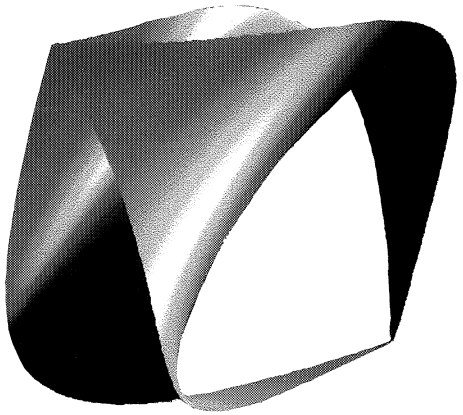
An aerial view of the Kings Buildings site.
Computer Science is housed in the James Clerk Maxwell building (to rear of centre of photograph, overlooking the golf course).



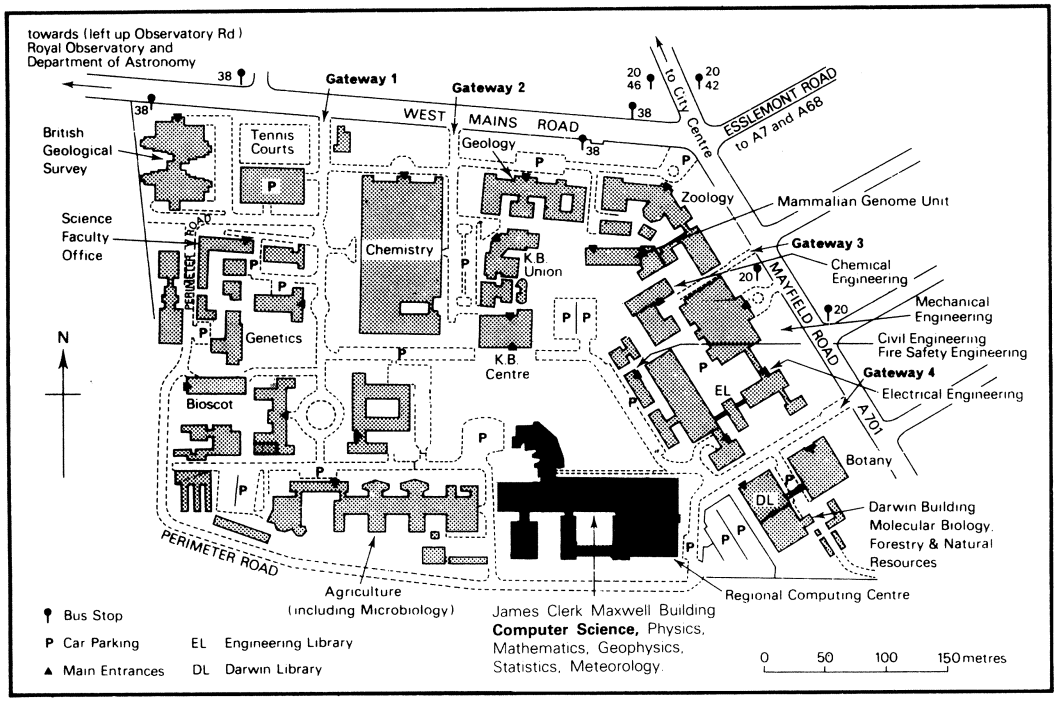
1987-8

University of Edinburgh

Department of Computer Science

DEPARTMENTAL HANDBOOK

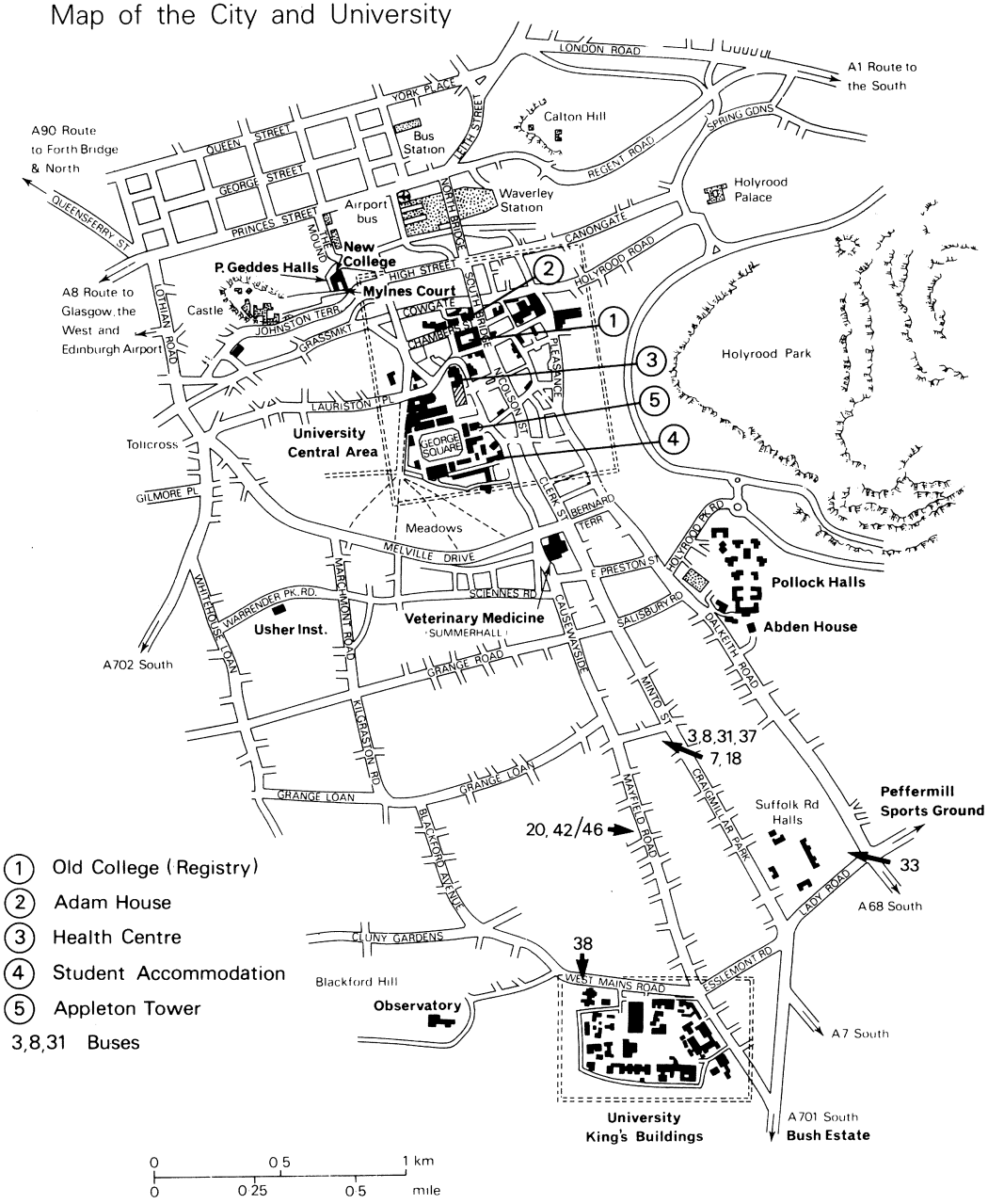
UNIVERSITY OF EDINBURGH
Map of Kings Buildings



See inside rear cover for map of the University within the city.

The front cover inset was produced by a Bezier surface editor from a plane surface by moving the corner control points on a coarse grid. This was then reduced to a finer grid for the plot shown. The image was produced on a departmental Advanced Personal Machine (APM) as part of a final year project.

UNIVERSITY OF EDINBURGH
Map of the City and University



1. Old College (Registry)
2. Adam House
3. Health Centre
4. Student Accommodation
5. Appleton Tower

3,8,31 Buses

See inside front cover for detailed map of the Kings Buildings site.

Research into novel (single processor) architectures is currently centred on the design and implementation of a sparse vector processing system, with support from SERC and High Level Hardware Ltd. High Level Hardware are the manufacturers of the Orion computer, a user microcodable system which is being used in this project to emulate the instruction set of the proposed sparse processor. The project also involves a considerable amount of work on compilers and on language extensions to give the programmer access to the sparse vector support mechanisms provided in hardware. The applicability of these mechanisms to other types of computing is also being investigated.

Concurrent Architectures

Research in this area is currently directed towards the design and implementation of highly efficient and scalable general-purpose systems. The primary aims of the research are to minimise the problems of communication overhead and algorithm-dependence, and to produce architectures for VLSI implementation.

The first of these systems (EPP1) is based on the concept of 'Context Flow' architecture. This is an implementation technique in which the program evaluation algorithm defines a cyclic graph structure, through which parallel contexts flow. This has several advantages over the Data Flow concept, and can be used to implement many different evaluation strategies.

A second system (EPP2) is also planned, based on work originally carried out at the University of Manchester. In this system a linear set of processors is interconnected by a common communication highway and interprocessor communication occurs by means of global broadcasts controlled in hardware by a data driven synchronisation mechanism.

Compilers

Edinburgh has a tradition of involvement in the areas of compiler and language development. Although most activity is currently based in the LFCS, centred on developments around the applicative language ML, two new projects are under active consideration by the Systems Group and these will be developed over the next two to three years. The first is concerned with building a set of compiler construction tools and researching the possibilities for automatic compiler generation. This is an area where the 'hand crafting' approach has generally been used in the past in Edinburgh. The second area is in designing a graphical programming tool. This is a spinoff from the Alvey performance project. Some consideration is also being given to the design of a new object oriented programming language.

Parallel Processing

One of the main problems in parallel processing is the mapping of the problem to be solved onto the underlying architecture. A number of projects are currently under way in this area, involving members of several of the research groups in the Department.

Sparsely-connected Parallel Computers

Many parallel algorithms are expressed in terms of an 'idealistic' parallel computer consisting of a collection of processors and a shared memory; such a model allows any pair of processors to communicate in one time step. This research considers 'realistic' parallel computers which have a collection of processors with local memory, connected together by a sparse network. Early work concentrated on the study of (i) efficient algorithms to simulate the idealistic computer on realistic computers with several interconnection schemes, and (ii) the feasibility of laying out various sparse networks in VLSI. Current work is concerned with (i) the relationship between the memory usage of idealistic algorithms and interconnection schemes, and (ii) (motivated by wafer-scale integration) dealing with processor or network failures on realistic computers.

Algorithmic Skeletons for Parallel Architectures

The project involves the specification of an 'idealistic' machine, which would enable users to map their problems (not necessarily obviously parallel) efficiently and conveniently on to a highly parallel architecture like a square grid of processor-memory pairs.

