



Introducción a la Programación y Computación 2



MsC Ing. Estuardo Zapeta

Agenda



Recomendaciones



Perfil Profesional

IT Project Manager & Data Science

Project Management

Provided technical direction on IT projects

Business Intelligence

ETL, Datawarehousing



IT Lecturer & Research

Software Architect

Technical leadership, asset management, strategic planning

Software Development



-UEDI-

<https://uedi.ingenieria.usac.edu.gt/>

-Google Meet-

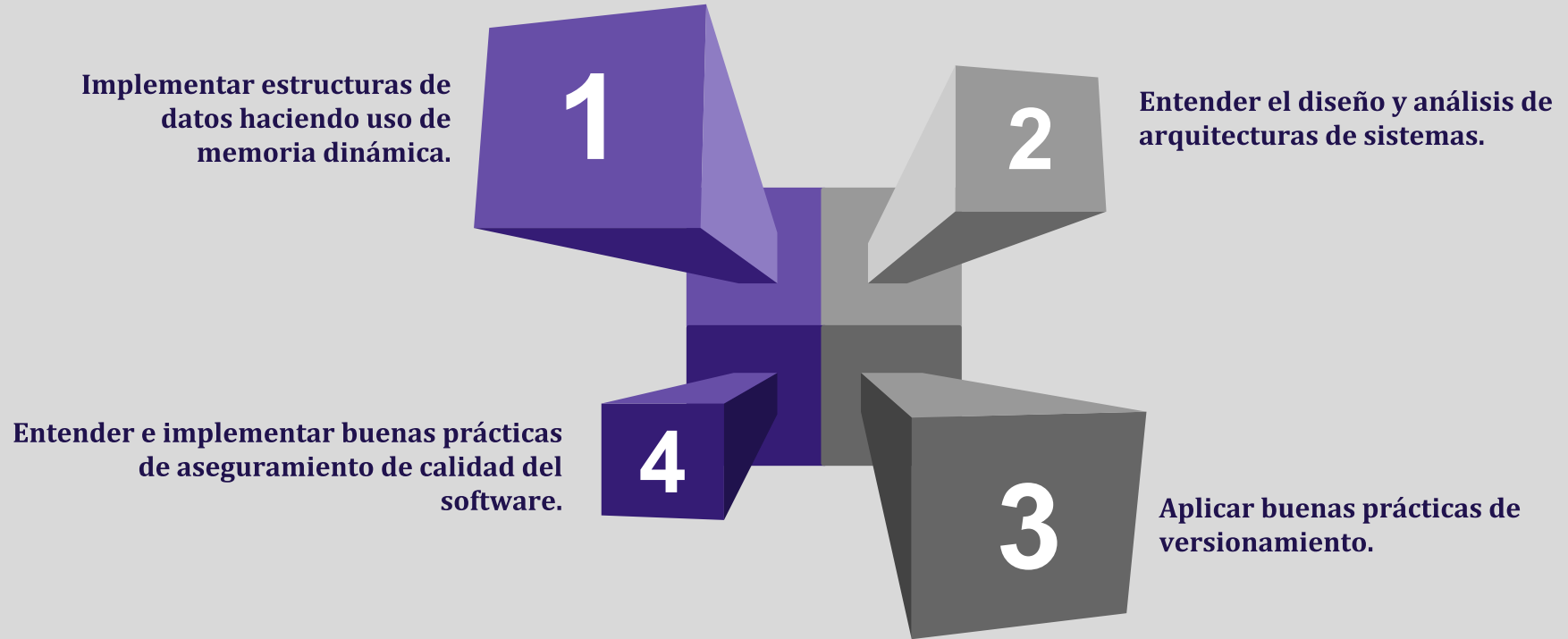
<https://meet.google.com/jpu-jpjt-uxs>

Introducción a la Programación y Computación II

“

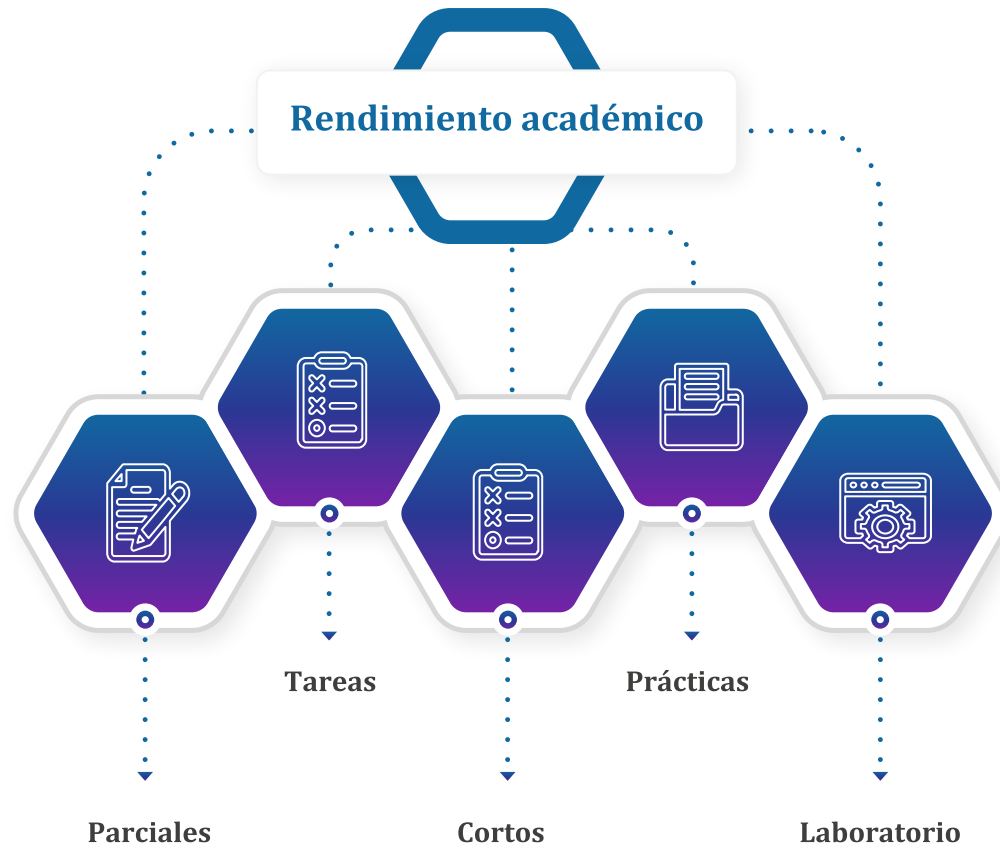
Este curso se encuentra diseñado para que el estudiante entienda e implemente estructuras de datos por medio de memoria dinámica, que conceptualmente entienda el diseño de sistemas a través de la definición de una arquitectura y para finalizar la aplicación de versionamiento durante el desarrollo de software.

