NATIONAL OPEN UNIVERSITY OF NIGERIA

FACULTY OF SCIENCE

COURSE CODE: CIT411

COURSE TITLE:

MICROCOMPUTERS AND MICROPROCESSORS

COURSE GUIDE

CIT411
MICROCOMPUTERS AND MICROPROCESSORS

Course Team	Ikuvwerha, L. (Developer) - UNILAG
	Prof. Obidairo, K. (Programme Leader) - NOUN
	Afolorunso, A. A. (Coordinator) -NOUN

NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
14/16 Ahmadu Bello Way
Victoria Island
Lagos

Abuja Office
5, Dar es Salaam Street
Off Aminu Kano Crescent
Wuse II, Abuja
Nigeria

e-mail: centralinfo@noun.edu.ng
URL: www.noun.edu.ng

Published By:
National Open University of Nigeria

First Printed 2012

Reviewed and Reprinted 2020

ISBN: 978-058-838-8

CONTENTS PAGE

Introduction

CIT411 Microcomputers and Microprocessors is a two-unit course, which introduces you to operating modern microprocessor-based system on assembly language and hardware level. The course presents basic concepts of microcomputer architecture, microprocessor architecture and programmer's model, assembly programming, binary code representation, interfacing, testing and development.

This course is divided into four modules. The first module deals with the review of review of basic concepts in microprocessor, types of microprocessors, functions of microprocessor.

The second module treats extensively assembly languages. Instruction set, Addressing modes and Interrupts are also discussed.

The third module deals with microcomputers, its uses and functions. Microcomputers Architecture. Microcomputers networks and direct Memory Access.

This Course Guide gives you a brief overview of the course contents, course duration, and course materials.

What You Will Learn in This course

The main purpose of this course is to acquaint you with the basic concept of microprocessor and microcomputers. The function, features of microprocessor and microcomputers. This course reviews the basic concepts in digital electronic, microprocessors, functions, operations and architecture, comparison of current microprocessors, multi-chip and single chip; i/o organisation; assembler language; comparison of instruction sets; address modes, stack operation; subroutines. I/O data transfer; bus control; daisy chaining, handshaking and so on.

You will also learn about the concept of interrupt structures; programmed transfer, DMA microcomputer systems; types of microprocessors; uses of microprocessors, microcomputer design for specific applications; microcomputer networking; interfacing microcomputer real-time control; laboratory exercises using an assembly language.

Course Aims

The aim of this course is to acquaint you with the major concepts of microprocessors; this is a key component in computer hardware. This course also seeks to teach you how the various components of the computer works and their inter relationship from the processor to other units.

The overall goal of this course is to build the foundation for you to pursue the research in the areas of computers engineering.

Course Objectives

Certain objectives have been set out to ensure that the course achieves its aims. Apart from the course objectives, every unit of this course has set objectives. In the course of the study, you will need to confirm, at the end of each unit, if you have met the objectives set at the beginning of each unit. Upon completing this course, you should be able to:

- explain the microprocessor architecture and its components
- explain the programmer's model of a microprocessor
- explain the programmer's model of microprocessor
- explain 8051 instruction set
- design and implement assembly language programs using 8085 instruction set
- discuss the concept of assembly language and the process of creating executable files from the source code
- design and implement assembly programs dealing with interrupts
- initialise input/output ports and write assembly programs using i/o operations
- operate microprocessor (cross-) development system
- define computer resources in terms of program space and execution time.
- differentiate between hardware and software products on microprocessor market
- recognise basic components and functionality of microcomputer testing and development equipment.

Working through This Course

In order to have a thorough understanding of the course units, you will need to read and understand the contents.
You should do the exercises in the Tutor-Marked Assignments and submit to your tutors.

Course Materials

These include:

1. Course Guide
2. Study Units
3. Recommended Texts
4. A file for your assignments and for records to monitor your progress.

Study Units

There are three modules broken into 12 units in this course.

Module 1

Unit 1	Basic Concept of Microprocessors
Unit 2	Microprocessor Architecture
Unit 3	Types of Microprocessors
Unit 4	Microprocessors Operation and Functions

Module 2

Unit 1	Assembly Language
Unit 2	Instruction set and Addressing Modes
Unit 3	Interrupts and System Bus
Unit 4	Input and Out (I/O) Transfer

Module 3

Unit 1	Basic Concept OF Microcomputer
Unit 2	Microcomputers Architecure
Unit 3	Microcomputers Networking
Unit 4	Direct Memory Access

Textbooks and References

MacKenzie, I. S. (1995). *The 68000 Microprocessor*. Prentice Hall.

Rafiquzzaman, M. (1995). *Microprocessor- and Microcomputer-Based System Design* (2nd ed.). CRC Press.

Hall, D. & Rood, A. (1995). *Microprocessor and Interfacing: Programming and Hardware: 68000 versions*. McGraw-Hill.

Clements, A. (1992). *Microprocessor System Design: 68000 Hardware, Software, and Interfacing* (2nd ed.). PWS Kent.

Hilf, W. & Nausch, A. (1989). *The 68000 Family: Architecture, Addressing Modes and Instruction Set*. Prentice Hall.

Heffernan G. (2001). "8051 Tutorial"

Krishna Kumar, M. (2004). Microprocessors and Microcontrollers/Architecture of Microprocessors Lecture Notes.

Mack, P. E. (2005). "The Microcomputer Revolution" http// www. Clemson.edu/caah/history/Facultypages/Pammack/lec122/micro. htm

Kastner, D. (2003). "Embedded System"

Nasrat, H. "Introduction to Computer & Microcomputers"

Microprocessor Architecture "http://www.ehow.com/way_5581467_microprocessor-architecture-"tutorials.html

Microprocessor Design "http://en.wikibooks.org/wiki/Microprocessor_Design/Microprocessors"

Introduction to 8085 Architecture and Programming " http://en.wikibooks.org/wiki/ Introduction to 8085 Architecture and Programming"

http://www.ehow.com/about_6389869_direct-memory-access_.html

Assignment File

These are of two types: the self-assessment exercises and the tutor-marked assignment. The self-assessment exercises will enable you monitor your performance by yourself, while the tutor-marked assignment is a supervised assignment. The assignments take a certain percentage of your total score in this course. The tutor-marked assignment will be assessed by your tutor within a specified period. The examination at the end of this course will aim at determining the level of mastery of the subject matter. This course includes 12 tutor-marked assignment and each must be done and submitted accordingly. Your best scores however, will be recorded for you. Be sure to send these assignments to your tutor before the deadline to avoid loss of marks.

Presentation Schedule

The Presentation Schedule included in your course materials gives you the important dates for the completion of tutor-marked assignment and attending tutorials. Remember, you are required to submit all your assignments by the due date. You should guard against lagging behind in your work.

Assessment

There are two aspects to the assessment of the course. First are the tutor-marked assignments; second, is a written examination.

In tackling the assignments, you are expected to apply information and knowledge acquired during this course. The assignments must be submitted to your tutor for formal assessment in accordance with the deadlines stated in the Assignment File. The work you submit to your tutor for assessment will count for 30% of your total course mark.

At the end of this course, you will need to sit for a final three-hour examination. This will also count for 70% of your total course mark.

Tutor-Marked Assignment (TMA)

There are 22 tutor-marked assignments in this course. You need to submit all the assignments. The total marks for the best three (3) assignments will be 30% of your total course mark.

Assignment questions for the units in this course are contained in the Assignment File. You should be able to complete your assignments from the information and materials contained in your set textbooks, reading and study units. However, you may wish to use other references to broaden your viewpoint and provide a deeper understanding of the subject.

When you have completed each assignment, send it together with form to your tutor. Make sure that each assignment reaches your tutor on or before the deadline given. If, however, you cannot complete your work on time, contact your tutor before the assignment is done to discuss the possibility of an extension.

Examination and Grading

The final examination for the course will carry 70% percentage of the total marks available for this course. The examination will cover every aspect of the course, so you are advised to revise all your corrected assignments before the examination.

This course endows you with the status of a teacher and that of a learner. This means that you teach yourself and that you learn, as your learning capabilities would allow. It also means that you are in a better position to determine and to ascertain the what, the how, and the when of your language learning. No teacher imposes any method of learning on you.

The course units are similarly designed with the introduction following the contents, then a set of objectives and then the dialogue and so on.
The objectives guide you as you go through the units to ascertain your knowledge of the required terms and expressions.

Course Marking Scheme

This table shows how the actual course marking is broken down.

Table 1: Course Marking Scheme

Assessment	**Marks**
Assignment 1- 4	Four assignments, best three marks of the four count at 30% of course marks
Final Examination	70% of overall course marks
Total	100% of course marks

Course Overview

Unit	**Title of Work**	**Weeks Activity**	**Assessment (End of Unit)**
	Course Guide		
	Module 1		
1	Basic Concept of Microprocessors		
2	Microprocessor Architecture		
3	Types of Microprocessors		
4	Microprocessors Operation and Functions		
	Module 2		
1	Assembly Language		
2	Instruction set and Addressing		

	Modes		
3	Interrupts and System Buses		
4	Input and Output (I/O) Transfer		
Module 3			
1	Basic Concept of Microcomputers		
2	Microcomputer Architecture		
3	Microcomputer Networking		
4	Direct Memory Access		
	Revision		
	Examination		
	Total		

How to Get the Best from This Course

In distance learning the study units replace the university lecturer. This is one of the great advantages of distance learning; you can read and work through specially designed study materials at your own pace, and at a time and place that suit you best. In the same way that a lecturer might give you some reading to do, the study units tell you when to read your set books or other material. Just as a lecturer might give you an in-class exercise, your study units provide exercises for you to do at appropriate points.

Each of the study units follows a common format. The first item is an introduction to the subject matter of the unit and how a particular unit is integrated with the other units and the course as a whole. Next is a set of learning objectives. These objectives enable you know what you should be able to do by the time you have completed the unit. You should use these objectives to guide your study. When you have finished the units you must go back and check whether you have achieved the objectives. If you make a habit of doing this you will significantly improve your chances of passing the course.

Remember that your tutor's job is to assist you. When you need help, do not hesitate to call and ask your tutor to provide it.

1. Read this Course Guide thoroughly.
2. Organise a study schedule. Refer to the 'Course Overview' for more details. Note the time you are expected to spend on each unit and how the assignments relate to the units. Whatever method you chose to use, you should decide on it and write in your own dates for working on each unit.
3. Once you have created your own study schedule, do everything you can to stick to it. The major reason that students fail is that they lag behind in their course work.

4. Turn to Unit 1 and read the introduction and the objectives for the unit.
5. Assemble the study materials. Information about what you need for a unit is given in the 'Overview' at the beginning of each unit. You will almost always need both the study unit you are working on and one of your set of books on your desk at the same time.
6. Work through the unit. The content of the unit itself has been arranged to provide a sequence for you to follow. As you work through the unit you will be instructed to read sections from your set books or other articles. Use the unit to guide your reading.
7. Review the objectives for each study unit to confirm that you have achieved them. If you feel unsure about any of the objectives, review the study material or consult your tutor.
8. When you are confident that you have achieved a unit's objectives, you can then start on the next unit. Proceed unit by unit through the course and try to pace your study so that you keep yourself on schedule.
9. When you have submitted an assignment to your tutor for marking, do not wait for its return before starting on the next unit. Keep to your schedule. When the assignment is returned, pay particular attention to your tutor's comments, both on the tutor-marked assignment form and also written on the assignment. Consult your tutor as soon as possible if you have any questions or problems.
10. After completing the last unit, review the course and prepare yourself for the final examination. Check that you have achieved the unit objectives (listed at the beginning of each unit) and the course objectives (listed in this Course Guide).

Summary

To design an electronic system, you need to know the difference between a microcontroller and a microprocessor. Knowing the difference will enable you decide which is the best choice to design an electronic system with. The microprocessor is the functional part of a microcomputer, using all of the various parts of the larger system. It reads binary instructions from the microcomputer's memory, accepts binary information from the input, processes that information and provides the results as output.

The course was planned and written to ensure that you acquire the proper knowledge and skills for the appropriate situations. The essence is to help you in acquiring the necessary knowledge and competence by equipping you with the necessary tools to accomplish this.

We hope that by the end of this course you would have acquired the required knowledge on microprocessors, Assembly language, Instruction sets and microcomputers.

We wish you success with the course and hope that you will find it both interesting and useful.

