Manual Tecnico

main.asm

```
.data
    ;******************************************************* Para Interfaz ********************
    margenStart         db '=====================================================', 0aH, 0dH, '
    messageWelcome      db '*  UNIVERSIDAD DE SAN CARLOS DE GUATEMALA              *', 0aH, 0dH, '
    margenEnd           db '*  SECCION: A                                         *', 0aH,  0dH,
    messageOption       db '*  1) CARGAR ARCHIVO                                  *', 0aH, 0dH, '
    messageOptionEnd    db '                              *', 0ah, 0dh,'============================
    messageLoad         db 0ah, 0dh, '=================== CARGAR ARCHIVO ====================', '
    messageInputPath    db 0ah, 0dh, '  :\> Input Path:', '$'
    messageSuccessPath  db 0ah, 0dh, '  File read successfully :)',0ah,0dh, '$'
    messageErrorPath    db 0ah, 0dh, '  Error Read File', '$'
    messageBash         db 0ah, 0dh, '================== CONSOLA ==========================', '
    ReadingKeyboad      db 0ah, 0dh, '  :\> ', '$'
    ErrorReadingKeyboad db 0ah, 0dh, '  COMANDO NO ENCONTRADO', '$'
    messageMedia        db 0ah, 0dh, '  Estadistico Media: ', '$'
    messageModa         db 0ah, 0dh, '  Estadistico Moda: ', '$'
    messageMediana      db 0ah, 0dh, '  Estadistico Mediana: ', '$'
    messageMax          db 0ah, 0dh, '  Estadistico Mayor: ', '$'
    messageMin          db 0ah, 0dh, '  Estadistico Menor: ', '$'
    messageID           db 0ah, 0dh, '  Resultado: ', '$'
    messageErrorOpen    db 0ah, 0dh, '  ERROR: Not is possible open this file', '$'
