SECTION 5 POST-GRADUATE MASTER OF SCIENCE COURSE

Course Co-ordinator: L.D. Smith.

5.1 Introduction

The MSc in Computer Systems Engineering and its sister course, the MSc in The Design and Manufacture of Microelectronic Systems, were established within the Faculty of Science at Edinburgh University during the 1978/79 academic session as part of collaboration between the Department of Computer Science and the Department of Electrical Engineering. Both courses are approved by the United Kingdom Science and Engineering Research Council (SERC) for the purposes of tenure of its advanced course studentships and both courses are supported by the SERC as part of its initiative in microelectronics.

5.2 MSc course structure

The MSc course consists of eight lecture modules, taught in the first half of the academic year and examined by conventional written examinations in April, followed by a full time project of six months duration examined by dissertation at the end of September. An oral examination, on any aspect of the course, may be required.

Under the modular scheme, a student is registered in either the Department of Computer Science for the degree of MSc in Computer Systems Engineering or the Department of Electrical Engineering for the degree of MSc in The Design and Manufacture of Microelectronic systems. In each case tutorial support, project supervision and the majority of lecture-based teaching is provided by that department. Up to about half of the lecture modules taken by a student may be taught by the other department.

Formally, the lecture modules taken by a student are prescribed by the head of the department in which the student is registered although in practice a considerable measure of choice is usually permitted. Lecture modules are chosen from a pool of about eight modules common to both MSc courses (section 5.3.1) and from sets of modules private to the particular MSc for which the student is registered. Those pertaining to the MSc in Computer Systems Engineering are described in section 5.3.2. Although the choice of modules is not completely free, most candidates can expect to choose eight from ten or twelve. Normally a candidate will not be permitted to study a module the content of which has already been substantially covered in the candidate's undergraduate work. This mainly affects candidates with a CS or EE background. Most lecture modules have practical work associated with them.

The central theme of the course described here is integrated circuit (IC) design. About half of the common modules (section 5.3.1) are explicitly concerned with IC design; it is this activity which enjoys direct support under the SERC's initiative in microelectronics. Overall, a broad spectrum of topics can be studied, from the direct construction of systems as very large scale integrated circuits to the implementation of conventional software-based systems in which the underlying computer is but a minor component. In its teaching of IC design the Department of Computer Science emphasises the application of VLSI technology in computer systems and the application of advanced computer-based design methods to IC design; advanced IC layout aids are

available on the Department's own DEC VAX11/780 computer system, accessible interactively from local design stations based on colour graphics display terminals. Students are encouraged to undertake significant projects involving the design, fabrication and testing of complete microelectronic systems. Suitably qualified applicants may be permitted to pursue project work in industry; this may be particularly attractive to potential applicants currently working in British industry.

Graduates from the MSc in Computer Systems Engineering will be particularly suited to employment in systems houses, in the computer industry, on the software side of companies manufacturing integrated circuits and anywhere where there is a need for knowlege of both sides of the hardware/software boundary.

5.3 Course modules

Section 5.3.1 decribes modules available to students registered for either the MSc in Computer Systems Engineering or the MSc in the Design and Manufacture of Microelectronic Systems; section 5.3.2 describes in general terms the scope of modules specific to the MSc in Computer Systems Engineering.

5.3.1 Modules common to both MSc courses

VLSI design (I and II) (two modules)

These modules, taught in the Department of Computer Science, examine the structured design methodology advocated by Mead and Conway 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 variety of integrated systems; it is performed using computer aids to structured design and the Department's own design stations, based on special-purpose colour graphics terminals. Designs are fabricated by the Edinburgh Microfabrication Facility and packaged chips are returned to students for functional testing.

LSIC design (I and II) (two modules)

These modules, taught in the Department of Electrical Engineering, examine the fundamentals of integrated circuit design and layout for both digital and analogue applications. A variety of approaches to layout are considered, inlcuding ULAs, standard cells and custom LSI. Practical work involves the use of the GAELIC computer aided IC layout system and SPICE, a circuit simulation package. Layouts are fabricated by the Edinburgh Microfabrication Facility and returned to students for testing, using sophisticated test equipment located in the Department of Electrical Engineering.

Production of Silicon Integrated Circuits

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, oxidation, diffusion, ion implantation, etc, to the evaluation of finished devices. Course work includes demonstrations of actual processing steps carried out in the Edinburgh Microfabrication Facility.

Design of 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. Some industrially used CAD systems are briefly considered and criteria for evaluating such systems are emphasised.

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 is performed on the Department's Motorola 6800 microprocessor development systems suported by the cross-assembly facilities provided by a PDP11/60 minicomputer.

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 such as Pascal. Emphasis is placed on language independent concepts and on choosing the right algorithm for the job. The design of data structures and the informal analysis of algorithms are also considered. Practical work is performed on the Department's own multi-access computer system and involves the development of a compiler for a machine-tool programming language.

student to experience some of its problems and rewards. With this intent, a high level programming language (a simple but usable subset of Pascal) is presented as a major theme of the term.

The term closes with a detailed examination of the application of computers to automated text analysis and document preparation, and looks at the potential impact of these and other developments in the office of the future.

The final term continues the applications theme, widening its scope to include health care, simulation and modelling techniques in sociology and economics, knowledge-based systems, the law, education and applications in the sciences. This year, geography has been selected as a particular example of the latter. The progress and future of man-machine speech communication is also reviewed.

4.4 Course Assessment

The overall assessment of performance on the course is based on a weighted average of marks achieved in (a) Practical work, (b) Essay assignments, and (c) Written examination. The practical work and essay assignments are carried out evenly throughout the course, so that a significant part of the overall assessment is conducted on a continuous basis.

There are class tests held at the end of terms 1 and 2. Approximately 60% of the course marks are associated with practicals and essays. A student achieving high marks in this continuous assessment and in the two class examinations may be awarded exemption from the June degree examination. A poor performance in the June examination may be recovered at the resit examination; a poor performance in the essay assignments may be compensated by writing additional essays in the summer vacation; but a satisfactory standard must be achieved in the practical work during the year, and there are no arrangements for retaking the practical work.
The course is not intended to lead to admission into later year Computer Science courses, although students who perform exceptionally well may be admitted to Computer Science 2.

systems, and a confidence in approaching others.

