

## *1. Define las características principales de una máquina multinivel.*

Una **máquina multinivel** es un modelo conceptual que representa el funcionamiento de un sistema computacional en diferentes niveles de abstracción. Sus características principales incluyen:

2. **Jerarquía de niveles:** Está compuesta por varios niveles de abstracción que van desde el hardware hasta las aplicaciones de usuario.
3. **Interdependencia entre niveles:** Cada nivel depende del nivel inferior y proporciona servicios al nivel superior.
4. **Uso de interfaces y traductores:** Los niveles interactúan mediante interfaces bien definidas, como ensambladores, compiladores, intérpretes y sistemas operativos.
5. **Modularidad:** Permite que los desarrolladores trabajen en un nivel sin necesidad de conocer todos los detalles de los otros niveles.
6. **Facilidad de desarrollo y mantenimiento:** Al separar las funciones en niveles, se simplifica la creación y mantenimiento de software y hardware.
- 7.

*2. Describe a detalle como es la comunicación entre cada nivel de una máquina multinivel (hardware-microarquitectura-sistema operativo-lenguaje de alto nivel-aplicación). Puedes incluir interfaces, traductores, protocolos, capas, etc.*

Cada nivel se comunica con el siguiente a través de interfaces bien definidas. Aquí te explico cómo ocurre esta interacción en una máquina multinivel típica:

### **1. Nivel de Hardware**

- Compuesto por circuitos electrónicos, procesadores, memoria y dispositivos de entrada/salida.
- Se comunica con la **microarquitectura** a través de señales eléctricas, registros y buses.

### **2. Nivel de Microarquitectura**

- Implementa la ejecución de instrucciones del conjunto de instrucciones (ISA).
- Contiene registros, ALU, unidad de control, cachés y otros elementos que optimizan la ejecución.
- Se comunica con el **nivel de sistema operativo** a través de interrupciones, excepciones y llamadas al sistema.

### **3. Nivel de Sistema Operativo**

- Administra recursos del hardware, como memoria, CPU y dispositivos de E/S.
- Expone una **interfaz de programación de aplicaciones (API)** que permite a los programas comunicarse con el hardware de manera abstracta.
- Utiliza **syscalls** (llamadas al sistema) para permitir que los programas de usuario accedan a servicios del sistema.
- Se comunica con el **lenguaje de alto nivel** a través de bibliotecas estándar y entornos de ejecución.

#### 4. Nivel de Lenguaje de Alto Nivel

- Lenguajes como C, Python, Java, etc.
- Se traduce a código máquina mediante **compiladores, ensambladores e intérpretes**.
- Se comunica con el **nivel de aplicación** proporcionando estructuras de datos y abstracciones.

#### 5. Nivel de Aplicación

- Son los programas que usa el usuario (navegadores, juegos, editores de texto, etc.).
- Se comunican con el **nivel de sistema operativo** usando **llamadas a la API del sistema** (syscalls) para acceder a recursos como archivos, red y memoria.

### 3. Características del lenguaje de bajo nivel.

Un **lenguaje de bajo nivel** está más cerca del hardware y tiene las siguientes características:

- **Alta eficiencia:** Se ejecuta directamente en el procesador sin necesidad de demasiada interpretación.
- **Uso directo de registros y memoria:** Puede acceder a direcciones de memoria y registros de manera explícita.
- **Dependencia de la arquitectura:** Está diseñado para un conjunto específico de instrucciones (ISA) como x86, ARM, etc.
- **Complejidad y dificultad de lectura:** Requiere conocimientos profundos de la arquitectura del hardware.
- **Velocidad de ejecución:** Más rápido que los lenguajes de alto nivel debido a la ausencia de abstracciones adicionales.

### 4. Características del lenguaje de alto nivel.

Un **lenguaje de alto nivel** está diseñado para facilitar la programación y tiene las siguientes características:

- **Abstracción del hardware:** No requiere conocimiento de la arquitectura subyacente.
- **Portabilidad:** El código fuente puede ejecutarse en diferentes sistemas con poca o ninguna modificación.
- **Legibilidad y facilidad de uso:** Su sintaxis es más cercana al lenguaje humano.
- **Gestión automática de memoria:** Muchos lenguajes manejan la memoria automáticamente (ej., garbage collection en Java o Python).
- **Uso de bibliotecas y frameworks:** Facilita la programación con módulos predefinidos.