    messageErrorCreate  db 0ah, 0dh, '  ERROR: Not is possible create the file', '$'
    messageErrorWrite   db 0ah, 0dh, '   ERROR: Not is possible write in the file', '$'
    messageErrorDelete  db 0ah, 0dh, '  ERROR: Not is possible delete the file', '$'
    messageErrorRead    db 0ah, 0dh, '  ERROR: Not is possible read the file', '$'
    messageErrorClose   db 0ah, 0dh, '  ERROR: Not is possible close the file', '$'
    ;*****************************************************************************************
    ;*****************************************************************end for interfaz*********
```

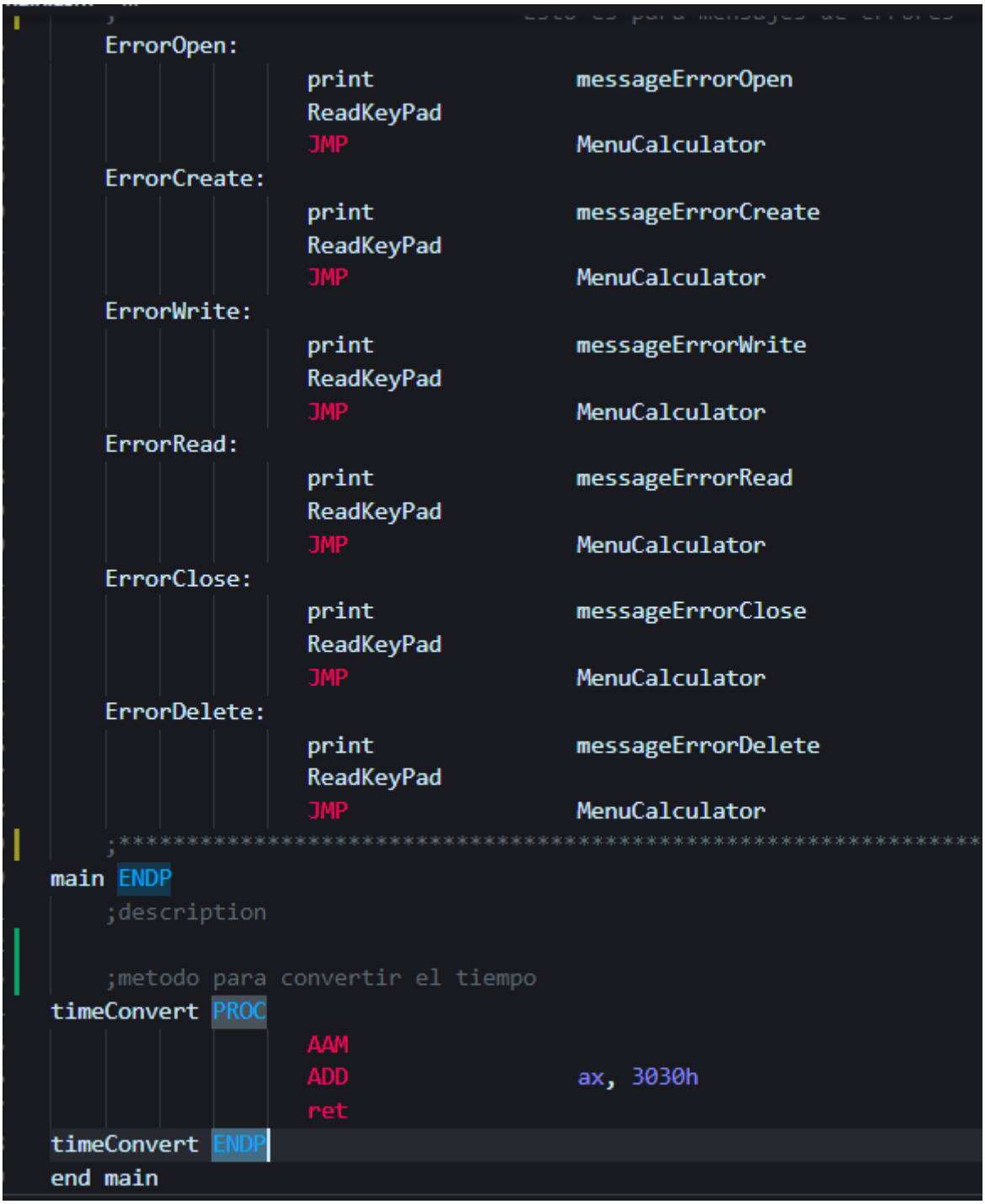
```
;*************************************************************************variables utilizadas*************************
;realiza un salto de linea
newLine             db 0ah, 0dh, '$'
;array que almacenarada el path del archivo de entrada
path                db 100 dup('$')
;array para almacenar el contenido del archivo
contentBufferJSON   db 50000 DUP('$')
;array que tiene como funcion de ser ayuda para almcenara ciertos valores del archivo para realizar operaciones
temp                db 100 DUP('$')
;sigNumber esta variable indica que si el numero es negativo o positivo
sigNumber           db 30h
;variable de tipo word para almacenar la operacion1
operation1          dw 00h, '$'
;variable de tipo word para almacenar la operacion2
operation2          dw 00h, '$'
;variable de tipo word para prubas de si estaba operando bien xD
operationTest       dw 00h, '$'
;variable que almacenara el signo menos
sign                db 00h, '$'
;variable utilizada para los archivos
handleFile          dw ?
