Compiladores y Lenguajes de Programación

Maria de Guadalupe Cota Ortiz

Organizaciones que rigen las normas para estandarización de Lenguajes de Programación

- **IEEE** (Instituto de Ingenieros Eléctricos y Electrónicos)
- **ANSI** (Instituto Nacional Estadounidense para estandarización de Normas)
- **W3C** (Web Services)
- **OSI** (Organización Internacional para Estandarización de Normas)

Estandarización

Estándares.- Definiciones o formatos que se aprueban o reconocen por organizaciones de estandarización.

Generalmente estos organismos están formados por el conjunto de empresas más representativas de un sector o de un campo de la producción.

Los estándares permiten que las industrias desarrollen componentes con las garantías suficientes de **interacción, funcionalidad y calidad**.

Lenguajes de Programación

Conceptos

- **Traductor.-** Es cualquier procesador de lenguajes que acepta un programa fuente escrito en cierto lenguaje como entrada, y produce programas equivalentes en otros lenguajes como salida.
- **Intérprete.-** Es el que ejecuta programas sin una traducción explícita, es decir, no hay una fase de traducción y otra de ejecución. El intérprete ejecuta las instrucciones.

¿Qué es un Compilador?

- Es una herramienta de programación que lee un **programa fuente** y lo traduce a un programa equivalente en otro lenguaje (**objeto**).

Etapas

- **Etapa de compilación.-** Proceso en el que se lee un código fuente en un lenguaje y se traduce a un programa equivalente en otro lenguaje (Objeto).
- **Etapa de Enlace (Link).-** Se enlaza el código del programa (**Objeto**) con recursos del lenguaje que se utilizan en el programa, y se produce un archivo **ejecutable**.