The idealistic machine is implemented as a set of algorithmic skeletons, corresponding to a number of problem-solving methodologies, which have a predefined implementation on the grid, and which can be fleshed out by the user with his own procedures. To date, three such skeletons have been considered, for problems involving recursive divide and conquer, task queues, and optimal iterative combination.

Parallel Parsing

A (systolic) array algorithm that allows recognition of general context-free languages has been extended to allow it to do parsing as well. The main idea behind the extension is the addition of counters to the processors in the array. These counters serve to set pointers to other processors during the recognition phase. Afterward, these pointers are used to reconfigure the array into a tree of processors representing the parse tree. Once the array is reconfigured, the parse of the

input string can be output. work is proceeding on error handling, characterisation of efficient context free grammars for the array parser, design of other parallel parsing algorithms, attributed parse tree evaluation.

Performance Modelling

The performance of computer systems and communication networks has been extensively studied for many years. Although considerable advances have been made suitable performance modelling tools are not generally available for systems designers and architects or analysts studying working systems. There is currently renewed interest in providing such tools because of three parallel developments. Firstly the emergence of practical distributed and parallel computer systems is generating new classes of problem, which are not solvable by existing analytical and numerical methods and are driving the search for new solution methods and more importantly focussing attention again on simulation programming. Secondly the arrival of powerful and inexpensive workstations with good graphics capabilities makes it worthwhile to explore the construction of novel tools to support simulation and analysis. Thirdly the formalisation and automation of the design process for software and computer systems, through Integrated Project Support Environment (IPSE) development is creating a new role for performance tools and exposing new problems of data sharing and verification of results.

The performance group at Edinburgh is linked through the Computer Performance Workshop, which it helped to found, to workers both in industry and in universities.

Currently, its major effort is in the Alvey funded project 'Improved methods for performance modelling', where Edinburgh is a collaborator with STC/ICL. This is investigating a design for an Integrated Modelling Support Environment (IMSE) and includes prototyping novel tools to run within such an environment. The design is intended to be readily extensible. Edinburgh is involved in the conceptual modelling of such an IMSE and in the prototyping of graphics tools both for the construction of process based discrete event simulations and for the support of experiments using such models.

A second stage of the IMSE project, to produce a complete working system and to integrate it more closely with the design process, is now getting underway. This is considering the introduction of shared data with IPSEs and the integration of further modelling techniques.

Modelling communications protocols is another major area of activity and some of this work has industrial funding.

ganisation which has equipment supplied by a variety of manufacturers, and uses a variety of systems on that equipment.

Within the Department the network will contain a number of different different technological components including Ethernet, Centrenet (a high performance LAN) and a contention switch. Among the processors will be a sparse vector processing computer and a number of parallel processing systems as well as conventional machines acting as personal workstations and dedicated servers (e.g. for cross-compilation).

The work consists of defining an abstract model for the computing environment, based upon the notion of allowing computational resources to access an idealised world by mapping members of a heterogeneous collection of objects into their own computational environment, and then producing protocols which can be used to implement the model efficiently and reliably on heterogeneous networks connecting heterogeneous systems.

Distributed Systems

Closely related to the work on EUNICORN is on-going work on the distributed system of APMs in the Department. The APMs have been used as discless workstations sharing a common filestore for a number of years, but a number of deficiencies have been apparent for some time. Work is currently in progress to replace the existing filestore mechanisms with a more sophisticated system which will offer both increased performance and functionality. The new system will facilitate the use of a number of file access and transport protocols in parallel, allowing users full access all their files from whichever computers/operating system they happen to be using.

Other work in this area involves an investigation of a more radical approach to distributed systems in which techniques developed for automata games are used to the control the actvities of a set of cooperating processors.

Novel Architectures

One of the advantages of an Integrated Computing Resource Network such as EUNICORN is that if offers the possibility of attaching a variety of processors to an existing infrastructure with minimal overhead. This allows the posibility of investigating a range of novel architectures, an activity which will become increasingly important in the world at large as facilities for the generation and implementation of VLSI design improve and become more cost-effective. Research into novel architectures (including concurrent architectures) will therefore be an on-going activity at Edinburgh.

Algorithms and Data Structures

In contrast to algebraic complexity, the problems here are combinatorial, and the model of computation is machine-based. The problems considered come from many areas, including graph theory, combinatorial enumeration and geometry. A particular concern at Edinburgh has been the study of *randomised* algorithms whose progress is influenced by the outcome of a sequence of coin tosses. Is it possible that randomised algorithms may have greater computational power that deterministic ones? Surprisingly the answer appears to be 'yes', and several probabilistic algorithms have been discovered which are faster than the best known deterministic counterparts.

This research 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, and upper and lower bounds on problem complexity.

Computer Systems

The Department has been active in computer systems design for many years and has expertise in operating systems, language support and hardware design and implementation, particularly in relation to local area networks, microprogrammed controllers and high-performance personal workstations. Within the Department there are now over sixty Advanced Personal Machines (APMs) which provide substantial local computing capability and use an Ethernet to gain access to network services such as filing and printing. A variety of new projects are now emerging which carry forward from this experience.

The Edinburgh Integrated Computing Resource Network

The widespread introduction of computer networks has generally resulted in the provision of distributed computing environments which can be classified as either (a) heterogeneous systems connected by explicit mechanisms, which are typically associated with wide area networks, and (b) homogeneous systems connected by implicit mechanisms, which are typically associated with local area networks. The aim of the Edinburgh University Integrated Computing Resource Network (EUNICORN) project is to investigate the communications infrastructure for a computing environment distributed over a local area and composed of heterogeneous systems as in case (a), but offering an integrated interface to the user as usually provided in case (b). This will harness the advantages of local area networking in order to supply sophisticated functionality to users within an or-

Stylistic Analysis

Stylometry has been defined as the scientific study of the usage of words in an attempt to resolve literary problems of authorship and chronology. The traditional methods of stylistic analysis have often been based upon subjective evaluation of internal textual evidence, often with unsatisfactory results. With the availability of computers as tools, new ideas have burgeoned and new approaches to these age-old problems have led to an increased need for statistical analysis of observational data. For example, it has now become practicable to study the usage of the most common words (such as determiners, conjunctions and prepositions) in a text: because of their number, such function-words have previously been neglected by literary scholars. Study of the positions of such words within the sentence has revealed that authors can be distinguished in this way. A new approach to problems of authorship and chronology is being brought into being by the study of such habits of composition.

Before any new technique is applied to disputed texts, it must be tested on texts of known provenance - as with other observational sciences. A wide range of texts is available for such testing, as well as an ever-growing collection of unanswered questions ranging from the composition of the Iliad to disputed wills in the U.S.A. Current work is concentrating on the thorny problems of authorship of Elizabethan and Jacobean drama and on the development of techniques for investigating poetry as well as prose.

This research into the methodology of answering such questions necessarily involves the development of further techniques and associated software. The amounts of data to be handled are often large and the processing poses interesting problems for the computer scientist.

Theory

The Department has played a strong part in the development of the semantic theory of computation in the last decade or so. It has had considerable assistance from SERC, which has allowed an intensive approach augmented with work on implementation of the research results - e.g. programming languages or reasoning tools. The projects described in this section are integrated into a whole in the Laboratory for Foundations of Computer Science, whose structure and role is described elsewhere in this Handbook.

Computer-Assisted Formal Reasoning

The department has worked for twelve years on the problem of doing proofs interactively with a computer, largely supported by SERC grants. The LCF system

which was designed by the department in 1974-6 introduced a notion of tactic or strategy which has become widely known. The idea is that a user communicates with the machine not only by asking it to embark on a certain problem, and to seek assistance when it needs to; the user also communicates guidance to the machine in the form of tactics or strategies. The means of expression for these strategies is a functional programming language ML - invented for this purpose but now widely used as a general-purpose language.

Recently, inspired by work at Cornell University, a new ingredient *Proof-editing* has been added to the methods of communication. This makes doing a logical proof with machine assistance rather like editing a program using a Structure Editor; in both cases the machine has the job of maintaining the logical correctness (or syntactic correctness) of the object concerned. It has been found that attribute grammars are a powerful method of implementation for such systems; it is also expected that the combination of Proof-editing and tactics will improve the Proof interface considerably.

There remains the serious problem that humans do rigorous proofs in many different notations or logics, and would like the machine to cooperate in this style. Until recently it has been unclear how to program a system which will interact in a uniform way but in whatever logical system the user requires. But recent work in both logic and computation theory promises solutions to this problem; thus work is progressing towards a truly versatile reasoning tool. This will be tested on applications of interest in Computer Science, such as program correctness. It should be remarked that work at Edinburgh in this area has led to the mechanical verification of a microcomputer design, by RSRE at Malvern.

Computational Category Theory

Category is that branch of mathematics which deals with concepts common to algebra and topology. For example homomomorphism in algebra and continuous map in topology are clearly analogous; in category theory they are subsumed by the concept *morphism* (or *arrow*). Similarly the product of algebas is analogous to the product of topological spaces, and these are subsumed in the categorical notion of product. Category theory is helpful in computation theory, particularly as a paradigm for suggesting appropriate definitions. At Edinburgh work in this area has sought to connect category theory more directly to programming by writing code in a functional programming language (ML) for many of the constructions of category theory, for example finding the limit of a finite diagram. Also a categorical programming language has been designed whose sole definitional mechanism is adjunction. It can mimic the main features of traditional functional programming languages including definition of data types and lazy

RESEARCH IN COMPUTER SCIENCE

Computer Science is a subject which is less easily compartmentalised than most other scientific and engineering disciplines, and so although researchers in the Department form themselves into groups such as the Complexity Group, the Systems Group, the Theory Group and the VLSI Group, there is considerable cross representation, and the research of any one group may well draw on the experience and expertise of others. Research in computer systems, for example, involves a number of interrelated activities, including work on architectures, networks, languages, etc., and the use of VLSI, no one of which can be carried out independently of all the others. The division of research into the areas described here does not therefore represent a strict compartmentalisation of activities within the Department. It shows, rather, the major emphases of one or more projects contributing to the overall programme of computer science research.

Computational Complexity

Computational Complexity is the quantitative study of the 'inherent difficulty' of solving computational problems. The study is motivated by the empirical observation that the resources required to accomplish various computational tasks vary dramatically between tasks. The meaning of 'resource' depends on the setting, but typical examples are time, space or hardware size. The study is wide-ranging, and its techniques rest on increasingly sophisticated mathematics.

Algebraic Complexity

Algebraic Complexity studies certain intrinsic requirements in the computation of algebraic functions, such as matrix multiplication, polynomial evaluation and interpolation. A machine independent model is used whose basic operations are addition, subtraction, multiplication and division (sometimes extended or even contracted). Most work assumes a single processor, although parallelism is also studied.

