Propuesta docente para la asignatura Organización de Computadoras - Lic. en Ciencias de la Computación

Claudio Zanellato, Javier Balladini

Facultad de Informática, Universidad Nacional del Comahue.
Neuquén, Argentina
{claudio.zanellato, javier.balladini}@fi.uncoma.edu.ar

Resumen Este artículo presenta las experiencias en la asignatura "Organización De Computadoras" correspondiente al grado en Licenciatura en Ciencias de la Computación. La asignatura estudia principalmente la frontera software-hardware de las computadoras, su estructura básica, funcionamiento y programación en lenguaje ensamblador. El objetivo de este trabajo es mostrar la propuesta didáctica empleada, que intenta solucionar los inconvenientes presentados por la metodología previamente utilizada: ausencia de relación entre la programación de alto y bajo nivel que producía problemas al alumno para comprender otras asignaturas, y dificultad para detectar las causas de los errores que los alumnos cometían en los exámenes. La solución se basa en el diseño de actividades de enseñanza y aprendizaje que promueven la comparación de ambos niveles de programación, y una novedosa metodología de evaluación con exámenes que intentan recrear un ambiente de trabajo profesional con múltiple información a disposición. Los resultados indican una mejoría en la calidad del aprendizaje alcanzado por los alumnos.
Palabras Claves: Estructura de Computadoras, Organización de Computadoras, Aprendizaje constructivo, Metodología de evaluación.

Abstract This paper presents the experiences of teaching "Computer Organization" course for the Bachelor degree in Computer Science. The subject focuses mainly at software-hardware boundary of computers, their basic structure, internal operation, and assembly language programming. The objective of this work is to show the academic methodology, which attempts to solve the problems presented by the methodology used in previous course instances: absence of relationship between the high level programming and low level programming that cause problems to the students to understand other subjects, and difficulty in detecting the causes of the errors made by students in exams. The solution is based on the design of teaching and learning activities that promote comparing both programming levels, and a novel evaluation methodology through exams that attempt to recreate a professional work environment with multiple information available. The results show an improvement of the learning quality achieved by students.
Keywords: Structure of Computers, Computer Organization, Constructive Learning, Evaluation Methodology.

1. Introducción

La Facultad de Informática de la Universidad Nacional del Comahue está ubicada en la ciudad de Neuquén, Argentina. Una de las carreras de esta facultad es la Licenciatura en Ciencias de la Computación, que incluye en el cuarto semestre la asignatura denominada "Organización de computadoras" (ODC). Esta asignatura es impartida por el Departamento de Ingeniería de Computadoras, y su objetivo (de la asignatura), según la modificación del año 2010 del plan de estudios [1], es lograr que el alumno:

- Desarrolle la comprensión del diseño y construcción de un sistema de cómputo.
- Sea capaz de centrar la atención en la frontera software-hardware, explorando los niveles del hardware conectados a este punto de contacto.
- Comprenda el concepto de programa almacenado, la representación de las instrucciones en la memoria de una computadora y los pasos para su ejecución.
- Incorpore temas de diseño básicos y elementos en técnicas digitales.
- Introduzca los conceptos de programación en lenguaje ensamblador.

Previamente, en esta asignatura, se enseñaba lenguaje C en un módulo complementario, cuyo objetivo era proveer al alumno de conocimientos básicos del lenguaje para la realización de trabajos prácticos de asignaturas consecuentes como Sistemas Operativos, y Redes y Teleprocesamiento. El módulo complementario tenía una relación prácticamente nula con los contenidos de la asignatura, y normalmente era impartido por un profesor diferente al del módulo principal. Esta metodología desaprovechaba oportunidades para reafirmar conceptos teóricos por falta de sincronización entre ambos módulos, e impedía a los alumnos establecer relaciones entre la programación de alto nivel y la de bajo nivel (lenguaje ensamblador). Esta falta de relación no solo dificulta el aprendizaje sino que también produce, en general, que los alumnos adquieran un conocimiento totalmente abstracto de la programación de alto nivel. Esto es perjudicial para comprender problemas de rendimiento de las aplicaciones, que se estudia en asignaturas posteriores como Arquitectura de Computadoras y Sistemas Paralelos (asignatura electiva).

La metodología de evaluación utilizada anteriormente en esta asignatura (para ambos módulos: principal y complementario), consistía en exámenes escritos en papel donde el alumno no podía contar con ningún otro material que no sea el manual del lenguaje ensamblador utilizado. Este sistema de evaluación impedía al alumno depurar sus programas en una computadora y verificar que su ejecución era correcta, y además, impedía a los profesores, que corregían los exámenes, poder detectar si una equivocación del alumno en la resolución de un ejercicio era por simple distracción o por algún fallo conceptual.

El objetivo de este trabajo es mostrar la propuesta didáctica empleada en la asignatura ODC que intenta solucionar la problemática planteada con la propuesta metodológica previa de la asignatura, en relación a la división de la asignatura en dos módulos, principal y complementario (lenguaje C), y mejorar el

sistema de evaluación con respecto a la detección del origen de los errores cometidos por los alumnos en los exámenes.

