Paper I: The Construction and Organisation of Computers

1. To understand the principles on which computers operate and the interrelation between the various units of a computer system.

Storage: Two-stage devices as the basis of storage, groups of such devices represented as binary patterns whether data or instructions. The concept of registers in the main processor to store fixed length patterns, not necessarily all of the same length. Comparison of parallel and serial access methods with regard to speed and complexity; illustrations can be found from past and present computers. The concept of a *stack* (push down list). Concept of addressable cells. Comparison of various levels of access to store; e.g. as illustrated by registers, ferrite core store, rotating magnetic media, reeled magnetic media etc.

Processing: It is important to appreciate thet the interpretation of a pattern of bits depends on the circumstances in which it is used which could be as character codes, numeric data, instructions or any other coded symbolic representation. Logical operations as the basis of processing, whether dealing with symbolic data or numeric data. Binary arithmetic (particularly addition and subtraction) can be built up from a basis of logical operations on bits (relate to sections 5, 6) through to floating point operations.

Control: How instructions are held in store, the use of fields within an instruction, sequence of operations: selecting, interpreting and executing an instruction and determining the next instruction to be dealt with. The flow of data (instructions and operands) between the various registers and units in a typical instruction. The concept of an interrupt and how it permits the suspension of the ‘normal’ sequence of instruction cycles; (this could be compared with an externally stimulated subroutine call).

Input/Output: Comparison of forms of data as presented or interpreted by the human with internally represented form of data. The necessity for conversion between internal and external forms, e.g. card code to 6—bit code. Introduction to the problem of handling input/output devices, including methode of addressing peripherals (e.g. different function codes, address within instruction, setting up register in input/output channel etc.), the use of buffer areas within store for assembling blocks of data prior to processing (or output) and the importance of standard interfaces; i.e. so that all peripherals appear of the same form to the central processor.

2. *Off—line Equipment*

The student should have a good general knowledge of the methods and problems associated with data preparation — not only those facilities with which he has direct experience. Critical comparison should be encouraged, taking into account such criteria as speed, cost of media, cost of units, space considerations, handling characteristics and user convenience. The most up—to—date techniques of input of data, not involving the punching of cards or paper tape, and the assignment of data to magnetic tapes either directly or via discs should be described.

3. *On—line Equipment*

The student should have a good general knowledge of the various methods which can be used for input and output of data, including methods which may not be within his practical experience. In particular he should appreciate that analog devices can be connected to a digital computer through D/A and A/D convertors (reference could be made to graphical methods of input and output in this context). The student should have detailed knowledge of the principles of operation of at least one input device, one output device and A/D, D/A conversion.

Principal functions and physical characteristics of the main storage media. The principles of operation of the devices, speed and storage capabilities. This should include calculations of comparative performance characteristics from given data. In this context a typical example is the effect on magnetic tape efficiency of the numbers of records per block.

4. *Detailed Aspects of Storage*

Use of registers to represent characters, integers and fractions — the advantages and limitations of fixed length. Comparison with variable length storage concepts and the need for field and record terminators in variable length operation, and ‘decimal arithmetic’ machines. General methods of conversion between number systems — with particular reference to binary, octal, hexadecimal and decimal as the systems most commonly met. Representation of negative numbers in binary including ‘ones’ and ‘twos’ complement methods. Relation between the two methods and the relevance to binary subtraction. Use of guard bits to protect against out of range numbers. General concepts of floating point; allocation of bits to various parts in binary representation; advantaåsees and disadvantages should include convenience of use as compared with the overheads in storage space accuracy and execution time.

5. *Machine Logic*

The student should be able to analyse logical problems and represent these in Boolean algebra terms (up to 4 independent variables). He should have a guneral idea of Algebrcic manipulation, Venn diagrams and Truth tables with detailed manipulative ability in at least one, enabling him to simplify Boolean expressions, and to interpret results according to the problem concerned. (Karnaugh maps are acceptable but not essential). Illustrative problems can be drawn from other parts of the syllabus. Conversion from Boolean expression (simplified form usually implied) to logic diagrams using either AND—OR—NOT or NAND or NOR systems of logic. Knowledge of how a particular type of gate is constructed (electronically or otherwise) is not required.

6. *Assembly Languages*

The student should have a good appreciation of the relevance of the instruction format and instruction set to the capability of a central processor. The discussion should not be restricted to the particular machine which is available to the student, although this could naturally form a basis for the study. The student should be aware of the relationship between the symbolic or mnemonic facilities in an assembly language and the internal representation at machine code level, and know in particular which facilities have one—one correspondence (usually processing instructions) and which may not have direct correspondence within the machine (e.g. storage or operating directives, macro—instructions). It is suggested that students undertake simple exercises to supplement these ideas.

7. *Systems Software*

The study of operating systems will be from the user's point of view rather than at a technical level. Although for some students it might be appropriate to study the operating system of the configuration with which they are familiar, it is probable that most students will study this section of the syllabus at a logical level and without reference to a particular machine.

Emphasis should be placed on the need for interrupts to enable the use of the central processor and its peripherals to be maximised and on how the software makes use of the interrupt facilities rather than on how the interrupts themselves are achieved at a hardware leve1.

