Parte I: Elementos del lenguaje Ada

UNIVERSIDAD DE CANTABRIA

Notas:

UNIVERSIDAD DE CANTABRIA

1. Introducción a los computadores y su programación.
   - Arquitectura básica de un computador.
   - El software del sistema.
   - Lenguajes de Alto Nivel.
   - El proceso de compilación.
   - El ciclo de vida del software

1.1 Arquitectura básica de un computador

UNIVERSIDAD DE CANTABRIA

Un computador es una máquina que:

- acepta información de entrada
- la procesa ejecutando paso a paso una secuencia de instrucciones o programa
- y produce una información de salida

El computador está por tanto compuesto por un equipo electrónico (hardware) y un conjunto de programas (software)

El computador puede realizar dos tipos de instrucciones:

- acciones
- decisiones

Notas:

UNIVERSIDAD DE CANTABRIA

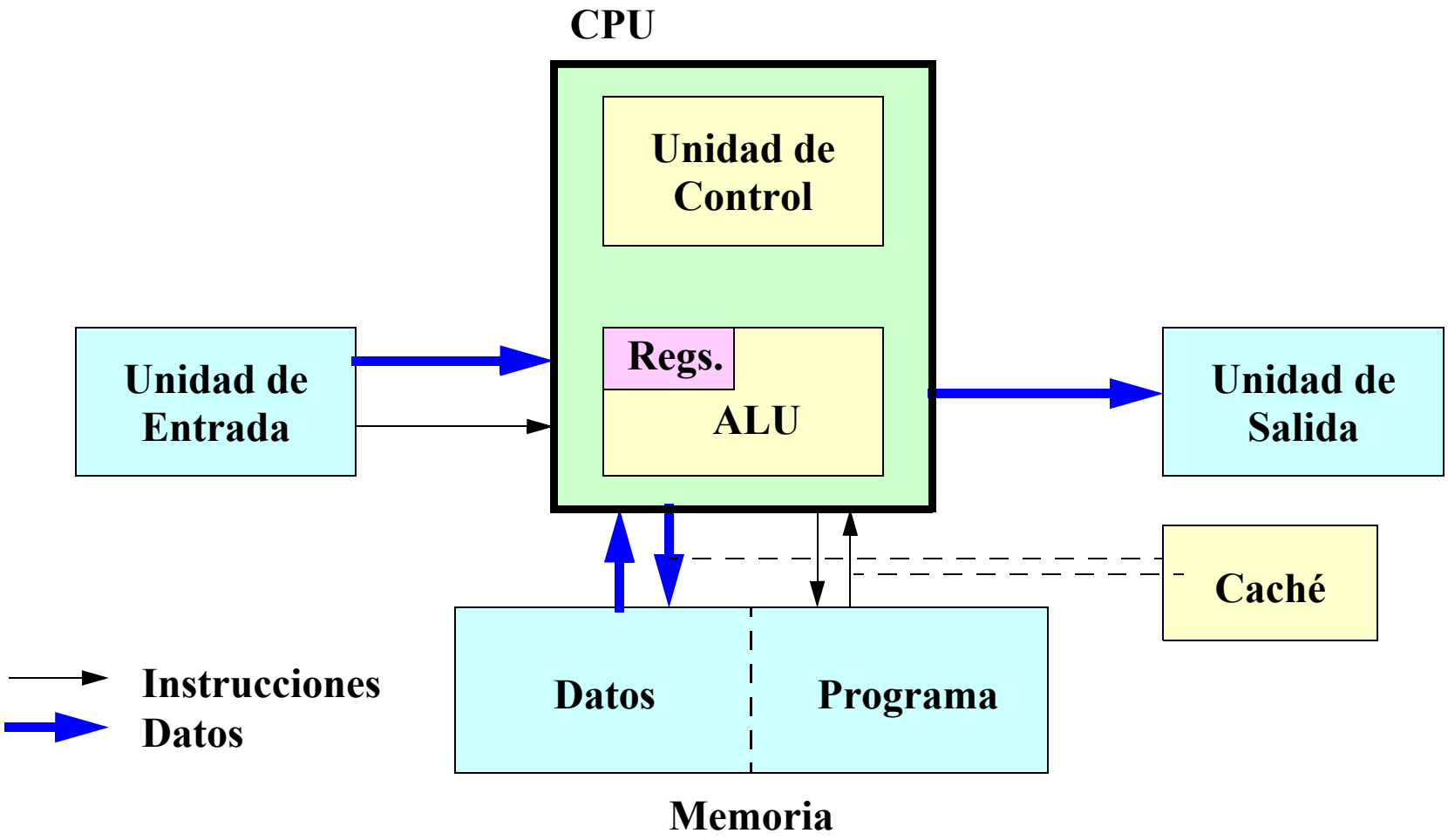
Un computador es una máquina que acepta información de entrada y la procesa ejecutando paso a paso una secuencia de instrucciones que posee almacenada, a la que llamamos programa, produciendo una información de salida.

Por ello, el computador está compuesto por un equipo electrónico, que denominamos hardware, y un conjunto de programas, que denominamos software.

Una de las características que distinguen al computador de una máquina automática es la capacidad de realizar dos tipos de instrucciones:

- **Acciones**. Tienen como consecuencia la modificación o intercambio de la información almacenada en el computador.
- **Decisiones**. Permiten comprobar el propio estado del computador y, en función del mismo, elegir uno de dos caminos alternativos.