4.2 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. Thursday at 3 p.m. is the preferred time for tutorials. The practical work is carried out in the Micro-computer laboratory situated on the fourth floor of the Appleton Tower. This year the laboratory will be equipped with ten of the latest Sirius advanced personal computers to add to the existing complement of ten Apple II computers and twelve EMAS video terminals. Laboratory sessions are two hours per week and take place on Tuesdays and Wednesdays.

4.3 Syllabus

In the first five weeks of the course the terminology and technology of computer systems are introduced, with the aim of reducing the mystique which often surrounds them. At the same time, the student is introduced to some of the basic tools and skills of computer use. The technology theme is maintained throughout the first term but later the emphasis of the presentation shifts from the merely descriptive to an examination of the present and future impact of computer technology on society. In this context employment, the police, government and democracy have been chosen for close examination.

The second half of the term investigates the task of identifying problems and needs and developing systems as solutions to these problems. Problems in areas which are familiar to most people are therefore taken, identified more carefully, and some of the difficulties and methods of solution are discussed. A number of software packages for word-processing, graphics, statistics and equipment control are demonstrated in lectures and the student is invited to explore them further in the practical sessions.

In the first part of the second term, the nature of large information handling systems is examined. A progression is shown from established systems to those that are in the early stages of experiment. A number of working systems for information retrieval are featured and the design issues which particularly affect the user of such systems are high-lighted.

Having studied and experimented with a variety of packages and information systems, the student will be aware that their major component is software and that this presents difficulties in its design and construction. However, writing small programs is easy and frequently useful. Whilst it is beyond the scope and intention of the course that the student become fully proficient at programming, the course aims to provide an introduction to the craft and enable the

5.3.2 Modules available only to MSc students in Computer Science

The following list of modules is typical of that offered by the Department of Computer Science. In any given year, modules may be added to or deleted from this list according to the availability of staff and equipment.

Introduction to operating systems

This module examines the structure of typical modern multi-task multi-user operating systems such as the Edinburgh Multi-Acess System, DEC's VAX/VMS and IBM's OS/360. Practical work is performed on the Department's own minicomputer development systems. This allows students to test some of the ideas presented in the lectures by writing an operating system of their own.

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 CCITT X25 (level-2). This is performed on the Department's own multi-access computer system.

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 using the Department's own colour raster-scan graphics terminals.

Design of databases

This module examines recent advances in the design of databases and in the technology for implementing them. Particular emphasis is placed upon the relationships between applications programming languages and databases and upon techniques for integrating database concepts into typical high-level programming languages. Practical work involves implementing part of a database management system or part of a database application. This is performed by small teams using the Department's own computing facilities.

Modelling and system peformance

This module examines how to measure and predict the performance of computer systems. Material covered includes mathematical modelling of systems, simulation models, measuring the performance of systems, fitting models to measurements and performance prediction. Practical work involves the modelling and measurement of the Department's own computing systems.

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 required to attend this course, regardless of their previous programming experience.

5.4 Entrance requirements

A good honours degree, or its equivalent, is required. Some knowlege of the physical sciences, electrical engineering, mathematics or computer science is essential, though this need not necessarily be at degree level (relevant industrial experience is satisfactory). Some experience of programming is helpful, though not essential for otherwise suitably qualified applicants. Applications from persons currently working in British industry are encouraged; potential applicants should contact the Department to discuss their needs, qualifications and experience.

Applicants whose first language is not English are advised that a high level of competence in written and spoken English is a prerequisite for this MSc course. Overseas students are advised to contact the nearest British Council office and to request to take the British Council test of proficiency in English.

5.5 Diploma Course

Students accepted for either the MSc in Computer Systems Engineering or the MSc in Design and Manufacture of Microelectronic Systems will also be registered for the corresponding Diploma course. Students who achieve a satisfactory standard in the April examinations will be subsequently registered for the MSc degree. Students who fail to achieve a satisfactory standard in the April examinations may be subsequently registered for the appropriate Diploma course, or failed outright, as recommended by the Board of Examiners.

5.6 Part time study

Suitably qualified candidates, currently working in British Industry, may be permitted to study for the MSc in Computer Systems Engineering on a part-time basis over a maximum of three years. Subject to timetabling constraints, locally-based part-time candidates may pursue two lecture modules per term (which may require attendance for as little as two afternoons per week). Remote candidates are required to study full-time for one term (ten weeks) per year. A candidate whose progress is judged to be satisfactory may be permitted to pursue his project work at his sponsoring company, subject to satisfactory supervision arrangements being made.

SECTION 4 UNDERGRADUATE COURSE IN INFORMATION SYSTEMS

4.1 Introduction

Course Co-ordinator: M.P. Atkinson

The first-year Informations Systems course was 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 facilities are 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

Microprocessor-based Real Time Control Systems

Real time control
- Sampling Theory
- Linear Control Theory
- Stability of Linear Systems

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.

assessment: exam paper and practical exercise

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.

assessment: practical exercise

VLSI Circuit Design Exercise

This module is a practical exercise in designing a digital subsystem using the design techniques introduced in the VLSI compulsory module.

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

It is intended that suitable designs will be fabricated, by the Electrical Engineering Department, and it is hoped that processed slices will be available by the end of the third term so that they may be tested over the summer.

assessment: practical exercise

3.4.4 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.

During the year, students give short seminars and submit written reports on their progress.

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 150 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 150 in all, are given in the Autumn term and early Spring term, and are divided into three broad sections: Complexity, Program Methodology 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. Most of the formal teaching is finished 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)

Measures of complexity, complexity of problems and algorithms, design of efficient algorithms, proofs of lower bounds. Recursive splitting and recurrence relations. Comparison problems. NP-complete problems. Multiplication, convolution, discrete Fourier transform and pattern matching. Matrix algorithms.

Computational Complexity (18 lectures; CS4 option course)

Time and space as complexity measures.
Deterministic and nondeterministic computation.
Complexity classes and hierarchies.
Transformations between problems and use of oracles.
Complete problems in complexity classes such as NP, PSPACE and £P.
Provably hard problems, including cryptographic applications.
Heuristic algorithms to solve hard problems.
Randomised algorithms - Las Vegas and Monte Carlo methods.
Algorithms making use of parallel computation, eg FFT.