Course Code	CIT411
Course Title	Microcomputers and Microprocessors
Course Team	Ikuvwerha, L. (Developer) - UNILAG Prof. Obidairo, K. (Programme Leader) - NOUN Afolorunso, A. A. (Coordinator) -NOUN

NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
14/16 Ahmadu Bello Way
Victoria Island
Lagos

Abuja Office
5, Dar es Salaam Street
Off Aminu Kano Crescent
Wuse II, Abuja
Nigeria

e-mail: centralinfo@noun.edu.ng
URL: www.noun.edu.ng

Published By:
National Open University of Nigeria

First Printed 2012

ISBN: 978-058-838-8

CONTENTS **PAGE**

MODULE 1

UNIT 1 BASIC CONCEPT OF MICROPROCESSORS

CONTENTS

1.0 INTRODUCTION

On a personal computer, the CPU (the control unit, arithmetic/logic unit and the inside memory) is usually contained on a single chip and sometimes is called a **microprocessor**. It is often identified by its model name or model number. From the technological viewpoint, a microprocessor is a VLSI (Very Large Scale Integration) single-chip implementation of a complete processing unit. Microprocessors are capable of accepting coded instructions for execution in **16-, 32- or even 64-bit word formats**, that is in groups of bits, bytes or characters, considered as an entity, and capable of being stored in one memory location.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain what a microprocessor is
- give historical development of the microprocessors
- discuss technological innovations of microprocessors.

3.0 MAIN CONTENT

3.1 Description of Microprocessor

The brain or engine of the PC is the processor (sometimes called microprocessor), or central processing unit (CPU). The CPU performs the system's calculating and processing. A microprocessor is an integrated circuit that holds most of the works of a computer. It can fetch an instruction, process it and delivered an output. When there is program memory, RAM, and other support circuitry on the chip, this is a micro-controller. Depending on how much data can be processed at once, this can be a 4, 8, 16, 32 (and more) bit processor.

The processor is usually the most expensive component in the system, costing up to four or more times greater than the motherboard it plugs into. Intel is generally credited with creating the first microprocessor in 1971 with the introduction of a chip called the 4004. Today, Intel still has control over the processor market, at least for PC systems. This means that all PC-compatible systems use either Intel processors or Intel-compatible processors from a handful of competitors (such as AMD or Cyrix). Integrated circuits are miniature complex circuits consisting of semiconductor devices, mainly diodes and transistors, interconnected with various passive elements. All these elements are formed upon or within a semiconductor, substrate called a *die.* The circuit configuration so formed is called a **chip,** the terms "chip" and "die" being often considered synonymous.

3.2 History of Microprocessor

The historical development of the microprocessors can document increases in clock speed (the speed at which a processor executes instructions) and number of transistors in chips. Since 1982, **Intel** has been the leading manufacturer of processors. They used a model number to identify the various chips (Intel 8088, 80286, 80386, 80486). Later, they decided to identify their microprocessors with names: **Pentium®** processors, **Celeron TM** processors (less expensive PCs) and **XeonTM** processors (workstations, servers). Pentium has been developed to be available with clock frequencies of up to 155 MHz and it uses a speedy version of CISC (Complex Instruction Set Computer) technology and a 64-bit internal data bus. Users of multimedia applications should obtain an Intel processor equipped with MMXTM technology, in which a set of instructions is built in the processor so it can manipulate and process multimedia data more efficiently.

An alternative to the Intel-style microprocessor is the ***Motorola*** microprocessor, which is in Apple Macintosh and Power Macintosh systems. The RISC (Reduced Instruction Set Computer) processor used in Apple's PowerPC introduced a new architecture that increased the speed of the processor because it had been designed to understand a relatively small number of basic instructions. The ***Alpha*** microprocessor, which was developed by Digital Equipment Corporation, is used primarily in workstations and high-end servers. Current models of the Alpha chip run at clock speeds from 300 to 600 MHz.

3.3 Technological Innovations of Microprocessors

Some of the technological innovations of microprocessors include the following.

1. **Digital signal processors (DSPs):** A specific new class of microprocessors intended for processing analog signals by sampling their momentary values and using digital processing methods for this
2. **Transputers:** This is 32-bit microprocessors with a memory and direct links for connection to other transputer units cooperating in a network system.
3. ***Programmable logic devices*** **(PLDs):** complex programmable logic systems consisting of several logic blocks connected via a programmable interconnect matrix.
4. **Fifth-generation computers (5G-computers):** They have CPU replaced by a *problem-solving and interference machine* (or called *interference engine, control structure, rule interpreter*) which enables dialogue between a computer and its user, often in a natural language, for instance.

A **coprocessor** is a special processor chip or circuit board that assists the processor in performing specific tasks. Users running engineering, scientific, or graphic applications, for instance, will notice a dramatic increase in speed applications that take advantage of a floating-point coprocessor (sometimes called math or numeric coprocessors).

When a coprocessor is not connected in the system and the functions of the coprocessor is done by the CPU by writing coprocessor instructions from a predetermined location in the form of a software routine and invoked by interrupt, it is known as '**coprocessor trap**'. There are three fields in a coprocessor instruction—these are code, address, and opcode. The code field distinguishes between the coprocessor and CPU instructions. The address field indicates the address of a particular coprocessor. This is required when several coprocessors are used in the system. The opcode field indicates the type of operation to be executed by the processor.

4.0 CONCLUSION

Microprocessor is the brain of the PC. It is the most expensive component in the system. It is interesting to note that the microprocessor had only existed for 10 years prior to the creation of the PC! The microprocessor was invented by Intel in 1971. The PC was created by IBM in 1981. Some of the technological innovations of microprocessors are: *Digital signal processors* (DSPs), *Transputers*, *Programmable logic devices* (PLDs), Fifth-generation computers (5G-computers) and a coprocessor.

5.0 SUMMARY

In this unit, we have discussed the component of the microprocessor; its history as well as its technological innovations. The brain or engine of the PC is the processor (sometimes called microprocessor), or central processing unit (CPU). The CPU performs the system's calculating and processing. The historical development of the microprocessors was also discussed in this unit.

6.0 TUTOR-MARKED ASSIGNMENT

1. What is a microprocessor?
2. When was the first microprocessor developed?
3. What is a coprocessor trap?
4. Explain the three fields contained in a coprocessor instruction.

7.0 REFERENCES/FURTHER READING

Rafiquzzaman, M. (1995). *Microprocessor- and Microcomputer-Based System Design* (2nd edition). CRC Press.

Heffernan, G. (2001). "8051 Tutorial"

UNIT 2 MICROPROCESSORS ARCHITECTURE

CONTENTS

1.0 INTRODUCTION

This unit looks at the architectural design of a microprocessor. The 8085 microprocessor is used to describe the architectural design of microprocessor. The function features and the various components of a microprocessor are also examined in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- describe the component of a microprocessor
- explain microprocessor architectural design.

3.0 MAIN CONTENT

3.1 Microprocessor Architectural Design

The salient features of 8085 microprocessor are listed below.

- It is 8-bit microprocessor.
- It is manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to 216 = 65536 bytes (64KB) memory locations through A_0-A_{15}.
- The first eight lines of address bus and eight lines of data bus are multiplexed $AD_0 – AD_7$.
- Data bus is a group of eight lines $D_0 – D_7$.
- It supports external interrupt request.

- A 16-bit program counters (PC)
- A 16-bit stack pointer (SP)
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- It is enclosed with 40 pins DIP (Dual in line package). The architectural structure of 8085 microprocessor is shown below. The components of the 8085 will be discussed in section 3.2.

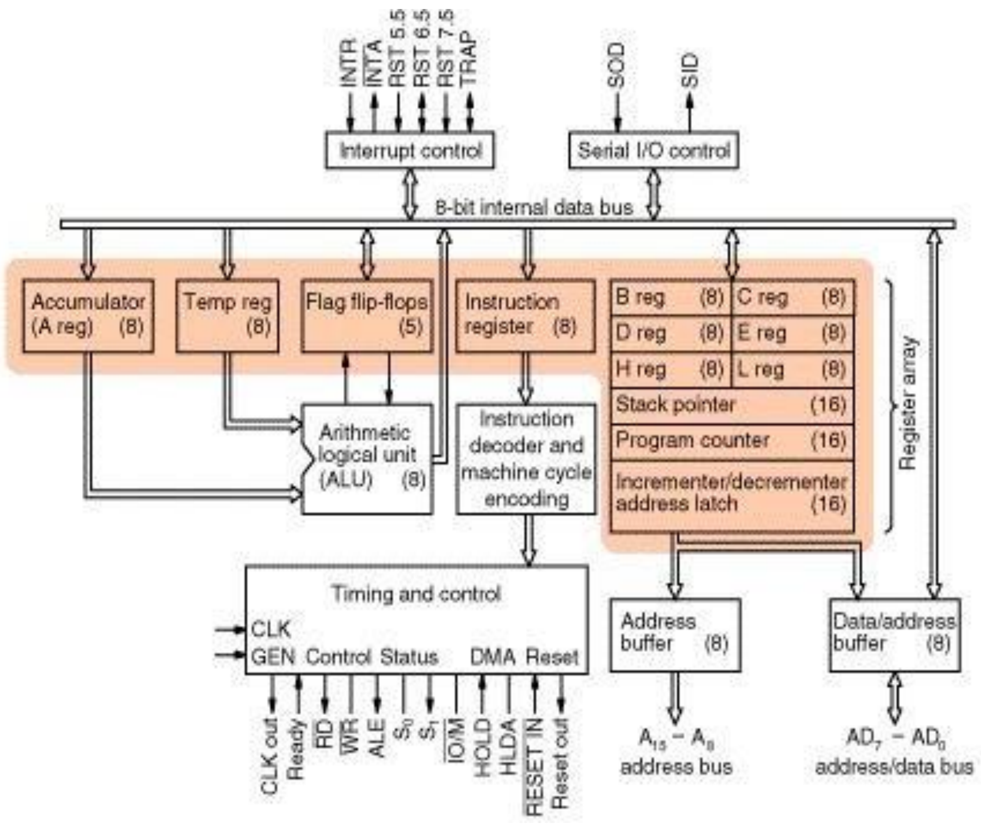


Figure 1: Architecture of 8085 Microprocessor

3.2 Components of 8085 Microprocessor

1. Control Unit

Generates signals within uP to carry out the instruction, which has been decoded. In reality causes certain connections between blocks of the uP to be opened or closed so that data goes where it is required, and so that ALU operations occur.

2. Arithmetic Logic Unit

The ALU performs the actual numerical and logic operation such as 'add', 'subtract', 'AND', 'OR', an so on. Uses data from memory and from accumulator to perform arithmetic. Always stores result of operation in accumulator.

3. Registers

The 8085/8080A-programming model includes six registers, one accumulator, and one flag register, as shown in Figure. In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows.

The 8085/8080A has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L as shown in the figure1. They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

4. Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations.

The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

5. Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; they are listed in the Table and their bit positions in the flag register are shown in the Figure below. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions. For example, after an addition of two numbers, if the sum in the accumulator id larger than eight bits, the flip-flop uses to indicate a carry -- called the Carry flag (CY) – is set to one. When an arithmetic operation results in zero, the flip-flop called the Zero (Z) flag is set to one. The first Figure shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five-bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the

8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction. These flags have critical importance in the decision-making process of the microprocessor.

The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

6. Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location

7. Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer. The stack concept is explained in the chapter “Stack and Subroutines.”

8. Instruction Register/Decoder

This is a temporary store for the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and ‘decodes’ or interprets the instruction. Decoded instruction then passed to next stage.

9. Memory Address Register

This component holds address, received from PC, of next program instruction. It also feeds the address bus with addresses of location of the program under execution.

10. Control Generator

This generates signals within uP to carry out the instruction which has been decoded. It causes certain connections between blocks of the uP to be opened or closed, so that data goes where it is required, and ALU operations can be carried out.

10. Register Selector

This block controls the use of the register stack in the example. It is a logic circuit, which switches between different registers in the set will receive instructions from control unit.

11. General Purpose Registers

uP requires extra registers for versatility. It can be used to store additional data during a program. More processors that are complex may have a variety of differently named registers.

12. Microprogramming

How does the µP knows what an instruction means, especially when it is only a binary number? The microprogram in a uP/uC is written by the chip designer and tells the uP/uC the meaning of each instruction uP/uC can then carry out operation.

3.3 Microprocessor Interfacing Technique

A microprocessor would not be of much use by itself. To perform useful work, it needs to be connected to other electronic components. To design a computer, a microprocessor must be interfaced to main memory, a graphic subsystem, disk memory, the keyboard and USB ports, to say the least.