5. Investiga el funcionamiento de las siguientes salidas del sistema (Syscalls) en arquitectura x86:

a. *Sys\_exit(1)*

- Código de syscall: 1
- Finaliza un proceso y devuelve un código de salida.
- Argumento: `ebx` contiene el código de salida.

b. *Sys\_write(4)*

- Código de syscall: 4
- Escribe datos en un archivo o salida estándar.

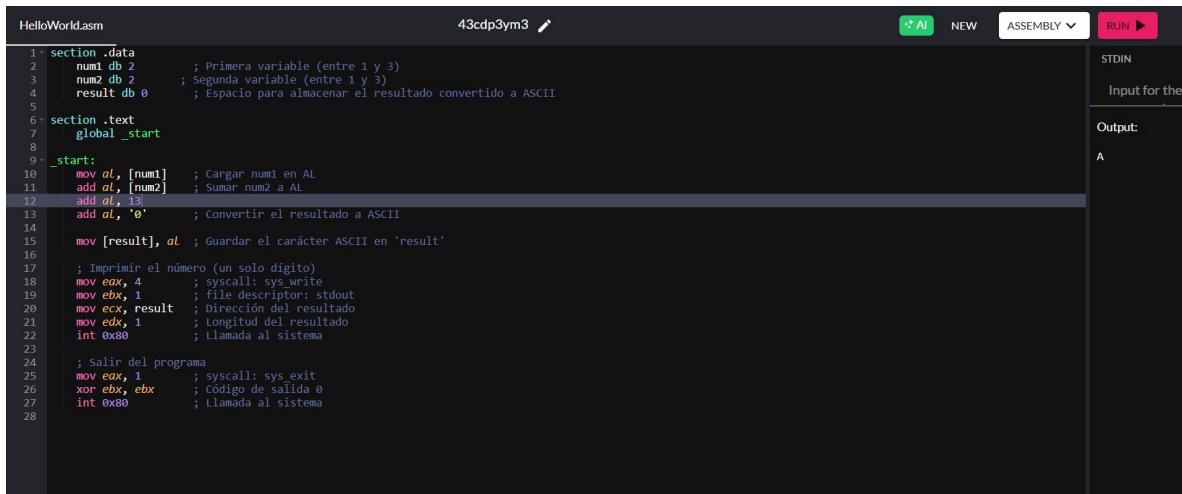
c. *Sys\_read(3)*

- Código de syscall: 3
- Lee datos desde un archivo o entrada estándar.

6. De acuerdo al código ensamblador anexo (también lo puedes encontrar aquí: <https://onecompiler.com/assembly/43ccesug6>), realiza lo siguiente:

a. Modifica el código para que imprima los siguientes caracteres utilizando solo sumas:

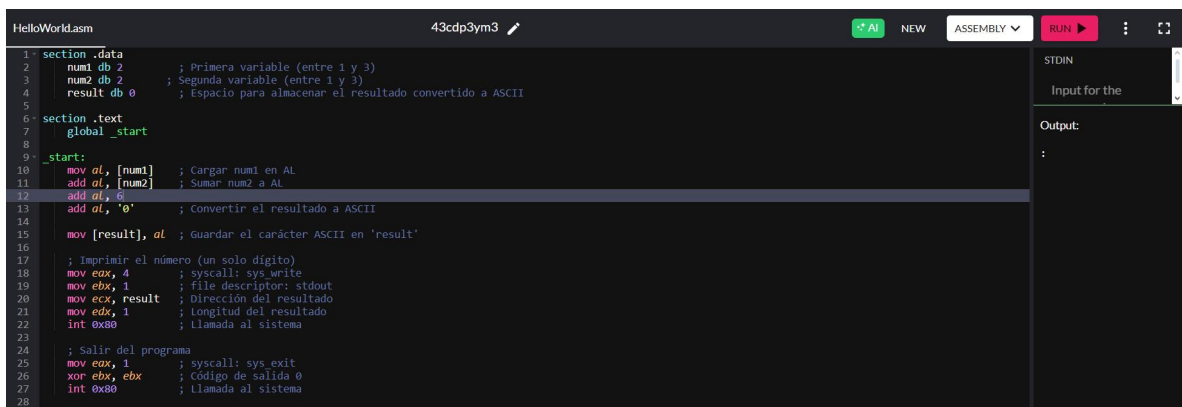
i. A



```
1 section .data
2     num1 db 2      ; Primera variable (entre 1 y 3)
3     num2 db 2      ; Segunda variable (entre 1 y 3)
4     result db 0     ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7     global _start
8
9 _start:
10    mov al, [num1]   ; Cargar num1 en AL
11    add al, [num2]   ; Sumar num2 a AL
12    add al, 13       ; Convertir el resultado a ASCII
13    add al, '0'      ; Convertir el resultado a ASCII
14
15    mov [result], al ; Guardar el carácter ASCII en 'result'
16
17    ; Imprimir el número (un solo dígito)
18    mov eax, 4       ; syscall: sys_write
19    mov ebx, 1       ; file descriptor: stdout
20    mov ecx, result   ; Dirección del resultado
21    mov edx, 1       ; longitud del resultado
22    int 0x80         ; Llamada al sistema
23
24    ; Salir del programa
25    mov eax, 1       ; syscall: sys_exit
26    xor ebx, ebx     ; código de salida 0
27    int 0x80         ; Llamada al sistema
28
```

