Universidad de San Carlos de Guatemala

Arquitectura de computadores y ensambladores 1

Escuela de Sistemas

Ing. Otto Rene Escobar Leiva

Aux Frederick Jonathan Faugier Pinto

Manual Técnico:

PRACTICA 2:

Nombre: John Henry López Mijangos

Carnet: 201710392

Guatemala 23 de JUNIO del 2022

INTRODUCCIÓN

El objetivo de esta práctica fue el desarrollo de una calculadora, esto a través de la manipulación de registros para la multiplicación,división, suma,resta y elevación de números, tomando en cuenta tanto el símbolo como el rango máximo de estos siendo usados números de hasta 2 bytes máximo, siendo también analizado los estadísticos como media, mediana, moda, números pares e impares y números primos de forma paralela al ingresar cada número . Se comprendió tanto la necesidad de limpiar y guardar de forma correcta los registros, así como la lectura de archivos externos con un formato en específico.

Objetivos

Objetivo General

- Aplicar los conocimientos adquiridos en el curso sobre el lenguaje ensamblador.

Objetivos Específicos

- Aplicar el conocimiento de operaciones básicas a nivel ensamblador.
- Conocer el funcionamiento de las interrupciones.
- Comprender el uso de la memoria en los programas informáticos.
- Consolidar los conocimientos de escritura\lectura de archivos.

Explicación del código:

```
.model small
.stack 100h
.data
```

Flujo menú: se le solicita al usuario escoja una opción, para indicarle al programa que realizar, se uso una comparación (cmp) para determinar qué número era el leído para posteriormente escoger alguna de las tres opciones.

```
main proc

        MOV ah,09h
        MOV dx,@data
```

```
    MOV ds, dx
    print encabezado

Ingresar:
    print menu
    XOR al, al
    MOV ah, 01h
    INT 21h
    CMP al, 31h  ; 1
    JE CargarArchivo
    CMP al, 32h  ; 2
    JE Consola
    CMP al, 33h  ; 3
    JE Salir
    XOR al, al
    JMP Ingresar


CargarArchivo:


    JMP Ingresar

Consola:
    print encConsola
    obtenerComandos
    JMP Ingresar

Salir:
    MOV ah,4ch
    INT 21h
```

Macros:

- OpenF: Abre un archivo.
- CloseF: Cierra un archivo
- WriteF: Escribe sobre un archivo el numero de bytes que se desee.
- CrearF: Crea un nuevo archivo.
- ComprarCadenas: Compara si dos cadenas son iguales
- calcularMedia: Calcula la media de los resultados obtenidos.
- calcularMayor: Calcula el mayor de los resultados obtenidos
- getChar: Lee un char desde el teclado.

- buscandoID: busca un ID para mostrar el resultado
- ConvertirAscci: Convierte un buffer a su equivalente en ASCII.
- ConvertirString: Convierte el numero guardado en el registro ax para poder mostrarlo en pantalla. Transferir: Transfiere lo que hay en un buffer a otro.
- crearReporte: Crear el reporte con sus nuevos datos.
- showDate: Obtiene la fecha para colocarlo en el reporte.
- ObtenerOperaciones: Obtiene las operaciones para colocarlo en el reporte
- ObtenerResultados: Obtiene los resultados para colocarlo en el reporte
- limpiarBuffer: Limpia un arreglo colocando '$' en cada posición
- leyendoJSON: Es el analizador que obtiene los resultados de las operaciones.
- print: Imprime en pantalla
- opcion1: Es la opción para cargar un archivo.

mCalculadora: se le solicitará al usuario teclee una opción para luego de darle enter determinar la opción decidida, si son operandos (+,/,x) se procederá a indicarle al programa que saque dos elementos de la pila para realizar la operación indicada y ingresarle el resultado a la pila, si no es ninguno de los elementos significa que es un número por lo cual se procederá con una macro a pasar de string a numero dicho valor para posteriormente ser ingresados a la cola, si el primer elemento tiene un "-" o bien está dando inicio a un número negativo o a una resta, por dicho motivo si se presenta se procederá a mandar a un bucle con el cual se procederá en determinar esto , si es un número negativo se elimina el elemento "-" se pasa a un número legible positivo y luego se niega para ponerlo como negativo para posteriormente ingresarlo a la pila, en cambio si solo es una barra "-" se procederá a realizar la misma acción que los operandos anteriormente mencionado solo que ahora restando.

```
Suma:
            ;print suma1
            MOV ax, cx
            ADD ax, bx

            XOR cx, cx
            MOV cx, ax
            PUSH ax
            MOV ax, cx
            ;ConvertirString bufferAux
            ;print saltoLinea
            ;print bufferAux
            MOV contadorNumero, 1
            JMP revisarPila
```

```
        XOR cx, cx
        INC si
        JMP CICLO
```

```
                49 references
1390 >      Suma:           ···
                13 references
1409 >      Resta: ···
                13 references
1436 >      Multiplicacion: ···
                13 references
1456 >      Division: ···
                44 references
1478 >      revisarPila: ···
                44 references
1508 >      regresarPila: ···
```

