



# Módulo 08

## La Arquitectura von Neumann (Pt. 1)



Organización de Computadoras  
Depto. Cs. e Ing. de la Comp.  
Universidad Nacional del Sur



# Copyright

- Copyright © **2011-2024** A. G. Stankevicius
- Se asegura la libertad para copiar, distribuir y modificar este documento de acuerdo a los términos de la **GNU Free Documentation License**, Versión 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera
- Una copia de esta licencia está siempre disponible en la página <http://www.gnu.org/copyleft/fdl.html>
- La versión transparente de este documento puede ser obtenida de la siguiente dirección:

<http://cs.uns.edu.ar/~ags/teaching>



# Contenidos

- Organización multinivel
- Concepto de máquina virtual
- Niveles en una máquina contemporánea
- Evolución histórica de los niveles
- Tecnologías de memoria principal
- Organización del procesador
- Arquitecturas de n-direcciones
- Subsistema de Entrada/Salida



# ¿Qué hay ahí adentro?



# Organización en capas

- Una computadora es una máquina que puede resolver problemas ejecutando las instrucciones que le van siendo indicadas
  - La secuencia de instrucciones que describen cómo hacer una cierta tarea se denomina **programa**
  - El conjunto de instrucciones que puede ejecutar un cierto procesador constituye su **lenguaje máquina**
  - El lenguaje coloquial empleado por las personas difiere considerablemente del lenguaje máquina



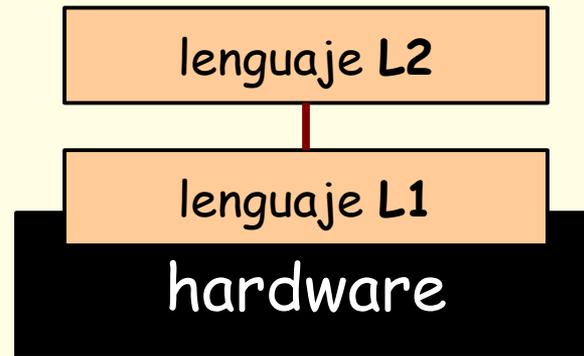
# Organización en capas

- Programar una computadora implica que las personas logren “hablar” con las máquinas
- ¿Cómo podemos reconciliar estos lenguajes tan diferentes?
  - ➔ Una posibilidad es hacer uso de una jerarquía de niveles en los que cada uno de éstos haga uso de un lenguaje con un mayor nivel de abstracción
  - ➔ La idea es que a medida que ascendamos en la jerarquía el lenguaje se vaya alejando de la máquina y al mismo tiempo acercando a nosotros



# Visión multinivel

- El contar con múltiples niveles permite crear nuevas máquinas sobre un mismo hardware:

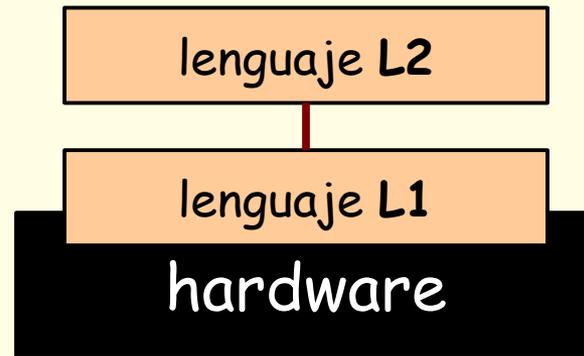


- **L1** estará compuesto de aquellas primitivas de uso conveniente
- Esto permite implementaciones flexibles, ya que **independiza a la máquina del hardware subyacente**



# Visión multinivel (cont)

- El contar con múltiples niveles permite crear nuevas máquinas sobre un mismo hardware:



- L2 estará más próximo a los algoritmos que se quieren implementar
- Esto facilita la programación y también posibilita la resolución de problemas más complejos



# Vinculación entre los niveles

- Existen esencialmente dos mecanismos que nos permiten vincular a los niveles entre sí:
  - ➔ Traducción: la **traducción** consiste en tomar un programa escrito en un cierto lenguaje para transformarlo en una secuencia de instrucciones equivalente de en un lenguaje de nivel inferior
  - ➔ Interpretación: la **interpretación** consiste en tomar instrucción por instrucción de un cierto lenguaje para ir transformándolas en tiempo de ejecución en un conjunto de instrucciones equivalente del lenguaje de nivel inferior