Output: A

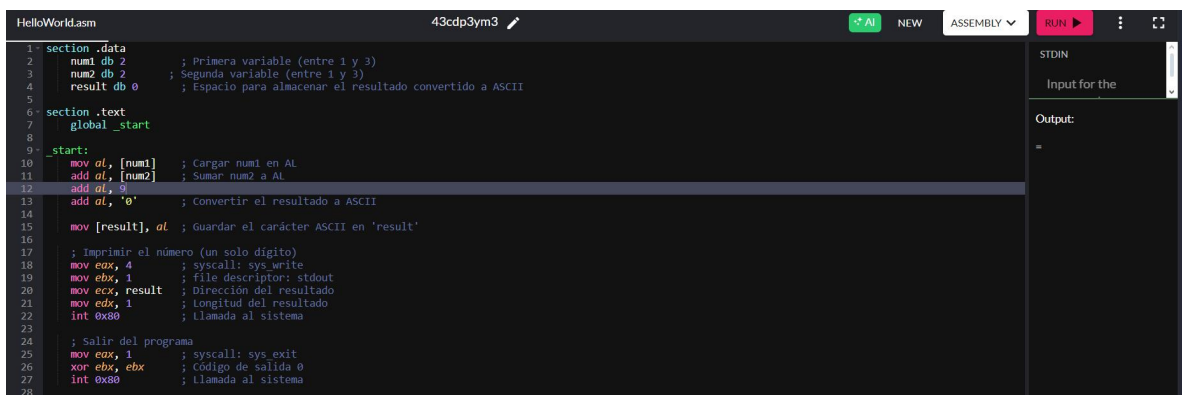
ii. :



```
1 section .data
2     num1 db 2      ; Primera variable (entre 1 y 3)
3     num2 db 2      ; Segunda variable (entre 1 y 3)
4     result db 0     ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7     global _start
8
9 _start:
10    mov al, [num1]   ; Cargar num1 en AL
11    add al, [num2]   ; Sumar num2 a AL
12    add al, 6        ; Convertir el resultado a ASCII
13    add al, '0'      ; Convertir el resultado a ASCII
14
15    mov [result], al ; Guardar el carácter ASCII en 'result'
16
17    ; Imprimir el número (un solo dígito)
18    mov eax, 4       ; syscall: sys_write
19    mov ebx, 1       ; file descriptor: stdout
20    mov ecx, result   ; Dirección del resultado
21    mov edx, 1       ; longitud del resultado
22    int 0x80         ; Llamada al sistema
23
24    ; Salir del programa
25    mov eax, 1       ; syscall: sys_exit
26    xor ebx, ebx     ; código de salida 0
27    int 0x80         ; Llamada al sistema
28
```

