Arquitectura de Computadores en seis créditos ECTS

J. Ortega, M. Anguita

Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I.I.T., Universidad de Granada
{julio, manguita}@.atc.ugr.es

Resumen. En este trabajo se describe el curso de Arquitectura de Computadores que se imparte en el grado en Ingeniería Informática de la Universidad de Granada. Se trata de un curso de seis créditos ECTS, y por lo tanto se dispone de alrededor de 150 horas para las clases teóricas y prácticas, y el trabajo personal y la evaluación de los estudiantes. Además, hay que tener en cuenta que se imparte en un cuatrimestre, coincidiendo con otras cuatro asignaturas del grado. En el artículo se analiza la organización de la docencia a partir de algunas de las teorías recientes sobre el proceso de aprendizaje, los siete principios para una buena educación de pregrado y de la interacción entre la asignatura de Arquitectura de Computadores, contemplada como disciplina que identifica los cuellos de botella del computador y propone técnicas para evitar sus efectos, y otras asignaturas (anteriores, concurrentes, y posteriores).

Palabras clave: Aprendizaje constructivo y colaborativo, Arquitectura de Computadores, Optimización de código, Paralelismo.

Abstract. A description of a semester course on Computer Architecture is provided in the paper. This course of six ECTS credits comprises about 150 hours of theoretical lectures and experimental activities, personal work, and evaluation tasks, and coincides with other semester courses on four different subjects. The way the teaching on Computer Architecture has been organized by taking into account some recent theories about the characteristicas of the learning process and the interaction among different (previous, concurrent, and future) subjects, are analyzed in the paper.

Keywords: Code Optimization, Collaborative learning, Computer Architecture, Constructional learning, Parallel Architectures.

1 Introducción

La Arquitectura de Computadores constituye uno de los pilares de la Ingeniería de Computadores. Una asignatura de Arquitectura de Computadores debe estudiar los principios fundamentales que intervienen en el diseño de nuevos dispositivos y

plataformas de cómputo para satisfacer los requisitos de las aplicaciones emergentes. Entre dichos principios están la ley de Amdahl, el aprovechamiento de la localidad espacial y temporal de los programas para reducir los tiempos medios de acceso a memoria, el procesamiento especulativo para mejorar el rendimiento de los procesadores segmentados con paralelismo entre instrucciones, etc.

Según lo que se establece en su guía docente, la asignatura de Arquitectura de Computadores del grado en Ingeniería Informática de la Universidad de Granada aborda la estructura y la clasificación de las arquitecturas paralelas (procesadores, multiprocesadores, multicomputadores y sistemas distribuidos) y del paralelismo de las aplicaciones. Por tanto, abarca el estudio de las arquitecturas con paralelismo entre instrucciones (ILP), los procesadores multihebra, multinúcleo y multiprocesadores; la evaluación de sus prestaciones; y su programación eficiente, incluyendo tanto los mecanismos y algoritmos básicos de optimización de código, como la programación paralela.

En la guía docente también se recogen los objetivos de la asignatura expresados como resultados de aprendizaje del estudiante ("el estudiante será capaz de ….."). Como puede verse en la Tabla 1, estos objetivos proponen un estudio de la Arquitectura de Computadores desde el punto de vista del aprovechamiento del paralelismo que proporcionan las arquitecturas actuales, más que desde el punto de vista del diseñador de dichas arquitecturas. A la vez que introducir los conceptos fundamentales de las arquitecturas paralelas que necesita un ingeniero de computadores, esta perspectiva nos permite poner de manifiesto la importancia asociada al conocimiento de las características de la arquitectura del computador para el desarrollo de aplicaciones eficientes, justificando la presencia de esta asignatura obligatoria en el segundo curso del grado en Ingeniería Informática.

Tabla 1. Objetivos de la Asignatura Arquitectura de Computadores

El estudiante será capaz de ….
• Utilizar y explicar las diferentes clasificaciones de arquitecturas paralelas.
• Distinguir entre procesamiento paralelo y procesamiento distribuido, y asociarlos con las arquitecturas que se utilizan para implementarlos.
• Relacionar el paralelismo implícito en una aplicación con las arquitecturas que lo aprovechan.
• Afrontar el análisis y el diseño de un núcleo ILP.
• Describir el papel del compilador/programador en una arquitectura ILP.
• Implementar código que aproveche la arquitectura ILP.
• Distinguir entre las prestaciones del procesador, del compilador y del programa.
• Explicar los conceptos de ganancia en prestaciones y la ley de Amdahl.
• Describir la estructura y organización de arquitecturas multihebra, multinúcleo y multiprocesador.
• Explicar el papel del compilador en una arquitectura multinúcleo y multiprocesador, expresar un algoritmo de forma apropiada para dichas arquitecturas, y escribir código eficiente para ellas.
• Explicar la necesidad de mantener la coherencia de memoria.
• Afrontar el análisis y diseño de protocolos de mantenimiento de coherencia en

multinúcleos y multiprocesadores.
- Distinguir entre los diferentes tipos de modelos de consistencia de memoria y explicar la influencia del modelo de consistencia de memoria en las prestaciones de un computador.
- Implementar código que aproveche el modelo de consistencia de memoria y las instrucciones máquina de sincronización, e implementar mecanismos básicos de sincronización.

