DAM/DAW

PROGRAMACIÓN

01.1

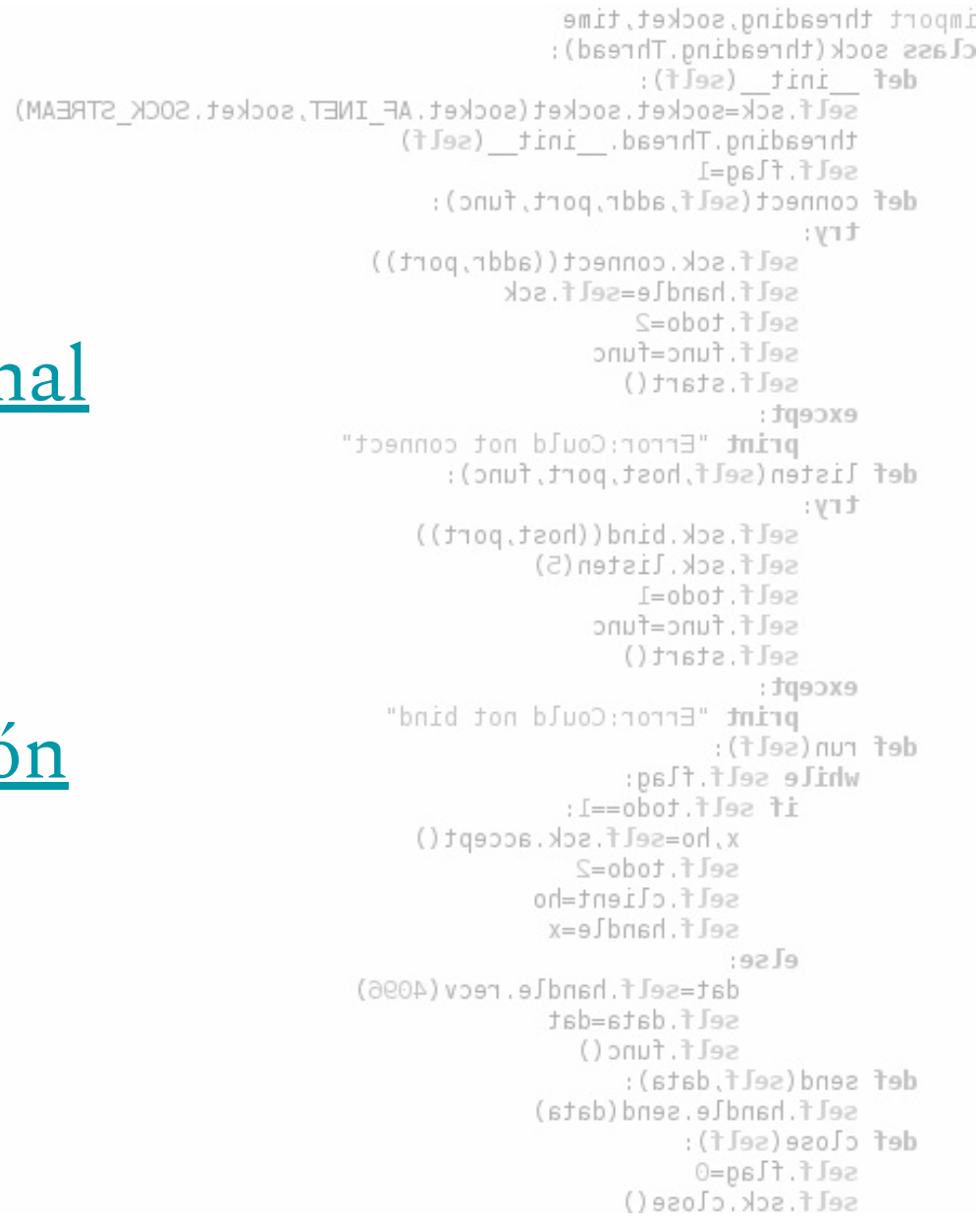
Introducción a la programación

ver: 3.3

fran@iessanclemente.net

Indice

- Introducción al curso
- Pensamiento computacional
- Los computadores
- Lenguajes de programación
- Desarrollo de software



Introducción al curso (I)

Objetivos

- Comprender la lógica de pensamiento computacional (*"Computational Thinking"*)
- Desarrollar la capacidad de formular soluciones computacionales para la resolución de problemas
- Instruir ("programar") la computadora para que realice aquellas tareas y funciones que precisemos

Introducción al curso (II)

Temas principales

- Representación del conocimiento mediante estructuras de datos y sus relaciones. Abstracción de procesos y tipos de datos
- Diseño y análisis de algoritmos para la resolución de problemas
- Organización y modularización de sistemas mediante el uso de clases de objetos y métodos
- Diseño, desarrollo y depurado de programas mediante el empleo de lenguajes de programación
- Currículo módulo de Programación de DAW/DAM:
  - https://www.edu.xunta.es/fp/familias-profesionais

Introducción al curso (y III)

Desarrollo del curso

- El curso se divide en tres partes temáticas correspondientes a las tres evaluaciones del curso:
  - 1ª Ev. Programación estructurada. Algoritmos
  - 2ª Ev. Programación orientada a objetos. Colecciones
  - 3ª Ev. Ficheros. Interfaces de usuario. Acceso a datos
- La evaluación de cada parte se realizará mediante una prueba teórico-práctica. Para superar el curso debe superarse cada una de las tres partes. Las partes no superadas podrán recuperarse de forma individual mediante la realización de un examen final
- Se proporcionarán a través de la plataforma web del centro todos los materiales necesarios para el desarrollo del curso. Puntualmente se podría proporcionar algún material en inglés
- Tutoría: ---

Pensamiento Computacional (I)

¿Qué puede hacer un computador?

Básicamente, dos cosas (aunque extremadamente bien):

- Realizar cálculos
  - un computador de escritorio típico puede ejecutar del orden de mil millones de instrucciones[1] por segundo!
- Recordar los resultados de esos cálculos
  - la capacidad de almacenamiento de los computadores actuales es de cientos de gigabytes. Un computador de escritorio típico podría almacenar del orden de un millón de libros!

[1] operaciones primitivas o *built-in*: operaciones aritméticas y lógicas simples; suma, resta, multiplicación, división, comparaciones,...

Pensamiento Computacional (II)

¿Es suficiente la potencia bruta?

- Un ejemplo: búsquedas en la WWW
  - Se estima que existen 2×10^9 webs, sobre 60×10^9 páginas...
  - Suponiendo 1000 palabras/pág; 10 op/encontrar palabra...
  - Una computadora típica necesitaría una semana!
    - Google procesa 7×10^9 consultas/día (15% nuevas)
- Existencia de problemas complejos o irresolubles...
  - Modelos atmosféricos a escala local, descifrado por fuerza bruta,...
  - Ejemplo: romper por fuerza bruta AES-128
    - Supercomputador Fugaku (#1 TOP500 2020); 513 PFlop/s
    - Suponiendo chequear 1 clave/1 Flop; 2^{128} claves
      - Necesitaría 21 billones de años![1]
- Necesitamos diseñar buenos algoritmos de computación!

[1] *Las leyes de la termodinámica ponen límites a un ataque por fuerza bruta a una clave de 256 bits*

Pensamiento Computacional (III)

Tipos de Conocimiento

- ¿Cómo hacemos para que el computador resuelva problemas por nosotros? ¿Qué conocimiento del problema necesita? ¿Cómo expresamos ese conocimiento?
- Conocimiento declarativo ("Saber qué")
  - Proposiciones que manifiestan información, hechos, ideas.
    - "El coche tiene la rueda trasera derecha pinchada"
- Conocimiento procedimental ("Saber cómo")
  - Procedimiento o "receta" para la resolución de un problema.
    - "Abrir maletero, extraer rueda de repuesto y herramientas"
    - "Localizar rueda pinchada y aflojar tuercas de sujección"
    - "Colocar el gato bajo el coche e izarlo hasta que la rueda esté en el aire"
    - "Extaer los tornillos de sujección de la rueda"...

Pensamiento Computacional (IV)

Un ejemplo numérico: cálculo de raíces cuadradas

Supón que queremos calcular la raíz cuadrada de un número X (ej: 16):

- "la raíz cuadrada de un número X es otro número R tal que R*R = X"

  Es declarativo. Nos dice qué es pero no cómo se resuelve

- Otra aproximación: receta[1] para deducir la raíz cuadrada de un número:

  1) Elegir al azar una estimación del resultado, R
  2) Si R*R suficientemente próximo a X, indicar R como solución y finalizar
  3) Si no, realizar una nueva estimación con la media de R y X/R
  4) Con la nueva estimación, volver a 2) y repetir el proceso

R	R*R (= 16)	X/R	(R + X/R)/2
3	9	5.33	4.1667
4.1667	17.36	3.84	4.0033
4.0033	16.026	3.997	4.0001

[1] *Herón de Alejandría*, ingeniero y matemático griego, s. I d.C.

Pensamiento Computacional (V)

Así que, estas recetas de resolución de problemas, se componen de:

- una secuencia ordenada de pasos simples
- un control de flujo que indica cuándo ejecutar cada paso
- una condición que indica cuándo detener el proceso

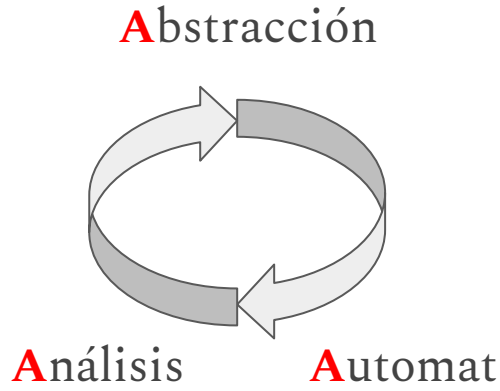
Conjunto bien definido, finito y ordenado de instrucciones que permite obtener una solución a partir de un estado inicial y un conjunto de datos de entrada. Para la misma entrada obtenemos siempre el mismo resultado.

[1] *Abu Abdallah Muḥammad ibn Mūsā* ***al-Jwārizmī***, matemático, astrónomo y geógrafo persa, s. IX-X d.C.

Pensamiento Computacional (y VI)

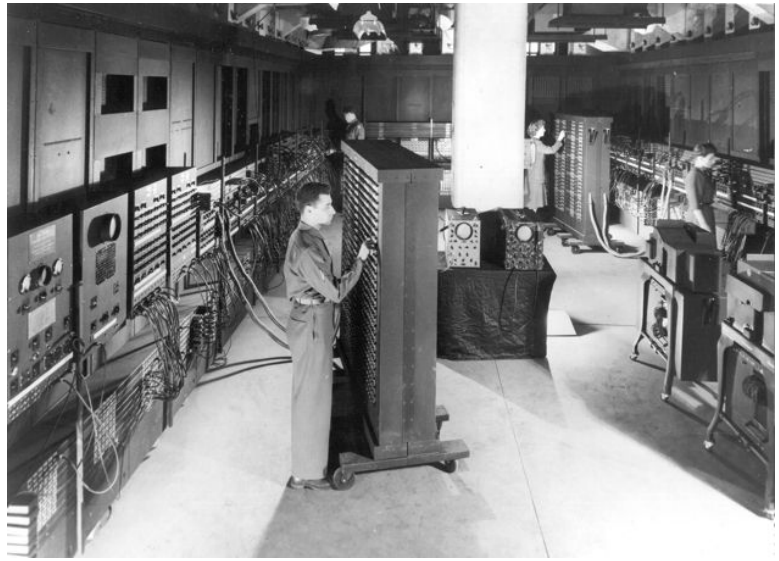
Definimos pensamiento computacional como...