The most highly developed area is concerned with bilinear forms. However, despite substantial progress, non-linear lower bounds are still elusive. For example, the best known lower bound for matrix multiplication is linear and has seen no improvement for at least ten years (all the advances in this much studied topic have been on upper bounds).

The only general technique for non-linear lower bounds has its basis in Algebraic Geometry. This has yielded optimal results for such problems as polynomial interpolation, but in the case of bilinear forms it is of no help. There is a pressing need for more techniques—especially ones which can handle bilinear problems.

data types.

Specifications

Specification of the problem is the first step in any systematic method of program development. For precision the specification may be written in a formal mathematical notation. This gives the possibility of developing a proof in formal logic that a program is a correct implementation of the specification. Work here has centred on specification languages with particular attention to those which make it possible to write modular specifications and to develop modular prorammes from these by successive refinement. The semantics of specification languages has been studied using tools based on category theory, notably the concept of *institution.* One approach has been to extend the language ML, which already has good modularity features to a non-executable specification language.

Semantics

There are two senses of the term 'Semantics' in relation to computation. The narrower sense is concerned with the mathematical presentation of the meaning of a programming language, and then with deducing properties of the language. The wider sense is concerned with computational models in general, independently of language. Work in Edinburgh is carried out in both senses. Starting more or less from Scott's theory of domains, specific results have been obtained about the semantics of the lambda-calculus (and hence of a wide class of languages), in particular about the structure of function spaces (which are essential to the understanding of procedures in, say, PASCAL). The well-known powerdomain construction, important for non-determinism, was discovered by researchers now in the department; this contributed in particular to the research on Concurrency which is described below.

More recently, increasing emphasis has been place on the semantic approach known as Structured Operational Semantics. The Department has been a main proponent of this method, both for sequential and for concurrent computational models. Much work has been done on its relationship with both denotational (e.g. domain-based) and logical semantic methods. It has also played a strong part in the research on type systems; here, a key theorem which is of vital practical importance asserts that a program which is correctly typed (according to the type thaory) is immune from the occurrence of certain errors at run-time.

Concurrency and Communication

Research on the theory of concurrent computation has been actively pursued for about ten years. The work has centred upon finding mathematical models which allow tractable proof methods for practical systems but which are also conceptually uncluttered - i.e. they require few basic concepts. This led in 1980 to the Calculus of Communicating Systems, a model with a strong algebraic character based upon structured operational semantics.

This work is widely known, and has led to many recent developments. Some these are under study at Edinburgh. One particularly hard problem is to find proof systems for composite concurrent systems which are truly modular - i.e. they allow proofs about subsystems to be composed into a proof about the whole system. Another topic is the study of the various equivalence relations over concurrent processes; it is not yet understood when to demand a strong form of equivalence and when a weaker form will suffice. The expressive power of languages for concurrency is also under investigation.

As an integral part of this theoretical work, the research group is building a 'Concurrency Workbench'. This is to be a tool for assisting reasoning about concurrent systems; it will assist not only in applications but in theoretical research as well, and of course in teaching. It is hoped to gain assistance in building a graphical interface for this workbench.

The Calculus of Communicating Systems may be used to give the semantics of many concurrent programming languages. The group is also developing a language based directly on the calculus, using ML for the sequential constructs. The extension of this work to a distributed implementation is currently under investigation.

Functional Programming

The theoretical group has a continuing concern with functional programming. This interest was originally stimulated by the fact that functional programming is useful for work both in artificial intelligence and in computer-assisted proof; more recently its usefulness has broadened, and the design and semantics of such languages has become a research topic of general importance.

The first in the Edinburgh sequence of functional languages was POP2, developed in the late sixties in the AI department (where some members of the theory group were then working). Then followed ML, whose purpose was to drive the LCF system, a program for computer-assisted reasoning. The next in the sequence was HOPE, which arose mainly from a concern to integrate the way in which abstract data-types are used in programs with the way people reason about them. ML and HOPE have recently been united in Standard ML, which

RESEARCH IN COMPUTER SCIENCE

It is possible to study for the M.Sc. degree on a part time basis over a period of three years.

Modules offered in 1987/88:

Computer Science: VLSI Design I & II; CAD Tools for VLSI; Digital Communications; Programming Techniques and Software Tools; Introduction to Operating Systems; Computer Systems Modelling; Database Systems; Advanced Graphics; Computer Architectures; Computer Architectures for Parallel Processing.

Artificial Intelligence: Programming in Prolog; Programming in Lisp; Representation and Inference I & II; Natural Language Processing; Machine Vision; Intelligent Assembly Systems; Intelligent Sensing and Control.

Electrical Engineering: Large Scale Integrated Circuit Design 1 & 2.

also has a powerful modules facility to help in structuring large programs. As with its predecessors, Standard ML is the group's medium for almost all the software which grows from its theoretical research. Besides implementing and using functional languages, the group is active in formal semantic methods applied to them.

In order to support a large research group, using a wide variety of machines, a portable compiler for ML has been developed. The compiler is written in ML and produces code for a functional abstract machine, which is then interpreted. The compiler also serves as a test-bed for experimenting with extensions to the ML language.

Applied Non-Standard Logics

There is increasing interest in theoretical computer science in studying logics whose interpretations, or models, are computational structures. One very general model, invented by Kleene, is the realizability interpretation. This was extended to various models of the languages of arithmetic and set theory. The resulting logics are non-standard in that they fail the law of excluded middle. Another general area of study is modal and temporal logics whose models are computation trees or histories. Such logics are appropriate for reasoning about programs in terms of their behaviour throughout execution.

Program Development in Type Theory

The various type theories of Per Martin-Löf seem to provide appropriate environments for program development. The identification of propositions and types provides a powerful specification language while the notion of proof objects as programs identifies the process of proof with program development. The group is investigating the requirements for a programmer's assistant based on type theory. Currently the use of Type Theory in specific application areas, e.g. user interface design, is being examined. The intention is to incorporate the experience thus gained into the design of a programmer's assistant. Such an assistant will be based on a generalised system of tactics.

Very Large Scale Integration

The two main themes of activity are firstly Compter Aided Design (CAD) tools and secondly Formal Verification techniques.

Computer Aided Design

The overall aim of this work is Silicon Compilation. That is, systems which accept high-level descriptions of either the structure or the required behaviour of circuits and from them synthesize complete designs ready for fabrication. Considerable progress has already been made in this direction. An early silicon compiler, implemented in cooperation with the Electrical Engineering Department, was the so-called 'FIRST' system. This was aimed at bit-serial signal processing applications and has been a notable success. More recently, a silicon compiler named 'U2', aimed at nucleonic instrumentation applications, has been developed in cooperation with the UK Atomic Energy Authority. Other silicon compilation research in progress concerns structures with a built-in-test capability. The focus of future work will lie in the problem of synthesizing designs from a purely behavioural description rather than from a structural description. This is a much more difficult task but progress has been made in the first stage, that of 'silicon pre-compilation'.

At an intermediate level of the design process, a textual language based composition system has been developed in which the floorplanning and layout of chips is specified in such a way that composition can be verified for correctness. The task of automatic floorplanning is a further area of activity as is that of automatic wire-routing methods. Novel VLSI architectures are also receiving attention. Work is in progress on array structures with general purpose logic elements. Both dynamically configurable and static mask-programmable configurations are under investigation.

Formal Verification of VLSI Hardware

Hardware verification is a research area of increasing importance due to the technological advances of VLSI fabrication and the resulting complexity of potential designs. Techniques are being developed which allow the correctness of a circuit design to be established by mathematical proof prior to fabrication. Central to this formal approach to circuit validation is a behavioural model in which to naturally and accurately represent the inherent concurrency of circuit behaviour.

Current research, funded by the Alvey Directorate(Software Engineering) explores the modelling capabilities of CIRCAL, a model of concurrent systems, both hardware and software. This has resulted in the discovery of a number of powerful validation techniques which allow both verification by proof and validation by simulation to be performed in a way which reflects the constructive, modelling philosophy adopted.

New VLSI design languages are required to support the design of verifiably correct devices using both manual and automatic design techniques. Such languages

POSTGRADUATE STUDY

Facilities are offered in the Department of Computer Science for postgraduate study by research (towards the degrees of M.Phil. and Ph.D.) and for study towards the degree of M.Sc. in Information Technology: Computer Systems Engineering.

Degrees by Research

Students can undertake full-time, supervised research leading towards the degrees of Master of Philosophy (minimum study period 21 months; normally 2 years) and Doctor of Philosophy (minimum period 33 months; normally 3 years). Candidates are registered, for the first year of study, as a 'Supervised Postgraduate Student' prior to transfer to the appropriate degree course. (The minimum study periods noted above include this year spent as a Supervised Postgraduate Student.)

During the first year, postgraduate students are expected to participate in an appropriate study programme. For those students registered for research in the computational theory area, a specific course is provided. Students intending to pursue research into Computer Systems or any other non-theoretical topic may be directed to particular advanced courses and are expected to attend postgraduate study seminars. Research interests in the Department of Computer Science are outlined in pp 25-37.

M.Sc. in Information Technology: Computer Systems Engineering

The School of Information Technology in the University of Edinburgh offers a 12 month program of postgraduate study leading to the M.Sc. in Information Technology. Each of the constituent departments of the School (Computer Science, Artificial Intelligence and Electrical Engineering) offers its own degree, with a student based in one department able to take courses offered by the other departments. The Computer Science Department offers the M.Sc. in Computer Systems Engineering.

The course runs from October to September. During the first half of the academic year, eight lecture modules are studied and examined from a total of twenty-one, at least four of which must be Computer Science modules. The following six months are spent on a full-time project, examined by dissertation, usually supervised within the department. Most lecture modules have practical work associated with them.

should permit the expression of behaviour as well as structure with the design process being viewed as the iterative replacement of behaviour by structure. A hierarchical design methodology is adopted with every design step requiring to be shown to be correct; that is the design meets its specification.

Vision

Computer vision research is concentrated on the theme of the '$2\frac{1}{2}$D sketch' - generation of descriptions of visible surfaces from stereo images. The aim is that 3D vision processes should be able to recognise objects by matching stored object-models to such surface descriptions. This work forms part of a large multi-site Alvey project in AI/IKBS vision. The project involves both industrial and academic collaborators and is concerned with investigating the interpretation by computer of stereo and moving images. Expected applications include automatic assembly and vehicle guidance systems.