For the design of industrial automation systems such as alternative energy system controller, the microprocessor will need to be interfaced to a host of electromechanical devices and sensors. For the design of supercomputers, microprocessors, the microprocessor must be interfaced to banks of not just 10 or 20, but thousands of other microprocessors.
Related Searches:

1. Interface Fundamentals

A number of intermediary electronic circuits are needed to interface a microprocessor to another component. Common microprocessor interface components include a peripheral interface controller (PIC), an interrupt controller, and drivers also known as **buffers**. Glue logic, a mish-mash of logic gates, is also often used to interface microprocessors.

2. 8255 Peripheral Interface Controller

The 8255 Peripheral Interface Controller is the essential interface used with 8086 microprocessor Common 8255 interface designs are stepper motors and digital to analog converter interfaces.

The 8255 peripheral interface controller takes signals from the 8086 microprocessors and redirects those signals to its own internal ports. The ports are directly connected to the peripheral to be controlled.

There are a number of control lines that connect directly between the 8086 and the 8255. These control lines are used to enable the 8255's ports for a read or write operation. The 8086 addresses the 8255 through the 8086 address lines and feeds it 8-bit data through the 8086 data bus.

4.0 CONCLUSION

The 8085 microprocessor is used to describe the architectural design of microprocessor. The features of 8085 microprocessor such as the flags stack pointer, general registers, and so on. Each of the components is discussed above and show in figure 1.

5.0 SUMMARY

In this unit, you have learnt the salient features of 8085 microprocessor, which are:

- It is 8-bit microprocessor
- It is manufactured with N-MOS technology
- It has 16-bit address bus and hence can address up to 216 = 65536 bytes (64KB) memory locations through A_0-A_{15}
- The first 8 lines of address bus and 8 lines of data bus are multiplexed AD_0 – AD_7
- Data bus is a group of 8 lines D_0 – D_7.

Some of the Components of 8085 Microprocessor

1. Control Unit
2. Arithmetic Logic Unit

3. Registers
4. Accumulator
5. Flags
6. Program Counter (PC)
7. Stack Pointer (SP)
8. Instruction Register/Decoder
9. Memory Address Register

6.0 TUTOR-MARKED ASSIGNMENT

Explain the architecture of microprocessor and their operation.

7.0 REFERENCES/FURTHER READING

Heffernan G. (2001). "8051 Tutorial"

Krishna Kumar, M. (2004). "Microprocessors and Microcontrollers/Architecture of Microprocessors Lecture Notes."

Mack, P. E. (2005). "The Microcomputer Revolution" http// www. Clemson.edu/caah/history/Facultypages/Pammack/lec122/micro. htm.

UNIT 3 TYPES OF MICROPROCESSORS

CONTENTS

1.0 INTRODUCTION

This unit explains types of microprocessors based on Hardware characteristic (RISC, CISC VLIW AND SUPERSCALE) and characteristic of application areas (General Purpose Processor and Special Purpose Processor). The types of microprocessor will also be examined using the Intel and Motorola microprocessors. The comparison of microprocessor is also discussed.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain the types of microprocessor based on the hardware characteristic and applications areas
- discuss various types of microprocessor especially the Intel and Motorola
- compare various types of microprocessor base on their features
- explain the differences in microprocessor architecture.

3.0 MAIN CONTENT

3.1 Types of Microprocessor Based on Hardware Characteristics and Application Areas

The types of Microprocessor are discussed based on two classification criteria:

1. hardware characteristics

- RISC
- CISC
- VLIW
- Superscalar

2. characteristics of application areas

- GPP (General Purpose Processor) / MCU (MicroController Unit)
- SPP (Special Purpose Processor)

i. ASIC (Application-Specific Integrated Circuit)
ii. ASIP (Application-Specific Instruction-set Processor)
iii. DSP (Digital Signal Processor)

Features of the Various Hardware Characteristic

1. Complex Instruction Set Computer (CISC)

- large number of complex addressing modes
- many versions of instructions for different operands
- different execution times for instructions
- few processor registers
- microprogrammed control logic

2. Reduced Instruction Set Computer (RISC)

- one instruction per clock cycle
- memory accesses by dedicated load/store instructions
- few addressing modes
- hard-wired control logic

3. Very Long Instruction Word (VLIW)

- statically determined instruction-level parallelism (under compiler control)
- instructions are composed of different machine operations whose execution is started in parallel
- many parallel functional units
- large register sets

4. Superscalar Processors

- subclass of RISCs or CISCs
- multiple instruction pipelines for overlapping execution of
- instructions
- parallelism not necessarily exposed to the compiler

Features based on Application Areas

General Purpose versus Specific Use

Microprocessors that are capable of performing a wide range of tasks are called **general purpose microprocessors**. General purpose microprocessors are typically the kind of CPUs found in desktop computer systems. These chips typically are capable of a wide range of tasks (integer and floating point arithmetic, external memory interface, general I/O, and so on). We shall discuss some other types of processor units available.

General Purpose

A general purpose processing unit, typically referred to as a "microprocessor" is a chip that is designed to be integrated into a larger system with peripherals and external RAM. These chips can typically be used with a very wide array of software.

DSP

A Digital Signal Processor, or DSP for short, is a chip that is specifically designed for fast arithmetic operations, especially addition and multiplication. These chips are designed with processing speed in mind, and do not typically have the same flexibility as general purpose microprocessors. DSPs also have special address generation units that can manage circular buffers, perform bit-reversed addressing, and simultaneously access multiple memory spaces with little to no overhead. They also support zero-overhead looping, and a single-cycle multiply-accumulate instruction. They are not typically more powerful than general purpose microprocessors, but can perform signal processing tasks using far less power (as in watts).

Embedded Controller

Embedded controllers or "microcontrollers" are microprocessors with additional hardware integrated into a single chip. Many microcontrollers have RAM, ROM, A/D and D/A converters, interrupt controllers, timers, and even oscillators built into the chip itself. These controllers are designed to be used in situations where a whole computer system is not available, and only a small amount of simple processing needs to be performed.

Programmable State Machines

The most simplistic of processors, programmable state machines are a minimalist microprocessor that is designed for small and simple operations. PSMs typically have very small amount of program ROM available, limited scratch-pad RAM, and they are also typically limited in the type and number of instructions that they can perform. PSMs can either be used stand-alone, or (more frequently) they are embedded directly into the design of a larger chip.

Graphics Processing Units

Computer graphics are so complicated that functions to process the visuals of video and game applications have been offloaded to a special type of processor known as a GPU. GPUs typically require specialized hardware to implement matrix multiplications and vector arithmetic. GPUs are typically also highly parallelized, performing shading calculations on multiple pixels and surfaces simultaneously.

3.2 Comparison of Microprocessor Intel and Motorola Microprocessors

1. Intel Microprocessors

The Early Intel Microprocessors

The first microprocessor to appear in the market was the Intel **4004**, a 4-bit data bus device. This device was followed by the **8008**, which had an 8-bit data bus. Two more 8-bit microprocessors (reference to the number of bits usually refers to the data bus unless stated otherwise), the **8080** and **8085** were introduced in the mid-1970s. These two devices could address only 216 memory locations.

The 80X86 Family of Microprocessors

Since its introduction in 1978, the so-called X86 architecture has undergone five major evolutionary stages. The term *architecture* in relation to microprocessors refers to the internal design and organisation of the device.

The first generation of the 80X86 family includes the **8086**, the **8088**, and the **80186**. Next, came the **80286**, followed by the **80386**, and then the **80486**. The **Pentium** is the fifth generation Intel microprocessor. Each generation built upon the basic concept of the first additional features and improved performance.

Intel 8086/8088 and 80186

Introduced in 1978, the **8086** was the first 80X86 family and is the basis for all Intel microprocessors that followed. The 8086 was a 16-bit microprocessor (16-bit data bus) and represented a significant departure from the earlier 8-bit devices. It owned 20 address lines (allowing 220 memory locations to be accessed). The various versions of the 8086 operated at clock frequencies of %, 8, or 10MHz.

The **8088** is essentially an 8086 with the 16-bit internal data bus multiplied down to an 8-bit external bus. It was intended to meet the demand for applications in simpler 8-bit systems and was used in the original IBM personal computer (PC).

The **80186** is an 8086 with several support functions such as clock generator, system controller, interrupt controller, and direct memory access (DMA) controller integrated on the chip. An increased clock frequency of 12.5MHz was added and the 5MHz available in the 8086/8088 was dropped, resulting in a selection of 8, 10, or 12.5MHz.

Intel 80286

Introduced in 1982, the memory addressing capability was increased to 24 address lines. First Intel processor to include an advanced mode of operation (used in the next generations of microprocessors) **- protected mode**. This mode allows access to additional memory locations and advanced programming features. It operates at the same clock frequencies as the 80186.

Intel 80386

Introduced in 1985, it was the first 32-bit Intel microprocessor (32-bit data bus + 32-bit address bus). The first Intel microprocessor to use instruction pipelining. All 80386 versions can operate in conjunction with math (floating-point) co-processors. More economical versions: 80386SX and 80386SL, in which the data bus is multiplied down to 16 bits and the address bus down to 24 bits. Versions that operate at clock frequencies of 16, 20, 25 and 33MHz are available.

Intel 80486

Introduced in 1989, it incorporates an 8Kbytes cache memory (shared for data and instruction).

The first Intel processor to present an internal floating-point unit (FPU). Different versions operate at clock frequencies from 25 to 100MHz (Intel 80486 DX4).

Pentium

Introduced in 1993, Pentium retains the 32-bit address bus of the 80486 but doubles the data bus to 64 bits. It presents two 8Kbytes cache memories (one for instruction, one for data) and dual pipeline method, known as superscalar architecture. At present, it operates at clock frequencies up to 1.75GHz, 20-stage pipeline, and 3-level cache memory architectures (**Itaniun**).

2. Motorola microprocessors

The Early Motorola Microprocessors

The first microprocessors from Motorola were all 8-bit devices (8-bit data bus). The **6800** appeared in 1975 with a clock frequency of 2MHz and capable of addressing 64 Kbytes of memory with a 16-bit address bus. In the **6802**, a 128Kbytes RAM was added for use in the place of some registers. The clock frequency was increased in the **6803** to 3.58MHz, and a UART (Universal Asynchronous Receiver/Transmitter) was added for serial communications. The last of the 8-bit microprocessors was the **6809** which offered an enhanced instruction set including a multiply instruction. All of the 8-bit devices retained the 16-bit address bus.

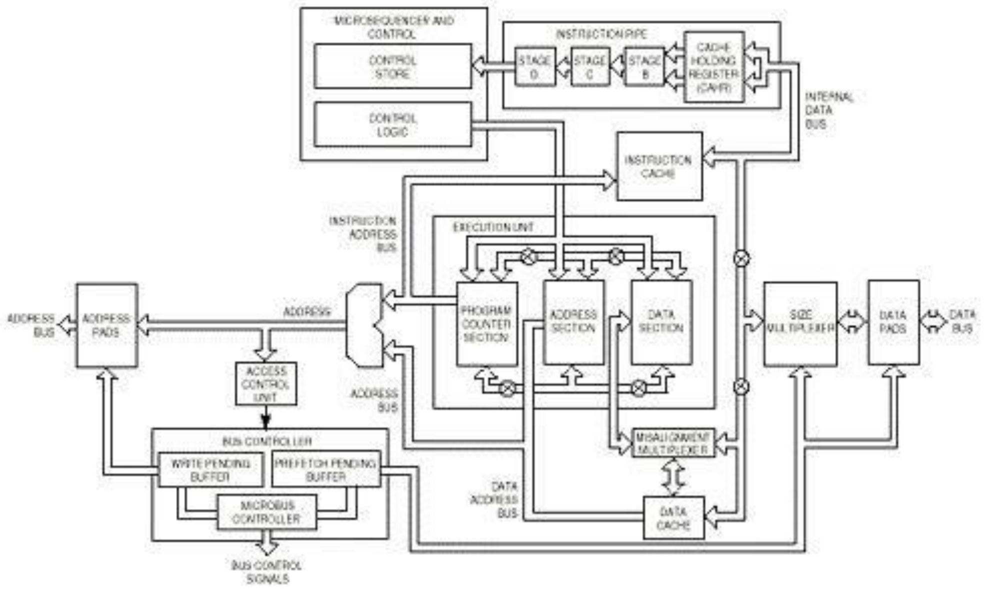


Figure 2: Motorola Microprocessor

The 680X0 Family of Microprocessors

The Motorola 680X0 family of microprocessors is split into five generations. It was started in 1978, with the MC68000, and was concluded in 1994 with the MC68060. The PowerPC family is the sixth generation of Motorola microprocessors.