El presente trabajo se organiza de la siguiente manera. Primero, se presenta el plan de estudios de la carrera y sus incumbencias profesionales, y se fundamenta la ubicación de la asignatura ODC en el plan. En segundo lugar, se expone la propuesta metodológica de la asignatura ODC, planteando los contenidos mínimos y extendidos del programa, los recursos necesarios para impartir las clases, la cantidad y distribución de las horas requeridas de clases y fuera de clases, la metodología utilizada para las clases teóricas y prácticas, una descripción de las actividades prácticas y herramientas de software utilizadas, y la metodología de evaluación y forma de acreditación de la asignatura. En tercer lugar, se detallan los resultados alcanzados empleando la metodología propuesta. Por último, se presentan las conclusiones del trabajo.

2. Plan de estudios, modalidad de acreditación, y fundamentación de la asignatura

La carrera Licenciatura en Ciencias de la Computación, dictada por la Facultad de Informática, está actualmente regida por las ordenanzas del Consejo Superior: 1004/98, 075/2010, 0647/2010 y 0235/2011. Según ellas, el egresado de la carrera debe ser un profesional capacitado para: "Actuar profesionalmente tanto en industrias como en organismos nacionales y privados de todo el país", "Planificar, dirigir y auditar Proyectos de Desarrollo de Software de cualquier escala", "Integrar y aplicar los conocimientos científicos del área", "Diseñar, desarrollar y mantener programas básicos y de aplicación (software)", "Evaluar y poner en funcionamiento el software ya desarrollado", "Efectuar estudios técnico–computacionales de proyectos que involucren uso de computadoras" y "Promover las aplicaciones de la informática a nuevas áreas".

El alumno logra la **acreditación de una asignatura** de tres formas diferentes. Una opción es matriculándose para cursar la asignatura, y superar su evaluación, estipulada por la cátedra (el equipo académico que imparte la asignatura), con una calificación mínima de 4. Una calificación podría comprender cualquier valor entre 0 y 10, y cada cátedra determina qué porcentaje de los exámenes debe estar correcto para alcanzar la calificación mínima de 4 (normalmente ronda el 70 %). Si el alumno no alcanza ese porcentaje para alcanzar la calificación de 4, aún podría obtener una **acreditación parcial** y temporal (2 años de duración) de la asignatura si superase otro porcentaje menor al anterior (por ejemplo: 50 %), también establecido por la cátedra. Esta acreditación parcial podría permitir al alumno cubrir requisitos para matricularse en otra asignatura, y también permite el acceso a la acreditación total de la asignatura mediante nuevas evaluaciones.

Las restantes dos opciones para obtener la acreditación (total) de la asignatura son: (1) presentándose directamente a una evaluación completa de la asignatura (modalidad que denominamos examen libre), y (2) mediante la acep-

tación por equivalencia de contenidos de una asignatura acreditada por otra institución universitaria.

La figura 1 muestra el plan de estudios de la carrera, donde para cada asignatura se indican los requisitos para matricularse, que se indican como "Cursada" cuando se requiere tener la acreditación parcial de asignaturas, y como "Aprobada" cuando se requiere tener la acreditación total de asignaturas. La asignatura ODC, está indicada con el número 11, y para matricularse el alumno debe cumplir con el requisito de haber "cursado" las asignaturas de Matemática Discreta, y Estructuras de Datos y Algoritmos, y "aprobada" la asignatura Elementos de Programación; estas relaciones pueden observarse en la figura 2. A su vez, ODC es requisito directo para poder matricularse en dos asignaturas que dicta el Departamento de Ingeniería de Computadoras: Arquitectura de Computadoras, y Sistemas Operativos.

	Primer Cuatrimestre				Segundo Cuatrimestre		
#	Nombre	Cursada	Aprobada	#	Nombre	Cursada	Aprobada
PRIMER AÑO							
1	ELEMENTOS DE ALGEBRA			3	ANALISIS MATEMATICO I		
2	RESOLUCION DE PROBLEMAS Y ALGORITMOS			4	MATEMATICA DISCRETA	1	
				5	ELEMENTOS DE PROGRAMACION	1 – 2	
SEGUNDO AÑO							
6	FUNDAMENTOS DE CS DE LA COMPUTACION	4 – 5		9	ANALISIS MATEMATICO II		3
7	ESTRUCTURA DE DATOS Y ALGORITMOS	3 – 5	2	10	PROGRAMACION ORIENTADA A OBJETOS	7	5
8	INGLES TECNICO			11	ORGANIZACIÓN DE COMPUTADORAS	4 – 7	5
TERCER AÑO							
12	ARQUITECTURA DE COMPUTADORAS	6 – 11	8	15	PROBABILIDAD Y ESTADISTICA	9	8
13	ANALISIS Y DISEÑO DE SISTEMAS	10	7 – 8	16	SISTEMAS OPERATIVOS	12	11
14	LOGICA PARA CS DE LA COMPUTACIÓN	7	6 – 8	17	TEORIA Y DISEÑO DE BASES DE DATOS	13-14	
CUARTO AÑO							
18	DESARROLLO DE SOFTWARE	15-17	13	21	INTELIGENCIA ARTIFICIAL		14
19	LENGUAJES DE PROGRAMACION	16-17	14	22	ADMINISTRACION Y GESTION DE PROYECTOS	18	
20	REDES Y TELEPROCESAMIENTO	16		23	COMPILADORES E INTERPRETES	19	
QUINTO AÑO							
24	ALGORITMOS Y COMPLEJIDAD	16		26	TESIS LICENCIATURA		
25	OPTATIVA I *			27	OPTATIVA II *		