# Mecanismo de traducción

- Características del mecanismo de traducción:
  - El programa que se utiliza para llevar adelante la traducción se denomina **compilador** y está compuesto de instrucciones del nivel inferior
  - El programa original de nivel superior **sólo se utiliza durante la traducción**
  - Luego, la computadora se desentiende de este programa ya que **en tiempo de ejecución sólo accederá al nuevo programa** de nivel inferior



# Mecanismo de interpretación

## ● Características del mecanismo de interpretación:

- El programa que se utiliza para llevar adelante la traducción se denomina **intérprete** y también está compuesto de instrucciones del nivel inferior
- En contraste, este mecanismo **no genera un nuevo programa en el nivel inferior**, ya que la conversión se hace dinámicamente en tiempo de ejecución
- Por esta razón, **se debe conservar el programa original de nivel superior hasta el tiempo de ejecución**, puesto que recién ahí se lleva adelante la conversión



# Concepto de máquina virtual

- Más que pensar en términos de compilación o de interpretación para vincular las distintas capas entre sí, es conveniente abstraerse completamente de las capas inferiores
  - La idea es imaginar que existe una **máquina virtual** que directamente ejecuta las instrucciones de ese nivel superior
  - El lenguaje de ese nivel superior **se convierte en un nuevo lenguaje máquina**, en esta ocasión de una máquina virtual



# Relación entre niveles

- Con el objeto de que la traducción y/o la interpretación resulten viables y eficientes es esencial que **los niveles adyacentes no difieran en gran medida**
  - ➔ Recordemos nuestro objetivo inicial, cada nuevo nivel tiene por objeto acercarnos más a los usuarios
  - ➔ En principio podemos ir incorporando nuevas capas, cada una con su respectivo lenguaje máquina, hasta que alcancemos un nivel cuyo lenguaje resulte satisfactorio, es decir, lo suficientemente próximo a los usuarios del sistema

