Actas de las Jenui, vol. 4. 2019.
Páginas: 319-326

Tecnología low-cost para motivar al alumno

Pablo Fuentes, Cristóbal Camarero, Carmen Martínez, Fernando Vallejo
Departamento de Ingeniería Informática y Electrónica
Universidad de Cantabria
{fuentesp,camareroc,martinezc,vallejof}@unican.es

Resumen

La docencia en Estructura y Organización de Computadores se imparte habitualmente utilizando una arquitectura como referencia, siendo MIPS una de las más usadas. Sin embargo, su baja penetración en el mercado actual tiene un efecto desmotivador en los alumnos. Esto, junto con la mayor relevancia de la arquitectura ARM, nos ha hecho plantearnos considerarla como referencia en nuestros planes de estudios. Con este objetivo, se ha desarrollado un proyecto de innovación docente para rediseñar las materias implicadas. Esta modificación ha implicado cambios en el laboratorio en el que se desarrollan las prácticas, para el que se ha seleccionado Raspberry Pi, una plataforma de bajo coste basada en ARM. Como principal diferencia respecto a otras soluciones, se ha elegido el sistema operativo RISC OS, que permite realizas prácticas de programación en ensamblador y Entrada/Salida en el mismo entorno. Además, se ha desarrollado un depurador en ARM llamado !UCDebug.

En este artículo se describen las diferentes etapas del proyecto y los resultados obtenidos. Cabe destacar que, en los dos cursos que lleva implantado, la motivación y asistencia de los alumnos ha aumentado considerablemente y la tasa de éxito ha subido hasta un $13\,\%$.

Abstract

Computer Organization and Design is usually instructed using a computer architecture as reference. The MIPS architecture has been traditionally widespread in that regard. However, its current low market share discourages the students. This, coupled with the greater relevance of the ARM architecture, has motivated our switch to the latter as the reference in our degree. With this aim, a redesign of the involved courses has been carried out through an educational innovation project. One of the effects has been a transition to the low-cost ARM Raspberry Pi platform for the computer lab where students perform their practical sessions. As a major difference to the approach followed by other universities, the RISC Operating System has been chosen, allowing to perform both assembly and Input/Output practices within the same environment. Moreover, an ARM debugger called !UCDebug has been developed to alleviate the lack of user-friendly debugging tools.

This article describes the stages of the project and the results achieved. Notably, student motivation and attendance has remarkably risen, and the success rate has grown up to 13% since the implementation.

Palabras clave

Estructura y organización de computadores, Prácticas de laboratorio, ARM, Raspberry Pi, RISC OS

1 Introducción

Motivar a los alumnos de las asignaturas de Estructura y Organización de Computadores es una tarea muy complicada. Se observa en el aula que el alumno no encuentra utilidad práctica al estudio de un lenguaje ensamblador, código máquina y, en general, todos los aspectos relacionados con los niveles bajos de la arquitectura. Esta falta de motivación se hace especialmente patente en las sesiones de laboratorio, realizadas mediante simulación o con prototipos de desarrollo, ya que tienen poca relación con lo que el alumno conoce.

Para el estudio de los conceptos básicos de estas materias se emplea tradicionalmente una determinada arquitectura, procurando que sea comercial, básica y pedagógica. En los últimos años, la mayoría de las universidades han optado por la arquitectura MIPS. Esta arquitectura se diseñó en la Universidad de Stanford en los años 80 y dio lugar a un conjunto de instrucciones muy simple. En las décadas de los 90 y 2000 tuvo un gran impacto en el mercado de servidores, sobre todo de Silicon Graphics, y en el desarrollo de procesadores empotrados, donde se concentra su uso actual. Debido a su auge en docencia, existen múltiples simuladores de MIPS que permiten realizar las prácticas básicas sin necesidad de contar con costosos equipos.

En el año 2003 la Universidad de Cantabria (UC)

implanta la arquitectura MIPS como hilo conductor en todas las asignaturas básicas del área de Arquitectura y Tecnología de Computadores. Inicialmente se plantean todas las sesiones prácticas sobre un módulo comercial de la empresa MyCable concebido para el prototipado de agendas electrónicas, al que se adaptó el software y dotó de un pequeño depurador de código. Con la llegada de los estudios de Ingeniería Informática a la UC creció el número de alumnos que realizaban prácticas sobre estos equipos, y se optó por emplear simuladores en las prácticas de lenguaje ensamblador y continuar con el hardware real en las prácticas de Entrada/Salida (E/S). Este cambio facilitó el desarrollo de las primeras prácticas de lenguaje máquina ya que el alumno, al no depender del equipo del laboratorio, podía completar la práctica en cualquier equipo. Sin embargo, para realizar las prácticas de E/S en el siguiente curso el alumno debía adaptarse al hardware real, muy diferente al simulador empleado previamente. El aprendizaje de un nuevo entorno, junto con la necesidad de acudir al laboratorio para realizar la práctica, provocaba un sentimiento generalizado de frustración. Esta frustración quedaba patente en las encuestas de calidad así como en conversaciones con los alumnos, que estaban desmotivados y no apreciaban la utilidad de las sesiones prácticas, realizándolas de forma rutinaria y apática. Como consecuencia, estas asignaturas presentaban una alta tasa de abandono, elevado número de suspensos y alta tendencia a la copia de las prácticas. Sin embargo, la experiencia de un curso cero realizado en la Universidad de Alicante [6] demuestra que un enfoque distinto en la impartición puede aumentar la implicación de los alumnos.