Motorola 68020

The MC68020 provides a code-compatible upgrade path from the MC68000. It offers slightly reduced functionality at a lower cost. Motorola entered the world of 32-bit microprocessors with the 68020. 4 Gbytes Direct Linear Address Space.

The MC68020 featured a 256-byte cache memory for instructions, an innovative feature at that time.

10 MIPS @ 33 MHz.
Available in 12, 16, 20, 25, and 33 MHz.
MC68EC020 Available in 16 and 25 MHz.

Motorola 68030

Another 256-byte cache was added for data.
Internal Harvard Architecture.
On-Chip Memory Management Unit (MC68 030).

Burst Memory Interface.
The MC68EC030 offers a lower cost embedded solution by removing the memory management unit.

18 MIPS @ 50MHz.
Available in 16, 20, 25, 33, 40, and 50 MHz.
MC68EC030 Available in 25 and 40 MHz.

Motorola 68040

The data and instruction caches (separated) were increased to 4kbytes each.

Internal Harvard Architecture.
On-Chip Memory Management Unit.
Burst Memory Interface.

On-chip math coprocessor (floating point unit) was added in the 68040.

44 MIPS @ 40 MHz.
Available in 25, 33, and 40 MHz.

MC68040V is a Low Power (3.3V) Version of MC68LC040.

PowerPC 601/601v

The PowerPC 601/601v Microprocessor: The PowerPC 601 microprocessor is the first 32-bit implementation of the PowerPC Reduced Instruction Set Computer (RISC) architecture. The PowerPC 601 microprocessor provides high levels of performance for desktop, workstation, and symmetric multiprocessing computer systems and offers design flexibility through operation at either 2.5 volts (601v) or 3.6 volts (601).

Superscalar Microprocessor: The PowerPC 601 microprocessor is a superscalar design capable of issuing and retiring three instructions per clock. Instructions issue to multiple execution units, execute in parallel, and can complete out of order, while preserving program correctness. The PowerPC 601 integrates three execution units: an integer unit (IU), a branch processing unit (BPU), and a floating-point unit (FPU). It also incorporates a memory management unit (MMU), a unified instruction and data cache, a real-time clock (RTC), and on-chip test capability. The ability to execute multiple instructions in parallel and the use of simple instructions with rapid execution times yield maximum efficiency and throughput for PowerPC systems.

Cache and MMU Support: The PowerPC 601 microprocessor includes an on-chip, 32-Kbyte, eight-way set-associative, physically addressed, unified (instruction and data) cache.

An on-chip MMU contains 256-entry, two-way set-associative, unified (instruction and data) translation lookaside buffer (UTLB) and provides support for demand paged virtual memory address translation and variable-sized block translation.

Flexible Bus Interface: The PowerPC 601 microprocessor has a high bandwidth, 64-bit data bus and a separate 32-bit address bus. The interface protocol allows multiple masters to access system resources through a central external arbiter. Additionally, on-chip snooping logic maintains cache coherency in multiprocessor applications.

PowerPC 601 Major Features:

Specifications Summary
32-Kbyte unified cache
Superscalar-3 instructions per clock
Multiple execution
64-bit data bus
L2 cache

Power consumption: Full operation - 10 watts maximum at 80 MHz.

Technology

3.6-volt implementation
0.6-micron static CMOS technology
120 mm2 die size
2.8 million transistors Packaging
304 CQFP

3.3 Types of Use

Microcontrollers and Microprocessors are used for different types of applications. People may be the most familiar with the desktop PC, but the fact is that desktop PCs make up only a small fraction of all microprocessors in use today. We will list here some of the basic uses for microprocessors.

Signal Processing

Signal processing is an area that demands high performance from microcontroller chips to perform complex mathematical tasks. Signal processing systems typically need to have low latency, and are very deadline driven. An example of a signal processing application is the decoding of digital television and radio signals.

Real Time Applications

Some tasks need to be performed so quickly that even the slightest delay or inefficiency can be detrimental. These applications are known as "real time systems", and timing is of the upmost importance. An example of a real-time system is the anti-lock braking system (ABS) controller in modern automobiles.

Throughput and Routing

Throughput and routing is the use of a processor where data is moved from one particular input to an output, without necessarily requiring any processing. An example is an internet router, which reads in data packets and sends them out on a different port.

Sensor monitoring

Many processors, especially small-embedded processors are used to monitor sensors. The microprocessor will either digitise and filter the sensor signals, or it will read the signals and produce status outputs (the sensor is good, the sensor is bad). An example of a sensor monitoring processor is the processor inside an antilock brake system: This processor reads the brake sensor to determine when the brakes have locked up, and then outputs a control signal to activate the rest of the system.

General Computing

A general purpose processor is like the kind of processor that is typically found inside a desktop PC. Names such as Intel and AMD are typically associated with this type of processor, and this is also the kind of processor that the public is most familiar with.

Graphics

Processing of digital graphics is an area where specialised processor units are frequently employed. With the advent of digital television, graphics processors are becoming more common. Graphics processors need to be able to perform multiple simultaneous operations. In digital video, for instance, a million pixels or more will need to be processed for every single frame, and a particular signal may have 60 frames per second! To the benefit of graphics processors, the colour value of a pixel is typically not dependent on the values of surrounding pixels, and therefore many pixels can typically be computed in parallel.

3.4 Comparison of Different Microprocessor Architecture

X64 vs. X86

A microprocessor may be listed for sale as an "x86 processor" or an "x64 processor." What is the difference, though?

An "x86" processor is a microprocessor that is capable of processing information in 32-bit pieces (called "instructions"). Each "bit" is a piece of information that the computer uses to transmit information, run computations and perform other such processes. A processor using x86 architecture is considered a successor technology to the original microprocessors used in the IBM PC. Since the original processor used in an IBM PC was based upon the Intel 8086 microprocessor, successive microprocessors using the same set of instructions to run have been named similarly--the 80286, 80386 and 80486, for example.

A microprocessor using x64 architecture is slightly different from the x86 processor. An x64 processor is capable of processing not only 32-bit instructions, but also 64-bit instructions as well. Because of the increased capability of the x64 microprocessor, a computer that utilises an x64 microprocessor is also capable of utilising more memory (128 GB maximum vs. 4 GB maximum) than a computer with an x86 microprocessor.

An x64 microprocessor, therefore, would be the better choice if you plan to use the computer for memory-intensive applications, or if you need better overall performance out of your computer system.

Dual-Core vs. Quad-Core Architecture

Many microprocessors made by Intel and AMD are multi-core processors. What this means is that within one microprocessor, there are two or more central processing units (cores) within one integrated circuit package.

As their names imply, "dual-core" multiprocessors have two CPU cores in the multiprocessor package, and "quad-core" multiprocessors have four CPU cores. Depending upon the software being used, the operating system the computer system has installed and the amount of memory available to the computer system, a quad-core system is capable of running more simultaneous processes than a dual-core system.

However, some older software or operating systems (such as Windows 95, Windows 98 or Windows Me) are not capable of utilising multi-core microprocessors. If the software or operating system is incapable of utilising the resources available, using a multi-core microprocessor is no different than using a single-core microprocessor.

4.0 CONCLUSION

A microprocessor is an integrated circuit that holds most of the works of a computer; it can fetch an instruction.

When there is program memory, RAM, and other support circuitry on the chip, this is a microcontroller. Depending on how much data can be processed at once, this can be a 4, 8, 16, 32 (and more) bit processor.

The vast majority of microprocessors are embedded microcontrollers. The second most common type of processors are desktop processors, such as Intel's Pentium or AMD's Athlon. Less common are the extremely powerful processors used in high-end servers, such as Sun's SPARC, IBM's Power, or Intel's Itanium.

Historically, microprocessors and microcontrollers have come in "standard sizes" of 8 bits, 16 bits, 32 bits, and 64 bits. These sizes are common, but that does not mean that other sizes are not available. Some microcontrollers (usually specially designed embedded chips) can come in other "non-standard" sizes such as 4 bits, 12 bits, 18 bits, or 24 bits. The number of bits represents how much physical memory can be directly addressed by the CPU. It also represents the amount of bits that can be read by one read/write operation. In some circumstances, these are different; for instance, many 8 bit microprocessors have an 8 bit data bus and a 16 bit address bus.

- 8 bit processors can read/write 1 byte at a time and can directly address 256 bytes
- 16 bit processors can read/write 2 bytes at a time, and can address 65,536 bytes (64 Kilobytes)
- 32 bit processors can read/write 4 bytes at a time, and can address 4,294,967,295 bytes (4 Gigabytes)
- 64 bit processors can read/write 8 bytes at a time, and can address 18,446,744,073,709,551,616 bytes (16 Exabytes)

This device was followed by the **8008**, which had an 8-bit data bus. Two more 8-bit microprocessors (reference to the number of bits usually refers to the data bus unless stated otherwise), the **8080** and **8085** were introduced in the mid-1970s. These two devices could address only 216 memory locations.

Two classification criteria:

- Hardware characteristics

a. RISC
b. CISC
c. VLIW
d. Superscalar

- Characteristics of application areas

a. GPP (General Purpose Processor) / MCU (MicroController Unit)
b. SPP (Special Purpose Processor)

- ASIC (Application-Specific Integrated Circuit)
- ASIP (Application-Specific Instruction-set Processor)
- DSP (Digital Signal Processor)

5.0 SUMMARY

In this unit, you learnt various types of microprocessor based on the hardware characteristic and applications areas. We also examined types of microprocessor using the Intel and Motorola microprocessors. The comparison of microprocessor was also discussed.

6.0 TUTOR-MARKED ASSIGNMENT

1. Explain types of microprocessor based on hardware characteristic.
2. Explain briefly any two types of microprocessor based on the following.

a. Intel
b. Motorola

3. Compare and contrast the following:

a. Dual-Core vs. Quad-Core Architecture
b. X64 vs. X86 processor

7.0 REFERENCES/FURTHER READING

Heffernan G. (2001). “8051 Tutorial”

Kastner, D. (2003). “Embedded System.”

Krishna, K. M. (2004). “Microprocessors and Microcontrollers/Architecture of Microprocessors” Lecture Notes.

Mack, P. E. (2005). “The Microcomputer Revolution” http// www. Clemson.edu/caah/history/Facultypages/Pammack/lec122/micro. htm

UNIT 4 MICROPROCESSORS OPERATION AND FUNCTIONS

CONTENTS

1.0 INTRODUCTION

A microprocessor manipulates data in a computer system. The central processing unit acts as the brain of a computer and consists of one or more microprocessors made up of several thousand transistors on a single integrated circuit. The microprocessor works in conjunction with other parts of the computer to compute arithmetic and logic functions to handle tasks using an instruction set to perform all tasks within a computer.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- list different operations of a microprocessor
- explain functions of a microprocessor.

3.0 MAIN CONTENT

3.1 Different Operations of a Microprocessor

The microprocessor works in conjunction with other parts of the computer to compute arithmetic and logic functions to handle tasks using an instruction set to perform all tasks within a computer.

1. Input and Output

The microprocessor accepts input from devices, such as a mouse, keyboard or scanner, and performs a function on that data. It makes a decision based on the data, the microprocessor computes the information, and then it sends the results to the output devices, such as a monitor or printer, as readable information for the user. For example, if a user using a word processor presses "m" on the keyboard, the microprocessor will accept that and send the letter "m" to the monitor.

2. Arithmetic Logic Unit

The arithmetic logic unit gathers information as input from the CPU registers and operands and then does the arithmetic operations (addition, subtraction, multiplication and division) and logic operations (AND, OR and XOR). During data processing, the ALU tests conditions and prepares to take different actions based on results. The ALU also gathers data from additional sources, including number systems, instructions, timing and data routing circuits, such as adders and subtracters.

3. Memory

The microprocessor accesses and stores binary instructions into memory, or circuits that store bits. Random access memory is a control memory that uses registers to temporarily store data. The microprocessor stores volatile data used by programs in RAM. Read-only memory stores data permanently on chips with instructions built in. It takes longer to access the information in ROM, but it does not lose information when a computer shuts down as does RAM.

4. Control Unit

The control unit directs the flow of operations and data by selecting one program statement at a time, interpreting it, and sending messages to the ALU or registers to carry out the instruction. It also decides where to keep information in memory and which devices to communicate with by interfacing with the ALU, memory and input/output devices. The control unit can also shut down a computer if it or another device, such as the power source, detects abnormal conditions.

