

# Presentación de Compiladores

## *Curso 2024/2025*

### **Profesores:**

Eduardo Martínez Graciá (G1/G3)

María Antonia Cárdenas Viedma (G2)

Manuel Gil Pérez (G1/G2/G3)

Ramón García Martínez (G3)

Denisse Misoo Kim (G1)

Dpto. de Ingeniería de la Información y las Comunicaciones

# Índice

## 1 Introducción

- Datos básicos
- Compiladores
- Objetivos

## 2 Guía docente

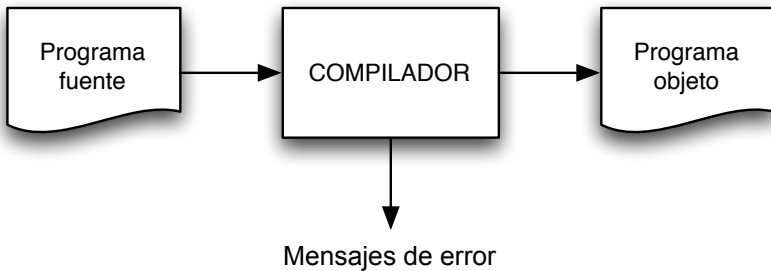
- Temario
- Metodología
- Prácticas
- Calendario académico
- Cronograma
- Evaluación
- Aulas y tutorías
- Bibliografía

## Datos básicos de la asignatura

<b>Asignatura:</b>	<i>Compiladores</i>
<b>Titulación:</b>	Grado en Ingeniería Informática
<b>Carácter:</b>	Obligatoria
<b>Curso:</b>	2º
<b>Temporización:</b>	2º Cuatrimestre
<b>Carga ECTS:</b>	6 créditos (25 horas / crédito)
<b>Requisitos:</b>	Conocimiento de la asignatura <b>Autómatas y Lenguajes Formales</b> . Programación en C.
<b>Descripción:</b>	La asignatura se centra en las técnicas básicas de análisis y síntesis dentro del proceso general de construcción de <b>compiladores</b> . Se pretende que su estudio dé al alumno la capacidad para diseñar un compilador completo para un lenguaje de programación dado, incidiendo en el aprendizaje y uso de las herramientas adecuadas para cada fase del proceso de traducción.

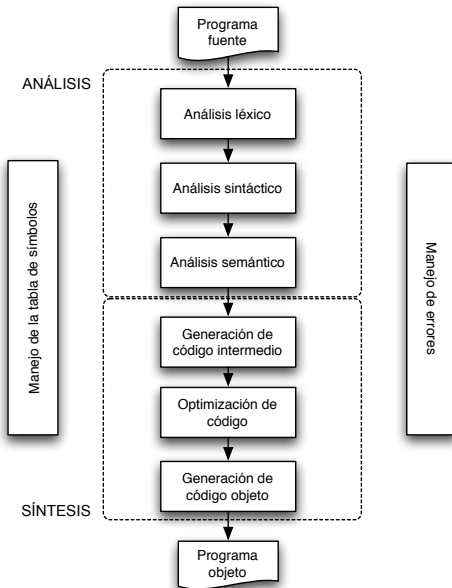
# Compiladores

- Esquema básico de un compilador:



# Fases de un compilador

- División del funcionamiento del compilador en fases:

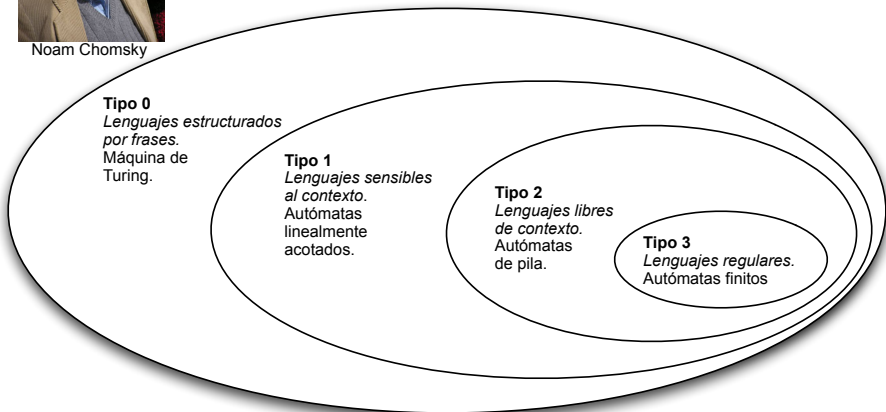


# Tipos de lenguajes



Noam Chomsky

La mayoría de los lenguajes de programación son de tipo 1.