;variable utilizada para almacenar el nombre del reporte
pathFile            db 50 DUP('$')
;variable que recibe el nombre del reporte
parentName          db 30 DUP('$')
;variable para mostrar el size de las variables, seria equivalente a un length xd
parentNameSize      dw 0
;array usado para almacenar los nombres de las operaciones
listNumbers         db 250 dup('$')
;array usado para almacenar los resultados de las operaciones
listValues          dw 250 dup('$')

;array para el resultado
resNumber           db 300 dup('$')
;variable para verificar si ya se almaceno un padre id
verifyPath          db 30h
;variable para contadores de ids
counterNumbers      dw 0
;variable para contadores de los valores de las operaciones
counterValue        dw 0
```

```
;*********************************************Para el reporte*************************************************************************************************
studentJSON         db '{', 0ah, 0dh, 09h, '"reporte": {', 0ah, 0dh, 09h, 09h, '"alumno": {', 0ah, 0dh, 09h, 09h, 09h, '"Nombre": "Ju
dayDateJSON         db 0ah, 0dh, 09h, 09h, '"Fecha": {', 0ah, 0dh, 09h, 09h, 09h, '"Dia": '
monthDateJSON       db ',', 0ah, 0dh, 09h, 09h, 09h, '"Mes": '
yearDateJSON        db ',', 0ah, 0dh, 09h, 09h, 09h, '"Año": 2020'
hourDateJSON        db 0ah, 0dh, 09h, 09h, '},', 0ah, 0dh, 09h, 09h, '"Hora": {', 0ah, 0dh, 09h, 09h, 09h, '"Hora": '
minuteDateJSON      db ',', 0ah, 0dh, 09h, 09h, 09h, '"Minutos": '
secondsDateJSON     db ',', 0ah, 0dh, 09h, 09h, 09h, '"Segundos": '
promResultJSON      db 0ah, 0dh, 09h, 09h, '},', 0ah, 0dh, 09h, 09h, '"resultados": {', 0ah, 0dh, 09h, 09h, 09h, '"Media": '
medianaResultJSON   db ',', 0ah, 0dh, 09h, 09h, 09h, '"Mediana": '
modaResultJSON      db ',', 0ah, 0dh, 09h, 09h, 09h, '"Moda": '
minResultJSON       db ',', 0ah, 0dh, 09h, 09h, 09h, '"Menor": '
maxResultJSON       db ',', 0ah, 0dh, 09h, 09h, 09h, '"Mayor": '
operationsJSON      db 0ah, 09h, 09h, '},', 0ah, 09h, 09h, '"'
operations1JSON     db '": ['
operations2JSON     db 0ah, 09h, 09h, 09h, '{', 0ah, 09h, 09h, 09h, 09h
doubleQuotes        db '"'
doubleDot           db ": "
operations3JSON     db 0ah, 09h, 09h, 09h, '},'
endJSON             db 0ah, 09h, 09h, ']', 0ah, 09h, '}', 0ah, '}'
dayDate             db 'dd'
monthDate           db 'mm'
hourDate            db 'hh'
minuteDate          db 'mm'
secondDate          db 'ss'
mediaAns            db '#'
medianaAns          db '#'
modaAns             db '#'
minAns              db '#'
maxAns              db '#'
;*************************************************************************************************************************************************************
;*****************************************end content for report*********************************************************************************************
```

```
.code
main PROC
    ;esto es para incializar el programa
                MOV             AX, @data
                MOV             DS, AX

    MenuCalculator:
    ;*********************impresion de la interfaz end dosBox*********************
                print           margenStart
                print           messageWelcome
                print           margenEnd
                print           messageOption
    ;lee la consola
                ReadKeyPad
                print           messageOptionEnd
    ;*****************************************************************
    ;compara que opcion fue usada, 1
                CMP             al, 31h
                JE              LOAD
    ;compara que opcion fue usada, 2
                CMP             al, 32h
                JE              BASH
    ;compara que opcion fue usada, 3
                CMP             al, 33h
                JE              EXIT

    LOAD:
    ;*****************limpia las variables utilizadas para la ejecucion de operaciones**********
                clearString     path
```

```
LOAD:
;****************limpia las variables utilizadas para la ejecucion de operaciones**********
                clearString         path
                clearString         pathFile
                clearString         contentBufferJSON
                clearString         temp
                clearString         listNumbers
                clearString         listValues
;*************************************************************************************************

;inicializa los contadores  en 0
                mov                 counterNumbers, 0
                mov                 counterValue, 0

