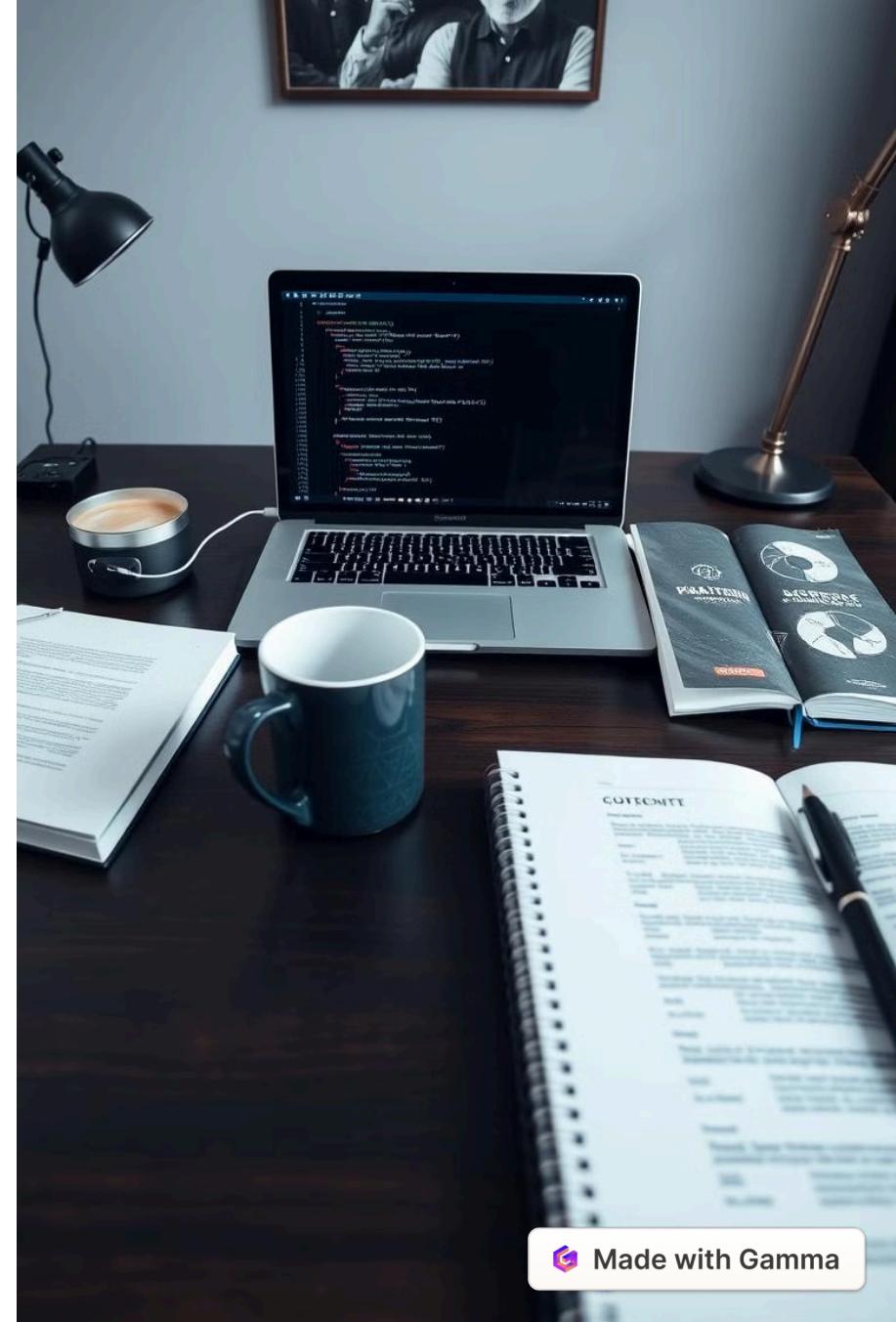


# Manejo de arrays en ensamblador (ASM)

El manejo de arrays en ensamblador (ASM) es una habilidad fundamental para los programadores que trabajan a bajo nivel. Permite la manipulación eficiente de estructuras de datos complejas y optimiza el rendimiento del software.