Hace tres años, en el grupo de ATC iniciamos el desarrollo de una serie de proyectos de innovación docente orientados a mejorar la motivación de los alumnos en estas materias. El cambio tenía como objetivo superar todos los inconvenientes que se habían observado en los laboratorios. El primer paso era que el alumno tuviese la visión de que la arquitectura empleada es realmente útil y actual. Tras elegir una arquitectura más idónea, había que dotar a las clases teóricas de mayor contenido práctico que ayudase en el desarrollo de las sesiones prácticas. Seguidamente, había que remodelar el laboratorio para impartir todas las prácticas sobre una única plataforma y que el alumno no dependiese en exclusiva del hardware del laboratorio, pero sin olvidar que el coste de la remodelación debía ser asumible por el Departamento. Por último, había que diseñar un conjunto de prácticas que resultase atractivo para el alumno y que cumpliese con los objetivos académicos de cada asignatura.

En este artículo se detalla el proceso de desarrollo e implantación del proyecto, así como los resultados obtenidos. La organización del trabajo es la siguiente. En el Apartado 2 se hace un resumen de la situación de la docencia ARM en la Universidad española, haciendo hincapié en las diferencias respecto a nuestro proyecto. A continuación, en el Apartado 3 se describe la solución propuesta en este proyecto para la docencia basada en ARM. El Apartado 4 contiene el detalle de la implantación del proyecto en los dos cursos académicos que lleva vigente. Finalmente, el Apartado 5 presenta los resultados obtenidos hasta la fecha, así como una discusión crítica de las aportaciones del proyecto y posibles mejoras y trabajo futuro.

2 Alternativas docentes

En 2016 se hizo un análisis de la docencia de Estructura y Organización de Computadores impartida en otras universidades del ámbito nacional. Como resultado se observó que el número de universidades que empleaban la arquitectura ARM como referencia era bastante reducido, contándose entre ellas la Universidad Complutense de Madrid, la Universidad de Cádiz, la Universidad Rovira I Virgili, la Universidad de Málaga, la Universidad del País Vasco y la Universitat Jaume I. De éstas, se hizo un análisis más a fondo del entorno y tipo de laboratorio empleado en las tres últimas.

En la Universidad del País Vasco [9] se empleaba la videoconsola Nintendo DS como hardware de laboratorio, empleando programación en ensamblador ARM y en código C. Aunque cumplía con el propósito de hacer atractiva la enseñanza, no se ajustaba a los objetivos de bajo coste y alta disponibilidad deseados para fomentar el aprendizaje autónomo. En la Universitat Jaume I se combinaba el uso de hardware real, concretamente la plataforma Arduino [3], con un simulador del repertorio *Thumb* (Qt ARMSim) [2]. Esto incumplía el objetivo marcado de manejar exclusivamente hardware real para la docencia.

Por último, en la Universidad de Málaga, cuyo material docente ha constitutido un gran referente para nuestro proyecto [13], se empleaba Raspberry Pi. Sin embargo, usaban dos entornos de laboratorio diferenciados: dado que el sistema operativo (SO) elegido para la programación en lenguaje ensamblador y en C (Raspbian) no permite desarrollar *drivers* de E/S, para dichas prácticas utilizaban programación en *Bare Metal*, trabajando directamente sobre el hardware sin ningún tipo de SO ni depurador. Esta dicotomía choca con el objetivo de emplear un único entorno de laboratorio para reducir la curva de aprendizaje del alumno.

3 Solución propuesta

En este apartado se detalla la solución elegida para abordar los problemas y limitaciones descritos en el

Apartado 2. En primer lugar, se ha escogido como equipo la Raspberry Pi[1], un dispositivo de bajo coste y amplia disponibilidad en el mercado, destinado inicialmente al segmento educativo. Este dispositivo presenta dos ventajas fundamentales:

- Emplea un procesador basado en la arquitectura ARM, que es una arquitectura RISC apta para el aprendizaje del funcionamiento de un procesador (los aspectos más fundamentales tienen un nivel de complejidad asequible) y de amplia presencia en el mercado (más de 11 000 millones de chips distribuidos en la primera mitad de 2018 [1]).
- Presenta un bajo coste y está disponible en canales de venta minorista, lo que permite reducir los costes de renovación y mantenimiento del laboratorio, pero también facilita que los alumnos puedan adquirir su propio equipo.