POSTGRADUATE STUDY

INFORMATION SYSTEMS 1

The aim of this course is to promote skills in the use of Information Technology and knowledge of its applications, both present and future. The course is designed to cater for the needs of the non-specialist. There are no prerequisites, students from any faculty, in any year and with any background (apart from Computer Science) are welcome.

Assessment is by examination and course work. The latter includes extensive computer based practical work through which students have the opportunity to develop skills and use and evaluate a representative range of computer applications. The course is organised as two half courses – IS1Ah and IS1Bh.

Topics include:

IS1Ah : Technologies of Information Systems – Hardware and Software; Principles of Computer Based Problem Solving – Elements of Software Design, Programming in PROLOG.

IS1Bh : Major Applications – Large Scale Systems, Information Retrieval, Design Aids, Office Automation; Social Impact; Future Trends.

Practical exercises on EMAS and Apple Macintoshes are designed to back up the lectures.

LABORATORY FOR FOUNDATIONS OF COMPUTER SCIENCE

Computer Science 480
(Conceptual Modelling)

Conceptual Modelling is concerned with representing knowledge and capturing the meaning of data. In the Artificial Intelligence community, this is usually termed *Knowledge Engineering* whilst in the Database community it is termed *Database Semantics.* Conceptual Modelling is also of importance in the worlds of Psychology, Philosophy and Linguistics; the course will draw on work in these areas in studying the requirements and application of conceptual models. The subject is also related to the work of the Programming Language community concerned with data abstraction – the development of conceptual languages – and programming methodologies.

Topics include: Data Abstractions, Conceptual Models, Conceptual Languages, Applications.

Computer Science 490
(Computational Complexity)

Machine based complexity theory involves studying the resources, primarily time and space, required to solve problems using computers. The course examines the classification of problems in terms of their difficulty when solved on one particular computational model, the Turing Machine, and indicates that this classification is also applicable to other models. Several completeness classes which group together 'equivalently difficult' problems are described. Some techniques for producing good, but not perfect, solutions to hard problems are introduced.

Topics include: Time and Space as Complexity Measures; Complexity Characteristics of Turing Machines; Complexity Classes – Containments and Hierarchies; Efficient Transformation of One Problem into Another; Complete Problems in the Classes NP, PSPACE, and #P; Provably Hard Problems; Easy-To-Solve Restructions of Hard Problems; Approximation Algorithms; Randomised Algorithms.

Participants are encouraged to do pencil and paper exercises related to the lecture material.

Computer Science 461
(Theory of Communicating Systems)

An introduction to the calculus of communicating systems giving both the theoretical foundation and examples of practical application.

Topics include: Discussion of Models of Communication, The Language CCS and its Operational Semantics, Observational Equivalence and Congruence, The Notions of Confluent and Determinate Behaviour.

Practical work will involve two or three assignments for which no computing facilities will be required.

Computer Science 470
(Advanced Graphics)

The course aims to teach fundamental techniques for generating line drawings and realistic shaded images of three dimensional scenes and extends this to include 'ray tracing' techniques, colour and texture.

Topics include: Algorithms for Display, Line Drawings, Curves and Surfaces, Body Modelling, Shading, Ray Tracing, Colour.

There are practical exercises to illustrate the principles learnt in the course. These are done using APMs with graphics and the Gould.

LABORATORY FOR FOUNDATIONS OF COMPUTER SCIENCE

The groups which now form the Laboratory (LFCS) have a long record of seminal research into the Foundations of Computer Science. This has included both research into the fundamental theory and, in amplification of this, construction of prototype systems.

This work has exerted a strong influence on the development of Computation Theory particularly in the fields of semantics, specification, machine assisted proof and implementations (both of languages and of proof systems). It is naturally the software products which have been most visible, in particular the functional programming language Standard ML, which is now in use in many locations around the world both in educational and in research and development environments. Methodologies have also been exported; examples are the Specification Language CLEAR and the algebraic Calculus of Communicating Systems.

Against this light the Laboratory was formed in January 1986 with the twin aims of intensifying the research and establishing formal links with Industry to ensure the application of this work in practical environments. Significant funding from the Science and Engineering Research Council, the Alvey Directorate and Industry has been awarded to establish the Laboratory and strengthen its research programme. After its initial period of operation it is approaching its planned size and is generally accepted as one of the leading centres in the world in Computer Science.

LFCS Staff

The Laboratory currently has about 45 members, some categories of these are detailed below.

Academic staff:

Prof. Robin Milner	Director
Prof. Rod Burstall	Co-Director
Prof. Gordon Plotkin	Co-Director
Mr. Stuart Anderson	
Dr. Kevin Mitchell	
Dr. Charles McCarty	
Dr. Donald Sannella	
Dr. Colin Stirling	

Support staff:

Dr. George Cleland	Assistant Director
(to be appointed shortly)	Systems Development Manager
Mr. Hugh Stabler	Systems Programmer

Dr. Cleland is the Assistant Director of the Laboratory and is responsible for management of the Laboratory's infrastructure and for developing its Programme of Industrial Interaction.

The Laboratory has been fortunate in being able to attract first-rate postdoctoral research fellows from all over the world:

Research Staff:

Dr. Arnon Avron	Dr. Tatsuya Hagino	Dr. Robert Harper
Dr. Furio Honsell	Dr. Lin Hui-Min	Dr. Ian Mason
Dr. Eugenio Moggi	Dr. Joachim Parrow	Dr. Nick Rothwell
Dr. Paul Taylor	Dr. David Walker	

There are also some twenty research students (also from many countries) working on predominantly theoretical research topics.

LFCS Research Programme

The individual areas of research are described elsewhere in this document but it is worth noting that they unite into a large and coherent research effort. This effort has a core of theoretical research and a practical component which explores application and implementation of the theory.

The practical part of the research is typified by constructing tools for proving properties of concurrent systems or for constructing and analysing specifications of complex systems. These systems or their descendants will typically be used by software engineers and will provide methods which allow rigorous proof of properties of the resulting system. These tools must support reasoning in a variety of mathematical logics, and almost half of the Laboratory's resource is committed to projects concerned with either building the interface mechanisms for constructing proofs or tackling the mathematical problems in integrating a number of logics into a single framework.

Computer Science 450
(Computer Systems Performance Evaluation)

The course discusses the problems of measuring, predicting and improving the performance of computer systems. The topics for discussion are tools and techniques which can be applied to projected and working systems. Particular emphasis is laid on measurement, workload characterisation and capacity management.

Topics include: Operational Analysis, Mean Value Analysis, Approximate Solution Modelling Techniques, Workload Characterisation, Measurement and Monitoring, Tuning, Program Behaviour, Capacity Planning, Computer Networks.

Practical work involves exercises and a case study.

Computer Science 460
(Digital Communications)

Until recently, computing and telecommunications were viewed as distinct subject areas. Now, with the advent of sophisticated computer networks, the two have merged to give us 'Information Technology'. The course examines the techniques used to implement networking starting from transmission of data bits along physical connections and building up to the distribution of computations over many processors. Both local and wide area networks are covered.

Topics include: The ISO Model of Communication, Interfacing with Hardware, Simple Information Theory, Error Detection and Correction, Link Control Protocols, Local Area Networks, Wide Area Networks and Data Routing Techniques, Transport Protocols, Data Transformations such as Compression and Cryptography, Distributed Computation, Social Implications of Networking.

Practical work involves an exercise implementing the X.25 link control protocol on an APM thus enabling communication with a second machine running a proven implementation.

Computer Science 430 (Compiler Techniques)

This course examines the creation of executing computer programs on typical hardware/operating system combinations. An understanding of the context, both hardware and software, within which programs execute is seen as central to the construction of compilers. The approach is to consider the solution of problems associated with representative languages on representative hardwares. The course is practically based and should give valuable insights into the nature of programs as well as an understanding of compilers.

Topics include: Language Issues, Hardware Architectures, Operating Systems, Parsing Techniques and Grammars, Code Generation, Portability, Global Optimisation, Runtime, Compile Time.

Practical work involves three exercises including two programming problems and one essay.

Computer Science 440 (Denotational Semantics)

Denotational Semantics is a powerful method for defining realistic programming languages and for specifying the meaning or actions of programs. It is also open to mathematical analysis and gives results in the theory of computation.

Topics include: Abstract Syntax; Sets and Semantic Domains; The Denotational Format; Definitions using Stores; Dealing with Recursion and Iteration; Environments, Procedures, Functions and Other Compound Data Types; Continuations.

There is no compulsory practical work but weekly exercises are strongly recommended.

Investigation into Programming Methodology has resulted in the functional programming language Standard ML, which is now used for most of the Laboratory's programming. Work in this area continues with formalising the semantics of this and other languages along with further language development both of ML and of the kernel language Pebble.

LFCS Computing Facility

The Laboratory has been fortunate in acquiring from a number of sources a powerful and flexible facility to support its systems-based activities. It is based upon a mixture of Sun, Vax and Pyramid equipment to provide the computational intensive resource. This is augmented by a number of Macintosh systems and a small number of IBM PC machines used for general purpose work. The facility to date has cost about £750,000.

The central file server is a Pyramid 98XE system with 2 Gb of disk store. This runs Sun's NFS system to provide central filestore facilities for the Sun workstations: these consist of 4 fileservers and and about 30 workstations including 2/120s (2), 3/50s (13), 3/73s (8), 3/110 (1), 3/160s (1) and 3/260s (4). This is backed up by a Vax/Unix system which allows us to maintain Vax code compatibility.

The facility allows the Laboratory to put a workstation on everyone's desk – a Sun of the appropriate power for those who need it and a Macintosh for those with lesser computational requirements.

Virtually all the Laboratory's development is based upon a window management concept. The use of MIT's X-windows system along with Sun's Network File System is allowing the Laboratory to build a range of applications and support environments which are portable, but still provide a good user interface.

Industrial Interaction

The element of the Laboratory which distinguishes it from merely being a collection of research projects is the formal structures which it uses to interact with practitioners. This takes place in a number of ways:

The **Affiliation Programme** provides, to Industrial R & D groups, a low cost means of keeping in touch with both the work of the Laboratory and with other significant developments through means such as: distributing copies of all LFCS

reports (these include both research reports and expository publications); an annual "Update" meeting; access to the Laboratory's prototype systems; informal discussion of affiliates "problems"; etc.

The Laboratory's **Course Programme** is now strongly developed with about a dozen course planned over the next 18 months. These range from practical courses in Functional Programming or Interactive Theorem Proving to those on theoretical topics such as Type Theory, Domain Theory and Denotational Semantics, all of which are now seen as essential in understanding the structure and behaviour of systems.