# Introducción a los arrays en ASM

## 1 Estructura Básica

Un array en ASM es una colección de elementos del mismo tipo almacenados en ubicaciones de memoria contiguas.

## 2 Direccionamiento Indirecto

El acceso a los elementos se realiza mediante cálculos de direcciones basados en el índice del elemento.

## 3 Versatilidad

Los arrays se pueden utilizar para almacenar todo tipo de datos, desde números hasta caracteres.



Made with Gamma

# Declaración de espacios de memoria para arrays

med. fyew

## Reserva de Memoria

Para declarar un array, se reserva un bloque de memoria con un tamaño específico según el tipo de dato.

## Dimensiones

Los arrays pueden ser unidimensionales (1D) o multidimensionales (2D, 3D, etc.), dependiendo de la aplicación.

## Nomenclatura

Es común utilizar nombres descriptivos para los arrays, como "miArray" o "datosUsuario".

## Inicialización

Los elementos del array se pueden inicializar con valores predefinidos o dejar que el programa los asigne.

nond isel9.2

r12014

nond isel6:4

mame isel6:3

rise14:6

destrtype

destrtype

destrtype



Made with Gamma

# Acceso a elementos individuales de un array

## Indexación

El acceso a un elemento específico del array se realiza mediante cálculos de dirección basados en el índice.

## Registros

Normalmente se utilizan registros como EAX, EBX o ECX para almacenar y manipular los índices.

## Optimización

El uso eficiente de registros y cálculos de direcciones es crucial para el rendimiento del código.



.1	1	1	1	1	14	11.3	15	1.13	1913
.1	1	3	9	1	11	1213	15	1.15	1916
.1	5	8	4	4	32	31.3	31	6.15	3630
.18	3	3	9	3	13	13.3	55	5.15	5410
3	1	1	1	7	13	11.3	31	1.17	1413
.5	3	8	4	6	38	55.5	56	8.25	5510
45	4	4	1	4	34	3115	37	6.33	3319
15	3	6	9	7	33	0214	15	9.10	0.10
11	1	6	1	7	39	3617	18	9.15	1520
15	5	1	6	1	19	1513	15	1310	1625
4	1	6	1	1	13	1.11	11	1.12	1.14
11	4	6	1	7	18	1512	14	3.12	1519
.16	8	7	7	8	33	3313	37	3153	1519
.15	5	6	19	8	03	1513	78	9.15	1015
.17	18	11.13	10	15	1529	33	1113	1516	
.14	1	1	12	18	45	1913	35	2135	1520
.36	5	1	8	8	46	1321	81	2038	1517
.16	3	15.18	17	38	2418	38	0033	2929	

# Ventajas del manejo de arrays como cadenas de caracteres

1

## Representación de Texto

Los arrays pueden almacenar y manipular secuencias de caracteres ASCII, lo que facilita el manejo de cadenas de texto.