Figura 1. Plan de estudios de la Licenciatura en Ciencias de la Computación

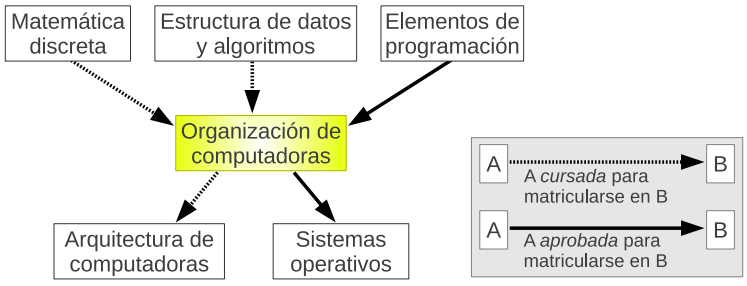

Figura 2. Interrelación de la asignatura objeto y afines

ODC cumple una función de suma importancia en la formación de los alumnos ya que es la primera asignatura de la carrera en la que los alumnos toman

contacto con el área de conocimiento propio del Departamento de Ingeniería de Computadoras. La consecución de los objetivos de la asignatura, que se detallaron en la sección 1 (Introducción), son fundamentales para lograr la construcción de conocimiento básico en el que se apoyan las restantes asignaturas que imparte este departamento (Arquitectura de computadoras, Sistemas operativos, Redes y teleprocesamiento, y algunas otras asignaturas de libre elección), y otras que imparten otros departamentos, como es el caso de Compiladores e Intérpretes, y Lenguajes de Programación.

3. Propuesta metodológica

3.1. Plan de la asignatura

Los **contenidos mínimos** según el plan de estudio son:

- Lenguajes, niveles y máquinas virtuales. Interconexiones. Terminología.
- Nivel de lógica digital. Sistemas combinacionales y secuenciales.
- Sistemas y aritmética de números. Sistema posicional de números. Conversión. Suma y resta de números no decimales. Representación de números negativos. Representación de números en punto flotante. Errores. Códigos de caracteres.
- Unidad Central de Proceso. Diferentes tipos de arquitecturas. Modos de direccionamiento. Memoria. Ciclo de instrucción. Tipos de instrucciones.
- Dispositivos de entrada/salida. Clasificación. Características y descripción de funcionamiento de periféricos convencionales.
- Subprogramas. Transferencia de datos. Subprogramas recursivos.
- Lenguaje ensamblador. Vinculación. Carga.
- Entrada/Salida. Drivers de E/S. Interrupciones. Prioridades, acceso directo a memoria. Procesos de interrupción.

Los **contenidos extendidos**, desarrollados por la cátedra, se presentan a continuación divididos en unidades temáticas:

UNIDAD I: Estructura de una Computadora. Introducción. Lenguajes, niveles y máquinas virtuales. Máquinas multiniveles actuales. Estructuras funcionales básicas. Evolución y generación. Unidad Central de Procesamiento. Unidad de control. Unidad aritmética-lógica. Memoria. Interconexiones. Terminología.

UNIDAD II: Sistemas Numéricos Posicionales. Eficiencia de almacenamiento. Tipos de representación de números. Bases: binaria, octal y hexadecimal. Conversión entre bases. Operaciones aritméticas de números binarios. Representación de números negativos. Signo magnitud. Complemento a la base. Complemento a la base disminuida. En exceso. Ventajas y desventajas de cada una. Rango. Redondeo. Truncar. Extensión de la precisión de los números binarios. Números fraccionarios. Aproximación de las representaciones fraccionarías. Conversión entre fracciones decimales y binarias. Punto flotante. Formato IEEE 754. Conversión. Información no numérica. Códigos ASCII y Unicode.

