada por el ensamblador , stack), puede confundirse con ésta pero se tratan de dos entes distintos dentro del programa. **Figura 3.**

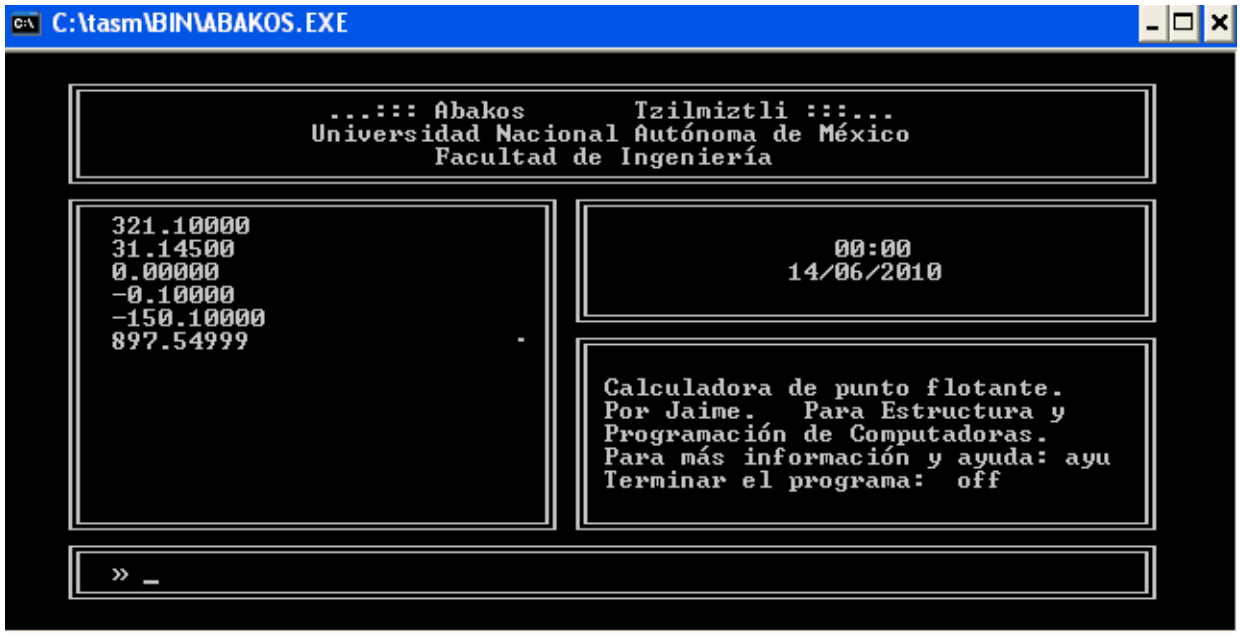
D_1	31.114	
D_2	-180.1	
D_3	-0.61	
D_4	0	
…	…	
D_9	150.1	← Índice
D_{10}		
D_{11}		
D_{12}		

Figura 3. Representación de la pila que utiliza la calculadora

Muestras del programa.

Impresiones de pantalla:





Código.