5. Information Exchange

The system bus connects the microprocessor to the peripherals, such as a keyboard, mouse, printer, scanner, speaker or digital camera. The microprocessor sends and receives data through the system bus to communicate with the peripherals. It only communicates with one peripheral at a time to not mix up any information and send it to the wrong place. The control unit controls the timing of the information exchange.

3.2 Function of Microprocessors

A microprocessor controls all functions of the CPU, or central processing unit, of a computer or other digital device. The microprocessor functions as an artificial brain. The entire function of the CPU is controlled by a single integrated circuit. The microprocessor is programmed to give and receive instructions from other components of the device. The system can control everything from small devices such as calculators and mobile phones, to large automobiles. Microprocessors are small chips that carry out all the roles of CPU. A device allows a computer to work. It performs in same distinct way whether incorporated on laptops or servers. The first ever microprocessor was introduced by Intel in the year 1971. The processor was called Intel4004 and carried out most simple operations related to mathematics.

It is the size of a chip, which contains billions of various transistors. They have the power to calculate mathematical operations using algorithms. Due to floating point processors, the microprocessors can conduct any operation or computation accurately at the earliest.

The microprocessor does enable to transfer data from one location to another. The information that you require is shifted to the hard drive in split seconds. The microprocessors are considered devices that make instant decisions and carry out multiple commands with the help of the decisions. The register and coder do help the microprocessor to carry out the required duties and instructions.

The two memories are responsible for any microprocessor to function properly. Firstly, the read only memory and secondly, random access memory, which together constitute the microprocessor. ROM as a program includes a finite set of instructions that is combined with a constant set of bytes. RAM includes a pre-defined set of bytes that can store a limited amount of information. The other function of microprocessor is to conduct and carry out executions in all kinds of formats be it data, video or audio. However, we have already entered the digital age whereby microprocessors sometimes do not exist. The recent developments in the field of technology, medicine, communications in the twenty first century has lead to various innovations in microprocessors. Microprocessors have improved our lifestyle due to new enabled, lighter and improved hand held machinery.

4.0 CONCLUSION

The microprocessor works in conjunction with other parts of the computer to compute arithmetic and logic functions to handle tasks using an instruction set to perform all tasks within a computer. The microprocessor functions as an artificial brain. The entire function of the CPU is controlled by a single integrated circuit. The microprocessor is programmed to give and receive instructions from other components of the device.

5.0 SUMMARY

In this unit, we have discussed how the microprocessor works in conjunction with other parts of the computer. We also discussed different operations of a microprocessor such as input and output, ALU, memory, control unit, and so on.

6.0 TUTOR-MARKED ASSIGNMENT

1. What are the main operations of microprocessor?
2. What are the main functions of microprocessor?

7.0 REFERENCES/FURTHER READING

Kastner, D. (2003). “Embedded System.”

Krishna Kumar, M. (2004). “Microprocessors and Microcontrollers/Architecture of microprocessors.” Lecture Notes.

Microprocessor Architecture “http://www.ehow.com/way_5581467_microprocessor-architecture-“tutorials.html.

MODULE 2

UNIT 1 ASSEMBLY LANGUAGE

CONTENTS

1.0 INTRODUCTION

An assembly language is a low-level language that contains a short word or "mnemonic" for each individual command that a microcontroller can follow. Assembly language is converted (by a program called an "assembler") into the binary machine code. The machine code is specific to each different type of machine. There are three reasons for using assembly language: speed, speed, and more speed. Even those who absolutely hate assembly language will admit that if speed is your primary concern, assembly language is the way to go. The 8085 Assembly language instruction set is used to explain the format and operation of assembly language. Assembly language has several benefits, which are stated in the main text.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain what assembly language is
- identify benefits of assembly language
- explain assembly language formats
- discuss 8085 assembly language instruction set.

3.0 MAIN CONTENT

3.1 What is Assembly Language?

An assembly language is a small language that contains a short word or "mnemonic" for each individual command that a microcontroller can follow. Each command gets a single mnemonic, and each mnemonic corresponds to a single machine command. Assembly language is converted (by a program called an "assembler") into the binary machine code. The machine code is specific to each different type of machine.

When using assembly language, you will store most of your variables in registers. There are only 14 general purpose registers; nevertheless, you will need to use memory words for other variables. A variable is just a location in memory that is reserved for use as that variable. To use that variable, you will need to first load its value into a register. If the value changes, you will need to write its value back into the memory location if you want to keep it. The variables are declared in assembly language by using a .DATA directive, and using a label to specify a certain word of memory.

Assembly language has several benefits, which are:

- **Speed:** Assembly language programs are generally the fastest programs around.
- **Space:** Assembly language programs are often the smallest.
- **Capability:** You can do things in assembly, which are difficult or impossible in high-level languages.
- **Knowledge:** Your knowledge of assembly language will help you write better programs

Number Representation for Different Bases

The following is an example showing the decimal number 46 represented in different number bases:

```
46d          ; 46 decimal
2Eh          ; 2Eh is 46 decimal represented as a hex number
56o          ; 56o is 46 decimal represented as an octal number
101110b      ; 101110b is 46 decimal represented as a binary
number.
```

Note a number digit must be used in the first character of a hexadecimal number. For example, the hexadecimal number A5h is illegally represented and should be represented as 0A5h.

The Basic Assembly Language Operators

1. The Arithmetic Operators

The arithmetic operators are:

+ add
- subtract
* multiply
/ divide

MOD modulo (result is the remainder following division)

2. The Logical Operators

The logical operators are:

AND Logical AND
OR Logical OR
XOR Logical XOR (exclusive OR)
NOT Logical NOT

3. The Relational Operators

The result of a relational operation is either true (represented by minus 1), or false (represented by zero). The relational operators are:

Equal to EQ =
not equal to NE <>
greater than GT >
greater than or equal to GE >=
less than LT <
less than or equal to LE <=
(note 'EQ' symbol and '= ' symbol have the same meaning)

4. Operator Precedence

Like a high-level language, assembly level programs define operator precedence. Operators with same precedence are evaluated left to right. Note, brackets ( ) means to
evaluate this first. HIGH indicates the high-byte and LOW indicates the low-byte.

Later examples will clarify the use of such special operators. The precedence list, highest first, is as follows:

( )
HIGH LOW
* / MOD SHL SHR
+ -
= <> < <= > >=
NOT
AND
OR XOR

3.2 Assembly Language Format

The format of an assembly language instruction is as follows.

```
INST      arg1,      arg2,    arg3 ; comment
```

The first thing on the line is the name of the instruction. This is generally a three or 4-letter abbreviation of what the instruction does. This is known as a *mnemonic*. In assembly language, whitespace does not matter, but there can only be one instruction on a line, and you cannot break up an instruction across multiple lines. The Smalltium assembler is not case sensitive (except for character constants in the data section).

The actual number and type of the arguments is variable and depends on which instruction you are using. In general, the arguments can be either registers or values. A register is specified by ``r#" where # is the register number (for example r0 or r12). Remember that r0 always contains the value 0, and r15 is the program counter. Because r15 is the program counter, you can also refer to it as ``pc". A value is specified by a decimal number or by a label. We will discuss labels a bit later.

3.3 8085 Assembly Language Instruction Set

Assembler Directives

The assembler directives are special instruction to the assembler program to define some specific operations but these directives are not part of the executable program.

Some of the most frequently assembler directives are listed as follows.

ORG: OriGinate, defines the starting address for the program in program (code) memory

EQU: EQUate, assigns a numeric value to a symbol identifier so as to make the program more readable.

DB: Define a Byte, puts a byte (8-bit number) number constant at this memory location

DW: Define a Word, puts a word (16-bit number) number constant at this memory location

DBIT: Define a Bit, defines a bit constant, which is stored in the bit addressable section if the Internal RAM.

END: This is the last statement in the source file to advise the assembler to stop the assembly process.

Types of Instructions

The assembly level instructions include: data transfer instructions, arithmetic instructions, logical instructions, program control instructions, and some special instructions such as the rotate instructions.

Data Transfer

Many computer operations are concerned with moving data from one location to another. The 8051 uses five different types of instruction to move data.

MOV MOVX MOVC
PUSH and POP XCH

MOV

In the 8051 the MOV instruction is concerned with moving data internally, that is between Internal RAM, SFR registers, general registers etc. MOVX and MOVC are
used in accessing external memory data. The MOV instruction has the following

format:

MOV destination <- source
The instruction copies (copy is a more accurate word than move) data from a defined source location to a destination location. Example MOV instructions are:

MOV R2, #80h ; Move immediate data value 80h to register R2
MOV R4, A ; Copy data from accumulator to register R4
MOV DPTR, #0F22Ch ; Move immediate value F22Ch to the DPTR register
MOV R2, 80h ; Copy data from 80h (Port 0 SFR) to R2
MOV 52h, #52h ; Copy immediate data value 52h to RAM location 52h
MOV 52h, 53h ; Copy data from RAM location 53h to RAM 52h
MOV A, @R0 ; Copy contents of location addressed in R0 to A (indirect addressing)

MOVX

The 8051 the external memory can be addressed using indirect addressing only. The DPTR register is used to hold the address of the external data (since DPTR is a 16-bit register it can address 64KByte locations: 216 = 64K). The 8 bit registers R0 or R1 can also be used for indirect addressing of external memory but the address range is limited to the lower 256 bytes of memory (28 = 256 bytes).

PUSH and POP

PUSH and POP instructions are used with the stack only. The SFR register SP contains the current stack address. Direct addressing is used as shown in the following examples:

PUSH 4Ch ; Contents of RAM location 4Ch is saved to the stack. SP is incremented.

PUSH 00h ; The content of R0 (which is at 00h in RAM) is saved to the stack and SP is incremented.

POP 80h ; The data from current SP address is copied to 80h and SP is decremented.

Arithmetic Instruction

Some key flags within the PSW, i.e. C, AC, OV, P, are utilised in many of the arithmetic instructions. The arithmetic instructions can be grouped as follows.

Addition

Register A (the accumulator) is used to hold the result of any addition operation.

Some simple addition examples are:

ADD A, #25h ; Adds the number 25h to A, putting sum in A
ADD A, R3 ; Adds the register R3 value to A, putting sum in A

Simple addition is done within the 8051 based on 8-bit numbers, but it is often required to add 16-bit numbers, or 24-bit numbers, and so on. This leads to the use of multiple byte (multi-precision) arithmetic. The least significant bytes are first added, and if a carry results, this carry is carried over in the addition of the next significant byte etc. This addition process is done at 8-bit precision steps to achieve multiprecision arithmetic. The ADDC instruction is used to include the carry bit in the addition process. Example instructions using ADDC are:

ADDC A, #55h ; Add contents of A, the number 55h, the carry bit; and put the sum in A

ADDC A, R4 ; Add the contents of A, the register R4, the carry bit; and put the sum in A.

Subtraction

Computer subtraction can be achieved using 2's complement arithmetic. Most computers also provide instructions to directly subtract signed or unsigned numbers.

The accumulator, register A, will contain the result (difference) of the subtraction operation. The C (carry) flag is treated as a borrow flag, which is always subtracted from the minuend during a subtraction operation. Some examples of subtraction instructions are:

SUBB A, #55d ; Subtract the number 55 (decimal) and the C flag from A; and put the result in A.

SUBB A, R6 ; Subtract R6 the C flag from A; and put the result in A.
SUBB A, 58h ; Subtract the number in RAM location 58h and the C flag From A; and put the result in A.

Increment/Decrement

The increment (INC) instruction has the effect of simply adding a binary one to a number while a decrement (DEC) instruction has the effect of subtracting a binary one from a number. The increment and decrement instructions can use the addressing modes: direct, indirect and register. The flags C, AC, and OV are not affected by the increment or decrement instructions. If a value of FFh is increment it overflows to 00h. If a value of 00h is decrement it underflows to FFh. The DPTR can overflow from FFFFh to 0000h. The DPTR register cannot be decremented using a DEC instruction (unfortunately!). Some example INC and DEC instructions are as follows:

```
INC R7      ; Increment register R7
INC A       ; Increment A
INC @R1     ; Increment the number which is the content of the
            address in R1
DEC A       ; Decrement register A
DEC 43h     ; Decrement the number in RAM address 43h
INC DPTR    ; Increment the DPTR register
```

Multiply/Divide

The 8051 supports 8-bit multiplication and division. This is low precision (8-bit) arithmetic but is useful for many simple control applications. The arithmetic is relatively fast since multiplication and division are implemented as single instructions. If better precision, or indeed, if floating point arithmetic is required then special software routines need to be written. For the MUL or DIV instructions, the A and B registers must be used and only unsigned numbers are supported.

