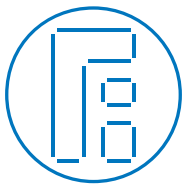
Presentación de Compiladores

Curso 2025/2026

Profesores:
Eduardo Martínez Graciá (G1/G3)
María Antonia Cárdenas Viedma (G2)
Manuel Gil Pérez (G2/G3)
Sergio López Bernal (G3)
Pedro Beltrán López (G3)

Dpto. de Ingeniería de la Información y las Comunicaciones

UNIVERSIDAD DE MURCIA



Facultad de Informática

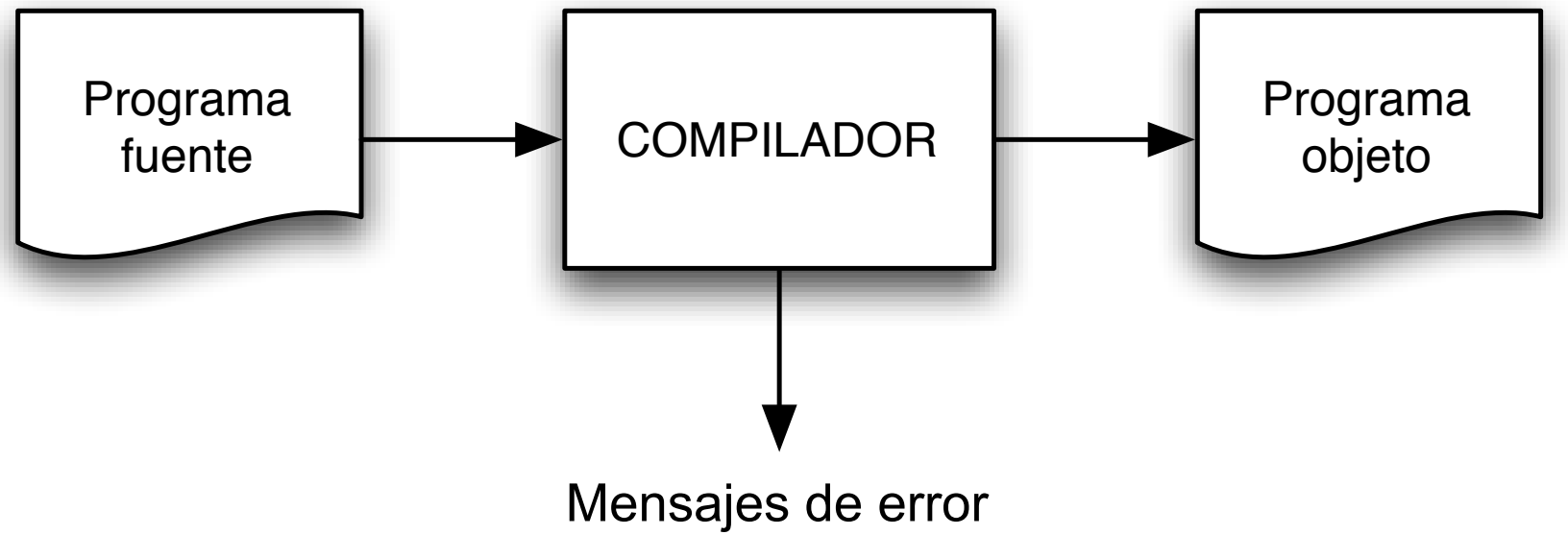
Índice

Datos básicos de la asignatura

Asignatura:	*Compiladores*
Titulación:	Grado en Ingeniería Informática
Carácter:	Obligatoria
Curso:	2º
Temporización:	2º Cuatrimestre
Carga ECTS:	6 créditos (25 horas / crédito)
Requisitos:	Conocimiento de la asignatura **Autómatas y Lenguajes Formales**. Programación en C.
Descripción:	La asignatura se centra en las técnicas básicas de análisis y síntesis dentro del proceso general de construcción de **compiladores**. 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.

