



I. TEMA: PROCESAMIENTO DE ARREGLOS EN ENSAMBLADOR

II. OBJETIVOS DE LA PRACTICA

Al finalizar la presente práctica, el estudiante:

1. Explica las particularidades del procesamiento de arreglos en los procesadores con arquitectura x86
2. Implementa aplicaciones para el procesamiento de arreglos utilizando el lenguaje de programación ensamblador para computadores con arquitectura x86

III. TRABAJO PREPARATORIO.

Para obtener mejores resultados, es necesario que el estudiante:

1. Conozca los conceptos básicos del modelo de memoria de los computadores x86
2. Conozca los modos de direccionamiento para el acceso a elementos de arreglos en procesadores con arquitectura x86.

IV. MATERIALES.

1. Sistema operativo Linux
2. Compilador NASM
3. Librería io.mac

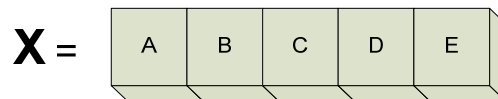


V. MARCO TEORICO

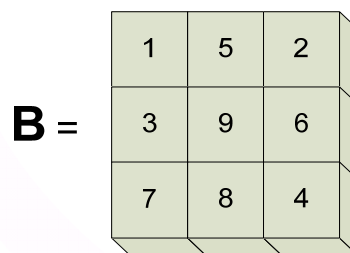
ARREGLOS

Un arreglo es una estructura de datos homogéneos; es decir de datos del mismo tipo, sean estos enteros, caracteres, reales, etc.

Los siguientes son ejemplos de arreglos:



X es un arreglo de 5 elementos en el que el tercer elemento es la letra C. X es, además, un arreglo unidimensional.



B es un arreglo bidimensional en el que un elemento Q, cualquiera, ocupa una posición que puede determinarse mediante los índices [i, j] (i – fila, j – columna)

El acceso a los elementos de un arreglo se realiza no en forma directa sino a través de su posición en el arreglo (la cual se indica mediante un índice), así por ejemplo, en los casos anteriores, para acceder al elemento D en el arreglo X, haremos referencia a este de la siguiente manera: X[3] (en donde 3 es el índice), y para acceder al elemento 6 en el arreglo B haremos referencia a este mediante B[1, 2].

Los arreglos son especialmente adecuados en el caso de que se desee almacenar información de la misma naturaleza, sobre los cuales se deberá realizar las mismas operaciones de procesamiento de datos



VI. TRABAJO DE LABORATORIO.

1. Escriba un programa que defina un vector de N elementos enteros de 32 bits y permita al usuario ingresar los datos correspondientes. Finalmente, el programa debe mostrar los elementos del vector.

