Department of Computer Science

Handbook
1980-81

University of Edinburgh

4.4.3 Mini-projects

An Operating System Implementation

The laboratory equipment for this mini-project consists of a pair of linked Interdata 74 mini-computers. One of these Interdatas serves as an operating system host, the other as an I/O server.

The I/O server is pre-programmed to provide a small high level set of I/O functions to the host. These functions provide access to a file containing job descriptions, access to files containing user jobs, and a means of mechanizing user I/O.

The major design aim of the operating system is to run correctly, and in as short an elapsed time as posible a set of jobs provided, and described in the job description file, by the I/O server.

Linear and Integer Programming

This individual project consists of a number of exercises drawn from the course unit of the same name.

Commercial and Industrial Data Processing

Each group undertaking this project carries out a qualitative and quantitative analysis of the information needs of a particular large organization, and produces a detailed report and a proposal for an appropriate information system.

Organizations covered in previous years include a whisky distiller, the administration of the Church of Scotland, the administration of the University of Edinburgh and the library of the University of Edinburgh.

Computer Graphics

The laboratory equipment for this project consists of the VAX 11/780 and the Department's high resolution (512x512 pixels) colour raster display.

The aim is to design and implement a software system which exploits the capabilities of a colour raster display to present significant amounts of data in an easily comprehensible form. These capabilities include an ability to display filled in areas in one of sixteen colour shades selected from a set of four thousand, and an ability to alter the image rapidly enough, within certain constraints, to allow a posibility of animation.

A Database Management System Implementation

This project is undertaken in groups of three, with one student designing and implementing the physical resresentation, one the programming language interface and manipulation algorithms, and the third the user interface. Implementation is on the VAX 11/780.

A representative and workable sample of the crucial components of a DBMS is designed and implemented. The data model chosen is a simple and flexible one, based on the work of W. Kent.

Micro-processor System Implementation

This project is undertaken in groups of two, with implementation on the micro-processor breadboarding systems.

The aim is to design and implement the functional components of an automotive instrument display system.

VLSI Circuit Design

This project has as its goal, the design of a digital subsystem using the VLSI design techniques introduced in the fourth year course of that name.

Students are encouraged to implement functions of their own invention, but may also chose a function from a given selection. Examples of possible functions are an associative memory, part of an ALU data path and a simple encoder.

CAD System Design

This project is undertaken individually, although the free exchange of ideas, and information gleaned from the literature, is strongly encouraged.

It involves the construction of a design rule checking program for IC designs expressed in CIF 2.0. The problems of design rule checking are introduced and discussed during tutorial sessions.

4.4.4 Honours Thesis

Each student is required to submit an honours thesis based on a significant research or implementation excercise, chosen from a set of topics drawn up by members of the Department. While not requiring the original research expected of a postgraduate thesis, the topics are designed to provide an opportunity for the display of initiative and persistence.

4.4.5 Assessment

There are two 3-hour written examinations in the June degree examination, which together contribute one third of the total marks, one third coming from the student's indivdual project work and one third from course work associated with the options.

First, Second or Third Class Honours may be awarded. For Honours in Computer Science, the results of the Computer Science 3 (Honours) and Computer Science 4 examinations are given equal weight. For joint Honours degrees, the results of Computer Science 3 (Honours) and the corresponding examination in the second Department concerned are given equal weight.

SECTION 5 POST-GRADUATE MASTER OF SCIENCE COURSE

Course Co-ordinator: L.D.Smith

5.1 Introduction

During the 1978/79 session a "modular" MSc scheme was introduced in the Faculty of Science at Edinburgh University. Currently, three departments collaborate in the scheme, the aim of which is to increase the cost-effectiveness of the postgraduate teaching by reducing the duplication of effort which arises when standard lecture material spans traditional subject boundaries.

Under the modular scheme a student is registered in a department in the usual manner and receives most of his or her teaching from that department. However, up to half of the lecture modules taken by a student may be taught by other departments in the scheme. This not only saves duplication of effort, but has the additional advantage of exposing students to a variety of points of view about current problems and issues.

All MSc courses in the modular scheme share a common format consisting of eight taught modules taken in the first half of the academic year followed by a project of six months duration, the results of which are presented as a dissertation. The courses are thus very practical in nature. All three courses are approved by the Science Research Council (SRC) for the purpose of tenure of its Advanced Course Studentships and, in addition, SRC has provided financial assistance to the University to increase support for two of them, the MSc in Computer Systems Engineering (described in section 5.2) and the MSc in Design and Manufacture of Microelectronic Systems (details of may be obtained from the Department of Electrical Engineering), as part of a special initiative designed to help the UK gain the maximum benefit from the availability of advanced microelectronic technology.

5.2 MSc in Computer Systems Engineering

The underlying theme of this course is the design and construction of computer systems. Lecture modules cover a broad spectrum of topics from the construction of systems directly using very large scale integrated circuit technology to the implementation of large software systems in which the underlying computer or microprocessor is a small component.

During the first six months of the course students attend eight lecture modules chosen from about twice that number. Concurrent with the lecture modules, students are expected to attend a series of seminars, the aim of which is to give a broad view of computer systems engineering. A proportion of these seminars are led by representatives from the computer and microelectronics industries.

In the final six months of the course students undertake an individual project, typically involving the design and construction of a relatively large software or hardware system. Past projects have included a high level language compiler for an INTEL 8086 microprocessor and a microprocessor based intelligent graphics display driver. Projects are assessed by means of a dissertation which must be submitted by the end of September.

Performance Evaluation

A survey of the tools and methods of performance evaluation; The problems of systems selection and performance prediction; Tuning a working system; Program behaviour.

Computer Aided Design of Digital Systems

Computer-based aids to the design and construction of digital systems are studied and related to several topics covered in the third year (Control of Data Flow, Analysis of Algorithms, Design of Data Bases and Compiler Construction).
Representation of design objects. Physical construction (assignment, wire-wrap schedules, printed circuit board tracking, etc).
Production of Documentation.
Design Verification.

Very Large Scale Integrated Circuit Design

VLSI design represents a new and exciting field of Computer Science. The treatment provides sufficient basic information about integrated devices, circuits, digital subsystems and system architecture to enable a participant to span the range of abstractions from the underlying physics to complete VLSI computer systems. Emphasis is laid on the need for computer aids and the importance of modularity of design.

Micro-processor Technology

The internal architecture of typical micro-processor devices.
The internal architecture of related LSI devices.
Internal interfaces within micro-computer systems.
External interfaces to micro-computer systems.
Micro-computer applications.

Data Base Systems

Data Models.
The relationship to programming languages.
DBMS architectures.
Implementation issues.

Linear and Integer Programming

The solution of Linear Programming problems by the Simplex Method constitutes the biggest commercial application of optimization techniques. Similarly, the Branch and Bound Method is the most widely used procedure for solving Integer Programming problems. A number of aspects of these two methods are studied:
The Simplex Method and a simple scheme for the solution of small problems by hand; Accuracy considerations and sparse problems; The revised Simplex Method; Branch and Bound Methods; Practical considerations affecting efficiency in Branch and Bound Methods.

A Calculus for Communicating Systems

The calculus is built on the notion of synchronized (or "handshake") communication. This notion covers not only the communication between independent modules of a system, but also the act of observing, or communicating with a system from outside.

The formalism of the calculus, explained through simple examples.
Applications of the calculus, including data flow, parallel numerical algorithms, descriptions of machines and data structures, and semantic description of high level programming languages.
A brief survey of, and comparison with, other models of concurrent computation (eg . Petri's Net Theory, Hoare's Communicating Sequential Processes).
The theory of "observation equivalence" of systems (that is, systems which look the same to a user or to another part of a larger system).

Algorithmic Graph Theory

Many real life problems to be tackled by computer scientists have natural graphical formulations; for example, the problems of routing, scheduling and the design of reliable computer networks. The aim of this unit is to introduce some of the algorithms which efficiently solve graphical problems, and to give some feel for the gulf which seems to exist between tractable and intractable (though often seemingly innocuous) problems.

Definitions and terminology, examples of the formulation of practical problems in graphical terms, illustration that finite does not imply computationally tractable, data structures for graphs, measuring the efficiency of algorithms.
Algorithms concerned with paths and communication: The shortest path in a graph, finding non-separable components, finding strong components.
Network flow: Maximum flow - minimum cut theorem, an algorithm for minimal s-t flow.
Optimisation: Minimal spanning tree, maximal matching in a bipartite graph, the Huffman algorithm.
Scheduling : The critical path method, minimum number of processors required to ensure no delay in completing a task.
Dealing with "hard" problems: Examples of NP-complete problems, methods of obtaining approximate solutions.

As a result of the Science Research Council's recent initiative in microelectronics, future projects will include the design and fabrication of very large scale integrated systems using silicon-gate nMOS technology.

5.3 Course modules

The following subsections comprise brief descriptions of the lecture modules available under the MSc in Computer Systems Engineering and the MSc in Design and Manufacture of Microelectronic Systems. Section 5.3.1 decribes modules available to students registered for either MSc degree, while section 5.3.2 describes the modules specific to the MSc in Computer Systems Engineering.

Certain modules, whose titles are marked with an asterisk, are also available to outside interested parties on payment of the appropriate fee.

5.3.1 Modules open to students from both MSc courses

Computer organisation and high-level assembly programming *

This module, taught in the Department of Computer Science, examines how to write programs in a high-level assembly language which manipulate the machine-specific features of typical mini- and micro-computers. The influence of different computer organisations on the writing of these programs is also considered and comparisons are made between a number of well known mini- and micro-computers. Practical work includes the writing of some of the programs described above and is performed on a user-friendly minicomputer-based program development system.

Programming techniques and software tools *

This module, taught in the Department of Computer Science, examines the practicalities of structured programming in a high-level language. Emphasis is placed on language independent concepts and on choosing the right language for the job. The design of data structures and the analysis of algorithms are also considered. Practical work includes the implementation of illustrative algorithms and data structures and is performed on the Department's own multi-access computer system.

