ARQUITECTURA DE SISTEMAS PARALELOS 1.

PROGRAMA RESUMIDO 2006/2007.

Escuela Técnica Superior de Ingeniería Informática. Cuarto Curso.

Profesores: D. Saturnino Vicente Díaz (coord.), D. Alejandro Linares Barranco, D. Francisco Gómez Rodríguez, D. Fernando Díaz del Río, D. Antón Civit Balcells.

Dpto. de Arquitectura y Tecnología de Computadores. L4 planta baja, derecha.

Página Web: http://www.atc.us.es/asignaturas/asp1

Temario de la asignatura Arquitectura de Sistemas Paralelos 1.

TEMA 0 PRESENTACIÓN DE LA ASIGNATURA (T: 1H).

TEMA 1 ANÁLISIS DE PRESTACIONES (T: 3H P: 1H).

1.1 Repaso de los modelos y principales conceptos de arquitectura.

- Arquitectura, organización y tecnología.
- Ejecución secuencial frente a ejecución paralela.

1.2 Parámetros cuantitativos para la evaluación de prestaciones.

- Rendimiento, productividad y coste.
- Parámetros clásicos: MIPS, MFLOPS, etc.
- Parámetros del tiempo de ejecución: CPI, frecuencia y número de instrucciones.
- Ley de Amdahl y ley de localidad.
- Modelado cuantitativo del acceso a memoria (bloqueos de memoria y $CPI_{bloqueo}$).
- Medias de rendimiento.

1.3 Programas para la evaluación de prestaciones (*benchmarks*).

- *Benchmarks* de juguete.
- *Benchmarks* sintéticos.
- Evaluación con programas reales.
- Colecciones de programas: el caso de los SPECInt y SPECFP.

1.4 Jerarquía de memoria: Optimizaciones

TEMA 2 ARQUITECTURAS ENCADENADAS BÁSICAS (T: 12H P: 3H).

2.1 Introducción: concepto de encadenamiento de instrucciones.

2.2 Juegos de instrucciones orientados al encadenamiento.

- Microinstrucciones, implementación cableada y etapas de una instrucción.
- Características de un “buen” procesador encadenado: la importancia de la decodificación y de los accesos a memoria.
- El problema del encadenamiento en los procesadores CISC y ventajas de los RISC.
- Factores involucrados en la aceleración de un procesador encadenado: CPI, periodo, número de instrucciones y bloqueos.
- Relación entre el juego de instrucciones y la compilación.

2.3 Descripción del procesador ejemplo (DLX).

- Características principales: tipos de datos, juegos de registros, direccionamiento, operaciones.
- Juego y formatos de instrucciones.

2.4 Implementación del encadenamiento del DLX.

- Implementación secuencial.
- Implementación orientada al encadenamiento.
- Detalle de la implementación.
- Consideraciones tecnológicas del encadenamiento.

2.5 Dependencias, riesgos y bloqueos.

- Dependencias en procesadores encadenados: tipos.
- Riesgos y bloqueos.
- Ejemplo de implementación de un bloqueo: inyección de NOP.
- Cronograma de instrucciones.
- Impacto en el CPI.

2.6 Dependencias estructurales.

- Resolución de dependencias estructurales en el DLX.
- Otras situaciones de riesgo estructural.

2.7 Dependencias de datos.

- Tipos: RAW, WAR, WAW.
- Desvíos o cortocircuitos (*bypasses*).
- Implementación de desvíos.
- Dependencias de memoria.

2.8 Dependencias de control.

- Dependencias, riesgos y bloqueos de control.
- Implementación adelantada de la resolución de los saltos.
- Implementaciones especiales: saltos retrasados.

2.9 Predicción de saltos.

- Predicción estática de salto condicionales.
- Predicción dinámica de salto condicionales: BPB, BTB, *folding*.
- Máquinas de estado predictivas y porcentaje de acierto en casos reales.

2.10 Instrucciones de duración variable.

- Implementación.
- Nuevos riesgos y cuantificación de bloqueos en programas científicos.

TEMA 3 PLANIFICACIÓN (*SCHEDULING*) DE INSTRUCCIONES (T: 6H P: 3H).

3.1 Conceptos fundamentales.

- Grafo y tipos de dependencias.
- Bloques básicos.
- Frecuencia de las dependencias.

3.2 Planificación estática.

- Técnicas básicas de optimización del compilador. Ejemplos.
- Límite teórico de ejecución dataflow.

3.3 Planificación dinámica.

- Algoritmo del marcador (scoreboard).
- Estaciones de reserva.
- Renombrado dinámico de registros.
- Implementación del Algoritmo de Tomasulo.

3.4 Desenrollado de bucles.

- Bucles con paralelismo de datos.
- Desenrollado y entrelazado sistemáticos de iteraciones.
- Impacto en las prestaciones y reducción de instrucciones.
- Dificultades de aplicación en CISC.

3.5 Técnicas software avanzadas.

- Segmentación software.

- Planificación de trazas.

3.6 Ejemplos reales: planificación estática *vs.* dinámica.

- El problema de la compatibilidad de las aplicaciones.
- Casos reales: Evolución de algunas familias en cuanto a la planificación

Prácticas de Arquitectura de Sistemas Paralelos 1.