Concretamente, se ha elegido la Raspberry Pi 1B+ porque es la última versión que empleó un procesador monocore, lo que simplifica el aprendizaje de la arquitectura y evita la confusión asociada a la ejecución simultánea de código en varios cores. Además, hay disponible suficiente documentación del procesador, permitiendo concebir prácticas que hagan uso de los terminales de E/S del equipo. Subsiguientes modelos están peor documentados y son más complejos, lo cual choca con la motivación del proyecto.

Como decisión práctica, se ha optado por combinar el puesto basado en Raspberry Pi con un puesto preexistente que empleaba PCs. Para evitar redundancia de periféricos, se ha empleado un conmutador de vídeo, teclado y ratón (KVM) que permite conmutar entre el uso del PC y el de la Raspberry Pi. No obstante, ambos puestos funcionan de forma completamente independiente, y es posible trabajar con cualquiera de ellos sin que el otro esté encendido.

Tras elegir la Raspberry Pi como equipo a usar en el laboratorio, se ha escogido un SO. La decisión de emplear un SO viene dada por la intención de emplear un puesto independiente, que no necesite un PC para realizar las prácticas, y que permita al alumno disponer de herramientas básicas como un editor de textos, preferiblemente con una interfaz visual que minimice el período de familiarización con el puesto. De los diversos SOs disponibles para Raspberry Pi, se ha elegido RISC OS. Se trata de un SO básico, que permite al alumno manejar el hardware directamente, evitando capas de abstracción software impuestas por el SO. Resulta, por tanto, más instructivo para el aprendizaje del manejo de la E/S, permitiendo al alumno desarrollar las funcionalidades de un driver real con el mismo entorno empleado en el resto de prácticas de programación en ensamblador.

Como desventaja, la interfaz del SO presenta ligeras diferencias respecto a otros más populares (Windows, distribuciones Linux, MAC OS), particularmente en la interacción a través del ratón. Sin embargo, este sistema cuenta con múltiples aplicaciones software desarrolladas para él, como editores de texto robustos y con resaltado de sintaxis (!StrongED [12]) o el software de compilación/ensamblado/linkado de GNU, GCC[2].

Tal y como se analizó en [4], una herramienta fundamental para el desarrollo de las prácticas del alumno es un depurador que permita a los alumnos detectar fallos en su código, para su posterior corrección. Una de las limitaciones de RISC OS es que sólo está disponible un depurador nativo basado en el uso de comandos en una ventana de tareas. Lamentablemente, su funcionamiento no es intuitivo ni cómodo, e introduce una complejidad adicional al aprendizaje de la arquitectura y de su programación. Una de las soluciones barajadas inicialmente era el empleo del depurador de GNU, GDB, una herramienta muy potente que no se restringe únicamente a código en ensamblador, y que los alumnos pueden haber manejado anteriormente para otros lenguajes de programación. Sin embargo, el depurador GDB no está disponible en RISC OS. Para suplir esta carencia, una de las líneas de actuación del proyecto ha sido el desarrollo de la herramienta !UCDebug, un depurador de código ensamblador ARM para RISC OS basado en una interfaz de ventanas.

El depurador !UCDebug es un software de código libre, disponible públicamente en un repositorio de GitHub [5]. Está escrito en lenguaje ensamblador ARM y en C, lo que permite un alto grado de control sobre la ejecución del código del alumno a la par que facilita el desarrollo de la interfaz gráfica. La herramienta permite ejecutar de principio a fin el código desarrollado por el alumno, así como introducir puntos de ruptura y ejecutar instrucción a instrucción. También permite interrumpir una ejecución en curso en el caso de que haya errores en el código que no lo detengan (por ejemplo, un bucle infinito), y depurar la ejecución de rutinas de atención a la interrupción (RTI). Hay que recalcar que no se trata de un simulador sino de un depurador, por lo que el resultado mostrado al alumno corresponde con la ejecución real en el procesador del código desarrollado.

Para su desarrollo se ha seguido una metodología iterativa-incremental, con un desarrollo en paralelo al avance del curso académico, corrigiendo fallos a medida que eran detectados por los desarrolladores o por los alumnos, e introduciendo nuevas características para responder a las necesidades de las siguientes sesiones prácticas. Esto ha generado una visión ligeramente negativa por parte de los alumnos, que no dispusieron de un software robusto hasta las últimas sesiones.