Computer aids to the design and construction of digital systems *

This module, taught in the Department of Computer Science, examines how to implement effective computer aids to the design of large scale digital integrated systems. The problems of computational complexity, data management, distribution of function and interface to the designer are emphasised. Practical work comprises a design study of a CAD system to meet some specified objectives.

Computer systems and VLSI technology (I and II) *

These modules, taught in the Department of Computer Science, examine the structured design methodology advocated by Mead and Conway [M&C80] and its application to the implementation of computer systems. The impact of very large scale integration on the architecture of computer systems, the effectiveness of special-purpose design aids, and the role of computer aids in the management of design complexity are emphasised. Practical work involves the design of a digital system such as a simple ALU and is performed using a variety of computer aids to structured design and the Department's own design stations which are based on special-purpose colour graphics terminals.

LSIC design

This module, taught in the Department of Electrical Engineering, examines the fundamentals of integrated circuit design and layout. Practical work involves use of the SPICE circuit simulation package and use of the GAELIC computer aided IC layout system to prepare designs for fabrication.

Digital systems design with microprocessors *

This module, taught in the Department of Electrical Engineering, examines how to use microprocessors to implement digital systems. The design of the interface hardware and systems software required to make a microprocessor useful are emphasised. Practical work includes the interfacing and handling of simple peripheral devices such as keyboards and lights and is performed on the Department's Motorola 6800 microprocessor development systems.

Silicon processing

This module, taught in the Department of Electrical Engineering, examines all aspects of the fabrication of integrated circuits, from the production of single crystal silicon through the techniques of photolithography, repitoxy, oxidation, diffusion, ion implantation, etc, to the evaluation of finished devices.

Hybrid circuits - technology and design *

This module, taught in the Department of Electrical Engineering, examines techniques for the design and manufacture of thick and thin film hybrid circuits and their application in analogue systems. The interaction between technology, circuit design and system performance is emphasised, particularly with respect to the trade-offs between cost, complexity, yield, accuracy and stability.

4.4 Computer Science 4 (4th year)

Course Co-ordinator: N. H. Shelness

4.4.1 Aim of Course

The final year honours course is designed to provide a student with an opportunity to:

1. Study subjects that were outwith the core curriculum studied in the third year.
2. Study subjects to a greater depth than they were studied in the third year.
3. Undertake a number of mini-projects which give first hand experience of implementation and/or problem solving in a particular subject area.
4. Produce an honours thesis based on a major implementation or research project.

Most fourth year course units last for four weeks each at a rate of three meetings a week, and commence in the first and sixth weeks of terms one and two. The VLSI unit lasts for ten weeks at the rate of one meeting a week, commencing in week one of term one.

4.4.2 Course Units

Local Area Communications

The advent of inexpensive, high bandwidth methods of interconnecting computing systems over distances of up to one kilometer is making possible both new applications of computing systems such as the electronic office, and altering the basic assumptions under which computing systems are built. This unit studies the technology and application of methods of local communication.

Computational Complexity

Time and space as resources for Turing machines, nondeterminism.
Complexity classes, padding arguments and time and space hierarchies.
Polynomial reducibilities and Cook's theorem.
Problems complete in PSPACE and #P.
Provably intractable problems, finite and infinite.
Time/space tradeoffs, pebbling and Savitch's theorem.
Axiomatic complexity theory.

Computer Graphics

Device-independent graphics.
Interactive graphics.
Three-dimensional graphical techniques.
Raster-scan graphics.

4.3.3 Composite Practicals

The aim of these practicals is to introduce students to the methods of coping with the problems that arise with the design and implementation of large scale computer systems.

It is intended that by following the series of composite practicals students will gain experience in deciding how to:

design system
discover and use relevant information
choose the appropriate means of implementation
schedule their work load
present their findings and implementations in a clear and concise way.

The practicals are:

Term 1

1. Compiler Construction
2. Communications Protocol Implementation

Term 2

3. Design and construction of a sequencer
4. Processor Design Study

Term 3

6. Computer Architecture Essay

4.3.4 Assessment

The degree examination for the Honours ct rse is held in June, and for the Ordinary degree in June with the resit in September.

(a) Honours. There are three 3-hour written examinations, which comprise two thirds of the total marks, the remaining one third coming from coursework. For Honours, a student must pass this examination in June, at the first attempt.

(b) Ordinary. An Ordinary pass is awarded for satisfactory performance on any two of the three papers and corresponding course work.

A class examination (one 3hr paper) is set at the end of the first term. Performance in this exam is taken into account when awarding merit certificates.

Project Planning and Management

This module, taught in the Department of Astronomy, examines how to plan and manage projects. The critical path analysis of complex projects is considered and the motivation of managers and workforce is emphasised.

5.3.2 Modules available only to MSc students in Computer Science

Fundamentals of programming

This module examines recent advances in methods of designing programs with particular emphasis on the use of abstract data types and applicative programming languages.

Compiler design

This module examines the design of compilers for typical modern high level programming languages such as PASCAL with particular emphasis on the practical aspects of parsing, code generation and code optimisation.

Introduction to operating systems

This module examines the structure of typical modern multi-task multi-user operating systems such as the Edinburgh Multi-Access System. Practical work is performed on a minicomputer development system and allows students to test some of the ideas presented in the lectures.

Digital communications and computer networks

This module examines the fundamentals of digital communications and their application to computer networks. Practical work typically involves the development of a protocol handler for a communications standard such as X25 (level-2) and is performed on the Department's own multi-access computer system.

Fundamentals of digital structures

This module examines some techniques for describing, analysing and implementing register transfer level digital systems. Fundamental limitations, the analysis of determinacy and the analysis of deadlock are considered in the context of both asynchronous modular systems and microprogrammed systems.

Advanced graphics

This module examines a number of advanced topics in computer graphics such as 3D-graphics, raster-scan graphics, device independent graphics (for example, the ACM CORE standard) and interactive graphics. Practical work is performed on the Department's colour raster-scan graphics terminals and typically involves animation or interactive games.

Design of databases

This module examines recent advances in the design of databases and in the technology for implementing them. Examples are drawn from diverse application areas such as horse-racing administration and computer aided aircraft design.

5.3.3 Introductory programming course

An introductory programming course is run in the week prior to the start of the academic year. All students intending to take modules taught in the Department of Computer Science are strongly recommended to attend this course, regardless of their previous programming experience.

5.4 Entrance requirements

The entrance requirement for the MSc in Computer Systems Engineering is a good honours degree, or its equivalent. Some knowlege of the physical sciences, Electrical Engineering, Mathematics or Computer Science is essential, though this need not necessarily be at degree level. Some experience of programming is helpful, though not essential for otherwise suitably qualified applicants.

5.5 Assessment

Course modules are examined in April by means of two written examinations, of three hours duration each, on the subject matter of the prescribed modules. The examinations may cover material presented in formal lectures, seminars and practical work.

Students who achieve a satisfactory standard in these examinations are permitted to proceed to the project phase of the degree. Students who fail to achieve a satisfactory standard may be permitted to transfer their registration to the appropriate Diploma course if the Board of Examiners so recommends.

In the final six months of the course each student works full time on an individual project. This is assessed by means of a dissertation reporting the work which must be submitted before the end of September.

An oral examination may be required and may cover any aspect of the lecture material, coursework and project work.

Computability and Formal Languages

Introduction, Cantor's diagonal argument, Turing's thesis. Turing machines, partial recursive functions, TM techniques, variations on TMs, simulation arguments.
Other models of computation, stack machines, counter machines, queue machines, equivalence to TMs.
Universal Turing machines. Undecidabiltiy, halting and other problems, recursive and recursively enumerable sets. McCarthy recursion equations, operational semantics. Chomsky hierarchy, grammars versus machines, closure operations, normal forms, decision processes, parsing.
Type 0 languages, context sensitive languages, regular languages, CF grammars, parse trees, Chomsky normal form.
Push down automata, equivalence problem. Parsing, Younger's method, LL(k) parsing.

Semantics

Operational semantics of recusion equations and of simple imperative programming language P, the approach and proof of their equivalence.
Data domains and function domains under complete partial order, monotonic and continuous functions, the least fixed-point operator, and its application in solving recursion equations. Proof of properties of recursively defined functions using least fixed-points.
Denotational semantics of the language P, using complete partial order domains, equivalence with operational semantics, proof of properties of language P, use of continuations, extensions to P (including declarations and procedures) and their semantics.

Systems Architecture (10 seminars)

Input and output organisation, handling of simple devices, intelligent devices e.g. FEP's, file store, etc.
Storage organisation, one level storage, storage hierarchy, caches, cost-performance, capabilities, addressing schemes.
Processor organisation, pipelines, stacks. Inter-system communication, buses, multi-processors.

The above concepts are illustrated with examples taken from IBM 360 & 370, DEC PDP-11 and VAX 11/780, CDC 6600/7600, Burroughs 5500/6500, ICL 2900. Reference is also made to other systems.

Operating Systems

Review of batch processing system programs, their components, operating characteristics, user services and their limitaions.
Implementaion techniques for parallel processing of input/output and interrupt handling.
Overall structure of multi-programming systems consideration of multi-processor configurations.
Details of addressing techniques, store management, file system design. Interprocess communication, design of system modules and interfaces.

Compiler Construction

Review of programming language structures, translation, loading, execution and storage allocation.
Compilation of simple expressions and statements. Organisation of a compiler including compile-time and run-time symbol tables.
Lexical scan, syntax scan, object code generation, error diagnosis. Object code optimization techniques and overall design. Use of compiler writing languages. Bootstrapping.

Data Base Design

This is an introductory course into the use and construction of DBMS. The ground covered includes a survey of applications, an introduction to physical storage and access methods, query languages, language embedding, the relational and network models.

Modelling and Systems Performance

Application of probability to computer systems, simple queueing theory, Markov chains, state diagrams, forward-backward equations, steady states, birth and death processes, Little's theorem, Kintchine-Pollaczek equation, network models, decomposability, Buzen's algorithm, diffusion approximation.
Simulation modelling, performance measurement of real systems, workload characterisation, synthetic workloads, bench marks, performance evaluation and prediction.