UNIDAD III: La Unidad Central de Proceso. Concepto de programa almacenado: instrucciones y datos. Principios de abstracción usados para construir sistemas como capas. Compilación e interpretación a través de las capas del sistema. Memoria. Estructura y Organización. Ordenamiento de los bytes. Códigos y detección de errores. Direccionamiento simbólico. Formato de las instrucciones. Registros de las computadoras, generales, de banderas. Procesadores de tres, dos, una y cero dirección. Ciclo de instrucción. Tipos de instrucciones. Movimiento de datos. Bifurcación. Procesadores con registros de propósito general. Procesadores RISC. La arquitectura MIPS. Grupo de instrucciones. Traducción de sentencias de lenguaje C en lenguaje ensamblador MIPS. Traducción de sentencias de lenguaje ensamblador MIPS en instrucciones de lenguaje máquina. Contenido de la memoria y registros durante la ejecución de un programa sencillo. Modos de direccionamiento. Objetivos. Modos de direccionamiento directos e indirectos. Inmediato, directo, registro, base, relativo. Código independiente de la posición

UNIDAD IV: Dispositivos de entrada salida. Métricas. Latencia. Productividad. Ancho de banda. Clasificación de los dispositivos de entrada-salida. Características y descripción de funcionamiento de periféricos convencionales. Terminales. Módems. Ratones. Impresoras. Unidades de disco y cintas magnéticos. Unidades ópticas. Discos magnéticos. Latencia. Tiempo de búsqueda. Tiempo de rotación. Velocidad de transferencia. Densidad de almacenamiento. Buses de Interconexión. Tipos de buses. Arbitración del bus. Métricas para evaluar un bus.

UNIDAD V: Sub-programas. Modelos de programación para implementar sub-programas. Instrucciones de invocación y retomo de sub-programas en procesador MIPS. Instrucciones de invocación y retomo de sub-programas en otros procesadores: Intel, Motorola. Comparación. Ventajas de cada uno. Mecanismos de hardware asociados. Parámetros, transferencia de datos. Pasaje de parámetros en registros y posiciones de memoria. Convención de registros. Procedimientos anidados. Área de parámetros. Asignación estática y dinámica de memoria. Pasaje de parámetros por la pila. Bloque de activación. Sub-programas recursivos.

UNIDAD VI: Lenguaje ensamblador. Vinculación. Carga. Características de un lenguaje de bajo nivel. Lenguaje y programa ensamblador. Formato de instrucciones. Instrucciones, pseudo-instrucciones, directivas. Símbolos locales y globales. Proceso de traducción a lenguaje máquina. Tabla de símbolos. Organización. Ensamblador de dos pasadas. Tiempo y operaciones en tiempo de ensamblado, carga y ejecución. Ensambladores y cargadores reubicables. Ensamblado de sub-programas independientes. Archivo objeto. Formato. Vinculación. Seudo-operaciones para símbolos externos y puntos de entrada. Técnicas de vinculación. Tabla de símbolos para vinculación de sub-programas ensamblados en forma independiente.

UNIDAD VII: Entrada-Salida. Programación directa de E/S con CPU. Direccionamiento de unidades de E/S. Programación de E/S. Puertos aislados y mapeo de memoria. Protocolos de E/S. Registros de interfases para datos y para control y estado. Simulación de MIPS. Operaciones de entrada. Ope-

raciones de salida. Drivers de E/S. Polling. Costo del Polling. Interrupciones. Estructura básica. Condiciones generales. Sistemas simples con interrupciones. Registros en MIPS para soportar interrupciones, para habilitación e inhibición de interrupciones. Prioridades. Identificación. Vectores de interrupción. Interrupciones internas. Excepciones. Llamadas al sistema. Acceso directo a memoria. Programación de un canal de DMA.

UNIDAD VIII: El nivel de lógica digital. Sistemas combinacionales y secuenciales. Sistemas combinacionales. Descripción de una función lógica. Tabla de verdad. Ecuación booleana. Forma canónica con términos mínimos y máximos. Simplificación de funciones lógicas. Diagramas de Veitch. Teoremas de De Morgan. Lógica AND, OR, NAND, NOR. Implementación de funciones lógicas. Circuitos físicos. Lógica positiva y negativa. Equivalencia de las funciones lógicas y los circuitos físicos. Circuitos AND, OR, NAND, NOR. Niveles integración. Circuitos integrados. Integración de bajo nivel SSI. Cantidad mínima de circuitos integrados para implementar una función lógica. Integración de mediano nivel MSI. Ecuaciones lógicas de multiplexores y decodificadores. Multiplexores y decodificadores para la implementación de funciones lógicas. Integración de alto nivel LSI. Circuitos combinacionales completos. RAM, ROM, EPROM. Arreglos lógicos programables PLA. Implementación de funciones lógicas con circuitos combinacionales LSI. Sistemas secuenciales. Autómata de Mealy. Autómata de Moore. Sistemas secuenciales sincrónicos. Integración de bajo nivel SSI. Flip-flops. Tipos de diagramas de estado y tablas de verdad de los flip- flops D, R-S, T, J-K. Implementación de contadores. Contador de Moebius. Registros. Buses. Sistemas secuenciales sincrónicos de control.