;*************************para cargar el archivo*****************************************************
                print               messageLoad
                print               messageInputPath
;pide la ubicacion del archivo a cargar
                GetPathFile         path
;abre el archivo
                openFile            path, handleFile
;lee el archivo
                readFile            contentBufferJSON, handleFile, SIZEOF contentBufferJSON
;cierra el archivo
                closeFile           handleFile
;*************************************************************************************************

;analiza el archivo de entrada y realiza las operaciones necesarias
                simpleWhileAnalisis contentBufferJSON
;genera el reporte
                generateReport
;vuelve al menu principal
                JMP                 MenuCalculator
```

```
                JMP                 MenuCalculator
BASH:
;aqui es para ingresar los comandos especiales
                print               messageBash
                print               ReadingKeyboad
                ReadKeyPad
;regresa al menu principal
                JMP                 MenuCalculator
EXIT:
;se acaba el programa
                MOV                 AH, 4ch
                INT                 21h
                print               messageOptionEnd

;*******************************Esto es para mensajes de opp
```



```
    ErrorOpen:
                print           messageErrorOpen
                ReadKeyPad
                JMP             MenuCalculator
    ErrorCreate:
                print           messageErrorCreate
                ReadKeyPad
                JMP             MenuCalculator
    ErrorWrite:
                print           messageErrorWrite
                ReadKeyPad
                JMP             MenuCalculator
    ErrorRead:
                print           messageErrorRead
                ReadKeyPad
                JMP             MenuCalculator
    ErrorClose:
                print           messageErrorClose
                ReadKeyPad
                JMP             MenuCalculator
    ErrorDelete:
                print           messageErrorDelete
                ReadKeyPad
                JMP             MenuCalculator
    ;*****************************************************************
main ENDP
    ;description

    ;metodo para convertir el tiempo
timeConvert PROC
                AAM
                ADD             ax, 3030h
                ret
timeConvert ENDP
end main
```

file.asm

```
;GetPahtFile sirve para introducir el directorio donde se encuentre algun archivo
GetPathFile MACRO array
                LOCAL       getCadena, endCadena, RemoveSpace
                ; setea el registro SI con 0
                MOV         SI, 0
    getCadena:
                ;lee la consola para introducir la ubicacion del archivo
                ReadKeyPad
                ;compara si hay un enter que seria un retorno de carro
                CMP         AL, 0dh
                JE          endCadena
                ;esto es por si se metio mal alguna letra poder borrar la de atras
                CMP         AL, 08h
                JE          RemoveSpace
                ;se mueve al path el contenido
                MOV         array[SI], AL
                ;incrementa SI
                INC         SI
                ;se vuelve a repetir el ciclo
                JMP         getCadena
    RemoveSpace:
                ;mueve un fin de cadena a AL
                MOV         AL, 24h
                ;decrementa SI SI--
                DEC         SI
                ;mueve el caracter al path una casilla anterior
                MOV         array[SI], AL
                ;regresa al getCadena para seguir anadiendo al path
                JMP         getCadena
    endCadena:
                ;termina de ingresar el path
                MOV         array[SI], 00h
ENDM
```

```
openFile MACRO file, handler
            MOV AH, 3dh         ;abre el archivo
            MOV AL, 10b     ;Acceso lectura/escritura
            lea DX, file        ;realiza una lectura con la informacion
            int 21h
            MOV handler, AX     ;transfiere la informacion al handler
            jc  ErrorOpen       ;si hay carry lanzara un error ya que no pudo abrir el archivo con exito
ENDM
```

```
createFile MACRO file, handler
              MOV AH, 3ch       ;crea el archivo
              MOV CX, 00h       ;mueve al inicio del archivo
              lea DX, file      ;realiza una lectura con la informacion