Multiplication

The MUL instruction is used as follows (note absence of a comma between the A and B operands):

```
MUL AB ; Multiply A by B.
```

The resulting product resides in registers A and B, the low-order byte is in A and the high order byte is in B.

Division

The DIV instruction is used as follows:

DIV AB ; A is divided by B.

The remainder is put in register B and the integer part of the quotient is put in register A.

Decimal Adjust (Special)

The 8051 performs all arithmetic in binary numbers (i.e. it does not support BCD arithmetic). If two BCD numbers are added then the result can be adjusted by using the DA, decimal adjust, instruction:

DA A ; Decimal adjust A following the addition of two BCD numbers.

Logical

Boolean Operations

Most control applications implement control logic using Boolean operators to act on the data. Most microcomputers provide a set of Boolean instructions that act on byte level data. However, the 8051 (somewhat uniquely) additionally provides Boolean instruction, which can operate on bit level data.

The following Boolean operations can operate on byte level or bit level data:

ANL Logical AND
ORL Logical OR
CPL Complement (logical NOT)
XRL Logical XOR (exclusive OR)
Logical operations at the BYTE level

The destination address of the operation can be the accumulator (register A), a general register, or a direct address. Status flags are not affected by these logical operations (unless PSW is directly manipulated). Example instructions are:

ANL A, #55h ; AND each bit in A with corresponding bit in number 55h, leaving the result in A.

ANL 42h, R4 ; AND each bit in RAM location 42h with corresponding bit in R4, leaving the result in RAM location 42h.

ORL A,@R1 ; OR each bit in A with corresponding bit in the number whose address is contained in R1 leaving the result in A.

XRL R4, 80h ; XOR each bit in R4 with corresponding bit in RAM location 80h (port 0), leaving result in A.

CPL R0 ; Complement each bit in R0
Logical operations at the BIT level

The C (carry) flag is the destination of most bit level logical operations. The carry flag can easily be tested using a branch (jump) instruction to quickly establish program flow control decisions following a bit level logical operation.

The following SFR registers only are addressable in bit level operations:

PSW IE IP TCON SCON

Examples of bit level logical operations are as follows.

SETB 2Fh ; Bit 7 of Internal RAM location 25h is set
CLR C ; Clear the carry flag (flag =0)
CPL 20h ; Complement bit 0 of Internal RAM location 24h
MOV C, 87h ; Move to carry flag the bit 7of Port 0 (SFR at 80h)
ANL C,90h ; AND C with the bit 0 of Port 1 (SFR at 90)
ORL C, 91h ; OR C with the bit 1 of Port 1 (SFR at 90)

Rotate Instructions

The ability to rotate the A register (accumulator) data is useful to allow examination of individual bits. The options for such rotation are as follows.

RL A ; Rotate A one bit to the left. Bit 7 rotates to the bit 0 position
RLC A ; The Carry flag is used as a ninth bit in the rotation loop

Accumulator

b7 b6 b5 b4 b3 b2 b1 b0
Carry flag
C

Accumulator

b7 b6 b5 b4 b3 b2 b1 b0
RR A ; Rotates A to the right (clockwise)
RRC A ; Rotates to the right and includes the carry bit as the 9th bit.

Swap instruction

The Swap instruction swaps the accumulator's high order nibble with the low-order nibble using the instruction:
SWAP A

Program Control Instructions

The 8051 supports three kinds of jump instructions:
LJMP, SJMP, AJMP

LJMP

LJMP (long jump) causes the program to branch to a destination address defined by the 16-bit operand in the jump instruction. Because a 16-bit address is used the instruction can cause a jump to any location within the 64KByte program space (216 = 64K). Some example instructions are:

```
LJMP LABEL_X    ; Jump to the specified label
LJMP 0F200h     ; Jump to address 0F200h
LJMP @A+DPTR    ; Jump to address which is the sum of DPTR and Reg. A
```

SJMP

SJMP (short jump) uses a single byte address. This address is a signed 8-bit number and allows the program to branch to a distance –128 bytes back from the current PC

Accumulator

b7 b6 b5 b4 b3 b2 b1 b0
Carry flag
C
Accumulator

b7 b6 b5 b4 b3 b2 b1 b0

Accumulator

b7 b6 b5 b4 b3 b2 b1 b0
High nibble low nibble address or +127 bytes forward from the current PC address. The address mode used with this form of jumping (or branching) is referred to as relative addressing, introduced earlier as the jump is calculated relative to the current PC address.

AJMP

This is a special 8051 jump instruction, which allows a jump with a 2KByte address boundary (a 2K page).

There is also a generic JMP instruction supported by many 8051 assemblers. The assembler will decide which type of jump instruction to use, LJMP, SJMP or AJMP, so as to choose the most efficient instruction.

Subroutines and Program Flow Control

A subroutine is called using the LCALL or the ACALL instruction.

LCALL

This instruction is used to call a subroutine at a specified address. The address is 16 bits long so the call can be made to any location within the 64KByte memory space.

When a LCALL instruction is executed, the current PC content is automatically pushed onto the stack of the PC. When the program returns from the subroutine the PC contents is returned from the stack so that the program can resume operation from the point where the LCALL was made The return from subroutine is achieved using the RET instruction, which simply pops the PC back from the stack.

ACALL

The ACALL instruction is logically similar to the LCALL but has a limited address range similar to the AJMP instruction.

ACALL is a generic call instruction supported by many 8051 assemblers. The assembler will decide which type of call instruction, LCALL or ACALL, to use to choose the most efficient instruction.

Program Control Using Conditional Jumps

Most 8051 jump instructions use an 8-bit destination address, based on relative addressing, that is addressing within the range –128 to +127 bytes.

When using a conditional jump instruction the programmer can simply specify a program label or a full 16-bit address for the conditional jump instruction's destination. The assembler will position the code and work out the correct 8-bit relative address for the instruction. Some example conditional jump instructions are:

JZ LABEL_1 ; Jump to LABEL_1 if accumulator is equal to zero
JNZ LABEL_X ; Jump to LABEL_X if accumulator is not equal to zero
JNC LABEL_Y ; Jump to LABEL_Y if the carry flag is not set
DJNZ R2, LABEL ; Decrement R2 and jump to LABEL if the resulting value of R2 is not zero.

CJNE R1, #55h , LABEL_2 ; Compare the magnitude of R1 and the number 55h and jump to LABEL_2 if the magnitudes are not equal.
Note, jump instructions such as DJNZ and CJNE are very powerful as they carry out a particular operation (e.g.: decrement, compare) and then make a decision based on the result of this operation. Some example code later will help to explain the context in which such instructions might be used.

Definition of Important Terms

Assembler	A program that translates assembly language into machine language.
assembler directive	A command in assembly language that causes the assembler to exhibit different behavior. Does not result in the generation of a machine language instruction.
Assembly language	A programming language that is very similar to machine language, but uses symbols instead of binary numbers.
condition flags	A register that is treated such that each bit has a different meaning, rather than all of them taken together as a number. These bits are set during a compare, and tested during a jump.
core	Another name for the system memory. A ``core image" is a file that contains an image of the

	contents of memory. The assembler program produces a core image, along with some extra comments to help you understand it.
ifetch loop	The control loop used inside a processor. Consists of a fetch, decode, and execute phase.
immediate value	A value that is immediately available in a machine language instruction word, rather than having to be loaded from a register or from memory.
label	A symbolic representation of a memory address.
machine language	The binary form of instructions that a processor can run directly.
mnemonic	A symbolic representation of an opcode.
opcode	Operation Code. The portion of an instruction word that specifies what kind of instruction it is.
program counter (PC)	The register used to keep track of where in the program the machine is currently executing.
register	A memory word located inside the processor. Used as temporary work space.
word	The unit of memory access. Usually defined by the size of the registers in the CPU. In the Smalltium, a word is 16 bits.

4.0 CONCLUSION

The major strength of assembly language is speed. It is closer to the machine language. You can do things in assembly language, which are difficult or impossible in high-level languages. Your knowledge of assembly language will help you write better programs.

5.0 SUMMARY

In this unit, we have defined assembly language and benefits of assembly language. Formats and sets of instruction of assembly language were also examined.

6.0 TUTOR-MARKED ASSIGNMENT

1. What is assembly language?
2. What are the benefits of assembly language?
3. What are assembly language directives?
4. Explain the following instructions:

i. MOV A, @R0 ii. PUSH 00h iii. ANL A, #55h
iv. ADD A, R3

5. Write an assembly program to find greatest between two numbers.

7.0 REFERENCES/FURTHER READING

Heffernan G. (2001). “8051 Tutorial”.

UNIT 2 INSTRUCTION SET AND ADDRESSING MODES

CONTENTS

1.0 INTRODUCTION

This unit discusses the instruction set format of 8085 microprocessor. The instruction set of arithmetic operation, data transfer, and machine control operation. The various addressing mode is also discussed.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain what are instructions set
- explain data transfer operation
- explain arithmetic operation
- discuss instruction set format
- explain addressing mode and the various types of addressing modes.

3.0 MAIN CONTENT

3.1 Instruction Set Classification

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**, determines what functions the microprocessor can perform. These instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

Data Transfer (Copy) Operations

This group of instructions copy data from a location called a source to another location called a destination, without modifying the contents of the source. In technical manuals, the term *data transfer* is used for this copying function. However, the term *transfer* is misleading; it creates the impression that the contents of the source are destroyed when, in fact, the contents are retained without any modification.

The various types of data transfer (copy) are listed below together with examples of each type:

1. Between Registers: Copy the contents of the register B into register D.
2. Specific data byte to a register or a memory location: Load register B with the data byte 32H.
3. Between a memory location and a register: From a memory location 2000H to register B.
4. Between an I/O device and the accumulator: From an input keyboard to the accumulator.

Arithmetic Operations

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

Addition - Any 8-bit number, or the contents of a register or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. No two other 8-bit registers can be added directly (e.g., the contents of register B cannot be added directly to the contents of the register C). The instruction DAD is an exception; it adds 16-bit data directly in register pairs.

Subtraction - Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. The subtraction is performed in 2's compliment, and the results if negative, are expressed in 2's complement. No two other registers can be subtracted directly.

Increment/Decrement - The 8-bit contents of a register or a memory location can be incremented or decrement by one. Similarly, the 16-bit contents of a register pair (such as BC) can be incremented or decrement by one. These increment and decrement operations differ from addition and subtraction in an important way; that is, they can be performed in any one of the registers or in a memory location.

Logical Operations

These instructions perform various logical operations with the contents of the accumulator.

AND, OR Exclusive-OR - Any 8-bit number, or the contents of a register, or of a memory location can be logically ANDed, Ored, or Exclusive-ORed with the contents of the accumulator. The results are stored in the accumulator.

Rotate- Each bit in the accumulator can be shifted either left or right to the next position.

Compare- Any 8-bit number or the contents of a register, or a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.

Complement - The contents of the accumulator can be complemented. All 0s are replaced by 1s and all 1s are replaced by 0s.

Branching Operations

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

Jump - Conditional jumps are an important aspect of the decision-making process in the programming. These instructions test for a certain conditions (for instance, Zero or Carry flag) and alter the program sequence when the condition is met. In addition, the instruction set includes an instruction called *unconditional jump.*

Call, Return, and Restart - These instructions change the sequence of a program either by calling a subroutine or by returning from a subroutine. The conditional Call and Return instructions also can test condition flags.

Machine Control Operations

These instructions control machine functions such as halt, interrupt, or do nothing. The microprocessor operations related to data manipulation can be summarised in four functions:

1. Copying data
2. Performing arithmetic operations
3. Performing logical operations
4. Testing for a given condition and alerting the program sequence

Some important aspects of the instruction set are noted below.

1. In data transfer, the contents of the source are not destroyed; only the contents of the destination are changed. The data copy instructions do not affect the flags.
2. Arithmetic and logical operations are performed with the contents of the accumulator, and the results are stored in the accumulator (with some expectations). The flags are affected according to the results.
3. Any register including the memory can be used for increment and decrement.
4. A program sequence can be changed either conditionally or by testing for a given data condition.

Instruction Format

An **instruction** is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the **operation code** (opcode), and the second is the data to be operated on, called the **operand.** The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

Instruction word size

The 8085 instruction set is classified into the following three groups according to word size:

1. One-word or 1-byte instructions
2. Two-word or 2-byte instructions
3. Three-word or 3-byte instructions

In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor. However, instructions are commonly referred to in terms of bytes rather than words.

One-Byte Instructions

A one-byte instruction includes the opcode and operand in the same byte. Operand(s) are internal register and are coded into the instruction.