Los **recursos físicos** utilizados para impartir las clases son: un aula con proyector y pizarra, un laboratorio de computadoras, y una plataforma de educación a distancia. El laboratorio está conformado por computadoras personales, con sistema operativo GNU/Linux, el compilador de lenguaje C denominado GCC[2], depurador GDB[3], y simuladores de la arquitectura MIPS[4]: SPIM[5] y MARS[6]. La plataforma de educación a distancia utilizada se denomina PEDCO (Plataforma de Educación a Distancia del COmahue)[7], basada en moodle[8], que los alumnos pueden acceder para descargar el material de la asignatura (apuntes, diapositivas de clases, trabajos prácticos, etc.), utilizar el foro, e informarse de comunicaciones de la cátedra.

Con respecto a los **recursos humanos**, la cátedra está conformada por un profesor responsable de la asignatura, un asistente encargado de la realización de trabajos prácticos, y un ayudante de clases prácticas.

El alumno logra la **acreditación de la asignatura** si realiza bien por lo menos el 70 % de de cada uno de los temas evaluados en los dos exámenes prácticos (o instancias de recuperación), un examen teórico, y un informe de un trabajo realizado (en grupo de dos personas) por el alumno. Si el alumno no alcanza ese 70 %, pero supera el 50 %, obtiene la acreditación parcial de la asignatura.

La **bibliografía** básica recomendada es la siguiente:

- David A. Patterson, John L. Hennessy. Estructura y diseño de computadores: La interfaz hardware/software. 2da edición. 2011. Editorial Reverté S.A. ISBN-10: 8429126201. ISBN-13: 9788429126204.
- Andrew S. Tanenbaum, Todd Austin. Structured Computer Organization. 6th edition. 2012. Prentice Hall. ISBN-10: 0132916525. ISBN-13: 9780132916-523.
- Brian W. Kerninghan, Dennis M. Ritchie. El Lenguaje de Programación C. 2da edición. 1991. Pearson Educación. ISBN-10: 9688802050, ISBN-13: 9789688802052.

La bibliografía complementaria es:

- John F. Wakerly. Microcomputer Architecture and Programming. 1st edition. 1981. John Wiley & Sons. ISBN-10: 0471052329. ISBN-13: 978047105-2326.
- William Stallings. Computer Organization and Architecture: Designing for Performance. 9th revised edition. 2012. Pearson Education. ISBN-10: 027376-9197. ISBN-13: 9780273769194.

La **carga horaria** del cursado de la asignatura es de 160 horas en total, distribuidas en 16 semanas. Cada semana involucra 10 horas de dedicación, repartidas en 4 horas de clases teóricas, 4 horas de clases prácticas (en laboratorio), y 2 horas de consultas fuera de clase.

3.2. Metodología de clases teóricas y prácticas

Desde esta asignatura se espera mejorar ciertas competencias y habilidades de los alumnos. En principio, se pretende que el estudiante adquiera los *conocimientos específicos* de la asignatura, es decir, los conceptos teóricos y su aplicación para resolver problemas que se le presenten.

Como complemento a estos conocimientos específicos, el egresado de la carrera deberá contar con diversos recursos metodológicos que le permitan desempeñarse exitosamente en su profesión (según incumbencias profesionales descriptas en la sección 2), y por lo tanto se planea que en esta asignatura el estudiante mejore las *capacidades metodológicas* de:

1. Analizar y sintetizar la información.
2. Organizar y planificar sus actividades de aprendizaje.
3. Resolver problemas y tomar decisiones.
4. Trabajar en equipo, compartiendo los conocimientos y sabiendo comunicarlos.
5. Aprendizaje autónomo.
6. Búsqueda y valoración de la información.
7. Comprensión de textos técnicos escritos en idioma inglés.

Para adquirir los conocimientos específicos y las capacidades metodológicas detalladas, se desarrollan diferentes actividades en dos tipos de clases: teóricas y

prácticas. Las clases teóricas intentan mejorar las *capacidades metodológicas*: 1, 2, 6 y 7. Las clases prácticas, se enfocan en las *capacidades metodológicas* 2, 3, 4, 5 y 7.

Las clases se imparten dos días a la semana, y la planificación de cada día es la siguiente: se dedican 2 horas de clases teóricas seguidas de 2 horas de clases prácticas, con un intervalo de descanso de 15 minutos entre la clase teórica y práctica. En contraposición a una planificación que dedique un día para clases teóricas y un día para clases prácticas, este esquema permite al alumno: un mayor dinamismo en sus tareas diarias, avanzar en sus conocimientos teóricos y prácticos de manera progresiva para aumentar la asimilación de los mismos, leer bibliografía recomendada por el profesor, y consultar dudas en una clase próxima en el tiempo (evitando una diferencia de 7 días).

El primer día de clases del curso se le hace entrega al estudiante de un cronograma que indica las actividades (clases o evaluación) a desarrollar durante cada día del curso. Asimismo, se entrega el "Plan de la asignatura" descripto en el apartado anterior. La cantidad media de alumnos matriculados por curso es de aproximadamente 35, aunque la asistencia a clases es un poco menor, alrededor de 22 alumnos. Esto se debe a que la asistencia a clases no es obligatoria, para permitir el estudio a alumnos que trabajan (en muchos casos, la baja condición económica impide que puedan dedicarse de manera completa al estudio y tengan que compatibilizarlo con tareas laborales; cabe mencionar que la educación en universidades públicas, como ésta, es libre y gratuita).