- Se define como el proceso de pensamiento asociado a la formulación de un problema y la obtención de su solución de modo que una computadora (humano o máquina) pueda resolverlo
- Incluye las siguientes etapas en la resolución de problemas:
  - Descomposición del problema en unidades de menor complejidad
  - Reconocimiento de patrones y similitudes con problemas análogos
  - Abstracción mediante la identificación de la información relevante
  - Diseño de algoritmos de resolución
- Se suele caracterizar como un proceso iterativo de tres etapas:
  - Abstracción. Formulación del problema
  - Automatización. Implementación de algoritmos
  - Análisis. Ejecución y evaluación de resultados



Los computadores (I)

- Los primeros computadores se basaban en una arquitectura de programa fijo, de forma que su diseño quedaba determinado por la finalidad última de la misma (ej: Bomba de Alan Turing).
- Suele considerarse a ENIAC (*Electronic Numeric Integrator and Calculator*), financiada por el U.S. Navy y desarrollada durante la II Guerra Mundial en la Moore School de la Universidad de Pennsylvania, como el primer computador electrónico de propósito general.
- Utilizada para el cálculo de tablas de fuego de artillería, la programación se realizaba manualmente conectando cables e inicializando conmutadores. Los datos se introducían mediante tarjetas perforadas

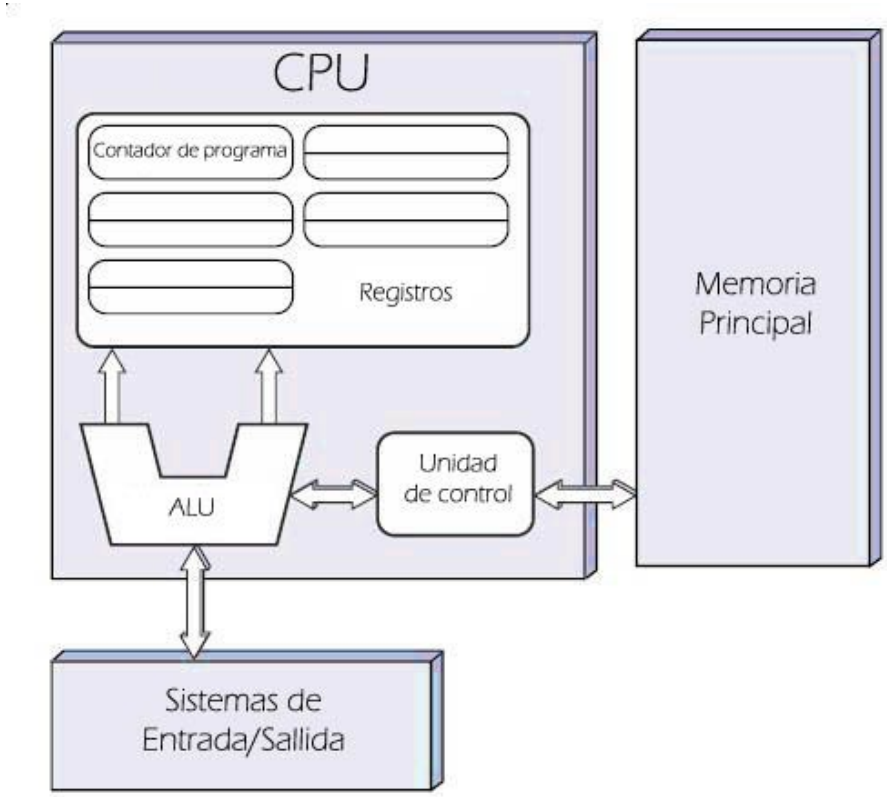


Los computadores (II)

Computadores de programa almacenado

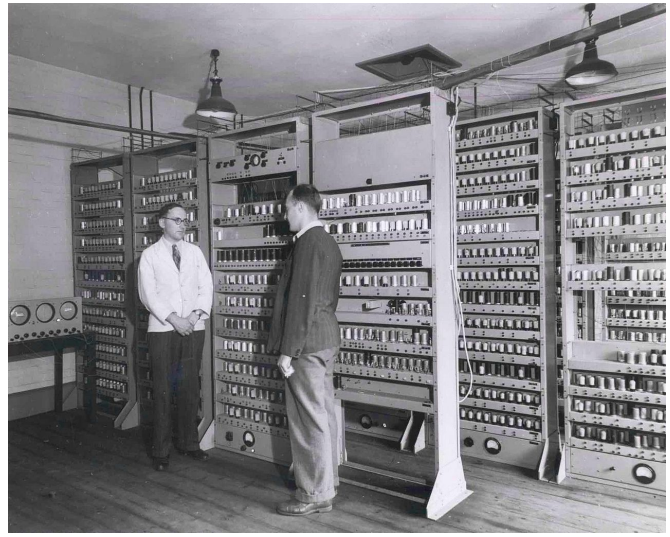
- En 1944, Eckert y Mauchly, creadores de ENIAC, incorporaron en el grupo de trabajo al matemático John von Neumann con la idea de mejorar la introducción y almacenamiento de los programas.

- En 1945 describen la arquitectura de un nuevo computador, EDVAC, que consta de una CPU formada por una ALU, una UC, registros PC (program counter), IR (instruction register),... y una memoria donde se almacenan tanto datos como programas. Es lo que conocemos como arquitectura von Neumann de computador de programa almacenado



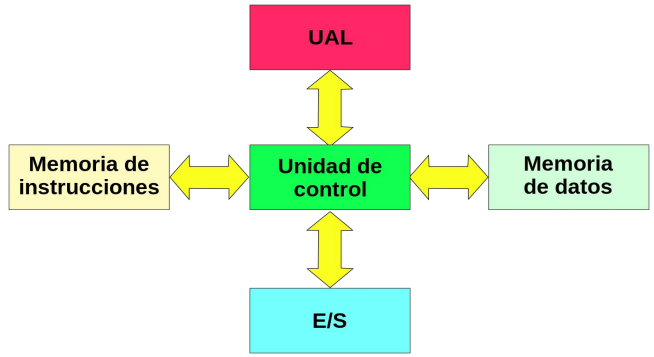
Los computadores (III)

- En 1949, Maurice Wilkes, U. de Cambridge, finalizó el desarrollo de EDSAC, considerado el primer computador de programa almacenado operativo a escala completa

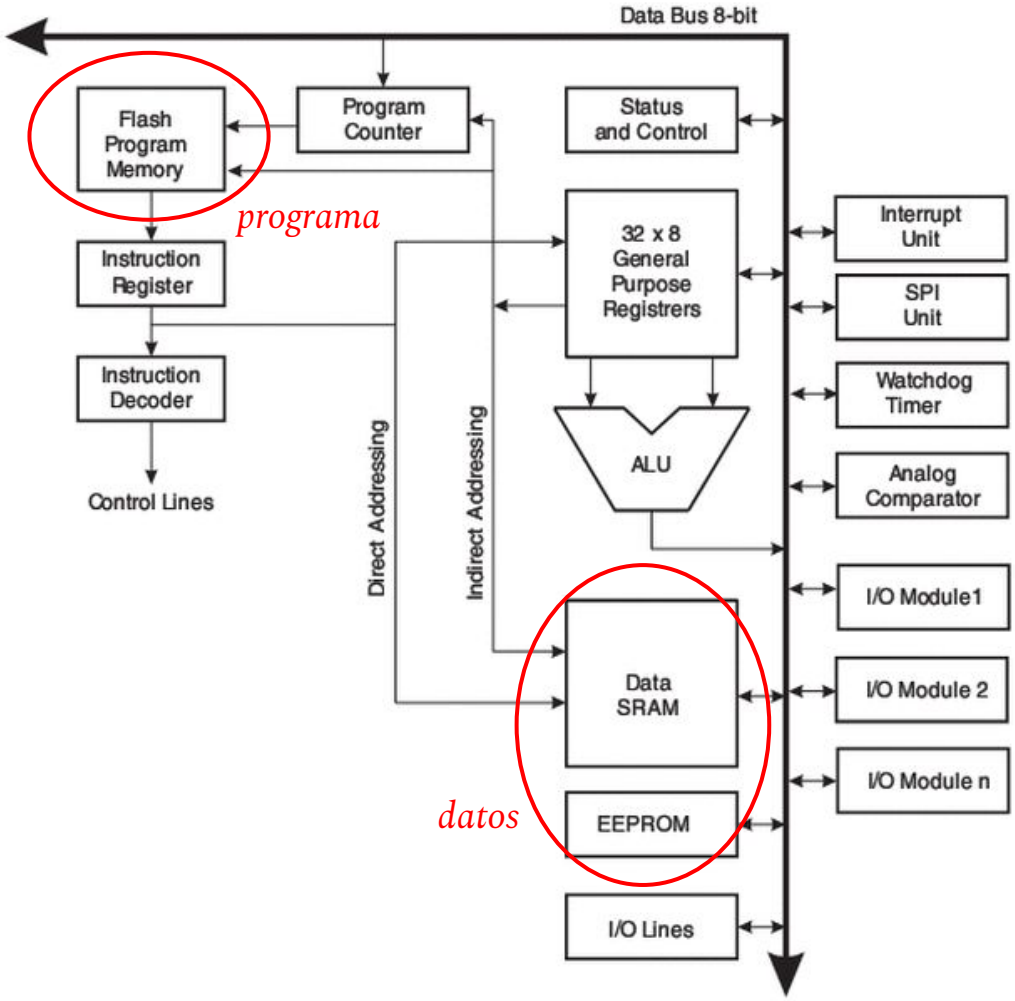


- En contraposición a la arquitectura von Neumann, la arquitectura Harvard define memorias y buses de instrucciones y datos diferenciados (IBM Harvard Mark I, 1944).

