

# Incorporating Multicore Programming in Bachelor of Science in Computer Engineering Program

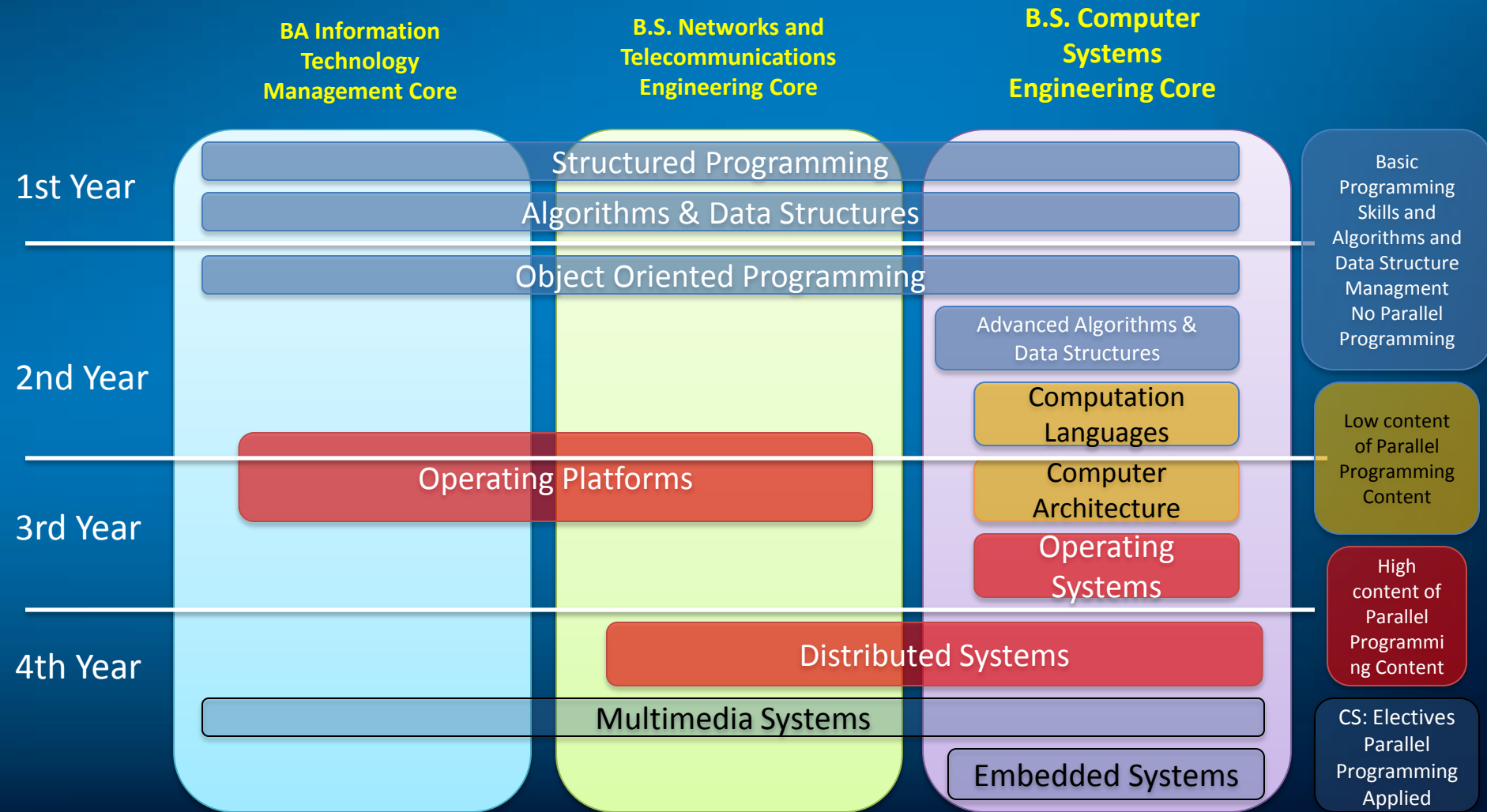
ITESO University  
Guadalajara, Jalisco  
México

# *Instituto Tecnológico y de Estudios Superiores de Occidente* Jesuit University in Guadalajara

## Population

- **Students: 8766**
  - 43.93 % Women
  - 56.07 % Men
- **From that population**
  - 74 Students in Networks and Communication engineering
  - 250 Students in Computer Systems engineering
  - 58 Students in major of Information Technology
  - 146 Students in Electronic engineering
  - 45 Students in Applied Informatics master degree
  - 90 Students in Electronic Design master degree
  - 7 Students in Integrated Circuits Design specialty
- **Professors: 312 staff + 990 course teachers**
  - Researchers (Staff): 77
  - SNI Researchers (Staff): 22
- **1 Campus in the Institute system:**