Analysis of Algorithms

Introductory concepts. Algorithms based on splitting. Recurrence relations. Comparison problems, analysis of sorting, merging and selection problems. Information theoretic lower bounds. The class NP. Reducibilities among problems. NP-completeness. Fast algorithms for multiplying integers and polynomials. The finite Fourier transform. Matrix problems, algorithms and reductions, (e.g. Boolean transitive closure, shortest paths, determinates).

SECTION 6 POSTGRADUATE STUDY PROGRAMME IN COMPUTATION THEORY

6.1 Outline and purpose of the programme

The Department of Computer Science at Edinburgh University offers a postgraduate study programme in the Theory of Computation, both pure and applied. In their first year, students attend an informal course of about 100 lectures, plus seminars, designed to give them a suitable grounding for research in this area. As the first year proceeds they are also guided towards a research topic.

These lectures are open to all, and a small number of one-year visitors and non-graduating students, who wish to attend but not to register for a degree may be accommodated.

Students require considerable mathematical training, for example an undergraduate degree in Mathematics, Mathematics and Computer Science or Mathematical Physics. Some computing experience is expected but a substantial knowledge of Computer Science is not a requirement.

The aim is to develop skill in applying mathematical ideas to computing problems, notably the proof of properties of programs, the quantitative analysis of algorithms, the complexity of computing tasks and the semantics of programming languages. Much progress has been made in the last ten years using ideas from mathematical logic, algebra, analysis, recursion theory, combinatorics and other branches of mathematics. This work is now beginning to affect methods of developing reliable and efficient software. Students will develop both mathematical understanding and practical programming skills.

6.2 Course Structure

The lectures, numbering about 100 in all, are given in the Autumn term and early Spring term, and are divided into three broad sections: Complexity, Pragmatics and Semantics. More advanced topics are covered in seminars in the second half of the Spring term. Theoretical and programming exercises are set in conjunction with the lectures. The course is taught by four members of the teaching staff - R. Burstall, R. Milner, G. Plotkin and L. Valiant - and by associated researchers.

All students do the associated coursework. Parts of the Complexity and Pragmatics material may perhaps be omitted by students who have covered these topics. Each student is asked to attend further lectures and seminars which both link and further develop these areas. Those with little practical experience may also be asked to attend a practical course chosen from the rest of the Computer Science syllabus.

Most of the lectures and seminars are completed by Easter, to enable students to concentrate fully upon their research topics thereafter.

6.3.1 Syllabus: Complexity

Analysis of Algorithms (18 lectures; CS3 course)

Introductory concepts. Algorithms based on splitting. Recurrence relations. Comparison problems, analysis of sorting, merging and selection problems. Information theoretic lower bounds. The class NP. Reducibilities among problems. NP-completeness. Fast algorithms for multiplying integers and polynomials. The finite Fourier transform. Matrix problems, algorithms and reductions, (e.g. Boolean transitive closure, shortest paths, determinates).

Computational Complexity (12 lectures; CS4 option course)

Time and space as resources for Turing machines, nondeterminism.
Complexity classes, padding arguments and time and space hierarchies.
Polynomial reducibilities and Cook's theorem.
Problems complete in PSPACE and #P.
Provably intractable problems, finite and infinite.
Time/space tradeoffs, pebbling and Savitch's theorem.
Axiomatic complexity theory.

Further topics in complexity (10 seminars)

Surveys of particular research areas e.g. parallelism, cryptography, interconnection patterns, probabilistic analysis, algebraic algorithms.

6.3.2 Syllabus: Pragmatics

Programming Languages (20 lectures/tutorials)

Getting started with the DEC-10. Introduction to POP-2. Introduction to LISP. Introduction to PASCAL. Programming language concepts: applicative languages, strongly typed languages, modules, data abstraction. Common data types: sets, sequences, bags, maps, trees, graphs. Programming exercises based on the above.

Elements of Computational Logic (with AI PG course)

First order predicate calculus and its semantics. Introduction to resolution based theorem proving.

Logical Analysis of Programs (8 lectures)

Methods of program specification. Methods of program verification. Floyd assertions, Hoare's axiomatic approach, inductive techniques, examples. Program transformation and synthesis.

Control of Data Flow

Overview: precisely describing digital systems; implementation independent properties of digital systems; fundamental limitations; cost and performance of various realisations (implementations) of digital systems.
Directed graph description of systems. Implementation independent properties; determinacy. Asynchronous modular systems. Pipeline schemata. Pipeline systems. Petri nets; reachability; vector addition systems; hierarchical modelling; liveness; safeness; proper termination. DEC RT-modules (PDP16 kits); interpretation of liveness, safeness and PT in terms of asynchronous modules and RT-modules; relationship of AMs to RTMs. Basic computation schemata; variable depth computations; determinacy of basic computation schemata. Basic modular systems; relationship to RTMs. Signalling level correctness of AM and RTM implementations.
The link with synchronous systems; unification of the synchronous view and the asynchronous view; microprogramming.
Implementing sequential automata with PLAs; microsequencers.
Microprogramming; levels of concurrency; describing microprograms; introduction to machine-independent microprogramming.
Fundamental limits: synchroniser failure; clock skew.
Other directed graph and token-flow models of computation.

Introduction to Digital Communications

Communications theory, digital communications, parallel and serial forms, codes, errors etc. Parallel synchronisation, handshakes etc,. Serial synchronization, bit synchronization, frame and character synchronization.
Packet formats and protocols, polled, asynchronous, half duplex, full duplex, BSC, HASP ML,. HDLC, DDCMP. Packet network control, congestion, flow control, error control.
Network services, high level protocols.

Programming Methodology

The ML language. An introduction to its use and its ideas.
The ADA language — a critical review.
The ADA project and its aims.
Expressions and statements, types and declarations, subprograms and parameters.
Packages and abstract data types.
Concurrency; seperate compilation; exception handling.
The co-routine concept.
Program design. Discussion of systematic design of programs, starting with formal and informal specification.
Attention to specifying modules, data types and procedures.
Examples of non-trivial program design. ML as a programming language.

4.3 Computer Science 3 (3rd year)

Course Co-ordinator: J.Tansley.

4.3.1 Introduction

The third year provides a basic foundation for the design and implementation of computer systems. Computer hardware design is given greater emphasis, reflecting recent developments in methods of hardware implementation. Software continues to be studied, though more effort is now devoted to the overall design of software systems and to their interaction with underlying hardware. Throughout the course the theoretical foundations of computational ideas are also examined.

4.3.2 Course Modules

Switching Theory

Boolean Algebra. Functions and expressions, duality, simplification and manipulation of expressions. Complete sets of logic primitives.
Combinational circuits. Analysis - Karnaugh maps, simplification, SP and PS forms, min terms. Decompostion of logic circuits.
Sequential circuits. Sequential machines, a simple taxonomy. Memory elements, delays, clocks. Analysis and synthesis of synchronous circuits. Functions not realizable by finite state machines, large scale design limitations.
Timing and Structure. Asychronous machines in detail, their analysis and synthesis.
Modular logic design. Universal logic modules, other logic modules e.g. selectors and multiplexors. Bounds on terminals. Cellular arrays.
Standard MSI and LSI components. The 7400 series etc. Use of these components. Logic simulation.

Computing Technology

Digital abstraction -- some necessary structural requirements for computation. The physical world and representation of information. The digital world some theoretical constructs.
The digital-analogue boundary, mainly from an electrical point of view. Information transmission, manipulation and storage. Some basic electrical theory, models at both an analogue and digital level.
Insulated gate MOS semi-conductor technolgy; an example technolgy for digital systems implementation.
Development of a design methodology.
Storage, RAM, ROM, dynamic and static cell designs.
Other technologies, semi-conductor and other solid state based devices. E.G. IIL, bubbles and magnetic media.
Timing and structure. Asynchronous and synchronous systems; the meta stability phenomenon.
VLSI system, intefaces, modules, system building, testing. The 68000 and a 64k ram.

6.3.3 Syllabus: Semantics

Denotational Semantics (12 lectures)

Syntactic and semantic domains. Notions of environment, abstract store and continuations. Semantic functions. Treatment of ALGOL and PASCAL features, input/output and errors, compile and run time type checking.

Domains (14 lectures) Complete partially ordered sets, approximation and limit, least fixed points and continuous functions. Typed lambda-calculus, operations on domains. The inverse-limit construction. Type-free lambda-calculus and other examples. Algebraic cpo's. Computability.

Algebras and Categories (6 lectures)

General algebras and homomorphisms. Introduction to category theory: categories, functors, natural transformations, freeness, colimits.

Operational Semantics (8 lectures)

Abstract machines. Evaluation. Semantics as inference. Relationship with denotational semantics.

7.1 Very Large Scale Integrated Circuits

The field of Very Large Scale Integrated (VLSI) Circuit design, that is the ability to integrate upwards of 100,000 devices on a single chip, provides a focus for research workers in a number of areas within the Department. Some of the areas are: novel systems architecture; computer-aided design; special-purpose devices; and system correctness and complexity. Emphasis is placed on principles of modular design and decomposition based on the work of Mead and Conway at Cal. Tech., and a library of artwork for standard components is being developed.

Edinburgh University has recently been designated for special support in this field, support which is reflected in the facilities available for this research. These include: software design aids on VAX, two colour graphics terminals and two colour plotters. In addition the Department has access to fabrication facilities in the Department of Electrical Engineering and the SRC Rutherford Laboratory.

7.2 Programming Methodology

This project, carried out by Rod Burstall, Malcolm Bird, David Rydeheard, John Scott and postgraduate students, is funded by the Science Research Council. The main topics are:-

a) Advanced programming languages, with particular emphasis on functional programming and the use of abstract data types and modularity to structure large programs. An experimental functional language HOPE has been designed and implemented (with the participation of David MacQueen, ISI, Los Angeles); it is strongly typed and is being extended with modularity features. Another language IVY is oriented to the exploitation of a rich collection of built-in data types.

b) Specification of problems and programs. Specifications of large programs can be complex and need to be presented in a clear and modular way. CLEAR is a specification language developed in collaboration with J.A. Goguen, SRI, Palo Alto. Extensions to CLEAR and applications of it are being investigated.

c) Applications of Algebra and Category Theory. The semantics of CLEAR has been given in an algebraic/categorical manner. This has stimulated work on embodying categorical concepts directly in programs, "categorical programming".