Del programa principal “abakos”:

```
            title       'Pila y funciones'
;-----MACROINSTRUCCIONES-----
;---Encapsula cmpsb para comparar cada comando parámetro: cadena de 3 caracteres
compi       macro       comand
            lea         si,comand
            mov         di,bx
            inc         di
            mov         cx,3        ;compara solamente tres caracteres
            inc         dx
      rep   cmpsb                   ;cmpsb (compara string c/byte) con etiqueta rep
            je          sigue       ;salta a comparación de comandos
            endm
;---Posiciona cursor posición (x,y)
cursor      macro       x,y
            push        x
            push        y
            call        posxy
            endm
;---Imprime un numero de punto flotante, tiene como parámetro la cadena destino,
;variables: la dirección de ésta y el número de dígitos postpunto, utiliza Float2String
printf      macro       cadenin
            push        offset cadenin
            push        decfloat
            call        Float2Str   ;procedimiento por MIBO(profesor) en oceotlMIBO.lib
            lea         dx,cadenin  ;imprime la cadena
            mov         ah,09h
            int         21h
            endm
;---Imprime una cadena cualquiera con terminación $
print       macro       cadenin
            lea         dx,cadenin
            mov         ah,09h
            int         21h
            endm
;-----ESTRUCTURAS-----
;---Pila, con 12 elementos y un indice
pila        struc
elem        dt          12 dup(0.0)       ;doce elementos adyacentes
indice      dw          1                 ;índice del tope de la pila
pila        ends
;Declaracion de modelo small y tipo de procesador y coprocesador
            .model      small
            .386                    ;Procesador 80386 y coprocesador 80387
            .387
            .stack      256         ;Tamaño del stack en memoria de 256
```

```
            .data                   ;Segmento de datos
;.....Variables
semi        dt          180.0   ;Pi radianes en grados
temp1       dw          ?           ;Variable auxiliar
;...Cadenas que limpian la pila y el prompt
clean1      db          27 dup(32),"$"
clean2      db          60 dup(32),"$"
pointer     db          25 dup(32),249,"$"    ;Cadena que contiene el apuntador de pila
;...Cadenas de mensajes pila con over y underflow
pilallena   db          7,"Oh ... la pila esta llena$"
pilavacia   db          7,"No!... la pila esta vacia$"
;...Cadena para almacenar los números de punto flotante
chain       db          32 dup (" ")
decfloat    dw          5           ;Numero de cifras postpunto flotante
stapel      pila        <>          ;Variable tipo pila
opc         db          ?           ;Opcion de comando
;...Cadenas que representan un comando:
ayud        db          'ayu' ;opc  1
divi        db          'div' ;opc  2
cose        db          'cos' ;opc  3
abso        db          'abs' ;opc  4
fact        db          'fac' ;opc  5
raiz        db          'sqr' ;opc  6
mult        db          'mul' ;opc  7
pote        db          'pow' ;opc  8
rsta        db          'res' ;opc  9
suma        db          'sum' ;opc  10
tang        db          'tan' ;opc  11
sale        db          'off' ;opc  12
nega        db          'neg' ;opc  13
ldpi        db          'pii' ;opc  14
gr2r        db          'rad' ;opc  15
r2gr        db          'gra' ;opc  16
saca        db          'pop' ;opc  17
loga        db          'log' ;opc  18
lnat        db          'lnt' ;opc  19
ente        db          'rnd' ;opc  20
npil        db          'new' ;opc  21
seno        db          'sen' ;opc  22
;...Valores de opc para los distintos comandos:
nayud       equ         01
ndivi       equ         02
ncose       equ         03
nabso       equ         04
nfact       equ         05
nraiz       equ         06
nmult       equ         07
npote       equ         08
nrsta       equ         09
nsuma       equ         10
```

```
ntang       equ         11
nsale       equ         12
nnega       equ         13
nldpi       equ         14
ngr2r       equ         15
nr2gr       equ         16
nsaca       equ         17
nloga       equ         18
nlnat       equ         19
nente       equ         20
nnpil       equ         21
nseno       equ         22
;...Variables auxiliares
retadd      dw          ?
bxsup       dw          ?
cxsup       dw          ?
aux         dt          0.0
printp      dw          7
            .code       ;Segmento de código
;-----Procedimientos externos
            extrn       Float2Str:proc,read:proc,obten:proc,limpia:proc,helpi:proc
            extrn       posxy:proc,interfaz:proc,imprDate:proc,limpiapv:proc
Start:
main:
            mov         ax,@data
            mov         ds,ax
            mov         es,ax
            finit
            call        interfaz
otra:
            call        imprDate
            call        imprimp
poste:      cursor      09,22
            print       clean2
            cursor      09,22
            call        obten
            call        compa
sigue:
            mov         opc,dl
            cmp         opc,22
            ja          @noinst
            call        compe
;----Operaciones que realizan los comandos sobre los valores que están en la pila
;-(stapel) del programa y continúan en etiqueta otra, menos off. Usa Coprocesador.
;---Operación de off (cerrar el programa)
@sale:      call        limpia      ;limpia pantalla
            mov         ah,4ch      ;entrega el control al S.O.
            int         21h
@ayud:      call        help        ;Operación de ayuda
            jmp         otra
```

```
@divi:      call        popx        ;Operación de división
            call        popx
            fdivr                   ;división invertida
            call        pushx