El paralelismo se puede considerar el paradigma que ha definido la evolución de la arquitectura de los computadores contemplada como la traducción de las posibilidades que ofrece la tecnología a capacidades reales y a mejora de prestaciones [1]. Tal y como se indica en [2], la forma en que se ha producido la evolución de los computadores se puede resumir en *paralelismo* y *localidad*: el paralelismo, aplicado en distintos niveles, permite reducir los tiempos de procesamiento gracias a la realización simultánea de operaciones y la localidad permite reducir los retardos en los accesos a los datos y las instrucciones. El compromiso a alcanzar en el uso de recursos (para conseguir el dispositivo/computador con mejores prestaciones) establece una interacción entre paralelismo y localidad que se manifiesta en problemas como la coherencia de cache, el modelo de consistencia de memoria, la necesidad de estructuras de comunicación entre procesadores y memoria y entre procesadores, y permite poner de manifiesto los conceptos esenciales para diseñar y evaluar un computador, y para desarrollar aplicaciones que utilicen eficientemente los recursos disponibles.

La Sección 2 resume las características principales del modelo de aprendizaje que hemos considerado. Después, en la Sección 3 se describe el contexto en el que se imparte la asignatura Arquitectura de Computadores dentro de los estudios de grado en Ingeniería Informática de la Universidad de Granada. Finalmente, la Sección 4 analiza la aproximación de enseñanza/aprendizaje que se ha implementado a partir de las ideas y consideraciones descritas en las secciones previas, y la Sección 5 cierra el artículo con las conclusiones, estructuradas a partir de los siete principios para una buena educación de pregrado [9].

2 El marco teórico del proceso de aprendizaje

Una estrategia de enseñanza/aprendizaje adecuada debe partir de teorías pedagógicas probadas y ampliamente aceptadas. Así, en esta Sección se describen las principales ideas de las teorías que sustentan la metodología utilizada en la asignatura, y se presentan algunas de las ideas más relevantes de teorías del conocimiento recientes.

El aprendizaje constructivo, o constructivismo, y el aprendizaje cooperativo son propuestas comúnmente utilizadas actualmente. El constructivismo propone evitar la memorización "enfatizando los contextos de aprendizaje" de manera que el conocimiento se construye a través de actividades lo más próximas posible al mundo real. De hecho, los modelos más aceptados actualmente indican que el aprendizaje se construye, y que la memoria no es un almacén donde se introducen cosas que luego se podrán sacar sin más: memorizar es más que almacenar [3].

La memorización es la primera etapa de un proceso de síntesis que hace posible entender de forma profunda y personal aquello que se ha recibido (leído, escuchado, visto, etc.), y hace intervenir a toda la mente ya que implica creatividad y juicio. El cerebro transforma los eventos cargados en la memoria a corto plazo en pensamientos memorizados tras un cierto tiempo, necesario para que sea posible esa transición desde la memoria a corto a la de largo plazo [5]. Estos dos tipos de memoria se basan en procesos biológicos diferentes. De hecho, deben sintetizarse cierto tipo de proteínas para conseguir la memorización a largo plazo. Los resultados de ciertos experimentos neurológicos han puesto de manifiesto que el paso a la memoria a largo plazo no sólo cambia la concentración de neurotransmisores para reforzar ciertas conexiones, sino que también aparecen nuevas conexiones (nuevas sinapsis). Por tanto, la memorización a largo plazo implica cambios anatómicos además de bioquímicos, con lo que debe activarse un gen para producir dichas proteínas y responder al entorno (en este caso, al proceso de aprendizaje). Además, la consolidación de la memoria a largo plazo implica una cadena de interacciones relativamente larga entre el hipocampo y el córtex cerebral. El hipocampo permite interrelacionar y construir recuerdos utilizando diferentes tipos de imágenes (visuales, espaciales, de tacto, etc.) almacenadas en diferentes áreas del cortex, junto con recuerdos recientes y antiguos. Una vez se ha consolidado el recuerdo, éste desaparece del hipocampo y permanece en el córtex.

Esta perspectiva es distinta de la que se nos ofrece cuando se piensa en Internet como una ayuda a la memoria personal [4]. Más que una ayuda, Internet sustituye la memorización, reemplazándola por accesos a bases de datos. Como se ha indicado más arriba, nuestra memoria no funciona como un disco duro que accede a la información a través de la correspondiente dirección. Mientras que la memoria de un computador recibe los bits y los almacena sin más, el cerebro continúa procesando la información tras recibirla y la calidad del recuerdo depende de la forma en que dicho procesamiento se ha llevado a cabo. Nuestra mente mejora mientras trabaja para memorizar nuevos conceptos.

