Elian Saúl Estrada Urbina
201806838

MANUAL DE TECNICO

Macros:

1.

```
print macro string

    mov ah, 09h
    lea dx, string
    int 21h

ENDM
```

macro para imprimir caracteres en la pantalla.

2.

```
getNumber macro buffer

    LOCAL NUMBER, SIGN, NEGATIVE, NUMRESULT, OUT_NUMBER

    xor ax, ax
    xor bx, bx
    xor si, si
    xor cx, cx

    mov bx, 0ah

    NUMBER:
        mov cl, buffer[si]

        cmp cl, 2dh             ;Compara si en el buffer en la posición es el signo "-"
        je SIGN

        cmp cl, 30h
        jb NEGATIVE

        cmp cl, 39h
        ja NEGATIVE

        sub cl, 48
        mul bx                  ;Multiplica lo que hay en ax por 10

        add ax, cx              ;Suma lo que hay en ax con cx para obtener el número final

        inc si
        jmp NUMBER

    SIGN:
        mov isNegative, 1
        inc si
        jmp NUMBER

    NEGATIVE:
        cmp isNegative, 1
        je NUMRESULT
        JMP OUT_NUMBER

    NUMRESULT:
        neg ax
        mov isNegative, 0

    OUT_NUMBER:

ENDM
```

Macro para pasar una cadena de caracteres a números hexadecimales

Elian Saúl Estrada Urbina
201806838

3.

```
getBuffer macro buffer

    LOCAL NUMBER, SIGN, NEGATIVE, FILL_BUFFR , OUT_BUFFER

    xor bx, bx
    xor si, si
    xor cx, cx
    xor dx, dx

    mov bx, 0ah

    NUMBER:

        test ax, ax
        js NEGATIVE

        div bx                  ;ax -> resultado; dx -> residuo; ax = 25, dx = 4, ax= 2, dx = 5  ax = 0; dx = 2

        add dx, 48              ; 4 + 48 = 52 -> 4; 5 +48 = 53 -> 5 2 + 48 50 -> 2
        inc cx
        push dx

        cmp ax, 00h
        je FILL_BUFFR

        xor dx, dx
        jmp NUMBER

    FILL_BUFFR:
        pop ax
        mov buffer[si], al
        inc si
        loop FILL_BUFFR

        jmp OUT_BUFFER

    SIGN:
        mov buffer[si], 2dh
        inc si
        jmp NUMBER

    NEGATIVE:
        neg ax
        jmp SIGN

    OUT_BUFFER:
        mov buffer[si], 24h


ENDM
```

Sirve para convertir los números en hexa a cadena de caracteres ascii

Elian Saúl Estrada Urbina
201806838

4.

```
getInput macro buffer, buffer2, ind

    LOCAL CHARACTER, OUT_INPUT

    xor si, si
    xor di, di


    CHARACTER:

        inputOptions flagTrue

        cmp al, 0dh
        je OUT_INPUT

        mov di, ind

        mov buffer[si], al
        mov buffer2[di], al
        inc si
        inc ind
        jmp CHARACTER

    OUT_INPUT:
        mov buffer[si], 24h

ENDM
```

Sirva para almacenar las entradas de los números en un buffer con caracteres ascii

Elian Saúl Estrada Urbina
201806838

5.

```
inputOptions macro option

    LOCAL SHOW_ENTER
    LOCAL HIDDE_ENTER
    LOCAL OUT_MACRO

    cmp option, 31h
    je SHOW_ENTER
    jne HIDDE_ENTER

    SHOW_ENTER:
        mov ah, 01h
        int 21h

        jmp OUT_MACRO

    HIDDE_ENTER:
        mov ah, 08h
        int 21h

    OUT_MACRO:

ENDM
```

Sirve para que el programa espere la entrada de teclado, si la bandera es verdadera entonces se muestra el carácter tecleado, si no no se muestra.

Elian Saúl Estrada Urbina
201806838

6.

```
copy macro bufferO, bufferD

    Local COPYB, EXIT

    xor si, si
    ;xor cx, cx

    COPYB:

        cmp bufferO[si], 24h
        je EXIT

        mov cl, bufferO[si]
        mov bufferD[si], cl
        inc si

        jmp COPYB


    EXIT:
    mov bufferD[si], 00h

ENDM
```

Sirve para copiar un buffer en otro buffer.