            jmp         otra
@cose:      call        popx        ;Operación de coseno
            fcos
            call        pushx
            jmp         otra
@seno:      call        popx        ;Operación de seno
            fsin
            call        pushx
            jmp         otra
@abso:      call        popx        ;Operación de valor absoluto
            fabs
            call        pushx
            jmp         otra
@raiz:      call        popx        ;Operación de raíz cuadrada
            fsqrt
            call        pushx
            jmp         otra
@mult:      call        popx        ;Operación de multiplicación
            call        popx
            fmul
            call        pushx
            jmp         otra
@rsta:      call        popx        ;Operación de resta
            call        popx
            fsubr
            call        pushx
            jmp         otra
@suma:      call        popx        ;Operación de suma
            call        popx
            fadd
            call        pushx
            jmp         otra
@tang:      call        popx        ;Operación tangente
            fptan
            fxch        st(1)
            call        pushx
            jmp         otra
@nega:      call        popx        ;Operación cambia signo
            fchs
            call        pushx
            jmp         otra
@ldpi:      fldpi                   ;Carga pi
            call        pushx
            jmp         otra
@fact:      call        popx        ;Factorial de un numero entero positivo
            fabs
            frndint
```

```
            ftst
            fstsw       ax           ;carga statusword del cop a ax
            and         ax,4500h     ;nos quedamos con c3,c2 y c0
            cmp         ax,4000h     ;verificamos si es cero
            je          @faccero     ;Continua en el factorial de cero
            fstp        aux
            fld         aux
@facto:
            fld         aux
            fld1
            fchs
            fadd
            ftst
            fstsw       ax           ;carga statusword del cop a ax
            and         ax,4500h     ;nos quedamos con c3,c2 y c0
            cmp         ax,4000h     ;verificamos si es cero
            je          @@facto
            fstp        aux
            fld         aux
            fmul
            jmp         @facto
@faccero:   fstp        aux          ;factorial del caso especial cero
            fld1
            jmp         @@faccero
@@facto:    fstp        aux
@@faccero:  call        pushx
            jmp         otra
@pote:      call        popx         ;Operación potencia
            call        popx
            fxch
            fistp       temp1
            mov         cx,temp1
            cmp         cx,0
            je          @pcero
            cmp         cx,1
            je          @@poten
            dec         cx
            fstp        aux
            fld         aux
            fld         aux
@poten:
            fmul
            fld         aux
            loop        @poten
            fstp        aux
            jmp         @@poten
@pcero      fld1                     ;caso especial exponente cero
            jmp         @@poten
@@poten:
            call        pushx
            jmp         otra
```

```
@gr2r:      call        popx        ;Conversión grados a radianes
            fldpi
            fmul
            fld         semi
            fdiv
            call        pushx
            jmp         otra
@r2gr:      call        popx        ;Conversión radianes a grados
            fldpi
            fdiv
            fld         semi
            fmul
            call        pushx
            jmp         otra
@saca:      call        popx        ;Elimina el valor tope de la pila
            jmp         otra
@loga:      fld1                    ;Logaritmo vulgar
            call        popx
            fyl2x
            fldl2t
            fdiv
            call        pushx
            jmp         otra
@lnat:      fld1                    ;Logaritmo natural
            call        popx
            fyl2x
            fldl2e
            fdiv
            call        pushx
            jmp         otra
@ente:      call        popx        ;Redondeo al entero más cercano
            frndint
            call        pushx
            jmp         otra
@npil:      call        vaciar      ;Vacía la pila
            jmp         otra
;---No es instrucción ni número (carga cero)
@noinst:    pop         retadd
            call        read
            call        pushx
            jmp         otra
;---Saltos aislados (fuera de procedimientos)
@point:
            cursor      8,printp
            print       pointer
            inc         printp
            jmp         vas
noprin:
            call        limpiapv
            jmp         sigprin
```

```
;--------PROCEDIMIENTOS
;.......Pila
;-----Procedimiento que realiza un pop en la pila (stapel)
popx        proc
            push        bx
            cmp         stapel.indice,1   ;Compara el índice (1:vacia)
            je          vacia
            lea         bx,stapel.elem    ;Obtiene la dirección de los elementos
            mov         cx,word ptr[stapel.indice]
            dec         cx
luq:        add         bx,10             ;Encuentra el elemento deseado
            loop        luq
            sub         bx,10
            fld         tbyte ptr[bx]     ;Carga al coprocesador
            dec         stapel.indice
            jmp         conti
vacia:      call        pvacia            ;Muestra mensaje de pila vacía
conti:      pop         bx
            ret
popx        endp
;-----Procedimiento que vacía la pila
vaciar      proc
            mov         stapel.indice,1
            ret
vaciar      endp
;-----Procedimiento que muestra mensaje de pila vacía