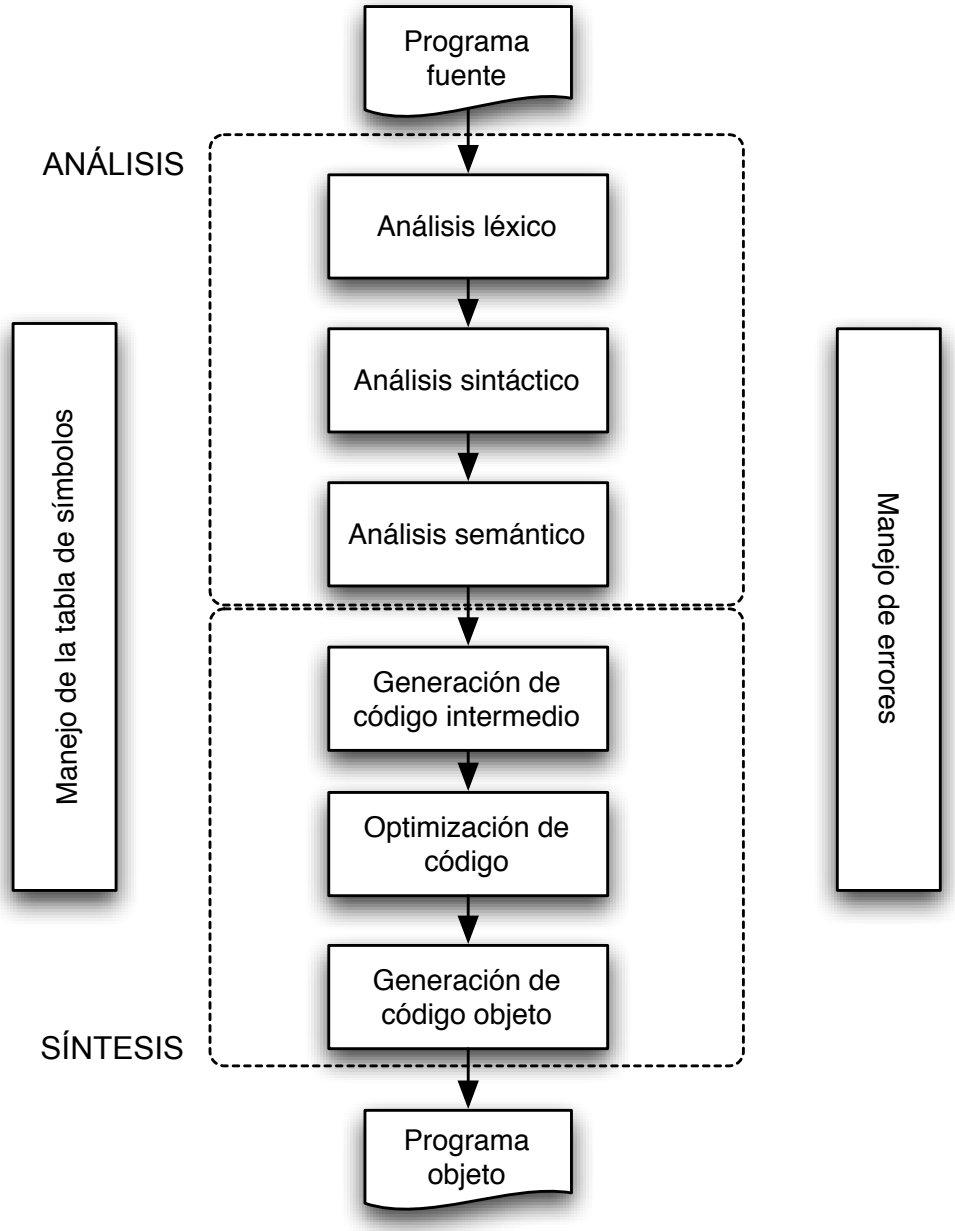
Compiladores

- Esquema básico de un compilador:

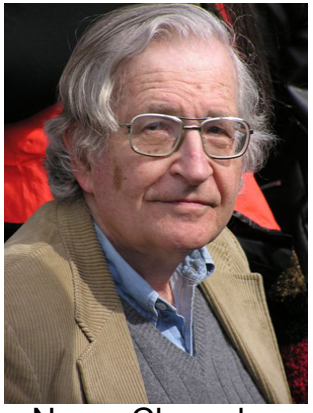


Fases de un compilador

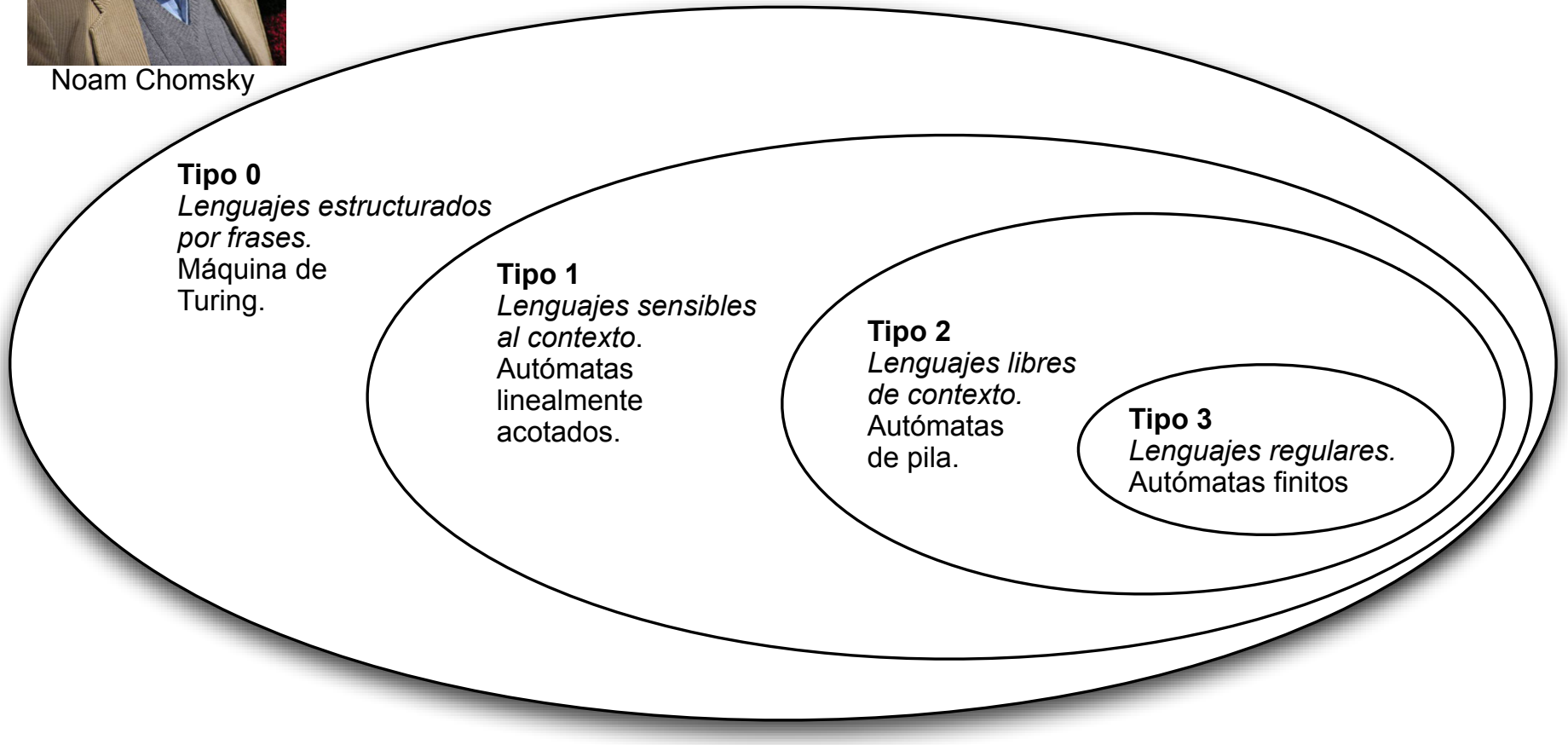
- Division del funcionamiento del compilador en fases:



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

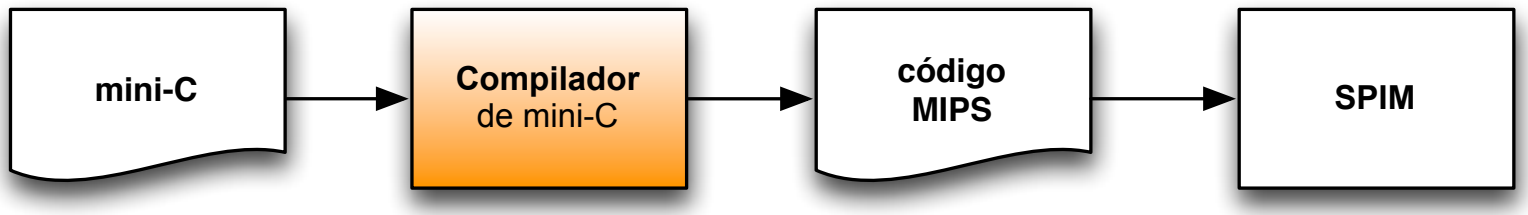
- Estudio de la teoría.
  - Sugerencia: leer los apuntes de cada tema antes de clase.
- Implementación del compilador.

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

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

Cronograma

Semana	Teoría	Prácticas
01	T1	
02	T1T2	Flex (P1)[1]
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[2]	ASem (P6)[2]
10	T3	ASem (P6)
11	T3	GenCod (P7)
12	T4	GenCod (P7)
13	T4	GenCod (P7)
14	Repaso	GenCod (P7)

[1] La teoría del grupo 3 y los laboratorios de los subgrupos 2.2 y 3.2 se recuperan el 30 de enero.

[2] Los laboratorios de los subgrupos 1.1, 1.2, 1.3, 1.4, 2.1, 2.4, 3.1 y 3.3 se recuperan el 13 de marzo.

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.
- Nota final = Teoría x 0,5 + Prácticas x 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 *(Eduardo Martínez)*
  - Subgrupo 1.2: Lab. 1.5, jueves de 12:20h a 14:00h *(Eduardo Martínez)*
  - Subgrupo 1.3: Lab. 2.3, jueves de 17:10h a 18:50h *(Eduardo Martínez)*
  - Subgrupo 1.4: Lab. 2.1, jueves de 18:50h a 20:30h *(Eduardo Martínez)*

Grupo 2

- Teoría: martes de 9:00h a 11:00h *(Mª Antonia Cárdenas)*
- Prácticas:
  - Subgrupo 2.1: Lab. 2.1, jueves de 9:00h a 10:40h *(Mª Antonia Cárdenas)*
  - Subgrupo 2.2: Lab. 2.2, miércoles de 12:20h a 14:00h *(Mª Antonia Cárdenas)*
  - Subgrupo 2.3: Lab. 2.3, martes de 12:20h a 14:00h *(Mª 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 *(Sergio López Bernal)*
  - Subgrupo 3.2: Lab. 1.5, miércoles de 18:50h a 20:30h *(Sergio López Bernal)*
  - 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 *(Manuel Gil)*

Tutorías

Eduardo Martínez Graciá

→ Martes 16:00h-19:00h.

- Email: edumart@um.es.

María Antonia Cárdenas Viedma

→ Martes 11:00h-12:00h.

→ Miércoles 10:00h-12:00h.

- Email: mariancv@um.es.

Manuel Gil Pérez

→ Lunes 11:30h-13:00h

→ Miércoles 17:00h-18:30h.

- Email: mgilperez@um.es.

Sergio López Bernal

→ Miércoles 10:00h-13:00h.

- Email: slopez@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.