Output: :

iii. =



```
1 section .data
2     num1 db 2      ; Primera variable (entre 1 y 3)
3     num2 db 2      ; Segunda variable (entre 1 y 3)
4     result db 0     ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7     global _start
8
9 _start:
10    mov al, [num1]   ; Cargar num1 en AL
11    add al, [num2]   ; Sumar num2 a AL
12    add al, 9        ; Convertir el resultado a ASCII
13    add al, '0'      ; Convertir el resultado a ASCII
14
15    mov [result], al ; Guardar el carácter ASCII en 'result'
16
17    ; Imprimir el número (un solo dígito)
18    mov eax, 4       ; syscall: sys_write
19    mov ebx, 1       ; file descriptor: stdout
20    mov ecx, result   ; Dirección del resultado
21    mov edx, 1       ; longitud del resultado
22    int 0x80         ; Llamada al sistema
23
24    ; Salir del programa
25    mov eax, 1       ; syscall: sys_exit
26    xor ebx, ebx     ; código de salida 0
27    int 0x80         ; Llamada al sistema
28
```

Output: =

iv. ?

```
HelloWorld.asm 43cdp3ym3
1 section .data
2     num1 db 2 ; Primera variable (entre 1 y 3)
3     num2 db 2 ; Segunda variable (entre 1 y 3)
4     result db 0 ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7     global _start
8
9 _start:
10    mov al, [num1] ; Cargar num1 en AL
11    add al, [num2] ; Sumar num2 a AL
12    add al, 11
13    add al, '0' ; Convertir el resultado a ASCII
14
15    mov [result], al ; Guardar el carácter ASCII en 'result'
16
17    ; Imprimir el número (un solo dígito)
18    mov eax, 4 ; syscall: sys_write
19    mov ebx, 1 ; file descriptor: stdout
20    mov ecx, result ; Dirección del resultado
21    mov edx, 1 ; longitud del resultado
22    int 0x80 ; Llamada al sistema
23
24    ; Salir del programa
25    mov eax, 1 ; syscall: sys_exit
26    xor ebx, ebx ; Código de salida 0
27    int 0x80 ; Llamada al sistema
28
```

V.

```
HelloWorld.asm 43cdp3ym3
1 section .data
2     num1 db 2 ; Primera variable (entre 1 y 3)
3     num2 db 2 ; Segunda variable (entre 1 y 3)
4     result db 0 ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7     global _start
8
9 _start:
10    mov al, [num1] ; Cargar num1 en AL
11    add al, [num2] ; Sumar num2 a AL
12    add al, 43
13    add al, '0' ; Convertir el resultado a ASCII
14
15    mov [result], al ; Guardar el carácter ASCII en 'result'
16
17    ; Imprimir el número (un solo dígito)
18    mov eax, 4 ; syscall: sys_write
19    mov ebx, 1 ; file descriptor: stdout
20    mov ecx, result ; Dirección del resultado
21    mov edx, 1 ; longitud del resultado
22    int 0x80 ; Llamada al sistema
23
24    ; Salir del programa
25    mov eax, 1 ; syscall: sys_exit
26    xor ebx, ebx ; Código de salida 0
27    int 0x80 ; Llamada al sistema
28
```

b. Ahora modificarlo para imprima los siguientes caracteres utilizando al menos una resta dentro del código:

i. B

HelloWorld.asm43cdp3ym3NEWASSEMBLYRUN

```
1 section .data
2 num1 db 2 ; Primera variable (entre 1 y 3)
3 num2 db 2 ; Segunda variable (entre 1 y 3)
4 result db 0 ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7 global _start
8
9 _start:
10 mov al, [num1] ; Cargar num1 en AL
11 add al, [num2] ; Sumar num2 a AL
12 sub al, 20
13 add al, 34
14 add al, '0' ; Convertir el resultado a ASCII
15
16 mov [result], al ; Guardar el carácter ASCII en 'result'
17
18 ; Imprimir el número (un solo dígito)
19 mov eax, 4 ; syscall: sys_write
20 mov ebx, 1 ; file descriptor: stdout
21 mov ecx, result ; Dirección del resultado
22 mov edx, 1 ; Longitud del resultado
23 int 0x80 ; Llamada al sistema
24
25 ; Salir del programa
26 mov eax, 1 ; syscall: sys_exit
27 xor ebx, ebx ; código de salida 0
28 int 0x80 ; Llamada al sistema
29
```

STDIN

Input for the

Output:

B

ii.

