1.	Record Nr.	UNINA9910437584803321
	Autore	Raynal Michel
	Titolo	Concurrent Programming: Algorithms, Principles, and Foundations / / by Michel Raynal
	Pubbl/distr/stampa	Berlin, Heidelberg : , : Springer Berlin Heidelberg : , : Imprint : Springer, , 2013
	ISBN	3-642-32027-9
	Edizione	[1st ed. 2013.]
	Descrizione fisica	1 online resource (528 p.)
	Disciplina	005.275
	Soggetti	Computers Computer organization Software engineering Computer hardware Theory of Computation Computer Systems Organization and Communication Networks Software Engineering/Programming and Operating Systems Computer Hardware
	Lingua di pubblicazione	Inglese
	Formato	Materiale a stampa
	Livello bibliografico	Monografia
	Note generali	Description based upon print version of record.
	Nota di bibliografia	Includes bibliographical references and index.
	Nota di contenuto	Part I - Lock-Based Synchronization -- Chap. 1 - The Mutual Exclusion Problem -- Chap. 2 - Solving Mutual Exclusion -- Chap. 3 - Lock-Based Concurrent Objects -- Part II - On the Foundations Side: The Atomicity Concept -- Chap. 4 - Atomicity: Formal Definition and Properties -- Part III - Mutex-Free Synchronization -- Chap. 5 - Mutex-Free Concurrent Objects -- Chap. 6 - Hybrid Concurrent Objects -- Chap. 7 - Wait-Free Objects from Read/Write Registers Only -- Chap. 8 - Snapshot Objects from Read/Write Registers Only -- Chap. 9 - Renaming Objects from Read/Write Registers Only -- Part IV - The Transactional Memory Approach -- Chap. 10 - Transactional Memory -- Part V - On the Foundations Side: From Safe Bits to Atomic Registers -- Chap. 11 - Safe, Regular and Atomic Read/Write Registers -- Chap. 12 - From Safe Bits to Atomic Bits: A Lower Bound and an Optimal Construction -- Chap. 13 - Bounded Constructions of Atomic b-Valued Registers -- Part VI - On the Foundations Side: The

Sommario/riassunto

The advent of new architectures and computing platforms means that synchronization and concurrent computing are among the most important topics in computing science. Concurrent programs are made up of cooperating entities -- processors, processes, agents, peers, sensors -- and synchronization is the set of concepts, rules and mechanisms that allow them to coordinate their local computations in order to realize a common task. This book is devoted to the most difficult part of concurrent programming, namely synchronization concepts, techniques and principles when the cooperating entities are asynchronous, communicate through a shared memory, and may experience failures. Synchronization is no longer a set of tricks but, due to research results in recent decades, it relies today on sane scientific foundations as explained in this book. In this book the author explains synchronization and the implementation of concurrent objects, presenting in a uniform and comprehensive way the major theoretical and practical results of the past 30 years. Among the key features of the book are a new look at lock-based synchronization (mutual exclusion, semaphores, monitors, path expressions); an introduction to the atomicity consistency criterion and its properties and a specific chapter on transactional memory; an introduction to mutex-freedom and associated progress conditions such as obstruction-freedom and wait-freedom; a presentation of Lamport's hierarchy of safe, regular and atomic registers and associated wait-free constructions; a description of numerous wait-free constructions of concurrent objects (queues, stacks, weak counters, snapshot objects, renaming objects, etc.); a presentation of the computability power of concurrent objects including the notions of universal construction, consensus number and the associated Herlihy's hierarchy; and a survey of failure detector-based constructions of consensus objects. The book is suitable for advanced undergraduate students and graduate students in computer science or computer engineering, graduate students in mathematics interested in the foundations of process synchronization, and practitioners and engineers who need to produce correct concurrent software. The reader should have a basic knowledge of algorithms and operating systems.