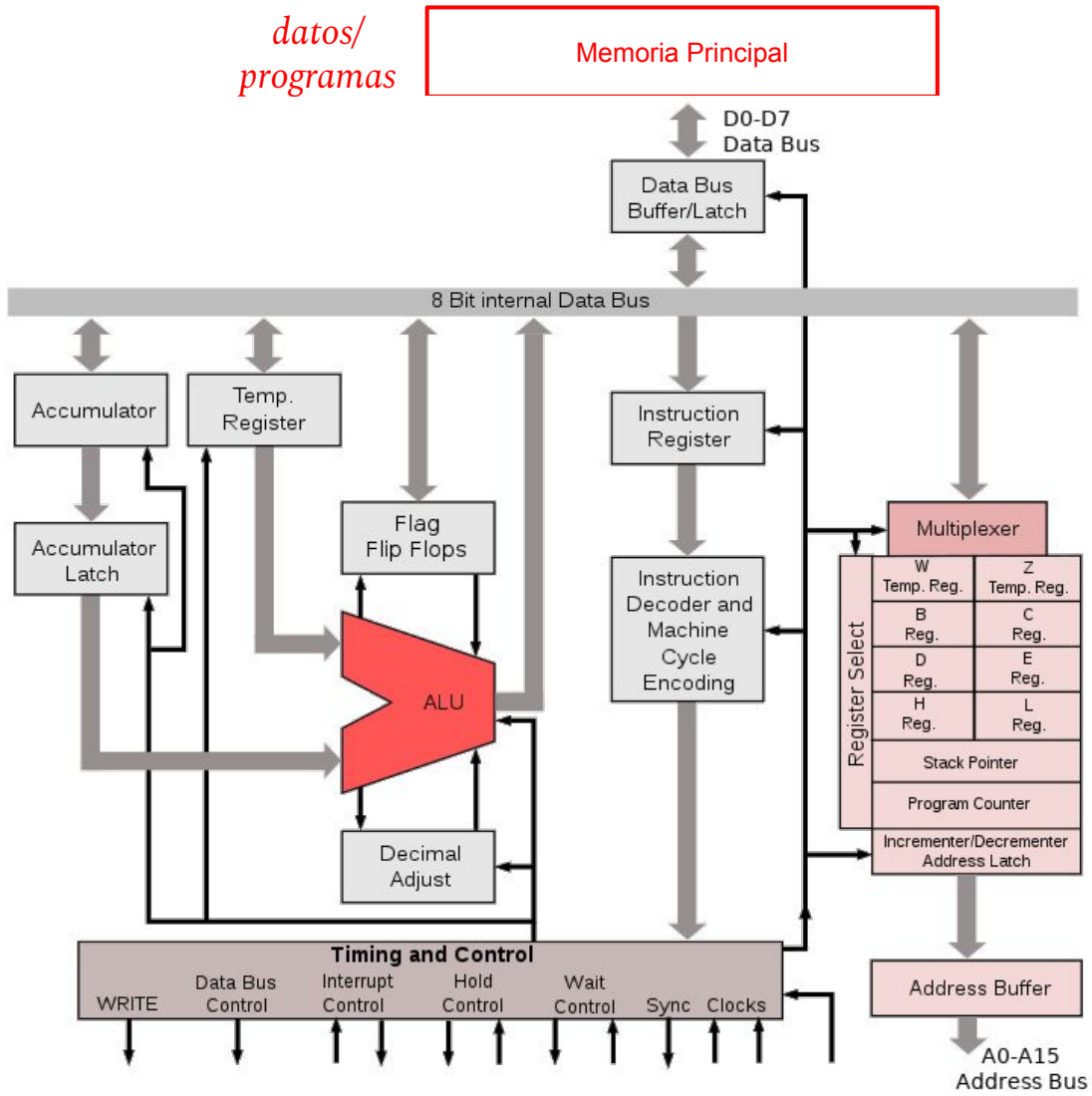
- Si bien en microcontroladores se mantiene esta diferenciación (Flash/PROG - SRAM/DAT), en computadores Harvard actuales se emplea una arq. Harvard modificada donde los programas se almacenan en memoria principal pero se emplean memorias caché y buses separados



Los computadores (IV)

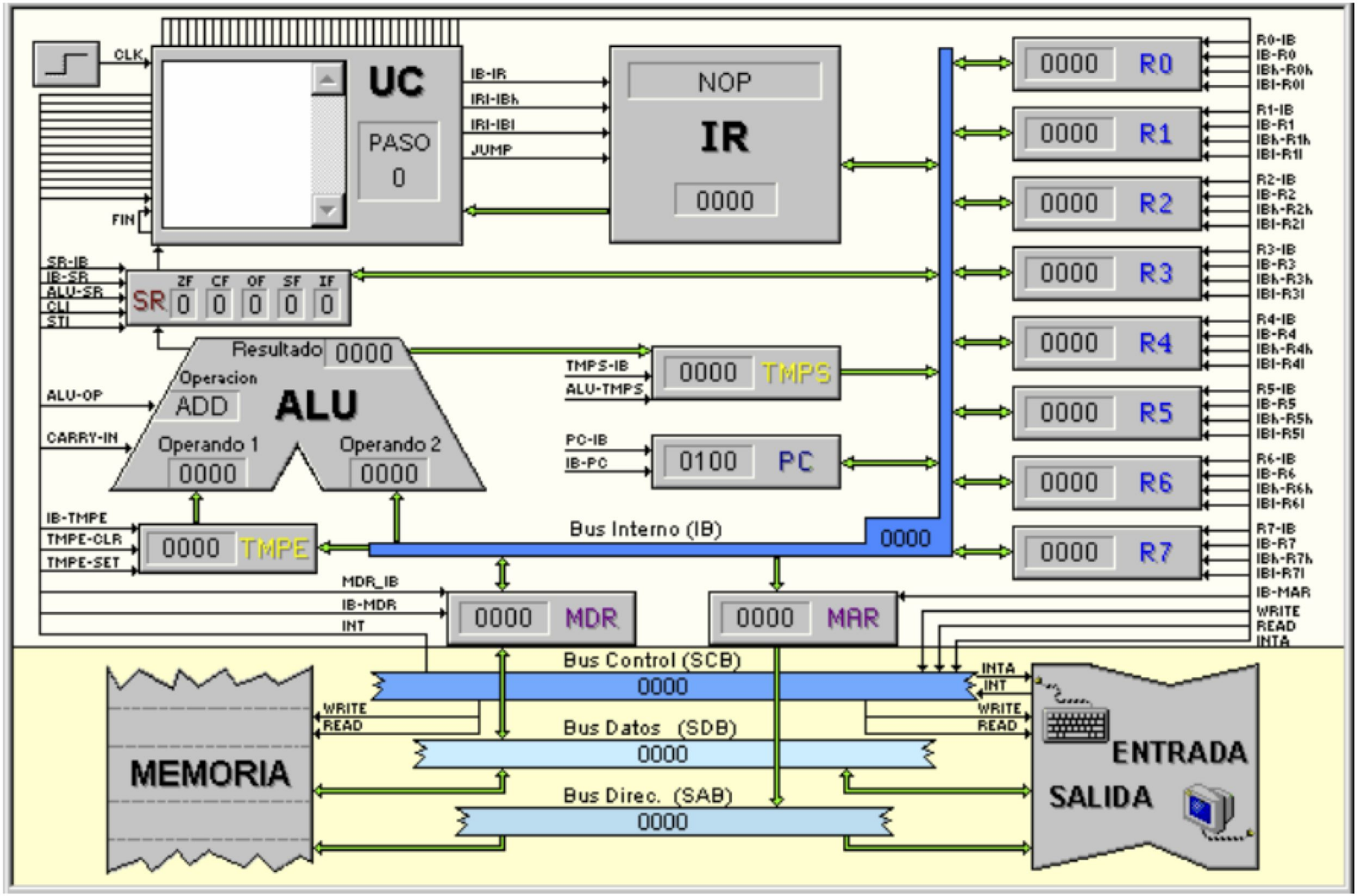


Arquitectura Atmel AVR - (Harvard)



Arquitectura intel 8080 - (von Neumann)

Los computadores (V)



Simulador computadora von Neumann

Los computadores (VI)

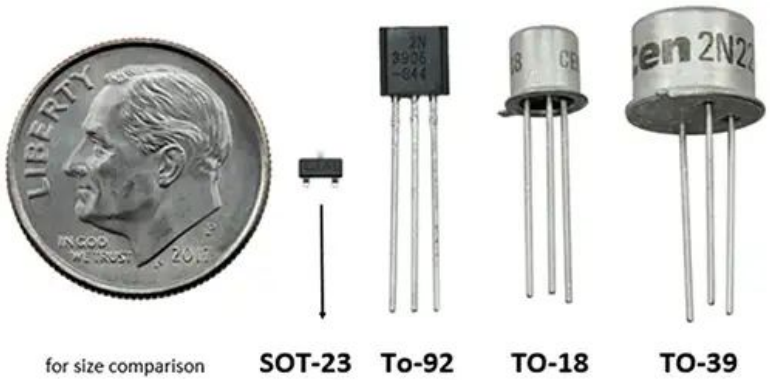
otros hitos de la historia de la computación...

- Eckert y Mauchly, forman la empresa Eckert-Mauchly y fabrican los primeros computadores comerciales. UNIVAC I, entregado en 1951, se vendió por 1 millón de dólares (se fabricaron 48 sistemas!)
- 1953: IBM fabrica su primera computadora a escala industrial, la IBM 650. Se amplía el uso del lenguaje ensamblador para la programación de las computadoras. Los ordenadores con transistores reemplazan a los de válvulas, marcando el comienzo de la 2ª generación de computadoras.
- 1957: Jack S. Kilby construye el primer circuito integrado. Un equipo de IBM dirigido por J.W. Backus desarrolla el lenguaje FORTRAN y el primer compilador del mismo para un IBM 704
- 1959: se diseña COBOL (*COmmon Business-Oriented Language*), patrocinado por el US DoD, como lenguaje portable de propósito general

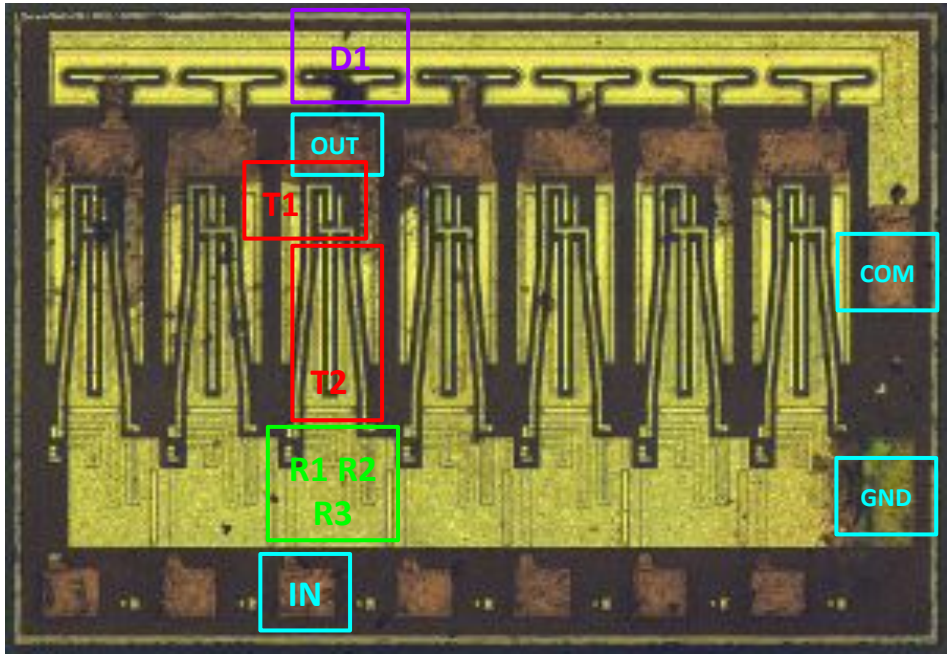
Los computadores (VII)