Las clases teóricas son del tipo expositivas, se utiliza un proyector para mostrar diapositivas y la pizarra para resolver algunos detalles y preguntas de los alumnos. Se abordan los temas desde un punto de vista general planteando los problemas propios de la organización y funcionamiento de las computadoras, y se complementa explicando diferentes soluciones aportadas por distintas arquitecturas, puntualizando ventajas y desventajas de cada una.

Las clases prácticas tienen el objetivo de intentar aplicar la mayoría de los conocimientos teóricos a la resolución de casos y problemas concretos, permitiendo al alumno alcanzar una comprensión más acabada de los temas de la asignatura, así como la adquisición de las competencias.

Las múltiples actividades de las clases prácticas están enmarcadas dentro de lo que denominamos "trabajos prácticos". Cada trabajo práctico permite al alumno construir conocimiento indispensable para la resolución del siguiente trabajo práctico. Las distintas actividades de un trabajo práctico construyen conocimiento base para la siguiente actividad del mismo trabajo práctico. A su vez, los ejercicios que plantea cada actividad son graduales y construyen conocimiento base para los siguientes ejercicios de la actividad. De esta manera, se va construyendo a partir de problemas y conceptos simples, otros con mayor grado de complejidad.

Al comenzar cada práctico, se realiza una exposición explicando el objetivo del práctico, los temas que se van a estudiar, se explica el uso de las herramientas a utilizar, y se mencionan posibles dificultades con que los alumnos pueden encontrarse. Durante las clases prácticas, los alumnos son guiados por los docen-

tes para alcanzar sus objetivos. Al concluirse el tiempo estipulado de resolución de cada trabajo práctico, se resuelven ciertos ejercicios fundamentales y abarcadores que están incluidos en el trabajo práctico, y aquellos que los alumnos soliciten porque han encontrado dificultades. La resolución de los ejercicios se realiza con la atención de la totalidad de los alumnos presentes, utilizando la pizarra, la computadora y el proyector, y promoviendo la participación de los alumnos para llegar a la solución.

3.3. Actividades prácticas y herramientas de software utilizadas

La elección de la arquitectura MIPS como base para la enseñanza del funcionamiento de una computadora y la programación en lenguaje de bajo nivel, o ensamblador, se debe a la simpleza de la arquitectura, la buena bibliografía que existe y la posibilidad de utilizar el simulador en diferentes arquitecturas y sistemas operativos (permitiendo que los alumnos continúen las prácticas fuera de clases). Como contrapartida, los alumnos no trabajan sobre una arquitectura física real, aunque esto se complementa con los ejercicios en el lenguaje C de alto nivel, que si se realizan sobre la máquina real. Las actividades prácticas buscan, siempre que sea posible, promover al alumno a relacionar la programación de alto y bajo nivel. Esta relación intenta evitar principalmente que los alumnos adquieran un conocimiento abstracto de la programación de alto nivel, que podría dificultar la comprensión de problemas de rendimiento de las aplicaciones (que se estudia en asignaturas de años más avanzados).

Las actividades prácticas se realizan en grupos de dos personas, y cada grupo utiliza una computadora del laboratorio. Las 64 horas dedicadas a actividades prácticas se distribuyen en un total de 7 trabajos prácticos. En cada trabajo práctico se indica la duración $a+e$, donde a representa la cantidad de clases que el alumno debe dedicar para su resolución, y e representa la cantidad de clases utilizadas por el profesor para resolver ejercicios fundamentales y abarcadores del mismo (metodología descripta en el apartado anterior); en el último trabajo práctico las horas son solo de dedicación del alumno (no existe e). Los trabajos prácticos se describen a continuación:

1. Componentes de un sistema de computación. El objetivo es comprender la relación entre los componentes de un sistema de computación: programas de aplicación, sistema operativo y hardware. Duración: 3+1 clases. Las actividades abarcan los siguientes temas:
   a) Introducción al entorno GNU/Linux y el intérprete de comandos.
   b) Introducción al lenguaje de programación C: ciclo de compilación, uso de GCC, estructura de un programa, depuración utilizando GDB.
   c) Introducción a los simuladores de MIPS.
   d) Código en ejecución: concepto de proceso, comandos para administrar procesos en GNU/ Linux.
2. Sistemas de numeración y representación digital de datos. Su objetivo es comprender el sistema de numeración posicional, la conversión entre sistemas de distintas bases, la representación binaria de números enteros y de coma

flotante, la suma y resta de números enteros en binario, y la representación binaria de texto. Duración: 3+1 clases. Las actividades abarcan los siguientes temas:

a) Sistemas de numeración.
b) Representación y aritmética de números enteros.
c) Representación de números de coma flotante estándar IEEE-754.
d) Representación de caracteres UTF-8.
e) Tipos de datos en el lenguaje C, arreglos, estructuras y uniones, conversión de tipos.
f) Tipos de datos en MIPS.