Por otro lado, la llegada de demasiada información puede sobrecargar nuestra memoria de corto plazo y afectar negativamente a la consolidación de los recuerdos y a la construcción de pensamientos estables. Hay que tener en cuenta que la clave de la consolidación de la memoria es la atención. La atención produce cambios cuantificables en el cerebro, dado que las neuronas del córtex envían señales que hacen que las neuronas del cerebro intermedio produzcan dopamina, y construyen un camino para que la dopamina llegue al hipocampo. Una vez en el hipocampo, la dopamina hace posible la consolidación de la memoria explícita al activar los genes que estimulan la síntesis de nuevas proteínas. La recepción de un número demasiado elevado de mensajes sobrecarga la memoria a corto plazo y hace difícil mantener la atención en algo [6]. Gracias a la plasticidad de nuestro cerebro, es posible que la información sea procesada rápidamente, pero sin la atención que permite mantenerla. Los cerebros se hacen expertos en olvidar.

Desde esta perspectiva, conocer la interrelación entre los niveles de descripción de un computador es fundamental puesto en un mercado global en el que la competitividad es despiadada, no basta con que las cosas funcionen sino que deben funcionar de la forma más eficiente posible. En el aprendizaje de cada asignatura, debe fomentarse el estudio interrelacionado de sus contenidos junto con los de otras

asignaturas, y plantear prácticas y problemas que pongan de manifiesto la interrelación existente entre los distintos niveles y la importancia de tenerlos en cuenta en el diseño óptimo de plataformas y aplicaciones. Definir un conjunto de preguntas pertinentes es esencial para el aprendizaje y la modificación de los modelos mentales erróneos con que los estudiantes se enfrentan a una asignatura. Según lo que hemos expuesto, no se puede aprender hasta que no se han formulado las preguntas adecuadas debido a que dichas preguntas permiten indexar la información que tenemos en memoria cuando buscamos la correspondiente respuesta [2]. Al intentar responder esas preguntas el estudiante puede comprobar que su modelo mental debe completarse para poder alcanzar esas respuestas, y eso es lo que el estudio de la nueva asignatura le va a ofrecer precisamente.

Por otra parte, existe otra razón más para fomentar el aprendizaje interrelacionado de asignaturas en los distintos cursos de una carrera. La elaboración de los Planes de Estudios no tiene en cuenta el número de asignaturas entre las que los estudiantes deben distribuir su tiempo. El número de exámenes y de actividades a realizar puede ser tan elevado que dificulta la posibilidad de profundizar en los conceptos y asentar lo aprendido. Por ejemplo, se podrían hacer estimaciones de tiempo mínimo de cómputo y perfil de uso de la memoria (fallos de cache, localidad de acceso a memoria, etc.) en la asignatura de Arquitectura de Computadores, utilizando alguno de los algoritmos presentados en asignaturas de algoritmia o de programación, impartidas con anterioridad, o simultáneamente. La interrelación entre asignaturas debería ser un objetivo importante desde el punto de vista de la coordinación docente, y no solo entre asignaturas impartidas por un mismo ámbito de conocimiento o departamento. En la siguiente sección se describe el contexto en el que se imparte Arquitectura de Computadores.

3 El contexto de la asignatura

La asignatura Arquitectura de Computadores se imparte en el segundo cuatrimestre del segundo curso del grado en Ingeniería Informática de la Universidad de Granada. Se trata de una asignatura obligatoria para todos los estudiantes del grado, que debe proporcionar formación específica de rama, y tiene asignados seis créditos ECTS. Estos seis créditos suponen un total de 150 horas de actividades que se han distribuido entre 30 horas de clases teóricas, otras 30 de clases prácticas, y 90 de actividades y trabajo no presencial para los estudiantes. Se supone, por tanto, un total de 1.5 horas de trabajo no presencial del estudiante por cada hora de clase (práctica o teórica), y ocho horas de trabajo (presencial y no presencial) a la semana. En la dirección http://grados.ugr.es/informatica/pages/infoacademica/estudios está la información detallada del plan de estudios del grado en Ingeniería Informática, incluyendo las guías de las asignaturas obligatorias de formación básica y de rama, y las obligatorias y optativas de cada una de las cinco especialidades contempladas en el título (Computación y Sistemas Inteligentes, Ingeniería del Software, Ingeniería de Computadores, Sistemas de Información, y Tecnologías de la Información).