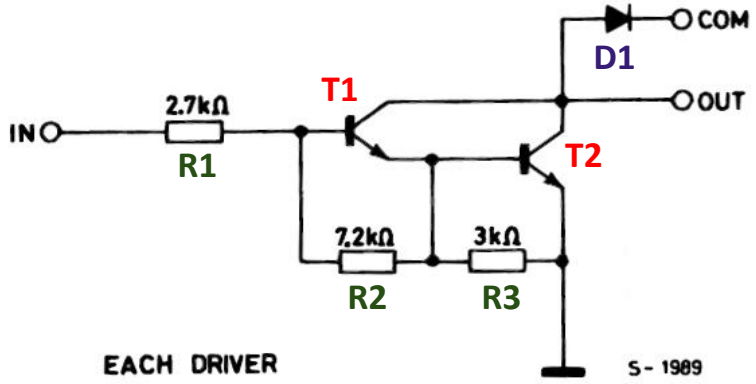
C. Impresos vs Integrados

Transistores para circuitos impresos



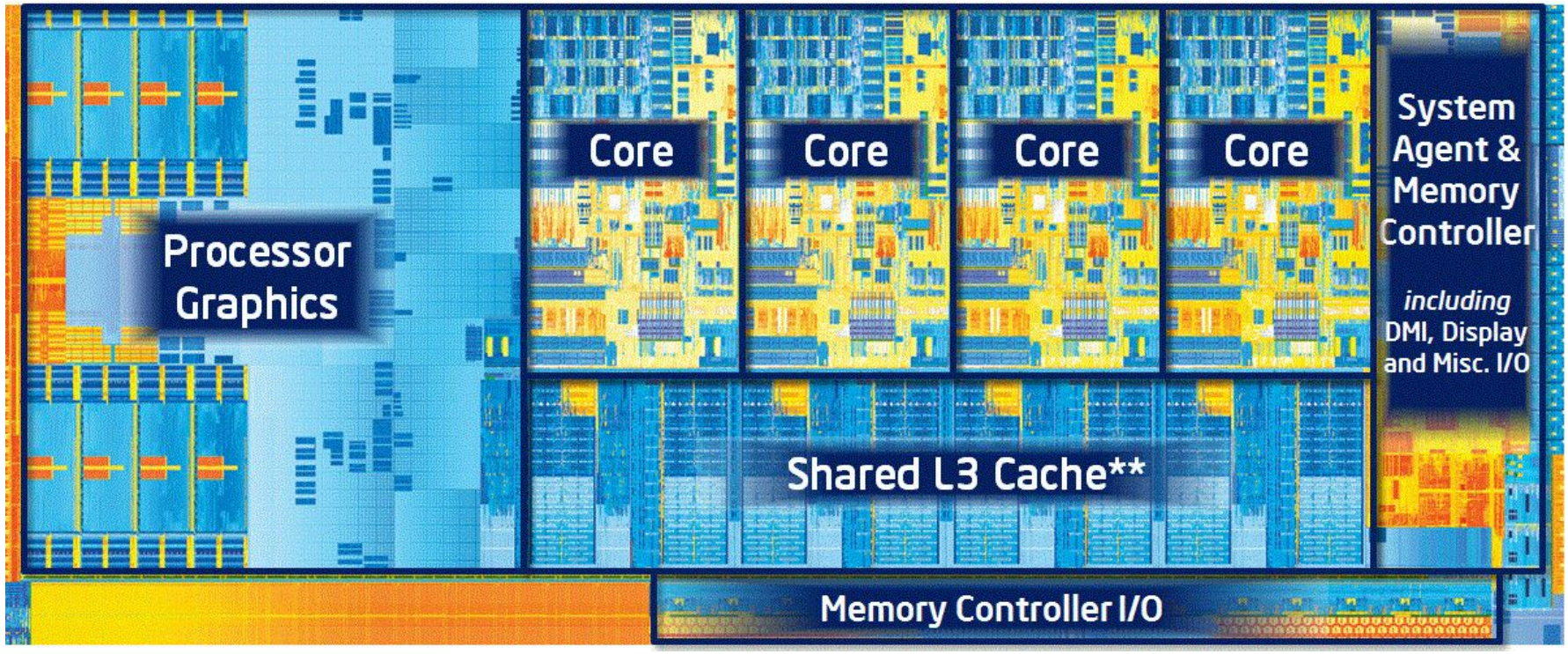
Ejemplo de Circuito Integrado: ***ULN2003***
(incluye 7 pares de transistores en configuración Darlington)





Series ULN-2003A
(each driver)

Los computadores (VIII)



Intel Intel Ivy Bridge (Core i7 3770K): 1,4 x 10^9 transistores; 160 mm^2

Los computadores (IX)

- 1964: IBM lanza la familia de computadores S/360, marcando el inicio de la 3ª generación en la que las placas de circuito impreso pasan a ser reemplazadas por placas de circuitos integrados.

  John Kemeny y Thomas Kurtz inventan el lenguaje BASIC en el Dartmouth College. Lenguaje de propósito general, interactivo y de fácil aprendizaje para ser usado por estudiantes noveles sin conocimientos del hardware del computador

- 1965: Digital Equipment Corporation (DEC) presenta el PDP-8, el primer microcomputador comercial, precursor del microprocesador

- 1969: Ken Thompson, Dennis Ritchie y Douglas McIlroy de los laboratorios Bell de AT&T crean el sistema operativo UNIX para un PDP-7

  Se publica el RFC 1 que describe la primera Internet (ARPANET)

Los computadores (X)

- 1970: El profesor suizo Niklaus Wirth publica el lenguaje PASCAL, orientado a facilitar el aprendizaje de la programación utilizando programación estructurada

- 1971: Intel presenta el primer microprocesador comercial, el Intel 4004. Era de 4 bits y encapsulaba 2.300 transistores
- 1972: Dennis Ritchie desarrolla el lenguaje de programación C con objeto de portar el sistema operativo UNIX. Aunque de propósito general, está orientado a la programación de sistemas

- 1975: Lanzamiento del Altair 8800, primer microcomputador personal, basado en la CPU Intel 8080 Bill Gates y Paul Allen fundan Microsoft para desarrollar y comercializar intérpretes de BASIC para el Altair 8800
- 1976: Zilog lanza el Z80, microprocesador compatible a nivel de código con el i8080. Se popularizó en los 80 gracias los computadores Sinclair Spectrum, Amstrad CPC y MSX

Los computadores (X)

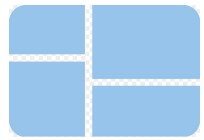
- 1976: Steve Wozniak, Steve Jobs y Ron Wayne (vendió sus acciones a los 12 días por 800$) fundan Apple.
- 1977: Apple presenta el Apple II, basado en el microprocesador MOS 6502 de 8 bits. Es el primer computador personal que se vende a gran escala

- 1981: IBM lanza al mercado el IBM PC basado en el microprocesador Intel 8088 de 16 bits, que se convertiría en un éxito comercial. De arquitectura abierta, marcaría una revolución en el campo de la computación personal y definiría nuevos estándares

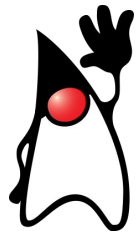
- 1982: Microsoft saca al mercado el MS-DOS, que será el sistema operativo más utilizado en los IBM PC-compatibles hasta mediados de los 90

Los computadores (y XI)

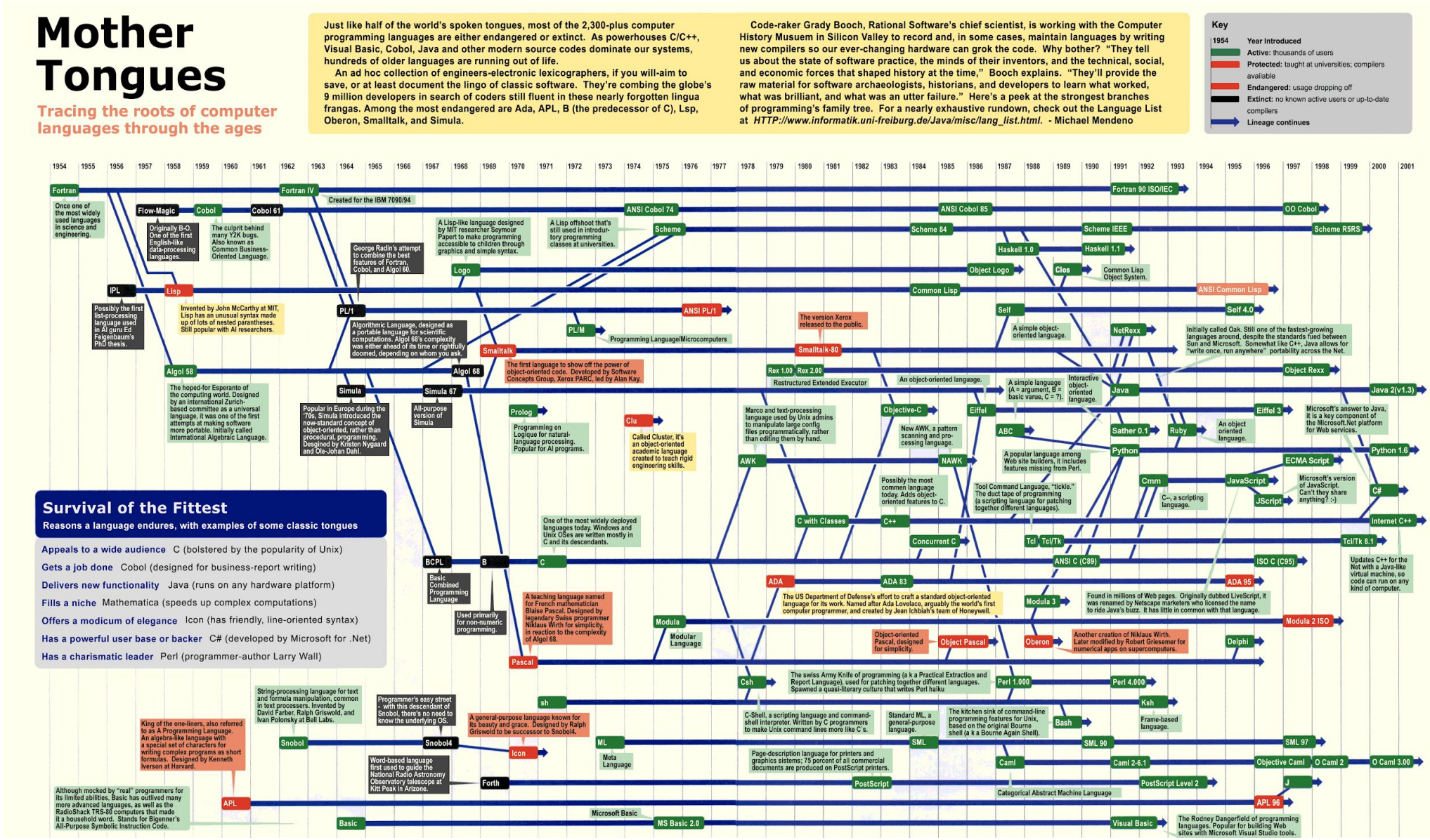
- 1983: Bjarne Stroustrup publica el lenguaje de programación C++
- 1985: Microsoft presenta el s.o. Windows 1.0
- 1990: Tim Berners-Lee idea el hipertexto para crear el World Wide Web (www). Sienta las bases del protocolo HTTP y el lenguaje HTML
  Guido van Rossum crea el lenguaje de programación Python
- 1991: Linus Torvalds comienza a desarrollar Linux
- 1996: Sun Microsystems publica la primera versión de Java
- 1998: Larry Page y Serguéi Brin fundan Google Inc.
- 2007: Apple lanza el iPhone en EEUU
  Google anuncia Android 1.0
- 2008: Sale al mercado el HTC Dream, primer smartphone Android








Lenguajes de programación (I)



https://ccrma.stanford.edu/courses/250a-fall-2005/docs/ComputerLanguagesChart.png

Lenguajes de programación (II)

Comunicándonos con el computador

- Los computadores son máquinas programables. Esto quiere decir que son capaces de ejecutar una secuencia arbitraria de instrucciones para completar una tarea determinada (programa)
- Los circuitos microprogramables son digitales, es decir, operan en base a dos únicos estados de tensión, "HIGH" y "LOW" que se simbolizan respectivamente como 1 y 0. Tanto el diseño de los circuitos como la programación de los mismos se basa en esta lógica binaria
- Para poder dar órdenes al computador hay que hablar en "su" lenguaje denominado código máquina. Cada computador (CPU) define su propio repertorio de instrucciones y es el único lenguaje que "entiende". Cada una de estas instrucciones está codificada como una serie de dígitos binarios que la CPU es capaz de interpretar. Los códigos máquina de distintas CPU son muy parecidos porque el nº de instrucciones es limitado y sus tecnologías hardware se basan en los mismos principios.