SISTEMA DE PROCESAMIENTO DE UN LENGUAJE

Fases de traducción

preprocesador

programa fuente

compilador

programa objeto en lenguaje ensamblador

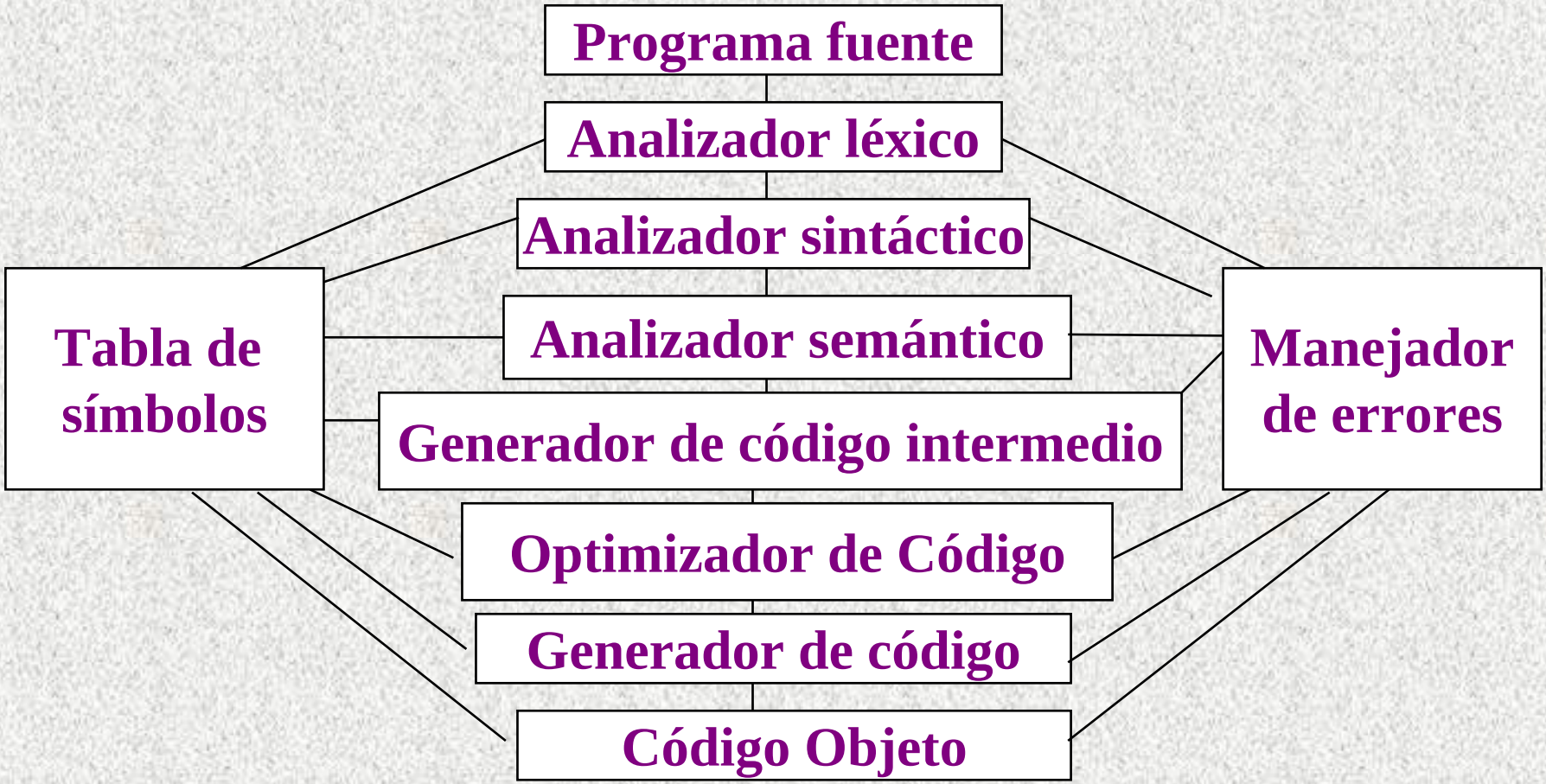
ensamblador

código de máquina relocalizable

editor de carga y enlace ← biblioteca archivos

código de máquina absoluto

Fases de un compilador



Generación de código intermedio

Una de las representaciones intermedias es la construcción del árbol de sintáxis y la del “código de tres direcciones”, que guarda el operador, el nombre del resultado y los dos operandos.

Con este lenguaje, el compilador debe generar un nombre temporal para guardar los valores calculados por cada instrucción

Ejemplo (instrucción while y asignación)

```
while a < b do
    if c < d then
        x := y + z
    else
    x := y - z
```

Cuádrupla

Operador	op1	op2	result
+	y	z	t1
:=	t1		x
-	y	z	t2
:=	t2		x

Código de tres direcciones:

```
L1:   if a < b   goto L2
      goto Lsiguiente
L2:   if  c < d   goto L3
L3:   t1:=  y + z
       x :=  t1
       goto L1
L4:    t2 := y – z
       x := t2
       goto L1
Lsiguiente:
```

Optimización de código

En esta etapa se trata de mejorar el código intermedio, de forma que sea un código más rápido de ejecutar.

Ejemplo

```
for ( i = 0 ; i < 10; ++ i ) {    j = 17 * i;    }
```

Se reemplaza por:

```
j = -17;
for ( i = 0; i < 10; ++ i ) {   j = j + 17;     }
```

```
for j from 0 to 10
        for i from 0 to 20
                a[i,j] = i + j
```

Se reemplaza por:

```
for i from 0 to 20
        for j from 0 to 10
                a[i,j] = i + j
```