[1] `https://www.raspberrypi.org/`

[2] `http://gcc.gnu.org/`

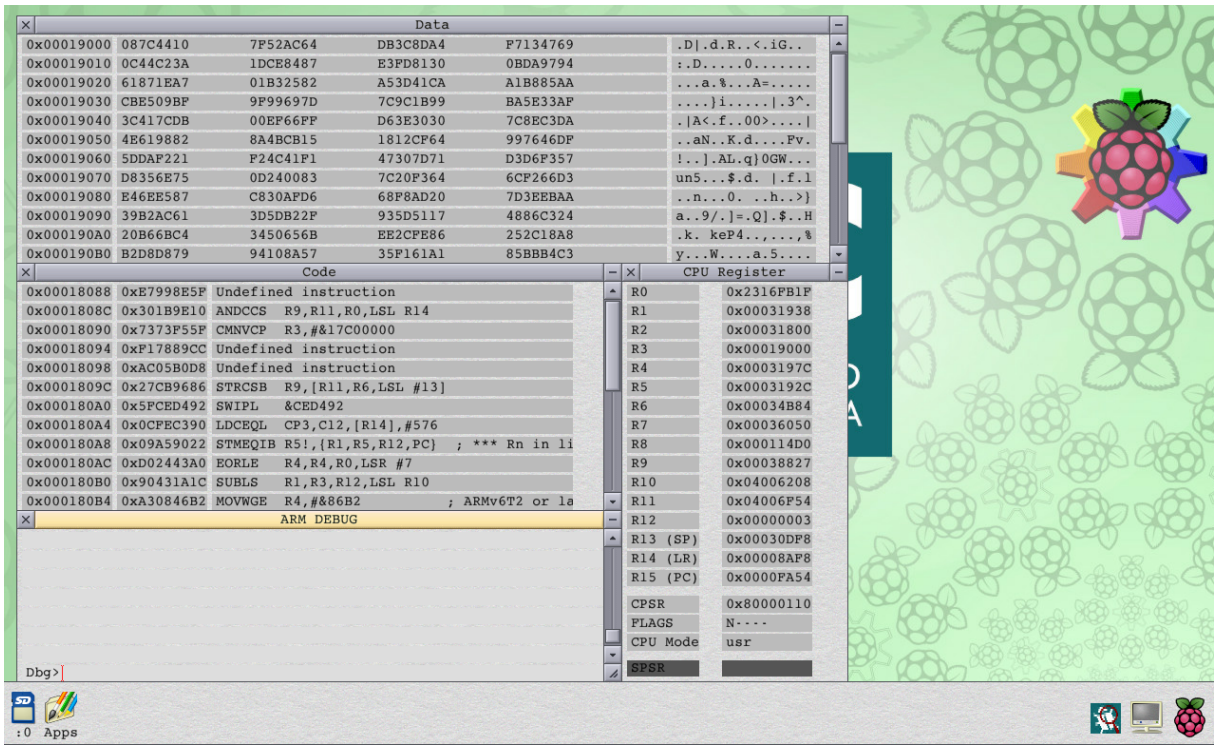

Figura 1: Captura de pantalla de la Raspberry Pi con el SO RISC OS y el depurador !UCDebug.

Como se observa en la Figura 1, el depurador consta de 4 ventanas por defecto. La ventana de código (Code) muestra una vista de memoria a partir del punto de entrada del código del alumno, con las direcciones de memoria en las que está alojado, y con las instrucciones en código máquina y en lenguaje ARM, lo que permite observar la traducción de pseudo-instrucciones. La ventana de datos (Data) presenta una vista de la memoria a partir del comienzo del segmento de datos del código, dividida en líneas de 128 bits. El contenido se puede mostrar por palabras, medias palabras (halfword) y bytes, y también se muestra su correspondencia con caracteres ASCII para facilitar la localización de cadenas de texto.

La ventana del banco de registros (CPU Register) muestra todos los registros de propósito general, así como el registro CPSR y, en los modos de ejecución en los que corresponda, el SPSR. Para facilitar la depuración, también hay una vista rápida que indica los flags del CPSR que están activos, y el modo de ejecución actual. Por último, la ventana de consola (ARM DEBUG) permite al usuario emplear una serie de comandos para interactuar con el depurador, por ejemplo para comenzar y detener la ejecución del código. En esta ventana se dispone de un buffer de comandos accesible con las teclas de flecha del teclado, para agilizar su uso.

El reparto en ventanas permite reorganizar la vista por defecto y minimizarlas, para ajustarse a las necesidades del usuario. Mediante un icono presente en la barra inferior de la pantalla durante la ejecución del depurador se puede también restablecer la colocación y tamaño de las ventanas a su estado inicial.

4 Implantación del proyecto

El proyecto presentado en este artículo tiene alcance en dos titulaciones: el grado en Ingeniería Informática (GII) y el grado en Ingeniería de Tecnologías de Telecomunicación (GITT). En el Cuadro 1 se muestran las asignaturas que han participado en la implantación del proyecto y sus características.