Objetivos



Contenido del curso





Sistema de Evaluación



Parciales

1er. Parcial	13 puntos
2do. Parcial	13 puntos
3er. Parcial	13 puntos
Total	39 puntos

Sistema de Evaluación



Actividades

Tareas y/o cortos	03 puntos
Prácticas	03 puntos
Total	06 puntos

Sistema de Evaluación



Laboratorio

Proyecto I	10 puntos
Proyecto II	10 puntos
Proyecto III	10 puntos
Total	30 puntos

Resumen de Sistema de Evaluación



Parciales	39 puntos
Tareas, cortos, prácticas	06 puntos
Proyecto de Laboratorio	30 puntos
Zona	75 puntos
Examen Final	25 puntos

Herramientas que se van a necesitar



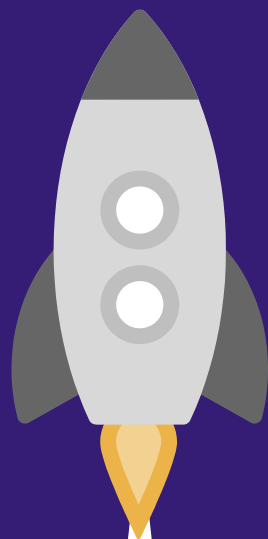
GitHub



Google Colaboratory

Contacto

estuardo.zapeta@gmail.com



Comencemos...



Gestión de la Memoria

El gestor de memoria del sistema operativo debe hacer de puente entre los requisitos de las aplicaciones y los mecanismos que proporciona el hardware de gestión de memoria.