## Objetivos principales de la asignatura

- 1 Reconocer la utilidad de los **modelos formales** subyacentes en el diseño de traductores.
- 2 **Diseñar un compilador** para un lenguaje de programación, haciendo uso de las distintas **técnicas** explicadas y también de las **herramientas automáticas** para generar algunos de los módulos.
- 3 Distinguir las **fases** de traducción de programas, desde el código fuente al ejecutable.
- 4 Razonar acerca de distintas características de los traductores, como **rendimiento, portabilidad y optimización**.
- 5 Conocer la **evolución** de los lenguajes de programación.
- 6 Entender la importancia y el poder de abstracción del concepto de **máquina virtual**.
- 7 Comparar y contrastar los modelos de ejecución **interpretados y compilados**, conociendo ventajas de cada uno.

# Temario

## Teoría

- Tema 1. Traductores e intérpretes.
- Tema 2. Análisis léxico.
- Tema 3. Análisis sintáctico.
- Tema 4. Análisis semántico y traducción dirigida por la sintaxis.
- Repaso.

## Prácticas

- P1: Manejo de la herramienta Flex.
- P2: **Módulo de análisis léxico con Flex.**
- P3: Manejo de la herramienta Bison para el análisis sintáctico.
- P4: Manejo de atributos y acciones semánticas en Bison.
- P5: **Módulo de análisis sintáctico con Bison.**
- P6: **Módulo de análisis semántico con Bison.**
- P7: **Módulo de generación de código con Bison.**



# Metodología

## Trabajo dirigido

- Teoría:
  - Lección magistral.
  - Resolución de ejercicios.
- Prácticas:
  - Laboratorios para la explicación de herramientas (P1, P3 y P4).
  - Resolución de la práctica global por partes (P2, P5, P6 y P7).
- Tutorías (presenciales y online).

## Trabajo autónomo

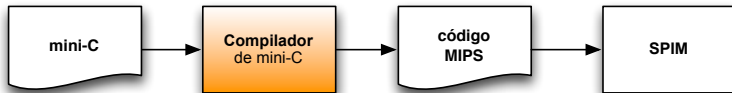
- Estudio de la teoría.
  - Sugerencia: leer los apuntes de cada tema antes de clase.
- Implementación del compilador.

## Datos generales sobre las prácticas

- Grupos de dos alumnos.
- Implementación de un procesador para un lenguaje simplificado.
  - Lenguaje reducido con sintaxis similar a C: **mini-C**.
  - Inicialmente sólo se usan variables enteras.
  - Con un conjunto mínimo de sentencias: `if`, `if-else`, `while`, `read` y `print`.
- Lenguaje de implementación del compilador: **C**.
- Sistema operativo: Linux.
  - Las prácticas deben funcionar en los laboratorios de la Facultad.
- Fases del compilador a implementar:
  - Análisis léxico (**Flex**).
  - Análisis sintáctico (**Bison**).
  - Análisis semántico (**Bison**).
  - Generación de código ensamblador (**Bison**).
  - Simulación del *código ensamblador de MIPS* (**Spim** o **Mars**).

## Descripción de las prácticas (1)

- Esquema de la práctica para el lenguaje **mini-C**:



- El programa compilador ejecutable se genera con **gcc**.
- El código MIPS se simula con el intérprete **Spim** o **Mars**.