Estructura de un computador



Notas:

UNIVERSIDAD DE CANTABRIA

Los computadores actuales son sistemas electrónicos digitales complejos capaces de realizar muchos millones de operaciones por segundo. Muchos computadores de propósito general presentan una estructura básica denominada “máquina de Von Neumann”, compuesta de cinco unidades funcionales:

- UNIDAD DE ENTRADA. Esta compuesta por aquellos elementos a través de los cuales el computador recibe información del exterior, tanto en lo que se refiere a datos como a programas. Sistemas típicos que forman la unidad de entrada son: teclados, digitalizadores, instrumentos de medida, etc.
- UNIDAD DE SALIDA. Compuesta por el conjunto de dispositivos destinados a proporcionar al exterior la información generada por el computador. Ejemplos típicos son: pantalla, impresora, plotter, máquina herramienta, etc.
- MEMORIA. Esta compuesta por el conjunto de circuitos y dispositivos electrónicos destinados a almacenar y retener los diferentes tipos de información que el computador utiliza o genera.
- UNIDAD DE CONTROL. Es la parte central del computador, encargada de interpretar las instrucciones del programa y ordenar su ejecución, generando las señales de control adecuadas que activan las diferentes unidades del computador.
- UNIDAD ARITMETICA Y LOGICA (ALU). Realiza las operaciones de transformación, comparación y transferencia de datos. La ALU y la unidad de control se integran, normalmente, en un sistema único denominado CPU (Central Processing Unit).

La memoria

UNIVERSIDAD DE CANTABRIA

Almacena dos tipos fundamentales de información:

- datos
  - numéricos
  - texto
- programas

En un computador suele haber varios tipos de memoria:

- registros de la CPU
- memoria de caché
  - con varios niveles
- memoria principal
- memoria secundaria o masiva

Notas:

UNIVERSIDAD DE CANTABRIA

Existen dos tipos básicos de información:

- Datos, que se utilizan o generan en la ejecución del programa. Pueden ser datos numéricos o datos con información alfanumérica (texto).
- Programa, constituida por la secuencia de instrucciones a ejecutar.

La memoria de un computador se puede clasificar en varios niveles, según su capacidad y tiempo de acceso:

- Registros de la CPU, internos a la misma, y de acceso muy rápido
- Memoria de caché: Situada entre la CPU y la memoria principal. Es una memoria electrónica de alta velocidad y poca capacidad. Se intenta colocar en ella los datos e instrucciones más frecuentemente usados por el computador.
- Memoria principal. Electrónica, directamente accesible en tiempos muy cortos, pudiendo a su vez ser memoria de lectura y escritura (RAM) o de solo lectura (ROM).
- Memoria masiva. Normalmente, en soportes magnéticos, con gran capacidad de almacenamiento pero con unos tiempos de acceso mucho mas altos que la anterior. Como ejemplo, los discos, diskettes, cintas, y dispositivos ópticos.

La unidad de memoria es un elemento fundamental de un computador. Su capacidad y tiempo de acceso fijan, en gran medida, la potencia del mismo.

Clasificación de los computadores por su capacidad

UNIVERSIDAD DE CANTABRIA

Categoría	Aplicación	Coste aproximado
Supercomputador	Cálculo numérico	> 1 M euros
“Mainframe”	Grandes bases de datos	0.1-1M euros
Servidor	Comunicaciones y bases de datos	3-100 K euro
Workstation	Diseño, ingeniería	5-30 K euros
PC	Oficinas, uso doméstico	300-5000 eur
Microcontrolador	Sistemas empotrados	> 1 euro

Notas:

UNIVERSIDAD DE CANTABRIA

El tipo de computador a emplear depende en gran medida de la aplicación a la que se destina. Existen varias categorías:

- **Supercomputador**. Aplicaciones científicas de cálculo numérico muy complejas. (IBM Roadrunner, IBM Blue Gene, Sun Ranger-SunBlade, Jaguar Cray XT4). Consultar: http://www.top500.org/
- **"Mainframe"**. Computador de propósito general grande, principalmente para aplicaciones de grandes bases de datos. (IBM System Z, Sun Fire E25K).
- **Servidor**. Computador de propósito general a mitad de camino entre un mainframe y una estación de trabajo. Da servicio a una red de computadores más pequeños. (IBM System P, Sun Spark Enterprise, Sun Blade).
- **Estación de Trabajo ("Workstation")**. Computador de propósito general, de bajo costo, generalmente con capacidad multiproceso. (IBM Intellistation Power, Sun Ultra 40)
- **Computador Personal**. Computador con un microprocesador como CPU, normalmente configurado para un aplicación específica. (PC, Apple Machintosh).
- **Microcontrolador**. Computador de propósito especial basado en un conjunto integrado de microprocesador, memoria y dispositivos de entrada/salida, y que forma parte de un sistema mayor, tal como una máquina, instrumento, electrodoméstico, etc.

1.2 El software del sistema

UNIVERSIDAD DE CANTABRIA

Es el software básico que se requiere para que el computador sea utilizable

Incluye:

- un sistema de arranque
  - carga el sistema operativo en la memoria
- un sistema operativo
- soporte para lenguajes de programación
- un entorno de desarrollo
- programas de aplicación
  - aplicaciones de propósito general

Las aplicaciones hechas a medida no son "software del sistema"

Notas:

UNIVERSIDAD DE CANTABRIA

Desde el punto de constitución física, el computador es una máquina compuesta de un entramado de circuitos electrónicos y dispositivos mecánicos de precisión. Sin embargo, el salto de máquina a computador es posible gracias a lo que constituye el **software**.

En el momento actual, un computador es un sistema electrónico muy complejo, capaz de realizar a muy alta velocidad una secuencia de operaciones, **de acuerdo con un programa previamente almacenado en sus elementos de memoria electrónica**.

El software del sistema proporciona un entorno que facilita la carga de programas en la memoria electrónica, así como la creación de programas nuevos.

Programación del computador

UNIVERSIDAD DE CANTABRIA

Las instrucciones de un programa son códigos numéricos almacenados en la memoria del computador

- la programación mediante códigos numéricos se conoce como *lenguaje máquina*
- es muy compleja

Por ello se necesitan lenguajes de programación más cercanos a los programadores

- y herramientas para convertir programas a lenguaje máquina

Notas:

UNIVERSIDAD DE CANTABRIA

Para programar y manejar un computador que aún no ha sido programado sería necesario conocer las operaciones básicas que puede realizar, así como la forma de codificarlas en la memoria electrónica del mismo. Esto es lo que se denomina la programación del computador en **lenguaje máquina**.