Algorithmic Graph Theory (18 lectures; CS4 option course)

Definitions and data structures for graphs. Eulerian circuits. Spanning trees. Depth first search in undirected and directed graphs: finding the non-seperable components of an undirected graph and the strongly-connected components of a directed graph. Network flow techniques and applications, Dinic's algorithm. Matchings in graphs.

Further topics in complexity (10 seminars)

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

Concurrency

Concurrent systems, where several independent but intercommunicating processes are active, occur in many areas of computer science. This course will discuss the problems which arise when designing such systems. Particular attention is given to the formalization of the communication requirements of the independent processes which combine to form these systems. A large part of the course is devoted to the exposition of a calculus of communicating systems, based on the notion of synchronised or "handshake" communication.
assessment: exam paper

Denotational Semantics

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.
assessment: exam paper

Group 3: Optional

Compiler Techniques

This module explores some of the issues involved in the production of a service compiler, with particular emphasis on factors which affect the speed of compilation, the quality of code produced, and the usefulness of the error reports.
The IMP compiler for the Motorola 68000 is taken as an illustrative example and students are invited to improve specific aspects of it.
assessment: exam paper and practical exercise

Computer Performance Evaluation

Measurement, monitors, modelling, statistical analysis, workload characterisation, performance prediction, tuning, program behaviour, case studies.
assessment: exam paper

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.
assessment: exam paper

Group 2: Theory

Computational Complexity

Time and space as complexity measures.
Deterministic and nondeterministic computation.
Complexity classes and hierarchies.
Transformations between problems and use of oracles.
Complete problems in complexity classes such as NP, PSPACE and £P.
Provably hard problems, including cryptographic applications.
Heuristic algorithms to solve hard problems.
Randomised algorithms - Las Vegas and Monte Carlo methods.
Algorithms making use of parallel computation, eg FFT.
assessment: exam paper

Algorithmic Graph Theory

Many real life problems to be tackled by computer scientists have natural graphical formulations; one could point, for example, to the problems of routing, scheduling and the design of reliable computer networks. The aim of this course 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.
The course will cover the following topics:
- Definitions and data structures for graphs.
- Eulerian circuits.
- Spanning trees.
- Depth first search in undirected and directed graphs: finding the non-separable components of an undirected graph and the strongly-connected components of a directed graph.
- Network flow techniques and applications, Dinic's algorithm.
- Matchings in graphs.

assessment: exam paper

6.3.2 Syllabus: Program Methodology

Program Design (6 CS3 lectures, 4 additional lectures)

Systematic design of programs starting with a specification. Specification of modules, data types and procedures. Examples of program design. ML will be used as the programming language.

Formal Tools for Program Development (8 lectures)

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

Program Logics and Automatic Deduction (8 lectures)

The unification algorithm and resolution principle in automatic proof. Natural deduction and its implementation. Programming logics (e.g. Pratt's dynamic logic, Constable's logic of programs), theory and application.

Further Topics

(a) Program Specification (8 lectures)

Purpose of specifications. Specification Languages. The algorithmic approach using initiality. Example specification. From specification to implementation.

(b) Prolog (2 lectures)

Prolog is based on first order logic used as a programming language. Application and advantages of Prolog will be discussed.

6.3.3 Syllabus: Semantics

Denotational Semantics (18 lectures; CS4 option course)

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.

Concurrency (18 lectures; CS4 option course)

Concurrent systems where several independent, but intercommunicating processes are active occur in many areas of computer science. This course will discuss the problems which arise when designing such systems. Particular attention is given to the formalization of the communication requirements of the independent processes which combine to form these systems. A large part of the course is devoted to the exposition of a calculus of communicating systems, based on the notion of synchronised or "handshake" communication.

Domains (12 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 (8 lectures)

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

Models of Parallelism (8 lectures)

An introduction to the various models of parallelism, including Petri nets and path expressions; their theory and application.

Further Topic

(a) Operational Semantics (4 lectures)

Semantics as inference. Relationships between operational and denotational semantics.

3.4 COMPUTER SCIENCE 4 Course Co-ordinator: I. B. Hansen

3.4.1 Aim of Course

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

- Study subjects outwith the core curriculum in previous years or study to a greater depth those taken before.
- Produce an honours thesis based on a major implementation or research project.

3.4.2 Assessment

The student must attend the following:

- Very Large Scale Integration
- Digital Communications and Computer Networks
- three optional courses which must include at least one theory course

These 5 courses provide half the assessment in Computer Science 4. The other half comes from the student's honours thesis.

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 both taken into account. For joint Honours degrees the examinations in both departments taken during the third and fourth years are taken into account.

3.4.3 Course Units

Most fourth year course units last for nine weeks each at a rate of two meetings a week.

Group 1: Compulsory

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.
assessment: exam paper and practical exercise

3.3.3 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 completing two composite practicals students will gain experience in deciding how to:

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

The practicals are:

Term 1: Implementation of an Operating System

The laboratory equipment for this course 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.

Term 2: Exercise in Computer Structures

Students write a microprogram interpreter to span the gap between the "bare" hardware and the machine language level. Possible languages to implement include M6800 machine code, Interdata 70 machine code, and a simplified version of a graphics protocol for implementing basic graphics operations on a 1024x1024 framestore.

The kit consists of a microcode-driven data path, a framestore and an interface for downloading the microcode and performing all the necessary I/O operations.

3.3.4 Assessment

The degree examination for the Honours course 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.

SECTION 7 RESEARCH PROJECTS

7.1 VLSI Systems

Very Large Scale Integration (VLSI) is a technology which provides a quantitative leap in the complexity of circuits that can be manufactured with consequent gains in cost-effectiveness. These quantitative changes have also induced qualitative changes which make VLSI currently a very active research area. Many complex systems architectures including highly concurrent systems can be implemented directly in silicon that hitherto have been unrealistic even to contemplate. VLSI research provides a focus of activity for several disciplines within the department. Work in progress covers investigations into systems architectures, design methodologies, computer design aids, simulation and testing aids, formal modelling and verification, complexity and so on.