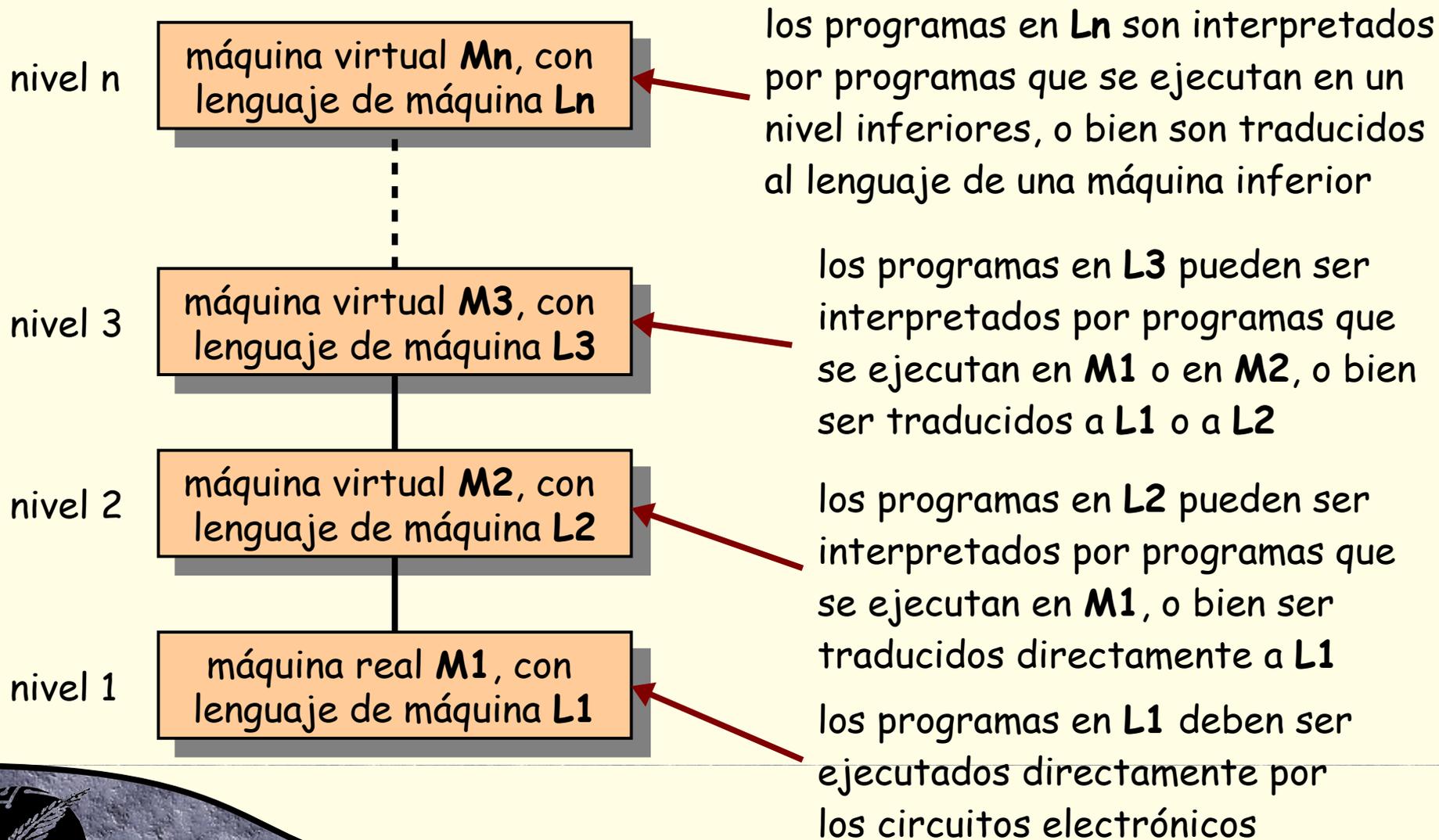


# Lenguajes vs. máquinas

- En cada nivel de esta organización en capas se identifican dos componentes principales:
  - La **máquina** (virtual o no), caracterizada por el conjunto de instrucciones que puede ejecutar
  - El **lenguaje**, caracterizado por las facilidades a disposición del programador en ese nivel
- Obsérvese que una máquina determina a su lenguaje de la misma manera que un lenguaje determina a su máquina



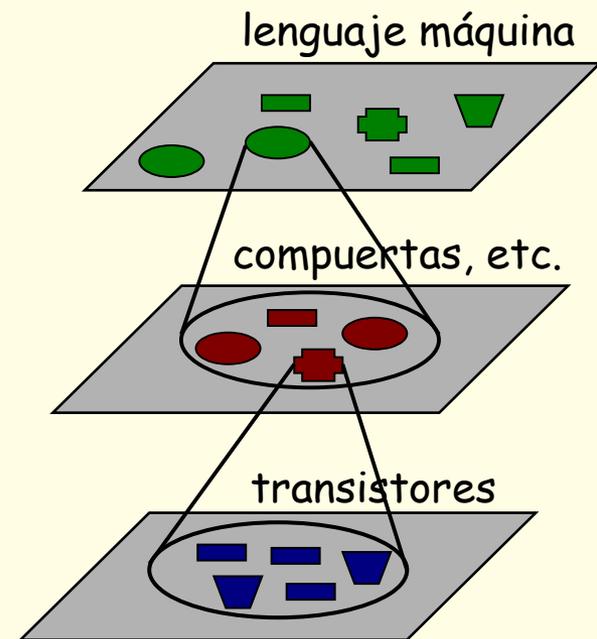
# Organización multinivel



# Organización multinivel

- Recordemos que esta organización multinivel permite atacar el complicado problema de relacionar humanos con computadoras de una manera paulatina

- La idea es que cada capa se construya con las capacidades provistas por la capa inferior
- A su vez, cada capa proveerá de mayores capacidades a las capas superiores



# Máquinas contemporáneas

- La organización multinivel de una **máquina contemporánea** se compone de dos o más niveles
  - ➔ Más aun, es posible que en función de las nuevas tecnologías abarque hasta seis o incluso más niveles
- Nuestro repaso de los distintos niveles que usualmente componen estas máquinas comienza a nivel del hardware
  - ➔ Un electrónico o un físico estaría en condiciones de concebir incluso varias capas adicionales por debajo



# Máquinas contemporáneas

## ● Nivel 0 (la máquina real, no virtual):

- ➔ Este es el nivel de corte de nuestro análisis, por debajo existen otros niveles cuyo estudio en detalle corresponde a otras ramas de la ciencia
- ➔ Está compuesto del conjunto de compuertas lógicas que implementa cada uno de las unidades funcionales que conforman la computadora
- ➔ Cabe destacar que se construye usando componentes analógicos los cuales, sin embargo, terminan evidenciando un comportamiento digital



# Máquinas contemporáneas

## ● Nivel 1 (microprogramación):

- Es el verdadero lenguaje máquina, pues en el nivel anterior no existe el concepto de programa como una secuencia de instrucciones a ser ejecutadas
- En este nivel corre un programa denominado **microprograma** el cual es capaz de interpretar las instrucciones del nivel 2
- Puede tener más de un intérprete corriendo en este nivel; naturalmente cada uno definirá un nuevo lenguaje de nivel 2 junto con su correspondiente máquina virtual