En la práctica, el lenguaje máquina resulta muy complicado para las personas. Se consigue una productividad mucho más alta con lenguajes de programación más cercanos a la persona, llamados lenguajes de alto nivel. Pero el computador lo único que entiende al final es lenguaje máquina, por lo que se necesitan traductores de lenguajes de programación de alto nivel a lenguaje máquina.

En una sección posterior ampliaremos la discusión sobre los lenguajes de programación.

El sistema operativo

UNIVERSIDAD DE CANTABRIA

Controla el uso por parte de los programas de aplicación de todos los recursos del computador: memoria, CPU, unidades de entrada y salida

Independiza al programa de aplicación del hardware

Proporciona comunicación con otros computadores

Ejemplos de sistemas operativos:

- MS-DOS: monoproceso, sin protección
- Windows 95/98/ME: multiproceso, semi-protegido, un solo usuario
- UNIX, Linux, Windows NT/2000/XP/Vista/Siete: multiproceso, protegido, múltiples usuarios (según versión)

Notas:

UNIVERSIDAD DE CANTABRIA

El **sistema operativo** está constituido por una serie de programas que permiten utilizar de una forma conceptual los diferentes elementos que constituyen los recursos del computador (pantalla, teclado, diskettes, discos, impresoras, etc.).

Así, mientras el sistema operativo presente el mismo modelo conceptual al usuario, el manejo de equipos basados en sistemas electrónicos totalmente diferentes parecerá que es el mismo para el usuario.

Por ejemplo un PC de IBM y uno de otro fabricante se manejan de igual modo y pueden ejecutar los mismos programas, no porque tengan el mismo diseño electrónico, sino porque el sistema operativo que utilizan presenta el mismo modelo conceptual (Windows Siete).

Más software del sistema

UNIVERSIDAD DE CANTABRIA

El entorno de desarrollo de programas suele constar de

- editores de texto
- herramientas CASE para análisis y diseño de programas
- depuradores
- herramientas de control de versiones

Los programas de aplicación son muy variados, dependiendo de la aplicación concreta del computador:

- procesador de textos
- bases de datos y hojas de cálculo
- navegador de red Internet (“browser”)
- programas de diseño gráfico (CAD), etc.

Notas:

UNIVERSIDAD DE CANTABRIA

El entorno de desarrollo permite la implementación de programas de computador, cubriendo idealmente todas las fases del ciclo de vida del software. Herramientas características de un entorno de desarrollo son:

- Editor de textos: permiten crear un texto a partir de un teclado. Los caracteres tecleados se añaden al texto y, además, hay órdenes de control para gestionar el aspecto y organización del texto. Normalmente todos los programas se escriben mediante un editor de texto
- Herramientas CASE para análisis y diseño de programas. Son herramientas avanzadas de ingeniería de software (CASE => Computer-Aided Software Engineering) que facilitan la labor de análisis y diseño del programa, previa a su codificación en un lenguaje de programación. Si no se dispone de estas herramientas el diseño puede hacerse manualmente, sobre papel.
- Depuradores: permiten ejecutar un programa en condiciones especiales que permiten su prueba. Permiten parar el programa en el punto deseado, consultar el estado de sus datos, continuarlo, etc.
- Herramientas de control de versiones. Son imprescindibles para gestionar los cambios incrementales en proyectos grandes de programación.

Programas de aplicación o expertos. Son programas desarrollados para resolver necesidades concretas dentro de un campo de interés humano que permite al usuario manejar un lenguaje muy próximo al campo especifico de que se trate. Programas de este tipo son, por ejemplo, los procesadores de texto, hojas de cálculo, bases de datos, simuladores, etc.

1.3. Lenguajes de programación

UNIVERSIDAD DE CANTABRIA

Recordamos que las instrucciones de un programa son códigos numéricos

Ejemplo de lenguaje máquina para el microprocesador 68000: suma de dos enteros:

Dirección	Código Binario	Código Ensamblador	Alto Nivel
$1000	0011101000111000	MOVE.W $1200,D5	Z=X+Y
$1002	0001001000000000		
$1004	1101101001111000	ADD.W $1202,D5	
$1006	0001001000000010		
$1008	0011000111000101	MOVE.W $D5,$1204	
$100A	0001001000000100		

Notas:

UNIVERSIDAD DE CANTABRIA

El ejemplo de arriba hace las siguientes operaciones:

- Mueve número que está en la posición de memoria $1200 al registro D5
- Suma el número que está en la posición de memoria $1202 al registro D5
- Mueve el resultado de la suma, contenido en el registro D5, a la posición de memoria $1204

Como puede verse la programación en lenguaje máquina es muy poco entendible para las personas.

Lenguajes de alto y bajo nivel

UNIVERSIDAD DE CANTABRIA

Necesitamos escribir programas en un lenguaje más cómodo para los humanos

- lenguaje de *bajo nivel* o ensamblador
  - cada instrucción corresponde a una instrucción de lenguaje máquina
  - es dependiente de la máquina
  - teóricamente más eficientes
- lenguajes de *alto nivel*
  - instrucciones más abstractas y avanzadas
  - son lenguajes independientes de la máquina
  - en la práctica, mucho más productivos

Notas:

UNIVERSIDAD DE CANTABRIA

Para hacer accesible la programación de un computador a cualquier persona, existen **lenguajes de programación** que tienen como función presentar al usuario el computador de acuerdo con un modelo abstracto (**informático**) sencillo e independiente de su estructura electrónica interna:

Los lenguajes tienen dos categorías

- **Lenguaje ensamblador o de bajo nivel**. Cada instrucción de lenguaje ensamblador se corresponde con una instrucción de lenguaje máquina, pero en lugar de codificarse mediante números se codifica mediante símbolos alfanuméricos, más fáciles de recordar. En teoría se puede conseguir más eficiencia, pues podemos usar toda la potencia ofrecida por la máquina. En la práctica, programar en lenguaje ensamblador es tedioso, difícil, con mucha facilidad para cometer errores y, por tanto, poco productivo
- **Lenguajes de alto nivel**. Permiten programar utilizando un lenguaje más próximo al humano, e independiente de la máquina. Ejemplos de este tipo de lenguajes son el Java, FORTRAN, C, BASIC, Pascal, Ada, etc. Los lenguajes de alto nivel producen un código un poco menos eficiente que el ensamblador, pero más sencillo de escribir y mantener, con menos errores, y mucho más productivo.