En este curso, cada alumno podrá escoger para hacer las prácticas entre dos métodos:

1. Clásico: prácticas guiadas a través de un boletín y con un profesor.

   En cada sesión de las prácticas clásicas se dará una explicación previa (aproximadamente 20 minutos) de los conceptos necesarios y de la metodología a seguir. La duración completa, incluyendo la explicación previa, elaboración del software y realización de pruebas es de alrededor de dos horas, aunque en función de la disponibilidad de laboratorios, las prácticas se podrán subdividir en varias sesiones.

   Se impartirán las siguientes prácticas clásicas a lo largo del cuatrimestre:

   Práctica 1. Análisis de cachés reales e impacto en las prestaciones.

   Práctica 2. Estudio de los bloqueos estructurales, de datos en el DLX.

   Práctica 3. Estudio de los bloqueos de control: predicción dinámica de saltos.

   Práctica 4. Planificación estática y dinámica: desenrollado y Algoritmo de Tomasulo.

2. ABP: prácticas usando la metodología llamada "Aprendizaje Basado en Problemas reales", en inglés PBL (*Problem-Based Learning*). Una explicación más detallada de este enfoque se describe en un documento que se dejará en copistería y en la web.

   Estas prácticas ABP se organizarán en problemas reales o "casos prácticos". Se impartirán los siguientes casos prácticos a lo largo del cuatrimestre:

   CP1. Análisis de cachés reales e impacto en las prestaciones.

   CP2. Estudio de los bloqueos estructurales, de datos y de control.

   CP3. Planificación estática y dinámica.

Sistema de Evaluación.

Como sistema de evaluación complementaria se realizarán dos pruebas bimestrales durante el cuatrimestre (siempre que el centro disponga de aulas libres para las mismas). La materia que entrará en cada prueba dependerá de la marcha del curso, pero será aproximadamente la mitad correspondiente de la asignatura. Cada una de ellas constará de una parte teórica y otra de problemas. Será necesario aprobar las partes teórica y de problemas por separado para poder superar cada prueba bimestral. Además, a la segunda prueba bimestral sólo se podrán presentar los alumnos que hubieran superado la primera. La nota final obtenida con las pruebas bimestrales será: 0.4*nota_primer_bimestre + 0.6*nota_segundo_bimestre siempre que nota_primer_bimestre ≥ 5.0. Las prácticas se valorarán como se indica abajo.

Como sistema de evaluación ordinario se realizará un examen final por convocatoria de la asignatura. Constará de una parte teórica, que tendrá un valor en torno al 30% de la nota del examen, y otra de problemas (aproximadamente el 70% de la puntuación final).

La correcta realización de las prácticas es condición indispensable para aprobar la asignatura. Para los alumnos que opten por las prácticas clásicas, la influencia máxima en la nota final de éstas será del 15%. Para los alumnos que opten por realizar las prácticas siguiendo la metodología ABP "Aprendizaje Basado en Problemas reales", (en inglés PBL), la influencia máxima de las prácticas en la nota final será de hasta el 33%.

BIBLIOGRAFIA BASICA.

[H&P96] J.L. Hennessy, Patterson "Computer Architecture. A Quantitative Approach". Morgan-Kaufmann (2 Edition),1996.

[ORTEGA] Julio Ortega Lopera; Mancia Anguita López; Alberto Prieto Espinosa. *Arquitectura De Computadores.* Edit. Paraninfo. ISBN: 8497322746. (2005).

[MUELL00] S. M. Mueller, W. J. Paul. *Computer Architecture: Complexity and Correctness.* Edit. Springer. 2000.

[H&P2000] Patterson, Hennessy. *Estructura y Diseño de Computadores. Interfaz Circuitería-Programación*. Reverté. 2000.

[TANE] A. S. Tanembaum. *Structured Computer Organization.* Prentice Hall. 1999. Quinta Edición.

BIBLIOGRAFIA COMPLEMENTARIA.

[H&P91] J.L. Hennessy, Patterson “Computer Architecture. A Quantitative Approach”. Morgan-Kaufmann (Edición primera).

[H&P03] J.L. Hennessy, Patterson “Computer Architecture. A Quantitative Approach”. Morgan-Kaufmann (Edición tercera).

[SHEN06] Shen, John Paul & Lipasti Mikko H. *Arquitectura De Computadores.* (Editorial McGraw-Hill) ISBN: 8448146425. 1ª edición (FEB-06).

[H&P95] D.A. Patterson, Hennessy. “Organización y Diseño de Computadores. La Interfaz Hard/Soft”. McGraw-Hill, 1995.

[STAL96] W. Stallings, “Computer Organization and Architecture. Designing for Performance”. Prentice Hall, 1996, 4º Ed.

[JAIN91] R. Jain, “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling”. John Wiley & Sons, 1991.

[GILA95] R. Giladi, N. Ahituv “SPEC as a Performance Evaluation Measure”. *IEEE Computer*, 1995 (pp. 33-42).

[KAIN96] R.Y. Kain, “Advanced Computer Architecture: A Systems Design Approach”. Prentice-Hall, 1996.

[KOGG81] P. M. Kogge “The Architecture of Pipelined Computers”. McGraw-Hill, 1981.

[LILJ91] D.J. Lilja, “Architectural Alternatives for Exploiting Paralelism”. IEEE Comp. Soc. Press, 1991.

[STON90] H.S. Stone, “High-Performance Computer Architecture”. Addison-Wesley (Second Edition) 1990.

[HWAN93] K. Hwang “Advanced Computer Architecture: Parallelism, Scalability, Programmability”. McGraw-Hill, 1993.