# Programación de Sistemas

Mtro. en IA José Rafael Rojano Cáceres
rrojano@gmail.com
http://www.uv.mx/rrojano

## Programa de la materia

- Disponible desde el sitio web
  - http://www.uv.mx/rrojano/programacion_sist/


## Organización del curso


- INTRODUCCIÓN A LA PROGRAMACIÓN DE SISTEMAS
- EVOLUCIÓN DE LOS COMPONENTES DE LA PS
- EDITORES
- ENSAMBLADORES
  - Funcionamiento de un ensamblador
  - La reubicación del código
- Diseño general de un ensamblador
- MACRO PROCESADORES
- LIGADORES Y CARGADORES
  - Clases de cargadores
  - Cargador compila y ejecuta
  - Cargador absoluto
  - Cargador relocalizable
  - Cargador de ligado directo
  - Cargador dinámico
- COMPILADORES
  - El proceso de compilación
- INTERPRETES
- ORGANIZACIÓN DE UNA COMPUTADORA
- TAXONOMÍA DE LOS SISTEMAS DE WINDOWS
- WIN 3.1
- WINDOWS 95
  - Arquitectura de Windows 95
- WINDOWS NT
  - Arquitectura de Windows NT
- CARACTERÍSTICAS DE LAS VERSIONES DE WINDOWS
- ARQUITECTURA DE UN SISTEMA UNIX
- LOS SABORES DE WINDOWS Y LOS CONTROLADORES
  - MODO REAL DE WINDOWS
  - MODO ESTÁNDAR DE WINDOWS
  - MODO MEJORADO DE WINDOWS
- MÁQUINAS VIRTUALES
- HARDWARE VIRTUAL
- CONTROLADORES DE DISPOSITIVOS VIRTUALES
- PROGRAMACIÓN DE SISTEMAS EN WINDOWS
  - PROGRAMACIÓN DE 16 BITS.
  - PROGRAMACIÓN DE 32 BITS.
  - PROGRAMACIÓN DE 64 BITS.
- INTERFACES PARA PROGRAMACIÓN DE SISTEMAS
- BIBLIOTECAS DE ENLACE DINÁMICO (DLL)
- PROGRAMACIÓN DE DLL PARA DIFERENTES LENGUAJES
- PROGRAMACIÓN EN C#
  - EJEMPLO DE ACCESO A DISPOSITIVO
- PROGRAMACIÓN DE APLICACIONES EN JAVA
  - EJEMPLO DE ACCESO A DISPOSITIVO
- BIBLIOGRAFÍA
- INFORMACIÓN ACERCA DEL PIC 8259 Y SUS IRQ
- PROGRAMACIÓN, PROGRAMACIÓN Y MÁS PROGRAMACIÓN DE USTEDES


## Acerca de las responsabilidades

- Cada uno es responsable de si mismo
  - Si usted está en su tercera inscripción es **SU** responsabilidad:
    - Asistir
    - Entregar las tareas
    - Participar
    - Desarrollar los proyectos
    - Etcétera
  - Si usted está en cualquier otro caso
    - Se aplican las mismas reglas

## Acerca de los trabajos

- Los trabajos copiados serán calificados con cero
- Los trabajos escritos deberán contar con su respectiva bibliografía
- Los trabajos deberán enviarse por correo electrónico con la información siguiente:
  - Word 2003
  - rrojano@gmail.com
  - Subject: Tarea No.
  - Cuerpo: Descripción de la tarea
- Si su trabajo se pierde, es su responsabilidad estar al tanto de que se recibió.
- Proyectos entregar en CD, fuentes y ejecutable (sin virus) y la forma de compilación, versión del compilador y manual de uso

## Evaluación

- Exámenes 50%
- Trabajos 50%

## Proyectos

- Construcción de un editor de comandos
- Construcción de una máquina de Turing
- Construcción de una DLL
- Programación de una interfase con una cámara de video
- Programación de una aplicación para arquitectura RISC
- Programación de un móvil

## ¿Preguntas?

## Iniciamos....

## Tópicos de la programación de sistemas

- La lógica digital *(arquitectura I)*
- Codificación de los datos *(arquitectura I)*
- La organización de los componentes *(arquitectura I)*
- Ensamblador *(arquitectura Ii)*
- Programación en lenguaje C *(Algoritmos)*
- Compilar, ensamblar, ligar, cargar *(Algoritmos)*
- Operaciones básicas del sistema operativo
- Interfaces con el sistema operativo
- Formatos de archivos
- Dispositivos y controladores
- Etc.

## Definición

- Según [Tischer 96] para entender qué es la programación de sistemas se debe partir del concepto de programación de una aplicación ... allí el punto que importa es el como se representa y se manipula la información. Los algoritmos no dependerán de la *arquitectura o sistema en cuestión*, sino que se pueden describirse de forma general para cualquier máquina imaginable (máquina de Turing). Lo que sí dependen del ***sistema*** es que pasa con la información una vez *dentro* del programa y la forma en que ésta saldrá una vez tratada. Así, las funciones que competen a la programación de sistemas son el: acceso a archivos, el teclado, la pantalla y el resto de dispositivos...

