GUÍA DOCENTE

PROGRAMACIÓN DE BAJO NIVEL

GRADO EN INGENIERÍA DEL SOFTWARE

MODALIDAD: PRESENCIAL

CURSO ACADÉMICO: 2026-2027

Denominación de la asignatura:	Programación de bajo nivel
Titulación:	Grado en Ingeniería del Software
Facultad o Centro:	Centro Universitario de Tecnología y Arte Digital
Materia:	Optatividad
Curso:	4
Cuatrimestre:	2
Carácter:	OP
Créditos ECTS:	3
Modalidad/es de enseñanza:	Presencial
Idioma:	Castellano
Profesor/a - email	Marcos Novalbos Mendiguchía / marcos.novalbos@u-tad.com
Página Web:	http://www.u-tad.com/

DESCRIPCIÓN DE LA ASIGNATURA

Descripción de la materia

Esta materia recoge algunos contenidos avanzados y/o especializados que pudiera requerir un ingeniero del software generalista

Descripción de la asignatura

Esta asignatura es relevante para adquirir conocimientos de programación orientada a hardware específico, resaltando la importancia de la eficiencia de código, el acceso directo a memoria y la gestión de recursos

COMPETENCIAS Y RESULTADOS DE APRENDIZAJE DE LA MATERIA

Competencias (genéricas, específicas y transversales)

COMPETIENCIAS BÁSICAS Y GENERALES

CB1: Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.

CB2: Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.

CB3: Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.

CB4: Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.

CB5: Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

CG1: Entender, planificar y resolver problemas a través del desarrollo de soluciones informáticas.

CG2: Desarrollar soluciones informáticas que sean respetuosas con el medio ambiente, los deberes sociales y recursos naturales, además de cumplir con la legislación y ética.

CG3: Aplicar los fundamentos científicos para la resolución de problemas informáticos

CG4: Entender la complejidad, simplificar y optimizar los sistemas informáticos

CG6: Trabajar en entornos de trabajo multidisciplinares demostrando capacidad de trabajo en equipo, versatilidad, flexibilidad, creatividad y respeto por el trabajo de los compañeros de otras áreas.

CG7: Aplicar los fundamentos creativos de generación de ideas en los proyectos de desarrollo software para entornos digitales.

CG9: Aprender, modificar y producir nuevas tecnologías informáticas

CG10: Aplicar las técnicas creativas para la realización de proyectos informáticos

CG11: Buscar, analizar y gestionar la información para poder extraer conocimiento de la misma.

COMPETENCIAS ESPECÍFICAS

CE10: Generar documentación de una aplicación de forma automática así como entender y manejar adecuadamente un gestor de versiones de código las que utilizan principios de ingeniería inversa.

CE15: Desarrollar aplicaciones distribuidas teniendo en cuenta la toleracia de los fallos, la adaptabilidad, el balance de carga y la predictividad del sistema.

CE17: Desarrollar aplicaciones que utilicen las características de paralelización de tarjetas gráficas y arquitecturas de altas prestaciones.

CE20: Testar en profundidad el funcionamiento y funcionalidad de una aplicación informática, elaborando planes de pruebas y empleando técnicas de diseño y programación orientado a las pruebas.

CE21: Evaluar la calidad de una aplicación informática desde el punto de vista de su diseño e implementación, aplicando métricas, procedimientos y estándares de medición de calidad del software.

COMPETENCIAS TRANSVERSALES

CT1: Conocer la definición y el alcance, así como poner en práctica los fundamentos de las metodologías de gestión de proyectos de desarrollo tecnológico.

CT2: Conocer los principales agentes del sector y el ciclo de vida completo de un proyecto en desarrollo y comercialización de contenidos digitales

CT4: Actualizar el conocimiento adquirido en el manejo de herramientas y tecnologías digitales en función del estado actual del sector y de las tecnologías empleadas.

CT5: Poseer las habilidades necesarias para el emprendimiento digital.