La formación básica del título comprende un total de 60 créditos ECTS distribuidos en diez asignaturas de seis créditos. Entre estas asignaturas se encuentra

Tecnología y Organización de Computadores (TOC), que proporciona los contenidos relacionados con el hardware de los computadores y su funcionamiento desde el nivel de lógica digital hasta el de transferencia entre registros, y Fundamentos Físicos y Tecnológicos (FFT), que considera los niveles físicos del hardware del computador y los principios físicos relacionados. El resto de asignaturas de formación básica se centran en los fundamentos de la programación (Fundamentos de la Programación, FP, y Metodología de la Programación, MP) y el software (Fundamentos del Software, FS), las competencias que todo ingeniero debe tener en el ámbito de las matemáticas, y los aspectos que relacionan la ingeniería con la empresa y la sociedad. Las asignaturas de FP y MP, y fundamentalmente TOC son las que están más directamente relacionadas con los contenidos previos necesarios para Arquitectura de Computadores, además de ciertos conocimientos que se adquieren en FFT y en las asignaturas de matemáticas.

Arquitectura de Computadores es una de las 15 asignaturas de seis créditos obligatorias de rama que se agrupan en cuatro materias: Programación e Ingeniería del Sofware; Bases de Datos, Sistemas de Información y Sistemas Inteligentes; Estructura y Arquitectura de Computadores; y Sistemas Operativos, Sistemas Distribuidos y Redes. La materia de Estructura y Arquitectura de Computadores está constituida por dos asignaturas además de Arquitectura de Computadores: Estructura de Computadores, e Ingeniería de Servidores. La asignatura Estructura de Computadores, que precede a la de Arquitectura de Computadores, estudia el repertorio de instrucciones , la programación en ensamblador, y la estructura de un computador en el nivel de lenguaje máquina, incluyendo el sistema de memoria, de entrada/salida, los buses, y la organización del procesador (control cableado/microprogramado, segmentación de cauce, CISC/RISC, etc.). En el primer cuatrimestre del tercer curso, y por tanto después de Arquitectura de Computadores, se imparte la asignatura Ingeniería de Servidores centrada en los componentes, el diseño y la configuración de un servidor medio, con especial hincapié en el almacenamiento, y los aspectos relacionados con el montaje e instalación, la administración y la evaluación de prestaciones de los servidores.

Asignadas al mismo cuatrimestre que Arquitectura de Computadores hay otras cuatro asignaturas de seis créditos ECTS (Algoritmos, Inteligencia Artificial, Fundamentos de Bases de Datos, y Fundamentos de Ingeniería del Software) y además de Estructura de Computadores, hay otras cuatro asignaturas obligatorias de rama más en el cuatrimestre anterior (Estructuras de Datos, Sistemas Operativos, Programación Orientada a Objetos, y Sistemas Concurrentes y Distribuidos). Aunque, en la práctica, no se puede garantizar que los estudiantes cursen las asignaturas en el orden establecido según los cursos y los cuatrimestres, es conveniente tener en cuenta qué es lo que sucede con más frecuencia y, como se ha indicado en la sección anterior, puede ser útil aprovechar la formación previa recibida por los estudiantes y la que están recibiendo al mismo tiempo que la propia asignatura. Por ejemplo, es posible programar actividades que pongan de manifiesto relaciones entre asignaturas diferentes, y también es necesario considerar la carga de trabajo a que pueden estar sometidos los estudiantes de cara a organizar la propia asignatura.

Además del contexto que define el plan de estudios en el que se imparte la asignatura, también hay que tener en cuenta el contexto externo determinado por las posibilidades profesionales de la titulación y las especialidades, y el marco científico-

tecnológico de la propia disciplina. La evolución de las arquitecturas está determinada tanto por las posibilidades que ofrece la tecnología como por los requisitos que establecen las aplicaciones dominantes en el mercado para disponer de computadores que puedan abordarlas eficientemente. Esa interacción entre tecnología, aplicaciones, mercados, y arquitectura de computadores ha originado una clara tendencia hacia el aprovechamiento del paralelismo (a distintos niveles) en el computador. Por un lado siempre han existido aplicaciones con una demanda significativa que necesitan más prestaciones que las que se pueden alcanzar en computadores con un procesador. Además, el ritmo de mejora exponencial asumido por los fabricantes de microprocesadores a partir de la ley de Moore, y las limitaciones en la potencia disipada por las microarquitecturas superescalares, han dado lugar a la generalización de microarquitecturas multinúcleo para los microprocesadores, de forma que se dispone de varios núcleos de procesamiento incluso en computadores personales. Al mismo tiempo, las mejoras en las prestaciones de comunicación han promovido multicomputadores, plataformas distribuidas, y paradigmas de cómputo que aprovechan la potencia de varios computadores, como la computación en *grid* o el "*cloud computing*". Incluso dentro del reciente paradigma denominado *Internet de las cosas* se pone de manifiesto la capacidad del procesamiento paralelo y distribuido para desarrollar aplicaciones innovadoras.