pvacia      proc
            cursor      7,19
            print       pilavacia
            ret
pvacia      endp
;-----Procedimiento que realiza un push en la pila (stapel)
pushx       proc        near
            push        bx
            cmp         stapel.indice,13        ;Compara el índice (13:llena)
            je          lleno
            lea         bx,stapel.elem          ;Obtiene la dirección de los elementos
            mov         cx,word ptr[stapel.indice]
lup:        add         bx,10                   ;Encuentra el elemento deseado
            loop        lup
            sub         bx,10
            fstp        tbyte ptr[bx]           ;Carga del coprocesador
            inc         stapel.indice
            jmp         cont
lleno:      call        pllena                  ;Muestra mensaje de pila llena
cont:       pop         bx
            ret
pushx       endp
```

```
;-----Procedimiento que muestra mensaje de pila llena
pllena      proc        near
            call        imprimp
            cursor      7,19
            print       pilallena
            jmp         poste
            ret
pllena      endp
;-----Procedimiento que imprime la pila
imprimp     proc        near
            cmp         stapel.indice,1         ;Verifica si está vacía
            je          noprin                  ;No imprime elemento alguno
            call        limpiap                 ;Limpia el lugar de la pila
            lea         bx,stapel.elem          ;Obtiene la dirección de los elementos
            mov         cx,word ptr[stapel.indice]
            dec         cx
            mov         printp,7
again:            ;Imprime hasta la cantidad máxima de elementos en la pila, solamente
            fld         tbyte ptr[bx]
            push        bx          ;se hace un respaldo de los registros bx y cx
            push        cx
            cursor      7,printp
            printf      chain ;imprime el numero
            pop         temp1
            pop         temp1
            pop         cx
            pop         bx
            add         bx,10
            inc         printp
            loop        again
sigprin:    ret
imprimp     endp
;.......Pila
;-----Procedimiento que compara cadena y asigna un valor a dx,
;--- dependiendo del comando
compa       proc        near
            cmp         byte ptr[bx],3    ;Compara el tamaño de caracteres en la cadena
            jne         @noinst                 ;Si es diferente, no es un comando
            xor         dx,dx
            cld                     ;Limpia la bandera de dirección para cmpsb
            compi       ayud
            compi       divi
            compi       cose
            compi       abso
            compi       fact
            compi       raiz
            compi       mult
            compi       pote
            compi       rsta
            compi       suma
```

```
            compi       tang
            compi       sale
            compi       nega
            compi       ldpi
            compi       gr2r
            compi       r2gr
            compi       saca
            compi       loga
            compi       lnat
            compi       ente
            compi       npil
            compi       seno
            inc         dx
            ret
compa       endp
;----_Procedimiento que compara el valor de opc con el de un comando
compe       proc  near
            cmp         opc,nayud
            je          @ayud
            cmp         opc,ndivi
            je          @divi
            cmp         opc,ncose
            je          @cose
            cmp         opc,nabso
            je          @abso
            cmp         opc,nfact
            je          @fact
            cmp         opc,nraiz
            je          @raiz
            cmp         opc,nmult
            je          @mult
            cmp         opc,npote
            je          @pote
            cmp         opc,nrsta
            je          @rsta
            cmp         opc,nsuma
            je          @suma
            cmp         opc,ntang
            je          @tang
            cmp         opc,nsale
            je          @sale
            cmp         opc,nnega
            je          @nega
            cmp         opc,nldpi
            je          @ldpi
            cmp         opc,ngr2r
            je          @gr2r
            cmp         opc,nr2gr
            je          @r2gr
            cmp         opc,nsaca
            je          @saca
```

```
            cmp         opc,nloga
            je          @loga
            cmp         opc,nlnat
            je          @lnat
            cmp         opc,nente
            je          @ente
            cmp         opc,nnpil
            je          @npil
            cmp         opc,nseno
            je          @seno
            ret
compe       endp
;-----Procedimiento que limpia la pila cuando tiene elementos (no mensaje de pila
;---con overflow u underflow)
limpiap     proc        near
            mov         cx,13
            mov         printp,7
clin:
            mov         ax,15
            sub         ax,stapel.indice
            cmp         cx,ax
            je          @point
            cursor      7,printp
            print       clean1
            inc         printp
vas:        loop        clin
            ret
limpiap     endp
;-----Procedimiento que muestra la ayuda en pantalla hasta que se presiona return
help        proc        near
            call        helpi
Readloop:   mov         ah, 0           ;Lee tecla
            int         16h
            cmp         al, 0dh         ;Compara con tecla return (ENTER)
            jne         ReadLoop
            call        interfaz
            jmp         otra
            ret
help        endp
            end         Start
```

