UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño.

Ingeniería en Software y Tecnologías Emergentes

Organización de Computadoras

Taller 5

Alejandro Palomares Ceseña

1. **Características principales de una máquina multinivel**

Una máquina multinivel es un modelo conceptual que describe cómo se organiza una computadora en niveles jerárquicos de abstracción, donde cada nivel oculta los detalles del nivel inferior y ofrece una interfaz más sencilla al nivel superior.

Características principales:

- Jerarquía de niveles: Cada nivel depende del inferior y sirve de base al superior (hardware → microarquitectura → sistema operativo → lenguaje de alto nivel → aplicación).
- Abstracción: Cada nivel traduce operaciones complejas en instrucciones más simples del nivel inferior.
- Interfaz definida: Cada nivel se comunica mediante interfaces o traductores bien definidos (por ejemplo, compiladores, drivers, API).
- Eficiencia y portabilidad: Permite ejecutar programas de alto nivel sin preocuparse por el hardware físico.
- Modularidad: Facilita el diseño, mantenimiento y evolución de los sistemas informáticos.

2. **Comunicación entre cada nivel de una máquina multinivel**

Nivel 0: Hardware físico
Es el nivel más bajo (procesador, memoria, dispositivos de E/S).
Ejecuta instrucciones binarias (lenguaje máquina).

Nivel 1: Microarquitectura
Interpreta las instrucciones del hardware y controla el flujo de datos.
Usa microinstrucciones para ejecutar las instrucciones máquina.
Comunicación: el hardware ejecuta operaciones definidas por la unidad de control (por microcódigo o lógica cableada).

Nivel 2: Sistema operativo
Gestiona el hardware y provee servicios básicos: manejo de memoria, archivos, procesos, dispositivos, etc.
Se comunica con la microarquitectura mediante llamadas al sistema (syscalls) y drivers.
Traduce peticiones de software (como “abrir archivo”) en operaciones de bajo nivel.

Nivel 3: Lenguaje de alto nivel
Ejemplo: C, Java, Python.
El código fuente se traduce al lenguaje máquina por medio de un compilador, intérprete o ensamblador.
Comunicación: el compilador traduce instrucciones de alto nivel en instrucciones que el sistema operativo y hardware puedan ejecutar.

Nivel 4: Aplicación
Programas que usa el usuario (navegador, editor, juego, etc.).
Se comunican con el sistema operativo mediante APIs o bibliotecas.

Estas APIs internamente hacen llamadas al sistema (syscalls) que terminan ejecutándose en el hardware.

Resumen visual de comunicación:

Aplicación

↓ (API / Librerías)

Lenguaje de alto nivel

↓ (Compilador / Intérprete)

Sistema Operativo

↓ (Drivers / Syscalls)

Microarquitectura

↓ (Microinstrucciones)

Hardware

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

- Cercano al hardware (poca abstracción).
- Difícil de leer y mantener.
- Control directo de memoria y registros.
- Alta eficiencia y velocidad.
- Usado para firmware, drivers y sistemas operativos.
- Ejemplo: lenguaje ensamblador (Assembly).

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

- Alta abstracción: más cercano al lenguaje humano.
- Fácil de leer, escribir y mantener.
- Independiente del hardware.
- Usa estructuras lógicas (if, while, funciones, objetos, etc.).
- Requiere traductor (compilador o intérprete).
- Ejemplo: Python, Java, C++, C#, etc.

5. **Funcionamiento de las salidas del sistema (Syscalls) en arquitectura x86**

Las syscalls (system calls) son servicios que ofrece el sistema operativo al usuario o programa para acceder a funciones del kernel (como leer, escribir o terminar procesos).

a) sys_exit(1)

- Número de syscall: 1
- Función: Termina el proceso actual y devuelve un código de salida al sistema operativo.
- Parámetro: Código de salida (por convención, 0 = éxito, ≠0 = error).
  Ejemplo (Assembly Linux x86):

```
mov eax, 1      ; código de syscall (sys_exit)
mov ebx, 0      ; código de salida
int 0x80        ; interrupción para llamar al kernel
```

El kernel libera los recursos del proceso y actualiza la tabla de procesos.

b) sys_write(4)