3. Instrucciones, registros y memoria. El objetivo es estudiar la codificación de instrucciones, la utilidad de los registros, los modos de direccionamiento y la organización y gestión (básica) de memoria de los programas en ejecución. Duración: 4+1 clases. Las actividades abarcan los temas:

   a) Modos de direccionamiento.
   b) Organización de la memoria: gestión de memoria en C y MIPS, orden de bytes, alineación de datos en memoria.
   c) Registros.
   d) Representación de instrucciones: código de instrucción, direcciones de memoria, formato de instrucción.

4. Conjunto de instrucciones. Su objetivo es conocer los diferentes tipos de instrucciones disponibles en las arquitecturas, y familiarizarse con el uso de las mismas en el desarrollo de programas. Duración: 4+1 clases. Las actividades involucran programación en MIPS y lenguaje C.
5. Subrutinas. El objetivo es comprender cómo funcionan las subrutinas y las distintas maneras de pasar argumentos, relacionando el uso de subrutinas en lenguajes de alto nivel con implementaciones en lenguaje ensamblador. Duración: 4+1 clases. Las actividades involucran programación de subrutinas en MIPS y lenguaje C.
6. Dispositivos de entrada y salida, interrupciones, excepciones y llamadas al sistema. El objetivo es comprender, mediante la programación en lenguaje ensamblador, las técnicas de comunicación con los dispositivos de entrada y salida, y el tratamiento de las interrupciones, excepciones y llamadas al sistema. Duración: 4+1 clases. Las actividades abarcan los siguientes temas:

   a) Dispositivos de entrada y salida en MIPS.
   b) Llamadas al sistema en MIPS.
   c) Interrupciones y excepciones en MIPS.

7. Circuitos combinacionales y secuenciales. El objetivo es comprender el funcionamiento de los circuitos digitales combinacionales y secuenciales. Duración: 4 clases. Al finalizar la cuarta clase, cada grupo de dos alumnos deberá entregar un informe con la resolución de todos los ejercicios. Las actividades incluyen fundamentalmente temas de compuertas lógicas, multiplexores, y *flip-flops*.

3.4. Evaluación

En la plataforma PEDCO, se habilita un test para cada práctico, que cada alumno puede opcionalmente realizar y le permite auto-evaluarse antes de realizar el examen correspondiente. Esta evaluación no se califica, y el alumno puede realizar múltiples veces el mismo test.

Cada uno de los dos exámenes de práctica se realizan en el laboratorio de computadoras, son exámenes individuales, destinando una computadora por alumno. Los exámenes consisten en la resolución de una serie de ejercicios de programación, en un tiempo determinado, y como recursos pueden tener cualquier tipo de información en papel o digital (apuntes, resolución de trabajos prácticos, diapositivas de teoría, libros), y acceso total a la red Internet mediante la computadora. Este sistema lo hemos denominado "evaluación a mundo abierto", debido a que los estudiantes disponen de acceso a Internet durante el examen, siendo la intención recrear posibles situaciones de su futura vida profesional, a excepción de que no se permite ningún tipo de interacción entre los estudiantes. Este sistema de evaluación, en donde el alumno puede utilizar todas las herramientas de programación, depuración, y manuales disponibles, hace que el alumno que tiene los conceptos claros pueda encontrar la solución de cada ejercicio de los exámenes; y así los errores se pueden relacionar directamente con fallos conceptuales.

En el examen teórico se pide desarrollar ciertos temas estudiados en las clases teóricas, que son seleccionados al azar. El examen se escribe en papel (sin computadora), y el alumno no tiene acceso a ningún material de lectura.

Los exámenes de práctica tienen sus correspondientes instancias de recuperación, no así el examen teórico y la evaluación del informe entregado (resolución del último trabajo práctico).

4. Resultados

En este apartado se indican los resultados cuantitativos y cualitativos relacionados a los alumnos matriculados que logran o no acreditar la asignatura. En lo siguiente, consideramos que un alumno "aprueba" un examen o la asignatura cuando realiza correctamente el 50 % de cada tema evaluado, es decir, el porcentaje exigido para la acreditación "parcial" de la asignatura.

La metodología de evaluación utilizada para la parte práctica de la asignatura podría hacer pensar que se ofrecen demasiadas facilidades a los alumnos para aprobar los exámenes, a causa del acceso a una enorme cantidad de información (desde apuntes, libros, programas anteriormente realizados, hasta acceso total a Internet), y la dificultad para evitar interacciones entre los alumnos mediante comunicaciones por Internet. Sin embargo, un porcentaje de aprobación de exámenes que se encuentra en alrededor del 60 % de los alumnos matriculados, nos permite confirmar que los exámenes no resultan simples para todos.