4.2.2 Syllabus

Computer Systems

A study of computer systems at several different levels. The material in the lectures is illustrated by reference to various computers used by the students. Practical work includes the opportunity to work with microprocessors.

The functional parts (processors, memories, input/ouput devices) and information flow round the system.
A more detailed study at the register-transfer level of a typical simple processor (a commonly used micro-processor); communication between processor and memory; the execution of a machine-code program; communication with the outside world; problems of interfacing input or output devices.
Some useful hardware components described at the digital logic level: gates, flip-flops, registers and counters.
Technologies commonly used to implement computer systems. Recent technological developments, and their influence on the design of computer systems.

Computational Structures

Further high-level language programming; simple techniques for proving program correctness; relationship between iterative and recursive programs.
Graphs, trees and lists; some algorithms relating to these structures; graphical output; sorting and searching techniques.
Abstract data structures; their use in program construction; further proof methods.
An introduction to some parsing and compiling techniques.

Theory

Mathematical preliminaries; review of sets, functions and relations; general notions in algebra.
Finite state automata; acceptors and transition graphs; regular expressions; minimization of automata.
The algebra of sets; Boolean algebra; the algebra of propositions; minimization of Boolean forms.

4.2.4 Assessment

There is a class examination at the end of the first and second terms. In each case this consists of a single two-hour paper.

In the degree examination in June there are two papers, each lasting two hours. The resit examination in September has the same form. In both the June and September examinations, performance of course work throughout the year is taken into consideration. The final assessment is made up as follows:

Paper 1	one third
Paper 2	one third
Course work	one third

Satisfactory completion of course work is a prerequisite for taking the degree examination in June or Spetember.

4.2 Computer Science 2 (2nd year).

Course Co-ordinator: R.Candlin.

During their second year, students move towards a more specialised practical and theoretical study of computer systems. They continue to build on the programming capability developed in Computer Science 1, but the emphasis changes towards a more general and abstract view of computer programs. This involves not only a study of various useful data structures like sets and trees, which can be used in many different contexts to model the external world, but also an introduction to methods of proving that algorithms compute correct results. Also during the second year, students begin to learn about the problems of the definition and translation of programming languages - work that will be continued in the third year course.

An important aspect of second year work is that students learn to back up their intuitive results with mathematical analysis. This point of view becomes even more important in the third and fourth years and the foundations for a mathematical treatment of computing systems are therefore laid at an early stage.

At the same time as students are moving in the direction of a greater abstraction, which permits the power of mathematics to be applied, they are learning how computers work. This is done by studying a simple computer system in detail, so that an understanding is gained of how a machine-code program is executed, and how input/output is controlled. Again, the emphasis is on those principles which are common to all computers, rather than on the idiosyncracies of a particular machine.

4.2.1 Course Structure

Classes are held in the James Clerk Maxwell Building. The course consists of 69 lectures, plus tutorials and practical work. Lectures take place at the following times:

Tuesday 2 p.m.
Tuesday 3.30 p.m.
Thursday 2 p.m.

There is one tutorial hour a week. The majority of tutorial groups meet on Thursday afternoons at 3 p.m. or 4 p.m. In addition, students are expected to spend about ten hours per week on work associated with the course (reading, programming, theory exercises and laboratory work). Students work in their own time, apart from a period of six weeks in the second term when the microprocessor exercises are scheduled. The microprocessor laboratory is open for several hours each day during this period.

7.3 Data Curator

This project, which is funded by the SRC, is carried out by Malcolm Atkinson, together with Ken Chisholm and Paul Cockshott and three research students; others are welcome to join it. The project has its own 32-bit processor on which it is building a database server for a local area network.

The work is distinguished by an interest in the relationship between databases and programming languages. Its long-term aim is to eliminate databases, at least as visible objects. To achieve this end, it intends to support persistent data in a different way. It is assumed that the notions of type as a means of description and constraint, modules as a means of identifying the duration of persistence, and the standard features of programming languages for manipulation can supplant the present ad hoc methods used in most DBMS. This will lead to a more coherent and consistent programming environment and to the development of programming languages to include all aspects of late dynamic binding of persistent data to specific program parts.

The project has constructed a message switching system which enables processes to be distributed over a network, a database kernel system which provides persistent space management, transactions and concurrency control. Under development is a compilation and execution system for a high-level language which incorporates a demonstration of the language ideas. Theoretical problems of specification, consistent identification and a consistent concept of type require attention. A model of the program development, compilation and execution process in the context of modular programs and persistent data has been defined. Work is required to discover how to present and operate on this model in a natural way.

7.4 Application studies in LCF

LCF is a fully implemented interactive proof system, in which properties of computations (for example, of programs) may be rigorously verified by a mixture of automatic and interactive methods.

The current project, being carried out by Robin Milner and Avra Cohn, in association with Mike Gordon and F.V. Jensen (visitor from Aarhus University, Denmark), is focussed upon case-studies of proof. The main aim is to evaluate the methodology which has been developed, consisting principally of (1) the organisation of problems and problem areas in a hierarchic structure of theories, and (2) the use of a powerful metalanguage ML (a high-level programming language in its own right) to raise the quality of interaction by programming and combining partial or total proof strategies. Of particular interest are the verification of compilers and parsers, and establishing properties of useful abstract types. Completed studies are: the verification of a simple but non-trivial compiler, the modelling and analysis of Backus' FP systems, and the investigation of some abstract data structures (for example, binary search trees).

7.5 Mathematical Techniques in the Design of Telecommunications Systems

There are a number of mathematical models, as distinct from programming languages, for concurrent communications systems; among them are Net Theory (Petri et al), Path Expressions (Campbell, Lauer et al) and CCS (Milner et al). The aim of this project, which is carried out by Robin Milner and Mike Shields and is jointly funded by the SRC and the Standard Telecommunication Laboratory at Harlow, is to conduct non-trivial case studies, proposed by STL, in the application of these models. The emphasis is upon the descriptive power of the models, and the techniques they offer for analysis, specification and verification of concurrent systems (particularly telecommunications systems). The outcome should be not only techniques useful in real design problems, but also feedback to the work in the Department on fundamental models of concurrency.

7.6 Semantics of Non-deterministic and Concurrent Computation

This research, which is being conducted by Robin Milner and Gordon Plotkin together with Matthew Hennessy and John Kennaway, concerns the foundations of non-deterministic and concurrent computation. The aim is to provide a uniform framework containing mathematical models for the intuitive ideas of an event, of process communication and of synchronisation. The mathematics involved is continuous, as advocated by Scott, and uses tools from algebra and category theory.

7.7 Semantics of Abstract Data Types

The aim of this project, carried out by Gordon Plotkin together with Mike Smyth, is to develop further the application of Scott's theory of computation to the study of the synthetic approach to abstract data types. It is intended to pursue a wide variety of topics ranging from the detailed study of practical examples to theoretical problems and to include systematic comparisons with other approaches.

7.8 Semantics and Correctness of Digital Systems

Work is being carried out by Mike Gordon to develop techniques for modelling the functional behaviour of clocked synchronous systems such as instruction set processors. Such systems can be described at many levels, for example:

1. In terms of the connections between sequential devices such as registers, memories etc., and combinational devices such as adders or multiplexors. At this level the primitive operations are the opening and closing of gates.
2. In terms of register transfers and tests. At this level the interconnection topology has been abstracted away.
3. In terms of higher level data types and their operations. For example, a processor could be described by its machine instructions on bytes, words, integers etc.

The goal is to devise natural and tractable mathematical models for these levels, and proof techniques for verifying that each level is correctly implemented by the one below it.

Half Courses

It is possible to take a half course in computer science in the first year as follows:-

Computer Science 1Ah: consists of the first half of Computer Science 1, that is the material described in the first paragraph of the previous section.

Computer Science 1Ch: consists of the second half of Computer Science 1 that is the material described in the second paragraph of the previous section. Prior attendance at Computer Science 1Ah is required for entry to this course.

4.1.4 Assessment

Assessment is made on the basis of the course work and the degree examination in June, the two components counting equally towards the final result. There are also class examinations in December and March whose prime purpose is to give information on progress to teachers and students but which also give practice for the June degree examination which takes the same short answer format.

A student who maintains a high standard throughout the year in both course work and class exams may be awarded a merit certificate. Any student awarded a merit certificate will also be granted exemption from the degree exam.

A student attending Computer Science 1 is treated, for assessment purposes, as though attending Computer Science 1Ah followed by 1Ch. There is a distinct break in the course work at the halfway stage. With the exception of the class exam in December, all written exams are in 2 distinct halves. A student who does not attain sufficient standard in Computer Science 1 as a whole may be awarded a pass in either Computer Science 1Ah or 1Ch separately.

There is also a practical examination in June from which the overwhelming majority of students gain exemption by achieving a satisfactory standard of course work during the year.

4.1 Computer Science 1 (1st year)

Course Co-ordinator: F. Stacey

4.1.1 Introduction

The aims of this course are to develop basic skill in the production of the software components of computer systems and to give an understanding of the principal problems and common solutions encountered in the design of effective systems. The first aim cannot be achieved without considerable practical experience. To this end much importance is attached to the practical work component of the course, to the extent that it forms an significant part of the final assessment.

4.1.2 Course structure

Lectures take place in the Appleton Tower at the following times:

Tuesday 2 p.m. (but see below)
Thursday 2 p.m.
Thursday 4 p.m.

Because of a time-table clash with Engineering Science 1, an alternative for the Tuesday lecture is given on Monday at 2 p.m. in the James Clerk Maxwell Building.