With the support that was provided to the department in respect of the MSc course in Computer Systems Engineering by the SERC, the department possesses very good facilities for designing VLSI circuits and implementing the aids that enable design work to progress. A number of high resolution colour graphics design stations are available together with colour pen-plotters to produce hard copy. It is also intended to make use of other developments taking place in the department towards Advanced Personal Machines and high performance graphics systems to produce even more sophisticated VLSI design systems in the near future.

7.2 Programming Methodology

This project, carried out by Rod Burstall, 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, Bell Labs, NJ); it is strongly typed and is being extended with modularity features permitting multiple representations. 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".

7.3 Data Curator

This project, which is funded by the SERC, 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 developed a number of basic persistence handling systems using algorithms similar to those of Challis. It has implemented an extended version of S-Algol, known as PS-Algol, which incorporates persistence in a simple manner. Experiments suggest higher rates of programming result, and we are currently investigating improved implementation, the appropriate programming environment for persistence, and using the system for a number of large data modelling and CAD projects to assess our approach. We are seeking funding for more work, both to develop and exploit the existing language, and to explore new languages adapted to the needs of long term persistence.

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 under the direction of Robin Milner, 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).

Operating Systems

Review of batch processing system programs, their components, operating characteristics, user services and their limitations.
Implementation 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.

Large Scale Systems Design

A review of large systems with some case studies and general applications. The problems associated with the design and implementation of such systems. Data storage and access methods, query languages, language embedding, the relational and network models. Some specific data base management systems.

Computer Systems Modelling

Application of probability to the description of 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 of computer systems, measurement, statistical analysis.

Computability

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, Undecidability, halting and other problems, recursive and recursively enumerable sets.
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.

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. Reducabilities 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).

3.3 COMPUTER SCIENCE 3 — Course Co-ordinator: D. Rees

3.3.1 Introduction

The third year provides a basic foundation in the topics of 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.

3.3.2 Course Modules

Computer Structures (I)

Detailed anatomy of computers, function and structure related to various sub-system components. Methods of describing computers.
Microprogramming. Horizontal and vertical micro-code control methods.
Gate level primitives and logic design. Various implementation methods including VLSI.
Computer arithmetic.
I/O mechanisms and bus organization.

Computer Structures (II)

High level languages and their relation to computer architecture. The choice of instruction sets and addressing mechanisms.
Asynchrony. Fundamental problems of arbitration. The time domain and asychronous control of computer systems. Self-timed systems.
The computer as an embedded component - some case studies. The use of micro computers in systems.
Bit sliced implementations and special purpose components.
The influence of technology - from transistors to gate arrays. Cost and performance evaluation.

Programming Methodology

The ML functional programming language: further concepts, including definition and use of abstract data types. Correctness proofs for functional programs.
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.
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; separate compilation; exception handling.

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 SERC 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, Gordon Plotkin and Matthew Hennessy, 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 Computational Complexity and Algorithms

This research, carried out by Gordon Brebner, Mark Jerrum, Les Valiant and Carl Sturtivant, studies the fundamental limits on the resources such as time and space used by computations. Topics explored include a unified algebraic theory of the complexity of algebraic and combinatorial problems, general purpose parallel computers, algorithms with good probabilistic behaviour, the size of basic hardware structures, and upper and lower bounds on problem complexity.

7.9 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 disputed wills.

3.2.2 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 use of 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.

Data Structures and Programming.

Brief revision of PASCAL programming.

Recursively defined data types (e.g. lists and trees); their representation in the applicative language ML, in PASCAL and in IMP. Reasoning about recursive functions. Simple structural induction.

Lists, trees and graphs; algorithms relating to these structures.

Expansion of the searching and sorting techniques introduced in CS1; examination of time and space requirements.

Introduction to a graphical output package (Edwin).

Compilers.

Overall organisation of a compiler. Parsing algorithms. Symbol tables. Further examination of the data structures required. Simple code generation techniques. Run-time storage allocation. Bootstrapping. Cross-compiling.

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.

3.2.3 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 September.

3.2 COMPUTER SCIENCE 2 — 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 a study of further programming languages, including an applicative language. They examine some abstract data structures, such as sets and trees, which can be used in many different contexts to model the external world, and represent these in different languages. They also start to examine simple methods of proving that algorithms compute correct results. Also during the second year, students learn about the problems of the definition and translation of programming languages - work that will be resumed in the fourth year course. They undertake an exercise on translating a simple high-level programming language into code for the microprocessor selected for study in the lectures on computer systems.

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.

3.2.1 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.

Students can choose to take the 2A or the 2B course. 2A is the prerequisite for entry into Computer Science 3. 2B students attend most of the same lectures, but undertake a lighter load of practical work.

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 KL-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 4 Megabytes of main store and 1200 Megabytes of disk storage. It supports a maximum of around 40 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 provide hands-on experience and perform specialised tasks. These machines include a Terak, 14 Interdata 70 series machines, and three 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.

Work is in progress on bringing into service an Ethernet-type communication network, connecting together a large number of advanced small systems, designed in the department. The first batch of these machines is expected to become available during the year. It is intended to use them, both as a vehicle for hardware experimentation and as powerful personal work-stations. Some of them will be equipped with a fast, high-resolution graphics processor.

SECTION 9 MEMBERS OF STAFF

9.1 Teaching Staff

Sidney Michaelson — Professor of Computer Science.
The study of literary style with special reference to problems of authorship and chronology.
Queue-related models of computing systems.

Rod Burstall — Professor of Computer Science.
Programming methodology: correctness proofs, program transformation, specification languages. Semantics using an algebraic/categorical approach.

Peter Schofield — Senior Lecturer and Head of Dept.
Programming techniques. Data structures.

Malcolm Atkinson — Lecturer.
Relationships between programming languages and database systems. Development of techniques for the design and construction of large-scale software.

Gordon Brebner — Demonstrator.
Complexity theory. Parallel computation.
Geographically distributed computer networks.

Irene Buchanan — Lecturer (part-time).
Computer-aided design and VLSI.

Rosemary Candlin — Lecturer.
Introductory teaching and specialised micro-processor systems.

Hamish Dewar — Lecturer.
Small machine architecture. Text processing.
Speech input/output.

John Gray — Lecturer (part-time).
Computer-aided design and VLSI.