4 La enseñanza/aprendizaje de Arquitectura de Computadores

El planteamiento de la asignatura Arquitectura de Computadores enfatiza el papel de la arquitectura como disciplina que identifica los cuellos de botella del computador y propone técnicas para evitar sus efectos, y asume que un conocimiento de dichas técnicas resulta útil al programador de aplicaciones (normalmente programador en lenguaje de alto nivel) para generar códigos eficientes. Por lo tanto, el punto de vista de la asignatura es el de la optimización de código a partir de las características de la arquitectura.

El modelo de capas que comúnmente se utiliza para describir el computador, y como consecuencia, para identificar distintas áreas de conocimiento y organizar administrativamente la docencia en los Departamentos Universitarios, también ha extendido una visión de caja negra de un nivel con respecto a otro, y si bien este planteamiento es aplicable en la mayoría de los casos, y sobre todo cuando se trata de proporcionar interfaces que aíslen al usuario de las complejidades de la máquina, no es aconsejable como la única perspectiva desde la que formar Ingenieros Informáticos. Así, usualmente se asume que técnicas como el uso de jerarquías de memoria, la ejecución desordenada y especulativa, los buffers de almacenamiento, etc., implementadas por las arquitecturas para mejorar las prestaciones del computador, son transparentes para el programador en lenguaje de alto nivel. Sin embargo, como mostraremos en esta Sección a través de algunos ejemplos, esto no es cierto en muchos casos y se precisa tener en cuenta ciertos detalles de los niveles más bajos para generar códigos de alto nivel eficientes, e incluso correctos.

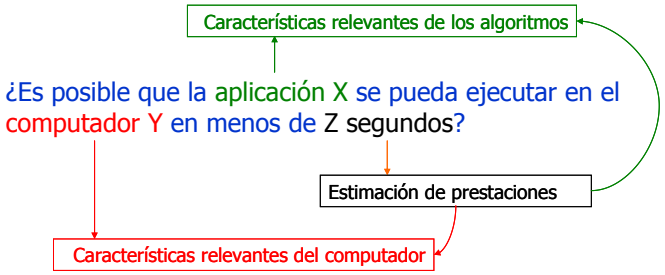

Figura 1. Punto de partida de la asignatura Arquitectura de Computadores

En la Figura 1 se resume la perspectiva de la asignatura Arquitectura de Computadores. Se parte de una pregunta para cuya respuesta la asignatura debe proporcionar al estudiante las correspondientes herramientas. Responder a esa pregunta implica la capacidad para analizar las características relevantes de los algoritmos utilizados en la aplicación, y la capacidad para estimar las prestaciones relevantes del computador a partir de sus características. Por otra parte, la mejora en las prestaciones puede conseguirse, tanto por cambios en las características del computador, cuyo análisis deben abordar los estudiantes, como por cambios en los programas que tengan en cuenta las características de la arquitectura del computador.

Tabla 2. Contenidos de la asignatura Arquitectura de Computadores

Tema 1. Arquitecturas paralelas: clasificación y prestaciones
Clases de arquitecturas paralelas
Evaluación de Prestaciones
Tema 2. Programación paralela
Herramientas, estilos, estructuras
Proceso de paralelización
Evaluación prestaciones
Tema 3. Arquitecturas con paralelismo a nivel de hrebra (*thread*)
Arquitecturas
Coherencia
Consistencia
Sincronización
Tema 4. Arquitecturas con paralelismo a nivel de instrucción (ILP)
Espacio ILP
Emisión de instrucciones
Saltos y excepciones
VLIW
Tema 5. Arquitecturas de propósito específico
SIMD, GPU y proc. de red

En la Figura 2 se ubican los contenidos de la asignatura (indicados en la Tabla 2) en el contexto de los procesadores y las arquitecturas paralelas. Tanto en un caso como en otro, la interacción con la jerarquía de memoria plantea una serie de problemas cuyas soluciones deben ser consideradas en el contexto de los procesadores paralelos y/o en el de las arquitecturas paralelas. Como se pone de manifiesto en la Figura 2, los contenidos de la asignatura surgen como consecuencia del aprovechamiento del paralelismo en sus distintos niveles para mejorar las prestaciones de los computadores. La práctica de esos contenidos vendrá, fundamentalmente, de la mano de la optimización de código y de la programación paralela.