Respecto a la precisión del sistema de evaluación, y tomando como muestra estadística a los alumnos que asisten a clases, existe una muy fuerte correlación entre el resultado de los exámenes y el concepto que la cátedra construyó de cada

alumno en base al rendimiento observado durante las clases (entre otros: participación y trabajo activo, avance en la resolución de los trabajos prácticos de acuerdo al cronograma). Más del 95 % de los alumnos que asisten regularmente a las clases y realizan las actividades propuestas (resuelven todos los trabajos prácticos, realizan los test de autoevaluación y consultan dudas), aprueban los exámenes de práctica realizando correctamente más del 80 % de cada tema evaluado.

Casi la totalidad de los alumnos que aprueban los exámenes de práctica son aquellos que asisten regularmente a clases y cumplen con las actividades propuestas. De los alumnos que no asisten regularmente a clases, solo unos pocos logran aprobar estos exámenes. Los motivos de que un 40 % de los alumnos matriculados no logren aprobar la asignatura, en general, son prácticamente los opuestos a los del porcentaje de aprobados. No asisten regularmente a las clases tanto teóricas como prácticas, no resuelven los trabajos prácticos y, normalmente, solo asisten a clases de consultas. Observada esta situación, se les aconseja a todos los alumnos, seguir en lo posible la metodología propuesta. Cabe destacar que la cátedra solo se ocupa de analizar posibles causas académicas sobre el rendimientos de los alumnos en la asignatura, mientras que otros tipos de causas son analizadas por diferentes áreas de la institución (principalmente el área de Tutorías y Becas)[1].

Normalmente, un 90 % de los alumnos que aprueban los exámenes prácticos también aprueban el examen teórico. Además, casi la totalidad de ellos alcanzan el nivel de acreditación total de la asignatura.

5. Conclusiones

En este trabajo se ha presentado la metodología de enseñanza y aprendizaje utilizada por el equipo de cátedra de la asignatura Organización de Computadoras, correspondiente al grado de Licenciatura en Ciencias de la Computación de la Universidad Nacional del Comahue, Argentina. La metodología soluciona los inconvenientes presentados por la metodología utilizada anteriormente en la asignatura: ausencia de relación entre la programación de alto y bajo nivel, y dificultad para detectar las causas de los errores que los alumnos cometían en los exámenes. Por un lado, se han adaptado las clases teóricas y prácticas, y se han desarrollado nuevas guías de trabajos prácticos y actividades que fomentan al alumno a establecer permanentes comparaciones de la programación de alto y bajo nivel. Por otro lado, un sistema de evaluación en computadora y "a mundo abierto" (incluso con acceso libre a Internet), permite al alumno disponer de toda información y herramientas necesarias para encontrar las soluciones a los ejercicios de los exámenes, y comprobar si funcionan adecuadamente; situación que

[1] Las causas más frecuentes que afectan el rendimiento de los alumnos son: incompatibilidad con los trabajos (muchos alumnos, en esta etapa de sus vidas, tienen actividades laborales parciales o completas), problemas de formación en etapas educativas anteriores, y desmotivación por el estudio.

intenta aproximarse al ambiente de trabajo real de un profesional. Esta metodología de evaluación permite, a los profesores, relacionar los errores cometidos por los alumnos, en los exámenes, con fallos conceptuales y no simples distracciones. Así, el proceso de evaluación logra un resultado más justo, y permite determinar con mayor precisión posibles falencias en la enseñanza.

Los resultados de aplicar la metodología propuesta indican una mejoría en la calidad del aprendizaje alcanzado por los alumnos. Esta realidad no solo se fundamenta en los índices obtenidos para esta asignatura, sino también en el seguimiento del desempeño de los alumnos en las materias siguientes. Para la cátedra, los resultados significan un acierto en la metodología, tanto en la forma de impartir las clases como en la forma de evaluación. Sin embargo, es razón actual de estudio, de la cátedra y de la institución, lograr mejorías en la cantidad de alumnos aprobados cuando los estudiantes no pueden asistir (por distintas circunstancias) a clases.

Referencias

1. Plan de la carrera Lic. en Cs. de la Computación: (Accedido en Abril de 2014) http://faiweb.uncoma.edu.ar/images/ordenanzas/Carreras/ord_0075_2010_23.pdf.
2. GCC, the GNU Compiler Collection: (Accedido en Abril de 2014) http://gcc.gnu.org/.
3. GDB: The GNU Project Debugger: (Accedido en Abril de 2014) http://www.sourceware.org/gdb/.
4. David A. Patterson, J.L.H.: Estructura y diseño de computadores: La interfaz hardware/software. 2da edn. Editorial Reverté, S.A. (2011) ISBN-10: 8429126201, ISBN-13: 9788429126204.
5. SPIM: A MIPS32 Simulator: (Accedido en Abril de 2014) http://spimsimulator.sourceforge.net/.
6. MARS (MIPS Assembler and Runtime Simulator): (Accedido en Abril de 2014) http://courses.missouristate.edu/kenvollmar/mars/.
7. Plataforma de Educacion a Distancia, Universidad Nacional del Comahue: (Accedido en Abril de 2014) http://pedco.uncoma.edu.ar/.
8. Moodle - Open-source learning platform: (Accedido en Abril de 2014) https://moodle.org/.