# Parallel Programming in ITESO – Bachelors Curriculum



# Operating Platforms

## Outline

- CPU Architecture
- Assembly Language
- Multicore Architecture
  - Hyper Threading and Multicore
- OS Processes
- Threads
  - Multicore Software Impact
  - Tasks & Threads (Win Threads)
  - Threads Synchronization
    - Data Races and Deadlocks
  - Using tools for performance & correctness
- Memory Management
- I/O Management

Knowledge\*

24 hrs

Apply\*

24 hrs

40 hrs/128 hrs = 37.5%

Total 128 hrs

\*We are referring Bloom Taxonomy of action verbs  
•<http://www.clemson.edu/assessment/assessmentpractices/referencematerials/documents/Blooms%20Taxonomy%20Action%20Verbs.pdf>

# Operating Systems

- Outline

- Computer System Structure
- Operating System Structure

- System Processes

- Threads

- ULT & KLT
- Pthreads & cloning processes
- Threading for improving performance on Multicore Architectures

*Apply\**

*24 hrs*

- create their own ULT Library

- CPU Scheduling

- Scheduling for SMP/Multicore Architectures

*Knowledge\**

*8 hrs*

- Synchronization & Concurrency

- Race Conditions, Deadlocks
- IPC Mechanisms

*Apply\**

*32 hrs*

- Memory Management

- I/O Management

- File Systems

*Total 128 hrs*

*64 hrs/128 = 50%*

\*We are referring Bloom Taxonomy of action verbs

•[http://www.clemson.edu/assessment/assessmentpractices/reference materials/documents/Blooms%20Taxonomy%20Action%20Verbs.pdf](http://www.clemson.edu/assessment/assessmentpractices/reference%20materials/documents/Blooms%20Taxonomy%20Action%20Verbs.pdf)

# Distributed Systems

- Recently changed to: Parallel Programming & Distributed Systems

- Outline

- Parallel Programming Concepts
- Tools for Parallel Programming
  - Compilers, Parallel Studio
- Windows Threads
- Parallel Programming Models & Design
- OpenMP
- MPI (Clusters Programming)
- Distributed Systems Introduction
- Sockets Programming
  - TCP/UDP
- Client/Server & P2P Applications
- Distributed Systems Synchronization Algorithms
- Atomic Transactions
- RPCs
- RMI's
- Corba
- Web Services

*Total 128 hrs*

*Apply\**

*64 hrs*

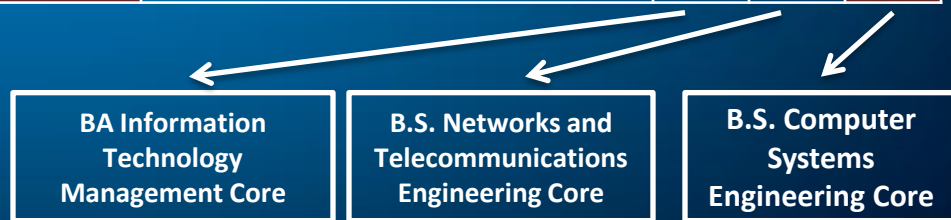
*64 hrs/128 = 50%*

\*We are referring Bloom Taxonomy of action verbs

•<http://www.clemson.edu/assessment/assessmentpractices/referencematerials/documents/Blooms%20Taxonomy%20Action%20Verbs.pdf>

# Learning Outcomes (1/2)

Operating Platforms	Operating Systems	Distributed Systems	Bachelor Programs		
Describe HT & MC like logical & physical CPUs			BA IT	BS N&T	
Understand the benefit of MC in multiprogramming environments	Understand the benefit of MC in multiprogramming environments		BA IT	BS N&T	BS CSE
Measure the difference of HT & MC	Measure the difference of HT & MC		BA IT	Bcs N&T	BS CSE
	Describe how threads are implemented in an OS				BS CSE
	OS Scheduling for SMP & Multicore				BS CSE
Problem decomposition in tasks		Problem decomposition in tasks	BA IT	BS N&T	BS CSE
	User Level Threads & Kernel Level Threads				BS CSE