For example:

Task	Opcode	operand	Binary code	Hex Code
Copy the contents of the accumulator in the register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator	ADD	B	1000 0000	80H
Invert (compliment) each bit in the accumulator	CMA		0010 1111	2FH

These instructions are one-byte instructions performing three different tasks. In the first instruction, both operand registers are specified. In the second instruction, the operand B is specified and the accumulator is assumed. Similarly, in the third instruction, the accumulator is assumed the implicit operand. These instructions are stored in 8- bit binary format in memory; each requires one memory location.

MOV rd, rs
rd <-- rs copies contents of rs into rd.
Coded as 01 ddd sss where ddd is a code for one of the 7 general registers which is the destination of the data, sss is the code of the source register.

Example: MOV A,B
Coded as 01111000 = 78H = 170 octal (octal was used extensively in instruction design of such processors).
ADD r
A <-- A + r

Two-Byte Instructions

In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode. For example:

Task	Opcode	Operand	Binary code	Hex code	
Load an 8-bit data byte in the accumulator	MVI	A,Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

Assume that the data byte is 32H. The assembly language instruction is written as

Mnemonics	**Hex code**
MVI A, 32H	**3E 32H**

The instruction would require two memory locations to store in memory.

MVI r, data
r <-- data

Example: MVI A,30H coded as 3EH 30H as two contiguous bytes. This is an example of immediate addressing.

ADI data
A <-- A + data
OUT port
0011 1110
DATA where port is an 8-bit device address. (Port) <-- A. Since the byte is not the data but points directly to where it is located this is called direct addressing.

Three-Byte Instructions

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address.

opcode + data byte + data byte

For example:

Task	**Opcode**	**Operand**	**Binary Code**	**Hex Code**	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	110 0011	C3	First byte
			1000 0101	85	Second Byte
			0010 0000	20	Third Byte

This instruction would require three memory locations to store in memory.

Three byte instructions - opcode + data byte + data byte
LXI rp, data16

rp is one of the pairs of registers BC, DE, HL used as 16-bit registers. The two data bytes are 16-bit data in L H order of significance.
rp <-- data16

Example:

LXI H,0520H coded as 21H 20H 50H in three bytes. This is also immediate addressing.

LDA addr

A <-- (addr) Addr is a 16-bit address in L H order. Example: LDA 2134H coded as 3AH 34H 21H. This is also an example of direct addressing.

3.2 The 8085 Addressing Modes

The instructions MOV B, A or MVI A, 82H are to copy data from a source into a destination. In these instructions, the source can be a register, an input port, or an 8-bit number (00H to FFH). Similarly, a destination can be a register or an output port. The sources and destination are operands. The various formats for specifying operands are called the **addressing modes**. For 8085, they include the following.

1. Immediate addressing
2. Register addressing
3. Direct addressing
4. Indirect addressing

Immediate addressing

Data is present in the instruction. Load the immediate data to the destination provided.

Example: MVI R, data
Register addressing
Data is provided through the registers.
Example: MOV Rd, Rs

Direct addressing

Used to accept data from outside devices to store in the accumulator or send the data stored in the accumulator to the outside device. Accept the data from the port 00H and store them into the accumulator or Send the data from the accumulator to the port 01H.

Example: IN 00H or OUT 01H.

Indirect Addressing

This means that the effective address is calculated by the processor. In addition, the contents of the address (and the one following) are used to form a second address. The second address is where the data is stored. Note that this requires several memory accesses; two accesses to retrieve the 16-bit address and a further access (or accesses) to retrieve the data, which is to be loaded into the register.

4.0 CONCLUSION

Instruction set is the set of machine language instructions that a processor can understand.

The microprocessor operations related to data manipulation can be summarised in four functions:

1. Copying data
2. Performing arithmetic operations
3. Performing logical operations
4. Testing for a given condition and alerting the program sequence

The various formats for specifying operands are called the ADDRESSING MODES. For 8085, they are:

1. Immediate addressing
2. Register addressing
3. Direct addressing
4. Indirect addressing

5.0 SUMMARY

This unit has been able to discuss the classification of instructions sets and their operation, which relate to the processor to perform data manipulation. The addressing mode and the types of addressing mode were also discussed.

6.0 TUTOR-MARKED ASSIGNMENT

1. What is an instruction set?
2. What is addressing modes?
3. Explain briefly four types of addressing modes.

7.0 REFERENCES/FURTHER READING

"Introduction to 8085 Architecture and Programming"
http://en.wikibooks.org/wiki/ "Introduction to 8085 Architecture and Programming"

UNIT 3 INTERRUPTS

CONTENTS

1.0 INTRODUCTION

As the name implies, an **interrupt** is some event, which interrupts normal program execution. As stated earlier, program flow is always sequential, being altered only by those instructions, which expressly cause program flow to deviate in some way. However, interrupts give us a mechanism to "put on hold" the normal program flow, execute a subroutine, and then resume normal program flow as if we had never left it. This subroutine, called an interrupt handler, is only executed when a certain event (interrupt) occurs. The event may be one of the timers "overflowing," receiving a character via the serial port, transmitting a character via the serial port, or one of two "external events."

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain what a interrupts is
- discuss the interrupts priorities
- explain what happen when an interrupt occur and end
- identify the common problems of interrupts.

3.0 MAIN CONTENT

3.1 Description of Interrupts

An interrupt can be defined as some event, which interrupts normal program execution. Program flow is always sequential, being altered only by those instructions, which expressly cause program flow to deviate in some way. However, interrupts give us a mechanism to "put on hold" the normal program flow, execute a subroutine, and then resume normal program flow as if we had never left it. This subroutine, called an **interrupt handler**, is only executed when a certain event (interrupt) occurs. The event may be one of the timers "overflowing," receiving a character via the serial port, transmitting a character via the serial port, or one of two "external events." The interrupt automatically transfers control to an interrupt processing routine, which takes some action in response to the condition that caused the interrupt, and then control, is returned to the program at which its execution was interrupted.

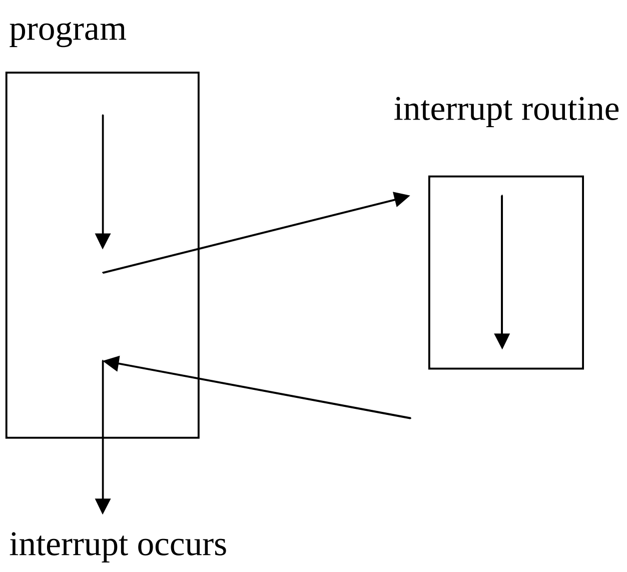


Figure 3: Interrupt

The ability to interrupt normal program execution when certain events occur makes it much easier and much more efficient to handle certain conditions. If it were not for interrupts, we would have to manually check in our main program whether the timers had overflow, whether we had received another character via the serial port, or if some external event had occurred. Besides making the main program ugly and hard to read, such a situation would make our program inefficient since wed be burning precious "instruction cycles" checking for events that usually do not happen.

3.2 Interrupt Priorities

The 8051 offers two levels of interrupt priority: high and low. By using interrupt priorities, you may assign higher priority to certain interrupt conditions. For instance, you may have enabled Timer 1 Interrupt, which is automatically called every time Timer 1 overflows. Additionally, you may have enabled the Serial Interrupt, which is called every time a character is received via the serial port. However, you may consider that receiving a character is much more important than the timer interrupt. In this case, if Timer 1 Interrupt is already executing you may wish that the serial interrupt itself interrupts the Timer 1 Interrupt. When the serial interrupt is complete, control passes back to Timer 1 Interrupt and finally back to the main program. You may accomplish this by assigning a high priority to the Serial Interrupt and a low priority to the Timer 1 Interrupt.

Interrupt priorities are controlled by the **IP** SFR (B8h). The IP SFR has the following format.

Bit	Name	Bit Address	Explanation of Function
7	-	-	Undefined
	-	-	Undefined
5	-	-	Undefined
4	PS	BCh	Serial Interrupt Priority
3	PT1	BBh	Timer 1 Interrupt Priority
2	PX1	BAh	External 1 Interrupt Priority
1	PT0	B9h	Timer 0 Interrupt Priority
0	PX0	B8h	External 0 Interrupt Priority

When considering interrupt priorities, the following rules apply:

- Nothing can interrupt a high-priority interrupt--not even another high priority interrupt.
- A high-priority interrupt may interrupt a low-priority interrupt.
- A low-priority interrupt may only occur if no other interrupt is already executing.
- If two interrupts occur at the same time, the interrupt with higher priority will execute first. If both interrupts are of the same priority, the interrupt, which is serviced first by polling sequence, will be executed first.

3.3 What Happens When an Interrupt Occurs and Ends?

When an interrupt is triggered, the following actions are taken automatically by the microcontroller.

- The current Program Counter is saved on the stack, low-byte first.
- Interrupts of the same and lower priority are blocked.
- In the case of Timer and External interrupts, the corresponding interrupt flag is cleared.
- Program execution transfers to the corresponding interrupt handler vector address.
- The Interrupt Handler Routine executes.

Take special note of the third step: If the interrupt being handled is a Timer or External interrupt, the microcontroller automatically clears the interrupt flag before passing control to your interrupt handler routine. This means it is not necessary that you clear the bit in your code.

An interrupt ends when your program executes the RETI (Return from Interrupt) instruction. When the RETI instruction is executed, the following actions are taken by the microcontroller.

- Two bytes are popped off the stack into the Program Counter to restore normal program execution.
- Interrupt status is restored to its pre-interrupt status.

3.4 Serial Interrupts

Serial Interrupts are slightly different from the rest of the interrupts. This is because there are two interrupt flags: RI and TI. If either flag is set, a serial interrupt is triggered. As you will recall from the section on the serial port, the RI bit is set when a byte is received by the serial port and the TI bit is set when a byte has been sent.

This means that when your serial interrupt is executed, it may have been triggered because the RI flag was set or because the TI flag was set--or because both flags were set. Thus, your routine must check the status of these flags to determine what action is appropriate. In addition, since the 8051 does not automatically clear the RI and TI flags you must clear these bits in your interrupt handler.

3.5 Common Problems with Interrupts

Interrupts are powerful tool available to the 8051 developer, but when used incorrectly they can be a source of a huge number of debugging hours. Errors in interrupt routines are often very difficult to diagnose and correct.

If you are using interrupts and your program is crashing or does not seem to be performing as you would expect, always review the following interrupt-related issues.

- **Register Protection**: Make sure you are protecting all your registers, as explained above. If you forget to protect a register that your main program is using, strange results may occur. In our example above we saw how failure to protect registers caused the main program to apparently calculate that 25h + 10h = 51h. If you witness problems with registers changing values unexpectedly or operations producing "incorrect" values, it is very likely that you have forgotten to protect registers. **ALWAYS PROTECT YOUR REGISTERS.**
- **Forgetting to restore protected values**: Another error associated with interrupts is pushing registers onto the stack (to protect them), and then forgetting to pop them off the stack before exiting the interrupt. For example, you may push ACC, B, and PSW onto the stack in order to protect them and subsequently pop only ACC and PSW off the stack before exiting. In this case, since you forgot to restore the value of "B", an extra value remains on the stack. When you execute the RETI instruction, the 8051 will use that value as the return address instead of the correct value. In this case, your program will almost certainly crash. **ALWAYS MAKE SURE YOU POP THE SAME NUMBER OF VALUES OFF THE STACK AS YOU PUSHED ON TO IT.**
- Using RET instead of RETI: Remember that interrupts are always terminated with the RETI instruction. It is easy to inadvertently use the RET instruction instead. However, the RET instruction will not end your interrupt. Usually, using a RET instead of a RETI will cause the illusion of your main program running normally, but your interrupt will only be executed once. If it appears that your interrupt mysteriously stops executing, verify that you are exiting with RETI.