Del contenido de la biblioteca “lichanat”:

```
title 'Biblioteca para calculadora de punto flotante (Tzilmiztli)'
;-----MACROINSTRUCCIONES-----
;---Posiciona cursor posición: coordenadas (x,y)
cursor      macro       x,y
            push        x
            push        y
            call        posxy
            endm
;---Interrupcion cursor
curint      macro
            mov         ah, 02h
            int         10h
            endm
;---Imprime marcos posición inicial: coordenadas(x,y), base, altura
cuadro      macro       x,y,b,h
            xor         bh,bh
            push        h
            push        b
            push        y
            push        x
            call        marco
            endm
;---Imprime cualquier cadena de caracteres hasta el fin($)
print       macro       cadenin
            lea         dx,cadenin
            mov         ah,09h
            int         21h
            endm
;Declaración de modelo small y tipo de procesador y coprocesador
            .model      small
            .386                    ;Procesador 80386
            .387                    ;Coprocesador numérico
            .stack      256         ;Tamaño de la pila (stack) en memoria
            .data                   ;Segmento de datos
;.....Variables para proceso read
sign        db    ?
temp1       dw    ?
ten         dt    10.0
numb        dt    0.0
cden        db    29,30 dup (?)
;.....Variables para mostrar la hora y la fecha
diez        db    10
time        db    "hh:mm","$"
date        db    "dd/mm/aaaa","$"
;.....Variables tipo string para mostrar datos y ayuda
clean1      db    26 dup(32),"$"
nombre      db    "..:::: Abakos       Tzilmiztli ::::..$"
goya        db    "Universidad Nacional Aut",162,"noma de M",130,"xico$"
inge        db    "Facultad de Ingenier",161,"a$"
```

```
txt1        db    "Calculadora de punto flotante.$"
txt2        db    "Por Jaime.   Para Estructura y $"
txt3        db    "Programaci",162,"n de Computadoras.$"
txt4        db    "Para m",160,"s informaci",162,"n y ayuda: ayu$"
txt5        db    "Terminar el programa:  off$"
titu        db    "Calculadora Abakos Tzilmiztli, realizada en lenguaje ensamblador.$"
ver         db    "Versi",162,"n    1.0$"
ayud1       db    "Para realizar alguna operaci",162,"n es necesario cargar uno o
m",160,"s $"
ayud2       db    "operandos (n",163,"meros reales) en la pila que se muestra en
pantalla,$"
ayud3       db    "posteriormente util",161,"cese el comando de la funci",162,"n
deseada.$"
ayud4       db    "En la pila pueden almacenarse un m",160,"ximo de 12 operandos.$"
opera       db    175," Operaciones: $"
oper1       db    "Aritm",130,"ticas: sum, res, mul, div, pow(potencia), sqr(ra",161,"z
cuadrada)$"
oper2       db    "Trascendentales: sen, cos, tan, log, lnt(logaritmo natural)$"
oper3       db    "Signo: abs(valor absoluto), neg(negativo)    Valor de ",227,": pii$"
oper4       db    "Redondear al entero cercano: rnd  Factorial (operador entero): fac$"
oper5       db    "Convertir grados a radianes: rad  Convertir radianes a grados: gra$"
oper6       db    "Sacar el valor del tope de la pila: pop    Vaciar la pila: new$"
excep       db    "S",161," se introduce un comando/n",163,"mero inv",160,"lido,"
excep2      db    "carga cero a la pila.$"
siga        db    178,178,178,177,177,176," Presione ",39,"return",39
siga2       db    " para continuar... ",176,177,177,178,178,178,"$"
prompt      db    175,"$"
;.....Variable auxiliar para almacenar dirección de retorno en las
funciones(procedimientos)
retadd      dw          ?
;.....Declaración de caracteres del cuadro
esqSD       equ         187         ;esquina superior derecha
esqSI       equ         201         ;esquina superior izquierda
esqID       equ         188         ;esquina inferior derecha
esqII       equ         200         ;esquina inferior izquierda
elemH       equ         205         ;elemento horizontal
elemV       equ         186         ;elemento vertical
x           db          ?           ;variable de coordenada x
y           db          ?           ;variable de coordenada y
b           db          ?           ;variable de base
h           db          ?           ;variable de altura
;....Variable coordenada y para la impresión de pila de la calculadora
printp      dw          7
            .code             ;Segmento de código
Start:
;-----Declaración de procedimientos públicos de la biblioteca
            public      read, obten, limpia, marco, limpiapv, helpi
            public      posxy, interfaz, imprDate, ConvHora, ConvFec

;:::  PROCEDIMIENTOS    :::
```

```
;-----Procedimiento que lee un número de punto flotante desde el teclado
;---y lo almacena en el tope del stack del coprocesador (FUENTE BREY,BARRY ;TIMP ;3RA)
read        proc
;...almacena un cero (cualquier caso, no instrucción)
            fldz
            inc         bx              ;apunta bx al primer elemento de la cadena
            mov         sign,0          ;clear sign
            call        get             ;read a character
            cmp         al,'-'          ;test for minus
            jne         read1           ;if not miuns
            mov         sign,0ffh       ;set sign for minus
            jmp         read3           ;Get integer part
read1:
            cmp         al,'+'          ;test for plus
            je          read3           ;get integer part
            cmp         al,'0'          ;test for number
            jb          read2
            cmp         al,'9'
            ja          read2           ;if a number
            jmp         read4
read2:
            ret
read3:
            call        get             ;read integer part
read4:
            cmp         al,'.'          ;test for fraction
            je          read7           ;if fraction
            cmp         al,'0'          ;test for number