Ensambladores, compiladores e intérpretes

UNIVERSIDAD DE CANTABRIA

Son programas que traducen un programa de aplicación escrito en un lenguaje de programación, a un programa en lenguaje máquina:

- **lenguaje ensamblador: se traduce mediante un programa ensamblador**
- **lenguajes de alto nivel: se traducen mediante compiladores e intérpretes**
  - **los compiladores traducen el programa de aplicación antes de que éste se ejecute**
  - **los intérpretes van traduciendo el programa de aplicación a medida que se va ejecutando**

Notas:

UNIVERSIDAD DE CANTABRIA

El vendedor de un computador, acompaña el equipo físico con un conjunto de programas de ayuda, desarrollados por él o por personas especializadas, que tienen como función presentar al usuario el computador de acuerdo con un modelo abstracto (**informático**) sencillo e independiente de su estructura electrónica interna:

- ensambladores, compiladores e intérpretes
- sistema operativo
- entorno de desarrollo
- programas de aplicación

Los ensambladores, compiladores e intérpretes traducen un programa escrito en un lenguaje más o menos cómodo para el usuario, a lenguaje máquina. Según el lenguaje utilizado, las herramientas son:

- **Lenguajes de bajo nivel:** se utiliza un programa ensamblador, que traduce los símbolos alfanuméricos a código máquina, por medio de algoritmos muy simples.
- **Lenguajes de alto nivel**. La traducción a lenguaje máquina se hace mediante compiladores (que traducen el programa escrito en lenguaje de alto nivel de forma completa antes de su ejecución) e intérpretes (que traducen cada instrucción mientras se ejecuta el programa). Los intérpretes son más lentos en la ejecución, ya que tienen que hacer el trabajo de traducción cada vez que se ejecuta el programa. Hoy en día es más habitual usar compiladores.

Lenguajes de alto nivel

UNIVERSIDAD DE CANTABRIA

Los lenguajes de programación de alto nivel son:

- una solución intermedia entre los lenguajes naturales y el lenguaje máquina
- son precisos, es decir, no ambiguos
- son relativamente simples (y por tanto poco expresivos y difíciles de usar)

Ejemplos de lenguajes de programación:

- *Fortran*: 1956, para cálculo científico
  - estándares: 1966, 1977, 1990, 1997, 2003
- *Cobol*: 1960, para aplicaciones de gestión
  - estándar actual: 2002

Notas:

UNIVERSIDAD DE CANTABRIA

Los lenguajes de programación de alto nivel han sido definidos como una solución intermedia entre los **lenguajes naturales humanos** y los lenguajes **máquina de los computadores**. Están bien definidos, en el sentido de que la tarea que se puede expresar con ellos no es ambigua y por lo tanto pueden ser traducidos en un programa máquina concreto, de forma automatizada y por el propio (aunque no necesariamente) computador que va a realizar la tarea.

Existen muchos lenguajes de alto nivel de propósito general, sus principales diferencias se encuentran en que poseen un conjunto de órdenes más adecuado para expresar tareas de un tipo concreto de problema o porque corresponden a distintos niveles de evolución de los ordenadores:

FORTRAN (FORmula TRANslation). Su nombre evidencia la orientación matemática de uno de los lenguajes de alto nivel más antiguos, que aún perduran. J. Backus lo desarrolló en 1956. Aunque ha perdido terreno frente a los lenguajes más modernos, todavía es ampliamente utilizado en aplicaciones científicas de grandes cálculos numéricos, porque probablemente, es el lenguaje con mayor número de librerías, desarrolladas y comprobadas por mucha gente, a lo largo de su historia.

COBOL (COmmon Business Oriented Language). Se trata del lenguaje que ha alcanzado una mayor resonancia en las tareas de gestión. Su desarrollo fue promovido por el por el Departamento de Defensa de EEUU, en 1960. Como resultado de su reciente expansión al campo de las estaciones de trabajo (workstations) el lenguaje ha sufrido muchas extensiones.

Ejemplos de lenguajes (cont.)

UNIVERSIDAD DE CANTABRIA

- *Lisp*: 1959, para inteligencia artificial
  - estandarizado por ANSI en 1994 (common LISP)
  - estandarizado por ISO en 1997, actualizado en 2007 (ISLISP)
- *Basic*: 1964, para docencia, interpretado
  - Visual Basic 9.0, 2007 (Microsoft)
- *Pascal*: 1969, para docencia, programación estructurada
- *C*: 1972, para programación del software del sistema
  - estandarizado en 1990, y 1999
- *Ada*: 1983, para propósito general, incluyendo sistemas de tiempo real
  - estandarizado en 1983

Notas:

UNIVERSIDAD DE CANTABRIA

LISP (LISt Processing). El Massachusetts Institute of Technology creó, en 1959 este lenguaje de alto nivel orientado a aplicaciones de inteligencia artificial. La programación de procesos recurrentes (edificados sobre datos procesados en los pasos anteriores) es uno de los puntos fuertes del LISP.

BASIC (Beginners All-purpose Symbolic Instruction Code). Nació entre 1964 y 1965 en el Dartmouth College como una herramienta para la enseñanza. Con el tiempo han ido proliferando los dialectos y versiones, hasta el punto de que era raro el fabricante que no desarrollaba un dialecto para sus propios equipos. Fue muy popular por su sencillez, pero tiene carencias importantes.

PASCAL (En honor del matemático francés Blaise Pascal). Es un lenguaje de programación desarrollado por el profesor Nicklaus Wirth, en 1969, en el Instituto Federal de Tecnología de Zurich partiendo de los fundamentos del ALGOL. Fue uno de los primeros lenguaje que incorporaron los conceptos de programación estructurada. Aunque fue muy popular, la dificultad para partir el programa en módulos y la falta de estandarización han hecho decaer su uso.