Generación de código

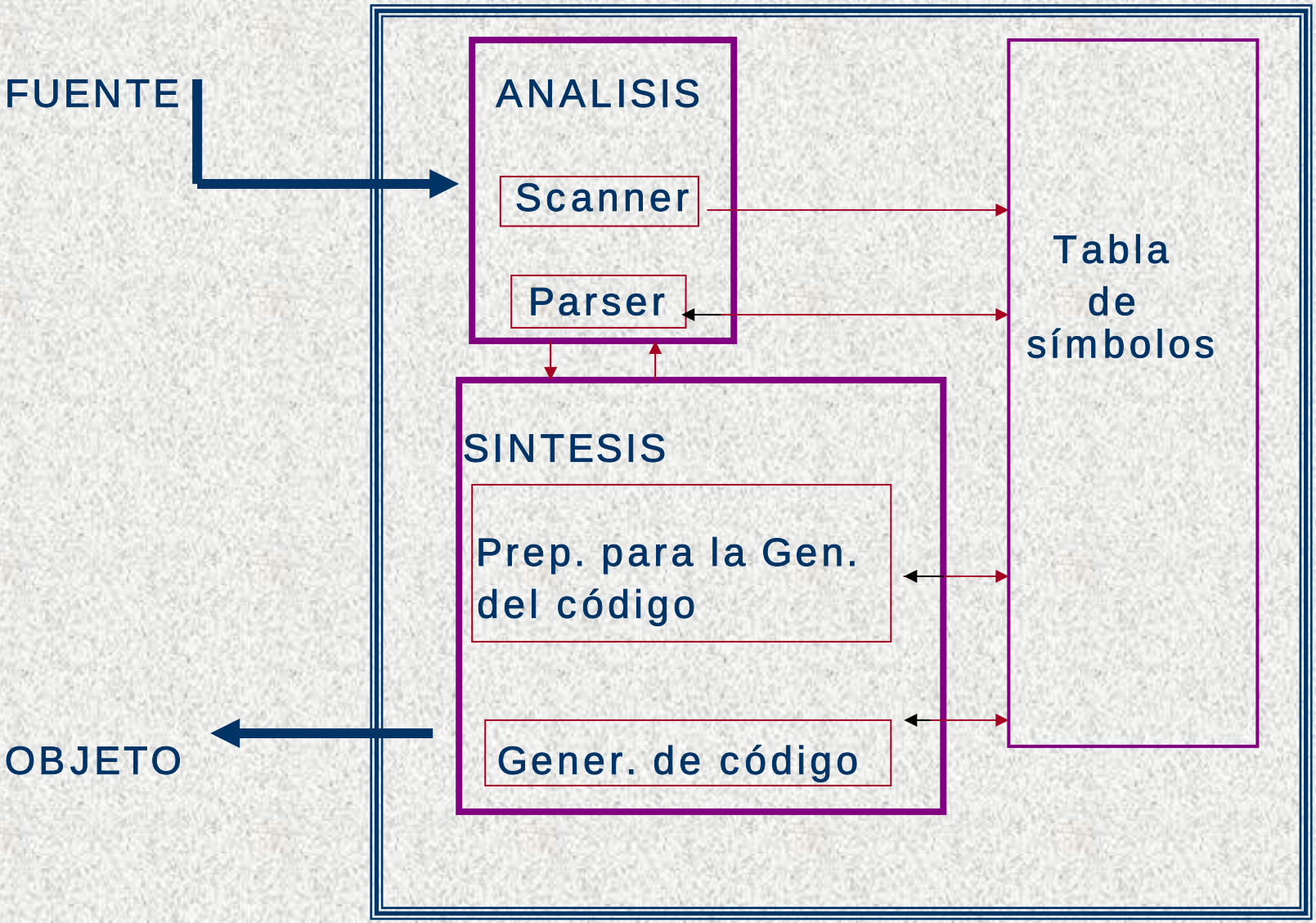
La fase final de un compilador es la generación de código objeto, que por lo general consiste en código máquina o código ensamblador.

Generación de código

Para cada tipo de proposición de tres direcciones, se puede diseñar un esqueleto de código que perfila el código objeto que ha de generarse. Por ejemplo, cada proposición de tres direcciones de la forma x := y + z, se puede traducir a la secuencia de codigo:

```
MOV   y, R0
ADD   z, R0
MOV R0, x
```

ESQUEMA DE BLOQUES DE UN COMPILADOR



Análisis y Síntesis

- **Análisis:** Es la etapa donde se divide el programa fuente en componentes y se crea una representación intermedia del mismo.
- **Síntesis:** Es la etapa donde se construye el programa objeto y es donde se requieren técnicas más especializadas para las tareas a realizar.

Análisis y Síntesis

En la etapa de análisis se determinan las operaciones que implica el programa fuente y se construye una estructura tipo arbol, llamada ‘arbol sintáctico’.

Analizador léxico

La función primordial es agrupar caracteres de la entrada en “Tokens”, los cuales serán proporcionados al analizador sintáctico y se encarga de introducir información preliminar en la tabla de símbolos.

Cuenta con un escáner que se compone de un conjunto de funciones, asignándose una a cada símbolo que se reconocerá.

Análisis Léxico

Ej. Posicion = inicial + velocidad * 60

Identificador:	Posicion
Operador de asignacion:	=
Identificador:	inicial
Operador aritmético:	+
Identificador:	velocidad
Operador aritmético:	*
Constante:	60

Tareas del Analizador Léxico

Realiza un escaneo del programa fuente hasta reconocer un "Token".