En la titulación GII, existe una fuerte dependencia entre las asignaturas de primer y segundo curso, planteándose unas como continuación de otras. En estas asignaturas se estudia el modelo de la máquina de Von Neumann, concentrándose el primer curso en el *Instruction Set Architecture* (ISA), mientras que en el segundo curso se aprende el manejo de la E/S e interrupciones. Esto da lugar a dos tipos de prácticas muy diferenciadas: programación en ensamblador y desarrollo de drivers. En GITT, la situación es la misma, pero con mucho menor impacto, puesto que la asignatura Computadores y Comunicaciones es de carácter optativo. Como se puede observar, entre todas las asignaturas suman más de 300 alumnos.

Por la amplia variedad de asignaturas implicadas, el

Asignatura	Titulación	Curso	Tipología	Alumnos (curso 17/18)	Alumnos (curso 18/19)
Introducción a los Computadores	GII	1	Básica	92	86
Estructura de Computadores	GII	2	Obligatoria	82	83
Organización de Computadores	GII	2	Obligatoria	78	73
Microprocesadores	GITT	3	Obligatoria	74	52
Computadores y Comunicaciones	GITT	4	Optativa	8	4

Cuadro 1: Detalle de las asignaturas involucradas en el proyecto de innovación.

impacto en la dependencia entre asignaturas de la titulación y el gran número de alumnos afectados, se decidió realizar la implantación del proyecto de forma gradual. Durante el curso 2016/17 se realizó una revisión completa de las guías docentes de las asignaturas implicadas por parte de los profesores responsables, aprovechando así la modificación de la arquitectura a impartir para coordinar nuevamente los contenidos de las asignaturas implicadas y solventar carencias que se habían observado en cursos previos. Así, se decidió por ejemplo añadir un tema más detallado de la implementación del procesador en primer curso, para mejorar el acceso a esta parte en segundo curso. Se realizó un estudio de la bibliografía disponible para la docencia de Estructura y Organización de Computadores basada en ARM. Como textos de referencia se seleccionaron los manuales de Harris & Harris [8] y Pyeatt [11], además de la versión actualizada del clásico de Patterson y Henessy [10], todos ellos textos de muy reciente publicación. Además, se seleccionó la plataforma más adecuada para el laboratorio, como se ha explicado en el Apartado 3. Durante los cursos 2017/18 y 2018/19 se realizó la implantación en los planes de estudio mencionados. A continuación, pasamos a detallar esta implantación.

En el curso 2017/18 se decide realizar el cambio de arquitectura y laboratorio en la asignatura Introducción a los Computadores, de primer curso. A este respecto, se modificaron consiguientemente las diferentes actividades realizadas en la misma. Por un lado, para las clases de teoría y prácticas de aula, se realizaron nuevos materiales de apoyo y se trabajó con la nueva bibliografía. Para las clases de teoría se realizaron nuevas transparencias, como material de apoyo. Los problemas propuestos como prácticas de aula también se cambiaron en este curso. Respecto al laboratorio, dado que esta asignatura hasta ahora se había realizado con simulación, se decidió una modificación híbrida, es decir, una primera parte del laboratorio utilizando un simulador y la segunda parte usando la Raspberry Pi. En ambos casos se han diseñado prácticas completamente nuevas respecto a cursos previos. Como simulador se seleccionó ARMSim#, desarrollado en la Universidad de Victoria (Canadá). Este simulador fue seleccionado puesto que las otras alternativas, o bien contemplaban un subconjunto de instrucciones muy reducido, no llegando al nivel esperado de la asignatura, o bien eran simuladores de desarrollo de ARM, demasiado complejos para una asignatura de primer curso. En la segunda parte del curso se realizaron varias prácticas con la Raspberry Pi y el entorno de depuración proporcionado por RISC OS. Durante el desarrollo del curso se pudieron apreciar varios aspectos que han influido enormemente en la implantación del curso 2018/19, que son:

- Los alumnos se motivaron mucho viendo que estudian una arquitectura con gran presencia en el mercado como es ARM.
- Aunque los alumnos lo recibieron bien, el simulador escogido tenía ciertos problemas de implementación, y al no ser código abierto no se pudieron solventar. Al ponernos en contacto con los desarrolladores, pudimos constatar que se encuentra sin soporte.
- Las prácticas en la Raspberry Pi fueron también muy motivadoras para los alumnos y la asistencia a las mismas fue masiva. El alumno tiene la sensación de estar manejando algo real.
- El desarrollo de código con el entorno de depuración nativo de RISC OS era lento y la capacidad de depuración limitada.
- Tener que introducir dos entornos de laboratorio (el simulado y el basado en hardware real) consumió demasiado tiempo, así como afrontar el cambio y su curva de aprendizaje.