Industrial Visitors are encouraged to spend periods of time in the Laboratory. These would typically be be industrial researchers intent on transferring the current technology of the Laboratory back to their parent company. It is expected that they will both study the work of the Laboratory in breadth and also apply the theories and methods of the Laboratory to particular problems. Besides the benefit to their own work that this will bring it will also provide valuable feedback to the research of the Laboratory.

Some **Collaborative Projects** are in place and other will follow. These allow Industry and the Laboratory to work jointly on a project – the Laboratory providing a rich environment and a wide range of ability and industrial partner providing the "real world" application and the commercial incentive to use it. The result of this kind of project is rapid transfer of skills and systems to the industrial environment and feedback to the Laboratory on the usefulness and the applicability of its work. These effects are similar to those of the Industrial Visitors scheme, but in greater depth.

The Laboratory publishes a report series comprising both research and expository material. Copies of these and other publications such as the Annual Report and the Prospectus can be obtain from George Cleland, the Assistant Director, from whom further general information on the Laboratory may be obtained.

Computer Science 411
(VLSI Circuit Design Practical)

The goal of this practical course is the design of a digital subsystem using the VLSI design techniques introduced in the VLSI Circuit Design lecture course. Students are encouraged to implement functions of their own invention, but may choose a function from a supplied selection. Examples of possible functions are: an associative memory, parts of an ALU data path, a DAP element, telephone dialler chips, digital stopwatch etc.

At the first meeting, the VLSI design tools available in the department are introduced. Thereafter participants discuss the progress of their designs during the weekly tutorial sessions.

Computer Science 420
(Concurrent Architectures)

In the last ten years there have been rapid developments in the field of concurrent architecture resulting in a plethora of concurrent systems. The primary motivation for using concurrent systems is high performance but, unfortunately, very few systems exhibit the ideal *linear* relationship between concurrency and performance. This course examines aspects of concurrency ranging from the expression of concurrency, and the principles of concurrent architecture, to a comprehensive survey of existing and proposed systems and their performance.

Topics include: Algorithmic Concurrency, Principles of Concurrent Architecture, Summary of SIMD Systems, Structured MIMD Architecture, Unstructured MIMD Architecture, Non-von Neuman Architecture, Future Developments.

COMPUTER SCIENCE 4

The final Honours course deals with advanced topics in Computer Science. The course provides the student with an opportunity to add to the core curriculum of the first three years and to study subjects from earlier years in greater depth. Throughout the whole of the fourth year, students undertake an original project – either research or implementation – under the direction of a member of staff. Approximately 50% of the time is expected to be spent working on the project.

At the end of the final year, a project report is presented as a typed and bound document to count towards the final degree awarded. Joint Honours students may also have to submit a dissertation, based on previously published work.

A degree of choice is available to final year Computer Science students since, although there are eleven possible modules, the students must take six. The choice can be somewhat limited for joint Honours students since modules must be taken from both streams.

Computer Science 410 (VLSI Circuit Design)

VLSI defines technologies for creating complex systems on a chip. The exploration of design possibilities, and the development of the computer aids essential to design implementation, make VLSI a new and exciting field of Computer Science. The course provides sufficient basic information about integrated devices, circuits, digital subsystems and system architectures to enable the participant to span the range of abstractions from the underlying physics to complete VLSI computer systems.

Topics include: Introduction to MOS Technology, Electrical Papameters - Clocking and Shift Register, Finite State Machines and the Programmable Logic Array, CMOS Process and Design Rules, Design Layout, Memory Systems, VLSI Models and the Assessment of Area and Time Requirements, VLSI Design Tools and their Underlying Problems, Parallel Computation using VLSI.

Practical exercises (primarily crayon and paper, but including simulation work near the end of the course) will be set.

DEPARTMENTAL COMPUTING FACILITIES

Computer Science 301 and 302 Projects

The aim of these projects is to introduce students to the methods of coping with problems that arise in the design and implementation of computer systems. They do not preclude any other course work which may be set in association with the lectures. It is intended that students will gain experience in:

- Designing clearly and coherently structured systems
- Choosing the appropriate means of implementation
- Discovering and using relevant information
- Scheduling their workload
- Presenting their findings and implementations clearly and consisely

Operating System Project

The Advanced Personal Machines (APMs) are used in implementing an operating system. The project is written up and presented as a typed and bound document. When marking, great attention is paid not only to what the student has achieved but also to the quality of presentation.

System Design Project

Students work in groups of about 8 on the design and implementation of the hardware and software of a small digital system e.g. a VDU. Practical work will be carried out mostly in the Summer term but groups will be formed at the start of the Spring term. Each group will establish its own management arrangements and assign responsibilities within the group for various parts of the project. During the Spring term the groups will meet regularly in workshop sessions to discuss their design and will investigate available systems and components in order to arrive at a specification of the system to be built. Each group will be able to 'employ' a member of staff in the department as a consultant but the amount of consultancy time will be taken into account when deciding the (hypothetical) cost of the project.

The project will be written up and presented as a typed and bound document. An audio/visual presentation will also be made and its quality will form part of the assessment of the project. Part of the assessment will also be peer group assessment by members of the group as to how well each of them has performed.

Computer Science 391
(Computability and Formal Languages)

The course centres round the *Church-Turing Thesis* which proposes that each one of a variety of different formal systems adequately formalises the intuitive concept of *(effective) computability*. This thesis implies that by studying any one of these systems, e.g. Turing machines, we can learn about the inherent theoretical limitations of computers: if a problem cannot be solved by a Turing machine then there is no computable solution to it at all.

Throughout the course, a number of concepts and techniques are introduced that have applications in many areas of Computer Science and will indeed occur - or already have occurred - in other courses. These include Turing machines, simulation methods, coding of programs, universal programs, diagonalisation arguments, problem reduction.

Topics include: Turing Machines; Primitive Recursive and Partial Recursive Functions; The Church-Turing Thesis; Manipulation of Computable Functions; Decidability, Partial Decidability and Undecidability.

Formal Language *topics include: Closure Properties of Regular Sets, Pumping Lemma for Context-free Languages, Closure Properties of Context-free Languages.*

Practical work will involve weekly pencil and paper exercises.

DEPARTMENTAL COMPUTING FACILITIES

The computing resources of the Department of Computer Science include both mainframes and powerful personal workstations, the latter sharing common filestores *via* Ethernet local area communication networks. At present, two types of single user machine are used – the SUN workstation and the Advanced Personal Machine (APM).

The SUN workstation is a proprietary machine with a high resolution screen running UNIX. About thirty-five of these are in use in the VLSI and Vision groups and within LFCS. The Advanced Personal Machine was designed and built within the department; the first workstation coming into use in 1981. These are modular machines currently based on a Motorola 68000 or 68010 main processor and have between 2 and 6 Megabytes of main store. At present, sixty-four of these are operated in personal offices and public terminal areas. The APM network has seven filestores with over a Gigabyte of total storage; twenty-five of the systems have 1024 × 1024 × 8 plane colour graphic framestores. The APMs are each about as powerful as the VAX 11/750 and are capable, *en masse*, of handling more work than all the departmental mainframes together. The APMs are used for second, third and fourth year undergraduate course work in subjects such as Real Time Systems, Graphics, VLSI and Operating Systems and for fourth year and M.Sc. projects.

The departmental mainframe, a VAX 11/780 running VMS, supports up to forty simultaneous users; its main uses being fourth year teaching, teaching support, text-processing, electronic mail and general administration. This has recently been augmented by a MicroVAX system. A second VAX mainframe, 11/750 running UNIX, is also housed in the department – its main use is within LFCS. The VAX mainframes are connected to both the departmental and ERCC networks. The laboratory also has a Pyramid 98XE RISC (Reduced Instruction Set Computer) system used as a filestore and central system. The departmental mainframe systems have a total of over 5 Gigabytes of disc store.

Under a collaborative agreement with the Gould Corporation, the Edinburgh University School of Information Technology (comprising the Departments of Computer Science, Artificial Intelligence and Electrical Engineering) were given a Gould Powernode 9080, running UNIX, to be used primarily for teaching. This supports up to sixty-four directly connected terminals plus X25 and Ethernet connections. The Department of Computer Science, through the School of Infor-

mation Technology, also has use of two GEC 63 series mainframes.

Other machines in use in the department include an Orion supermini computer forming part of a SERC/industry supported research project, two further microVAXex which form parts of industrially supported research projects, five PERQs and a number of Apple Macintoshes which are mainly used in first year teaching and administration. A Real Time Systems laboratory, situated in the machine halls, is equipped with several robot arms and a variety of mechanisms and microprocessor prototyping kits used in automation experiments. All departmental computing facilities provide access to a variety of special purpose peripherals including high quality laser document printers and plotters as well as line printers.

The Department of Computer Science and the Edinburgh Regional Computing Centre (ERCC) were jointly involved in the setting up of a microcomputer laboratory in Appleton Tower at the central university site. This comprises mostly Apple Macintosh and Sirius microcomputers along with a selection of terminals, linked *via* Ednet, accessing the ERCC machines (including two ICL 2976 systems linked with two ICL Distributed Array Processors (DAPs), two ICL 2988 systems and an Amdahl V7). These systems all run EMAS (Edinburgh Multi Access System).

Computer Science 390
(Analysis of Algorithms)

The CS3 Analysis of Algorithms course is concerned with the application of (mostly elementary)mathematical techniques to the design and analysis of efficient algorithms. The range of techniqes for algorithm design introduced in the second year Foundations course is augmented and developed. In addition, the theory of NP completeness is used as a tool for indicating those problems which are inherently intractable.

Topics include: Introductory Concepts, Comparison Problems, The Classes P and NP, Algebraic Problems, Graph Algorithms.

Practical work will involve pencil and paper exercises and one small programming exercise.

Computer Science 350
(Computer Systems Modelling)

An introduction to the use of modelling to study the performance of computer systems. The course covers the application of probability, queueing theory and discrete event simulation to develop useful models.

Topics include: Discrete Event Simulation Modelling, Stochastic Processes and Markov Chains, Networks of Queues, Approximation Techniques.

Practicals will involve 3 modelling exercises.

Computer Science 380
(Database Systems)

A database may be regarded as a model of some appliction reality. The course concentrates on this modelling process and examines the capabilities of different database systems to capture the relevant properties of the application reality. In addition, the software support required to control shared access to a database will be discussed.

Topics include: Data Models, Data Semantics, Database Management Systems, Distributed Databases.

Practical work will involve the design and construction of a database.

MEMBERS OF STAFF

Computer Science 340
(Programming Methodology)

Great emphasis is currently being laid on the 'quality' of software, and this module aims to deal with techniques which may be used to ensure the production of good quality software. There are two main areas of concern: programming in the small and programming in the large. Programming in the small is concerned with systematic program design methods, both formal and informal, and with ways of reasoning mathematically about program correctness. Programming in the large is concerned with the management of large pieces of software, where many individuals may each contribute a part of the whole, or where the complexity of the system is such that no one individual could possibly maintain a full grasp of the whole at any one time.