            jb          read5
            cmp         al,'9'
            ja          read5
            fld         ten             ;form integer
            fmul
            xor         ah,ah
            sub         al,'0'
            mov         temp1,ax
            fiadd       temp1
            jmp         read3
read5:
            cmp         sign,0          ;adjust sign
            jne         read6
            ret
read6:
            fchs
            ret
read7:
            fld1                        ;form fraction
            fld         ten
            fdiv
```

```
read8:
            call  get                   ;read character
            cmp         al,'0'          ;test for number
            jb          read9
            cmp         al,'9'
            ja          read9
            xor         ah,ah
            sub         al,'0'
            mov         temp1,ax
            fild        temp1
            fmul        st,st(1)        ;load number
            fadd        st(2),st        ;form fraction
            fcomp
            fld         ten
            fdiv
            jmp         read8
read9:
            fcomp
            jmp         read5           ;clear stack
            ret
read        endp
;-----Procedimiento que obtiene un caracter de una cadena (para read:proc)
get         proc
            mov         al,[bx]
            inc         bx
            ret
get         endp
;-----Procedimiento que obtiene una cadena del buffer del teclado con eco a pantalla
obten       proc
            lea         dx,cden
            mov         ah,0ah
            int         21h
            mov         bx,dx
            inc         bx              ;apunta al segundo elemento de la cadena
(tamaño de cadena)
            ret
obten       endp
;-----Procedimiento que limpia la pantalla
limpia      proc
            mov         ah,00h
            mov         al,03h
            int         10h
            ret
limpia      endp
;-----Procedimiento que posiciona el cursor en coordenadas x,y
;---Variables: coordenada y, coordenada x en el stack (push y,push x)
```

```
posxy       proc
            pop         retadd          ;almacena la dirección anterior a ser llamada
            pop         ax
            mov         dh, al
            pop         ax
            mov         dl, al
            push        bx
            xor         bx,bx           ;cero a bx, pagina cero de interrupción 10h
            curint
            pop         bx
            push        retadd          ;regresa la dirección a la pila
            ret
posxy       endp
;-----Procedimiento que muestra marcos, texto y fecha en pantalla
interfaz    proc
            call        limpia          ;limpia pantalla
            cuadro      4,1,69,3        ;dibuja marcos
            cuadro      4,6,30,13
            cuadro      37,6,36,4
            cuadro      37,12,36,7
            cuadro      4,21,69,1
            cursor      21,2            ;imprime datos
            print       nombre
            cursor      20,3
            print       goya
            cursor      28,4
            print       inge
            cursor      39,14
            print       txt1
            cursor      39,15
            print       txt2
            cursor      39,16
            print       txt3
            cursor      39,17
            print       txt4
            cursor      39,18
            print       txt5
            cursor      7,22            ;imprime el prompt
            print       prompt
            call        imprDate        ;imprime fecha y hora
            ret
interfaz    endp
;-----Procedimiento que imprime las cadenas Hora y Fecha en la posición indicada
imprDate    proc
            mov         ah, 2Ch         ;interrupción para obtener 1a hora
            int         21h
            call        ConvHora        ;cast de la hora a una cadena
            cursor      54,8
            print       time
```

```
            mov         ah, 2ah          ;interrupción para obtener la fecha
            int         21h
            call        ConvFec          ;cast de la fecha a una cadena
            cursor      51,9
            print       date
            ret
imprDate    endp
;-----Procedimiento que obtiene la Hora y la escribe a una cadena
ConvHora    proc
            mov         ax, 00
            mov         al, ch           ;obtiene las horas en decimal
            div         byte PTR [diez]
            add         al, 30h          ;los convierte a su respectivo caracter
            add         ah, 30h
            lea         bx, time         ;los almacena en la cadena time
            mov         [bx],al
            inc         bx
            mov         [bx],ah
            inc         bx               ;brincamos los puntos
            inc         bx
            mov         ax, 00
            mov         al, cl           ;obtiene los minutos en decimal
            div         byte PTR [diez]
            add         al, 30h          ;los convierte a su respectivo caracter
            add         ah, 30h
            mov         [bx],al          ;los almacena en la cadena time
            inc         bx
            mov         [bx],ah
            ret
ConvHora    endp
;-----Procedimiento que obtiene la Fecha y la escribe a una cadena
ConvFec     proc
            mov         ax, 00
            mov         al, dl           ;obtiene el día del mes en decimal
            div         byte PTR [diez]
            add         al, 30h          ;los convierte a su respectivo caracter
            add         ah, 30h
            lea         bx, date         ;los almacena en la cadena date
            mov         [bx],al
            inc         bx
            mov         [bx],ah
            inc         bx               ;brincamos la diagonal
            inc         bx
            mov         ax, 00
