

THE INSTITUTE OF COMMERCIAL MANAGEMENT

SUBJECT SYLLABUS



**Hardware and Operating systems**

**Unit code: HOS-0908**

**Level: Advanced Diploma**

**Credits: 20**

**Unit leader: PY**

**Pre-requisites:**

**Main Aims of the Unit:**

This unit will provide an overview of hardware systems including technical specifications of input and output devices, memory and storage. Students should be able to produce a simple "pseudo code" to illustrate how data is handled in the processor. Further details are provided on operating systems and their function.

**Main Topics of Study:**

**A. Memory**

1. Types of main memory. ROM, RAM, DRAM, SRAM, EPROM, EEPROM, Registers, Cache.
2. Survey of relative speeds and costs of memory from fast registers to slow secondary memory (tape). Read and write cycle of memory.
3. Technical characteristics of types of main memory.
4. Memory maps.
5. Address bus. Operation codes. Reduced Instruction Set Codes (RISC). Decoding addresses.

**B. CPU**

1. Von Neumann model of the stored program.
2. Developments on this model – Pipelining, Array Processors, Multiprocessors.
3. Component parts of the CPU.
4. BIOS.
5. Data bus and address bus in relation to other parts of the computer system.
6. Registers. Accumulator(s), MAR, MBR, CIR, PC. Register transfer language (Listed below).
7. Simplified instruction set. A subset is listed at the end of this syllabus.
8. Shift and Mask instructions used to extract/combine parts of data in a single location.
9. Addressing modes – Immediate, Direct, Indirect, Indexed.

**C. Fetch-Execute Cycle**

1. Overview of the cycle.
2. Taken in turn, how each of the basic registers are used throughout the fetch-execute cycle. Use of register transfer language to describe these operations.
3. Complete analysis of a 3-instruction assembly routine as it passes through the fetch execute cycle – the analysis to be expressed in register transfer language.

**D. Complex memory structures**

1. Stack. Limitations of stacks. PUSH and POP. Recursion – simple examples such as determining factorial 5 ( $5! = 5 \times 4 \times 3 \times 2 \times 1$ ) by a recursive method.
2. Queue. Holding a queue in memory – circular queue using an array. Algorithm to add or remove an item from a queue.
3. Linked lists. Algorithms to locate, delete or insert an item using a linked list.
4. Binary tree. Algorithms to locate, delete or insert an item using a binary tree.

**E. Input devices**

1. Survey of current input devices available – keyboard, concept keyboard, barcode reader, optical mark sense reader, optical character reader, magnetic ink character reader, digitiser.
2. Technical aspects of how each of the above devices read their inputs. This includes being able to locate the position of the data on the capture form. Candidates need to see samples.
3. Buffers. Double buffering.

**F. Output Devices**

1. Survey of the range of printers. Comparison of the features offered by each type of printer.
2. Technical aspects of how each type converts the data it receives into printed output.
3. VDU. Video RAM. Colour. Raster scanning. Graphics card.
4. Plotters - Flat bed, drum. Their operations.

**G. Storage devices and media**

1. Difference between device and medium. Role of the buffer in data transfer.
2. Survey of storage devices. Magnetic tape, cartridge and tape streamer. Diskette, hard disk, Winchester. USB pen drive. Range of optical devices – CD and DVD.
3. Definitions of storage terms – file, record, field, cylinder, track, sector, header label, inter-block gap. File allocation table (FAT).
4. How data is physically stored on a storage medium.
5. How each of the storage devices read and records data.
6. Data transfer checks. Parity and its purpose. Odd/even parity - be able to give numeric examples of parity checking showing acceptance/failure. Cyclic redundancy checks.

**H. Logic Devices and Boolean algebra**

1. AND, OR, NOT. Truth tables. Karnaugh maps.
2. Boolean algebra laws – associative, commutative, de Morgan's Theorem.
3. Simplification of Boolean expressions and proving identities using truth tables, Karnaugh maps, and algebraic manipulation.
4. Devise simple circuits for Boolean expressions. Half and Full Adders.
5. Conversion of expressions into NAND or NOR units.
6. Flip-flops and their uses.

**G. Operating System – (Note: Basic commands are covered in GENERAL PURPOSE SOFTWARE module).**

1. Range of operating systems now in use – DOS, Windows and UNIX. Means of control using GUI or commands.
2. Purpose of the operating system. General survey of features offered.
3. Interrupt handling. Priorities. Interrupt vector table.
4. Scheduling – high, medium and low level.
5. Memory management. Partitioning. Paging. Virtual memory.
6. File management.
7. Multiprogramming. Concurrent processing. Deadlock and locking.