## Componentes de la programación de sistemas


Fig. 1. Componentes de la programación de sistemas

[Donovan 72]

## Del código fuente al ejecutable

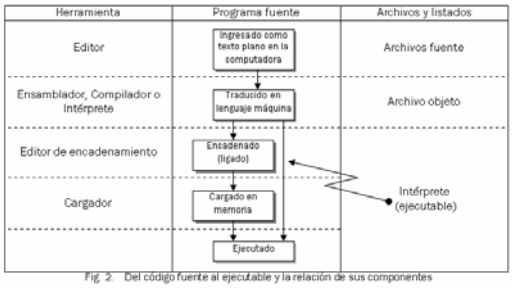


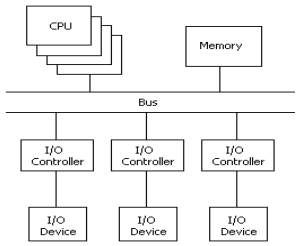
Fig. 2: Del código fuente al ejecutable y la relación de sus componentes

[Englander 02, Pág. 670]

## Organización de la computadora

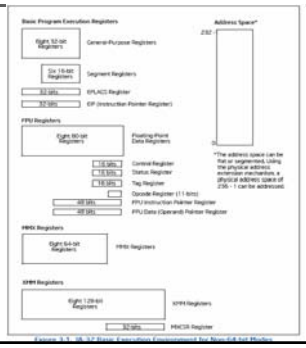
Un repaso

## Esquema general de una CPU

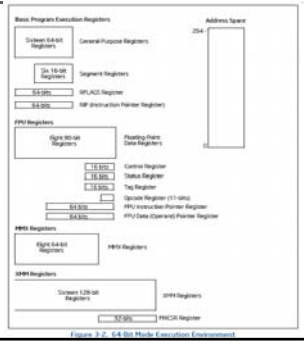



http://www.intel.com/products/processor/manuals/index.htm
Intel® 64 and IA-32 Architectures
Software Developer's Manual, Volume 1: Basic Architecture

## Arquitecturas Intel 32 bits



## Arquitecturas Intel 64 bits



## Los componentes de la PS

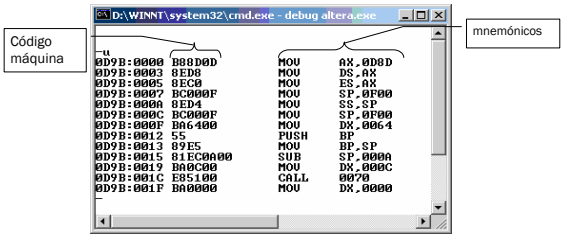
## Editores

- Desde el punto de vista de los editores existen dos clasificaciones según [Englander 02, Pág. 672]:
  - Los editores de línea y los de pantalla.
- Los editores de línea aceptan las instrucciones una a una tal y como si fuese una ventana de comandos (shell) del SO.
  - (¿piensa usted en algún ejemplo).
- Los editores de pantalla son más parecidos a un procesador de textos, uno se puede desplazar libremente por la pantalla y simplemente editar. Dentro de esta categoría existen algunos editores particulares que permiten trabajar en modo comando o modo edición
  - (Misma pregunta, ¿?).

## Ensambladores

- Los ensambladores nacieron a partir de la necesidad de simplificar la **traducción** de las instrucciones, las cuales originalmente se ingresaban byte a byte en la máquina.
- Por ello a los ensamblador se les conoce también como **traductores** [Englander 02, Pág. 674].
- El **lenguaje ensamblador** permite representar las operaciones y las direcciones de las instrucciones por mnemónicos.
- Un **ensamblador** es una utilería que línea a línea traduce el código fuente en lenguaje ensamblador a un código binario ejecutable, [Englander 02, Pág. 676].
  - Ejemplo Fig. 3:

## Fig 3. Listado en debug




## Diseño general de un ensamblador

- Según [Donovan 72] Un ensamblador debería realizar las siguientes tareas:
- Generar instrucciones
  - Traducir los nemotécnicos a su código de operación
  - Evaluar los argumentos, encontrar el valor de cada símbolo, procesar literales y asignar direcciones.
- Procesar pseudos operaciones (palabras reservadas)

## Pasadas en el ensamblador (1/2)

- Paso 1: Definir símbolos y literales
  1. Determinar longitud de la instrucción de máquina
  2. Llevar un contador de posición
  3. Recordar los símbolos encontrados hasta el paso 2
  4. Procesar algunas pseudo instrucciones
  5. Recordad las literales

## Pasadas en el ensamblador (2/2)

- Paso 2: Generar programa objeto
  1. Establecer el valor para los símbolos
  2. Generar instrucciones
  3. Generar datos
  4. procesar pseudo instrucciones
- Véase la Figura 3.3 de [Donovan 72, Pág. 63]. Para mayores detalles de su construcción refiérase al capítulo tres del mismo libro.
- Lectura adicional **Ensambladores** [Tanenbaum 00, Págs. 498-506]

## *Macro procesadores*

- Generalmente en ensamblador se requiere de la repetición de porciones de código, para ello se utilizan las macro instrucciones o simplemente llamadas macros. Las *macro instrucciones* son abreviaciones de instrucciones en una sola línea.