Lenguajes de programación (III)

Lenguajes de bajo nivel

- Los primeros lenguajes de programación eran lenguajes de bajo nivel, es decir, muy ligados a la propia arquitectura de la CPU sobre la que se ejecutaban. Esta fuerte dependencia impide que un programa escrito para una CPU se pueda ejecutar directamente en otra arquitectura.
- Tienen la ventaja que nos permiten extraer las máximas ventajas del hardware sobre el que se ejecutan, haciendo un uso óptimo tanto de la memoria del computador como del tiempo de procesado
- Como contrapartida, desarrollar un programa de bajo nivel exige una cantidad sustancial de tiempo y un claro conocimiento del funcionamiento y arquitectura interna del procesador
- Hoy día su uso está restringido a pequeños programas, segmentos de código críticos y que se deban ejecutar de la forma más eficientemente posible (drivers, rutinas gráficas, microcontroladores, virus, exploits,...)

Lenguajes de programación (IV)

1ª Generación de Lenguajes

- La 1GL está compuesta por estos lenguajes o código máquina que puede interpretar cada procesador. En las primeras computadoras los programas escritos en código máquina consistían en reconfiguraciones del cableado y hardware de la propia máquina. Con la aparición de computadores digitales de programa almacenado se empezó a usar la codificación binaria de dichas instrucciones.

2ª Generación de Lenguajes

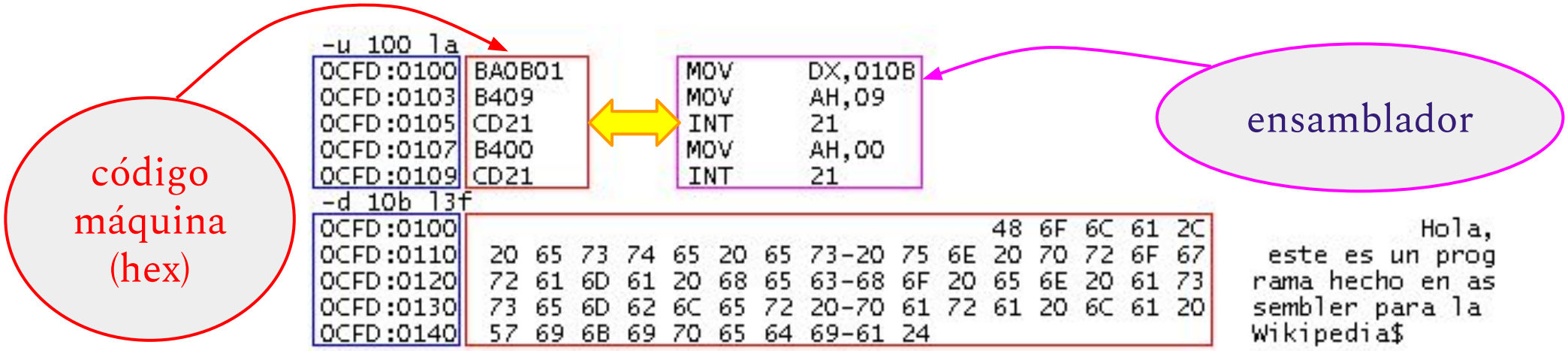
- La 2GL se corresponde con los lenguajes ensamblador ("*assembly*"). Con objeto de facilitar la introducción de programas (codificación en binario!), se diseñó un lenguaje de bajo nivel formado por "mnemónicos" o representaciones simbólicas (MOV, ADD, PUSH,...) del código máquina binario de los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables

Lenguajes de programación (V)

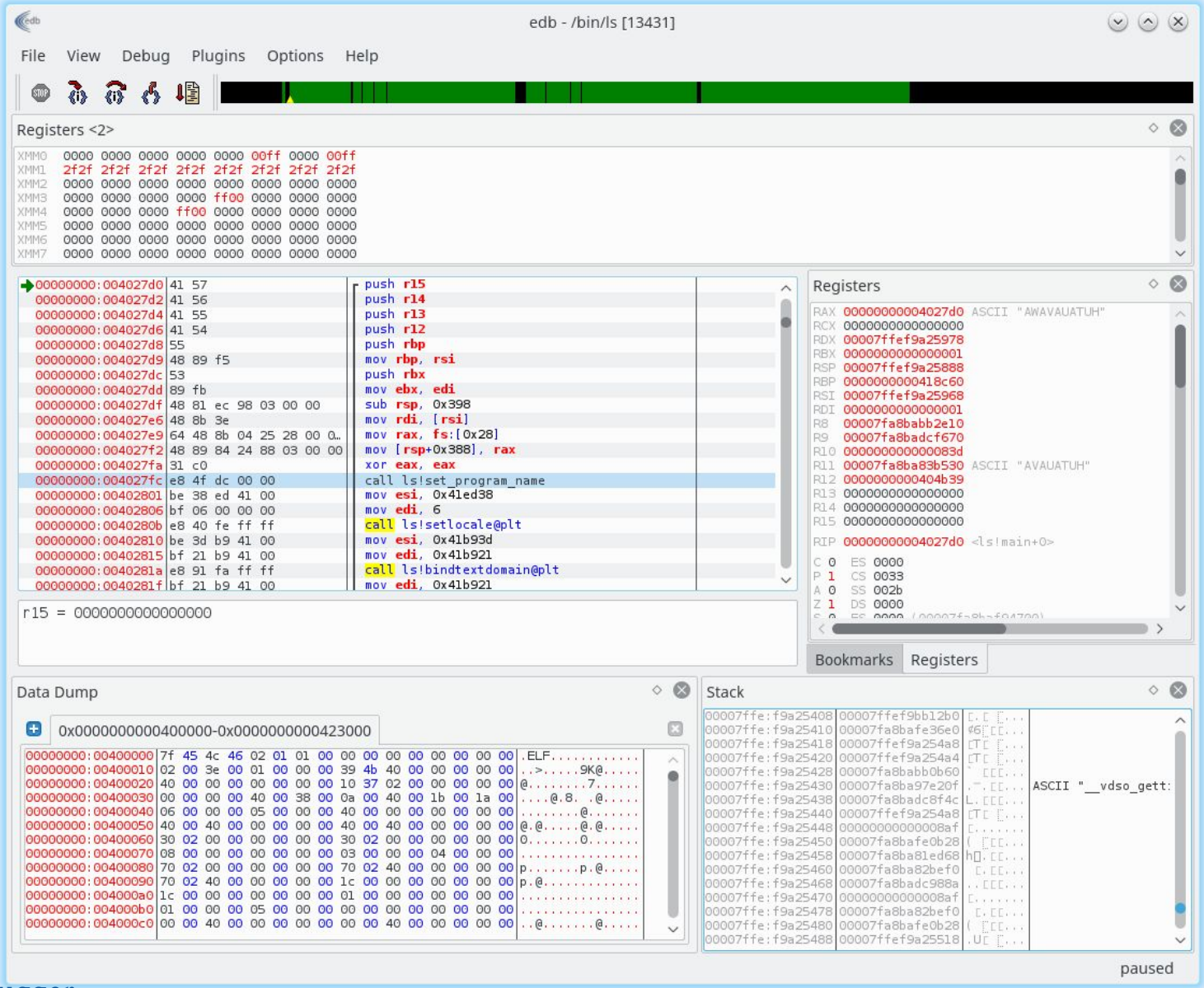
Ensamblador

- Cada arquitectura de procesador tiene su propio lenguaje ensamblador
- Constituye la representación más directa del código máquina, presentando prácticamente una relación 1:1
- Para realizar la "traducción" del código fuente en ensamblador al código máquina (código objeto) necesitamos un programa ensamblador ("assembler")

- Ejemplo de código ensamblador y máquina del Intel 8088 (MS-DOS):



Lenguajes de programación (VI)



edb x86/x86-64 debugger

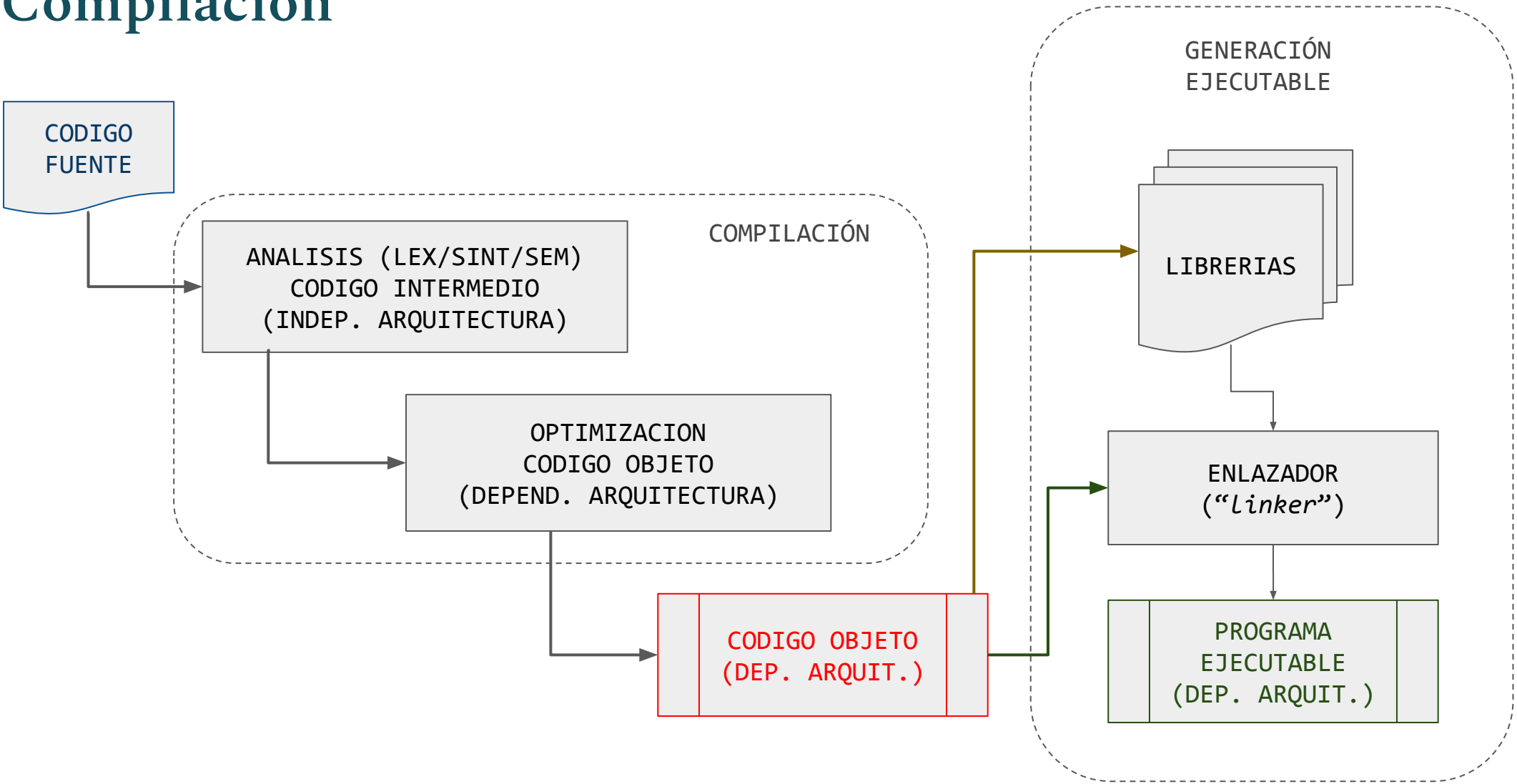
Lenguajes de programación (VII)