Solución

```
;Nombre      :   cargarVector.asm
;Proposito   :   carga enteros en un arreglo unidimensional
;Autor       :   Edwin Carrasco (basado en [2])
;FCreacion   :   28/07/2008
;FModific.   :   08/10/2019
;compilar    :   nasm -f elf cargarVector.asm
;            :   ld -m elf_i386 -s -o cargarVector cargarVector.o
;            :   io.o
;            :   ./cargarVector

%include "io.mac"

MAX_SIZE EQU 5

section .data
    mensaje      db "ESTE PROGRAMA PERMITE INGRESAR HASTA 05
                    ENTEROS A UN VECTOR",10,0
    entrada      db "Ingrese 5 valores diferentes de cero ",13
                    db "(si ingresa cero, termina la lectura de
                    datos):",10, 0
    salida       db "Los elementos del vector son: ",10,0
    imprimirTab  db "",9,0
    invitacion   db "Ingrese el elemento ", 0
    pedirDatos   db " del vector: ", 0
    arregloVacio db "El vector esta vacio. ",0
    pregunta     db "Terminar? (S/N): ",0

section .bss
    arreglo resd MAX_SIZE

section .text
    global _start

_start:
    PutStr mensaje

leer_datos:
    PutStr entrada      ; solicitar datos
    xor esi,esi         ; ESI = 0 (ESI se utiliza como indice)
    mov ecx,MAX_SIZE

leer_siguiete:
    PutStr invitacion
    PutLInt esi
    PutStr pedirDatos
    GetLInt eax
    mov [arreglo+esi*4],eax
```



```
inc esi                ; incrementar indice de arreglo
cmp eax,0              ; se ingreso un cero?
loopne leer_siguiente ; iterar no mas de MAX_SIZE veces

exit_loop:
; si la entrada termina con un cero,
; decrementar esi para mantener el tamaño del arreglo
jnz omitir
dec esi

omitir:
mov ecx, esi           ; esi almacena el tamaño del arreglo
jecxz arreglo_Vacio   ; si ecx = 0, el arreglo esta vacio
xor esi,esi            ; inicializar indice a cero
PutStr salida

mostrar_siguiente:
PutLInt [arreglo+ESI*4]
PutStr imprimirTab
inc esi
loop mostrar_siguiente
jmp short preguntar

arreglo_Vacio:
PutStr arregloVacio   ; mostrar mensaje de arreglo vacio
nwn

preguntar:
nwn
PutStr pregunta       ; preguntar al usuario si desea terminar
GetCh AL
cmp AL,'S'            ; si la respuesta no es 'S'
jne leer_datos        ; repetir el bucle

terminado:
;volver al sistema
mov eax,1
int 80h
```

PRUEBA DE FUNCIONAMIENTO:

```
root@aar:/home/ecp/codigo/AC# nasm -f elf cargarVector.asm
root@aar:/home/ecp/codigo/AC# ld -m elf_i386 -s -o cargarVector cargarVector.o io.o
root@aar:/home/ecp/codigo/AC# ./cargarVector
ESTE PROGRAMA PERMITE INGRESAR HASTA 05 ENTEROS A UN VECTOR
(si ingresa cero, termina la lectura de datos):
Ingrese el elemento 0 del vector: 8
Ingrese el elemento 1 del vector: 24
Ingrese el elemento 2 del vector: 64
Ingrese el elemento 3 del vector: 43
Ingrese el elemento 4 del vector: 7
Los elementos del vector son:
8      24      64      43      7
Terminar? (S/N): S
root@aar:/home/ecp/codigo/AC#
```



2. Escriba un programa que sume los elementos de un vector de N elementos enteros. El usuario debe definir el tamaño del vector, así como ingresar los elementos del vector en tiempo de ejecución. Finalmente, el programa debe mostrar los elementos del vector y la suma de sus elementos.

Solución

```
;Nombre      : sumaVector.asm
;Proposito   : sumar los elementos de un vector
;Autor       : Edwin Carrasco
;FCreacion   : 04/08/2020
;FModific.   : ---
;Compilar    : nasm -f elf sumaVector.asm
;            : ld -m elf_i386 -s -o sumaVector sumaVector.o io.o
;            : ./sumaVector

%include "io.mac"

section .data
    proposito    db "ESTE PROGRAMA SUMA LOS ELEMENTOS DE UN VECTOR DE ENTEROS",10,0
    tamano       db "Defina el tamaño del vector: ", 0
    entrada      db "A continuación ingrese los elementos del vector:",10, 0
    salida       db "Los elementos del vector son: ",10,0
    imprimirTab  db " ",9,0
    invitacion   db "Ingrese el elemento ", 0
    pedirDatos   db " del vector: ", 0
    resultado    db "La suma de los elementos del vector es: ",0

section .bss
    nroElementos resd1 ; tamaño de arreglo
    vector resd 1000 ; espacio reservado para el vector

section .text
    global _start

_start:
    ;Indicar proposito del programa
    PutStr proposito

    ;Leer tamaño del vector
    PutStr tamano
    GetInt [nroElementos]

    ;Leer elementos del vector
leer_datos:
    PutStr entrada ; solicitar datos
    xor esi,esi ; esi = 0 (esi se utiliza como indice)
    mov ecx,[nroElementos]

leer_siguiete:
    PutStr invitacion
    PutLInt esi
    PutStr pedirDatos
```



```
GetLInt  eax
mov  [vector + esi*4],eax
inc esi      ; incrementar indice de arreglo
loopnz leer_siguiente

;Procesar
xor esi, esi
xor eax, eax
mov ecx, [nroElementos]

procesar_sgte:
add eax, [vector + esi*4]
inc esi
loopnz procesar_sgte

;Mostrar Vector
mov ecx, [nroElementos]
xor esi, esi
PutStr salida

mostrar_siguiente:
PutLInt [vector + esi*4]
PutStr imprimirTab
inc esi
loopnz mostrar_siguiente
nwln

;Mostrar resultados
PutStr resultado
PutLInt  eax
nwln

;Volver al sistema
mov  eax, 1
int  80h
```