- Número de syscall: 4
- Función: Escribe datos en un archivo o dispositivo (por ejemplo, pantalla o archivo).
- Parámetros:
  - ebx: descriptor de archivo (1 = stdout, 2 = stderr, etc.)
  - ecx: dirección del buffer (texto o datos a escribir)
  - edx: número de bytes a escribir
    Ejemplo:

```
mov eax, 4      ; sys_write
mov ebx, 1      ; stdout
mov ecx, msg    ; dirección del mensaje
mov edx, len    ; longitud
int 0x80
```

Escribe el contenido del buffer en el destino especificado.

c) sys_read(3)

- Número de syscall: 3
- Función: Lee datos de un archivo o entrada (por ejemplo, teclado).
- Parámetros:
  - ebx: descriptor de archivo (0 = stdin)
  - ecx: dirección del buffer donde se almacenará lo leído
  - edx: número máximo de bytes a leer
    Ejemplo:

mov eax, 3 ; sys_read

mov ebx, 0 ; stdin

mov ecx, buffer ; destino

mov edx, 100 ; máximo 100 bytes

int 0x80

El kernel coloca los datos leídos del dispositivo en el buffer indicado.

1. Modifica el código para que imprima los

siguientes caracteres utilizando solo sumas:

- A


```
section .data
    num1 db 2          ; Primera variable (entre 1 y 3)
    num2 db 15         ; Segunda variable (entre 1 y 3)
    result db 0        ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    add al, [num2]     ; Sumar num2 a AL
    add al, '0'        ; Convertir el resultado a ASCII

    mov [result], al   ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4         ; syscall: sys_write
    mov ebx, 1         ; file descriptor: stdout
    mov ecx, result    ; Dirección del resultado
    mov edx, 1         ; Longitud del resultado
    int 0x80           ; Llamada al sistema

    ; Salir del programa
    mov eax, 1         ; syscall: sys_exit
    xor ebx, ebx       ; Código de salida 0
    int 0x80           ; Llamada al sistema
```

- :


```
section .data
    num1 db 2          ; Primera variable (entre 1 y 3)
    num2 db 8          ; Segunda variable (entre 1 y 3)
    result db 0        ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    add al, [num2]     ; Sumar num2 a AL
    add al, '0'        ; Convertir el resultado a ASCII

    mov [result], al   ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4         ; syscall: sys_write
    mov ebx, 1         ; file descriptor: stdout
    mov ecx, result    ; Dirección del resultado
    mov edx, 1         ; Longitud del resultado
    int 0x80           ; Llamada al sistema

    ; Salir del programa
    mov eax, 1         ; syscall: sys_exit
    xor ebx, ebx       ; Código de salida 0
    int 0x80           ; Llamada al sistema
```

- =


```
section .data
    num1 db 2          ; Primera variable (entre 1 y 3)
    num2 db 11         ; Segunda variable (entre 1 y 3)
    result db 0        ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    add al, [num2]     ; Sumar num2 a AL
    add al, '0'        ; Convertir el resultado a ASCII
```

- ?


```
section .data
    num1 db 2          ; Primera variable (entre 1 y 3)
    num2 db 13         ; Segunda variable (entre 1 y 3)
    result db 0        ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    add al, [num2]     ; Sumar num2 a AL
    add al, '0'        ; Convertir el resultado a ASCII

    mov [result], al   ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4         ; syscall: sys_write
    mov ebx, 1         ; file descriptor: stdout
    mov ecx, result    ; Dirección del resultado
    mov edx, 1         ; Longitud del resultado
    int 0x80           ; Llamada al sistema

    ; Salir del programa
    mov eax, 1         ; syscall: sys_exit
    xor ebx, ebx       ; Código de salida 0
    int 0x80           ; Llamada al sistema
```

- _


```
section .data
    num1 db 2          ; Primera variable (entre 1 y 3)
    num2 db 45         ; Segunda variable (entre 1 y 3)
    result db 0        ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    add al, [num2]     ; Sumar num2 a AL
    add al, '0'        ; Convertir el resultado a ASCII

    mov [result], al   ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4         ; syscall: sys_write
    mov ebx, 1         ; file descriptor: stdout
    mov ecx, result    ; Dirección del resultado
    mov edx, 1         ; Longitud del resultado
    int 0x80           ; Llamada al sistema

    ; Salir del programa
    mov eax, 1         ; syscall: sys_exit
    xor ebx, ebx       ; Código de salida 0
    int 0x80           ; Llamada al sistema
```