Lenguajes de Alto Nivel

- A medida que los computadores fueron evolucionando, se desarrollaron nuevos lenguajes más próximos al lenguaje humano (alto nivel), independientes del hardware del computador, más fáciles de usar y productivos. A costa del sacrificio en eficiencia, permiten un desarrollo más rápido y portabilidad del código fuente entre arquitecturas.
- Están formados por un conjunto de construcciones, próximas al lenguaje natural, junto con reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones
- El uso de estos lenguajes implica que debe existir un proceso de traducción del programa escrito en uno de estos lenguaje (código fuente) al código máquina que puede entender la máquina. Los programas encargados de este proceso se denominan compiladores (traducen el programa completo antes de su ejecución) e intérpretes (traducen línea a línea a medida que se va ejecutando el programa)

Lenguajes de programación (VIII)

Compilación



Lenguajes de programación (IX)

3ª Generación

- La 3GL está compuesta por lenguajes de alto nivel de propósito general como COBOL, C, Python, Java,... Si bien cualquier programa escrito en un lenguaje se debería poder realizar en otro, suelen estar orientados hacia diferentes campos o áreas.
- En lugar de tratar con registros, pilas ("*stacks*") y direcciones de memoria, se utilizan variables, matrices, objetos, expresiones booleanas, funciones, bucles, hilos y otros conceptos de informática abstracta, enfocados en la facilidad de uso y no en la eficiencia óptima del programa
- Generan código más compacto, sencillo y comprensible
- Su nivel de abstracción del hardware permite que un mismo programa pueda ser portado a diferentes arquitecturas
- Permiten crear programas con muchas menos líneas de código y facilitan la reutilización del mismo, reduciendo notablemente los costes de desarrollo

Lenguajes de programación (X)

Paradigmas de programación

- Los lenguajes 3GL soportan diferentes paradigmas de programación o "enfoques" a la hora de diseñar nuestros programas:
- Programación imperativa. Base de la mayor parte de lenguajes de programación, los programas se expresan como secuencias de comandos que el computador debe ejecutar. Ej: Basic, C, Pascal, Java, Python,...
- Programación procedimental. Tipo de programación imperativa en la que el programa se compone de funciones o subrutinas. Junto con las estructuras de control (secuencia, selección, iteración) componen la base de la denominada programación estructurada
- Programación orientada a objetos. Basada en la imperativa, pero incluye la definición de clases de objetos y sus relaciones para encapsular y representar el dominio del problema a resolver. Ej.: C++, Java, Python, Objective-C, C#, VB.NET, JavaScript, PHP (v5+),...

Lenguajes de programación (XI)

4ª Generación

- Programación dirigida por eventos ("*event-driven*"). La estructura y ejecución de los programas quedan determinados por los sucesos (eventos) producidos durante la ejecución del mismo como, por ejemplo, durante la interacción del usuario con el mismo. Es la base del desarrollo de interfaces gráficas de usuario (GUI). Ej: Visual Basic,...
- Si bien la definición de 4GL se ha ido modificando a lo largo del tiempo, suele asociarse a lenguajes de un mayor nivel de abstracción del hardware que los 3GL y, habitualmente, de dominio específico (bases de datos, informes, interfaces,...). Suelen basarse en la programación declarativa (en contraposición a la imperativa). donde se declaran relaciones, condiciones, proposiciones, ecuaciones o transformaciones que describen el problema y su solución. Es decir, se indica "qué" queremos obtener, no "cómo". Ej: Prolog, Haskell, SQL

Lenguajes de programación (XII)

Estructura de los lenguajes de programación

- Cada lenguaje de programación está formado por un conjunto de construcciones primitivas, una sintaxis, una semántica estática y una semántica
- Por analogía a las lenguas, las primitivas equivaldrían a las palabras, así como números y operadores
- La sintaxis establece qué combinaciones de esas palabras y símbolos está bien formada.
- Por ejemplo, en castellano, la frase "perro moto niño" no sería válida, pues precisa de la existencia de, al menos, un predicado que incluya un verbo. Del mismo modo, en Java la secuencia de primitivas 3.5 + 2.0 es válida, pero la secuencia 3.5 2.0, no
- Aunque los errores sintácticos son los más comunes, son los menos peligrosos. Intérpretes y compiladores nos informarán de sus existencia y evitarán la ejecución de dicho código erróneo

Lenguajes de programación (XIII)

- La semántica estática define qué sentencias sintácticamente válidas, tienen significado
- Por ejemplo, en castellano, la frase “Yo corren rápido” es correcta sintácticamente, pues es de la forma <pronombre> <verbo> <adverbio>. Sin embargo, no es correcta desde el punto de vista de la semántica estática, pues no hay correlación entre la persona gramatical del pronombre (1ª singular) y verbo (3ª plural). En Java, por ejemplo, la sentencia 3.5/”abc” es válida sintácticamente (<literal> <operador> <literal>), pero generaría un error, pues no podemos dividir un número entre un texto (cadena de caracteres)
- En cuanto a la detección de este tipo de errores, durante los procesos de compilación o interpretación, hay mucha variedad de unos lenguajes a otros. Así como Java realiza realiza mucho trabajo de chequeo de errores de semántica estática antes de permitir la ejecución, o Python durante la ejecución, el compilador de C realiza mucho menos

Lenguajes de programación (XIV)

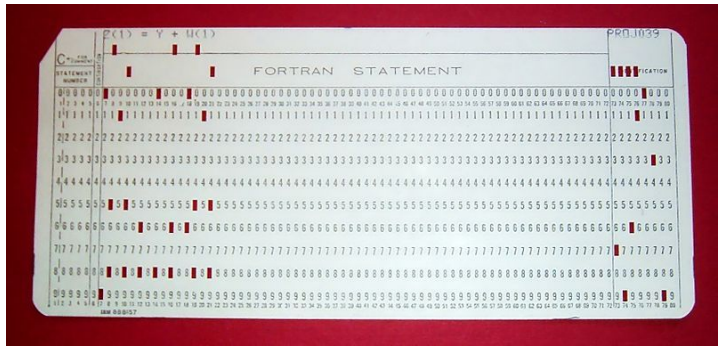
- El último concepto, el de semántica, es independiente del lenguaje que estemos empleando.
- Se refiere al hecho de que todo programa correcto desde el punto de la sintaxis y la semántica estática, va a realizar alguna tarea, es decir, tiene significado, tiene semántica.
- Esto no quiere decir que haga "lo que nosotros esperamos". Es decir, los errores semánticos, que son los más difíciles de detectar, nos pueden llevar a situaciones indeseables como las siguientes:
  - Finalizar de manera abrupta e inesperada ("*crash*")
  - Entrar en un *bucle* de ejecución *infinito*
  - o la peor de todas, completar la ejecución pero producir un resultado incorrecto.
- Para solventar estas situaciones indeseables, especialmente la última, una vez finalizado un programa, se ejecutan completas baterías de pruebas para comprobar que el programa funciona correctamente

Lenguajes de programación (XV)

Algunos "ilustres"...

FORTRAN (*FORmulaTRANslation*)

- Lenguaje de propósito general, compilado e imperativo especialmente dirigido a la computación científica y numérica, areas de uso de los computadores en las décadas de los 50-60 (carrera armamentística y espacial,...)
- En los años 50, un equipo de IBM dirigido por J.W. Backus desarrolla el lenguaje y el primer compilador del mismo para un IBM 704
- Base de numerosos lenguajes posteriores (BASIC), se mantiene en evolución y se sigue empleando en tareas de supercomputación (modelos climáticos, astronomía, química,...)



Lenguajes de programación (XVI)

COBOL (*COmmon Business-Oriented Language*)

- Lenguaje de propósito general, compilado, portable, imperativo y procedimental especialmente dirigido a la computación en áreas de negocios, financieras y administrativas.
- Debido a los elevados costes de desarrollo en los años 50, se creó un grupo de trabajo, patrocinado por el US DoD, para el desarrollo de un lenguaje portable de propósito general para el procesamiento de datos. Diseñado en 1959 y estandarizado en 1968, fue ampliamente adoptado por gobiernos y empresas.
- COBOL es aún ampliamente usado en aplicaciones desplegadas sobre computadores tipo *mainframe*, como trabajos de procesamiento *batch* y *transaccional*. Debido al descenso de su popularidad y el retiro de programadores COBOL experimentados, muchas aplicaciones se están migrando hacia nuevas plataformas (J2EE) y sólo se realizan tareas de mantenimiento de los miles de millones de líneas de código existentes

Lenguajes de programación (XVII)

BASIC (*Beginner's All-purpose Symbolic Instruction Code*)

- La evolución del hardware y velocidad de proceso de las computadoras, permitió el desarrollo de sistemas multiusuario de tiempo compartido, permitiendo la compartición de tiempo de proceso por múltiples usuarios
- En 1964 John Kemeny y Thomas Kurtz diseñan el lenguaje BASIC en el Dartmouth College. Lenguaje de propósito general, interactivo (interpretado) y de fácil aprendizaje, fue creado para acercar la programación a estudiantes (y profesores) que no eran de áreas científico-técnicas y, por tanto, sin conocimientos del hardware del computador
- Con la aparición de las primeras microcomputadoras (Altair 8800) el BASIC se extiende rápidamente, ya que la mayoría de lenguajes del momento eran demasiado grandes para las pequeñas memorias de estos equipos y los medios de almacenamiento (cinta de papel) lentos