# Microprogramación

- Recordemos que la microprogramación aparece como alternativa al hacer uso de la ley de equivalencia entre **HW** y **SW**
- Es decir, en este nivel es posible optar por ejecutar directamente en hardware las instrucciones más simples y resolver vía software las más complejas
- La idea de los diseñadores del sistema era alcanzar un adecuado **balance entre costo y desempeño**



# Microprogramación

- A su vez, hay que considerar que la técnica de microprogramación floreció en la época en la que la memoria principal era muy lenta
  - En ese contexto contar con un set de instrucciones complejo permitía generar programas más compactos y con menor requerimiento de ancho de banda
  - Este tipo de arquitectura se denomina justamente **Complex Instruction Set Computer (CISC)**
  - La familia Intel **x86** es posiblemente el ejemplo más conocido de arquitectura **CISC**



# Microprogramación

- La utilidad de la microprogramación empezó a ceder de la mano de dos avances:
  - ➔ Inicialmente el **crecimiento exponencial de la cantidad de transistores disponibles** permitió implementar en hardware más y más funcionalidad del nivel 2
  - ➔ Naturalmente el mayor impacto vino de la mano de la **mejora en el desempeño de la memoria principal**
- En la actualidad se opta por **simplificar el set de instrucciones**, dando a lugar a lo que hoy en día se conoce como arquitecturas **RISC**



# Máquinas contemporáneas

## ● Nivel 2 (máquina convencional):

- Este es el primer nivel al alcance del programador, más allá de que el verdadero lenguaje máquina sea el provisto por el nivel inmediato inferior
- Es posible que un mismo **HW** soporte múltiples máquinas convencionales (a través de múltiples intérpretes corriendo en el nivel anterior)
- En las computadoras de tipo **RISC** se ejecutan las instrucciones de este nivel directamente en **HW**, es decir, **no cuentan con la capa de microprogramación**



# Máquinas contemporáneas

## ● Nivel 3 (sistema operativo):

- Se trata de un nivel híbrido, gran cantidad de las instrucciones de este nivel son las mismas que las del nivel inmediato inferior
- Existen muchas más variantes de máquinas en este nivel que en los niveles anteriores
- Las nuevas facilidades provistas por este nivel fueron incorporadas por un intérprete llamado **sistema operativos** por razones históricas como veremos
- En este nivel aparece la posibilidad de **correr dos o más programas en simultáneo**



# Máquinas contemporáneas

- Existe una separación fundamental entre los niveles 3 y 4:
  - A diferencia de los niveles superiores, los niveles 1, 2 y 3 **no fueron diseñados para ser usados directamente por el programador de aplicaciones**
  - Los primeros tres niveles tiene por objeto sustentar a los intérpretes necesarios para que las máquinas virtuales de nivel superior funcionen correctamente
  - Los encargados de implementar estos programas se denominan **programadores de sistema**



# Máquinas contemporáneas

## ● Nivel 4 (lenguaje ensamblador):

- Este es el primer nivel en donde se hace uso de los programas traductores
- Por otra parte, **es el primer nivel que cuenta con un lenguaje simbólico** (los niveles inferiores sólo cuentan con un lenguaje numérico, compuesto de 1s y 0s)
- Esta cambio hace que el nivel 4 sea el primer nivel inteligible por las personas
- El programa encargado de traducir los programas de este nivel se denomina **ensamblador**