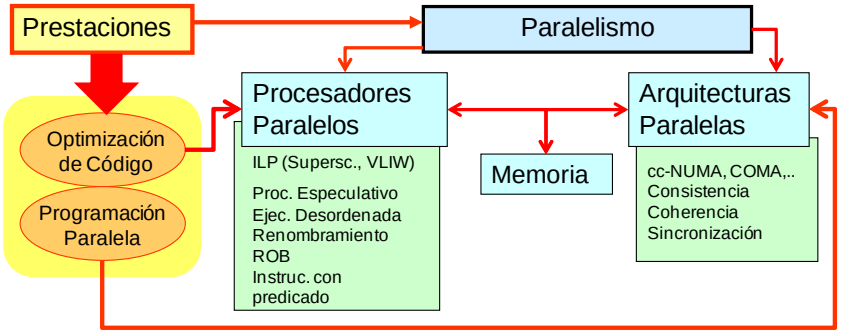

Figura 2. Interrelación entre contenidos, conceptos y actividades

La Figura 2 ilustra la perspectiva que se utiliza para la enseñanza de la asignatura basada en la interrelación de los distintos tópicos para crear árboles de relaciones. A continuación se ilustra la perspectiva de la asignatura utilizando alguno de los contenidos de la asignatura, como es el caso de la consistencia de memoria y la sincronización.

Los *modelos de consistencia* de memoria permiten poner de manifiesto la interacción entre niveles de descripción del computador y, a través del análisis del papel de las interfaces correspondientes, precisar la perspectiva con que se aborda la asignatura Arquitectura de Computadores. El modelo de consistencia de memoria de un computador de memoria compartida especifica la perspectiva que el programador tiene de la memoria, permitiendo deducir el valor de una variable tras su lectura. Por tanto, el modelo de consistencia de memoria constituye la interfaz de memoria en un multiprocesador de memoria compartida. En el caso de un computador con un único procesador, el valor que se tendría de una lectura correspondería al de la última escritura realizada en esa variable, y esa última escritura se determina a partir del orden secuencial que el programa define en las instrucciones, es decir, el *orden de programa*. La extensión de este modelo a un multiprocesador es el denominado modelo de *consistencia secuencial*, en el que todas las operaciones de memoria son vistas una a una, según una secuencia, por todos los procesadores (atomicidad), y las operaciones de cada procesador se ejecutan según el orden de programa. Así, en el caso de los fragmentos de código de la Figura 3, el modelo de consistencia secuencial permite que el programador sepa que C=1. Por otra parte, los códigos de la Figura 3 constituyen un ejemplo de *sincronización*, a través de los accesos a B, para coordinar el acceso a la variable A.

Inicialmente A=0 y B=0 (A y B variables compartidas)	
Hebra en procesador P1 A=1; B=1;	Hebra en procesador P2 While (B==0) {}; C=A;

Figura 3. Fragmentos de código en procesadores P1 y P2

El modelo de consistencia secuencial facilita el trabajo del programador de alto nivel, pero también hay que tener en cuenta el modelo de consistencia de memoria que utiliza el procesador. Es decir, la perspectiva de la memoria que se tiene a nivel de lenguaje máquina. A este nivel, la necesidad de mejorar las prestaciones del procesador ha hecho que la mayoría de los microprocesadores comerciales actuales implementen los denominados modelos de consistencia relajados (*relaxed models* o *weak models*) que permiten cambiar el orden de algunas (o en los casos más agresivos, todas) lecturas y escrituras. Así, en los programas multihebra, las distintas hebras pueden tener distintas imágenes de la memoria como consecuencia del uso de estos modelos de memoria relajados. Por ejemplo, si en el caso de la Figura 3, el procesador P1 desordena las escrituras en A y B, el resultado de C podría ser igual a 0. Esta situación hace que para mantener la consistencia secuencial, el compilador tendría que evitar determinados cambios en el orden de las instrucciones, o insertar instrucciones (incluidas en el propio repertorio del procesador) para que el procesador cambie dinámicamente el orden de ciertos accesos a memoria. El coste asociado a esas optimizaciones que no puede aplicar el compilador en determinadas situaciones, o a la ejecución de las instrucciones que hay que incluir para garantizar la consistencia secuencial, puede afectar seriamente a las prestaciones del programa. El conocimiento de estas situaciones es útil para el programador de alto nivel, que podría intentar evitar estos costes para mantener el nivel de prestaciones requerido.

Como puede verse, la aproximación al concepto de modelo de consistencia secuencial permite ilustrar el punto de vista desde el que se plantea la asignatura de Arquitectura de Computadores. Por un lado está la perspectiva del programador, que debe aprovechar los recursos de la máquina de la forma más eficiente posible. Para ello debe ser consciente de los recursos que ofrece la arquitectura a través de la correspondiente interfaz y de qué condicionantes genera en la traducción a código máquina que el compilador va a generar a partir del programa que ha elaborado. La necesidad de medir y alcanzar el nivel de prestaciones que requieren las aplicaciones se encuentra en la base de este planteamiento.