X

HelloWorld.asm43cdp3ym3NEWASSEMBLYRUN

```
1 section .data
2 num1 db 2 ; Primera variable (entre 1 y 3)
3 num2 db 2 ; Segunda variable (entre 1 y 3)
4 result db 0 ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7 global _start
8
9 _start:
10 mov al, [num1] ; Cargar num1 en AL
11 add al, [num2] ; Sumar num2 a AL
12 sub al, 2
13 add al, 38
14 add al, '0' ; Convertir el resultado a ASCII
15
16 mov [result], al ; Guardar el carácter ASCII en 'result'
17
18 ; Imprimir el número (un solo dígito)
19 mov eax, 4 ; syscall: sys_write
20 mov ebx, 1 ; file descriptor: stdout
21 mov ecx, result ; Dirección del resultado
22 mov edx, 1 ; Longitud del resultado
23 int 0x80 ; Llamada al sistema
24
25 ; Salir del programa
26 mov eax, 1 ; syscall: sys_exit
27 xor ebx, ebx ; código de salida 0
28 int 0x80 ; Llamada al sistema
29
```

STDIN

Input for the

Output:

X

iii.

+

HelloWorld.asm43cdp3ym3NEWASSEMBLYRUN

```
1 section .data
2 num1 db 2 ; Primera variable (entre 1 y 3)
3 num2 db 2 ; Segunda variable (entre 1 y 3)
4 result db 0 ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7 global _start
8
9 _start:
10 mov al, [num1] ; Cargar num1 en AL
11 add al, [num2] ; Sumar num2 a AL
12 sub al, 40
13 add al, 38
14 add al, '0' ; Convertir el resultado a ASCII
15
16 mov [result], al ; Guardar el carácter ASCII en 'result'
17
18 ; Imprimir el número (un solo dígito)
19 mov eax, 4 ; syscall: sys_write
20 mov ebx, 1 ; file descriptor: stdout
21 mov ecx, result ; Dirección del resultado
22 mov edx, 1 ; Longitud del resultado
23 int 0x80 ; Llamada al sistema
24
25 ; Salir del programa
26 mov eax, 1 ; syscall: sys_exit
27 xor ebx, ebx ; código de salida 0
28 int 0x80 ; Llamada al sistema
29
```

STDIN

Input for the

Output:

+

iv.

6

HelloWorld.asm43cdp3ym3NEWASSEMBLYRUN

```
1 section .data
2     num1 db 2 ; Primera variable (entre 1 y 3)
3     num2 db 2 ; Segunda variable (entre 1 y 3)
4     result db 0 ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7     global _start
8
9 _start:
10    mov al, [num1] ; Cargar num1 en AL
11    add al, [num2] ; Sumar num2 a AL
12    sub al, 40
13    add al, 27
14    add al, '0' ; Convertir el resultado a ASCII
15
16    mov [result], al ; Guardar el carácter ASCII en 'result'
17
18    ; Imprimir el número (un solo dígito)
19    mov eax, 4 ; syscall: sys_write
20    mov ebx, 1 ; file descriptor: stdout
21    mov ecx, result ; Dirección del resultado
22    mov edx, 1 ; longitud del resultado
23    int 0x80 ; llamada al sistema
24
25    ; Salir del programa
26    mov eax, 1 ; syscall: sys_exit
27    xor ebx, ebx ; código de salida 0
28    int 0x80 ; llamada al sistema
```

STDIN

Input for the

Output:

v.

{

HelloWorld.asm43cdp3ym3NEWASSEMBLYRUN

```
1 section .data
2     num1 db 2 ; Primera variable (entre 1 y 3)
3     num2 db 2 ; Segunda variable (entre 1 y 3)
4     result db 0 ; Espacio para almacenar el resultado convertido a ASCII
5
6 section .text
7     global _start
8
9 _start:
10    mov al, [num1] ; Cargar num1 en AL
11    add al, [num2] ; Sumar num2 a AL
12    sub al, 40
13    add al, 111
14    add al, '0' ; Convertir el resultado a ASCII
15
16    mov [result], al ; Guardar el carácter ASCII en 'result'
17
18    ; Imprimir el número (un solo dígito)
19    mov eax, 4 ; syscall: sys_write
20    mov ebx, 1 ; file descriptor: stdout
21    mov ecx, result ; Dirección del resultado
22    mov edx, 1 ; longitud del resultado
23    int 0x80 ; llamada al sistema
24
25    ; Salir del programa
26    mov eax, 1 ; syscall: sys_exit
27    xor ebx, ebx ; código de salida 0
28    int 0x80 ; llamada al sistema
29
```

STDIN

Input for the

Output:

{