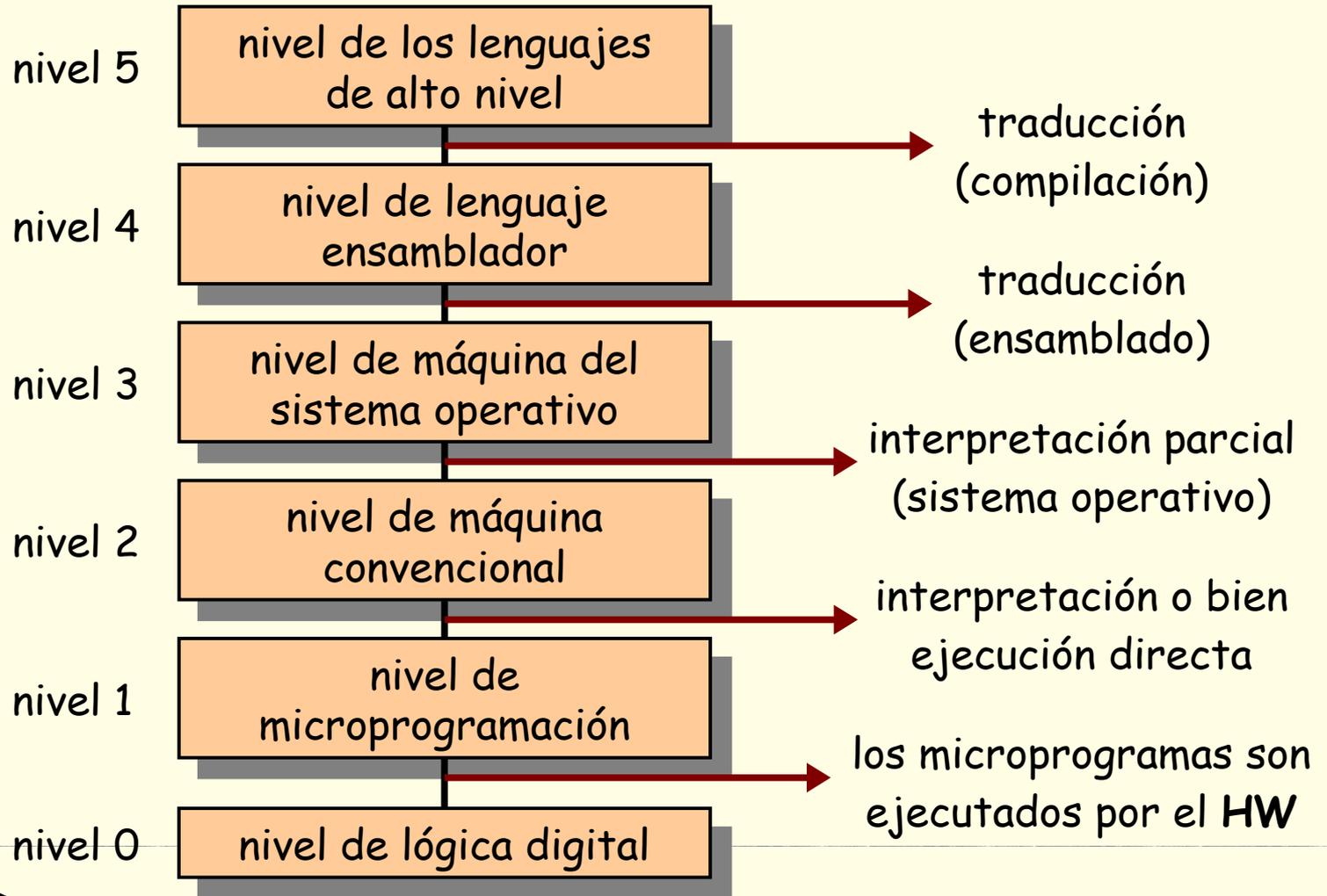
# Máquinas contemporáneas

## ● Nivel 5 (lenguajes de alto nivel):

- Este nivel está compuesto por los distintos lenguajes de programación de alto nivel, tales como **Java**, **Pascal** y **C** entre otros
- Nótese que, como es usual, cada lenguaje define una máquina virtual diferente para este nivel
- Los programas escritos en estos lenguajes suelen ser traducidos a instrucciones de los niveles 3 o 4 mediante compiladores, si bien también existen algunos lenguajes de programación que son directamente interpretados



# Síntesis



# Ejemplo

- A manera de ilustración de las diferencias entre los lenguajes de las distintas capas, analicemos el siguiente problema:
  - Se desea mostrar por pantalla la suma de los cuadrados de los primeros 100 naturales
- Supongamos que el lenguaje del nivel 5 es el lenguaje **C**, y que la arquitectura de los niveles inferiores es la **MIPS**
  - **MIPS** fue la primer arquitectura **RISC**



# Ejemplo

- El programa de nivel 5 para resolver este problema podría ser:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    int i, sum = 0;
    for (i=0; i<=100; i++)
        sum += i * i ;
    printf("The sum from 0..100 is %d\n", sum);
}
```



# Ejemplo

- Este mismo programa expresado en el lenguaje del nivel 4 resulta:

```
.text
.align 2
.globl main
main:
    subu    $sp, $sp, 32
    sw     $ra, 20($sp)
    sd     $a0, 32($sp)
    sw     $0, 24($sp)
    sw     $0, 28($sp)
loop:
    lw     $t6, 28($sp)
    mul   $t7, $t6, $t6
    lw     $t8, 24($sp)
    addu  $t9, $t8, $t7
    sw     $t9, 24($sp)
    addu  $t0, $t6, 1
    sw     $t0, 28($sp)
    ble   $t0, 100, loop
    la    $a0, str
    lw    $a1, 24($sp)
    jal  printf
    move $v0, $0
    lw   $ra, 20($sp)
    addu $sp, $sp, 32
    jr   $ra