Durante el presente curso 2018/19 se ha realizado la segunda parte de implantación del proyecto. En este primer cuatrimestre, las asignaturas involucradas han sido Estructura de Computadores y Microprocesadores. Estructura de Computadores, que se plantea como continuación de Introducción a los Computadores, tiene principalmente prácticas de desarrollo de drivers de dispositivos de E/S. Por tanto, la mayoría de los alumnos de esta asignatura han realizado prácticas con la Raspberry Pi en el curso anterior, pero con el entorno de depuración de RISC OS. En esta asignatura, realizarán las prácticas con la misma plataforma, pero con el depurador !UCDebug. Sin embargo, los alumnos de Microprocesadores nunca han trabajado en lenguaje

ensamblador, y el objetivo es que en esta asignatura aprendan desde las bases, hasta llegar a programar algún driver. Dada la experiencia del curso pasado con el laboratorio mixto (simulador y hardware), se decide que en el curso 2018/19 todas las prácticas serán con Raspberry Pi. Como se describió en el Apartado 3, el desarrollo del depurador ha seguido la metodología iterativa-incremental, por lo que cada 2 semanas aproximadamente había que actualizarlo. En ambas asignaturas los drivers realizados han manejado una placa BerryClip+[3] y el *system timer*. Como primeras impresiones, que se analizarán más a fondo en el apartado de resultados, se ha podido observar un impacto enorme en la motivación del alumno por el uso del hardware real, llegando a pedir los alumnos prácticas de carácter opcional, para "aprender más".

A continuación en el próximo cuatrimestre, se implantarán los cambios en las asignaturas Organización de Computadores e Introducción a los Computadores. En cuanto a la primera, está previsto manejar la nueva bibliografía y realizar implementaciones del procesador ARM monociclo y segmentado en el software de diseño lógico TkGate [7]. En Introducción a los Computadores, a la vista de la experiencia del curso pasado y de este cuatrimentre con la programación básica en Microprocesadores, se ha decidido prescindir del entorno de simulación, y realizar todas las prácticas con Raspberry Pi y !UCDebug. Por un lado, esta decisión permitirá realizar sólo un seminario de introducción al laboratorio, lo que ahorrará tiempo. Por otro lado, el manejo de un único entorno de prácticas durante todo el cuatrimestre supondrá un menor *overhead* para el alumno. Esta decisión además beneficiará enormemente a la asignatura Estructura de Computadores, puesto que los alumnos pasarán a trabajar en un entorno de laboratorio ya conocido, permitiendo que el alumno se centre en el contenido de la materia desde el primer día. Consideramos que ésta es una muestra del éxito de nuestra propuesta, puesto que responde a nuestra principal motivación: unificar el laboratorio en el mayor número posible de asignaturas relacionadas. De hecho, para siguientes cursos se está planteando realizar prácticas usando el entorno Raspberry Pi y RISC OS en Organización de Computadores y Sistemas Operativos.

5 Resultados y discusión

A continuación se detallan los resultados de las encuestas de satisfacción realizadas y los resultados académicos, concluyendo el trabajo con una discusión de los resultados.

[3] https://bitbucket.org/MattHawkinsUK/rpispy-berryclip-plus/downloads/

Pregunta	Media	Desviación
1	7.49	1.10
2	7.34	1.48
3	6.86	1.14
4	7.49	1.82
5	**7.7**	1.27

Cuadro 2: Resultados de las encuestas

5.1 Encuestas de satisfacción

Al final del primer cuatrimestre de este curso realizamos una serie de encuestas a los alumnos de Estructura de Computadores y de Microprocesadores. Con la intención de realizar una encuesta lo bastante sencilla para que todos los alumnos participasen, pero que aportase suficiente información, se formularon las siguientes preguntas:

1. !UCDebug es fácil de utilizar.
2. !UCDebug facilita la depuración.
3. Estoy muy satisfecho con !UCDebug.
4. Las prácticas en el laboratorio sirven para asentar los conocimientos de la teoría.
5. El cambio realizado en la asignatura es positivo.

De éstas, la quinta pregunta se planteaba exclusivamente a los alumnos que habían cursado la asignatura con antelación al cambio. Seguidamente, se pedía que aportasen al menos una sugerencia de mejora para el próximo curso.

Se recogieron un total de 81 encuestas (46 en GII y 35 en GITT), siendo este un número menor del deseado (135 alumnos en total). Consideramos que se recogieron menos encuestas porque al finalizar el cuatrimestre la asistencia al laboratorio es menor, debido a la presión de la evaluación en los alumnos. La idea inicial era analizar cada titulación por separado, pero teniendo en cuenta que los resultados son muy similares, presentamos los resultados de forma conjunta en el Cuadro 2. Como se puede apreciar, todos los ítems están valorados prácticamente con la misma nota, de media notable. La pregunta con más valoración es la 5º, dirigida a alumnos que habían cursado la asignatura con MIPS, lo que nos dice que el cambio ha sido bien recibido.