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

- B

HelloWorld.asm 43xt2kpkz AI NEW ASSEMBLY RUN

```asm
section .data
    num1 db 19          ; Primera variable (entre 1 y 3)
    num2 db 1       ; Segunda variable (entre 1 y 3)
    result db 0         ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]    ; Cargar num1 en AL
    sub al, [num2]    ; Resta num2 a AL
    add al, '0'       ; Convertir el resultado a ASCII

    mov [result], al  ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4        ; syscall: sys_write
    mov ebx, 1        ; file descriptor: stdout
    mov ecx, result   ; Dirección del resultado
    mov edx, 1        ; Longitud del resultado
    int 0x80          ; Llamada al sistema

    ; Salir del programa
    mov eax, 1        ; syscall: sys_exit
    xor ebx, ebx      ; Código de salida 0
    int 0x80          ; Llamada al sistema
```

STDIN

Input for the program (Optional)

Output:

B

- X

HelloWorld.asm 43xt2kpkz AI NEW ASSEMBLY RUN

```asm
section .data
    num1 db 73          ; Primera variable (entre 1 y 3)
    num2 db 1       ; Segunda variable (entre 1 y 3)
    result db 0         ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]    ; Cargar num1 en AL
    sub al, [num2]    ; Resta num2 a AL
    add al, '0'       ; Convertir el resultado a ASCII

    mov [result], al  ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4        ; syscall: sys_write
    mov ebx, 1        ; file descriptor: stdout
    mov ecx, result   ; Dirección del resultado
    mov edx, 1        ; Longitud del resultado
    int 0x80          ; Llamada al sistema

    ; Salir del programa
    mov eax, 1        ; syscall: sys_exit
    xor ebx, ebx      ; Código de salida 0
    int 0x80          ; Llamada al sistema
```

STDIN

Input for the program (Optional)

Output:

x

- +

HelloWorld.asm 43xt2kpkz AI NEW ASSEMBLY RUN

```asm
section .data
    num1 db 1          ; Primera variable (entre 1 y 3)
    num2 db 6       ; Segunda variable (entre 1 y 3)
    result db 0         ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]    ; Cargar num1 en AL
    sub al, [num2]    ; Resta num2 a AL
    add al, '0'       ; Convertir el resultado a ASCII

    mov [result], al  ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4        ; syscall: sys_write
    mov ebx, 1        ; file descriptor: stdout
    mov ecx, result   ; Dirección del resultado
    mov edx, 1        ; Longitud del resultado
    int 0x80          ; Llamada al sistema

    ; Salir del programa
    mov eax, 1        ; syscall: sys_exit
    xor ebx, ebx      ; Código de salida 0
    int 0x80          ; Llamada al sistema
```

STDIN

Input for the program (Optional)

Output:

+

- ‘

HelloWorld.asm 43zadz94g AI NEW ASSEMBLY RUN

```
section .data
    num1 db 3          ; Primera variable (entre 1 y 3)
    num2 db 12         ; Segunda variable (entre 1 y 3)
    result db 0        ; Espacio para almacenar el resultado convertido a ASCII

section .text
    global _start

_start:
    mov al, [num1]     ; Cargar num1 en AL
    sub al, [num2]     ; Restar num2 a AL
    add al, '0'        ; Convertir el resultado a ASCII

    mov [result], al  ; Guardar el carácter ASCII en 'result'

    ; Imprimir el número (un solo dígito)
    mov eax, 4         ; syscall: sys_write
    mov ebx, 1         ; file descriptor: stdout
    mov ecx, result    ; Dirección del resultado
    mov edx, 1         ; Longitud del resultado
    int 0x80           ; Llamada al sistema

    ; Salir del programa
    mov eax, 1         ; syscall: sys_exit
    xor ebx, ebx       ; Código de salida 0
    int 0x80           ; Llamada al sistema
```

STDIN

Input for the program (Optional)

Output:

.

- {