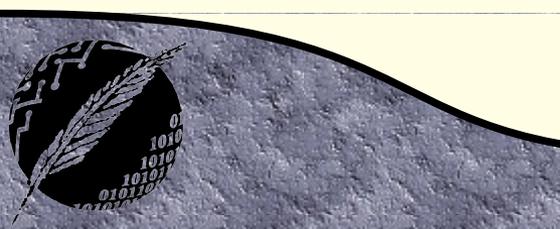
.data
.align 0
str:
.asciiz "The sum from 0 .. 100 is %d\n"
```



# Ejemplo

- Finalmente, expresado en el lenguaje propio del nivel 2 resulta:

```
00100111101111011111111111100000
10101111101111110000000000010100
1010111110100100000000000100000
1010111110100101000000000100100
101011111010000000000000011000
101011111010000000000000011100
100011111010111000000000011100
100011111011100000000000011000
000000111001110000000000011001
001001011100100000000000000001
0010100100000001000000001100101
101011111010100000000000011100
0000000000000000011110000010010
00000110000111110010000100001
000101000010000011111111110111
101011111011100100000000011000
001111000000100000100000000000
100011111010010100000000011000
000110000010000000000011101100
0010010010000100000001000110000
1000111110111111000000000010100
0010011110111101000000000100000
00000111110000000000000001000
0000000000000000000100000100001
```



# Evolución histórica

- Las primeras computadoras allá por la década del '40 sólo contaban con dos niveles:
  - El nivel de la máquina convencional donde se realizaba la programación
  - El nivel de la lógica digital donde eran ejecutados los programas
- No había manera alguna de que la computación en estos términos se tornara masiva
  - ¡Los programadores eran en ocasiones los propios constructores de la computadora!



# Evolución histórica

- En **1951** Maurice Wilkes propuso **una arquitectura con tres niveles**, con el objetivo de simplificar el hardware del nivel inferior
  - ➔ Wilkes en esencia **inventó la microprogramación**
  - ➔ Esta simplificación a nivel de **HW** se logra en virtud de que el repertorio de las microinstrucciones es menor que el de las instrucciones a nivel de la máquina convencional
  - ➔ En esa época las computadoras todavía se construían con válvulas, por lo que esta simplificación a nivel de **HW implicaba un cuantioso ahorro**



# Evolución histórica

- La década del '50 también se caracteriza por el desarrollo de los primeros **lenguajes de programación**:
  - ➔ Los lenguajes ensamblador de las distintas computadoras de la época (tales como la **EDVAC**, la primera computadora de programa almacenado)
  - ➔ Los lenguajes de programación imperativos **Fortran** (1957) y **Cobol** (1959) así como el lenguaje de programación funcional **Lisp** (1958) fueron diseñados e implementados a lo largo de esa década



# Evolución histórica

- Alrededor de la década del '60 se intentó automatizar las repetitivas tareas que tenían que llevar a cabo los operadores
  - Recordemos lo enrevesado que resultaba ejecutar un programa escrito en alguno de estos lenguajes de programación en la época de las tarjetas perforadas y de las cintas de carrete abierto
  - Un programa llamado **sistema operativo** cuyo propósito era asistir al operador se mantenía en ejecución todo el tiempo



# Evolución histórica

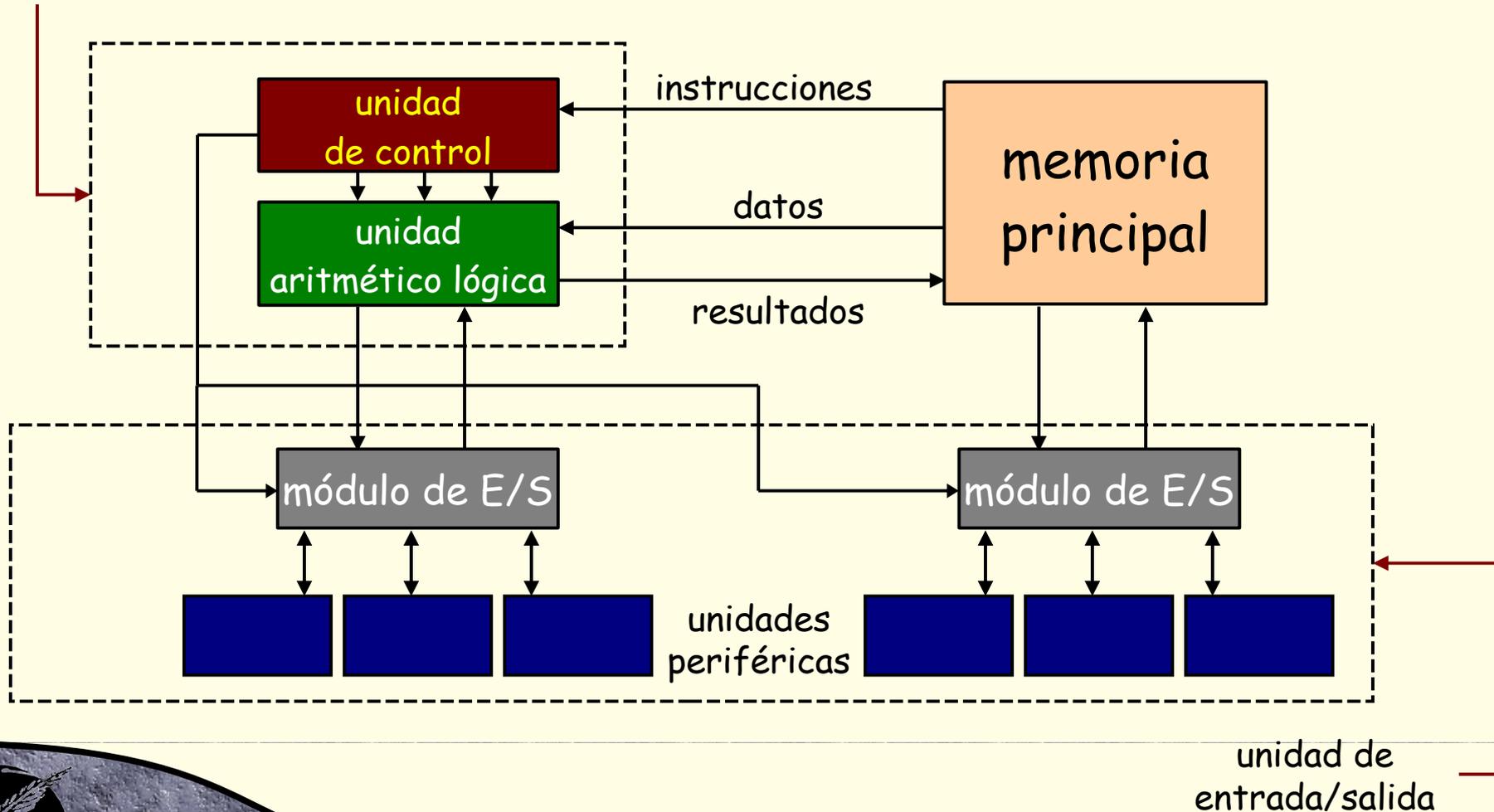