En cuanto a los comentarios y/o propuestas de mejora, los más habituales son:

- Dedicar más tiempo a prácticas.
- Hacer que el depurador sea más intuitivo.
- Que el depurador concrete más los errores.
- Implantar un sistema de préstamos de Raspberry Pi en la facultad.

Dos de las propuestas están muy relacionadas con el depurador, en cuanto a su uso y capacidad de depuración. Entendemos que hacer un depurador más in-

tuitivo es algo beneficioso y estamos trabajando en ello. Sin embargo, el depurador muestra los errores con cierta ambigüedad como criterio de diseño, para hacer más pedagógica la herramienta, puesto que fuerza al alumno a analizar el contexto en el que se ha producido el error y a determinar su causa. El alumno tiene que coger agilidad en detectar, con los mensajes del entorno de depuración, el tipo de errores y cómo solucionarlos. En cuanto a dedicar más tiempo en prácticas, entendemos que el alumno percibe las prácticas como algo beneficioso para al aprendizaje de la materia. Nos parece curioso que, precisamente, una de las motivaciones para seleccionar la Raspberry Pi era su bajo coste, que podría facilitar que los alumnos se la comprasen, y así practicar en casa de forma autónoma. Sin embargo, tanto la primera como la última sugerencia, ponen de manifiesto que, al menos este primer curso, no están dispuestos a hacerlo. Quizás cuando vean este cambio como algo estable, el número de alumnos que la compre aumente, aunque se está estudiando un sistema de préstamos gestionado por la biblioteca.

5.2 Resultados académicos

En este curso se ha completado el primer ciclo de la implantación. Al haberse cursado las dos asignaturas donde se usa el nuevo laboratorio con los cambios previstos, podemos hacer una valoración preliminar y positiva de los resultados. Aunque los datos se limitan a un único curso, las sensaciones mientras se iban impartiendo las asignaturas hacen prever que esta tendencia pueda mantenerse en el futuro. Para analizar los resultados académicos nos vamos a fijar en la tasa de éxito y la tasa de abandono de las asignaturas, antes y después de la implantación. Para el cálculo de ambas tasas usamos la metodología empleada en los informes de calidad de la titulación. La tasa de éxito nos indica el número de alumnos que han aprobado la asignatura frente a los alumnos que se han presentado. La tasa de abandono nos indica el número de alumnos que no se han presentado a ninguna prueba de la asignatura frente a los alumnos matriculados.

Los datos mostrados en el Cuadro 3 presentan los resultados obtenidos en el último curso que empleaba la arquitectura MIPS frente al curso donde ya se ha implantado el nuevo programa. En el caso de la asignatura de primero son datos de las dos convocatorias del curso, mientras que en las otras dos asignaturas se limitan a la convocatoria ordinaria, al no haberse realizado aún la convocatoria extraordinaria del curso 2018/19. De todos los resultados puede apreciarse que en general se ha logrado disminuir la tasa de abandono, lo que indica una mayor motivación por parte del alumnado en estas asignaturas, y un fuerte incremento en la tasa de éxito.

Además, se ha añadido un estudio de la asignatura EC restringido a aquellos alumnos que el curso anterior habían realizado IC, es decir, aquellos alumnos que han cursado ambas asignaturas con la nueva estructura. De este estudio se observa que el incremento en la tasa de aprobados ha sido superior en este conjunto reducido respecto al total de la clase, dato que en años previos era similar. Este hecho parece señalar que el incremento de la tasa de éxito también se debe a la mayor motivación del alumno.

5.3 Discusión

A la vista de estos resultados podemos, en nuestra opinión, obtener las siguientes conclusiones.

- *El cambio ha sido positivo*: la asistencia, los resultados académicos y las encuestas a los alumnos así lo confirman.
- *Hay que continuar desarrollando !UCDebug*: se han conseguido realizar las prácticas con éxito, pero a la vista de la experiencia y los resultados obtenidos, hay que aplicar diversas mejoras sobre la herramienta para garantizar su usabilidad y capacidad pedagógica.
- *El manejo de hardware real mejora la visión del alumno en cuanto a la Estructura y Organización de Computadores*: mientras que el manejo de simuladores puede ser muy útil y cumplir de forma suficiente su función en este tipo de asignaturas, hemos podido constatar que el manejo de hardware real hace que el alumno ubique perfectamente el énfasis y objetivos y tenga una visión global de la arquitectura.
- *El aprendizaje de RISC OS es asequible*: las principales dificultades, como muestran los comentarios, vienen del manejo de RISC OS a nivel de usuario. Sin embargo, estas dificultades de manejo se minimizan con el uso. El manejo de software nuevo nos parece algo beneficioso para el perfil de alumno, lo que nos lleva a concluir que son más las ventajas que aporta la inclusión de RISC OS en el laboratorio que los inconvenientes que pueda causar su aprendizaje.