Igor Hansen — Lecturer.
Microcoded hardware description and analysis. Computer system architecture. Microprogrammable processors for graphics, communications and high level languages. Multiprocessor systems.

Matthew Hennessy — Lecturer.
The mathematical semantics of programming languages and the design of program-oriented logical proof systems.

3.1.5 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 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.

The main topic of the first half of the course, occupying about 24 lectures over 8 or 9 weeks now begins; it is an introduction to programming through the medium of the language Pascal. Almost every feature of the language is covered in lectures and practical work. From the outset great importance is attached to the production of well written (structured) programs, by the exposition of suitable examples and by criticism of programs produced by the students.

The conclusion of this series of lectures marks the half way stage in the course, by which time the students should have acquired a basic competence in programming. The lectures of the second half are largely devoted to general programming topics, approached from a more abstract level; the practical continues to consist of writing Pascal programs.

In parallel with the programming topics the start of the second half of the course includes about 5 lectures on computer organisation, to fill in some of the background to the system and ideas the students have been using. The characteristics of the principal hardware and software components of a typical computer system are described (storage, processors, order codes, file systems, etc.)

The main topic of the second half of the course is given the title Programming Methods, and about 24 lectures are devoted to it. Under this heading various strategies for the solution of problems are explained and illustrated, also a number of common programming techniques are introduced. Some frequently used data types are described as abstract entities with a variety of possible realisations; the types include arithmetic values, enumerated types, sets, queues, stacks and dictionaries; realised using bit-strings, arrays, linked lists, trees etc. The problems used for illustration include searching, sorting, generation of permutations, string matching, back-tracking techniques and dynamic programming.

Towards the end of the course a series of about 4 lectures are given on the historical development of programming languages, designed to demonstrate the increasing level of abstraction involved and, at the same time, to give students a flavour of some other languages.

3.1.4 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 3 paragraphs of the previous section.

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

Mark Jerrum	Lecturer. Complexity of computation: combinatorial algorithms, algebraic models of computation.
Robin Milner	Reader. Semantics of programming languages. Applications of mathematical logic to formalise the statement and proof of assertions concerning programs. Abstract models of concurrent computation.
Gordon Plotkin	Reader. (on leave of absence Oct 82 - Feb 83) The denotational semantics of programming languages with emphasis on concurrency. Computational and inductive logic.
David Rees	Senior Lecturer. VLSI design and associated design tools. Design and implementation of multi-access operating systems.
Frank Stacey	Lecturer. Systems software.
Lee Smith	Lecturer. Design and simulation of digital systems. Computer system architecture. Integrated circuit design.
Jeff Tansley	Lecturer. Switching theory. Descriptive methods in the design and construction of digital systems. Digital methods in signal and image processing.
Les Valiant	Reader. (on leave of absence until Dec 82) Computational complexity, analysis of algorithms, automata theory, combinatorial mathematics, parallel computation.
Alec Wight	Lecturer. Computer performance evaluation. Remote terminal emulation. Workload characterisation.

Visitors

P. Mosses	(Aarhus) Jan 83 - Dec 83 Analytical Semantics of Concurrency.

9.2 Computing officers

Gordon Hughes	CAD tools. Portable graphics systems.
Kathy Humphry	(part-time). First-year teaching. Filestore management.
Fred King	Semi-conductor stores. Hardware design using micro-processors.
Rob Procter	Micro-computer systems. New technology for graphical devices. Social impact of computers.
Rainer Thonnes	Software for communications and micro-processors.
Douglas Tudhope	(joint with Royal Observatory). Image processing. Computer Graphics.
Allan Vernon	VAX management.

9.3 Secretarial staff

Heather Carlin	(part-time).
Kate Duncan	Secretary to Professor Michaelson.
Alison Fleming	(part-time).
Eleanor Kerse	Secretary to Professor Burstall.
Dorothy McKie	(part-time).
Gina Temple	(part-time).

9.4 Technical staff

John Dow	Maintenance. Communications.
Jimmy Johnstone	Information Systems Laboratory. Video interfaces.
Peter Lindsay	Graphical devices. Link interfaces.
Ian Thomson	Equipment construction. Communications.
Ian Mathers	Junior technician.

3.1 COMPUTER SCIENCE 1 — Course Co-ordinator: R. Milner

3.1.1 Introduction

The aims of this first year 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 a significant part of the final assessment.

3.1.2 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.

Students are assigned to a small tutorial group which meets for one hour at about fortnightly intervals, the frequency depending on the nature of the course work going on at the time. About two thirds of these tutorial groups meet in the Appleton tower on either Tuesday or Friday afternoons. The remaining third meet in the James Clerk Maxwell building, most of them on Monday afternoon. In those weeks when there are no tutorial group meetings, students are expected to attend a larger one hour problem class during which they work under supervision on the current practical assignment or some closely related topic.

In addition to these time-tabled occasions students are expected to spend about 10 hours a week on private study and the current practical assignment (programming).

3.1.3 Syllabus

During the first few weeks of the course the lectures and practicals are largely devoted to an introduction to some data description and manipulation languages. Methods of defining and operating on a (large) body of related data are presented in sufficient depth for the student to be able to form complex queries on such a database available during the practical sessions.

Concurrently with this topic students are taught the necessary minimum of features of the computer system which is used throughout the course. This involves learning how to manage his/her own files and how to use a text editor.

own merits, but as there are more applicants than places, a good standard of performance in Higher or A-level examinations has become 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.

3.0.2 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 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.

3.0.3 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.

9.5 Research staff

Ken Chisholm	Data Curator. Programming languages and Databases. Computer-Aided Design.
Paul Cockshott	Data Curator. Programming languages and Databases. Message-based languages.
Gerardo Costa	Semantics of Concurrent Computation. Semantics of programming languages. Concurrency. Non-determinism.
George Milne	Formal Models for the Description of Circuit Behaviour.
Andrew Morton	Stylistic Analysis. The study of literary style in relation to questions of authorship and chronology.
Alan Mycroft	Applicative Languages. Semantics and optimisation of applicative and related languages.
David Rydeheard	Programming Methodology. Applications of category theory to programming. Specification languages.
Donald Sannella	Programming Methodology. Program specification.
Dave Schmidt	Programming Methodology. Application of denotational semantics and category theory to the design and analysis of programming languages and systems.
John Scott	Programming Methodology. Abstract data types.
Mike Shields	Mathematical Techniques in Telecomms. Mathematical models, languages and proof techniques for concurrent computation.
Mike Smyth	Abstract Data Types. Algebraic approach to data types. Computability in a general setting. Some aspects of concurrency.
Colin Stirling	Semantics of Concurrent Computation. Concurrency. Modal logics.