Objetivos del Sistema de Gestión de Memoria

1. Ofrecer a cada proceso un espacio lógico propio

2. Proporcionar protección entre los procesos.

3. Permitir que los procesos compartan memoria.

4. Dar soporte a las distintas regiones del proceso

5. Maximizar el rendimiento del sistema

6. Proporcionar a los procesos mapas de memoria muy grandes





1. Ofrecer a cada proceso un espacio lógico propio

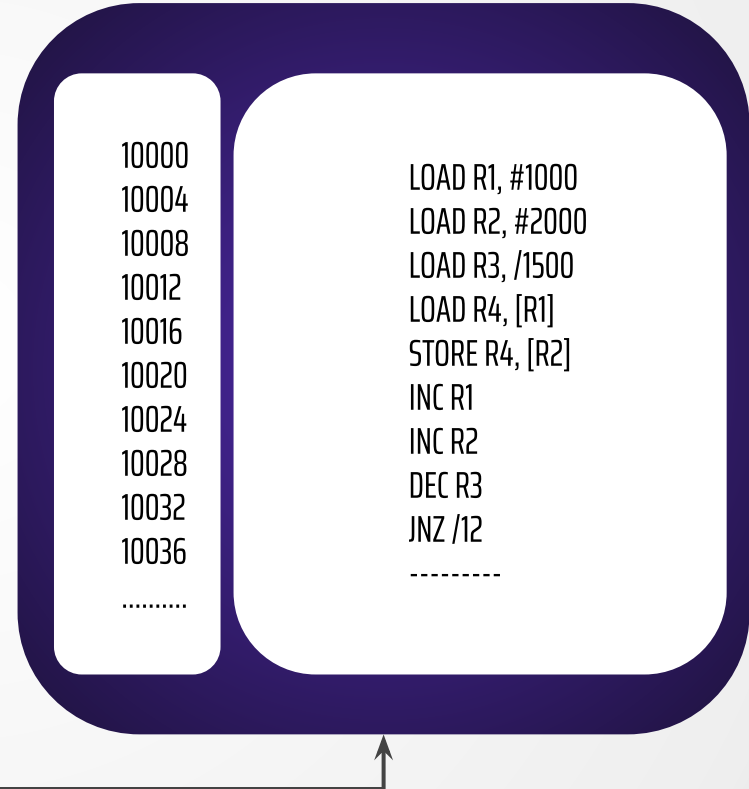
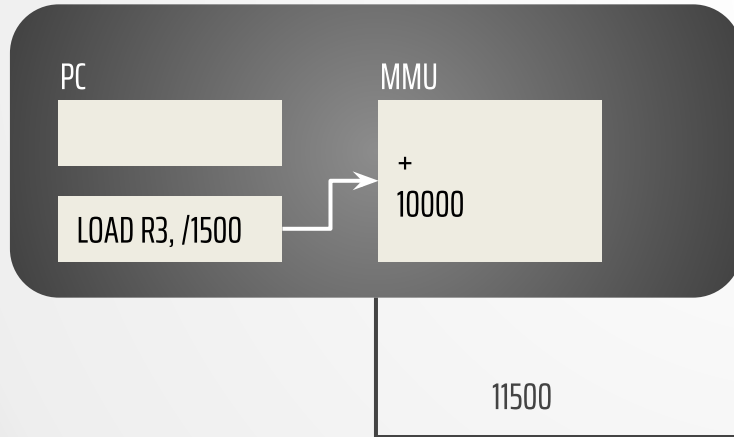
En un sistema operativo multiprogramado no se puede conocer la posición de memoria que ocupará un programa cuando se ejecute, puesto que dependerá del estado de ocupación de la memoria, pudiendo variar, por tanto, en sucesivas ejecuciones del mismo.

Supóngase, por ejemplo, un fragmento de un programa que copia el contenido de un vector almacenado a partir de la dirección 1000 en otro almacenado a partir de la 2000, estando el tamaño del vector almacenado en la dirección 1500. El código almacenado en el archivo ejecutable, mostrado en un lenguaje ensamblador hipotético, sería el mostrado en la Figura.

0	LOAD R1, #1000
4	LOAD R2, #2000
8	LOAD R3, /1500
12	LOAD R4, [R1]
16	STORE R4, [R2]
20	INC R1
24	INC R2
28	DEC R3
32	JNZ /12
36	-----

En un sistema con multiprogramación es necesario realizar un proceso de traducción (**reubicación**) de las direcciones de memoria a las que hacen referencia las instrucciones de un programa (**direcciones lógicas**) para que se correspondan con las direcciones de memoria principal asignadas al mismo (**direcciones físicas**). En el ejemplo planteado previamente, si al programa se le asigna una zona de memoria contigua a partir de la dirección 10000, habría que traducir todas las direcciones que genera el programa añadiéndoles esa cantidad.