## Descripción de las prácticas (2)

### Ejemplo de programa escrito en mini-C

```
void main() {  
    const a = 0, b = 0;  
    var c = 5 + 2 - 2;  
    print "Inicio del programa\n";  
    if (a) print "a","\n";  
    else if (b) print "No a y b\n";  
        else while(c) {  
            print "c = ",c,"\n";  
            c = c - 2 + 1;  
        }  
    print "Final","\n";  
}
```

## Descripción de las prácticas (3)

- El compilador de mini-C genera *código ensamblador de MIPS*

### Ejemplo de código ensamblador

```
# Seccion de datos
.data
$str1:
.asciiz "Inicio del programa\n"
$str2:
.asciiz "a"
$str3:
.asciiz "\n"
$str4:
.asciiz "No a y b\n"
$str5:
.asciiz "c = "
$str6:
.asciiz "\n"
$str7:
.asciiz "Final"
$str8:
.asciiz "\n"
_a:
.word 0
_b:
.word 0
_c:
.word 0
```

```
# Seccion de codigo
.text
.globl main
main:
li $t0, 0
sw $t0, _a
li $t0, 0
sw $t0, _b
li $t0, 5
li $t1, 2
add $t2, $t0, $t1
li $t0, 2
sub $t1, $t2, $t0
sw $t1, _c
la $a0, $str1
li $v0, 4
syscall
lw $t0, _a
beqz $t0, $l5
la $a0, $str2
li $v0, 4
syscall
.
.
.
```

# Calendario académico

Enero	6	7	8	9	10	11	12	CONVOCATORIA DE EXÁMENES
	13	14	15	16	17	18	19	
	20	21	22	23	24 L	25	26	
	27	28	29	30	31	1	2	
Febrero	3	4	5	6	7	8	9	SANTO TOMÁS DE AQUINO/ FECHA TOPE DE ENTREGA ACTAS
	10	11	12	13	14	15	16	
	17	18	19	20	21	22	23	
	24	25	26	27	28	1	2	
Marzo	3	4	5	6	7	8	9	SAN JOSÉ
	10	11	12	13	14	15	16	
	17	18	19	20	21X	22	23	
	24	25	26	27	28	29	30	
Abril	31	1	2	3	4	5	6	SEMANA SANTA Y FIESTAS DE PRIMAVERA DÍA DEL TRABAJO
	7	8	9	10	11 J	12	13	
	14	15	16	17	18	19	20	
	21	22	23	24	25	26	27	
Mayo	28	29	30	1	2	3	4	CONVOCATORIA DE EXÁMENES
	5	6	7	8	9	10	11	
	12	13	14	15	16	17	18	
	19	20	21	22	23	24	25	
Junio	26	27	28	29	30	31	1	FECHA TOPE DE ENTREGA DE ACTAS Día de la Región CONVOCATORIA DE EXÁMENES FECHA TOPE DE ENTREGA DE ACTAS
	2	3	4	5	6	7	8	
	9	10	11	12	13	14	15	
	16	17	18	19	20	21	22	
	23	24	25	26	27	28	29	
	30	1	2	3	4	5	6	

# Cronograma

Semana	Teoría	Prácticas
01	T1	
02	T1T2 <sup>1</sup>	Flex (P1)
03	T2T3	Flex (P1)/ALex (P2)
04	T3	ALex (P2)
05	T3	Bison (P3)
06	T3	Bison (P4)
07	T3	ASint (P5)
08	T3	ASint (P5)
09	T3 <sup>2</sup>	ASem (P6) <sup>2</sup>
10	T3	ASem (P6)
11	T3	GenCod (P7)
12	T4	GenCod (P7)
13	T4	GenCod (P7) <sup>3</sup>
14	Repaso	GenCod (P7)

<sup>1</sup>La teoría del G1 se recupera el 24 de enero.

<sup>2</sup>La teoría del G3 y los laboratorios del G2 y el G3 se recuperan el 21 de marzo.

<sup>3</sup>Los laboratorios del jueves de los grupos G1, G2 y G3 se recuperan el 11 de abril.

# Evaluación

- La **teoría** se evalúa con un examen final en el que podrán aparecer preguntas relacionadas directamente con las prácticas.
- Las **prácticas** se evalúan con la entrega del proyecto final:
  - Este debe cumplir los requisitos mínimos propuestos.
  - Será necesario superar un control para demostrar que se han alcanzado las competencias mínimas, aunque la nota será la obtenida en el traductor.
  - En algunos casos se realizará una entrevista para defender la implementación.
- $\text{Nota final} = \text{Teoría} \times 0,5 + \text{Prácticas} \times 0,5$ .
- Para aprobar la asignatura es necesario aprobar ambas partes.
- Se guardará cada parte hasta la convocatoria de enero **siempre que no se detecte plagio o copia en ninguna de las pruebas de evaluación**.
- Podéis consultar las **fechas de exámenes y entrega de prácticas** en la web de la facultad:

<https://www.um.es/en/web/estudios/grados/informatica/horarios-examenes>



## Consejos para afrontar la asignatura

- La asistencia a clase no es obligatoria, pero sí es muy recomendable.
- Evitar centrarse en usar exámenes resueltos para estudiar.
- Los apuntes son el instrumento principal de estudio.
- Estudiar la lógica de los algoritmos.
- Es recomendable asistir a las tutorías no sólo al final del curso.
- Es muy recomendable ser disciplinado con la cronología de las prácticas.

# Horarios

## Grupo 1 - PCEO

- Teoría: lunes de 9:00h a 11:00h (*Eduardo Martínez*)
- Prácticas:
  - Subgrupo 1.1: Lab. 1.4, jueves de 9:00h a 10:40h (*Manuel Gil*)
  - Subgrupo 1.2: Lab. 1.5, jueves de 12:20h a 14:00h (*Denisse Misoo Kim*)
  - Subgrupo 1.3: Lab. 2.3, jueves de 17:10h a 18:50h (*Eduardo Martínez*)

## Grupo 2

- Teoría: martes de 9:00h a 11:00h (*M<sup>a</sup> Antonia Cárdenas*)
- Prácticas:
  - Subgrupo 2.1: Lab. 2.1, jueves de 9:00h a 10:40h (*M<sup>a</sup> Antonia Cárdenas*)
  - Subgrupo 2.2: Lab. 2.2, miércoles de 12:20h a 14:00h (*M<sup>a</sup> Antonia Cárdenas*)
  - Subgrupo 2.3: Lab. 2.3, martes de 12:20h a 14:00h (*M<sup>a</sup> Antonia Cárdenas*)
  - Subgrupo 2.4: Lab. 1.6, jueves de 10:40h a 12:20h (*Manuel Gil*)

## Grupo 3

- Teoría: miércoles de 15:30h a 17:30h (*Eduardo Martínez*)
- Prácticas:
  - Subgrupo 3.1: Lab. 1.4, jueves de 18:50h a 20:30h (*Manuel Gil*)
  - Subgrupo 3.2: Lab. 1.5, miércoles de 18:50h a 20:30h (*Ramón García*)
  - Subgrupo 3.3: Lab. 1.6, jueves 17:10h a 18:50h (*Manuel Gil*)
  - Subgrupo 3.4: Lab. 2.2, martes 18:50h a 20:30h (*Ramón García*)

# Tutorías

## Eduardo Martínez Graciá

- Lunes 16:00h-19:00h.
- Email: [edumart@um.es](mailto:edumart@um.es).

## María Antonia Cárdenas Viedma

- Martes 11:00h-12:00h.
- Miércoles 10:00h-12:00h.
- Email: [mariancv@um.es](mailto:mariancv@um.es).

## Manuel Gil Pérez

- Lunes 11:30h-13:00h
- Miércoles 17:00h-18:30h.
- Email: [mgilperez@um.es](mailto:mgilperez@um.es).

## Ramón García Martínez

- Miércoles 16:30h-18:30h
- Jueves 17:30h-18:30h.
- Email: [ramongar@um.es](mailto:ramongar@um.es).

## Denisse Misoo Kim

- Miércoles 11:30h-13:30h
- Jueves 10:00h-11:00h.
- Email: [denissemisoo.kim@um.es](mailto:denissemisoo.kim@um.es).

# Bibliografía



Alfred Aho

Jeffrey Ullman

Ravi Sethi

- Bibliografía **básica**:
  - A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman. *Compiladores: principios, técnicas y herramientas*. 2008.
  - Apuntes de la asignatura en el Aula Virtual.
  - C. Donnelly y R. Stallman. *Bison. The Yacc-compatible parser generator, v2.4.2*. 2010.
  - V. Paxson. *Lexical Analysis With Flex*. 2007.
- Bibliografía **extendida**:
  - 15 entradas bibliográficas adicionales.
  - Consultar la guía docente en el Aula Virtual.