"Token": Unidad léxica indivisible.

Ejemplos: While, if, = =, =>, etc.

La secuencia de caracteres correspondiente se llama "lexema" (componente léxico)

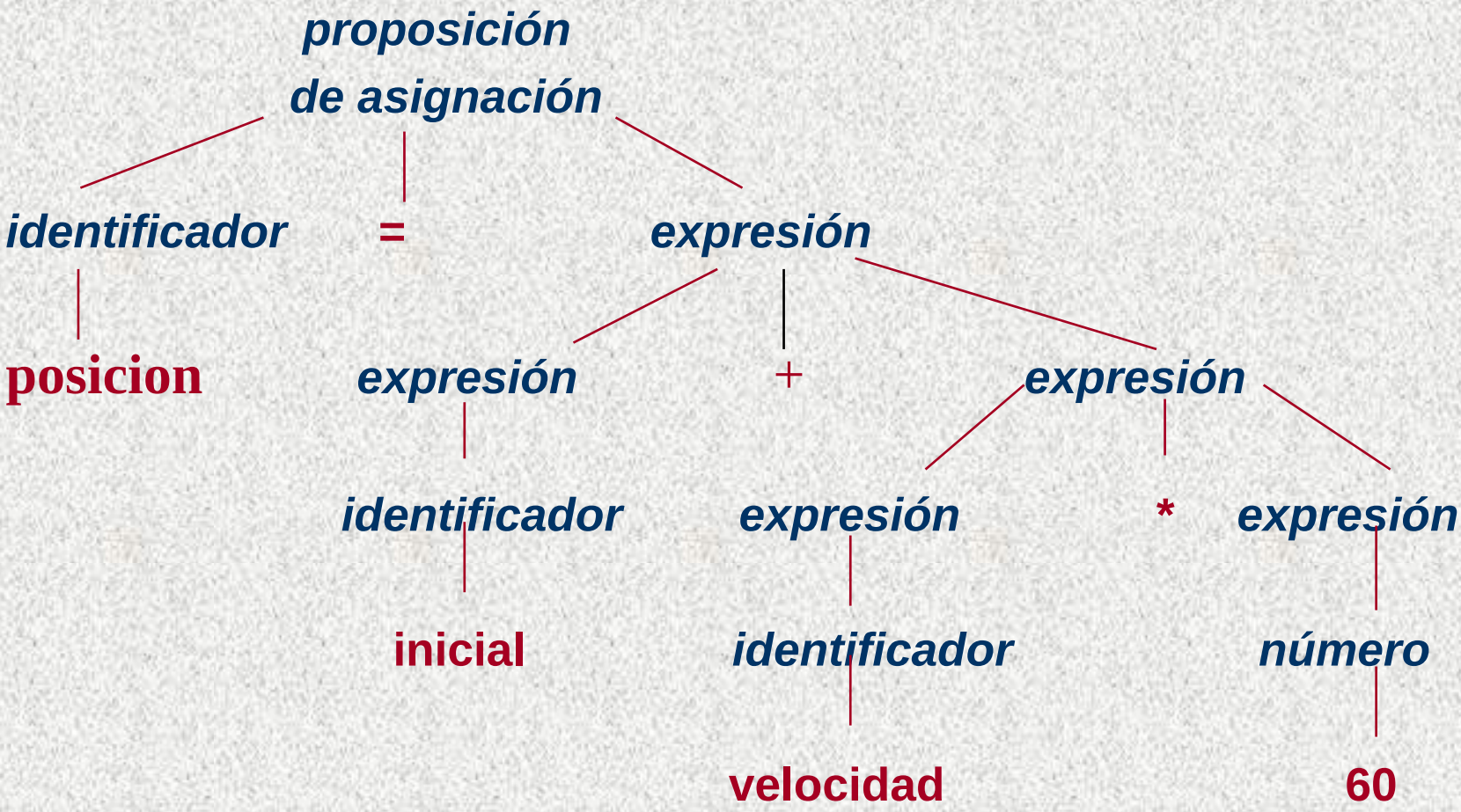
Tareas del Analizador Léxico

- Procesa directivas al compilador
- Introduce información preliminar en la tabla de símbolos
- Elimina separadores innecesarios
- Sustituye macros

Ejemplos de tokens

- Palabras reservadas
- Identificadores
- Operadores
- Constantes
- Símbolos de puntuación
- Símbolos especiales
- etc.