Topics include: Logic and Proofs, The model *Approach to Specification, The* axiomatic *Approach to Specification, Program Proofs, Software Engineering.*

Practical work will involve weekly pencil and paper exercises.

Computer Science 341
(Language Semantics and Implementation)

The aim of this course is to present a unified view of programming language semantics and implementation, based upon the linked notions of structured operational semantics and abstract machines. Both declarative and imperative languages will be treated.

Topics include: Structured Operational Semantics, Static Semantics(Type Checking), Equivalence and Robustness Properties of Abstract Machines, Translation of Imperative and Functional Languages to Abstract Machine Code and their Runtime Support, Operational Semantics and Abstract Machines in Compiler Compilers.

Practicals will involve weekly exercises, both pencil and paper and computing.

Computer Science 321
(Computer Architecture)

Computer architects concern themselves with techniques which will improve the performance and/or reduce the cost of computer systems. Over the years, changes in technology have played a major role in providing these improvements but a whole variety of different machine architectures have been developed taking advantage of these changes. This course examines a number of technological and architectural techniques and their interrelationships and shows how the architectures of modern supercomputers have evolved. Students will be required to write an essay on a topic in computer architecture.

Topics include: Instruction Formats, Storage Hierarchies, Pipelines, Instruction Buffering, Parallel Functional Units, Vector Processors.

Computer Science 330
(Operating Systems)

A wide range of principles and techniques used in building operating systems is expanded in this course. Considerable importance is attached to the practicalities of design and implementation. The aim is to provide the student with a principled basis for designing useful systems.

Topics include: Concurrent Processes, Virtual Memory, CPU Scheduling, Deadlocks, File Systems, Distributed Systems.

Practical work mostly involves reading material for discussion in tutorials, with an introduction to system level programming on the APMs in preparation for the major practical.

MEMBERS OF STAFF

Professor R.M. Burstall : Design of Pebble - a functional language with strong type checking and dependant types. Development of a proof editor using incremental evaluation for attribute grammars to give an attractive user interface. Development of the 'institutions' concept to give an abstract categorical basis for the design of specification languages.

Professor R.N. Ibbett : Processor and Network Architecture: the design and implementation of hardware, firmware and protocols to support an integrated computing resource network incorporating a high performance local area network and a variety of parallel and specialised processors including, in particular, a sparse vector processor.

Professor S. Michaelson : Computers in the Humanities e.g. stylometry, uses in museums. System modelling especially queueing models. Design of distributed systems. Office automation.

Professor A.J.R.G. Milner : Director of Laboratory for Foundations of Computer Science.
Research interests in mathematical models of computation particularly those supporting proof methodologies using machine assistance; how to present logics to machines; models and proof theory for concurrent computation; operational semantics of high level programming constructs.

Professor G.D. Plotkin : Applications of logic in Computer Science, especially denotational and operational semantics of programming languages. Current work is on concurrency, intuitionistic modal logics, general proof theory. Also some interest in Artificial Intelligence and Situation Theory.

Senior Lecturers

Dr. D.J. Rees : VLSI design and associated Computer Aided Design tools. Silicon Compilation from high level behavioural and structural descriptions to mask level artwork. Hardware Description Languages for behaviour and for structural composition. Novel VLSI architectures.

Mr. P.D.A. Schofield : Head of Department.
The use of computers in teaching, assessment and departmental administration – particularly the automatic marking of and detection of plagiarism in practical assignments. Data structures; sorting and searching algorithms.

Lecturers

Mr. S.O. Anderson : Use of programming logics in the specification and development of verified programs; in particular the use of Martin-Lof's type theory and the provision of proof assistants for the theory. Development of theories of particular application areas within type theory (e.g. user interface design).

Dr. A. Blake : Computer Vision – Development of theory of computer programs for analysis of monocular and stereo images. Applications include industrial assembly robots and autonomous vehicules.

Dr. G.J. Brebner : The complexity analysis of 'realistic' parallel computing systems when used to implement 'idealistic' parallel algorithms. The design and implementation of a computing environment distributed over a local area and composed of heterogeneous systems but offering an integrated interface to the user.

Dr. E.R.S. Candlin : Distributed systems. Languages for real-time systems. Vision systems for automation.

Dr. A. Deas : Silicon Compilation.

Dr. M.R. Jerrum : Computational Complexity: data structures, algorithms (especially probabilistic), combinatorics, Boolean network complexity.

Dr. K.A. Karlorkoti : Computational Complexity with special interest in algebraic complexity. This includes matrix problems and bounds on formula size of algebraic functions.

Dr. C.D. McCarty : Intuitional mathematics, constructive logic, non-classical logics, computability theory, classical recursion theory, classical and constructive set theory, theory of domains, topics in the formal semantics of natural language.

Mr. R.A. McKenzie : VLSI design, VLSI architecture for graphics, solid body modelling, extracting solid description from images.

Dr. K.N.P. Mitchell : Functional programming language design and implementation; Concurrency and the formal verification of parallel algorithms; The use of CCS and its derivatives as the basis of a parallel programming language.

COMPUTER SCIENCE 3

The third year Computer Science course provides a basic foundation for the design and implementation of computer systems. Software oriented lectures concentrate on the overall design of software systems and their interaction with the underlying hardware; hardware lectures reflect recent developments in methods of hardware implementation. Throughout the course the theoretical foundations of Computer Science are examined.

The course consists of ten modules, all of which are compulsory for single Honours Computer Science students; joint Honours students generally take half the Computer Science modules. (See Appendix for details). Each module consists of 18 lectures. In addition to the lecture modules, there are two major practical modules, CS301 and CS302, which are also compulsory.

Computer Science 310 (VLSI Circuit Design)

An introduction to VLSI design from a systems perspective ie. at an architectural level rather than a 'device physics' level. CAD tools for VLSI design and Silicon Compilation will also be discussed.

Topics include: Structured Design, Architectures, CAD Tools.

Practical exercises on APMs are designed to back up the lectures.

Computer Science 320 (Computer Design)

This course details the principles and practices employed in the design of computing hardware. It describes how processors, memory and I/O systems are built up, starting at the level of electronic devices, into machines capable of executing complex instruction sets.

Topics include: Gate Level Design, Register Level Design, Memory Design, I/O Design, Buses, Microprocessor Families, Computer Design Examples.

Practical work involves five design/implementation exercises.

Computer Science 2h (REAL TIME)

This course is concerned with the design of computer systems which have to interact with unpredictable events in the outside world. Such systems are characterised by a high degree of parallelism and tight time constraints and are particularly difficult to implement correctly. Systems of this type are important in many fields of Science, Engineering and Medicine – for example, on-line monitoring and control of equipment.

One of the aims is to give students a good understanding of the processes involved when a computer accepts and interprets data from its environment or acts to affect changes in it. The course covers all aspects of this interaction – interfacing, the interpretation of signals, speech and vision systems and control algorithms.

The second strong theme is that of software design for real-time systems. This part of the course includes a detailed study of OCCAM as an example of a high level language explicitly designed to handle parallelism.

Topics include: Characteristics of Real Time Systems, Interaction of the Computer with its Environment, Interpretation of Signals, Speech Synthesis, Image Processing, Computer Control, Communications, Programming for parallelism.

Practical work involves simple experiments to demonstrate various transducers and interfaces and to give a feel for absolute timing. Practicals are also included on Low level I/O including Interupts and Accurate Timing, Ethernet Communications, Robot Control, Vision Systems and an OCCAM exercise.

Miss M.C. Norrie : The formalism of data models, conceptual modelling and conceptual programming languages, heterogeneous distributed database systems.

Mr. R.J. Pooley : Performance analysis and language design. Engaged in an Alvey project to build an integrated modelling support environment using graphical programming techniques. Compiler consultant to Sparse Machine project. Design of efficient languages for object oriented programming.

Dr. R.N. Procter : Information Systems – design and applications, computer aided learning, Information Technology education, social and organisational impact of Information Technology, soft machines.

Dr. D.T. Sannella : Algebraic specification and formal program development, mechanised reasoning, programming methodology and functional programming languages.

Mr. F. Stacey : All aspects of distributed operating systems, particularly the provision of file services. The specification and verification of (parallel) algorithms used in the implementation of such services.

Dr. C.P. Stirling : Semantics of concurrency and the application of modal and tense logics to verification of concurrent programs.

Mr. B.C. Tompsett : Software engineering problems of constructing and maintaining large pieces of software. Engineering of portable operating systems and portable compilers. Wide area networks. Software project planning and management.

Dr. N.P. Topham : Distributed and parallel architectures for VLSI, language-driven architectures and architectures for Artificial Intelligence applications. Current research work centres round the design and implementation of a VLSI context-flow machine.

Dr. A.S. Wight : Computer systems performance evaluation. Flow and congestion control in computer communications networks. Performance modelling environments. Workload characterisation.

Demonstrators

Mr. T.M. Hopkins : High bandwidth local area networks. Interconnection of local area networks. Network support for special purpose high perfor-

mance computing engines. Specialised processor design for sparse vector computation.

Mr. A. Sinclair : Complexity theory, design and analysis of algorithms, randomised computation and probabilistic algorithms, approximation algorithms for combinatorial problems.

Computing Officers

Mr. M. Allen	Non-LFCS UNIX support, text processing
Mr. D.W. Baines	Gandalf Contention Switch, CS2 support
Ms. J.T. Blishen	VLSI, publicity
Mr. M. Brown	Performance
Mr. J.H. Butler	Service Manager
Mr. K.J. Chisholm	Graphics, databases
Dr. G.L. Cleland	Assistant Director, LFCS
Ms. L. Desjardins	(Computing Support Officer) VAX database support
Mr. R.J. Green	CS1 support, VLSI
Mr. A. Howitt	CS3 hardware lab., hardware projects
Ms. K.M. Humphry	IS1 teaching, VAX database
Mr. M.R. King	APM hardware design, high performance personal computers
Dr. G.D.M. Ross	Filestores, file transfer
Mr. A.J. Scobie	VAX manager, APMs
Mr. H.R. Stabler	LFCS Systems Programmer
Mr. R.W. Thonnes	APMs, BEPI support
Mr. J.S. Turnbull	IS1, CS2 (Real Time) support
Dr. D.A. Welch	Administration
Mr. J. Wexler	Operating Systems, publicity
Miss L. Wilkie	(Computing Support Officer) Orion Manager, Advisory/Support

Secretarial Staff

Ms. H.L. Carlin	Secretary to Head of Department (part-time)
Mrs. K.B. Duncan	Secretary to Professor Michaelson
Mrs. A.M. Fleming	Secretary to Professor Ibbett (part-time)
Miss E.A. Kerse	Secretary to Professor Burstall
Miss K.A. McCall	Secretary
Mrs. D.A. McKie	Secretary to Professor Milner
Mrs. M.W. Melvin	Secretary
Miss J.M.K. Ratcliff	Secretary to Dr. Cleland

COMPUTER SCIENCE 2

In second year Computer Science, three half courses are available — Computer Systems which runs for the first half of the academic year, Foundations and Real Time Systems which run concurrently in the second half of the year. Students usually study Systems and Foundations, with engineering students studying Computer Systems and Real Time Systems. It is possible to take all three half courses.