Por todo esto, nuestro balance en cuanto al cambio realizado, es positivo. Como trabajo futuro, continuaremos desarrollando !UCDebug para acercarnos más a las expectativas del alumno y estudiaremos la manera de integrar el laboratorio basado en Raspberry Pi en otras asignaturas de las titulaciones.

6 Agradecimientos

Los autores agradecen la colaboración de los alumnos de las asignaturas involucradas por su participación. Además, agradecer a David Herreros, Mario Ibáñez,

Tasa	Éxito	Abandono	Éxito	Abandono
	MIPS (curso 16/17)		**ARM (curso 17/18)**	
Introducción a los Computadores (IC)	75 %	32 %	83 %	19 %
	MIPS (curso 17/18)		**ARM (curso 18/19)**	
Microprocesadores	64 %	13 %	68 %	23 %
Estructura de Computadores (EC)	47 %	17 %	60 %	6 %
EC para alumnos que cursaron IC el año anterior	47 %	11 %	63 %	2 %

Cuadro 3: Tasa de éxito y abandono en la asignatura Estructura de Computadores.

Daniel Padilla y Daniel Torre su participación en el proyecto. Este trabajo ha sido parcialmente financiado por la III Convocatoria de Proyectos de Innovación Docente, del Vicerrectorado de Ordenación Académica y Profesorado de la Universidad de Cantabria, el Ministerio de Economía, Industria y Competitividad bajo contrato TIN2016-76635-C2-2-R (AEI/FEDER, UE) y el Ministerio de Ciencia, Innovación y Universidades bajo beca Juan de la Cierva FJCI-2017-31643.

Referencias

[1] Arm Holdings. *Arm FY2018 Q2 Key Performance Indicators report.* Publicado en Octubre 2018; Revisado Noviembre 2018. Disponible en `https://www.arm.com/company/investors/financial-results`

[2] S. Barrachina Mir, G. Fabregat Llueca, J. C. Fernández Fernández, y G. León Navarro. ARMSim y QtARMSim: simulador de ARM para docencia. En *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática*, páginas 2–9, Universitat Oberta La Salle, 2015.

[3] S. Barrachina Mir, G. Fabregat Llueca, y J. V. Martí Avilés. Utilizando Arduino DUE en la docencia de la entrada/salida. En *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática,* páginas 58–65, Universitat Oberta La Salle, 2015.

[4] C. Camarero, E. Z. Suárez, E. Stafford, F. Vallejo, y C. Martínez. Enseñanza Práctica de Estructura y Organización de Computadores con Raspberry Pi. En *Jornadas Sarteco 2017,* Septiembre 2017.

[5] C. Camarero, F. Vallejo, y P. Fuentes. *!UCDebug, an ARM Debugger for RISC OS.* Disponible en `https://fuentesp.github.io/UCDebug/`. Último acceso: Enero 2019.

[6] F. J. Gallego Durán, R. Satorre Cuerda, P. Compañ Rosique, y C. Villagrá Arnedo. El código máquina mola. En *Actas de las XXIV Jornadas de la Enseñanza Universitaria de la Informática,* páginas 149–156, Universitat Oberta de Catalunya, 2018.

[7] J. Hansen. *TkGate, A graphical editor and event-driven simulator for digital circuits.* Disponible en `https://github.com/bnoordhuis/tkgate/`. Último acceso: Enero 2019.

[8] D. Harris y S. Harris. *Digital design and computer architecture.* Morgan Kaufmann, Waltham, Massachusetts, 2010.

[9] E. Larraza Mendiluze, N. Garay Vitoria, J. I. Martin, J. Muguerza, T. Ruiz Vazquez, I. Soraluze, J. F. Lukas, y K. Santiago. Game-Console-based projects for learning the computer input/output subsystem. *IEEE Transactions on Education,* vol. 56, no. 4, páginas 453–458, 2013.

[10] D. A. Patterson y J. L. Hennessy. *Computer Organization and Design ARM Edition: The Hardware Software Interface.* Morgan Kaufmann, 2016.

[11] L. D. Pyeatt. *Modern assembly language programming with the ARM processor.* Newnes/Elsevier, Kidlington, UK; Cambridge, USA, 2016.

[12] G. Vik, J. Whitington, C. Hetherington, y F. Graute. StrongEd - a programmer's text editor for RISC OS. Disponible en `http://stronged.iconbar.com/`. Último acceso: Enero 2019.

[13] A. J. Villena, R. Asenjo, y F. J. Corbera. *Prácticas de Ensamblador Basadas en Raspberry Pi.* Repositorio Institucional de la Universidad de Málaga, RiUMA, 2016. Disponible en `https://antoniovillena.es/imagenes_foros/LibroDePracticas.pdf`