Análisis Sintáctico - Jerárquico



Análisis Semántico

- Verificación de tipos
- Verificación de declaraciones de identificadores, funciones y librerías utilizadas.
- Verifica que se cumpla con las restricciones aplicadas en el lenguaje que se trate.

Tabla de símbolos

En ella se registran los atributos de los identificadores detectados en el programa fuente, como pueden ser: nombre, tipo, dirección de memoria, ámbito, inicialización, y en el caso de nombres de procedimientos sus argumentos con método (por valor o por referencia), nombres y tipos.

Registro de entidades en un programa

Cuando se quiere validar que una entidad en el lenguaje puede ser operada bajo una determinada circunstancia, lo primero que debe determinar es ‘de que tipo es’ dicha entidad.

Normalmente se requiere consultar una tabla de símbolos donde se encuentren registradas las entidades, entre otras cosas con sus tipos y direcciones. Esta tabla se construye normalmente en la ‘primera pasada’ de revisión del código fuente

Ejemplo

```
Entero a, b[5];
Apuntador_a Entero c;
Caracter d[10];
```

Entidad	Tipo	Dirección
a	entero	
b	arreglo de entero	
c	apuntador a entero	
d	arreglo de caracter	

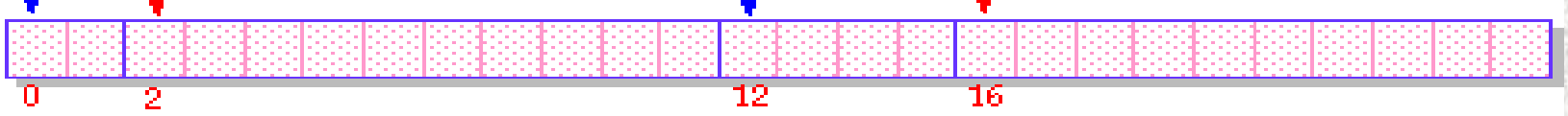
Si la representación de cada tipo está dada por la siguiente tabla

1	Caracter	11	Arreglo Caracter
2	Entero	12	Arreglo Entero
3	Real	13	Arreglo Real
4	Booleano	14	Arreglo Booleano
5	Secuencia	15	Arreglo Secuencia
6	Apuntador Caracter	16	Arreglo Ap. Caracter
7	Apuntador Entero	17	Arreglo Ap. Entero
8	Apuntador Real	18	Arreglo Ap. Real
9	Apuntador Booleano	19	Arreglo Ap. Booleano
10	Apuntador Secuencia	20	Arreglo Ap. Secuencia

Ejemplo

```
Entero a, b[5];
Apuntador_a Entero c;
Caracter d[10];
```

Entidad	Tipo	Dirección
a	2	0
b	12	2
c	7	12
d	11	16



¿Qué es un lenguaje de Programación?

Un lenguaje de programación se utiliza para implementar notaciones que permiten describir algoritmos y estructuras de datos.

Está constituído por un conjunto finito de elementos, reglas de construcción, y una computadora virtual que se encarga de su implementación en tiempo de ejecución.

Atributos de un buen lenguaje de programación:

Claridad, sencillez y unidad

- Es deseable contar con un número mínimo de conceptos distintos cuyas reglas sean tan sencillas y regulares como sea posible.

- La sintaxis afecta la facilidad con que un programa se puede escribir, poner a prueba y más tarde entender y modificar.

Naturalidad en la aplicación.

- El lenguaje deberá suministrar estructuras de datos, operaciones, estructuras de control y una sintaxis natural apropiada.
- Los algoritmos secuenciales, concurrentes, etc., tienen estructuras naturales diferentes.

Naturalidad en la aplicación.

Existen lenguajes particularmente adecuados para cierta clase de aplicaciones como los que permiten el diseño orientado a objetos.

Apoyo para la abstracción

Una tarea importante en la programación es proyectar las abstracciones adecuadas para la solución de un problema y luego implementarlas según las capacidades del lenguaje de programación a utilizar.