4.1.3 Syllabus

Lectures and practicals during the first 2 weeks of the course form an introduction to computing systems and their use through a large multi-access system. During the remaining 10 weeks of the first half of the course 2 lecture streams run in parallel. The first, at 2 lectures per week and almost all the practical, is devoted to programming in a systematic way using the language PASCAL. The second stream deals with methods of reasoning precisely about potentially very large amounts of data. Emphasis is given to a relational model of databases. The abstract ideas are presented against the background of a number of actual systems which are described in sufficient detail to illustrate the differences between theory and current practice.

For the second half of the course (12 weeks) the emphasis is changed somewhat. A stream of 1 lecture per week develops the basic programming skills acquired, introducing a selection of standard techniques such as list and tree processing and a selection of standard searching and sorting algorithms. The last few lectures in this stream are used to introduce other programming languages in order to set PASCAL in a more general context. The other stream (of 2 lectures per week) concerns itself with aspects of computer systems in general. Twelve lectures are devoted to a functional description of the principal hardware and software components: memories, processors, operating systems, memory management and file store management. A further 6 lectures are devoted to a description of the technologies of communications and computer graphics. The mutual interactions of developments in these technologies and computer technology are then described. The last few lectures are devoted to a description of the effects of the preceding technologies on the printing and publishing industry.

7.9 Computational Complexity and Algorithms

This research, conducted by Les Valiant together with Martin Furer and Kyriakos Kalorkoti, studies the space and time required for computations. Topics include a unified algebraic theory of the complexity of algebraic and combinatorial computations, algorithms with good probabilistic behaviour, size of basic hardware structures, classifying problems by combinatorial difficulty and lower bounds on complexity.

7.10 Stylistic Analysis

Many problems in literary studies have their origin in the uncertain authorship of the texts. This is particularly true when historical information is scarce and the decision about the authorship of the texts must be based on an examination of their contents. Students of style claim that from an examination of any composition its author may be determined, given a text large enough to supply a reasonable specimen and given enough material with which to compare it. However, traditional stylistic analysis is seriously defective. Where the differences between two texts are large and numerous, that is, where a method is least required, the results are most convincing; where the differences are few and slight, that is, where a method would be most useful, traditional analysis is least effective.

Clearly something better is needed than subjective evaluations of style. A science of stylometry is needed in which personal elements in composition are measured so that the works of any writer can be distinguished from those of colleagues who might be writing on the same subject, at the same time, for similar reasons and in an identical historical and cultural situation.

This research, conducted by Sidney Michaelson and Andrew Morton, is developing such a science and applying it to a wide variety of authorship problems, ranging from the Homeric question to to disputed wills.

SECTION 8 COMPUTING FACILITIES

The University computing service is provided by the Edinburgh Regional Computing Centre. The principal mainframe facility for University users is the twin ICL 2972 system housed in the James Clerk Maxwell Building. This installation runs the Edinburgh Multi-Access System (EMAS) to support in excess of 100 simultaneous users at interactive terminals sited throughout the University and connected by a local network. The network also permits access to a number of other processors, including another large ICL 2900 installation, and to a variety of devices, such as printers, plotters and type-setters. Specific research projects have access to the Science Research Council's DEC-10 at Edinburgh; others have their own dedicated machines.

The principal departmental computing facility is a VAX 11/780 housed in the Department's machine halls. This machine was installed at the end of 1978 and has since been upgraded to 2 Megabytes of main store and 1200 Megabytes of disk storage. It supports a maximum of around thirty simultaneous users on VMS and is connected to the ERCC network. The main languages available are Pascal, IMP and Fortran.

In addition the Department has a number of mini-computers which are used to provide hands-on experience and in a number of specialised roles. These machines include a PDP-15/20, a PDP-15/40, an ICL 7502, a Terak, 14 Interdata 70 series machines, and two high-quality graphics terminals based on PDP-11 processors. Most of the machines are equipped with high-speed general-purpose interfaces and peripheral devices are interfaced to the same standard to permit the maximum flexibility of inter-connection. The interfaces, which were designed in the Department, also allow inter-processor communication at rates up to 2 Megabaud.

Most of the small systems have no independent file or backing storage but rely on connection to a common filestore for their filing systems. The filestore, based on an Interdata 70, supports about 16 client machines (some with multiple users) with two 60 Megabyte disk drives; it also spools line-printer output.

The Department and ERCC have been jointly concerned with the setting up of the micro-computer laboratory in the Appleton Tower, of which the Information Systems course is a major user.

There is a micro-processor laboratory in the machine halls equipped with prototyping kits which are used for experiments using the 6800 and Z80 series micros in particular. There is also a well-equipped electronics workshop which contains a selection of electronic measuring equipment and hand tools.

necessary for an applicant to be offered a place. In the past, grades of AABB at Higher, or BCC at A-level have been required. There are no prerequisite subjects, and students who have taken only Arts subjects are accepted, provided they have shown their ability. However, there is a considerable amount of mathematically-based material taught in the third and fourth years, so that it is preferable for a student to have continued with Mathematics to Higher or A-level. Applicants for joint degrees must of course satisfy the requirements of the other subject as well.

4.0.2 Overall course description

For their first two years, students spend one third of their time on Computer Science. Many students have had experience of working with computers at school, but there are many who have not, and the first year forms a broad introduction to the subject. From the very first, students are expected to obtain a practical experience of using a computer, which in this case is the University's large multi-access system, 2900 EMAS. In the second and subsequent years they have the opportunity of working with a variety of other computer systems ranging from the Department's VAX multi-access system, down through small mini-computers, to microprocessor systems. In the early years, students write software for existing hardware configurations; later they construct their own hardware from standard components, and have the opportunity to design their own VLSI chips.

Throughout the course, emphasis is laid on the idea that a computer system is a fusion of hardware and software design. The fundamental ideas behind computers and computing can also be represented in mathematical terms, and this abstract and theoretical treatment forms an important part of the third and fourth year courses.

4.0.3 Assessment

There is a written examination in June each year, which is referred to as a "degree" examination. Passes in the first, second and third years are counted towards an Ordinary B.Sc. The Honours degree is awarded on the basis of the third and fourth year examinations. In all these examinations, work carried out during the year contributes towards the assessment.

In the first three years there are also less formal "class" examinations during the year, which enable students to check their own progress.

SECTION 4 UNDERGRADUATE COURSES IN COMPUTER SCIENCE

4.0 General Director of Studies: R. Candlin

Several Honours degrees are available at Edinburgh; a single Honours degree in Computer Science, and five joint Honours degrees in which Computer Science is combined equally with another subject. The following degrees are offered:

Computer Science
Computer Science & Electronics
Computer Science & Management Science
Computer Science & Mathematics
Computer Science & Physics
Computer Science & Statistics

An Honours degree normally lasts four years. During each of the first two years, students take three subjects, of which one is Computer Science. Students who intend to take a joint degree also have to take prescribed courses relevant to that subject. There is usually the possibility of taking one or two "outside" subjects, which are not essential components of a given Honours course, but which give students the opportunity to broaden their interests. In their final two years, students follow courses in their chosen speciality.

The degree structure is very flexible, and students can to a certain extent keep their options open until the end of their first or second year, as far as their choice of degree is concerned.

For example, a student registered for a joint degree can easily change to a single Honours degree in one or other component by opting to do so at the beginning of the third year. Many students take the first and second year courses of Computer Science as outside subjects for other Honours degrees (for example in Engineering or Business Studies). In some cases, they become so interested in Computer Science that they decide to transfer to Computer Science Honours. This can usually be done quite easily, provided that Computer Science has been included as a subject from the first year.

Computer Science is often an important component of the Ordinary B.Sc. degree, which provides a three-year course for those who do not wish to specialise. A recently introduced compromise between the unspecialised Ordinary B.Sc. degree, and the highly specialised Honours degree is the Ordinary B.Sc. in a designated discipline. For this degree, students continue their study of Computer Science into their third year, by following selected parts of the full Honours course.

4.0.1 Entry Requirements

Prospective students must satisfy the University's "general entrance requirement", which demands a certain level of achievement over a range of subjects. Details can be found in the University Undergraduate prospectus. For most Computer Science Honours courses, qualifications above the minimum are required. Each application is considered on its own merits, but as there are more applicants than places, a good standard of performance in Higher or A-level examinations has become

SECTION 9 MEMBERS OF STAFF

9.1 Teaching staff

S. Michaelson B.Sc., A.R.C.S., F.I.M.A., F.R.S.E., F.R.S.A.; Professor of Computer Science.
The study of literary style with special reference to problems of authorship and chronology.
Queue-related models of computing systems.

R.M. Burstall M.A., M.Sc., Ph.D.; Professor of Computer Science.
Programming methodology: correctness proofs, program transformation, specification languages. Semantics using an algebraic/categorical approach.

P.D.A. Schofield B.Sc., A.R.C.S.; Senior Lecturer and Head of Dept.
Programming techniques. Data structures.

M.P. Atkinson B.A., M.A., Dip.Comp.Sci., Ph.D.; Lecturer.
Relationships between programming languages and database systems. Development of techniques for the design and construction of large-scale software.

I. Buchanan B.Sc., Ph.D.; Part-time lecturer.
Computer-aided design and VLSI.

E.R.S. Candlin M.A., Ph.D.; Lecturer.
Introductory teaching and specialised micro-processor systems.

H. Dewar B.A., B.Phil.; Lecturer.
Small machine architecture. Text processing.
Speech input/output.

G.H. Efland B.S., M.S.; Demonstrator (until 31/3/81).
VLSI design

J.P. Gray B.Sc., Ph.D.; Part-time lecturer.
Computer-aided design and VLSI.

I. Hansen Mgr., Ph.D.; Lecturer.
Design methodology for microware description and analysis.

M.R. Jerrum B.A.; Demonstrator.
Complexity of computation.

P. McLellan B.A.; Demonstrator.
Programming techniques. Distributed filing systems.
Small machine systems.

A.J.R.G. Milner — B.A.; Reader.
Semantics of programming languages. Applications of mathematical logic to formalise the statement and proof of assertions concerning programs. Abstract models of concurrent computation.