9.6 Full-time Postgraduate students on Ph.D./M.Phil. course

Neil Bergmann	Silicon compilation.
Ilaria Castellani	
Rocco De Nicola	Semantics of Processes.
Pedro Hepp	Common query evaluator for relational databases.
Thomas Horton	
Krishna Kulkarni	VLSI design.
Mark Millington	Theory of computation.
Flemming Nielson	Semantics of concurrency and flow analysis.
Gabriel Owoso	Programming languages and data bases.
Alberto Pettorossi	Applicative languages. Concurrency.
Kuchi Prasad	
Hanne Riis	Proof of correctness and complexity of programs.
Brian Ritchie	
George Ross	Concurrent access to large-scale databases.
Allen Stoughton	
Carl Sturtivant	Complexity of algorithms.
Niklas Traub	

SECTION 3 UNDERGRADUATE COURSES IN COMPUTER SCIENCE

3.0 General

Director of Studies: R. Candlin
A. Wight

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.

3.0.1 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

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 may be obtained by writing to the Secretary to the University, Old College, South Bridge, Edinburgh.

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.

Postgraduate fees covering tuition and practical work have recently been increased by the Government to approximately £4000 per annum, but United Kingdom and EEC students pay approximately £1800; some special grants are available to cover the difference for outstanding overseas students. The British Council also offer studentships to overseas students.