              int 21h
              MOV handler, AX   ;transfiere la informacion al handler
              jc  ErrorCreate   ;si hay carry lanzara un error ya que no se creo con exito el archivo
ENDM
```

```
;macro para escribir el archivo
writeFile MACRO array, handler, numBytes
                PUSH CX
                PUSH DX

                MOV  AH, 40h
                MOV  BX, handler
                MOV  CX, numBytes
                lea  DX, array
                int  21h
                jc   ErrorWrite

                POP  DX
                POP  CX
ENDM
```

```
readFile MACRO array, handler, numBytes
                MOV AH, 3fh
                MOV BX, handler
                MOV CX, numBytes
                lea DX, array
                int 21h
                jc  ErrorRead
                print messageSuccessPath
ENDM

;macro para cerrar el archivo
closeFile MACRO handler
                 MOV AH, 3eh
                 mov handler, BX
                 int 21h
                 jc  ErrorClose
ENDM
;macro para eliminar el archivo
deleteFile MACRO buffer
                  MOV AH, 41h
                  lea DX, buffer
                  jc  ErrorDelete
ENDM
```

analyzer.asm

```
simpleWhileAnalisis MACRO buffer
                    LOCAL       While, Continue, endW, IDS, SaveID, SaveFatherC, SaveWhile, endWSF, seachNumber, WhileNum, endWNumber
                    LOCAL       checkOp, div0, div1, div2, div3, mul0, mul1, mul2, mul3, add0, add1, add2, add3, sub0, sub1, sub2, sub3
                    LOCAL       id0, id1, id2,  Whilecheck, WhileSI, endWhileSI, searchTrash, negNumbers
                    LOCAL       PRODUCT, ADITION, SUBSTRACTION, DIVISION, continueOpeartion, saveOperation, notOperations, negNumbers2
                    mov         si, 0
                    mov         cx, 0
                    mov         [verifyPath], 30h
                    mov         ax, 0
                    mov         ah, '^'
                    PUSH        ax
    While:
                    mov         dh, buffer[si]
                    cmp         dh, 22h  ;aqui verificamos si vienen IDS, para guardarlos para el uso del a consola xd
                    je          IDS
                    jmp         Continue
    Continue:
                    cmp         dh, '$' ;transfiere la direccion al registro data
                    je          endW
                    inc         si    ;verifica si hay mas " para seguir guardando los ids, para las operaciones hijos xd
                    jmp         While
    IDS:
                    inc         si
                    mov         dh, buffer[si]
                    cmp         dh, 22h  ;ahora si verificamos si encuentra la " de cierre para guardar los IDS
                    je          Whilecheck

                    cmp         dh, 23h  ;esto es para las variables que declare en el main con # donde iran los resultados xd
                    je          seachNumber

                    PUSH        SI    ;guardamos la posicion
                    mov         si, 0
                    mov         si, cx
                    mov         temp[si], dh
                    inc         cx      ;incrementamos el contador para el registro del ciclo xd
```

```
    Whilecheck:         ...
    SaveID:             ...
    WhileSI:            ...
    endWhileSI:         ...
;aqui hace el analisis de las palabras div, add, sub, mul, id
    div0:               ...
    div1:               ...
    div2:               ...
    div3:
    add0:               ...
    add1:               ...
    add2:               ...
    add3:               ...
    sub0:               ...
    sub1:               ...
    sub2:               ...
    sub3:               ...
    mul0:               ...
    mul1:               ...
    mul2:               ...
    mul3:               ...
    id0:                ...
    id1:                ...
    id2:                ...
    ;salva el padre id
    SaveFatherC:        ...
    SaveWhile:          ...
    endWSF:             ...
;
```

```
    ;salva el padre id
    SaveFatherC:        ...
    SaveWhile:          ...
    endWSF:             ...
;
    seachNumber:
    searchTrash:        ...
    WhileNum:           ...
    negNumbers:         ...
    endWNumber:         ...
    negNumbers2:        ...
    notOperations:      ...
    saveOperation:      ...
    continueOpeartion:  ...
    ADITION:            ...
    SUBSTRACTION:       ...
    DIVISION:           ...
    PRODUCT:            ...
    endW:               ...