G. Plotkin — B.Sc., Ph.D.; Lecturer.
The denotational semantics of programming languages with emphasis on concurrency. Computational and inductive logic.

P. Rashidi — B.Sc., Ph.D., A.C.G.I., D.I.C.; Lecturer.
Signal processing and digital filtering. Micro-computer systems. Numerical analysis.

D.J. Rees — B.Sc., Ph.D., A.R.C.S; Lecturer.
VLSI design and associated design tools. Design and implementation of multi-access operating systems.

J.S. Rohl — Visiting professor (until February 1981)
Programming and program transformation. Compilers.

N. Shelness — B.A.; Lecturer.
Research into loosely-coupled computer systems.

F. Stacey — B.Sc; Lecturer.
Systems software.

L.D. Smith — M.A.; Lecturer.
Design and simulation of digital systems. Computer system architecture. Integrated circuit design.

J. Tansley — B.Sc; Lecturer.
Switching theory. Descriptive methods in the design and construction of digital systems. Digital methods in signal and image processing.

L.G. Valiant — B.A., M.A., D.I.C., Ph.D.; Lecturer.
Computational complexity, analysis of algorithms, automata theory, combinatorial mathematics, parallel computation.

A.S. Wight — M.A., Ph.D.; Lecturer (on leave of absence)
Computer performance evaluation. Remote terminal emulation. Workload characterisation.

3.4 Course Assessment

The overall assessment of performance on the course is based on the following in the indicated proportions:

Practical Work 30%
Essay Assignments 30%
Examination 40%

There are class tests held at the end of terms 1 and 2. If the combined results of the class tests, the essay assignments and the practical work generate a clear indication of good performance, exemption will be granted from the final examination. A poor performance in the Degree Examination may be recovered at the resit examination, a poor performance in the essay assignments may be compensated by writing two further essays in the summer vacation, but a pass is required in the practical work alone, and there are no arrangements for retaking the practical work.
The course does not normally lead to admission into the second-year Computer Science course, although students who perform exceptionally well may be admitted to Computer Science 2.

3.2 Presentation of the course

The course is made up of lectures, tutorials and practical work (including a number of essays). Lectures are presented in the Appleton Tower at the following times:

Tuesday 2 p.m.
Thursday 2 p.m.
Thursday 4 p.m.

Tutorials provide an opportunity to discuss any aspect of the course with fellow-students and a member of the Department. Tutorials are held at the rate of one per fortnight during each teaching term. The practical work is carried out in the Micro-computer Laboratory situated on the fourth floor of the Appleton Tower which is equipped with about a dozen Apple 2 and Terak systems as well as video terminals and equipment for experiments in micro-processor control. Laboratory sessions last two hours, at the rate of one a week.

3.3 Syllabus

In the first four weeks of the course, an attempt is made to introduce the terminology and technology of this subject area, so that it loses any mythology, daunting aspect or popular press image it may have. The next period of 5 weeks investigates the task of identifying problems and complete systems to "satisfy" people with those problems. Problems which are familiar to most people are therefore taken, identified more carefully, and some of the difficulties and methods of solution are discussed. This is illustrated with various aspects of handling text and documents, and finally by some user's experiences in architecture.

In the next section, lasting about 5 weeks, the nature of large information handling systems is examined . A progression is shown, from the established systems, to those that are in the stages of early experiment. The student is encouraged to discriminate between what we can achieve, what we can specify, and what we should vaguely like to have. This discrimination is also encouraged by a review of various advanced aspects of applying computers. This leads to a treatment of a particular design problem (large scale electronic circuit design) as an example of the computer as a complementary tool to man's ability to make priority and value judgements and to be creative. Having studied, and experimented with, a number of information systems, the student will be aware that the major component of such a system is its software, and that this presents difficulties in its construction. However, writing small programs is easy, and frequently useful. To develop this ability and to permit an appreciation of the problems of the software industry, the student is introduced to a high level programming language (a simple but usable subset of Pascal).

In the first part of the final term a general application area which is of interest to many people is considered in detail. This year medicine and health care is taken as an example. The aim is to note the diversity of ways in which the computer can have effects and to observe where these have been beneficial, and where they have been deletorious. The course concludes by relating the earlier topics to current events and society, and with a number of visits to computer installations.

9.2 Computing officers

K. Humphry	M.A. (part-time). First-year teaching. Filestore management.
M.R. King	B.Sc., M.Sc. Hardware design using micro-processors. Semi-conductor stores.
W.A. Laing	B.Sc., M.Phil. The development of VLSI design and implementation aids. Analysis and development of virtual memory operating systems.
R.N. Procter	B.Sc., Ph.D. Micro-computer systems. New technology for graphical devices. Social impact of computers.
A. Vernon	B.Sc. VAX management.

9.3 Secretarial staff

H. Carlin (part-time)

J. Fisher

A. Fleming (part-time)

E. Kerse

D. McKie (part-time)

G. Temple (part-time)

9.4 Technical staff

J. Dow	Maintenance. Communications.
J. Johnstone	Information Systems Laboratory. Video interfaces.
P.J. Lindsay	Graphical devices. Link interfaces.
I. Thomson	Equipment construction. Communications.

9.5 Research staff

M.R. Bird	B.A.; Programming methodology. Mathematical theory of computation. Applications of category theory.
K.J. Chisholm	B.Sc.; Data Curator. Programming languages and Databases. Computer-Aided Design.
W.P. Cockshott	B.A., M.Sc.; Data Curator. Programming languages and Databases. Message-based languages.
A.J. Cohn	B.S., Ph.D.; LCF Applications. Semantics. Machine proof. Program correctness.
M. Furer	Dr.Sc.; Complexity. Studies in computational complexity.
M.J.C. Gordon	B.A., Ph.D.; Correctness of Digital Systems. Formal description of languages and systems, verification, functional programming.
M.C.B. Hennessy	B.Sc., M.A., Ph.D.; Concurrent Computation. The mathematical semantics of programming languages and the design of program-oriented logical proof systems.
K. Kalorkoti	B.Sc., Ph.D.; Complexity. Studies in computational complexity.
J.R. Kennaway	B.Sc., M.Sc.; Concurrent Computation. Semantics of non-deterministic and concurrent computation
A.Q. Morton	M.A., B.D.; British Academy Librarian. The study of literary style in relation to questions of authorship and chronology.
D.E. Rydeheard	B.A.; Programming Methodology. Applications of category theory to programming. Specification languages.
J.J. Scott	M.A., Ph.D.; Programming Methodology. Abstract data types.
M.W. Shields	B.A., Ph.D.; Mathematical Techniques in Telecomms. Mathematical models, languages and proof techniques for concurrent computation.
M.B. Smyth	B.A., M.Phil., M.Sc., D.Phil.; Abstract Data Types. Algebraic approach to data types. Computability in a general setting. Some aspects of concurrency.

SECTION 3 UNDERGRADUATE COURSE IN INFORMATION SYSTEMS

3.1 Introduction

Course Co-ordinator: M.P. Atkinson

The first-year Informations Systems course is offered for the first time in Session 1980/81. The motive in presenting this course is to increase the number of people in society who are well informed about computers. The computer is becoming very much cheaper, a process which has been accelerated by the microprocessor. This removes one of the impediments to its increased application. (It is already a significant tool in many aspects of our society). There are other impediments to its application which are not so easily overcome. Deciding what it should be used for, precisely what is required and how these facilities should be made available, are typical examples. Such decisions cannot become or remain the prerogative of the computer scientist alone. It is important that future managers, politicians, lawyers, doctors etc., should be well equipped to make or influence such decisions. It is the aim of this course to nudge them into taking an interest in this field, and initiating them into the acquisition of the relevant knowledge.

It is envisaged that those on the course may one day find themselves in a position akin to one of the following:

a) As a scientist, doctor, linguist or sociologist who has to decide whether it is worth trying to use a computer to assist in the solution of a particular problem.
b) As a manager who has to decide whether resources should be made available for some computing project.
c) As a politician who has to decide what changes in social economic policy are feasible in a given timescale.
d) As a lawyer who has to draft new laws to incorporate the changes appearing in society as a consequence of our dependence on digital systems.
e) As a member of a design team or planning body considering the design of a new information system.

It is quite impractical for many of the people who will find themselves in these positions to become experts in the construction of computer systems. It is, however, entirely sensible and desirable that as many as possible should be well informed, in general terms, about computer-based systems. It is this function which the course sets out to fulfil. Consequently, students from any faculty, in any year and with any background (other than having already attended a Computer Science course) will be welcomed on the course.

The kind of understanding which the course aims to develop is:

1) An appreciation of the flexibility of digital systems and the diversity of their potential application;
2) An appreciation of the interaction between the various technical developments which are taking place today;
3) A critical faculty with which to evaluate statements made about information systems;
4) An awareness of the many factors significant in the design of an information system, and some preliminary skills in assessing whether these factors have been treated successfully and some ability to contribute to such designs;
5) A reasonable fluency, based on experience, in using information systems, and a confidence in approaching others.

Applicants from Commonwealth countries may be eligible for Commonwealth scholarships. Details of these can be obtained from the High Commission in the student's own country. The Department does not have Demonstratorships or Teaching Assistantships for students. Small amounts of money can be earned by supervising practical classes or doing occasional programming work.

Fees covering tuition and practical work have recently been increased by the Government to approximately £3000 per annum, but United Kingdom and EEC students pay approximately £1500; some special grants are available to cover the difference for outstanding overseas students. The British Council also offer studentships to overseas students. Student living costs in Edinburgh are approximately £2000 per annum.