## ● Continúa:

- ➔ Recordemos también que el lenguaje de este nuevo nivel es híbrido, está compuesto mayormente por las instrucciones del nivel inmediato inferior y sólo una pocas nuevas instrucciones propias
- ➔ Las nuevas instrucciones se conocían como **macros al sistema operativo** o también **llamadas al supervisor**
- ➔ Un importante avance que incorporaron los sistemas operativos fue posibilitar el compartir el **HW** entre múltiples usuarios en simultáneo, dando a lugar a lo que se conoce como **sistemas de tiempo compartido**



# Arquitectura von Neumann

CPU



# Memoria

- En el modelo sugerido por la arquitectura von Neumann, la memoria principal es utilizada para almacenar tanto programas como datos
- La memoria se clasifica en función de distintos aspectos:
  - Por la forma de **acceso**
  - Por la forma de **retención de los datos**
  - Por el grado de **persistencia de la información**



# Tipos de memoria

## ● Por la forma de acceso:

- Memoria de acceso secuencial: este tipo de memoria estipula que los datos en ella almacenados deben ser accedidos en un determinado orden secuencial
- Memoria de acceso aleatorio: esta es la memoria convencional, en la cual no existe restricción alguna en el orden en el cual se deban acceder los datos en ella almacenados
- Memoria de acceso por contenido: esta memoria, denominada **asociativa**, se caracteriza por ser capaz de resolver búsquedas por contenido



# Tipos de memoria

- Por la forma de retención de los datos:
  - ➔ Memoria estática: en este tipo de memoria una vez almacenado un dato el mismo queda retenido sin ser alterado hasta que vuelva a ser modificado. Logra este comportamiento insumiendo mayor cantidad de componentes por cada celda
  - ➔ Memoria dinámica: en este tipo de memoria los datos son retenidos por breves intervalos de tiempo. Es decir, el sistema debe **leer y volver a escribir cada locación de memoria antes de que se pierda su contenido**. El beneficio es que insume una menor cantidad de componentes por cada celda



# Tipos de memoria

- Por el grado de permanencia de la información:
  - ➔ Volátiles: en este tipo de memoria la información almacenada se pierde en el mismo instante que se corte el suministro de corriente eléctrica
  - ➔ No volátiles: en este tipo de memoria la información almacenada persiste en el tiempo, más allá de que se disponga o no de una fuente ininterrumpida de alimentación



# Tecnologías de memoria

- Tecnologías para la memoria principal:
  - Memoria **RAM** (Random Access Memory)
  - Memoria **ROM** (Read-Only Memory)
  - Memoria **PROM** (Programmable **ROM**)
  - Memoria **EPROM** (Erasable **PROM**)
  - Memoria **EEPROM** (Electrically Erasable **PROM**)
  - Memoria **MRAM** (Magnetoresistively **RAM**)
  - Memoria estilo **FLASH**



# Gemini, ¡proceda!

A Contame en qué anda la tecnología de memoria magnetoresistiva

Show drafts ▾