## Implementación

- Una posible implementación de un macro procesador consiste en:
  1. Reconocimiento de la macro. Se establecen los marcadores dentro de los cuales vivirá el código (*macro* y *endmacro*).
  2. Almacenamiento de la definición de la macro. El macro procesador deberá ser capaz de recuperar el código definido para expandirlo en una macro.
  3. Reconocimiento de llamadas. Se debe distinguir entre macros e instrucciones invocadas.
  4. Expansión de llamadas y argumentos. El procesador debe sustituir todas las llamadas.
- Véase [Donovan 72] capítulo cuatro Pág.133 para mayores detalles para la implementación de un macro procesador.
- Lectura adicional **macros** [Tanenbaum 00, Págs. 494-498]

## *Ligadores y cargadores*

- El **proceso de ligado** consiste en reunir o encadenar en un solo archivo ejecutable diferentes módulos objeto.
- Un **cargador** es un programa que coloca el código ejecutable en memoria para su ejecución. Las funciones generales que tiene un cargador según se describe en [Donovan 02, Pág. 149] son:

## Funciones de un cargador

- Designar espacio en memoria para el programa (asignación)
- Resolver referencias simbólicas entre módulos (ligado)
- Ajustar todas las direcciones del código de acuerdo al espacio disponible en memoria (relocalización)
- Físicamente colocar todas las instrucciones y los datos en la memoria (carga)

## Clases de cargadores

- **Cargador compila y ejecuta**
- **Cargador absoluto**
- **Cargador relocalizable**
- **Cargador de ligado directo**
- **Cargador dinámico**

## **Cargador compila y ejecuta**

- Una posible forma de ejecutar la función de carga es a través de un cargador denominado **cargador** "**compila y ejecuta**", para ello se contempla que en el mismo proceso de compilado las instrucciones generadas se coloquen directamente en memoria. El **cargador consiste** en este caso de una sola instrucción que transfiere el control a la dirección de inicio del nuevo programa.

## Cargador absoluto

- También se encuentran los llamados **cargadores absolutos** este tipo de programa dependen de que se le especifique previamente en donde serán ejecutados los programas en memoria. De esta forma el se encargará de colocar el código en las direcciones que especifique el programa.

## Cargador relocalizable

- Para evitar los problemas que obviamente se tienen al tener direcciones estáticas en el código como son el traslape entre módulos, se tienen los **cargadores relocalizables**. Para este tipo de cargador el ensamblador se encarga como previamente se manejo de proporcionar mayor información en el ejecutable a través de una tabla o vector que contiene entre otros datos, la longitud del programa, la dirección de la primera instrucción, la longitud del mismo vector.

## Cargador de ligado directo

- Una clase particular de cargador que ofrece mayores ventajas se denomina **cargador de ligado directo**. Para que este tipo de cargador funcione adecuadamente el ensamblador debe proveer los siguientes detalles:
- La longitud de los segmentos
- Una lista de los símbolos externos
- Una lista de símbolos no definidos pero referenciados
- Información acerca de las direcciones constantes y que tipo de valores tienen
- El código máquina y las direcciones relativas

## Binder

- En un esquema como el anterior el cargado puede consumir demasiado espacio de la memoria, por ello se puede optar por otro esquema de carga mediante lo que se denomina un **cargador *binder*** tal software funciona idénticamente al cargador de ligado directo con la diferencia de colocar la carga no en memoria, sino en módulos.

## Cargador dinámico

- Un cargador dinámico es útil cuando no es posible asignar en memoria un programa completo, dicho cargador se basa en el binder para operar de esta forma. Cada estructura se coloca dinámicamente en memoria. En este esquema cada módulo se va intercalando en memoria conforme se requiere, ese es el concepto de cargador dinámico.

## Bibliografía


1. [Donovan 72] John Donovan, Systems Programming, McGraw Hill, 1972
2. [Englander 02] Irv Englander, Arquitectura computacional 2da Edición, CECSA, 2002
3. [Jurgens 91] David Jurgens, Help PC 2.10 software de referencia, 1991.
4. [Oney 96] Michael Oney, Systems Programming for Windows 95, Microsoft Press, 1996
5. [Powell 01] Robert Powell, C# and the .NET Framework The C++ perspective, Sams, 2001
6. [Tanenbaum 00] Andrew Tanenbaum, Organización de computadoras un enfoque estructurado, Pearson Education, 2000
7. [Tischer 96] Michael Tischer, PC Interno 5, Marcombo, 1996.
8. [Duran 07] Luis Duran Rodriguez, El gran libro del PC Interno, AlfaOmega, 2007
9. [Conger 92] Conger, James L. , Windows API bible : the definitive programmer's reference, Waite Group Pr

## Tarea

- Definir con sus propias palabras que es una máquina de Turing.
- Investigar los comandos de un editor de comandos y realizar un ejemplo
  - Incluya imágenes de su práctica
- Investigar y presentar diferentes cabeceras de archivos como PE, Coff, MZ, etc.