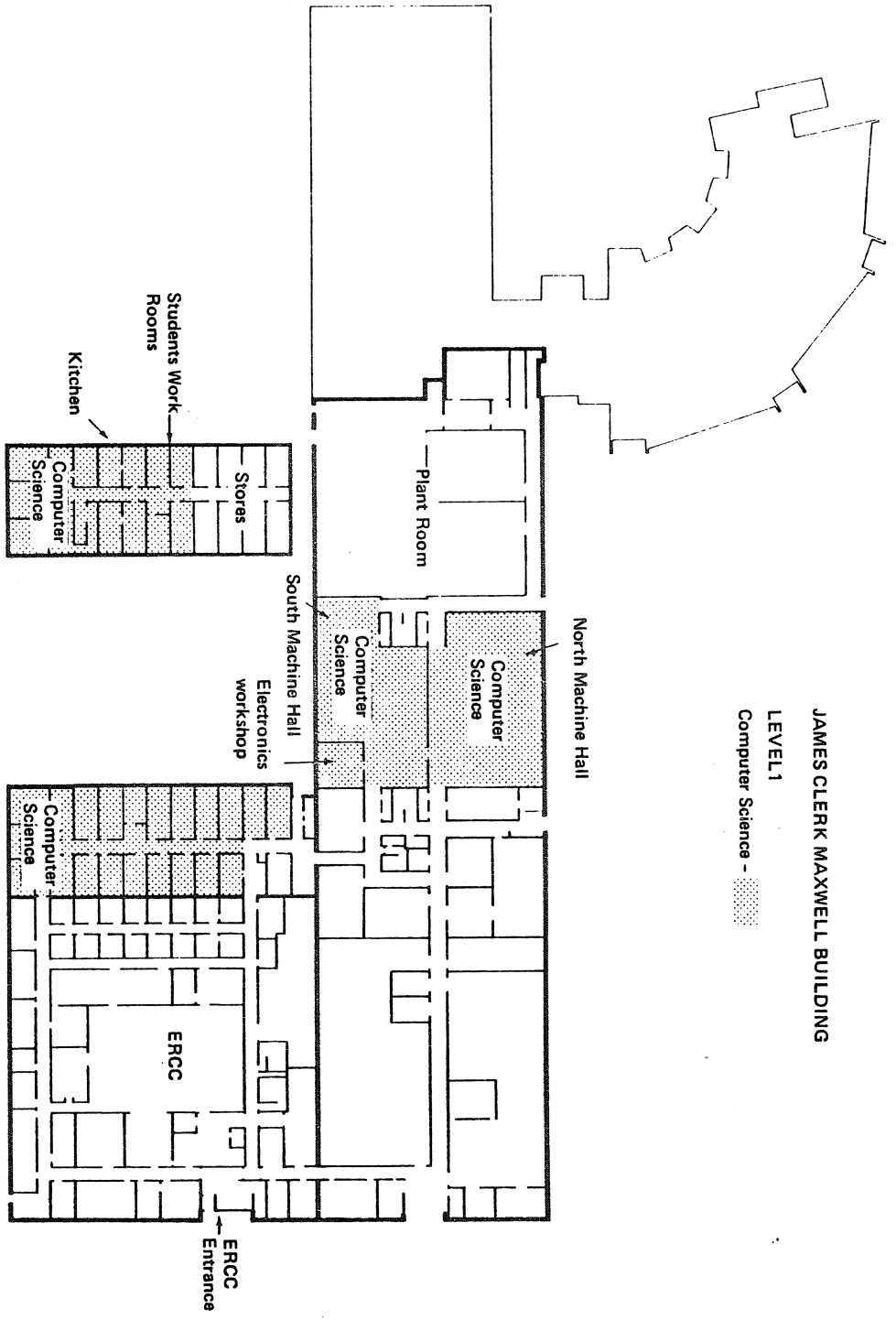
Postgraduate admissions

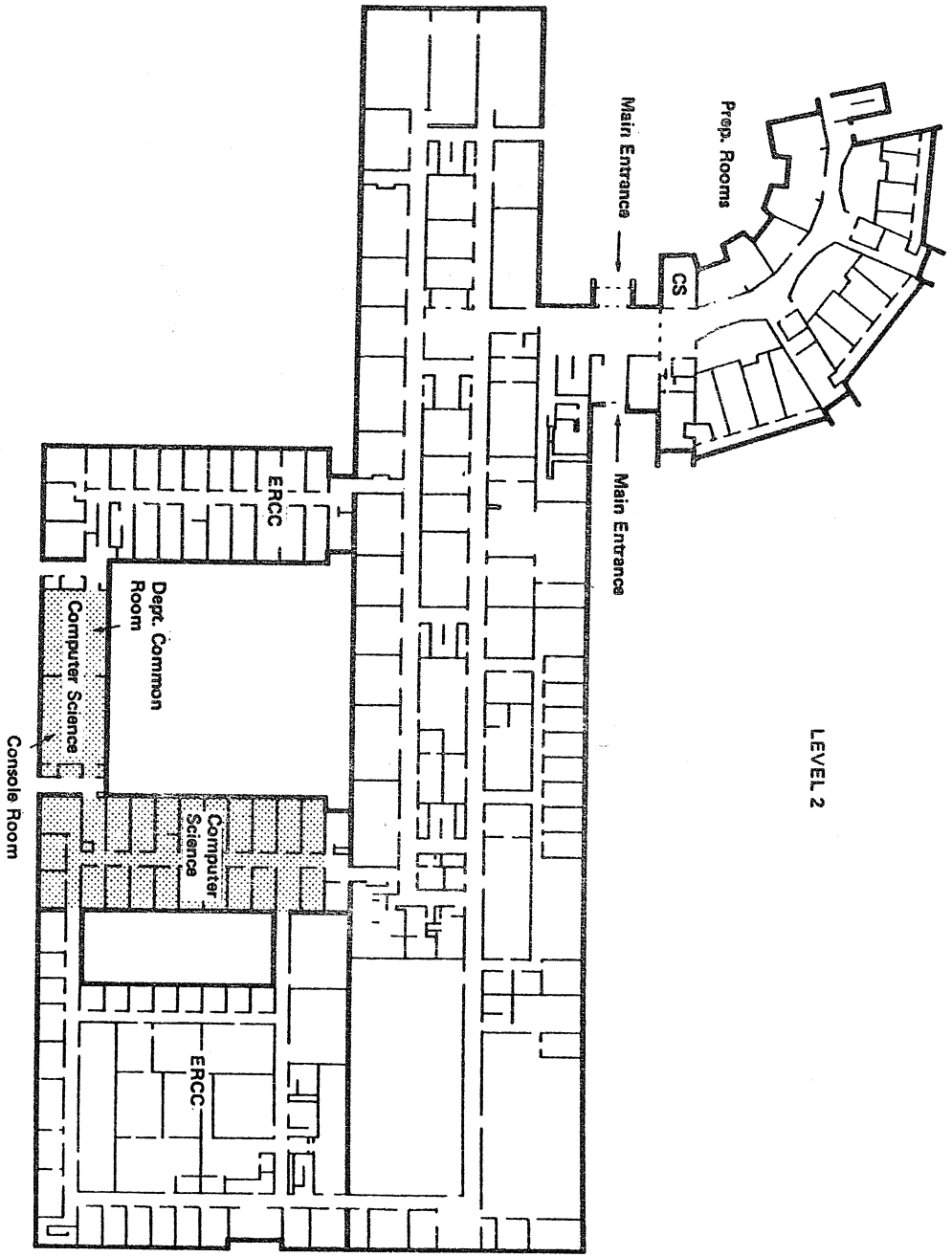
Admissions to Research degrees are handled by Dr. L.G. Valiant and to the M.Sc. course by Dr. D.J. Rees. The Department's address is:

Department of Computer Science,
University of Edinburgh,
The King's Buildings,
Mayfield Road,
EDINBURGH EH9 3JZ.

Related Departments

The Department of Artificial Intelligence, the Machine Intelligence Research Unit and the Department of Electrical Engineering conduct their own postgraduate programmmes and applications for these should be sent direct to them. Students are normally free to attend lectures and seminars which interest them in other departments.





SECTION 2 SURVEY OF COURSES

2.1 Undergraduate courses

The Department offers a first-year course in Information Systems (described in Section 3) and an Honours course in Computer Science (described in Section 4). The first of these is a non-specialist course designed for those who would like to develop an understanding of the computer systems and computer-based facilities which are increasingly encountered in all walks of life. The second aims to provide a grounding in the principles of computation and the art of programming, and, in the later years, to analyse a number of aspects of the design and construction of computer systems.

2.2 Postgraduate study

The Department offers facilities to study for the Research degrees of M.Phil. and Ph.D. and to participate in a course leading to an M.Sc. degree in Computer Systems Engineering (described in Section 5). The M.Sc. is a one-year course with a specific syllabus and timetable. For the Research degrees, the normal period of registration is two years for the M.Phil. and three years for the Ph.D. Part-time study is available if the candidate is either a member of staff of the University or of an 'Associated Institution', or an Edinburgh graduate. Candidates are normally registered in the category of 'Supervised Postgraduate Student' for the first year of study, prior to transfer to the degree course considered appropriate, with back-dating of registration.

During their first year, postgraduate students are expected to participate in an appropriate study programme. For those interested in computational theory a specific course is provided (see Section 6). Students intending to pursue systems and other research may be directed to particular advanced courses, and are expected to participate in postgraduate study seminars. It is normally expected that, during their first year, students will focus their attention on one of the programmes of research being pursued within the Department. Some accommodation of interests is possible, but it is usually in the student's interest to be associated with one of the existing active research areas.

Prospective research students are encouraged to visit the Department, both to see research in progress and to discuss particular opportunities. A summary of current research work is provided in Section 7. A short note on the research interests of individual members of the Department is included in section 9.

Financial Support for Postgraduates

United Kingdom students may be eligible for Science Research Council Studentships, the value of which depends on age, experience and family circumstances. A first-class or upper second-class degree is usually required. The Department usually receives a quota of studentships each Spring to which students can be nominated.

The University has a small number of postgraduate awards open to a wide class of applicants including those from overseas. Competition for these is extremely keen. Details can be obtained by writing to the Secretary to the University, Old College, South Bridge, Edinburgh.

SECTION 1 INTRODUCTION

This handbook, summarising the work of the Department of Computer Science at Edinburgh University, is intended as an introduction to the Department for visitors, new students and staff.