ENDM
```

macros.asm

```
print MACRO cadena
            LOCAL       flagPrint
            SetData
    flagPrint:
            MOV         AH, 09h             ;ALmacena la poSIcion esto le indica a ah que escriba en consola
            MOV         DX, offset cadena   ;ALmacena la direccion en el registro DX
            int         21h                 ;para el kernel de dos
            RemoveData
ENDM

ReadKeyPad MACRO
             MOV AH, 01h  ;ALmacena un 1 en AH que esto indica que se ponga en modo de escritura
             int 21h      ;para el kernel de dos
ENDM
```

```
;macro para obtener el texto completo
GetText MACRO buffer
                LOCAL       getCadena, moveSpace, EndC
                SetData

                MOV         SI, 0
    getCadena:
                ReadKeyPad
                CMP         AL, 0dh
                JE          EndC
                CMP         AL, 08h
                JE          moveSpace
                MOV         buffer[SI], AL
                INC         SI
                JMP         getCadena

    moveSpace:
                MOV         AL, 24h
                dec         SI
                MOV         buffer[SI], AL
                JMP         getCadena

    EndC:
                MOV         AL, '$'
                MOV         buffer[SI], AL
                RemoveData
ENDM
```

```
;macro para limpiar la cadena o variables utilizando
clearString MACRO buffer
                LOCAL       repeatClear
                SetData
                mov         si, 0
                mov         cx, 0
                mov         cx, SIZEOF buffer

    repeatClear:
                mov         buffer[si], '$'
                inc         si
                LOOP        repeatClear
                RemoveData
ENDM
```

```
;macro para convertir int a string
splitText MACRO array, seek ···
;macro para compara cadenas
compareString MACRO buffer, command, equal ···
;macro para convertir un string (si se le puede llmar de esta manera) a un ascii (caracter)
intToString MACRO number, output
            LOCAL       beginConv, EndConv
            SetData
            mov         AX, 0
            mov         bx, 0
            mov         cx, 0
            mov         bx, 10    ;este 10 sirve como tempiliar para unir los numeros a uno solo por unidades, decenas etc
            mov         si, 0

    beginConv:
            mov         CL, number[si] ;mueve el valor del numero en la poscion SI al registro cl
            cmp         CL, 30h ;compara si ya termino de leer el string
            jl          EndConv
            cmp         CL, 39h ;compara si ya termino de leer el string
            jg          EndConv
            inc         si ;incrementa si
            sub         cl, 30h ;compara si ya termino de leer el string
            mul         bx ;realiza la multiplicacion para la conversion
            add         ax, cx ;suma ax y cx
            JMP         beginConv ;se va a finalizar la conversion
    EndConv:
            mov         output, ax ;mueve a salida el registro ax
            RemoveData ;libera la informacion en la pila
ENDM
```

```
;macro para obtener un numero
getNumber MACRO buffer
            LOCAL       beginInt, endInt
            mov         si, 0
    beginInt:
            ReadKeyPad
            cmp         al, 0dh             ;compara si hay un reset
            je          endInt              ;si es asi es porque ya termino la cadena
            mov         buffer[si], al      ;movemos lo que hay de al asi la posicion del buffer
            inc         si
            JMP         beginInt            ;repite lo mismo si es posible
    endInt:
            mov         buffer[si], 00h     ;se verifica si ya no hay nada
ENDM


;macro para almcenar informacion de los registros en la pila
SetData MACRO
            PUSH AX ;almacena ax en la pila
            PUSH BX ;almacena bx en la pila
            PUSH CX ;almacena cx en la pila
            PUSH DX ;almacena dx en la pila
            PUSH SI ;almacena si en la pila
            PUSH DI ;almacena di en la pila
ENDM
;macro para liberar la informacion de los registro de la pila
RemoveData MACRO
             POP DI ;libera di de la pila
             POP SI ;libera si de la pila
             POP DX ;libera dx de la pila
             POP CX ;libera cx de la pila
             POP BX ;libera bx de la pila
             POP AX ;libera ax de la pila
ENDM
```

Time.asm

```
getDate MACRO
    SetData ;salva la informacion en la pila
    MOV AH, 2ah ;obtiene la fecha de sitema operativo (windows)
    int 21h ;realiza una interrupcion 21

    MOV AL, DL
    CALL timeConvert ;llama al metodo que convierte el tiempo
    mov dayDate[0], AH ;almcena en el arreglo de dia el valor en la posicion 0
    mov dayDate[1], AL ;almcena en el arreglo de dia el valor en la posicion 1

