

Universidad De San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Arquitectura de Computadores y Ensambladores 1
Sección B



MANUAL TÉCNICO

Juan de Dios de Paz Romero
2016-03041

Introducción

La práctica #3 del laboratorio de arquitecturas y computadoras 1 consiste en la elaboración de una pequeña calculadora elaborada en lenguaje ensamblador. La aplicación tiene como objetivo leer los datos ingresados por los usuarios mediante un archivo de tipo *.arq* el cual mediante un lenguaje de etiquetas, contendrá una serie de operaciones a realizar (operaciones aritméticas básicas); debe tener los operandos y operadores de tal manera que se especifique la precedencia de operaciones. Los datos de los resultados de las operaciones se guardarán, para posteriormente poder ser visualizadas mediante un reporte, dicho reporte será en fichero con extensión *html*.

Objetivos

General:

Brindar al lector una guía, la cual contenga la información del manejo de clases, atributos, métodos y del desarrollo de la interfaz gráfica para facilitar futuras actualizaciones y modificaciones realizadas por terceros.

Específicos:

- Mostrar al lector una descripción lo más completa y detallada posible del SO, IDE entre otros utilizados para el desarrollo de la aplicación.
- Proporcionar al lector una concepción y explicación técnica - formal de los procesos y relaciones entre métodos y atributos que conforman la parte operativa de la aplicación.

Descripción de la Solución

Para poder desarrollar este proyecto se analizó lo que se solicitaba, sus restricciones tanto de lógica, así como operatividad, el ambiente y la amigabilidad de la interfaz para los futuros usuarios de la aplicación.

Entre las consideraciones encontramos con mayor prioridad están:

- Se utilizó para el desarrollo el compilado MASM y para la aplicación se manejó una plataforma de 16 bits tanto en las instrucciones a bajo nivel como en la emulación, esta última debido a emulador DOSBox.
- Para leer y ejecutar las operaciones se decidió hacer un pequeño proceso de compilación considerando primero el análisis léxico y durante el sintáctico la ejecución de las operaciones. Principalmente para la ejecución se creó una variable que consta de una cadena binaria que funcionara como pila al momento de operar las operaciones utilizando la notación infija.

```
bufferentrada db 999 dup('$') ;Guarda la ruta de nuestro archivo
handlerentrada dw ? ;Guarda el numero de archivo abierto
bufferInformacion db 9999 dup('$');Guarda la informacion contenida en el archivo de entrada

lex db 8999 dup('$') ; guarda la tabla de tokens
iSintactico dw 0000h ; guarda la posicion de la tabla analizada
ilexico dw 0000h ; guarda la posicion en el buffer de informacion donde vamos
datos db 99 dup(20 dup('$')) ; guarda el nombre de todos los id's

identi db 99 dup(20 dup('$')) ; guarda los nombres de los identificadores
iidenti dw 0000h

tablaNum db 2000 dup(00h) ; guarda los numeros encontrados
tablaRes db 198 dup(00h) ; guarda los resultados de los id's
pila db 2000 dup(00h)
ipila dw 0000h
ipdatos db 00h ; guarda la posicion en el char
idatos dw 0000h ; guarda la posicion en la tabla datos
pdatos db 30h ; guarda si es el primer caracter
inum dw 0000h ; guarda la posicion del numero
nnum db 30h ; bandera por si es negativo
pnum db 30h ; bandera por si es el primero
ires db 00h ; guarda la posicion del resultado homologa a idatos

saltoL db 10,13,'$' ; salto de linea para consola DOSBox
debuguer db 10,13,'>>> Lleganding <<',10,13,'$'
soloID db 20 dup('$')
cargado db 0
```

Imagen I: Variables utilizadas para el análisis léxico y sintáctico, así como los apuntadores de posición.

```

apilar macro pila, indice, dato
    mov ax,dato
    mov bx,indice
    mov pila[bx],al
    mov pila[bx+1],ah
    mov bx,indice
    add bx,02h
    mov indice,bx
endm

desapilar macro pila, indice, dato
    mov bx,indice
    sub bx,02h
    mov indice,bx
    mov al,pila[bx]
    mov ah,pila[bx+1]
    mov dato,ax
    mov pila[bx],00h
    mov pila[bx+1],00h
endm

```

Imagen II: Macros utilizadas para manejar la cadena binaria que funciona como pila

- La interacción del usuario se vio manejada con el uso buffers de entrada y de salida de información desde y hacia pantalla; además se utilizaron MACROS que manipulaban todos los buffers y las operaciones realizadas entre ellos.
- Todo el código utilizado para la elaboración se encuentra adjunto a este manual para la fácil inspección, análisis posterior y en caso de ser necesario poder realizar cambios en calidad de mantenimiento y optimización.

Sistema Operativo

El sistema operativo en el que se llevó a cabo la realización del proyecto fue Windows 10 de 64 bits

Edición de Windows

Windows 10 Home Single Language

© 2019 Microsoft Corporation. Todos los derechos reservados.

Sistema

Procesador:	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
Memoria instalada (RAM):	8.00 GB (7.87 GB utilizable)
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil:	La entrada táctil o manuscrita no está disponible para esta pantalla

Apéndice

Macros

- Obtención de caracteres

```
getChar macro
    mov ah,01h
    int 21h
endm
```

- Obtención de fecha del sistema:

La fecha es guardada en un arreglo previamente definido como *date*.

```
getDate macro
    mov ah,2ah
    int 21h
    ;day
    mov al, dl
    call convert
    mov date[0], ah
    mov date[1], al
    ;month
    mov al, dh
```

```

        call convert
        mov date[3], ah
        mov date[4], al
        ;year
        ;mov year, cx
endm

```

- **Creación de fichero html:**

Buffer es el nombre del arreglo el cual contendrá el nombre del fichero a crear.

```

crear macro buffer, handler
        mov ah,3ch
        mov cx,00h
        lea dx,buffer
        int 21h
        jc Error4
        mov handler,ax
endm

```

- **Escritura en fichero**

Buffer es el nombre del arreglo el cual contendrá los datos que desea escribirse en el fichero, a su vez es necesario indicar el numero de bits a escribir.

```

escribir macro handler, buffer, numbytes
        mov ah,40h
        mov bx,handler
        mov cx,numbytes
        lea dx,buffer
        int 21h
        jc Error5
endm

```

- **Finalizar la aplicación**

```

exit macro handler
        mov ah,3eh
        mov bx,handler
        int 21h
        jc Error7
endm

```