C. Es un lenguaje de programación desarrollado por la Bell Laboratories, en principio para trabajar con el sistema operativo UNIX. Quizás por ello, sigue siendo uno de los lenguajes más populares hoy en día. Es un lenguaje que, al mismo tiempo que permite una programación en alto nivel, permite una gran aproximación a la máquina. Muchos lo consideran un lenguaje intermedio entre alto y bajo nivel. Como estos últimos, presenta alta eficiencia y escasa fiabilidad. Es fácil cometer errores en C.

Ejemplos de lenguajes (cont.)

UNIVERSIDAD DE CANTABRIA

- *Smaltalk*: 1980, para programación orientada a objetos
  - estandarizado en 1998
- *C++*: 1987, extensión mejorada del C que incorpora programación orientada a objetos
  - estandarizado en 1998, corregido en 2003
- *Java*: 1995, para programación orientada a objetos en sistemas distribuidos (red Internet)
  - versión actual: Java 6 (2006)
  - ofrecido como software libre en 2006
- *Ada 95, Ada 2005*: versiones mejoradas del anterior, incluyendo programación orientada a objetos
  - estandarizado en 1995 y luego en 2005

Notas:

UNIVERSIDAD DE CANTABRIA

SMALTALK: Lenguaje de programación orientada a objetos puro. Es muy ineficiente con respecto a lenguajes procedurales como el C o el Ada, pero es cómodo de usar y de programar en él.

JAVA: Lenguaje derivado del C en cuanto a sintaxis, pero más parecido al Ada 95 en cuanto a las comprobaciones que hace el compilador y soporte de la programación concurrente. Está pensado para su ejecución en sistemas distribuidos (internet). Existe un código intermedio, bien definido, que puede intercambiarse entre computadores diferentes para luego ser traducido y ejecutado.

C++: Extensión del lenguaje C que mejora algunos de sus inconvenientes, y añade construcciones de programación orientada a objetos. Entre las mejoras destacan una mayor comprobación de los tipos de datos por parte del compilador, las excepciones, y las plantillas genéricas.

ADA (En honor de Lady Augusta ADA Byron). El ADA es un lenguaje inspirado en el PASCAL, que fue promovido por el Departamento de Defensa de Los EEUU. El objetivo de su desarrollo era conseguir un lenguaje con posibilidades de convertirse en un estándar universal y que facilitara la ingeniería de software y el mantenimiento de los programas. Entre sus campos de aplicación se incluyen los sistemas de tiempo real y los sistemas de alta integridad. El 1995 se revisó el lenguaje para mejorarlo y para añadirle construcciones de programación orientada al objeto. En 2005 se finalizó una nueva versión que añade mejoras en el uso de objetos y nuevas características para la programación de tiempo real.

Ejemplos de lenguajes (cont.)

UNIVERSIDAD DE CANTABRIA

Otros lenguajes de programación:

- *C#* (C sharp): orientado a objetos y orientado a componentes; sintaxis basada en C++
  - introducido por Microsoft en 2000, estandarizado por ISO
  - versión actual: 3.0 (2007)

Ejemplos de lenguajes (cont.)

UNIVERSIDAD DE CANTABRIA

Lenguajes de “scripts”:

- *PHP*: desarrollado para hacer páginas Web dinámicas, soportado por la mayoría de servidores Web
  - desarrollado en 1995, versión actual 5 (2008)
- *Perl*: interpretado, con tipos dinámicos, no es orientado a objetos. Tiene facilidades de manipulación de textos
  - desarrollado en 1987, versión actual 5.10 (2007)
- *Python*: habitualmente interpretado, orientado a objetos, con tipos dinámicos; hace énfasis en la legibilidad
  - es de finales de los 80; versión actual 3.0 (2008)
- *JavaScript*: interpretado, orientado a objetos, soportado por muchos navegadores Web. Sintaxis inspirada en Java
  - creado en 1995, propiedad de Sun Microsystems

Ejemplos de lenguajes (cont.)

UNIVERSIDAD DE CANTABRIA

Lenguajes de “scripts” (continuación):

- ***Ruby***: Lenguaje de *“scripts”*, interpretado, orientado a objetos, con tipos dinámicos. Sintaxis inspirada en Perl y Smalltalk
  - creado en 1993, no estandarizado