Como se ha comentado en la Sección 2, la construcción de mapas que interrelacionen conceptos es importante para memorizar y conocer. En la Figura 4 se proporciona una descripción informal de relaciones entre distintos contenidos de la asignatura. El árbol de relaciones de la figura se puede completar añadiendo todos los conceptos de la asignatura (no se han incluido aquí por falta de espacio) y las dependencias entre ellos. Como punto de partida se ha considerado un programa de alto nivel a partir del que el compilador genera el correspondiente programa en lenguaje máquina. Precisamente el repertorio de instrucciones máquina constituye la interfaz entre el software y el hardware, y penetrando en el significado de esas instrucciones se pueden ir presentando conceptos como consistencia de memoria, coherencia, sincronización, procesamiento especulativo, ejecución desordenada de instrucciones, etc., cuya influencia en las prestaciones se puede analizar.

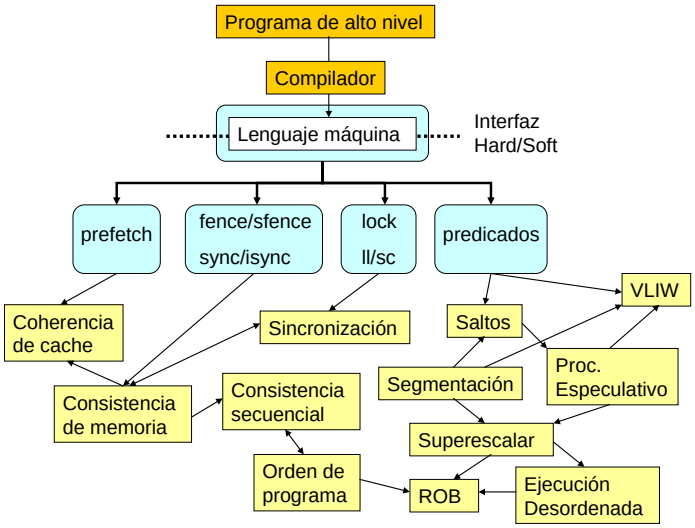

Figura 4. Árbol de relaciones entre contenidos de la asignatura

El constructivismo se aplica mediante la elaboración de programas en C, incluyendo la biblioteca OpenMP para la elaboración de programas paralelos con varias hebras, a través de los que el estudiante puede experimentar con distintas estrategias para mejorar las prestaciones. Para ello, se explorará tanto el paralelismo aprovechable en arquitecturas multinúcleo, como otras alternartivas de optimización de código aplicables a núcleos superescalares. El uso de simuladores en la asignatura, se ha considerado interesante como herramientas y animaciones para que el estudiante pueda entender los conceptos de los procesadores ILP y de coherencia de cache, y para la realización de problemas. Precisamente es en la realización de problemas en clase donde se van a aplicar los principios del aprendizaje colaborativo, ya que se constituirán grupos de estudiantes para presentar en clase los problemas planteados en las distintas relaciones.

5 Conclusiones

Para concluir este trabajo, se resumen las principales actividades y estrategias de la asignatura Arquitectura de Computadores desde el punto de vista de la puesta en práctica de los siete principios para una buena educación de pregrado [9]. El primero de estos siete principios hace referencia al contacto entre profesores y estudiantes. Para reforzar ese contacto se hace un uso extensivo del correo electrónico o herramientas como SWAD [10], animando a los estudiantes a que hagan uso de ellos para expresar sus impresiones acerca de la asignatura. En relación con este primer principio es importante conseguir el mayor acercamiento posible a los estudiantes, conocerles por su nombre en la medida de lo posible (en grupos no demasiado numerosos) y ser sensible a la idiosincrasia y a la problemática propia del contexto

social en el que se desenvuelven los estudiantes. En cuanto al segundo principio, centrado en la cooperación entre estudiantes, en la asignatura se elaboran relaciones de problemas sobre los contenidos de cada tema, y se definen grupos de trabajo para resolverlos y presentarlos en clase. Uno de los miembros del grupo, elegido aleatoriamente por el profesor, expondrá el problema y la calificación que obtenga será la que se asigne a todo el grupo. Con ello se fomenta la implicación de todo el grupo en el trabajo y el aprendizaje colaborativo.

El tercer principio incide sobre el aprendizaje activo. Según el modelo de memorización que se ha descrito, basado en el establecimiento de relaciones entre los contenidos que hemos descrito brevemente, el constructivismo constituye una buena estrategia didáctica, al buscar la generación del conocimiento a través de la realización de actividades lo más próximas al mundo real, más que la simple memorización de contenidos. Para ello, debe conseguirse que los estudiantes sean parte activa en su propio proceso de aprendizaje. Este tercer principio se tiene en cuenta en la asignatura de Arquitectura de Computadores a través de la resolución de problemas relacionados con los contenidos de cada uno de los temas de la asignatura y mediante las prácticas. Se pretende que los problemas correspondan a situaciones realistas, aunque con un nivel de complejidad moderado que permita su resolución en tiempos razonables. Las prácticas están relacionadas con la programación paralela multihebra y con la optimización de código. A través de ellas, el estudiante comprueba como el conocimiento de la arquitectura del computador le permite mejorar el rendimiento de sus programas, a la vez que aprende a utilizar herramientas de programación y a evaluar prestaciones.