✦ ¡Claro! Con gusto te explico en qué anda la tecnología de memoria magnetoresistiva (MRAM).

**La memoria MRAM es una tecnología prometedora que busca revolucionar el mundo del almacenamiento de datos.** A diferencia de las memorias tradicionales, como la DRAM, la MRAM utiliza la magnetización de materiales para almacenar información, lo que le confiere varias ventajas:

- **No volatilidad:** Los datos se mantienen incluso cuando se corta la alimentación eléctrica, al igual que en una memoria flash.
- **Alta velocidad:** La MRAM puede leer y escribir datos muy rápidamente, acercándose a la velocidad de la DRAM.
- **Alta densidad:** Permite almacenar grandes cantidades de datos en un espacio reducido.
- **Gran durabilidad:** Es resistente a las radiaciones y a las altas temperaturas, lo que la hace ideal para aplicaciones industriales y aeroespaciales.
- **Bajo consumo energético:** Al no requerir una constante recarga de la carga eléctrica, consume menos energía que la DRAM.

¿En qué anda la tecnología MRAM actualmente?

La MRAM ha estado en desarrollo durante varias décadas y ha experimentado avances significativos en los últimos años. Actualmente, se encuentra en una etapa de madurez y se utiliza en aplicaciones específicas donde se requiere alta velocidad, baja latencia y resistencia. Algunos de los avances más destacados son:

- **Miniaturización:** Los fabricantes están logrando reducir el tamaño de las celdas de memoria MRAM, lo que permite aumentar la densidad de almacenamiento.
- **Integración en dispositivos:** La MRAM se está integrando en una variedad de dispositivos, como microcontroladores, tarjetas inteligentes y sistemas embebidos.
- **Nuevos materiales:** Se están investigando nuevos materiales magnéticos para mejorar el rendimiento y la eficiencia de la MRAM.
- **Aplicaciones emergentes:** La MRAM se está explorando en aplicaciones como la inteligencia artificial, el Internet de las Cosas (IoT) y la computación de alto rendimiento.

¿Cuáles son los desafíos y el futuro de la MRAM?

A pesar de sus ventajas, la MRAM aún enfrenta algunos desafíos, como:

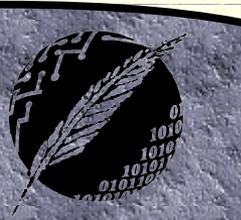
- **Costo:** La producción de chips MRAM aún es más costosa que la de otros tipos de memoria.
- **Velocidad de escritura:** Aunque la velocidad de lectura es muy alta, la velocidad de escritura aún puede ser un cuello de botella en algunas aplicaciones.

Sin embargo, se espera que estos desafíos se superen con el tiempo, y se prevé que la MRAM se convierta en una tecnología dominante en los próximos años.

**En resumen, la memoria MRAM es una tecnología prometedora con un gran potencial para transformar el mundo de la computación.** Su combinación de velocidad, densidad, durabilidad y no volatilidad la convierte en una opción atractiva para una amplia gama de aplicaciones.

¿Te gustaría saber más sobre alguna aplicación específica de la memoria MRAM o sobre su funcionamiento técnico?

No dudes en preguntar.



# ¿Preguntas?