It may be appropriate to illustrate the conflicts and priorities that exist by with some everyday task such as the preparation of a meal.

Paper 2: *The Use.of Computers*

SECTION A — GENERAL PRINCIPALS

1. *Data Structures*

Students should know how to create and to access tables and arrays in stores or on a backing store medium.

Exercises using flow charts should be set to demonstrate the principles involved, and program segments written to reinforce the procedures.

Files

The structure and organisation of files in core, on backing store media and on input and output media such as cards and paper tape should be studied.

The use of files, records and block terminators should be demonstrated. File labels and their contents and the part played by them in the system should be discussed.

The various modes of accessing direct access files should be described and the concept of index and overflow procedures dealt with. Flowcharting of the appropriate procedures can be carried out as exercises.

2. *Programming Languages*

The relationship between the hardware, the machine code and the assembly language should be described. For example, an instruction in an assembly language for the addition of a number in floating point form may generate a single machine code instruction if there is a hardware floating point unit, otherwise it may generate instructions to jump to a subroutine with appropriate parameters. Comparisons can also be made between the forms of addressing in assembly code and in machine code.

The teacher should select a limited sub—set of the assembly to demonstrate how

1. instructions are modified e.g. unconditional and conditional transfer of control,

2. addresses are represented e.g. absolute, indexed, modified, relative, deferred, immediate (where the address part is used as the operand), symbolic and extended,
3. sub—routines are entered and left,
4. programs are located and relocated,
5. directives are used,
6. software and software packages are used.

3. *Use of a High Level Language*

It is expected that the form and uses of various types of languages will be discussed. Apart from the high language discussed in detail, these may include program gencrators, machine tool control and the command languages for operating systems. (See also the notes on practical facilities and their use).

4. *Systematic Approach to Program Development and Documentation*

This section of the syllabus should be related to the practical work.

5. *A Survey of Current Applications*

In surveying current applications and the techniques involved in solving the problems associated with them it is intended that general principles should be discussed thus backing up the theory taught in other parts of the syllabus. Case studies for instance of six different problems and their solutions could be research and used for this purpose. Typical fields of study are accounting procedures, techniques appropriate to medicine using appropriate files and tables, banking procedures involving the use of document readers and files in general, public utility billing procedures using OCR, library procedures using special kinds of file indexing, and direct input of data into computer systems such as in the case of computer graphics or on—line solutions of mathematical problems involving data communications systems.

Questions set on this part of the syllabus will require students to apply their experience and knowledge derived from the case studies to other problems and so long as the solutions to the problems set are feasible and logical the examiners will accept them. Inventiveness and imagination are to be encouraged in this field.

SECTION B - PROBLEM SOLVING TECHNIQUES

6. *File Creation and Handling*

The procedures listed in the syllabus are commonly used in data processing. Students should study these using both flowcharts and programs to reinforce their understanding.

7. *Management Techniques in Information Systems*

The techniques used in systems analysis and design coupled with *overall* systems flowcharts should be dealt with.

Types of processing, the hardware used, the media used and the file and data organisation should be discussed.

In this section, pseudo—real time techniques should be included together with the Datel facilities made available by the G.P.O.

8. *Accuracy of Calculations*

Students should be taught that, in general, every arithmetical operation in a computer introduces a rounding off error. With floating point binary arithmetic, the basic equation

f1 (x * y) = (x * y) (1 + ϵ)

should be justified where f1 denotes the calculated value of performing the single operation • which may be either +, -, × or ÷ and ϵ is the relative error. The limit for |ϵ| depending on the word length and methods of rounding off should be discussed.

The effect of the arithmetical rounding off errors in calculating simple expressions by alternative formulae should be considered:

e.g. x(y - z) and xy - xz,

or, (√x - √y) and (x - y)/(√x + √y).

The effects of error in intial data may be considered by examining the sensitivity of the solution of, say, a quadratic equation with two close roots or a pair of simultaneous ill—conditioned equations.

9. *Iterative Processes*

The concept of order of convergence may be illustrated with first and second order processes for solving, say, a cubic equation. The Newton—Raphson method should be derived as a second order process requiring derivatives to be calculated and this may be compared with the secant method which does not require the calculation of derivatives. The general condition that $x_{i+1} = f(x_i)$ converges to a solution of x = f(x) if |f'(x)| < 1 in the neighbourhood of the solution and if x is sufficiently near to the solution may be illustrated geometrically. If f'(x) = 0 at the solution the method should be shown to be at least quadratically convergent.

10. *Simulation*

The concepts of frequency distributions, histograms and pseudo—random numbers for simulation studies should be introduced.

Methods, e.g. congruence methods, of generating pseudo—random numbers which are equally distributed throughout a range (e.g. should be described, and possible difficulties associated with short cycles should be outlined. The need for simulating numbers from non—uniform distributions should be illustrated in examples and the simplest method of converting from uniformly distributed pseudo—random numbers to a distribution corresponding to a given histogram should be described.

The need for exhibiting results in the form of a histogram should be illustrated by examples. Methods of constructing histograms based on direct calculation of the address of the cell in the

case of cells of equal size, and the method of binary search for general histograms should be described.