PRUEBA DE FUNCIONAMIENTO:

```
ecp ~ > codigo nasm -f elf sumaVector.asm
ecp ~ > codigo ld -m elf_i386 -s -o sumaVector sumaVector.o io.o
ecp ~ > codigo ./sumaVector
ESTE PROGRAMA SUMA LOS ELEMENTOS DE UN VECTOR DE ENTEROS
Defina el tamaño del vector: 6
A continuación ingrese los elementos del vector:
Ingrese el elemento 0 del vector: 1
Ingrese el elemento 1 del vector: 2
Ingrese el elemento 2 del vector: 3
Ingrese el elemento 3 del vector: 4
Ingrese el elemento 4 del vector: 5
Ingrese el elemento 5 del vector: 6
Los elementos del vector son:
1      2      3      4      5      6
La suma de los elementos del vector es: 21
```



VII. PRACTICAS DE LABORATORIO

1. Escriba un programa que determine si un entero E, ingresado por el usuario, se encuentra en un vector de N enteros. N, E y los elementos del vector son ingresados por el usuario en tiempo de ejecución.
2. Escriba un programa que muestre el valor y la posición de la primera ocurrencia del elemento más frecuentemente encontrado en un vector de N enteros. Tanto N como los elementos del vector son definidos por el usuario en tiempo de ejecución.

Por ejemplo, para el vector $V = \{2; 6; 9; 9; 8; 7; 5; 9\}$, la respuesta será:

Valor = 9.

Posición = 2.

En caso de haber más de un elemento con la misma frecuencia, considerar para la respuesta, el que sea mayor.



UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ORGANIZACIÓN Y ARQUITECTURA DEL COMPUTADOR
GUÍA DE LABORATORIO

ECP 8 de 9

VIII. EVALUACION

La evaluación de las actividades realizadas en la presente guía de práctica se hará en función de la siguiente tabla:

ACTIVIDAD	Procedimental	
	Sesión 01	Sesión 02
Resolución del ejercicio propuesto 01	--	08
Resolución del ejercicio propuesto 02	--	12
TOTAL	--	20



IX. BIBLIOGRAFIA

1. Anvin, P. <http://nasm.sourceforge.net/>. Sitio web del compilador NASM.
2. Bartlett, J. “*Programming From The Ground Up*”. Bartlett Publishing, 2003
3. Brey, B. “*Los Microprocesadores Intel. Arquitectura, Programación e Interfaces*”. Prentice Hall 3Ed.
4. Carter, P. “*PC Assembly Language*” 2005
5. Dandamudi. “*Guide To Assembly Language Programming In Linux*”. Springer 2005
6. Hyde, R. “*Art of Assembly Language Programming*”. Nostarch Press 1Ed.
7. Leto J. “*Writing a useful Program With NASM*”. <http://leto.net/writing/nasm.php>.
8. Smith B. E., Johnson M. T. “*Programming the Intel 80386*” Editorial Scott, Foresman And Company, 1987.
9. Toal R. “*NASM Tutorial*”. <https://cs.lmu.edu/~ray/notes/nasmtutorial/>.
10. Unknown. “*Matrix in NASM*”. <https://yazary.blogspot.com/2013/01/matrix-in-nasm.html>.