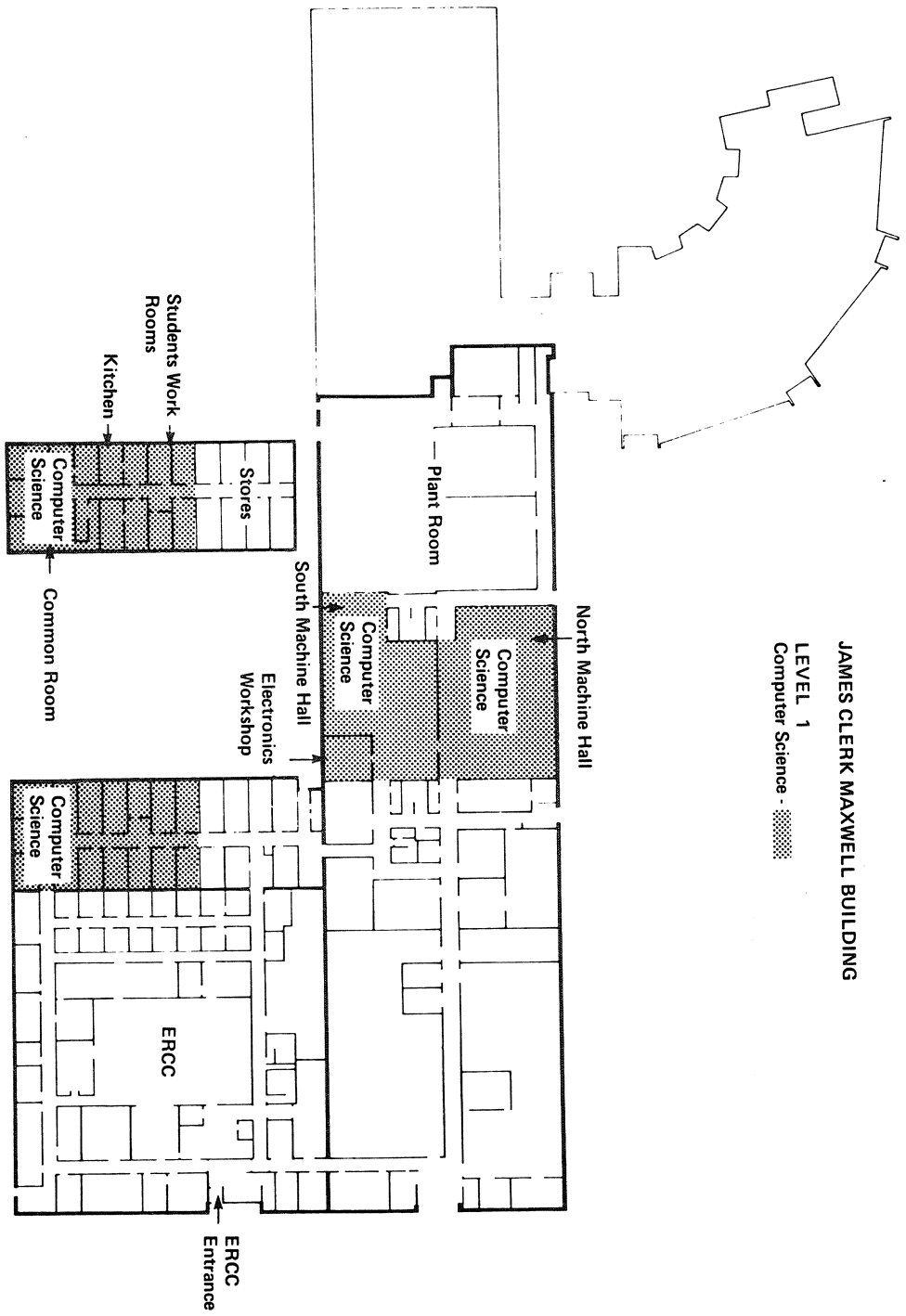
Postgraduate admissions

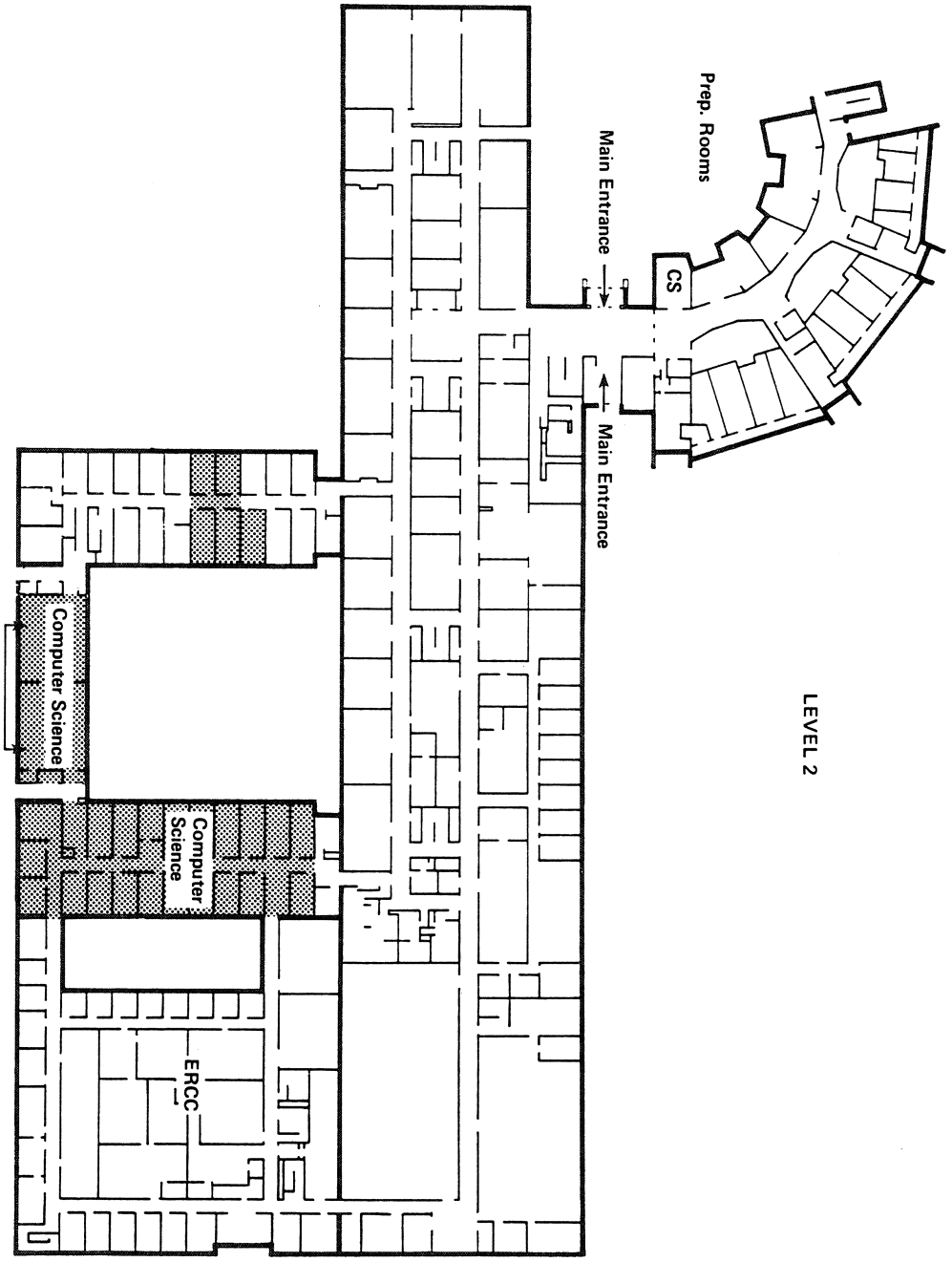
Admissions to Research degrees are handled by Prof. R. Burstall and to the M.Sc. course by J. Tansley. The Department's address is:

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 an Honours course in Computer Science over four years (described in Section 3) and a first-year course in Information Systems (described in Section 4). The latter course is 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 Computer Science course 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.

Undergraduate Admissions

Candidates for undergraduate degrees must apply through the Universities Central Council on Admissions. The UCCA Handbook - 'How to Apply for Admission to a University' - and an application form may be obtained from schools or from:

The Secretary,
UCCA,
PO Box 28,
CHELTENHAM, GL50 1HY.

Further enquiries about admissions should be addressed to:

Faculty of Science Office (Admissions),
University of Edinburgh,
West Mains Road,
EDINBURGH, EH9 3JY.

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.

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.

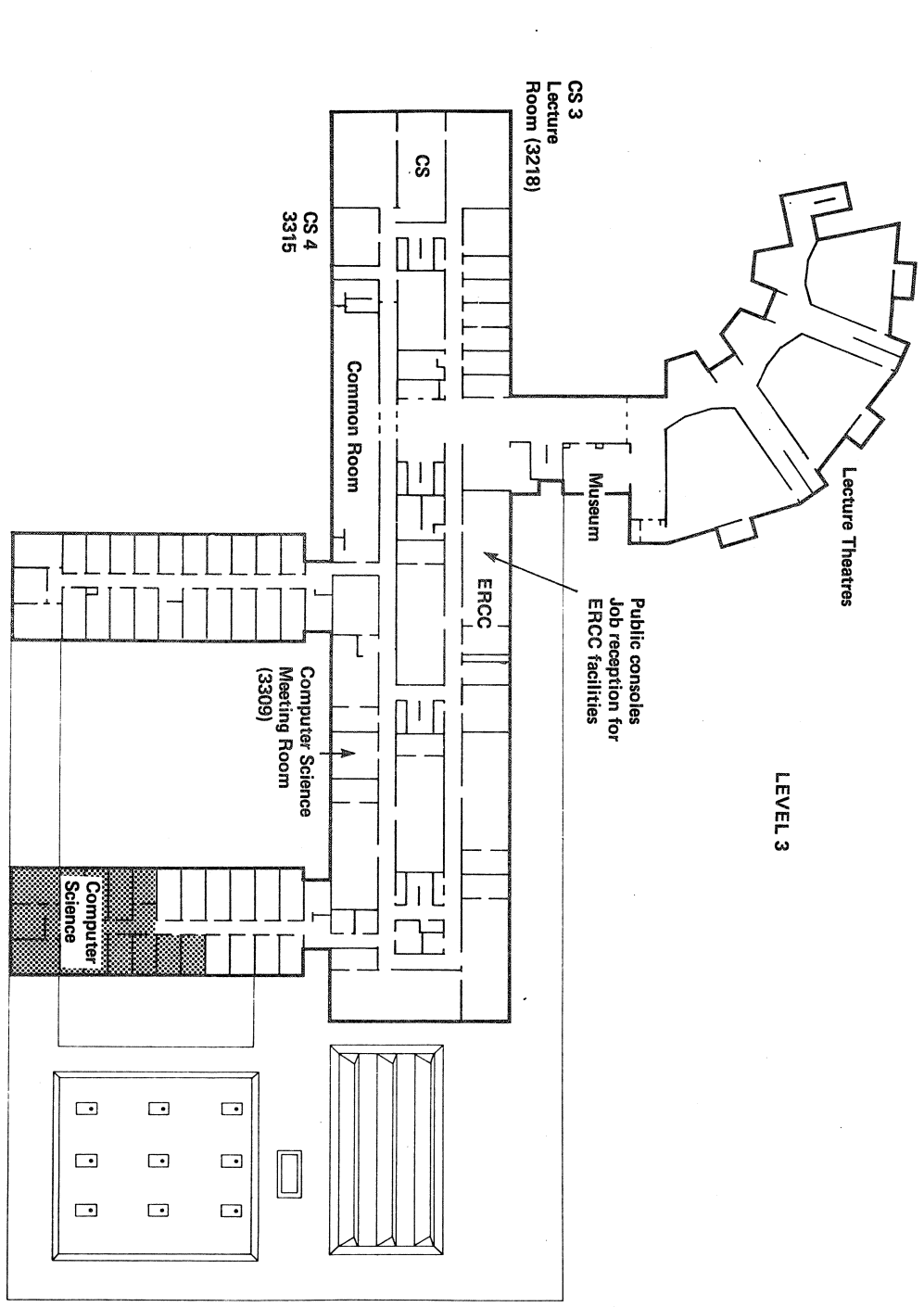
As funds became available, the Department purchased its own small computers and the number of staff was increased from the original half-dozen. Second and third year courses were introduced in 1968 and 1970 - the latter made possible by the co-operation of the Mathematics Department in setting up a joint 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.