The Department of Computer Science was created in 1966 when the then Computer Unit was sub-divided as a consequence of the Flowers Report on computing for the Universities and Research Councils. The Department inherited responsibility for teaching and research in Computer Science. There already existed a Postgraduate Diploma and a first-year course in the subject, together with a major research and development project, the Edinburgh Multi-Access Project, from which sprang the interactive computing system (EMAS) which provides the service now enjoyed by the University. The Flowers Report established major Computing Centres at Edinburgh, London and Manchester and the Edinburgh Regional Computing Centre was soon active in the role of providing service to the University and research institutes in the Edinburgh area.

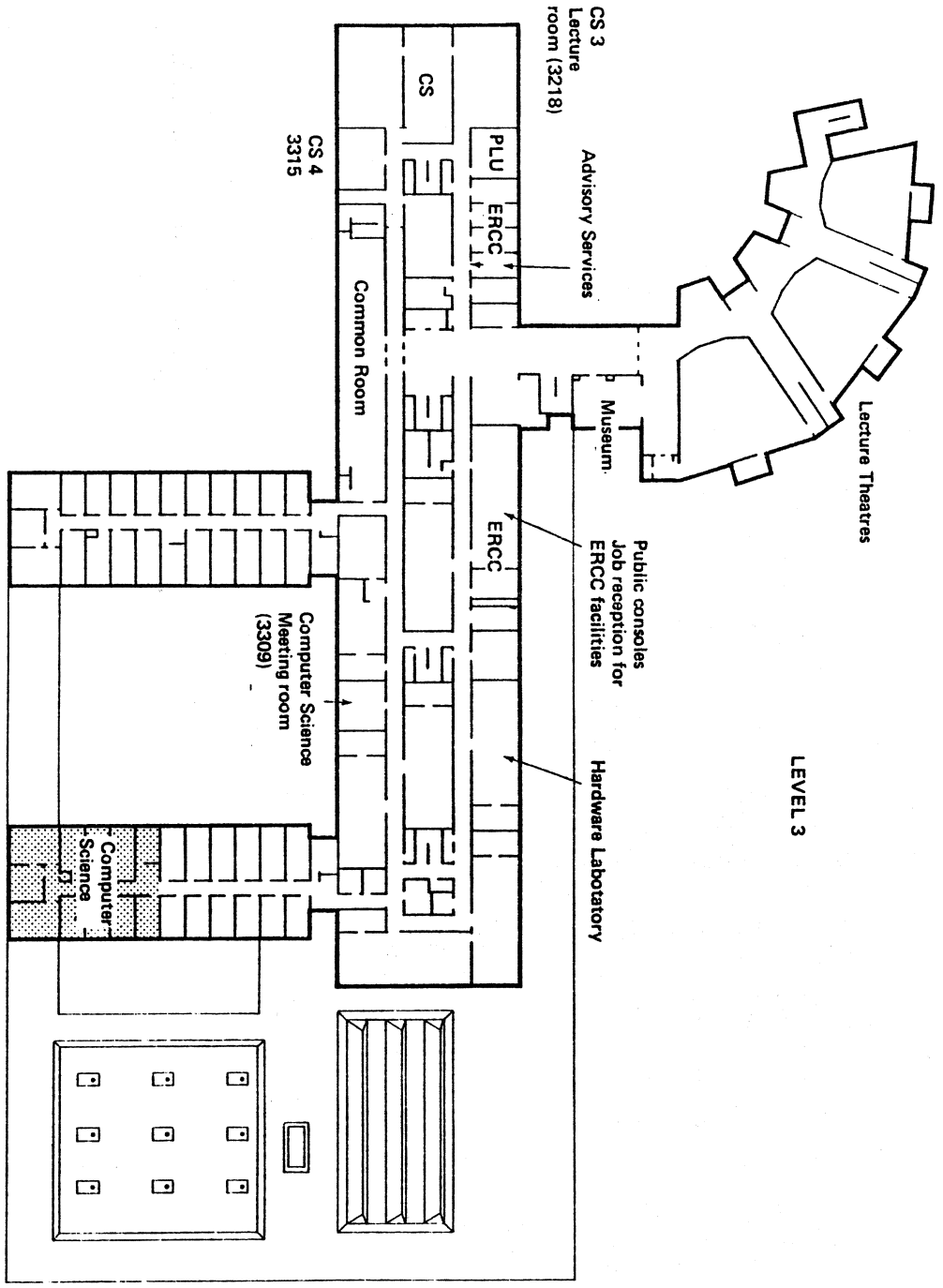
As funds became available, the Department purchased its own small computers and the number of staff was increased from the original half-dozen. Second and third year courses were introduced in 1968 and 1970 - the latter made possible by the co-operation of the Mathematics Department in setting up a joint Honours school. At this time the Department moved to new accommodation in the James Clerk Maxwell Building, thereby acquiring the laboratory space required for the kind of teaching that it regarded as central to courses in Computer Science. By 1972 it became possible to launch a Final Honours year in the subject and an Honours degree in Computer Science alone became available.

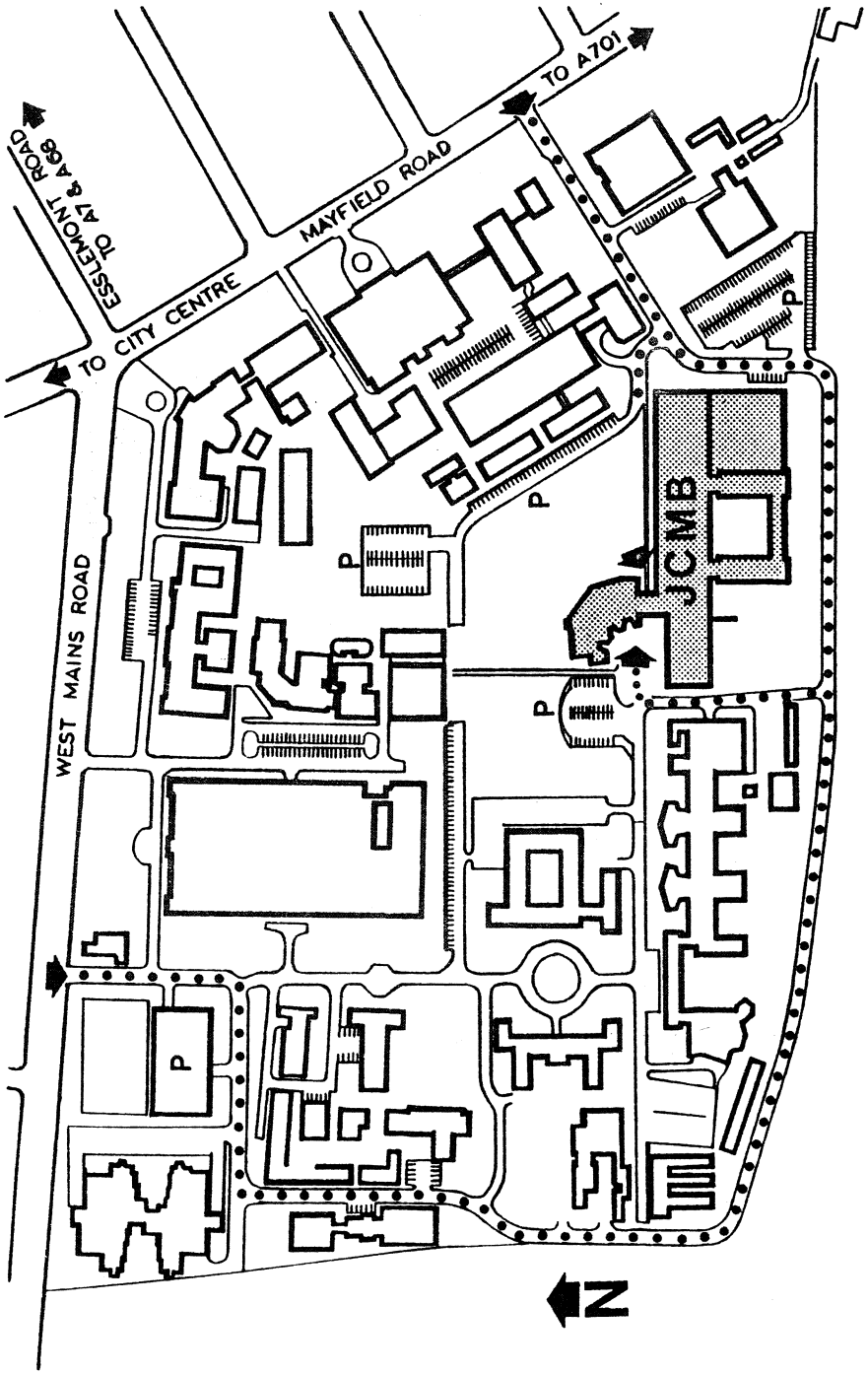
Since then, teaching effort has been devoted to improving the structure and content of the course, not least as a result of the feedback from the first contingents of students to go through it. The Department has been fortunate in the level of support it has been able to achieve and the laboratory facilities have been continuously upgraded.

On the research side, the project on multi-access systems came to an end and was largely replaced by work on software for mini-computers. Early work on Compter-Aided Design has now been succeeded by a group actively working on design methodology for Very Large Scale Integrated (VLSI) circuits. In the last few years, work in the field of Theory of Computation has expanded and this is now a major research focus. Other areas include distributed systems, communications, databases and micro-processor control.

The Department is housed in the James Clerk Maxwell Building at The King's Buildings, which is the Science campus at Edinburgh University situated about two miles from the City centre. Few would claim that the appearance of the buiding is worthy of Edinburgh's architectural tradition, but it does provide extensive and modern accommodation for the six departments which occupy it, including lecture theatres, machine halls, a library and common-rooms.

All the teaching for the second and subsequent years in Computer Science is carried out in the James Clerk Maxwell Building, but most of the first-year teaching takes place in the Appleton Tower in the central area.





UNIVERSITY OF EDINBURGH

COMPUTER SCIENCE DEPARTMENT HANDBOOK

Editors: Hamish Dewar
Jeff Tansley