            mov         al, dh           ;obtiene el mes en decimal
            div         byte PTR [diez]
            add         al, 30h          ;los convierte a su respectivo caracter
            add         ah, 30h
            mov         [bx],al          ;los almacena en la cadena date
            inc         bx
            mov         [bx],ah
```

```
            mov         ax, 00
            mov         ax, cx
            add         bx, 05              ;va al fin de cadena para almacenar el año
            div         byte PTR [diez]
            add         ah, 30h             ;lo convierte a su respectivo caracter
            mov         [bx],ah             ;lo almacena en la cadena date
            dec         bx
            mov         ah, 00h
            div         byte PTR [diez]
            add         ah, 30h             ;lo convierte a su respectivo caracter
            mov         [bx],ah             ;lo almacena en la cadena date
            dec         bx
            mov         ah, 00h
            div         byte PTR [diez]
            add         ah, 30h             ;los convierte a su respectivo caracter
            add         al, 30h             ;los almacena en la cadena date
            mov         [bx],ah
            dec         bx
            mov         [bx],al
            ;dec        bx
            ret
ConvFec     endp
;-----Procedimiento que dibuja un marco rectangular en coordenadas x,y ,base y altura
;---específicos, almacenados en el stack como: push h,push b,push y,push x
marco       proc
            pop         retadd              ;almacena la dirección anterior a ser llamada
;.....Línea Horizontal Superior
            pop         ax                  ;almacena variables
            mov         x,al
            pop         ax
            mov         y,al
            pop         ax
            mov         b,al
            pop         ax
            mov         h,al
            mov         dl, x               ;posiciona cursor
            mov         dh, y
            curint
            mov         dl, esqSI           ;imprime elemento de la esquina
            mov         ah, 02h
            int         21h
            mov         dl, x               ;posiciona cursor después de la esquina
            inc         dl
            curint
            mov         cl, b               ;imprime varios elemH (b)
            mov         al, elemH
            mov         ah, 0Ah
            int         10h
            add         dl, b               ;posiciona cursor después de elemH's
```

```
            curint
            mov         dl, esqSD         ;imprime la otra esquina
            mov         ah, 02h
            int         21h
;.....Líneas Verticales
            xor         cx,cx
            mov         cl, h             ;la altura del cuadro
lupeVer:                                  ;loop para imprimir varios elemV
            mov         dl, x             ;posiciona cursor una posición abajo
            inc         dh
            curint
            mov         dl, elemV
            mov         ah, 02h
            int         21h
            mov         dl, x
            add         dl,b
            inc         dl
            curint
            mov         dl, elemV
            mov         ah, 02h
            int         21h
            loop        lupeVer
            inc         dh                ;posiciona cursor una posición abajo
            mov         dl, x
            curint
;.....Línea Horizontal Inferior
            mov         dl, esqII         ;imprime elemento de la esquina
            mov         ah, 02h
            int         21h
            mov         dl, x
            inc         dl                ;posiciona cursor después de la esquina
            curint
            mov         cl,   b           ;imprime varios elemH (b)
            mov         al, elemH
            mov         ah, 0Ah
            int         10h
            add         dl, b             ;posiciona cursor después de elemH's
            curint
            mov         dl, esqID         ;imprime la otra esquina
            mov         ah, 02h
            int         21h
            push        retadd
            ret
marco       endp
;-----Procedimiento que muestra la ayuda en pantalla
helpi       proc        near
            call        limpia
            cuadro      4,1,69,3
            cuadro      4,6,69,16
            cursor      21,2
            print       nombre
```

```
            cursor      20,3
            print       goya
            cursor      28,4
            print       inge
            cursor      7,7
            print       titu
            cursor      30,8
            print       ver
            cursor      7,10
            print       ayud1
            cursor      7,11
            print       ayud2
            cursor      7,12
            print       ayud3
            cursor      7,13
            print       ayud4
            cursor      7,14
            print       excep
            cursor      7,15
            print       opera
            cursor      7,16
            print       oper1
            cursor      7,17
            print       oper2
            cursor      7,18
            print       oper3
            cursor      7,19
            print       oper4
            cursor      7,20
            print       oper5
            cursor      7,21
            print       oper6
            cursor      14,22
            print       siga
            ret
helpi       endp
;-----Procedimiento que limpia la pila cuando se encuentra vacía
limpiapv    proc
            mov         cx,12
            mov         printp,7
clinv:
            cursor      7,printp
            print       clean1
            inc         printp
vasv:       loop        clinv
            ret
limpiapv    endp
            end
```

