concerns, *Lambda Calculi* is a very good book and is a timely addition to the theoretical computer scientist's bookshelf.

MIKE JOY
Warwick University

AMOS R. OMONDI
Computer Arithmetic Systems. Prentice Hall International. 1994. ISBN 0 13 334301 4 £22.95. 520pp. softbound.

Computer systems are built from various sub-modules and peripheral devices. One of the vital elements is the computer arithmetic module. This excellent book deals with the various arithmetic operations, different associated algorithms and their detailed analysis. Cost-performance comparison of different algorithmic models have also been shown.

Case studies of different actually implemented units have been discussed in each chapter. Annotated bibliographic notes provide the readers with material for further studies. And the chosen set of exercises makes sure that the fundamentals do get absorbed.

This book consists of 8 chapters. There is a functional division of the contents into three parts.

Chapters 1 to 4 cover the basic algorithms and associated hardware details for fixed point number systems. One point may be noteworthy: coverage of topic in the Chapter 1 does touch upon the generalized number systems, including an introduction to floating point numbers and a more general title for this chapter might have been more apt.

Chapters 5 to 7 cover the floating point number concepts, associated operations and the hardware details.

Originality, clarity of expression and depth of knowledge stand out very clearly in this book. The case-studies giving the practical implementation details of the theoretical algorithms and particularly some different unusual implementations (for very fast operations in both fixed and floating point number systems) place the book at two levels simultaneously : it can be treated as a text book as well as excellent reference material for further research in the respective fields.

Chapter 8 deals with unconventional number systems and associated arithmetic, namely—

a. the residue number system with its unique carry-lessfast addition, subtraction, multiplication;
b. the decimal number system and arithmetic; and
c. redundant signed-digit number systems and arithmetic.

An extensive bibliography has been compiled at the end.

The appendices on pipelining with respect to high performance computers, shifter design and the separate copy of the design library are well appreciated. But I strongly feel that the appendices and design library should be amplified and recompiled into a separate volume which emphasises real-world designs and acts as a perfect complement to the book being reviewed.

I enthusiastically recommend this book as an useful addition to any academic or highly design-oriented scientific library. Moreover, this book will also be extremely suitable as a textbook for many universities.

S. SANYAL
Tata Institute of Fundamental Research, Bombay

JENS PALSBERG and MICHEAL I. SCHWARTZBACH
Object-Oriented Type Systems. John Wiley. 1994. ISBN 0 471 94128 X £24.95. 180pp. hardbound.

Object-oriented languages have often seemed removed from the mainstream. Although we can all accept simple notions of 'class' and 'object with methods' belonging to it as covered in many introductory courses, the necessary various (and varied) extensions to this concept to construct a generally acceptable language appear to lack a comprehensive conceptual model. Partly this is due to the variation in terminology between (say) C++ and Smalltalk exacerbated by discussions as to what being 'object-oriented' really means, and to the rôle (or otherwise) of types and by the 'our language does it this way' effect. It is also due to very real theoretical problems which mean that the progression from untyped Smalltalk to typed languages like Eiffel is sufficiently complicated to result in the latter containing type insecurities. Indeed the whole situation has echoes of the Lisp vs. Algol debate (recall Pascal's early type insecurities concerning function types). Indeed, languages like ML have become popular to a large extent because they offer the type security of Pascal with much of the flexibility of their weakly typed counterparts.

Palsberg and Schwartzbach centre their book around their BOPL (Basic Object Programming Language) and extensions thereto adding types, inheritance and genericity. BOPL is easy to understand and has a simple evaluation mechanism (although I disapprove of the authors following the trend to present such a machine only informally in English). Moreover, extensions for inheritance and genericity are defined by source-to-source transformation into the original BOPL. This has the great advantages of simplicity and of exposing otherwise non-obvious semantic choices which are normally glossed over by language designers who have dismissed (or even overlooked) alternatives. In particular, the interaction between inheritance and recursion is well exposed.