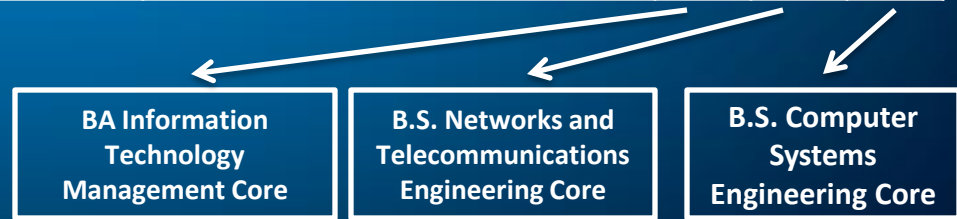
*Knowledge\**

*Apply\**

\*We are referring Bloom Taxonomy of action verbs  
 •<http://www.clemson.edu/assessment/assessmentpractices/reference materials/documents/Blooms%20Taxonomy%20Action%20Verbs.pdf>

# Learning Outcomes (2/2)

Operating Platforms	Operating Systems	Distributed Systems	Bachelor Programs		
	Threading with OS Threads (Clone)				
Threading with OS Threads (Windows)	Threading with Posix Library (Pthreads)	Threading with OS Threads (Windows)			
Race conditions and Deadlocks	Race conditions and Deadlocks	Race conditions and Deadlocks	BA IT	Bcs N&T	BS CSE
	How IPCs are implemented				BS CSE
Using tools for performance and correctness		Using tools for performance and correctness	BA IT	BS N&T	BS CSE
	Use of IPCs for process & threads synchronization				BS CSE
		Parallel Applications in Shared Memory vs. Non Shared memory		BS N&T	BS CSE



**Knowledge\***

**Apply\***

\*We are referring Bloom Taxonomy of action verbs  
[http://www.clemson.edu/assessment/assessmentpractices/reference materials/documents/Blooms%20Taxonomy%20Action%20Verbs.pdf](http://www.clemson.edu/assessment/assessmentpractices/reference%20materials/documents/Blooms%20Taxonomy%20Action%20Verbs.pdf)



# Notes developed included in our courses (In Spanish) (1/2)

**Introduction**  
Iniciando con Programación Paralela  
para Procesadores Multicore

**When and Where Parallel Programming**

**Basic Parallel Programming Concepts**  
Win Threads/Race Conditions/Synchronization/Deadlocks

José Luis Elvira Valenzuela

**Using Parallel Studio To Analyze Applications**

## Tabla de contenido

1	INTRODUCCIÓN.....	1
1.1	Temáticas de los módulos de los cursos de los procesadores	2
1.2	¿Por qué programar en paralelo?	3
1.3	¿Es posible en todo el mundo?	4
1.4	Modelos de programación para los procesadores multicore: Paralelo Simple	6
1.4.1	Completado para Parallel Studio	8
1.5	Resumen del capítulo	14
2	¿CUÁNDO Y DÓNDE PROGRAMAR EN PARALELO?	17
2.1	Concepto intuitivo de ES	17
2.2	Encontrando las secciones de código intuitivo con Intel Parallel Analyzer	20
2.3	Resumen del capítulo	24
3	CONCEPTOS BÁSICOS DE PROGRAMACIÓN PARALELA.....	27
3.1	Procesos e hilos en el sistema operativo	27
3.2	Hilos a nivel usuario	30
3.3	Hilos en Windows®	33
3.3.1	Creación de hilos en Windows®	34
3.3.2	Esperando la terminación de un hilo en Windows®	35
3.3.3	Esperando la terminación de muchos hilos	36
3.3.4	Esperando por hilos a los hilos	37
3.4	Condiciones de carrera y sincronización	40
3.4.1	Soluciones por software	44
3.4.2	Esperando	46
3.4.3	Secciones críticas	48
3.4.4	Mutex	49
3.4.5	Semáforos	61
3.4.6	Eventos	63
3.5	Cuidados con la sincronización	67
3.5.1	Exclusión mutua	68
3.5.2	Interbloqueo	72
3.6	Resumen del capítulo	76
4	APROFUNDOS DE PARALLEL STUDIO PARA PROGRAMAR EN PARALELO.....	77
4.1	Encontrando errores en un programa con hilos con Parallel Inspector	78
4.1.1	Encontrando una violación de orden	79
4.1.2	Encontrando interbloqueo	82
4.2	Encontrando los errores de sincronización con Parallel Analyzer	84
4.2.1	Condición de carrera	84
4.3	Caso práctico: Calculando el producto punto	87
4.3.1	Identificando los problemas de código incorrecto	89
4.3.2	Paralelizando el programa en hilos	90
4.3.3	Buscando los errores en la paralelización	97
4.3.4	Corrigiendo los errores de la paralelización	100
4.3.5	Revisando las direcciones	101
4.3.6	Aislamiento de eventos	102
4.3.7	Resolviendo los problemas de rendimiento	103
4.3.8	Aislamiento de la solución final	106
4.4	Resumen del capítulo	108