    mov al, dh
    CALL timeConvert        ;llama al metodo que convierte el tiempo
    mov monthDate[0], AH    ;almcena en el arreglo de mes el valor en la posicion 0
    mov monthDate[1], al     ;almcena en el arreglo de mes el valor en la posicion 1
    RemoveData ;libera la informacion de la pila
ENDM
```

```
getTime MACRO
    SetData ;almacena la informacion en la pila
    mov ah, 2ch ;obtiene el tiempo actual del sistema
    int 21h ;realiza una interrupcion 21

    mov al, ch
    CALL timeConvert ;llama al metodo que convierte el tiempo
    mov hourDate[0], ah ;almcena en el arreglo de hora el valor en la posicion 0
    mov hourDate[1], al ;almcena en el arreglo de hora el valor en la posicion 1

    mov al, cl
    CALL timeConvert ;llama al metodo que convierte el tiempo
    mov minuteDate[0], ah ;almcena en el arreglo de minuto el valor en la posicion 0
    mov minuteDate[1], al ;almcena en el arreglo de minuto el valor en la posicion 1

    mov al, dh
    CALL timeConvert ;llama al metodo que convierte el tiempo
    mov secondDate[0], ah ;almcena en el arreglo de segundo el valor en la posicion 0
    mov secondDate[1], al ;almcena en el arreglo de segundo el valor en la posicion 1

    RemoveData ;libera la informacion de la pila
ENDM
```

report.asm

```
generateReport MACRO
                getDate ;obtiene la fecha del dia actual
                getTime ;obtiene el tiempo de dia actua

                deleteFile          pathFile ;elimina el archivo si es que existe en el directorio
                createFile          pathFile, handleFile ;recrea el archivo si lo es necesario
                openFile            pathFile, handleFile ;abre el archivo en mode escritura

                ;*********************************Escribe el cuerpo del reporte*********************************
                writeFile           studentJSON, handleFile, SIZEOF studentJSON
                writeFile           dayDateJSON, handleFile, SIZEOF dayDateJSON
                writeFile           dayDate, handleFile, SIZEOF dayDate
                writeFile           monthDateJSON, handleFile, SIZEOF monthDateJSON
                writeFile           monthDate, handleFile, SIZEOF monthDate
                writeFile           yearDateJSON, handleFile, SIZEOF yearDateJSON
                writeFile           hourDateJSON, handleFile, SIZEOF hourDateJSON
                writeFile           hourDate, handleFile, SIZEOF hourDate
                writeFile           minuteDateJSON, handleFile, SIZEOF minuteDateJSON
                writeFile           minuteDate, handleFile, SIZEOF minuteDate
                writeFile           secondsDateJSON, handleFile, SIZEOF secondsDateJSON
                writeFile           secondDate, handleFile, SIZEOF secondDate
                writeFile           promResultJSON, handleFile, SIZEOF promResultJSON
                writeFile           mediaAns, handleFile,  SIZEOF mediaAns
                writeFile           medianaResultJSON, handleFile, SIZEOF medianaResultJSON
                writeFile           medianaAns, handleFile, SIZEOF medianaAns
                writeFile           modaResultJSON, handleFile, SIZEOF modaResultJSON
                writeFile           modaAns, handleFile, SIZEOF modaAns
                writeFile           minResultJSON, handleFile, SIZEOF minResultJSON
                writeFile           minAns, handleFile, SIZEOF minAns
                writeFile           maxResultJSON, handleFile, SIZEOF maxResultJSON
                writeFile           maxAns, handleFile, SIZEOF maxAns
               ;*************************************************************************************************

                writeFile           operationsJSON, handleFile, SIZEOF operationsJSON ;escribe el padre de las operaciones
                lengthString16BITS  parentName, parentNameSize ;pone el id correspondiente la padre del archivo
                writeFile           parentName, handleFile, parentNameSize ;obtiene de la longitud del padre

                generatePartOperations ;genera las operaciones del archivo operacion1, operacion2 etc

                writeFile           endJSON, handleFile, SIZEOF endJSON ;termina de escribir el archivo



                closeFile           handleFile ;cierra el archivo
```

```
lengthString16BITS MACRO buffer, incBuffer
                    LOCAL       While, endString
                    SetData
                    mov         si, 0               ;limpiamos el contador
    While:
                    mov         dh, buffer[si]      ;almacenamos en dh el caracter de la cadena a obtener la longitud
                    cmp         dh, '$'             ;verifica si dh ya tiene el $
                    je          endString           ; si lo tiene entonces pues terminamos de iterar el while xd