Resultados de aprendizaje

Al acabar la titulación, el graduado o graduada será capaz de:

- Entender el ciclo de aseguramiento de la calidad del software

- Diseñar un plan de prueba de software

- Conocer los entornos de prueba más habituales de la industria

- Desarrollar una aplicación intensiva en el uso de GPU

- Ser capaz de medir rendimiento en aplicaciones distribuidas.

CONTENIDO

Programación concurrente sobre la Unidad de Procesamiento Gráfico

Profiling de sistemas distribuidos

TEMARIO

Tema 1: Introducción a lenguajes de bajo nivel

• Repaso C/C++

• Repaso a arquitecturas HW

Tema 2A: Uso de ficheros "binarios"

• Implementación de librería de lectura para un formato dado

Tema 2B: Juegos de instrucciones vectorial

• Uso de juegos de instrucciones vectorial para X86_64: AVX/SSE

Tema 3: Coprocesadores

• Introducción a paralelismo

• Uso de GPUs como coprocesador: NVidia CUDA

Tema 4: Driver Linux

• Crear un driver para sistema de ficheros "simple"

ACTIVIDADES FORMATIVAS Y METODOLOGÍAS DOCENTES

Actividades formativas

Actividad Formativa	Horas totales	Horas presenciales
Clases teóricas / Expositivas	16	16
Clases Prácticas	11	11
Tutorías	2	1
Estudio independiente y trabajo autónomo del alumno	25	0
Elaboración de trabajos (en grupo o individuales)	18	0
Actividades de Evaluación	3	3

Metodologías docentes

Método expositivo o lección magistral

Aprendizaje de casos

Aprendizaje basado en la resolución de problemas

Aprendizaje cooperativo o colaborativo

Aprendizaje por indagación

Metodología Flipped classroom o aula invertida

Gamificación

Just in time Teaching (JITT) o aula a tiempo

Método expositivo o lección magistral

Método del caso

Aprendizaje basado en la resolución de problemas

Aprendizaje cooperativo o colaborativo

Aprendizaje por indagación

Metodología flipped classroom o aula invertida

Gamificación

DESARROLLO TEMPORAL

UNIDADES DIDÁCTICAS / TEMAS PERÍODO TEMPORAL

Tema 1: Introducción a lenguajes de bajo nivel

1 semanas

Tema 2A: Uso de ficheros "binarios"

2 semanas

Tema 2B: Juegos de instrucciones vectorial

2 semanas

Tema 3: Coprocesadores

3 semanas

Tema 4: Driver Linux

2 semana

SISTEMA DE EVALUACIÓN

ACTIVIDAD DE EVALUACIÓN	VALORACIÓN MÍNIMA RESPECTO A LA CALIFICACIÓN FINAL (%)	VALORACIÓN MÁXIMA RESPECTO A LA CALIFICACIÓN FINAL (%)
Evaluación de la participación en clase, en prácticas o en proyectos de la asignatura	10	30

Evaluación de trabajos, proyectos, informes, memorias	30	60
Prueba Objetiva	30	60

CRITERIOS ESPECÍFICOS DE EVALUACIÓN

ACTIVIDAD DE EVALUACIÓN	CONVOCATORIA ORDINARIA	CONVOCATORIA EXTRAORDINARIA
Evaluación de la participación en clase, en prácticas o en proyectos de la asignatura	10	10
Evaluación de trabajos, proyectos, informes, memorias	60	60
Prueba Objetiva	30	30

Consideraciones generales acerca de la evaluación

La evaluación de la participación en clase, en prácticas o en proyectos de la asignatura se realizará a partir de la asistencia y la participación activa en clase y la entrega de ejercicios no obligatorios. Eso incluirá la entrega de trabajos obligatorios a tiempo y de trabajos no obligatorios. Este aspecto representará el 10% de la calificación final de la asignatura en la convocatoria ordinaria.

