FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

PROGRAMA DE CURSO

Código	Nombre			
CC5320	Programación Consciente de la Arquitectura			
Nombre en Inglés				
Awareness of architecture in programming				
SCT	Unidades Docentes	Horas de Cátedra	Horas Docencia Auxiliar	Horas de Trabajo Personal
6	10	1,5	1.5	7
Requisitos			Carácter del Curso	
CC4301 Arquitectura de Computadores			Electivo	
Resultados de Aprendizaje				
Al término del curso el alumno será capaz de generar código u optimizar código existente, el cual se adaptará a las características particulares de la arquitectura sobre la cual se ejecutará, usando de manera eficiente los recursos disponibles.				

Metodología Docente	Evaluación General
El curso tiene una vertiente mayormente práctica, apoyada sobre una base teórica. La base teórica se desarrollará en clases expositivas y de discusión con los alumnos. Las clases prácticas de la asignatura se desarrollará en laboratorios, mediante la realización de un conjunto de tareas tuteladas. La realización de estas prácticas se hará íntegramente en las horas de laboratorio asignadas.	La evaluación se basa en 1 control y 5 o más tareas. Adicionalmente habrá un examen para los alumnos que deseen subir su nota final. Se sigue la ponderación que se plantea a continuación: NC= C1*0.35 NT=(T1+T2+T3+T4+T5)/5 NF=0.65*NT + max(NC,Examen) Examen es opcional.

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Unidades Temáticas

Número	Nombre de la Unidad	Duración en Semanas
1	Introducción a la Optimización de Código	1

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Ley de Amdhal y speedup • Métricas a considerar • Herramientas a utilizar • Sobre la correctitud de programas optimizados	Al término de esta unidad se espera que el alumno: • Pueda describir lo que significa "programar consciente de la arquitectura". • Puede distinguir en qué casos se deben y pueden aplicar optimizaciones de código. • Aprenda una metodología de trabajo para optimizar un programa. • Conozca las herramientas adecuadas para medir rendimiento de programas. • Puede distinguir que optimizaciones son ejecutadas por el compilador y cuáles no.	[1] Secciones: 5.1, 5.2, 5.14 y 5.15

Número	Nombre de la Unidad	Duración en Semanas
2	Herramientas de Medición del Rendimiento	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Generar y analizar código ejecutable. • Medición del tiempo • Medición de perfiles (profiling) • Medición de trazabilidad	Al término de esta unidad se espera que el alumno: • Aprenda las funcionalidades de las herramientas presentadas. • Sepa diferenciar las ventajas y desventajas de cada herramienta.	Páginas del manual para comandos time, strace, ltrace, gprof, gcc, icc, times(), getrusage(). Tutoriales de Oprofile y IA-32 Hardware Counter description.

Número	Nombre de la Unidad	Duración en Semanas
3	Reducción de operaciones de larga latencia	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Latencia y Throughput • Reducción de trabajo • Especialización de rutinas • Reducción de overhead por OS y librerías.	Al término de esta unidad se espera que el alumno: • Identifque los cuellos de botella en un código. • Identifique las opciones del compilador que pueden evitar/reducir operaciones de larga latencia. • Identificar optimizaciones que no pueden ser realizadas por el compilador. • Reprogramar partes del código usando la metodología propuesta.	[4] (Apendix C). [6.2] Ejemplos de código con manipulación de bits. [5] Capítulos 2 y 10 .

Número	Nombre de la Unidad	Duración en Semanas
4	Optimizaciones del flujo de control	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Instrucciones de salto y ejecución • Herramientas de análisis • Inlining • Code hoisting • Loop unrolling, collapsing, fusion • Orden de evaluación de condiciones. • Remoción de saltos con predicción difícil.	Al término de esta unidad se espera que el alumno: • Describa cómo el procesador maneja las instrucciones de salto. • Identifique las opciones del compilador que pueden optimizar el flujo de control. • Identificar optimizaciones que no pueden ser realizadas por el compilador. • Utilizar la herramienta Oprofile para obtener información de las instrucciones de salto.	[1] Secciones: 3.6, 5.4, 5.5, 5.7 y 5.8 Branch Paper introductorio sobre branch predictors: ”Combining Branch Predictors” de S. McFarling [4] Sección 3.4.1