UCP





2. Protección

En un sistema con multiprogramación el problema se acentúa ya que no sólo hay que proteger al sistema operativo sino también a los procesos entre sí. El mecanismo de protección en este tipo de sistemas necesita del apoyo del hardware puesto que es necesario validar cada una de las direcciones que genera un programa en tiempo de ejecución.

3. Compartimiento de memoria

Para cumplir el requisito de protección, el sistema operativo debe crear espacios lógicos independientes y disjuntos para los procesos. Sin embargo, en ciertas situaciones, bajo la supervisión y control del sistema operativo, puede ser provechoso que los procesos puedan compartir memoria.



4. Soporte de las regiones del proceso

El mapa de un proceso no es homogéneo sino que está formado por distintos tipos de regiones con diferentes características y propiedades. Dado que el sistema operativo conoce qué regiones incluye el mapa de memoria de cada proceso, el gestor de memoria con el apoyo del hardware debería dar soporte a las características específicas de cada región.





5. Maximizar el rendimiento

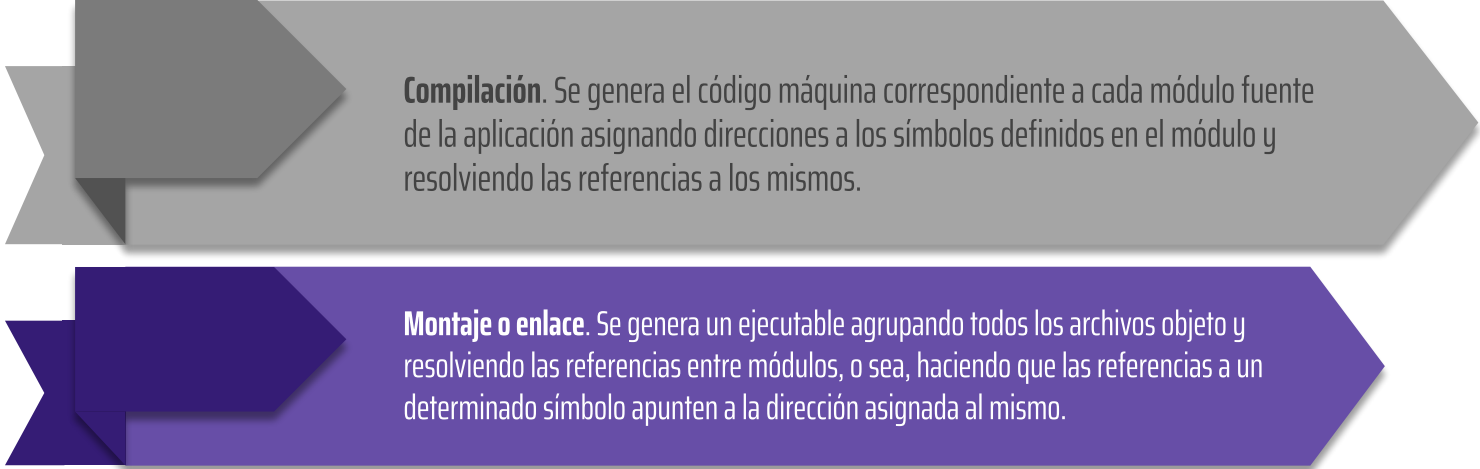
El gestor de memoria debe, realizar un reparto de la memoria entre los procesos intentando que quepa el mayor número de ellos en memoria y minimizando el desperdicio inherente al reparto. Para ello, debe establecerse una política de asignación adecuada. La política de asignación determina qué direcciones de memoria se asignan para satisfacer una determinada petición.



6. Mapas de memoria muy grandes para los procesos

Para esto se ideó la técnica de la memoria virtual, que permite proporcionar a un proceso de forma transparente un mapa de memoria considerablemente mayor que la memoria física existente en el sistema.

Fases de la generación de un ejecutable



Compilación. Se genera el código máquina correspondiente a cada módulo fuente de la aplicación asignando direcciones a los símbolos definidos en el módulo y resolviendo las referencias a los mismos.

Montaje o enlace. Se genera un ejecutable agrupando todos los archivos objeto y resolviendo las referencias entre módulos, o sea, haciendo que las referencias a un determinado símbolo apunten a la dirección asignada al mismo.

Hoja de Trabajo 1

- ¿En qué consiste la técnica de memoria virtual?
- ¿Qué es la memoria de intercambio (swapping)?