Otros aspectos a considerar en el Diseño y Desarollo de LP

1) La computadora donde se van a ejecutar los programas:

- **Arquitectura del procesador**
- **Aspectos del Sistema Operativo**

2) El Modelo de ejecución o computadora virtual que apoya al lenguaje en el equipo real

3) El paradigma o modelo de computación que el lenguaje implementa.

Otros factores importantes

- Facilidad para verificar programas.
- Entorno de programación amigable.- Por ejemplo uso de menús, mouse, etc.
- Portabilidad de los programas hacia otras plataformas.
- Costo de uso.- Consiste en facilitar la modificación, reparación y extensión de programas en períodos posteriores.

Maquina Virtual

Máquinas Virtuales

Un sistema de computador se compone de capas:

- El hardware que es el nivel más bajo.
- El núcleo que se ejecuta en el siguiente nivel y utiliza las instrucciones del hardware para crear un conjunto de llamadas al sistema que las capas exteriores pueden usar.

Máquinas Virtuales

Las Máquinas Virtuales se construyeron para simplificar el proceso del control del hardware de un ordenador porque extienden y enmascaran la funcionalidad del hardware a través de procedimientos y datos abstractos.

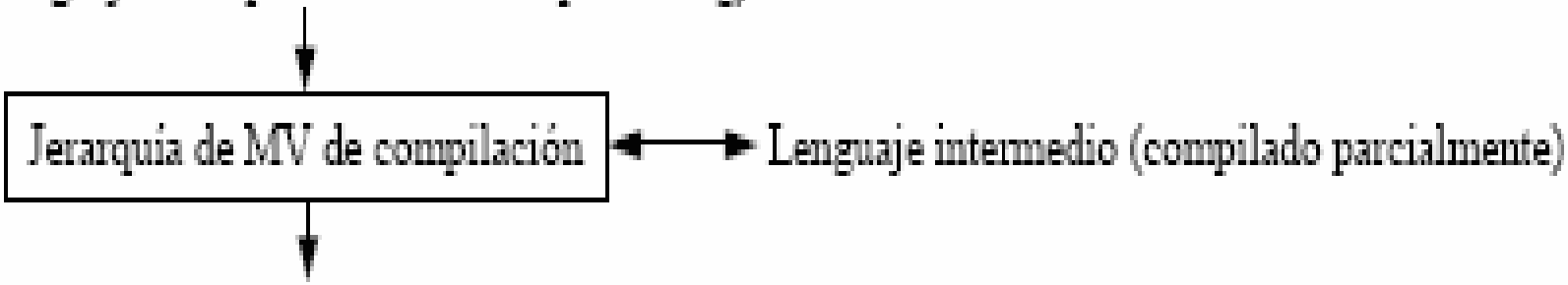
Se pueden identificar dos tipos de Máquinas Virtuales;

Concretas.- las que juegan un papel en la preparación de un programa para su ejecución (tiempo de compilación)

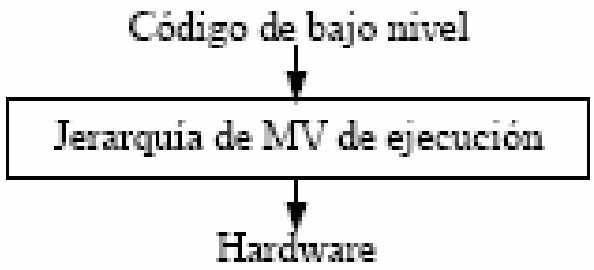
Y las que permiten la ejecución de dicho programa