Computer Science 2h (SYSTEMS)

This course continues the study of some themes introduced in Computer Science 1A: systems hardware and programming languages and their translation. A new topic, computer graphics, is also introduced.

Topics include: Functional Programming, Graphics, Computer Architecture and Assembly Language Programming, Compilers.

There are practical exercises assosiated with all parts of the Systems course: ML programming, graphics, assembly language programming and compiler construction.

Computer Science 2h (FOUNDATIONS)

The Computer Science Foundations course aims to provide the necessary mathematical tools demanded by subsequent (third and fourth year) courses. If the course provided *only* this it would be extremely dry therefore no concept or principle will be introduced without accompanying computer science applications. An attempt will be made to draw the examples from as many areas of computer science as possible – to this extent the course can be viewed as a 'sampler' for courses offered in the subsequent years. In summary, this course aims to show that computer science is a subject which involves many conceptual issues worthy of attention.

Topics include: Preliminaries, Finite Automata and Regular Expressions, Design and Analysis of Algorithms, Combinatorial Logic, Phrase Structured Grammars.

COMPUTER SCIENCE 1B

This course is intended for science and engineering students who do not intend to pursue computer science beyond the end of first year. Some of the lectures are taken in common with CS1A, but in general this course will be more applications oriented than CS1A and there will be greater emphasis on the use of programming packages.

Syllabus for CS1B1h

The aim of this half course is to give students a basic competence in programming, including the use of programming packages, and a general knowledge of modern computer systems. Pascal is chosen as the programming language, because it provides a good foundation for learning programming concepts, and includes most features found in other procedural languages. Thus students are introduced to concepts such as: program structure, procedures, data types, data structures and control structures, all illustrated by practical examples.

Topics include: Pascal Programming, Computer Hardware.

Syllabus for CS1B2h

The second half of CS1B starts by extending the students' knowledge of data structures before proceeding to a brief historical perspective on a wide range of languages and a more detailed comparison of a modern imperative language (e.g. ADA) with a modern applicative language (e.g. ML).

Lisp is presented in enough detail to give students a working knowledge of the language, which, together with a brief description of Prolog, introduces students to the important area of programs which have some 'reasoning' capability.

Topics include: Data Structures, Comparison of Languages, Lisp and Prolog, Expert Systems, Modelling and Simulation, Graphics and CAD.

Practicals

Associated with CS1B1h there will be four short Pascal exercises and two medium-sized practicals. Associated with CS1B2h there will be one exercise in Lisp/Expert Systems and one Pascal exercise in modelling/simulation. These exercises will also build on the student's knowledge of data structures.

Technical Staff

Mr. I.T. Conkey	Communications
Mr. J.C. Dow	Laboratory Superintendant
Mr. A. Duncan	Junior Technician
Mr. M.L. Graham	APM Production
Mr. D.C. Hamilton	Workshop
Mr. J. Johnstone	APM Production
Mr. P.J. Lindsay	Real Time and Special Systems
Mr. I.S. Marr	Junior Technician
Mr. T.S. Wigham	APM Production

Research Assistants

RA/Supervisor	Project
Ruth Addinall (R.N. Ibbett)	Sparse Vector Project
Arnon Avron (R. Milner)	Proof Project
Genbao Feng (D.J. Rees)	Design of mask programmable cellular logic array structures
Tatsuya Hagino (R. Burstall)	Categorical data types
Bob Harper (R. Milner)	Application of logic to program construction
Furio Honsell (R. Milner)	LFCS
Lin Huimin (R. Burstall)	SML interface to X window

Ian Mason (R. Milner)	Verification, Transformation and the LF
Eugenio Moggi (G.D. Plotkin)	Lambda calculus and categories
Joachim Parrow (R. Milner)	Tools and techniques for analysis of concurrent communicating systems
Nick Rothwell (R. Milner)	ML development project
Paul Taylor (R. Burstall)	Specifications in ML
David Walker (R. Milner)	Mathematical theory of concurrency
Andrew Zisserman (A. Blake)	Surface reconstruction and representation in computer vision

Full Time M.Phil./ Ph.D. Students

Student/Supervisor	Project
Simon Ambler (C.D. McCarty)	Logic and the Theory of Computation
Duncan Baillie (R.N. Ibbett)	Synergy in Computing Systems
Dave Berry (R. Milner)	Generating Interactive Programming Environments

COMPUTER SCIENCE 1A

This course is intended for students who intend to pursue computer science as a single or joint Honours course in subsequent years.

Syllabus for CS1A1h

The aim of this half course is to give students a basic competence in programming, and a general knowledge of modern computer systems. Pascal is chosen as the programming language, because it provides a good foundation for learning programming concepts, and includes most features found in other procedural languages. Thus students are introduced to concepts such as: program structure, procedures, data types, data structures and control structures, all illustrated by practical examples.

Topics include: Pascal Programming, Computer Hardware.

Syllabus for CS1A2h

The second half of CS1A is divided into three major sections. The first provides an in-depth analysis of data structures, using Pascal implementations as examples, and provides a brief introduction, through Prolog, to programs which have some 'reasoning' capability.

Sorting is an important underlying requirement of many applications in computing, and the second section of CS1A2 not only provides a comparison of a number of different sorting algorithms but also uses these algorithms to introduce a number of different programming techniques.

The final section is concerned with the mechanisms by which (Pascal) programs are run on a computer and thus deals with the run-time environment and with aspects of compilation.

Topics include: Data Structures, Introduction to Prolog, Sorting Techniques, Implementation of High Level Languages.

Practicals

Associated with CS1A1h there will be four short Pascal exercises and two more substantial practicals. Associated with CS1A2h there will be one practical associated with data structures, one with sorting and one with parsing.

Gavin Brelstaff (A. Blake)	Inferring Surface Properties from Specularities
Simon Brown (C.D. McCarty)	Logics, Models and Concurrency
Carlo Cecci (K.N.P. Mitchell)	Semantics and Pragmatics of Declarative Programming Languages
Murray Cole (M.R. Jerrum/G.J. Brebner)	Efficient Implementation of Parallel Algorithms
Mads Dam (G.D. Plotkin/C.P. Stirling)	Denotational Models for Concurrency
Bruce Davie (D.J. Rees)	Design Languages and Design Methods for VLSI
Mark Davoren (R.N. Ibbett)	Distributed Operating Systems
Jordi Farres-Casals (D.T. Sannella)	Specification and Semantics
James Harland (D.T. Sannella)	Logic Programming
Martin Illsley (R.M. Burstall/G.D. Plotkin)	Program Transformations and Synthesis
Claire Jones (G.D. Plotkin)	Proof Editing or Probabilistic Semantics
Tom Kean (D.J. Rees)	Micro-Grain Architectures for VLSI
Andreas Knobel (C.D. McCarty/M.R. Jerrum)	Realisation and the Problem of Full Abstraction

Laurent Langlois — Algorithms for Parallel Parsing
(S. Michaelson/D.J. Rees)

Gary Law — A Triadic Network Model
(R.N. Ibbett)

Tim Lees — Parallel Computer Architectures
(N.P. Topham)

Constantine Marinos — Computer Vision
(A. Blake)

Faron Moller — Theory of Concurrent Processes
(R. Milner)

Ian Nixon — The Application of VLSI to Nucleonic Instrumentation
(D.J. Rees)

Pawel Paczkowski — Concurrency: Semantics and Formal Reasoning about Programs
(C.P. Stirling/C.D. McCarty)

Steve Proctor — Computer Networks and Distributed Computation
(G.J. Brebner)

David Pym — Theory of Computation
(G.D. Plotkin)

Tom Stiemerling — VLSI Architecture of Concurrent Processors particularly Pipelined Multi-Microprogrammed Processors
(N.P. Topham)

Sun Yong — Computational Theory
(G.D. Plotkin/R.M. Burstall)

Mads Tofte — Operational Semantics and Type Inference
(R. Milner)

Chris Tofts — Theory Of Computation
(R. Milner)

UNDERGRADUATE COURSES

The degree structure at Edinburgh is very flexible and students can usually keep their options open (as far as final degree is concerned) to the end of the first and, in some cases, the second year.

The first year Computer Science course assumes no prior knowledge of computer systems and therefore forms a broad introduction to the subject. Many students have met with computers at school or through home whereas others have not. From the beginning of the course, students are encouraged to obtain practical experience using computers. In the first year, the computer system used is the University's large mainframe EMAS (Edinburgh Multi Access System). In the second and subsequent years, students have the opportunity to work with a variety of other computer systems ranging from the departmental VAX or Gould multi access systems, through mini-computers to microprocessor systems. In early years, students write software for existing hardware configurations; later they may construct their own hardware from standard components and are able to design their own VLSI (Very Large Scale Integrated (circuit)) chips. Throughout the course, emphasis is laid on the concept that a computer system is a fusion of hardware and software design - each playing a fundamental role in the efficient running of the overall system.

A fundamental understanding of Computer Science can be achieved from an abstract and theoretical treatment and this forms an important part of the third and final year courses.

Course Assessment

In the first three years, class examinations are usually held at the end of each term. Although these do not count directly in the final degree, some guide to the student's own progress is obtained. Good results in the first year class examinations however can earn exemption from the first year degree examination.

Written examinations ('degree' examinations) are held annually in June. Passes in the first three years (seven in all) are counted towards an Ordinary degree. For Honours, the final degree is assigned on the basis of the results of third and fourth year examinations. In all the degree examinations, course work throughout the year is counted in the final assessment.