Tabla de Contenidos

iii

# Notes developed included in our courses (In Spanish) (2/2)

5	DISEÑO DE UN PROGRAMA EN PARALELO	109
5.1	Descripción de tareas	110
5.1.1	Paralelismo en los datos	111
5.1.2	Paralelismo en las tareas	116
5.2	Criterios para elegir entre las dos opciones con tareas	120
5.2.1	Granularidad	121
5.2.2	Balanceo de carga	122
5.2.3	Dependencias	131
5.2.4	¿Cómo administrar las dependencias?	134
5.3	Planificación de las tareas	137
5.3.1	Planificación estática	137
5.3.2	Planificación dinámica	138
5.4	Evaluando el diseño a través de Parallel Studio	142
5.4.1	Balanceo de carga	143
5.5	Resumen del capítulo	147
6	HILOS IMPLÍCITOS: OPENMP	149
6.1	Estableciendo el ambiente para OpenMP	150
6.2	Regiones Paralelas	154
6.3	Servicios para el trabajo compartido	156
6.3.1	Cierras	157
6.3.2	Secciones	163
6.3.3	Tareas	166
6.4	Accediendo a los datos	168
6.4.1	Cierrado privado	169
6.4.2	Cierrado compartido	170
6.4.3	Cierrado false compartido	171
6.4.4	Cierrado false compartido	171
6.4.5	Cierrado threadprivate	172
6.4.6	Sección crítica	172
6.4.7	Operaciones atómicas	173
6.4.8	Reducción	174
6.5	Otras directivas para el sincronismo con estructura	176
6.5.1	Directiva nowait	177
6.5.2	Directiva barrier	177
6.5.3	Directiva master	178
6.5.4	Directiva single	179
6.6	Funciones de la API de OpenMP	181
6.7	Resumen del capítulo	181
7	LIBRERÍAS DE ALTO NIVEL: INTEL THREADING BUILDING BLOCKS <sup>2</sup>	183
7.1	Introducción	183
7.1.1	¿Qué es Intel Threading Building Blocks?	183
7.1.2	Para usar la librería Intel TBB	184
7.2	Algoritmos paralelos	185
7.2.1	Cierrado paralelos	185
7.2.2	Cierrado paralelos anidados	187
7.2.3	Ordenamiento paralelo de un vector	188
7.3	Planificación y flujo de tareas	190
7.4	Clases derivadas de la librería para el uso conveniente de estructuras de datos	191
7.4.1	Cola concurrente	191
7.4.2	Vector concurrente	192
7.4.3	Tabla Hash concurrente	192
7.5	Funciones atómicas	193

iv Iniciando con Programación Paralela para Procesadores Múltiples

7.6	Bugs	193
7.7	Resumen del capítulo	193

**Designing a Parallel Application**  
Tasks Decomposition/Tasks Scheduling/Performance Evaluation

**Implicit Threads OpenMP**

**Intel Threading Building Blocks**  
(Actually in Development)

v Tabla de Contenidos



# Intel Challenge

- Multicore programming contest called “ITESO-Intel Challenge”.
  - Intel as a sponsor of the competition
  - Since Autumn 2006, Intel Challenge has been realized 6 times at ITESO
  - Within ACM programming contest
  - In the challenge, participants has to optimize a CPU intensive program to run in Multicore architecture.
- 
- Open to undergraduate students of all universities in Guadalajara and Mexico.
  - Las edition also included a new version of this contest aimed at high school student with Two-folded intention:
    - Introduce high school students to Multicore programming
    - Develop an interest to become IT professionals.
- 
- The contest aims to disseminate:
    - What Multicore means
    - Why parallel programming to achieve maximum performance
    - Software industry challenges
    - How to thread an application
    - Using tools for correctness and maximize performance



# Faculty Training

- ITESO has collaborated with Intel Education in Faculty Training for to participate teaching Multi-Core programming to universities in México and Latin America.
  - Created the Spanish version of the Multi-core Programming for Academia Module
  - Since 2007 course has been replicated 10 times to professors of more than 50 universities
    - <http://software.intel.com/en-us/articles/iac-recognition>



# Professional Applications Projects

- ITESO has defined a program called Professional Application Projects (PAP)
- Students involved in a real-world industrial project
  - Over a period ranging from eight to twelve months.
  - Two consecutive school terms in order to obtain their degree.
- Since 2009, there have been PAPs related to Multicore Programming
- Involving collaboration of industrial partners
- Participated in the development of a traffic simulator
  - Particularly optimizing code to perform calculations using Multicore architectures.