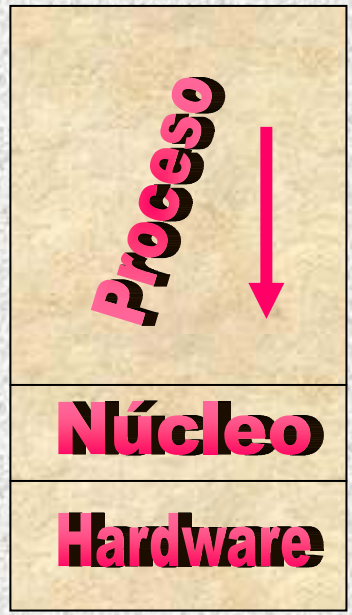
Tiempo de compilación

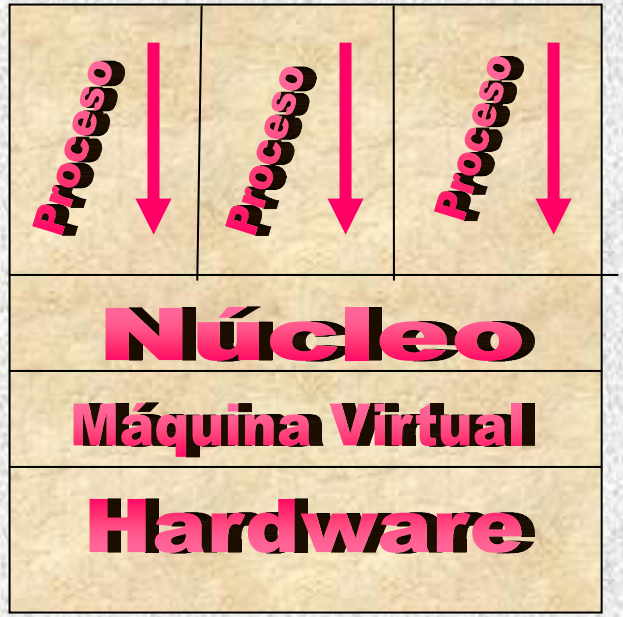
Lenguaje de alto nivel (como Java y C++, pero no los lenguajes interpretados como Lisp o Prolog)



Tiempo de ejecución
(programa compilado)