Tom Waring (S. Michaelson/A.R. Deas)	Silicon Assembly
Rod Widdowson (D.J. Rees)	Associative Processing in VLSI
Eric Wilson (S. Michaelson)	Computer-assisted Study of Old English Word Order

B.Sc./B.Eng. HONOURS DEGREES IN COMPUTER SCIENCE

The Department of Computer Science at Edinburgh offers several Honours degrees, as well as providing undergraduate courses contributing to other degrees, outside of Computer Science. The Honours degrees offered are

- Computer Science (B.Sc./B.Eng.)
- Computer Science and Electronics (B.Eng.)
- Computer Science and Physics (B.Sc.)
- Computer Science and Mathematics (B.Sc.)
- Computer Science and Statistics (B.Sc.)
- Computer Science and Artificial Intelligence (B.Sc.)
- Computer Science and Management Science (B.Sc.)

The B.Eng. degree was introduced to conform to an Engineering Council directive that all degrees leading to accreditation should be designated B.Eng. These degrees are intended for students wishing to become members of the relevant professional Engineering Institution. Computer Science degrees give exemption from the British Computer Society's membership examinations. An Honours degree in Computer Science taken under the B.Eng. regulations includes elements of 'The Engineer in Society' – topics relevant to a practising engineer.

An Honours degree in Computer Science (single or joint Honours) lasts four years during the first two of which students take three subjects, one being Computer Science. Students intent on studying for a joint Honours degree must also take the prescribed course(s) relevant to that subject. During the first two years, a substantial fraction of time can be spent on Support Courses. This is an important element in the education of an Edinburgh graduate since it presents an opportunity for the students to broaden their interests. In the final two years, students study courses in their chosen speciality/ies only.

APPENDIX

HONOURS DEGREES IN COMPUTER SCIENCE

research working towards the degrees of M.Phil. or Ph.D. The School of Information Technology was established to coordinate undergraduate and postgraduate teaching in Information Technology and to ensure that all courses in Information Technology subjects reflected the important new developements taking place in the area.

Research in the Department of Computer Science started in 1966 with a multi access project from which developed the interactive computing system EMAS (Edinburgh Multi Access System). This now provides a computing service for the University. The Computer Science department at Edinburgh has an international reputation for the quality of its research. There are several very active groups working in a number of different fields — new programming languages, integrated circuit design, graphics, communication systems and high performance computer systems for example.

The Department is situated in the James Clerk Maxwell Building (JCMB) at The Kings Buildings (the science campus at Edinburgh University), about 2 miles from the centre of the city. This building is shared with several other departments including Mathematics, Physics, Meteorology and Statistics and provides extensive accomodation including lecture theatres, machine halls, a library and several common rooms.

Departmental Computing Facilities

There is a wide range of computer systems in the Department of Computer Science. Facilities include a VAX 11/780 mainframe (ECSVAX) which is used by all staff and third and fourth year students and over 70 single user machines. These workstations fall into two categories – the SUNs and the APMs (Advanced Personal Machines). The APM was designed and built in the department and provided a service five years before any similar commercially available machine. It offers many facilities including colour graphics. The Department of Computer Science also has a share in a Gould and two GEC mainframes with the other members of the School of Information Technology and has access through the ERCC to the University computing facilities.

First Year:

Computer Science 1A
2 other courses

Second Year:

Computer Science 2A:-

CS201 Computer systems
CS202 Foundations

2 other courses

Third Year:

Computer Science 3:-

CS310 VLSI Circuit Design
CS320 Computer Design
CS321 Computer Architecture
CS330 Operating Systems
CS340 Programming Methodology
CS341 Language Semantics & Implementation
CS350 Computer Systems Modelling
CS380 Database Systems
CS390 Analysis of Algorithms
CS391 Computability

2 Major Practicals

Fourth Year:

6 lecture modules, at least 5 from Computer Science 4:-

CS410 VLSI Circuit Design
CS411 VSLI Circuit Design Practical
CS420 Concurrent Architectures
CS430 Compiler Techniques
CS440 Denotational Semantics
CS450 Computer Systems Performance Evaluation
CS460 Digital Communications
CS461 Theory of Communicating Systems
CS470 Advanced Graphics
CS480 Conceptional Modelling
CS490 Computational Complexity

A Major Project

COMPUTER SCIENCE & ELECTRONICS

First Year:

Computer Science 1A
Mathematics 1
1 of:-

- Engineering 1
- Physics 1
- Electronics 1h + Computer Science 1Ah
- Electronics 1h + Physics 1h

Second Year:

Computer Science 2B:-

- CS201 Computer Systems
- CS203 Real Time Systems

Electrical Engineering 2
Mathematics 2

Third Year:

From Computer Science 3:-

- CS320 Computer Design
- CS321 Computer Architecture
- CS330 Operating Systems
- 2 other modules

1 Major Practical

From Electronic and Electrical Engineering 3:-

- Analogue Communications
- Digital Circuits
- Microelectronic Devices
- Microelectronic Systems
- Systems Theory

Fourth Year:

6 lecture modules, at least 2 from each department

A Major Project in one discipline
A Dissertation in the other

INTRODUCTION

The City and its University

The University of Edinburgh was founded over 400 years ago in 1583 — some twenty years before the Union of the Crowns which made James VI (I) king of both Scotland and England. Throughout its history it has numbered a host of distinguished scholars amongst its teachers and students. Its situation in the capital of Scotland, one of the world's most beautiful cities, has helped maintain this tradition.

Today, the University of Edinburgh is one of the largest in the UK. Its students enjoying not only the many cultural, sporting and social aspects of the city but also the advantages of studying at a well-endowed university. An example of the latter is provided by the unusually large number of optional courses which the university, because of its size, can offer to its students and the resulting flexibility of its degree system.

The Department of Computer Science

The Department of Computer Science at Edinburgh was established in 1966 along with the Edinburgh Regional Computing Centre (ERCC) when the, then, Computer Unit was reviewed. It was created as a small department with six staff and has now grown to one of the largest departments in the Science faculty with five professors.

In 1966, teaching in Computer Science consisted of a postgraduate Diploma and a first year undergraduate course in the subject. This has been substantially increased over the years and the department now offers an Honours degree in Computer Science as well as joint Honours degrees with Electronics, Physics, Mathematics, Statistics and Management Science. In 1980, a terminal first year course, Information Systems 1, was introduced aimed specifically at non-Computer Scientists. The department also offers a number of postgraduate courses ranging from a M.Sc. in Information Technology (run jointly with Electrical Engineering and Artificial Intelligence as the School of Information Technology) to study by

COMPUTER SCIENCE & PHYSICS

First Year:

Computer Science 1
Physics 1A
Mathematics 1

Second Year:

Computer Science 2B:-

CS201 Computer Systems
CS203 Real Time Systems

Physics 2A
Mathematics 2

Third Year:

From Computer Science 3:-

CS310 VLSI Design
CS320 Computer Design
CS321 Computer Architecture
CS330 Operating Systems
1 other module

1 Major Practical

From Physics 3:-

Mathematical Methods 1
Quantum Physics 1
Statistical Physics 1
Electromagnetism or Electromagnetic Theory 1

and *normally* 1 of:-

Acoustics & Dynamics of Fluids
Nuclear Physics 1
Solid State Physics 1
Quantum Physics 2

Fourth Year:

6 lecture modules, 3 from each department

A Major Project relating computer science and physics

COMPUTER SCIENCE & MATHEMATICS

First Year:

Computer Science 1
Mathematics 1A
Applied Mathematics 1Ah
1 other half course

Second Year:

Computer Science 2A:-

CS201 Computer Systems
CS202 Foundations

Mathematics 2A
1 other course

Third Year:

From Computer Science 3:-

CS321 Computer Architecture
CS330 Operating Systems
CS340 Programme Methodology
CS390 Analysis of Algorithms
CS391 Computability

1 Major Practical

From Mathematics 3A:-

Probability
Algebra
Mathematical Programming
Real Analysis
Set Theory

Fourth Year:

7 lecture modules including:-

2 in theoretical computer science
1 other computer science module

Mathematical Logic
2 other mathematics modules

A Major Project relating mathematics and computer science

INTRODUCTION

COMPUTER SCIENCE & STATISTICS

First Year:

Computer Science 1A
Mathematics 1
1 other course

Second Year:

Computer Science 2A:-

CS201 Computer Systems
CS202 Foundations

Mathematics 2
Statistics 2A

Third Year:

From Computer Science 3:-

Any 7 lecture modules

1 major Practical

Statistics 3

Fourth Year:

From Computer Science 4:-

2 lecture modules

Statistics 4:-

5 lecture modules

A Major Project relating statistics and computer science

ARTIFICIAL INTELLIGENCE & COMPUTER SCIENCE

First Year:

Artificial Intelligence 1 (recommended)
Computer Science 1A
Mathematics 1

Second Year:

Artificial Intelligence 2
Computer Science 2:-

CS201 Computer Systems
CS202 Foundations
CS203 Real Time Systems

1 other half course

Third Year:

Artificial Intelligence 3:-

Knowledge Representation 1
Knowledge Representation 2
Natural Language Processing
Vision

From Computer Science 3:-

CS321 Computer Architecture
CS330 Operating Systems
CS340 Programming Methodology
CS380 Database Systems
CS390 Analysis of Algorithms
CS391 Computability

1 Major Practical

Fourth Year:

6 lecture modules, at least 2 from each department

Artificial Intelligence 4:-

Expert Systems
Mathematical Reasoning
Robotics

A Major Project in one discipline
A Dissertation in the other

Table of Contents

Introduction	1
Honours Degrees in Computer Science	3
Undergraduate Courses	5
Postgraduate Study	23
Research in Computer Science	25
Laboratory for Foundations of Computer Science (LFCS)	38
Departmental Computing Facilities	42
Members of Staff	44
Appendix	

COMPUTER SCIENCE & MANAGEMENT SCIENCE

First Year:

Computer Science 1
Business Studies 1
1 other course

Second Year:

Computer Science 2A:-

CS201 Computer Systems
CS202 Foundations

Business Studies 2:-

Management Science A
1 other half course

1 other course

Third Year:

From Computer Science 3:-

7 lecture modules

or

6 lecture modules + 1 Major Practical

From Business Studies 2/3:-

Management Science B
1 other Business Studies 2/3h half course

OR

From Computer Science 3:-

8 lecture modules

2 Major Practicals

From Business Studies 2/3:-

Management Science B

Fourth Year:

6 lecture modules:-

2 from Computer Science 4
2 from Business Studies Honours
2 other modules

A Major Project relating management science and computer science
A dissertation in management science

University of Edinburgh

Department of Computer Science

DEPARTMENTAL HANDBOOK

January 1987

For further information and additional copies of this document please contact:

Dr. D.A. Welch
Department of Computer Science
University of Edinburgh
Mayfield Road
Edinburgh EH9 3JZ Telephone 031-667-1081 ext. 2807

E-mail: DAW@uk.ac.ed.ecsvax