Lenguajes de programación (XVIII)

- En 1975, Bill Gates y Paul Allen fundan Microsoft para comercializar un intérprete de BASIC para el Altair 8800. BASIC se convierte en el lenguaje estándar del Apple II y, en 1979, IBM incluye una ROM en sus IBM PC con un intérprete de BASIC de Microsoft
- Pero es sobre todo en los 80 cuando BASIC vive su época dorada. La venta de millones de microordenadores personales como el Commodore 64, ZX Spectrum, MSX,... que incluían un intérprete BASIC en la ROM del sistema, hace que el el lenguaje llegue al público general y se convierta en el punto de entrada al mundo de la programación para millones de personal a lo largo y ancho del mundo
- El desarrollo de las interfaces gráficas de usuario a finales de los 80, hace que el uso de BASIC comience a declinar hasta que, en 1991, Microsoft publica Visual Basic. Basado en la sintaxis de BASIC pero más potente, se convierte en uno de los lenguajes más utilizados para la plataforma Windows. Las últimas versiones, incluidas en la plataforma .NET (VB.NET), soportan orientación a objetos

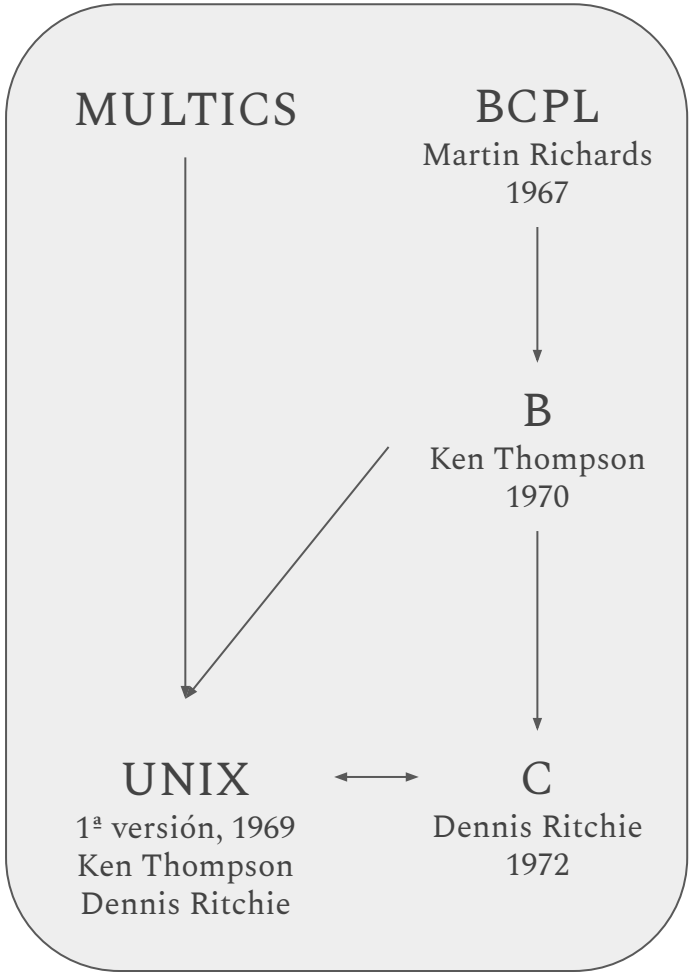
Lenguajes de programación (XIX)

C

- Es un lenguaje de propósito general, compilado, que soporta programación modular y orientación a objetos (C++)
- Presenta características de bajo nivel, lo que permite el desarrollo de programas muy eficientes. Debido a ello, suele estar vinculado al desarrollo de sistemas
- Su desarrollo fue paralelo al del sistema operativo UNIX. En los años 70, Ken Thompson, de los Bell Labs de AT&T, inicia junto a antiguos colaboradores del proyecto MULTICS del GE 645 (Joe Ossanna, Rudd Canaday y Dennis Ritchie), el desarrollo de un nuevo sistema operativo denominado UNICS (posteriormente UNIX) para un PDP-7
- El desarrollo se realiza en ensamblador, debido a la experiencia que ya tenía Thompson en esta máquina cuando portó “Space Travel” desde el GE 635 al PDP-7. Thomson inicia el desarrollo de un nuevo lenguaje de alto nivel, denominado B, para evitar el desarrollo en ensamblador

Lenguajes de programación (XX)

- Desde ese momento, todas las mejoras de UNIX empiezan a escribirse en B y Dennis Ritchie inicia el desarrollo de un nuevo lenguaje a partir de B, que denomina C
- En 1970, Bell Labs compra varios PDP-11 y el equipo comienza a portar UNIX a los nuevos equipos reescrito en C. En 1973 se finaliza el desarrollo del kernel UNIX. Posteriormente, en 1975, Bell Labs libera de forma gratuita (por las leyes antimopolio de EEUU) la versión 6 de UNIX
- En 1983, Bjarne Stroustrup publica el lenguaje C++, extensión de C que soporta OO
- Desde entonces, el lenguaje C se convierte en uno de los lenguajes más usado, siendo la base de lenguajes posteriores (Java, PHP,...)



Lenguajes de programación (XXI)

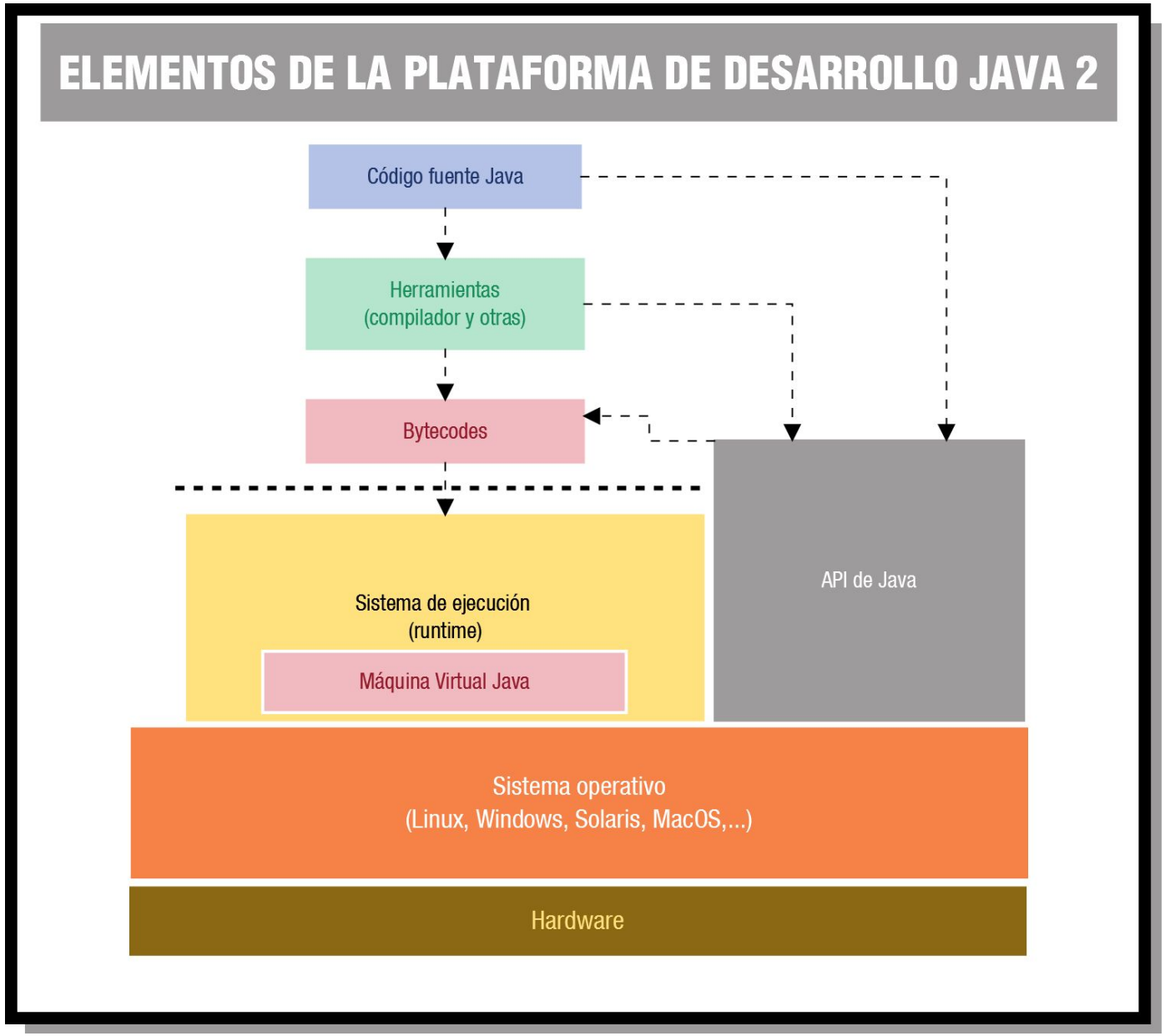
Java

- Java es un lenguaje orientado a objetos de propósito general e independiente de la plataforma
- Fue concebido en 1991 por un equipo de Sun Microsystems dirigido por James Gosling. Denominado “Oak” originalmente, fue renombrado a “Java” en 1995.
- La motivación principal en su desarrollo fue la de crear un lenguaje independiente de la plataforma que pudiera usarse para crear software embebido en todo tipo de dispositivos electrónicos de consumo (microondas, televisiones, controles remotos,...)
- Aunque era técnicamente posible compilar un programa C++ para cualquier tipo de CPU, era necesario disponer del compilador correspondiente para dicha arquitectura. Gosling y su equipo decidieron desarrollar un lenguaje que permitiera la ejecución del código en cualquier tipo de CPU ("*write once, run anywhere*" WORA)

Lenguajes de programación (XXII)

- En lugar de compilar el código fuente en Java a código objeto de alguna arquitectura (como en C), se genera un código intermedio (bytecode) independiente de la plataforma. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware). En la JVM se interpreta el bytecode convirtiendo a código máquina (compilador JIT) y se ejecuta
- El desarrollo de Java coincidió con la "explosión" de Internet. Debido a la necesidad de programas portables por la propia concepción heterogénea de la red, provocó que Java se desplazara de la electrónica de consumo a la programación en Internet, convirtiéndose en una absoluta historia de éxito de los lenguajes de programación
- Parte de su éxito vino derivada también de la adopción de la sintaxis del lenguaje C y cierta simplificación respecto a C++, lo que facilitó que la comunidad de programadores lo adoptara

Lenguajes de programación (XXIII)



Lenguajes de programación (y XXIV)

Python

- Es un lenguaje de propósito general, interpretado y de tipado dinámico que soporta diferentes paradigmas de programación como la programación estructurada o la orientada a objetos
- Su desarrollo se inició a finales de los ochenta como un proyecto personal del programador holandés Guido van Rossum que, en ese momento, se encontraba trabajando en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica) en Holanda
- Fue diseñado como un lenguaje muy legible y fácil de aprender. Es, por ello, muy usado en la enseñanza de la programación
- Dispone de una gran biblioteca de módulos, escritos tanto en Python como en C, que facilitan su uso en todo tipo de campos. Algunos en los que está teniendo un fuerte crecimiento son la robótica y la IA
- Google es uno de sus más fervientes defensores (*Google Code*, *Youtube*, ...)