2

## Operaciones Comunes

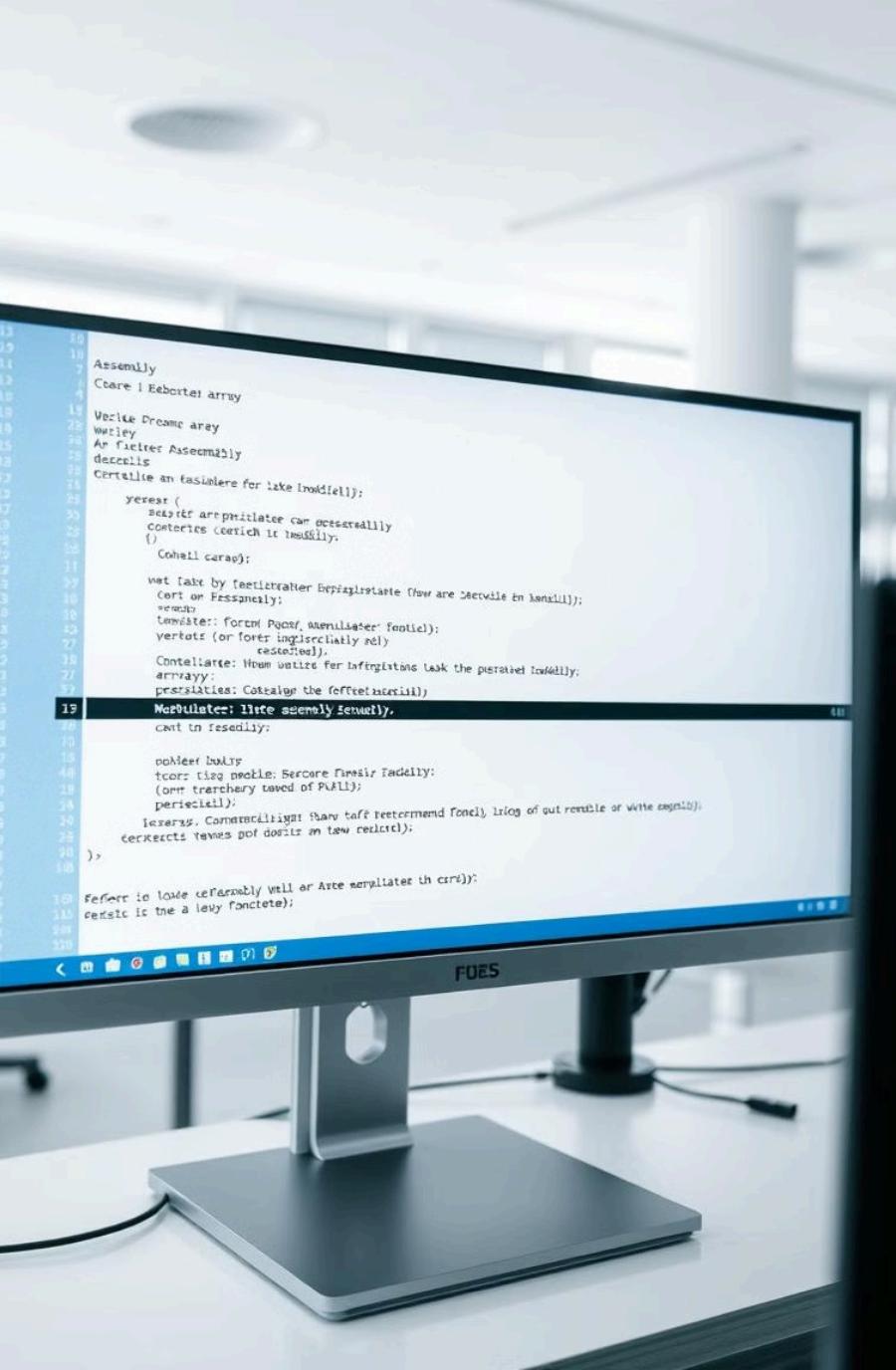
Funciones como la concatenación, búsqueda y reemplazo de caracteres se vuelven sencillas de implementar.

3

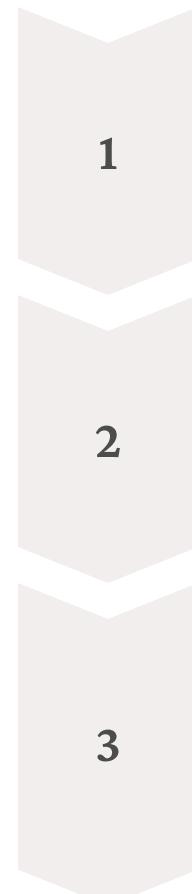
## Eficiencia

El acceso y procesamiento de cadenas de texto en ASM es más eficiente que en otros lenguajes de alto nivel.





# Operaciones comunes con arrays (lectura, escritura, iteración)



## Lectura

Acceder a un elemento específico del array y almacenarlo en un registro para su procesamiento.