**Learning Outcomes for the Unit:**

At the end of this Unit, students will be able to:

1. Explain how input and output devices work
2. Evaluate a range of storage devices and methods of data handling
3. Understand how internal logic is handled
4. Describe the primary functions of operating systems and how they are carried out.
5. Produce simple algorithms for data handling

The numbers below show which of the above module learning outcomes are related to particular cognitive and key skills:

Knowledge & Understanding 1-5  
 Analysis 3, 5  
 Synthesis/Creativity 5  
 Evaluation 2, 4  
 Interactive & group Skills -  
 Self-appraisal/Reflection on Practice -  
 Planning and Management of Learning -  
 Problem Solving 4, 5  
 Communication & Presentation -  
 Other skills (please specify) -

### **Learning and teaching methods/strategies used to enable the achievement of learning outcomes:**

Learning takes place on a number of levels through lectures, class discussion including problem review and analysis. Formal lectures provide a foundation of information on which the student builds through directed learning and self managed learning outside of the class. The students are actively encouraged to form study groups to discuss course material which fosters a greater depth learning experience.

### **Assessment methods weightings which enable students to demonstrate the learning outcomes of the Unit:**

3 hour examination: 100%  
(Answer any 5 questions from 8, each question representing 20% of the paper.)

### **Main text**

Refer to the ICM website for notes on this subject

### **Alternative Texts and Further Reading**

The Principles of Computer Hardware by A Clements – (Oxford University Press).  
ISBN 019 856454 6 (Third edition).

Computer Science for Advanced Level by R Bradley – (Stanley Thornes )  
ISBN 0 7487 4046 5 (Fourth edition).

Operating Systems incorporating UNIX and WINDOWS by Colin Ritchie (Letts)  
ISBN 1 85805 302 1 (Third Edition)

### **Guideline for Teaching and Learning Time (10 hours per credit)**

Lectures / Seminars / Tutorials / Workshops: 50 hours  
Tutorial support includes feedback on assignments and may vary by college according to local needs and wishes.

Directed learning: 50 hours  
Advance reading and preparation / Class preparation / Background reading / Group study / Portfolio / Diary etc

Self managed learning: 100 hours  
Working through the course text and completing assignments as required will take up the bulk of the learning time. In addition students are expected to engage with the tutor and other students and to undertake further reading using the web and/or libraries.

## Guidelines

- This module covers the TECHNICAL aspects of hardware. Candidates should be aware that user actions are likely to feature only as a MINOR part in any question. Tutors should consult technical books for details. It is not intended for a casual user of hardware devices. Two excellent books are listed below.
- Some notes are provided for a typical instruction set which could be used relating to the fetch-execute cycle. The instruction set only considers NUMERIC data. If candidates use any other instruction set, they will be required to define each instruction they use OR a copy of the instruction set must be included with EACH batch of entries for this module. See the list below.
- Some previous candidates found it difficult to describe exactly what happens at each step during a fetch-execute cycle. So, Register Transfer Language (RTL) has been introduced to enable operations within the fetch-execute to be described concisely and precisely. See the list below.
- Candidates should be able to produce an algorithm (pseudo-code is preferred) for manipulation of queues, linked-list and binary trees. This might best be achieved by providing students with one pseudo-code answer (e.g. add item to a linked list) to show the expected format and requiring the student to adapt it to one of the others.

## NOTES

### 1. Register Transfer Language (RTL)

RTL is actually a notation rather than a language. There are four main elements of it.

- [x] Refers to the CONTENTS of register x.
- M(a) Refers to the CONTENTS of MEMORY with address a
- [p] <- [q] Transfer contents of register q to register p.
- M(a) = 5 Memory with address a CONTAINS the value 5. (Statement, not a transfer)

Examples of its use:

- |    |                        |   |
|----|------------------------|---|
| 1. | [PC] <- [PC] + 1       | Program counter incremented – e.g. after new instruction has been loaded from memory.   |
| 2. | [PC] <- [CIR(address)] | Address part of current instruction register is transferred to the PC after ALU has detected a TRUE condition for a BRANCH instruction.   |
| 3. | [CU] <- [CIR(op-code)] | Control unit receives the operation code part of the current instruction in order to decode it and determine how to execute this instruction.   |
| 4. | [MBR] <- [M(MAR)]      | The contents of memory whose address is currently in MAR is passed to MBR. This applies<br>EITHER in the fetch phase where the next instruction must be collected from memory to be passed to CIR<br>OR in the execution phase where data is collected from memory for processing in the ALU. |

All stages of the FETCH-EXCEUTE CYCLE can be described by a series of RTL statements. The first few statements WILL be the same for each instruction but once the op-code has been decoded, the execute-phase will dictate different actions.