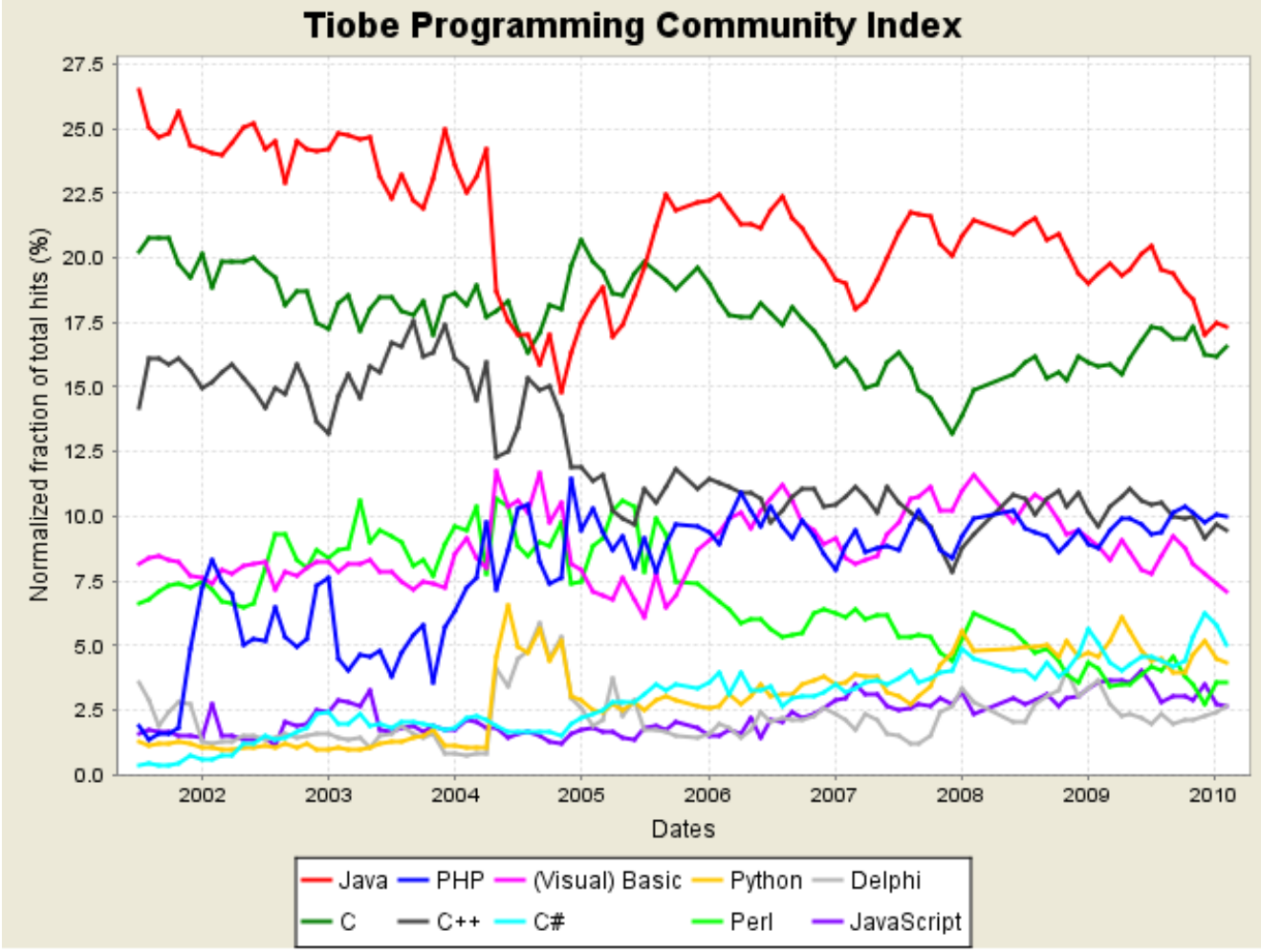
Notas:

UNIVERSIDAD DE CANTABRIA

Un lenguaje de “scripts” es un lenguaje de programación para controlar la ejecución de otras aplicaciones. Suelen ser interpretados.

Un lenguaje con tipos dinámicos es aquel en el que la comprobación del tipo de dato que se está utilizando se hace durante la ejecución, en lugar de hacerse de manera estática durante la compilación.

Ranking de lenguajes de programación



Notas:

UNIVERSIDAD DE CANTABRIA

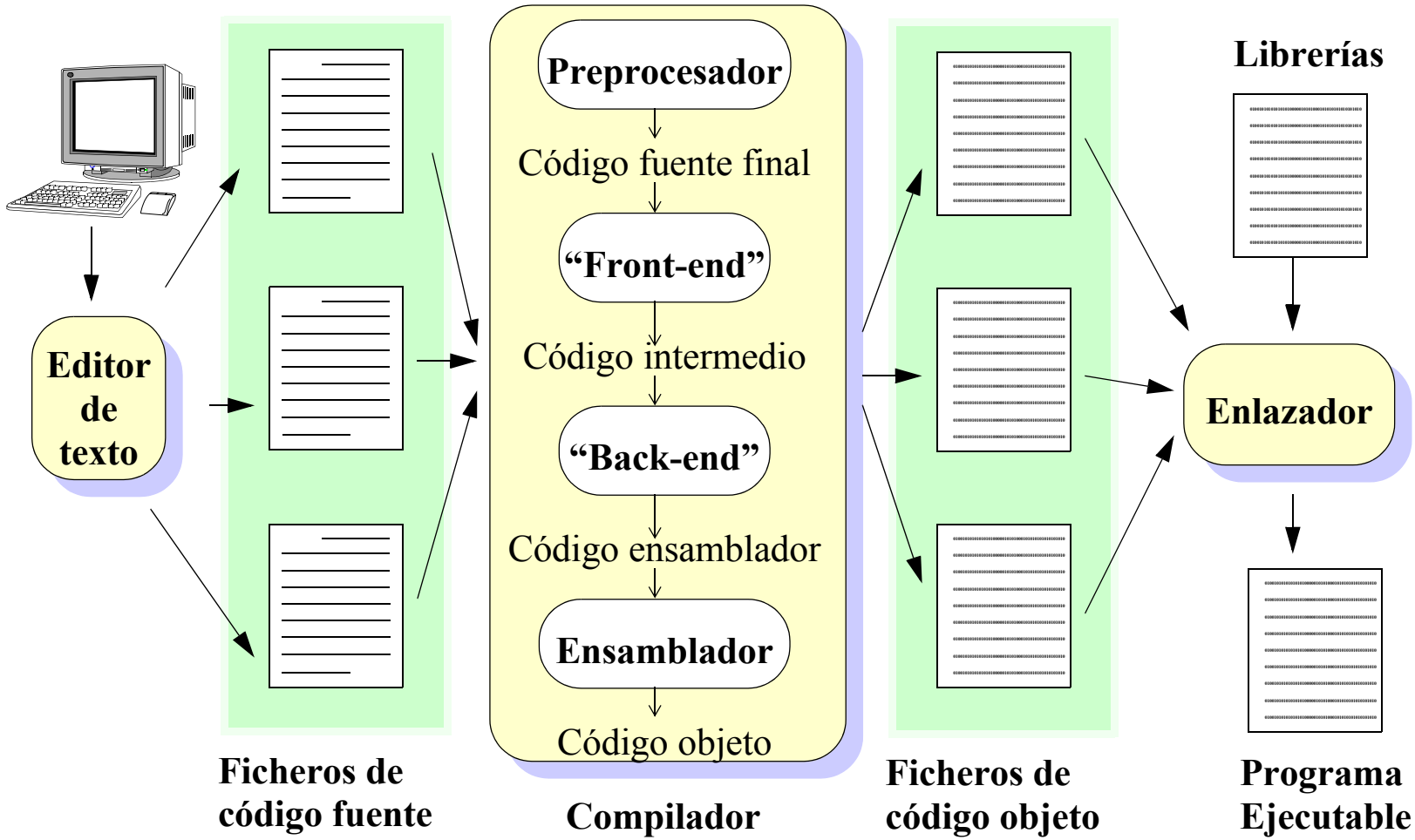
Fuente:

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Fecha: Feb 2010

El lenguaje Ada ocupa el lugar número 30 en este ranking, una posición discreta que muestra que este lenguaje es utilizado fundamentalmente para aplicaciones especiales en las que se requiere un alto grado de fiabilidad.