## Escritura

Asignar un valor a un elemento del array utilizando cálculos de dirección y el registro apropiado.

## Iteración

Recorrer los elementos del array utilizando saltos condicionales y actualizando los índices de forma programática.

# Uso de saltos condicionales para recorrer arrays



## Incremento de Índice

Incrementar el índice del array para acceder al siguiente elemento.



## Condición de Parada

Verificar si se ha llegado al final del array utilizando saltos condicionales.



## Estructura de Bucle

Implementar un bucle que permita recorrer todos los elementos del array.

```
11 T800. 16
12 T600. lct();
14 T301. 8(9lt=:1c)
13 T802. 8(9ltalbicag(,))
13 T800. 8(9lea(byiin),)
14 T803. 8(9ltaleirassedjay(),c)
14 T800. 8(9(t==ii));
13 T800. 8(9lti=ii:ic)
19 T309. 8(9ltaleirassedjat:(,))
17 T801. 8(9lta(crjin),)
14 T600. 8(9l=>:t))
16 T600. 8(9let:t)
13 T701. 8(9lat(isteri,:))
16 T800. 8(9ltaleirasssesed(eiy),)
15 T600. 8(9let=t:ij)
16 T600. 8(9lca(ciraser:i,(,))
17 T800. 8(9lat:t:in))
18 T800. 8(9l1==t:ij)
18 T600. 8(9l=>:1c)
27 T800. 8(9lat:ct1ii,)
23 T800. 8(9lar:i))
26 T800. 8(9let=ii))
26 T800. 8(9]at:it))
25 T800. 8(9l==:it)
16 T800. 8(9lea(ct:ij)
```

```
18 import re,sys,os,stat,imp,ctc,urllib,script,replicated,tomcat (face display),  
19 tupper:ichiandlsstitc(,300.16 sray;  
19 tonortion.:inplechsationt, cupse_pclicelopptos:lst.(leternttes =vitoplo.logs)  
10 :)  
11 >  
15 rcarohte :stpcrlflectablehlhpgldettsuclectalt,21769,41871-212) =ablil=(pc,deptll_ster181> 201108),  
23 iopeontt,:nwellcancelass-inNo aenigratetb aelc8732 ceray,laterives; retrriirgy)  
29 precbecti((( _seslsalti0l6)= arrvy))  
39 cerodetblr(:  
29 procelistalisittion, sccalomffircg(lrgy)  
29
```

# Ejemplos prácticos de manejo de arrays en ASM

Declaración	Reservar espacio en memoria para un array de enteros.
Inicialización	Asignar valores predefinidos a los elementos del array.
Acceso	Leer y escribir elementos del array utilizando cálculos de dirección.
Iteración	Recorrer el array utilizando saltos condicionales y actualizar los índices.



Memoniy/etrSennery

Betcome Perfiemmery

Assembly Cretfiesetriens

# Consideraciones de desempeño y optimización

## Acceso Eficiente

Minimizar los cálculos de direcciones y el uso de registros para mejorar la velocidad de acceso a los elementos del array.

## Optimización de Bucles

Optimizar la estructura de los bucles que recorren el array, reduciendo las operaciones innecesarias.

## Almacenamiento Adecuado

Utilizar el tipo de dato más apropiado para los elementos del array, evitando desperdicio de memoria.

## Segmentación de Memoria

Distribuir los arrays en secciones de memoria contiguas para aprovechar la localidad de los datos.



# Conclusiones y recomendaciones

1

## Dominio Fundamental

El manejo de arrays en ASM es una habilidad esencial para programadores de bajo nivel.

2

## Versatilidad

Los arrays permiten almacenar y procesar eficientemente todo tipo de datos, incluyendo cadenas de texto.

3

## Optimización

La implementación eficiente de operaciones con arrays es crucial para el rendimiento del software.

4

## Práctica Constante

Desarrollar y practicar el manejo de arrays en ASM es fundamental para dominar la programación de bajo nivel.