                    inc         si                  ;incrementamos en caso de que aun no haya $
                    jmp         While               ;vuelve a iterar
    endString:
                    mov         incBuffer, si       ;almacenamos el size del si en incBuffer para indicar la longitud de la cadena
                    RemoveData

ENDM
```

```
lengthString32BITS MACRO buffer, incBuffer
                    LOCAL       While, endString
                    SetData
                    mov         si, 0               ;limpiamos el contador
    While:
                    mov         dx, buffer[si]      ;almacenamos en dh el caracter de la cadena a obtener la longitud
                    cmp         dx, '$'             ;verifica si dh ya tiene el $
                    je          endString           ; si lo tiene entonces pues terminamos de iterar el while xd
                    inc         si                  ;incrementamos en caso de que aun no haya $
                    jmp         While               ;vuelve a iterar
    endString:
                    mov         incBuffer, si       ;almacenamos el size del si en incBuffer para indicar la longitud de la cadena
                    RemoveData

ENDM
```

```
generatePartOperations MACRO
                        LOCAL               start, sEnd, While1, endWhile1, While2


                       ;empieza a escribir { y la estructura para las operaciones
                        writeFile           operations1JSON, handleFile, SIZEOF operations1JSON

                       ;********************SETEAMOS los registros SI, DI y cX con 0 ***********
                        mov                 si, 0
                        mov                 di, 0
                        mov                 cx, 0
                       ;*********************************************************************

    start:              ;movemos el id de las lista de ids al registro dh en la posicion que tiene SI
                        mov                 dh, listNumbers[si]
                        ;compara si ya la lista no tiene IDS
                        cmp                 dh, '$'
                        ;de ser asi termina el proceso en sEnd
                        je                  sEnd
                        ;estos es para escribir lods ids en lso reportes por ejemplo  ,"
                        writeFile           operations2JSON, handleFile, SIZEOF operations2JSON
                        writeFile           doubleQuotes, handleFile, SIZEOF doubleQuotes

    While1:
                       ;movemos el valor de los ids a dh
                        mov                 dh, listNumbers[si]
                       ;compara si en temp existe el valor tope que separa cada id
                        mov                 temp, dh
                        cmp                 temp, '^'
                       ;si lo tiene se salta a mover la operacion para el reporte
                        je                  endWhile1
                        ;incrementa si
                        inc                 si
                        ;obtiene la longitud del id
                        lengthString16BITS temp, parentNameSize
                        ; escribe el id en el archivo
                        writeFile           temp, handleFile, parentNameSize
                        jmp                 While1
```

```
                        jmp                 While1
    endWhile1:
                       ;realiza un cierre de comillas para los ids
                        writeFile           doubleQuotes, handleFile, SIZEOF doubleQuotes
                        writeFile           doubleDot, handleFile, SIZEOF doubleDot

                       ;convierte el valor a strign de la lista de valores de las operaciones
                        splitText           operationTest, listValues[di]
                       ;obtiene la longitud de la operacion
                        lengthString32BITS operationTest, parentNameSize
                       ;escribe el resultado de la opearcion en el reporte
                        writeFile           operationTest, handleFile, parentNameSize
                        ;limpia la variable
                        clearString         operationTest
                        ;incremetno de di dos veces por que es de tipo dw
                        inc                 di
                        inc                 di
                       ;ciera la operacion hija y continua leyendo las lista y escribiendo en el reporte si es que encuentras mas
                        writeFile           operations3JSON, handleFile, SIZEOF operations3JSON

                        inc                 SI
                        JMP                 start
    sEnd:
```