1.4. El proceso de compilación



Notas:

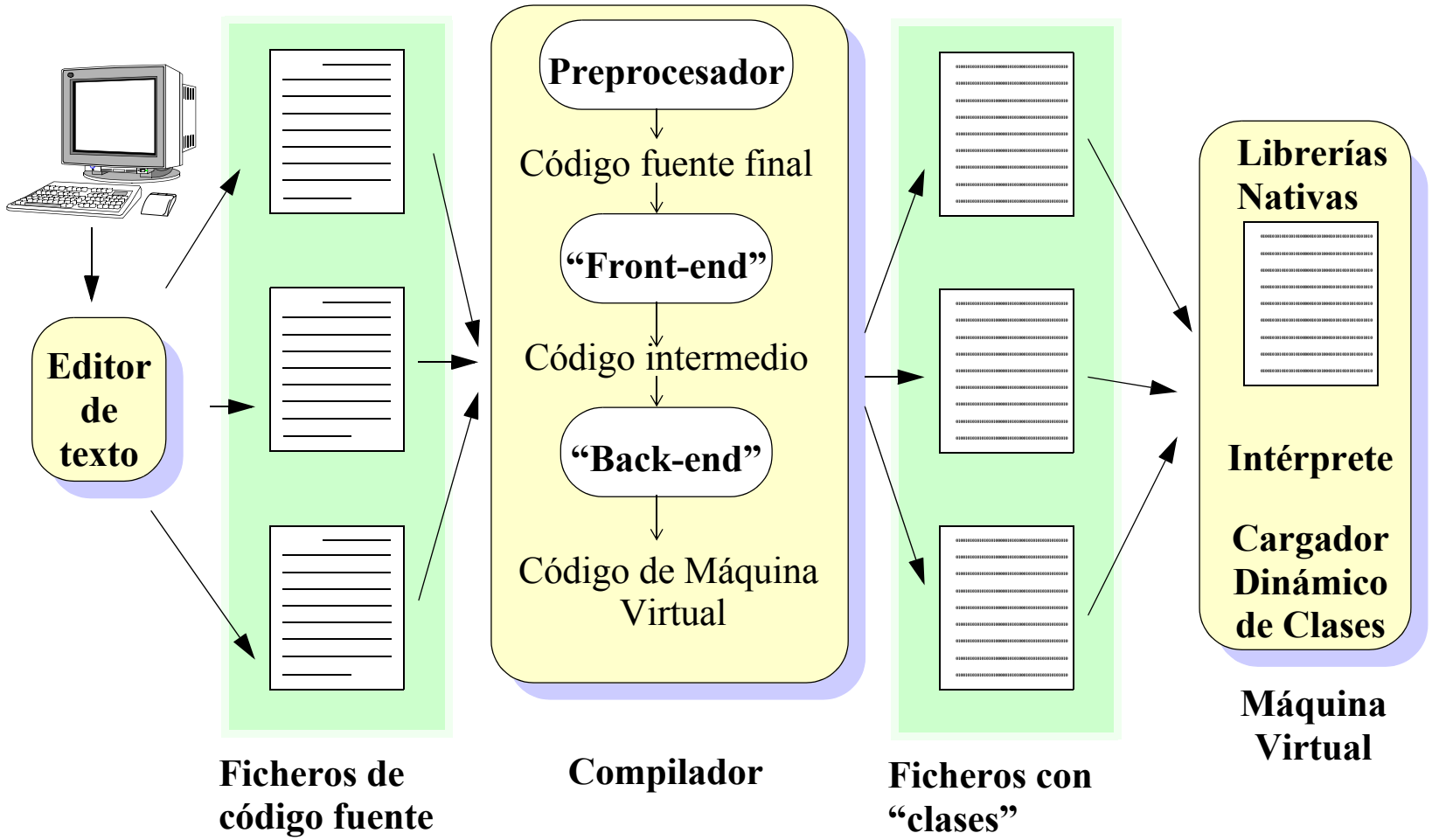
UNIVERSIDAD DE CANTABRIA

Un compilador puede definirse como una herramienta automática de traducción que lee un programa escrito en un lenguaje (el lenguaje fuente) y lo traduce a un programa equivalente en otro lenguaje (lenguaje objeto). En el proceso de traducción el compilador notifica al usuario de la presencia de errores en el programa fuente.

La variedad de compiladores que pueden aparecer es muy alta. Existen miles de lenguajes fuente. Igualmente ocurre con los lenguajes objeto: pueden ser otros lenguajes de programación, o el lenguaje máquina de cualquier computador entre un microprocesador o un supercomputador. En cualquier caso las tareas que debe realizar son las mismas.

Además del compilador también son necesarios otros programas para crear un programa ejecutable: el preprocesador, el ensamblador, el enlazador, y el cargador. En la figura de arriba se muestra un proceso de compilación típico.