A lo largo del curso se plantearán actividades, ejercicios y problemas que deberán ser entregadas antes de la fecha indicada a través de la plataforma virtual. Cada trabajo obligatorio tendrá asociado un examen para obtener la nota final del mismo. La media de las notas de cada trabajo supondrá el 60% de la calificación final de la asignatura en la convocatoria ordinaria.

No se admitirán trabajos fuera de forma y fecha sin causa justificada. En caso de que la nota media de los trabajos pedidos junto con la nota de examen final no alcancen el aprobado en ordinaria, se deberá repetir los trabajos suspendidos en extraordinaria para obtener una nota media mínima de 5.0.

Para aprobar la asignatura en la convocatoria ordinaria, es imprescindible que la nota media final (60% ejercicios +30% examen + 10% asistencia y participación) sea al menos 5.0 (sobre 10) En caso de no cumplirse alguno de estos requisitos, la asignatura se considerará automáticamente suspensa independientemente del resto de calificaciones.

En caso de no conseguir el aprobado en la convocatoria ordinaria, el alumno podrá presentarse a la convocatoria extraordinaria. Se realizará un examen final que estará dividido en las siguientes partes:

- 30% De examen correspondiente a la prueba objetiva suspenso

- 70% De examen correspondiente a pruebas de trabajos de clase suspensos

El alumno sólo deberá presentarse y superar las partes del examen de extraordinaria que fueron suspendidas en ordinaria.

Para aprobar la asignatura en la convocatoria extraordinaria, es imprescindible que la nota media final sea al menos 5.0 (sobre 10).

-En convocatoria extraordinaria se guardará la nota de todo lo entregado hasta el momento que esté aprobado

En los exámenes no se permite el uso de apuntes ni de calculadoras científicas programables, para lo que el alumno debe remitirse a las instrucciones específicas del profesor sobre este tema. Tampoco está permitido el uso de chatgpt u otras herramientas de generación de código automático.

Todo el código y trabajos entregados por los alumnos deberán ser ORIGINALES. Quiere decir que deberán haber sido desarrollados por los alumnos a lo largo de la asignatura, sin ayuda externa. En caso de usar código/librerías externas a lo suministrado por el profesor, deberá estar debidamente documentado y justificado. Se permite consultar documentación externa a la asignatura, pero el código entregado por el alumno deberá respetar las leyes de copyright y licencias software vigentes. En todo caso, el alumno deberá ser capaz de explicar el código usado y entregado durante el curso.

Copias entre trabajos: Se entenderá como copia de trabajo aquellos proyectos que contengan partes iguales o muy similares, que no cumplan las reglas establecidas en los párrafos anteriores. Las copias de trabajos conllevarán la completa suspensión de la asignatura, sin posibilidad de recuperación en la convocatoria actual. Será el profesor el que decida la gravedad de la copia, y la decisión final podrá ser consultada y revocada por el resto del equipo docente en caso de necesitar una segunda opinión.

No se conservarán calificaciones de ningún tipo entre distintos cursos académicos, únicamente entre convocatorias ordinaria y extraordinaria del mismo curso.

No está permitido el uso de teléfonos móviles en el aula durante el período de evaluación continua, excepto indicación expresa en sentido contrario del profesor.

BIBLIOGRAFÍA / WEBGRAFÍA

Bibliografía Básica:

- ADDISON WESLEY Ed 1(2010); CUDA by Example

Bibliografía Recomendada:

- PRENTICE HALL Ed. 0006 (2010) Assembly Language for X86

MATERIALES, SOFTWARE Y HERRAMIENTAS NECESARIAS

Tipología del aula

Aula teórica

Equipo de proyección y pizarra

Materiales:

Ordenadores con sistema operativo Linux y tarjetas gráficas NVidia serie GTX400 o superior.

No valen Máquinas virtuales, es necesario un Linux “nativo” para poder realizar todas las prácticas

Software:

Sistema Operativo Linux