El cuarto principio correspondiente a la necesidad de proporcionar realimentación a los estudiantes se fomenta en la asignatura al resolver los problemas en clase y al proporcionar los resultados de las pruebas a través de Internet, mediante herramientas como SWAD. Como ya se indicó en relación con el primero de los principios, el uso del correo electrónico y de SWAD facilita el acceso del estudiante al profesor para plantear cuestiones y recibir la correspondiente realimentación con mayor rapidez.

El quinto principio se refiere a la organización eficiente del tiempo. Cada una de las prácticas de la asignatura incluye ejercicios que deben entregarse dentro de unos plazos establecidos. Existe una distribución temporal de los distintos temas, prácticas, clases de problemas, establecida y conocida por el estudiante desde el primer día de clase y que se ajusta a la dificultad y amplitud de los distintos contenidos (a medida que se sucedan los cursos académicos, sin duda esta distribución temporal se irá mejorando). El sexto principio busca fomentar las altas expectativas en los estudiantes, y para ello se refuerza positivamente el esfuerzo y el trabajo bien hecho poniéndolo de manifiesto en las clases de problemas, y difundiendo los trabajos de calidad realizados por los estudiantes. Las calificaciones obtenidas por los estudiantes son conocidas por todos, de forma que se ponga de manifiesto que el esfuerzo, independientemente de los resultados que conlleva en cuanto al aprendizaje, también se refleja en dichas calificaciones.

Por último, el séptimo principio corresponde al respeto a las distintas formas de aprendizaje. La asignatura tiene un componente teórico importante, pero se trabaja sobre relaciones de problemas y prácticas en las que se aplica dicha teoría. Al mismo tiempo, se proporcionan simuladores (WinDLX, WinSuperDLX [11]) y animaciones [12] que permiten la realimentación de lo estudiado (cuarto principio) y ofrecen

alternativas para acercarse a los conceptos. La asignatura incluye actividades de programación, aparte de las de diseño y estimaciones cuantitativas utilizando modelos matemáticos sencillos. Además, se contemplan formas distintas de evaluación que incluyen, tanto el uso de pruebas para la evaluación continua a través de la entrega de actividades, la presentación de problemas y entrevistas en prácticas, como un examen final escrito.

Agradecimientos: a los compañeros del Departamento de Arquitectura y Tecnología de Computadores que imparten las prácticas de Arquitectura de Computadores en el curso 2011/2012: Ana Cara, Maribel García Arenas, Enrique J. Fernández Sánchez, Luis J. Herrera, Christian Morillas, y Javier Pérez Florido.

Referencias

1. Ortega, J.; Anguita, M.; Prieto, A.:"Arquitectura de Computadores". Thomson-Paraninfo, 2005.
2. Culler, D.E.; Singh, J.P.; Gupta, A.:"Parallel Computer Architecture. A Hardware/Software Approach". Morgan Kaufmann Pub. Inc., 1999.
3. Bain, K.:"Lo que hacen los mejores profesores universitarios". Publicacions Universitat de València, 2006.
4. Carr, N.: "The Shallows. What the Internet is doing to our Brains". Ed. W.W. Norton, 2010 (Edición en Castellano en Santillana Ediciones Generales S.L., 2011).
5. Kandel, E.R.:"En busca de la memoria: una nueva ciencia de la mente". Katz Editores, 2007.
6. Klingberg, T.:"The overflowing brain: Information overload and the limits of working memory". Oxford University Press, 2008.
7. ACM/AIS/IEEE-CS (The Joint Task Force for Computing Curricula 2005): "Computing Curricula 2005. The Overview Report". Septiembre de 2005. (http://www.acm.org/education/curricula-recommendations)
8. Woh, M.; Mudge, T.; Chakrabarti, C.:"Mobile Supercomputers for the Next-Generation Cell Phone". IEEE Computer, pp.81-85. Enero, 2010.
9. Chickering, A.W.: Gamson, Z.F.:"Applying the Seven Principles for Good Practice in Undergraduate Education". En "New Directions for Teaching and Learning", vol. 47, San Francisco: Jossey-Bass Inc.
10. Cañas, A.; Martínez Ortigosa, E.; Fernández, F.J.; Anguita, M.; Ros, E.; Díaz, A.F.: "Plataforma de Teleformación SWAD". Conferencia IADIS Ibero-Americana WWW/Internet 2004, pp. 89-96, 2004.
11. WinSuperDLX: http://osl.ugr.es/software-libre-en-la-ugr/winsuperdlx/
12. Animación de Coherencia de Cache: http://lorca.act.uji.es/projects/ccp/