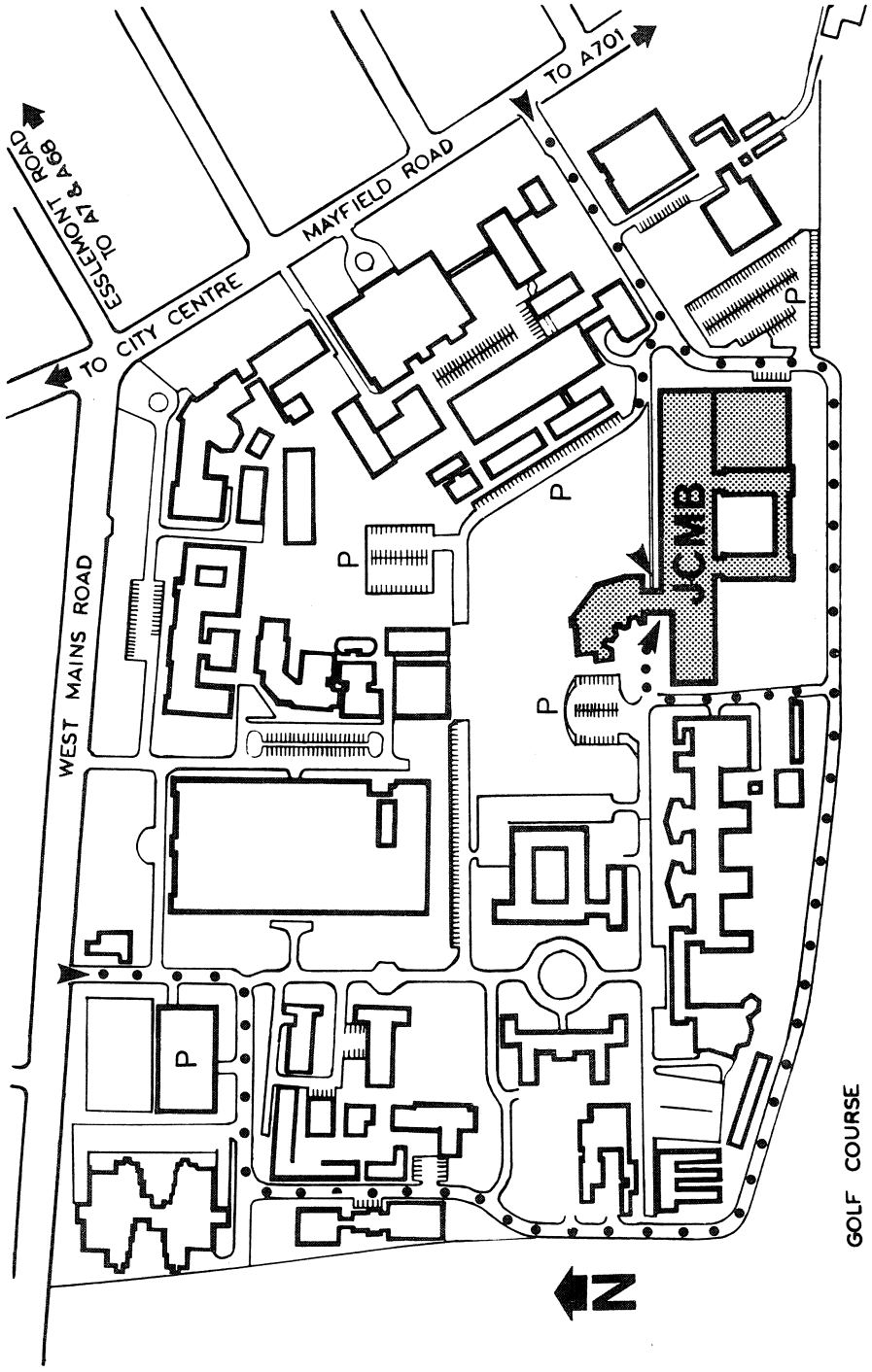
Since then, teaching effort has been devoted to improving the structure and content of the course, not least as a result of the feedback from the first contingents of students to go through it. The Department has been fortunate in the level of support it has been able to achieve and the laboratory facilities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

DEPARTMENT OF COMPUTER SCIENCE

HANDBOOK 1982-1983

UNIVERSITY OF EDINBURGH