Conclusiones.

La realización de la calculadora fue de suma ayuda al aprendizaje del lenguaje ensamblador, a utilizar el coprocesador matemático, varias interrupciones y las instrucciones fundamentales que el procesador realiza sobre sus registros y las localidades de memoria, en pocas palabras, a controlar una computadora en un nivel más cercano y vislumbrar todas las posibilidades que significa controlar a este nivel. No solo es mejor en cuanto a tiempo de ejecución y espacio utilizado en la memoria, si no a las posibilidades abiertas.

Un enorme inconveniente es el tiempo que se tiene que dedicar a un programa que con algún otro lenguaje se puede realizar en algunas horas, en ensamblador puede llevar días de diseño y desarrollo. Por otra parte es un lenguaje que al igual que abre posibilidades, cierra otras, que parecería casi imposible resolver dada la complejidad del algoritmo que se necesite.

Realizar toda la programación en bajo nivel parece un trabajo mortal, pero si se combina con modularmente con algún otro lenguaje y se aprovechan las ventajas del ensamblador, pueden realizarse muchas cosas más que si solo se trata de un lenguaje de alto nivel.

La realización de un proyecto aunque pequeño que parezca de esta complejidad y superior es muy reconfortante, dado el tiempo invertido en el esfuerzo.

Lo mejor es que siempre queda – en la calculadora, este caso – algo que hacer, pudiendo mejorar el programa haciéndolo cada vez más robusto e incluso dándole una interfaz gráfica mejor, mejores funciones, cualquier cosa que se pueda ocurrir es posible una vez perdiendo el miedo y adquiriendo confianza. Por eso puede ser un proyecto casi interminable, habría que plantearse un objetivo y tener un límite de lo que se desea, las opciones se ven casi ilimitadas.

Bibliografía

Abel, Peter, IBM PC Assembly Language and Programming, 3ra edición, Englewood Cliffs, New Jersey: Prentice Hall, 1995

Brey, Barry B., *The intel microprocessors : 8086/ 8088, 80186, 80286, and 80486 : architecture, programming, and interfacing,* 3ra edición, New York : Macmillan, 1994

Referencias Web

http://webster.cs.ucr.edu/AoA/DOS/AoADosIndex.html
"The Art of Assembly Language Program"
Consultado desde 05/05/2010 hasta 12/10/2010

http://en.wikipedia.org
"Wikipedia" páginas varias: Abacus, Software development,
Consultado desde 03/10/2010 hasta 12/10/2010

http://www.vocabulario.com.mx
"Vocabulario náhuatl, totonaco"
Consultado desde 05/10/2010 hasta 10/10/2010