Variables y Cadenas utilizadas

```
ca > ASM nuevo.asm > finReporte
    inicioResultados db 0ah, '      "resultados":',
                        0ah, '      {',
                        0ah, '          "Media": '
    4 references
    bufferMediana db 0ah,    '          "Mediana": " "'
    4 references
    bufferMenor db 0ah,      '          "Menor": '
    4 references
    bufferMayor db 0ah,      '          "Mayor": '
    0 references
    inicioPadre db 0ah,    '        "operaciones":'
    4 references
    comilla db 0ah, '       "'
    4 references
    cierreComillas db  '":'
    4 references
    cierreComillas2 db '":', '$'
    4 references
    inicioArreglo db 0ah,  '        ['
    4 references
    finArreglo db 0ah,     '        ]'
    29 references
    coma db ','
    4 references
    coma2 db ',','$'
    4 references
    inicioOR db 0ah, 09h, 09h, 09h,'{',
                0ah, 09h, 09h, 09h, 09h, '"', '$'
    4 references
    finOR db    0ah, 09h, 09h, 09h ,'}', '$'
```

Comparar Strings: recorre dos cadenas al mismo tiempo hasta que el corrimiento llegue al símbolo dólar en ambas al mismo tiempo, si no sucede entonces no son iguales, afectando a una variable global que indica su igualdad.

```
compararCadenas2 macro cadena1, cadena2, cantidad,resul
    LOCAL SALIR

    ;PUSH si
    ;XOR si, si
    ;MOV si, cx

    MOV al, 0
    LEA si, cadena1
    LEA di, cadena2
    PUSH ds
    POP es
    CLD
    MOV cx, cantidad
    ;POP si

    REPE CMPSB
    JA SALIR
    JB SALIR
    MOV al, 1


    SALIR:
endm
```

LeerJson: el truco para la lectura de json es encontrar las doble comillas y leer las variables, tanto los números como variables leídas letra por letra fueron capturados a través de métodos especiales presentados más adelante, primero revisara que el documento cumpla con las características previamente establecidas en el instructivo del proyecto y que simularán un JSON, se ira buscando las comillas para posteriormente capturar el id que albergan estas, decidiendo que realizar dependiendo del id, ya sea el ingreso de un numero o bien una operación.

```
leyendoJSON  macro buffer
        LOCAL SALIR, BuscarID, guardarID, guardarPadre, ObtenerNumero, BuscarNumero , Operacion, Multiplicacion
        LOCAL IngresarID, CONTINUE, EndNumero, abrirLlave, cerrarLlave, FinOperacion, Division, Suma, Resta
        LOCAL operarFIN, guardaIDoperacion, Csuma, Cresta, Cmultiplicacion, Cdivision, guardarOperaciones
        LOCAL numeroNegativo
        limpiarBuffer bufferOperaciones
        limpiarBuffer nombrePadre
        limpiarBuffer nombreReporte
        limpiarBuffer bufferAux
        limpiarBuffer nombreOperacion
        limpiarBuffer operaciones
        limpiarBuffer bufferMediaR
        limpiarBuffer bufferMedianaR
        limpiarBuffer bufferMayorR
        limpiarBuffer bufferMenorR
        limpiarBuffer bufferModaR

        XOR si, si
        XOR cx, cx
        XOR ax, ax
        XOR bx, bx
        XOR dx, dx
        MOV contadorPadre, 0
        MOV finOpe, 0
        MOV contadorLLaves, 0
        MOV nega2, 0
        MOV contadorNumero, 0
        MOV totalOperaciones, 0
        MOV contadorguardar, 0
```

```
191 references
CICLO: ···

CONTINUE:        ···

abrirLlave: ···

cerrarLlave: ···

FinOperacion: ···

guardarOperaciones: ···

validar_guardado: ···

base1: ···

base2: ···

base3: ···

base4: ···

base5: ···

base6: ···

base7: ···

base8: ···

base9: ···
```

mCapturarNumero: se recorre cada posición de archivo a sabiendas de estar encima de un decimal hasta hallar una coma, posteriormente se procederá a salir del método

mFechaTime: Llama a interrupciones con las cuales es posible obtener la fecha y hora actuales, posteriormente obtenido los números se procederá a colocar en strings para posteriormente ser impresos en un documento externo.

```
HORA:
        MOV si, 0
        MOV ah, 2ch
        int 21h
```

```
    mov al, ch
    aam
    mov bx, ax

    mov dl, ':'
    MOV bufferFecha[si],dl
    inc si

MINUTOS:
    MOV AH,2CH    ; To get System Time
    INT 21H
    MOV AL,CL     ; Minutes is in CL
    AAM
    MOV BX,AX

    mov dl, '-'
    MOV bufferFecha[si],dl
    inc si

MES:
    MOV AH,2AH    ; To get System Date
    INT 21H
    MOV AL,DH     ; Month is in DH
    AAM
    MOV BX,AX
```

Equipo Requerido:

- 2 gb ram
- procesador de 2 nucleos 2.3 ghz
- Dosbox 0.74-3
- Visual Code

CONCLUSIONES

1. A la hora de manipular archivos externos es necesario limpiar los registros utilizados, si en algún momento llegamos a realizar una operación como la división luego de abrir dichos archivos con el programa.
2. La mala utilización de registros puede dar errores como el paro del funcionamiento de un programa.

3. Los lenguajes de alto nivel facilitan mucho todo lo concerniente a la programación permitiendo que nos despreocupemos del uso correcto de los registros.
4. Las macros son una funcionalidad que permiten en gran medida el ahorro de líneas de código siendo estas reutilizables en otros proyectos.
5. No se pueden mover dos variables al mismo tiempo, debe de existir un registro intercesor entre los dos.
6. Si no se cuidan y limpian los registros el programa en alguna parte posiblemente quede estancado