Compilación para máquina virtual



Notas:

UNIVERSIDAD DE CANTABRIA

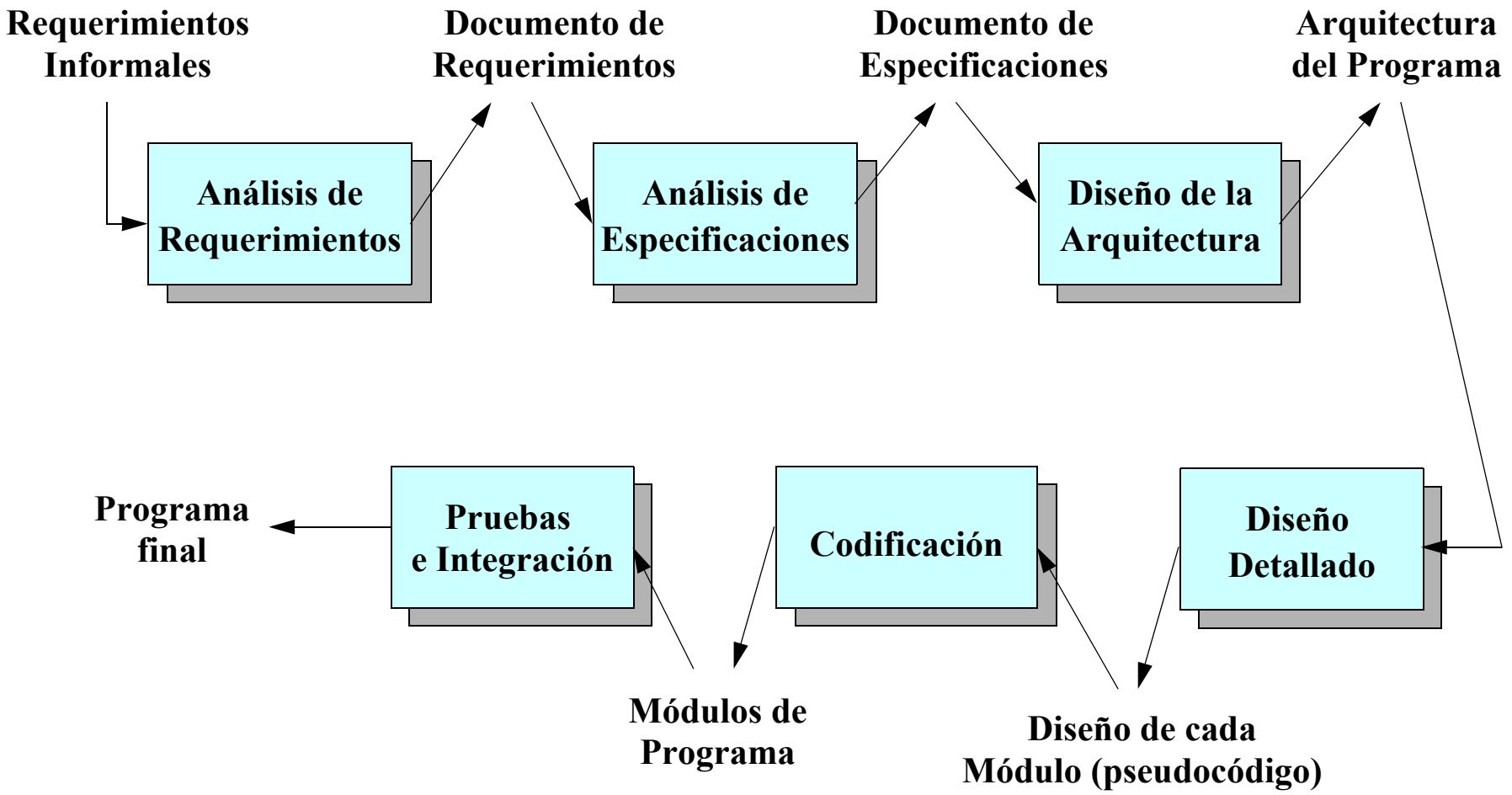
La arquitectura del entorno de ejecución de programas Java se basa en una máquina virtual, que se ejecuta en el computador para interpretar las instrucciones del programa del usuario descritas en un código intermedio especial, llamado código de máquina virtual Java (Java Byte Code).

La idea principal de esta arquitectura es la de “Escribir una vez, ejecutar en cualquier sitio”. El compilador de Java no genera código máquina de un computador concreto, sino un código especial, que luego es interpretado por otro programa, llamado máquina virtual, que existe en cada computador en el que se desea ejecutar el programa Java. De este modo, un programa Java se puede ejecutar indistintamente en cualquier computador que disponga de esa máquina virtual, sin necesidad de recompilarlo.

Adicionalmente, en la arquitectura Java los programas no se enlazan antes de su ejecución, sino que se utiliza un enlazado dinámico. Cuando se hace una llamada a una operación de un módulo (clase) que no está cargado en la máquina virtual, ésta se encarga de buscar ese módulo y cargarlo en ese momento en la máquina virtual. Desde el programa del usuario se pueden utilizar operaciones “nativas”, suministradas por la máquina virtual, escritas generalmente en código máquina, y que pueden acceder a los dispositivos hardware del computador.

El resultado da lugar a programas muy portables, muy dinámicos, aunque poco eficientes. Existen también compiladores Java que traducen directamente a lenguaje máquina, así como compiladores en otros lenguajes (Ada) que traducen a código de máquina virtual Java.

1.5. El ciclo de vida del software



Notas:

UNIVERSIDAD DE CANTABRIA

Los pasos que se siguen generalmente a la hora de desarrollar un programa son los siguientes:

- *Análisis de requerimientos*: Se define el problema a resolver y todos los objetivos que se pretenden, pero sin indicar la forma en la que se resuelve.
- *Especificación*: Se determina la forma en la que se resolverá el problema, pero sin entrar aún en su implementación informática. Se determina asimismo la interfaz con el usuario.
- *Diseño del programa:* Se divide el problema en módulos, se especifica lo que hace cada módulo, así como las interfaces de cada uno de ellos.
- *Diseño detallado de los módulos*: Para cada módulo se diseñan detalladamente las estructuras de datos y los algoritmos a emplear, normalmente descritos mediante pseudocódigo.
- *Codificación*: Se escribe el programa en el lenguaje de programación elegido.
- *Pruebas de módulos*: Se prueban los módulos del programa aisladamente y se corrigen los fallos hasta conseguir un funcionamiento correcto.
- *Integración y Prueba de sistema*: Se unen todos los módulos, y se prueba el funcionamiento del programa completo.

Distribución del esfuerzo de la actividad software (sin tener en cuenta el mantenimiento):

- Análisis y Diseño : 38%
- Codificación : 20%
- Test e integración: 42%