This same convention is used below to describe the operation of each of the assembler instructions.

### 2. Assembler instruction set

This is a simplified instruction set suitable for illustrating features covered by this module. IF candidates use the following set, they will NOT be expected to define the instructions they use to illustrate their answers. If another set is used, an explanation of each different operation code used MUST be defined as part of the answer. Only NUMERIC data will be considered. RTL is used to describe the operation of each op-code.

## Assumptions:

1. Each instruction contains only ONE-ADDRESS.
2. The start of memory for an assembled program is:
  - Address 0 PC – program counter
  - Address 1 Single accumulator (acc) for use by the ALU.
3. Mnemonic addresses may be used. n can therefore be replaced by PRICE.
4. The single accumulator is implied in ALL instructions where it is applicable. Thus LDA PRICE means Load contents of Memory address PRICE into the accumulator.
5. ANY memory address can be used for indexed addressing. In which case, in the instruction set below showing Immediate Addressing, n will be replaced by n(M[n]) for Indexed Addressing. e.g. LDA PRICE(INDEX) Here PRICE effectively marks the start of a PRICE table (in consecutive memory locations) and INDEX marks which element of that table should be loaded into memory. INDEX will normally be set to zero to mark the start of the table (i.e. memory position PRICE + 0) and then incremented to move onto the next price in the table.

<u>Instruction</u>	<u>Action</u>	<u>Nominal description</u>	<u>Address method</u>
Instructions involving numbers into/out of accumulator			
LDN	n [Acc] <- n	Load number	(Immediate addressing)
LDA	n [Acc] <- [M(n)]	Load contents	(Direct addressing)
ADN	n [Acc] <- [Acc] + n	Add number	(Immediate)
ADD	n [Acc] <- [Acc] + [M(n)]	Add contents	(Direct)
SUN	n [Acc] <- [Acc] - n	Subtract number	(Immediate)
SUB	n [Acc] <- [Acc] - [M(n)]	Subtract contents	(Direct)
MUN	n [Acc] <- [Acc] x n	Multiply by number	(Immediate)
MUL	n [Acc] <- [Acc] x [M(n)]	Multiply by contents	(Direct)
DIN	n [Acc] <- [Acc] / n	Divide by number	(Immediate)
DIV	n [Acc] & [Acc] / [M(n)]	Divide by contents	(Direct)
STA	n [M(n)] & [Acc]	Store accumulator contents	(Direct)

## Branching instructions

BRN	n [PC] <- [CIR(address)]	Unconditional branch	
BZE	n [PC] <- [CIR(address)]	ONLY IF [Acc] = 0	Conditional branch
BNZ	n [PC] <- [CIR(address)]	ONLY IF [Acc] ≠ 0	Conditional branch
BGE	n [PC] <- [CIR(address)]	ONLY IF [Acc] = 0 OR [Acc] > 0	Conditional branch
BLE	n [PC] <- [CIR(address)]	ONLY IF [Acc] = 0 OR [Acc] < 0	Conditional branch
BGT	n [PC] <- [CIR(address)]	ONLY IF [Acc] > 0	Conditional branch
BLT	n [PC] <- [CIR(address)]	ONLY IF [Acc] < 0	Conditional branch

## Input / Output

REA	n [Acc] < {keyboard}	n has no function but must be present = Read number
WRI	n [Screen] <- [Acc]	Using n character positions on the screen = Write number
WNL	n [Screen]	Write n NEWLINES to screen.

Text data will not be considered

## Shift Instructions

SLL	n [Acc] <- [Acc]	LOGICAL Shift n bits to the LEFT
SLR	n [Acc] <- [Acc]	LOGICAL Shift n bits to the RIGHT
SAL	n [Acc] <- [Acc]	ARITHMETIC Shift n bits to the LEFT
SAR	n [Acc] <- [Acc]	ARITHMETIC Shift n bits to the RIGHT
SCL	n [Acc] <- [Acc]	CIRCULAR Shift n bits to the LEFT
SCR	n [Acc] <- [Acc]	CIRCULAR Shift n bits to the RIGHT

## Mask Instructions

AND	n	$[Acc] \leftarrow [Acc] \text{ AND } [M(n)]$	Only "1" in corresponding positions of Acc and memory carried forward.
OR	n	$[Acc] \leftarrow [Acc] \text{ OR } [M(n)]$	"1" is carried forward in each bit if there is a "1" in either corresponding positions of Acc and memory.
NOT	n	$[Acc] \leftarrow \text{NOT } [Acc]$	ALL Bits of accumulator are inverted.