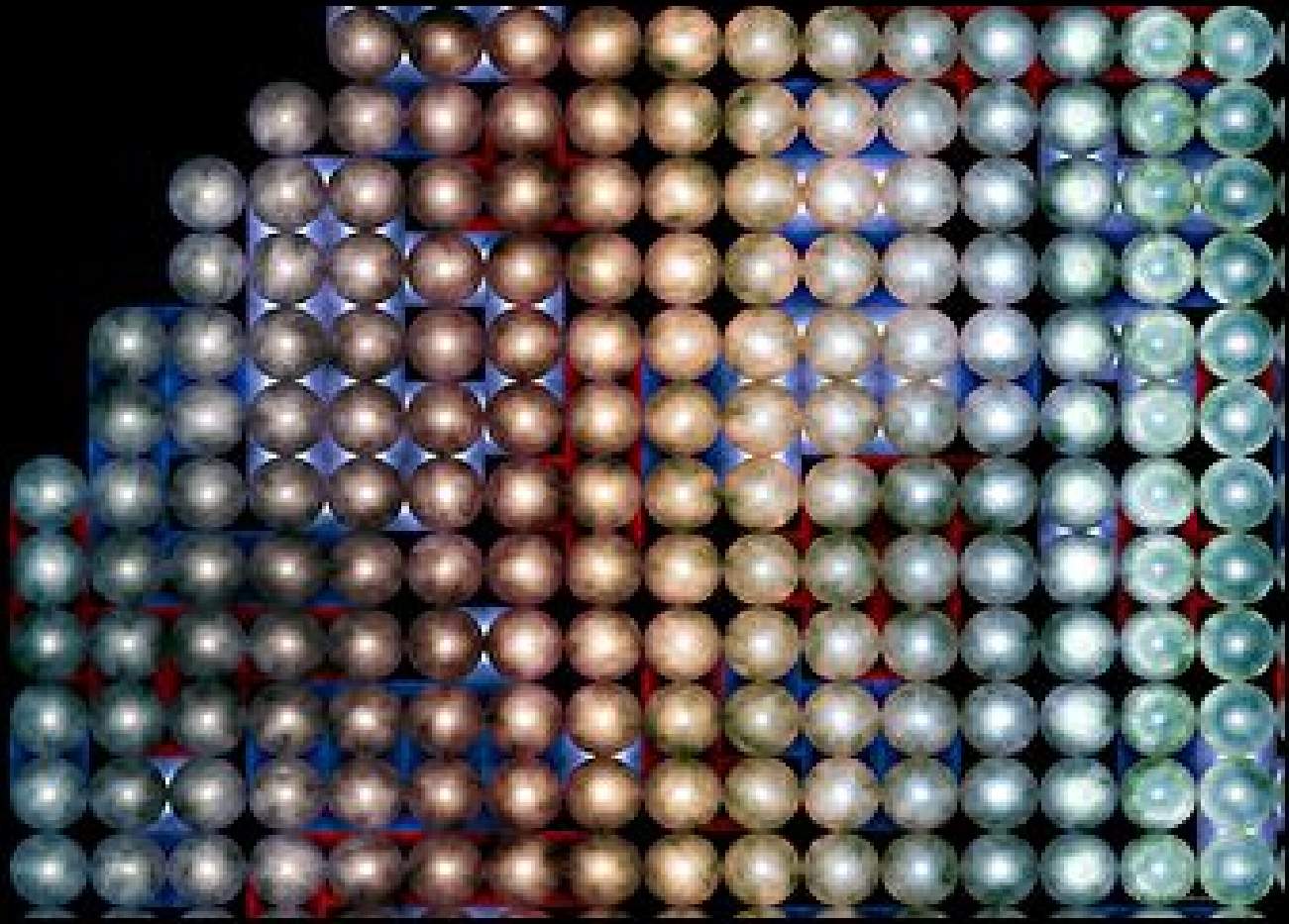




- Algunos sistemas operativos permiten a los programas de aplicación invocar fácilmente a los programas del sistema como si formaran parte de la máquina misma. Este enfoque de capas se denomina “Máquina Virtual”.

Máquinas Virtuales

- Para que las Máquinas Virtuales puedan implementarse, es necesario utilizar métodos de planificación de la CPU (Unidad Central de Proceso) y técnicas de uso de Memoria Virtual. De esta forma, un sistema operativo puede crear la ilusión de que múltiples procesos se ejecutan a la vez. (Ejemplo: MVJ VMWare, CyGwin, etc)



Computadora Virtual

Computadora Virtual

- Las estructuras de datos y algoritmos en tiempo de ejecución que se emplean para la ejecución de un programa definen una computadora virtual que está previamente diseñada por la implementación del lenguaje de programación.

Computadora Virtual

- Si los lenguajes de programación se definen en términos de sus computadoras virtuales, de tal forma que cada lenguaje está asociado con una sola computadora virtual, entonces la descripción de la semántica de cada lenguaje en términos de su computadora virtual es distinta.

Computadora Virtual

- Cuando un lenguaje de programación se está implementando en una computadora particular, se debe determinar primero la computadora virtual que representa una interpretación de la semántica del lenguaje y luego se debe construir a partir de los elementos del hardware y software que suministra la computadora real.

Computadora Virtual

- Por ejemplo, si la computadora virtual contiene una variable entera simple X, el implementador puede optar por:
- Representar X directamente como una localidad de almacenamiento en memoria que contenga el valor de X, o
- Representar X por una localidad de alm. que contenga una “marca de tipo” que designe “entero” junto con un apuntador hacia otra localidad de almacenamiento que contiene el valor de X.

Computadora Virtual

- La organización y estructura de una implementación de lenguaje están determinadas por estas múltiples decisiones que toma el implementador, considerando los diversos recursos de hardware, software disponibles y los costos de uso en la computadora real.

Computadora Virtual

- **Factores que diferencían la implementación del mismo lenguaje en distintas plataformas:**
  1. **La concepción de la computadora virtual.**
  2. **Los recursos de la computadora en la cual se va a implementar el lenguaje**
  3. **Las decisiones respecto de la simulación de elementos de la computadora virtual usando recursos de la computadora real, así como la construcción del traductor que apoyará las opciones de representación de la computadora virtual.**

Jerarquías de computadoras

Cuando un programador decide hacer un programa en algún lenguaje, la computadora virtual está formada por una jerarquía de computadoras virtuales

Capas de computadora virtuales para un programa en C

Computadora Virtual implementada por el modelo de ejecución desarrollado en el programa en C para uso de la computadora virtual del Lenguaje C

Computadora Virtual del Lenguaje C (Implementada por rutinas de biblioteca en tiempo de ejecución cargadas con el programa compilado)

Computadora Virtual del Sistema Operativo (implementada por programas en lenguaje máquina en ejecución en la computadora virtual de Firmware)

Computadora Virtual de Firmware (instrucciones en lenguaje máquina implementadas por un microcódigo ejecutado por la computadora real)

Computadora de Hardware Real (Implementada por dispositivos físicos)

lcota@hades.mat.uson.mx
lcota@gauss.mat.uson.mx