Desarrollo de software (I)

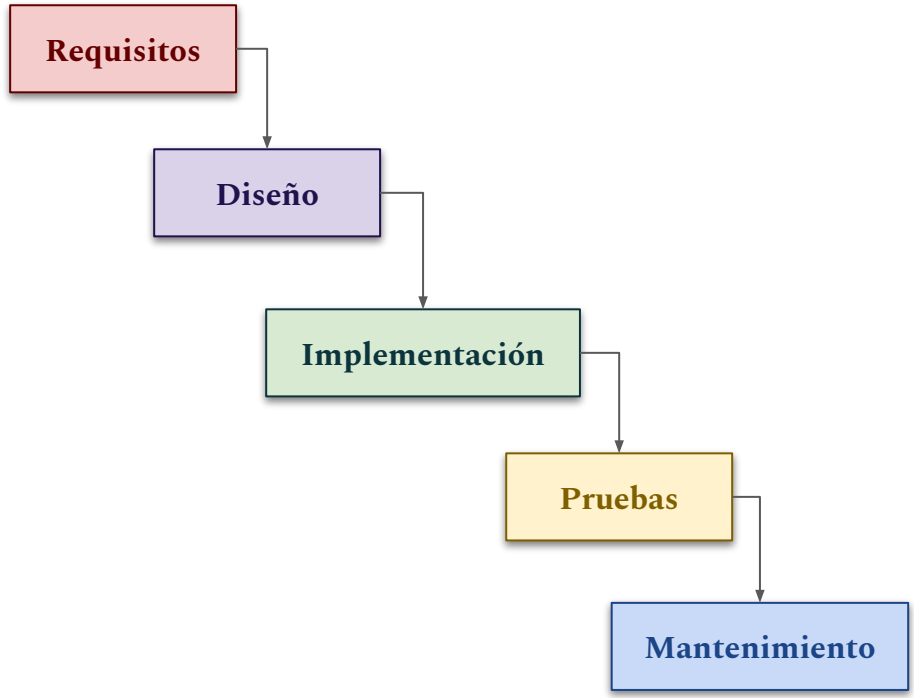
Modelos de desarrollo de software

- En Ingeniería del Software, se definen diversos modelos y metodologías para el desarrollo y mantenimiento del software.
- Todas estas metodologías dividen el ciclo de vida del software en diversas etapas o fases para la mejor gestión y desarrollo del mismo. Entre estas fases nos vamos a encontrar:
  - Fases de definición: planificación del proyecto, análisis de requisitos
  - Fase de desarrollo: diseño, generación y pruebas del código
  - Fase de mantenimiento: corrección de errores
- Algunos de los distintos modelos de desarrollo son:
  - Modelo en cascada
  - Desarrollo mediante prototipos
  - Desarrollo en espiral
  - Desarrollo ágil

Desarrollo de software (II)

Modelo en Cascada

- Modelo de desarrollo clásico, se basa en un enfoque que ordena las etapas del ciclo de vida del software de forma rigurosa. Plantea una aproximación sistemática secuencial del proceso de desarrollo del software, que se inicia con la especificación de requisitos del cliente y continúa con la planificación, el modelado, la construcción y el despliegue para culminar en el soporte del software terminado



- Aunque presenta un elevado control de la programación del desarrollo, su excesiva rigidez provoca que errores iniciales se trasladen por todo el ciclo de vida. Además, la participación del usuario es mínima

Desarrollo de software (III)

Modelo en espiral

- Modelo propuesto por Barry Boehm en 1986, está diseñado como una espiral en la que cada bucle o iteración representa un conjunto de actividades a realizar.
- Las fases de cada iteración:
  - Se determinan los objetivos y las alternativas para alcanzarlos
  - Se analizan los riesgos de las diferentes alternativas
  - Se desarrolla y verifica el software
  - Se planifica la siguiente iteración

- Aunque reduce los riesgos del proyecto e integra el mantenimiento con el desarrollo, da lugar a proyectos costosos y largos en el tiempo. El IEEE lo clasifica como modelo no operativo

Desarrollo de software (IV)

Desarrollo mediante prototipos

- Se basa en la creación paulatina de prototipos con funcionalidad parcial, tratando de reducir los riesgos debidos a malas interpretaciones o cambios de las especificaciones o requerimientos iniciales
- Adecuado en entornos propensos a cambios o con especificaciones poco claras. En ocasiones, se emplean como parte de otros modelos más tradicionales
- Se involucra al usuario durante el desarrollo, lo cual incrementa la aceptación del producto final

Desarrollo ágil

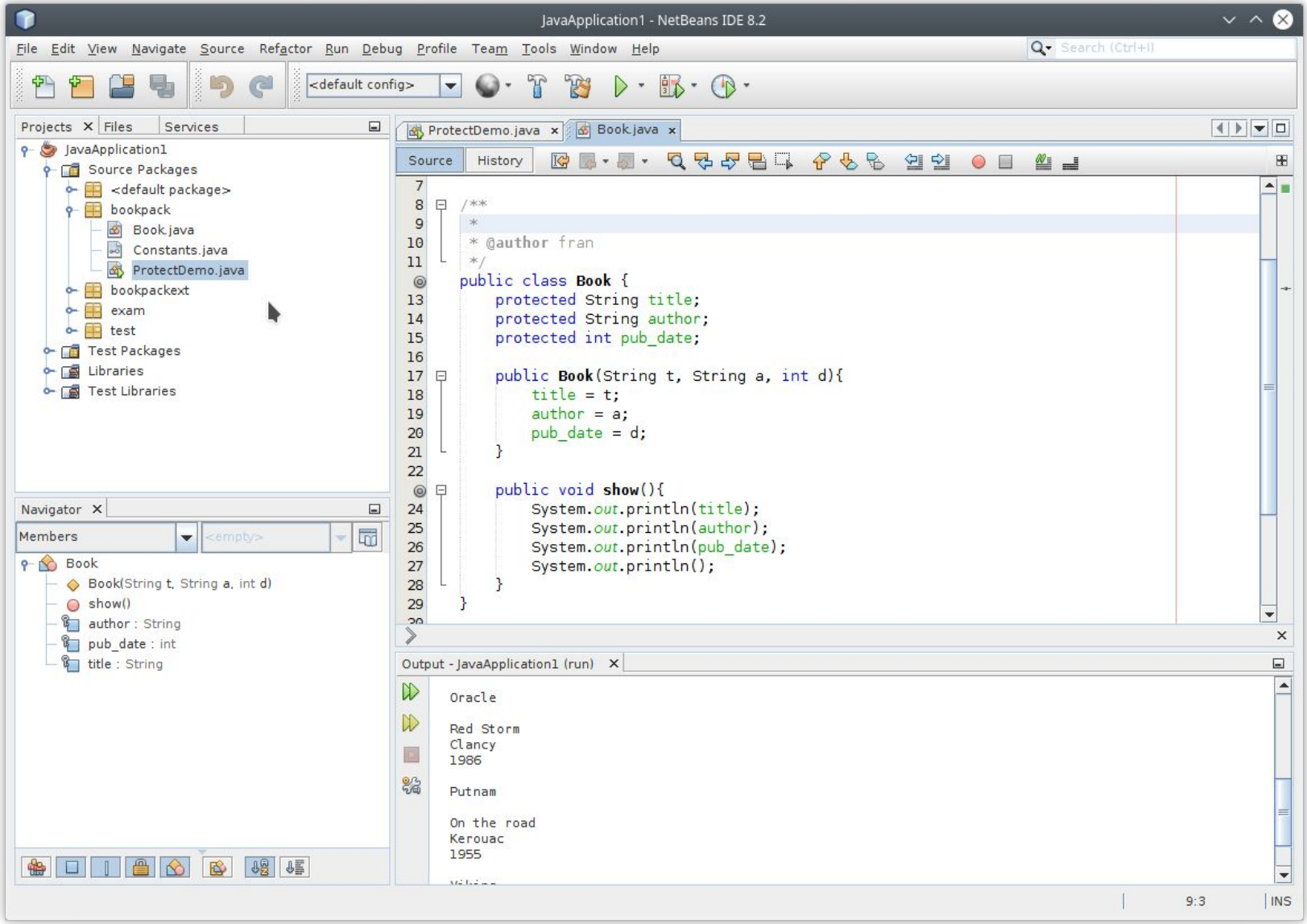
- Se basa en el desarrollo iterativo e incremental. A diferencia de las metodologías tradicionales, utiliza la retroalimentación (*feedback*) en lugar de la planificación como principal mecanismo de control
- El tipo de desarrollo ágil más destacado es la programación extrema (XP) Características: simplicidad, comunicación, *feedback*, *pair-programming*

Desarrollo de software (V)

Entorno de Desarrollo Integrado (IDE)

- Para desarrollar una aplicación no necesitamos más que un simple editor de texto para crear nuestro código fuente y un compilador (o intérprete) del lenguaje que hayamos usado para traducirlo al código máquina del computador
- Sin embargo, a medida que nuestros programas se vuelvan más complejos e incluyen más archivos y librerías, se hace necesario el empleo de un entorno más potente que permita: gestión de los archivos del proyecto, múltiples pestañas de edición, diseño de interfaces gráficas, depurado avanzado, autocompletado de código, resaltado de sintaxis,...
- El objetivo último es mejorar la productividad del programador integrando en un único entorno todas las herramientas de uso habitual junto con ayudas a la edición, compilación y depuración

Desarrollo de software (VI)



IDE NetBeans 8.2

Desarrollo de software (VII)

- Existen multitud de IDE's en el mercado, tanto libres como de pago. Pueden estar diseñados para un lenguaje de programación específico o permitir trabajar con diferentes lenguajes.
- Aunque las capacidades que ofrecen pueden ser diferentes, especialmente en lo relativo a extensibilidad, todos ofrecen un mínimo de características comunes:
  - Creación de proyectos
  - Edición múltiple
  - Resaltado de sintaxis
  - Detección de errores sintácticos
  - Ejecución y Depuración
- En el siguiente enlace, se muestra una comparativa de diferentes IDE's:

  https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments

Desarrollo de software (y VIII)

IDE	License	Written in Java only	Windows	Linux	macOS	GUI builder
BlueJ	GPL2+GNU linking exception	Yes	Yes	Yes	Yes	No
Eclipse JDT	EPL	No	Yes	Yes	Yes	Yes
Geany	GPL	No	Yes	Yes	Yes	No
Greenfoot	GPL	Yes	Yes	Yes	Yes	No
IntelliJ IDEA	C. Ed: Apache License v2.0, U. Ed: proprietary	Yes	Yes	Yes	Yes	Yes
JBuilder	Proprietary	Yes	Yes	Yes	Yes	Yes
JCreator	Proprietary	No	Yes	No	No	No
JDeveloper	Proprietary (freeware)	Yes	Yes	Yes	Yes	Yes
NetBeans	CDDL, GPL2	Yes	Yes	Yes	Yes	Yes
Rational App Developer	Proprietary	Yes	Yes	Yes	No	Yes
Xcode (Apple)	Proprietary	No	No	No	Yes	Yes

Algunos IDE para Java