Número	Nombre de la Unidad	Duración en Semanas
5	Optimizaciones Conscientes de la Memoria	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Jerarquía de memoria en sistemas • Instrucciones de acceso a memoria • Principio de localidad • Técnicas dependientes de la microarquitectura. • Herramientas de análisis • Aliasing • Alineamiento de datos • Ancho de banda de la memoria • Localidad temporal y espacial	Al término de esta unidad se espera que el alumno: • Describa los componentes de la microarquitectura involucrados en los accesos a memoria. • Enumerar las limitaciones de esos componentes. • Identificar opciones de compilación para optimizar accesos a memoria. • Identificar optimizaciones que no pueden ser realizadas por el compilador. • Utilizar la herramienta Oprofile para obtener información sobre el manejo de la memoria.	[1] Sección 6 [4]

Número	Nombre de la Unidad	Duración en Semanas
6	Vectorización	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Instrucciones vectoriales SIMD • Loops vectorizables • Arquitecturas con soporte vectorial MMX, SSE,VMX,AVX. • Opciones de compilación y autovectorización. • Programación con instrucciones intrínsecas SSE.	Al término de esta unidad se espera que el alumno: • Enumerar diferencias entre instrucciones escalares y vectoriales. • Identificar que partes del código pueden ser vectorizadas. • Saber si el compilador ha sido capaz de vectorizar código automá-ticamente. • Programar utilizando instrucciones vectoriales en lenguaje C mediante intrínsecas SSE2.	[4] Capítulos 4, 5 y 6 y Apendix C. Autovectoriza-tion status of GCC. http://gcc.gnu.org/projects/tree-ssa/vectoriza-tion.html.

Número	Nombre de la Unidad	Duración en Semanas
7	Paralelismo Usando Threads	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
• Arquitecturas multicore • Pthreads y OpenMP • Multiplicación de matrices usando multiples threads. • Diferencias en speedup usando código no-optimizado y optimizado.	Al término de esta unidad se espera que el alumno: • Use la librería Pthreads para aprovechar arquitecturas multi-core. • Sincronización de 2 o más threads: mutex, variables de condición. • Use los pragmas OpenMP. • Estudie los efectos de la optimización de código en arquitecturas multi-core.	Página de manual sobre pthreads [3]

Número	Nombre de la Unidad	Duración en Semanas
8	Auto-tuning de Aplicaciones	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Revisión bibliográfica de auto-tuning de aplicaciones y programación multi-core y many-core.	Al término de esta unidad se espera que el alumno: • Conozca temas avanzadas en optimización de código como el auto-tuning de aplicaciones y la optimización de código en arquitecturas multi-core y many-core.	Artículos relacionados

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Bibliografía

El curso es autocontenido y no requiere bibliografía complemetaria si se asiste a clase. Para profundizar más sobre las materias vistas se recomiendan los siguientes libros:

Básicos:

[1] Bryant, Randal and O'Hallaron David, **Computer Systems: A Programmer's Perspective**, Pearson International Edition, 2011/2.

[2] John L. Hennessy, David A. Patterson, **Computer architecture : a quantitative approach**, Elsevier, Morgan Kaufmann, 2007.

Complementarios:

[3] David E. Culler, Jaswinder Pal Singh, **Parallel computer architecture : a hardware/software approach**, Morgan Kaufmann Publishers, 1999.

[4] Manuales de optimización Intel.

[5] Henry S. Warren, **Hacker's Delight**, Addison-Wesley, 2012.

[6] Diversos sitios web con información útil sobre optimización de código y aplicaciones. Entre ellos, los más importantes son:

- [6.1] http://csapp.cs.cmu.edu/public/waside.html
- [6.2] http://graphics.stanford.edu/~seander/bithacks.html
- [6.3] http://www.hackersdelight.org/

Vigencia desde:	Marzo 2014
Elaborado por:	Oscar Peredo