4.0 CONCLUSION

In general, your interrupt routine must protect the following registers:

- PSW
- DPTR (DPH/DPL)
- PSW
- ACC
- B
- Registers R0-R7

Remember that PSW consists of many individual bits that are set by various 8051 instructions. Unless you are absolutely sure of what you are doing and have a complete understanding of what instructions set what bits, it is generally a good idea to *always* protect PSW by pushing and

An interrupt is a signal that causes the computer to alter its normal flow of instruction execution. For instance, divide by zero, time out, I/O operation

Some interrupt types:

-I/O interrupt: Generated by an I/O channel or device.

Program interrupt: Generated by some condition that occurs during program execution (eg. divide by zero, illegal machine instruction, etc.)
Timer interrupt: Generated by an internal timer within the CPU
Supervisor call interrupt (SVC): A SVC instruction generates an interrupt that transfers control to an OS service routine. The CPU switches from user mode to supervisor mode.

When an interrupt occurs, the status of the CPU is saved (registers, program counter etc.) and the interrupt service routine is executed. After the interrupt is processed, they are restored by the interrupt routine and control is transferred to the program. Saving and restoring the processor status is called context switching.

Interrupt priority: During the processing of an interrupt, all interrupts of equal or lower priority are inhibited, higher priority interrupts are allowed. At the end of an interrupt, if there are more than one interrupt pending, one with the highest priority is executed.
Interrupt masking: is used to prevent certain interrupts from occurring while the first one is being processed.

5.0 SUMMARY

In this unit, we have defined the term interrupts. Interrupts priorities, problems associated with using interrupt, what happens when an interrupt occur and end were also discussed.

6.0 TUTOR-MARKED ASSIGNMENT

1. What is an interrupts?
2. What are various interrupts used in 8085.
3. Explain what happen when an interrupt occur and end.

7.0 REFERENCES/FURTHER READING

"Introduction to 8085 Architecture and Programming."
http://en.wikibooks.org/wiki/ "Introduction to 8085 Architecture and Programming"

UNIT 4 INPUT AND OUT (I/O) TRANSFER / SYSTEM BUS

CONTENTS

1.0 INTRODUCTION

In computing, **input/output**, or **I/O**, refers to the communication between an information processing system (such as a computer), and the outside world, possibly a human, or another information processing system. Inputs are the signals or data received by the system, and outputs are the signals or data sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an input or output operation. I/O devices are used by a person (or other system) to communicate with a computer. For instance, a keyboard or a mouse may be an input device for a computer, while monitors and printers are considered output devices for a computer. Devices for communication between computers, such as modems and network cards, typically serve for both input and output. A bus is a collection of wires on which electrical signals pass between components in the system. The 80x86 family has three major busses: the *address* bus, the *data* bus, and the *control* bus.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain input and output device
- list function of some input and out device
- identify the major port for input and output.
- describe system bus
- identify and differentiate between different types of system buses.

3.0 MAIN CONTENT

3.1 Input and Output Device/Transfer

In computing, the term I/O is short for "Input/output," which refers to the transfer of data either to or from a computer application. A keyboard acts as an input device, transferring whatever you type into the computer, while a monitor or printer would be an example of an output device, since they present the transferred data to the user. Blocking I/O plays a key role in the entire process, helping manage the flow of input and output data.

Input Device

The term "input devices" is used commonly to refer to external hardware components, used to feed data and control signals to a computer system. These devices, along with output devices, constitute the process of human-computer interaction. Output devices are used to present the processed data (information) to an end-user or operator. In this manner, an end-user (a human) communicates with a processing machine (computer) and utilizes it for different tasks. Input devices are usually categorised according to their functions and operations, which determine the nature of input these devices are used to give.

1. Keyboards

Keyboards are oldest type of input devices, which are used extensively to give discrete data, commands to a computer system in the form of text. A normal keyboard has 101 to 104 keys--depending upon the region of origin--which contains all the alphabetical characters (A to Z), numerical figures (0 to 9), and some control (enter, shift,) and function keys (F1 to F12) used in wide range of operations. A keyboard acts as an input device in a manner that whenever a key is pressed, a pre-defined value in the form of electrical signals is transferred to processor, indicating exactly what key is pressed. In this way, different commands and instructions are forwarded to the processor, by using a keyboard.

2. Pointing Devices

Pointing devices are input devices, which, unlike keyboards, feed a computer system with spatial or continuous data. By the term “spatial”, it means that pointing devices involve some form of mechanical motion asserted by the end-user, which results in sending control signals to a processor. Further, they are referred as pointing devices because their mechanical movement is depicted graphically in the form of a pointer (or cursor) over a monitor screen, which is an output device. Most commonly used pointing devices include mouse, trackball, touchpad, game joystick, light pen.

3. Video Input Devices

As their name suggests, video input devices are used to transfer video signals to a computer system. These video signals are transferred in the form digital bits, which are first captured from external world as analog optical (light) signals. Some frequently used video input devices with computer systems include webcams, digital video cameras, and barcode readers. Usually, these input devices require specific software (driver) programs to interface their signals to the processor, which in turn processes their input digital signals and show them in the form of a video on a monitor screen.

4. Audio Input Devices

Another important category of input devices encompasses all those external hardware components, which transfer audio analog signals to computer processor, again in the form of digital bits. These bits are processed and forwarded to audio output devices like speakers, headphones. Audio input devices can be further categorised as audio signal capturing or audio signal generating devices. Audio signal capturing devices carry a microphone to capture audio signals from external environment, while audio signal generating devices use internal electronic circuitry to produce audio signals.

5. Hybrid Input Devices

Hybrid input devices provide a combined single interface for multiple types of input signals to be transferred to the computer processor. These devices are usually used in computer video games, and carry different features for gamer accessibility, like volume control, position control, and so on. The domain of hybrid input devices contains sophisticated and simple hardware devices, including gamepads, game controllers, and mechanical gaming paddles.

Output Device

Peripherals are devices that connect to a computer but are not an integrated or necessary part of the computer. Peripherals are used for input and output. When a camera, for example, is attached to a computer to download photos it becomes a peripheral input device because files on the camera are being moved to the computer. A webcam built into a laptop is also a peripheral.

1. Printer

Printers are common computer peripherals. They are used to write out documents and print photographs. Printers are available in a wide variety of sizes, styles and prices. All-in-one printers are able to scan, receive faxes and make photocopies as well as print documents. Other printers can create high quality photographs or print onto CDs and DVDs. Portable printers are small and lightweight devices that can be used with laptops for mobile offices. Inkjet printers produce a good quality image and print on canvas as well as paper. Laser printers are usually faster and produce a crisper image than inkjet printers. Although prices vary, they may be more expensive than inkjet printers. Solid ink printers produce a high quality color image. They may be slower than laser printers.

2. Speakers

Speakers allow users to enjoy listening to music, movies and other audio files stored on the computer. Speakers are computer peripherals that output sound. Most laptop computers have built-in speakers, which are adequate but external speakers produce better quality sound and higher volume for playing movies and music. External speakers are connected through either a USB port or an audio out port marked with a headset symbol. Most desktops use two external speakers, however, serious gamers or those who want better quality sound often add a third speaker for sounds in the low tones. Speakers vary a great deal in quality, price and configuration.

3. External Memory Device

External memory is used to store data for transport to another computer or for backup in case of computer failure. Jump drives, memory sticks or flash drives are small devices that plug into computers' USB ports. Any type of file can be transferred from the computer to the jump drive. They are primarily used for physically transporting files from one computer to another computer. Jump drives are relatively inexpensive and are sometimes made into key chains or jewelry. External hard drives are another type of external memory. They are larger and more expensive but hold more data. They are used for storing photos and other large files that computers' memories may not be able to store. They are also used for backing up important files in case the computers suffer hard drive crashes or other failures. They also offer encryption to keep sensitive data secure.

Input and Output (I/O) types:

- Programmed I/O
- Interrupt driven I/O (Interrupt service routine)
- Direct memory access (DMA)

Port: an addressable register

- input only ports: CPU only reads from it
- output only ports: CPU only writes to it
- ports for both input and output

Every port has a number. A port can be referred to in an assembly language program by its number, such as: IN 1

Status port: checked by the CPU to see if a key has been pressed and data is ready to be sent to CPU

Data port: when a key is pressed, its ASCII code is found in the data port

Polling: CPU continually examines the status port until a key is pressed (Programmed I/O)

Ex: Reading a character from keyboard

KEYIN: IN   0   ; read status port into A
ANA A ; test A (10_H – key is pressed, 00_H – otherwise)
JZ   KEYIN ; jump if 00_H
IN 1; take data into A
RET

Data port: character to be printed is put in this port

Control port: when the character is ready in the data port, strobe signal (that tells the printer that it can take the character in data port) is sent.

A timing loop must be used in the CPU so that the characters are never passed to the printer at a rate faster than it can accept.

- Special characters:

a. *carriage return*: causes the printer head to move to the left side of the paper
b. *line feed*: causes the printer to advance the paper by one line

Ex: Writing a character to printer

```
CHAROUT: OUT   3        ; put char. in accumulator into output port
         CALL  DELAY        ; make sure printer is ready
         MVI   A, 10H       ; set up strobe word (10H – char. in
         data port is available for printing)
         OUT   2        ; put it into control port to start strobe signal
         to printer
         MVI   A, 00H       ; reset strobe (00H – char. in data port
         is not available for printing)
         OUT   2        ; put it into control port to finish strobe
         signal
         RET
```

3.2 The System Bus

The system bus connects the various components of a VNA machine. The 80x86 family has three major busses: the address bus, the data bus, and the control bus. A bus is a collection of wires on which electrical signals pass between components in the system. These busses vary from processor to processor. However, each bus carries comparable information on all processors; for instance, the data bus may have a different implementation on the 80386 than on the 8088, but both carry data between the processor, I/O, and memory.

A typical 80x86 system component uses standard TTL logic levels. This means each wire on a bus uses a standard voltage level to represent zero and one1. We will always specify zero and one rather than the electrical levels because these levels vary on different processors (especially laptops).

The Data Bus

The 80x86 processors use the data bus to shuffle data between the various components in a computer system. The size of this bus varies widely in the 80x86 families. Indeed, this bus defines the “size” of the processor.

On typical 80x86 systems, the data bus contains eight, 16, 32, or 64 lines. The 8088 and 80188 microprocessors have an eight-bit data bus (eight data lines). The 8086, 80186, 80286, and 80386SX processors have a 16 bit data bus. Future versions of the chip (the 80686/80786?) may have a larger bus.

The Address Bus

The data bus on an 80x86 family processor transfers information between a particular memory location or I/O device and the CPU. The only question is, "*Which memory location or I/O device?*" The address bus answers that question. To differentiate memory locations and I/O devices, the system designer assigns a unique memory address to each memory element and I/O device. When the software wants to access some particular memory location or I/O device, it places the corresponding address on the address bus. Circuitry associated with the memory or I/O device recognises this address and instructs the memory or I/O device to read the data from or place data on the data bus. In either case, all other memory locations ignore the request. Only the device whose address matches the value on the address bus responds.

The Control Bus

The control bus is an eclectic collection of signals that control how the processor communicates with the rest of the system. Consider for a moment the data bus. The CPU sends data to memory and receives data from memory on the data bus. This prompts the question, "Is it sending or receiving?" There are two lines on the control bus, *read* and *write,* which specify the direction of data flow. Other signals include system clocks, interrupt lines, status lines, and so on. The address space is the set of all addressable memory locations.

The *read* and *write* control lines control the direction of data on the data bus. When both contain a logic one, the CPU and memory-I/O are not communicating with one another. If the read line is low (logic zero), the CPU is reading data from memory (that is, the system is transferring data from memory to the CPU). If the write line is low, the system transfers data from the CPU to memory.

4.0 CONCLUSION

Blocking I/O plays a key role in the entire process, helping manage the flow of input and output data.

5.0 SUMMARY

Port: it an addressable registers, which has the following:

- *input only ports*: CPU only reads from it
- *output only ports*: CPU only writes to it
- *ports for both input and output*

Data port: character to be printed is put in this port
Control port: when the character is ready in the data port, strobe signal (that tells the printer that it can take the character in data port) is sent.
Special characters:

- *carriage return*: causes the printer head to move to the left side of the paper
- *line feed*: causes the printer to advance the paper by one line

The system bus connects the various components of a VNA machine. The 80x86 family has three major busses: the *address* bus, the *data* bus, and the *control* bus. A bus is a collection of wires on which electrical signals pass between components in the system. These busses vary from processor to processor.

Address bus: carries the address of a unique memory or input/output (I/O) device 2

Data bus: carries data stored in memory (or an I/O device) to the CPU or from the CPU to the memory (or I/O device).

Control bus: is a collection of control signals that coordinate and synchronize the whole system.

6.0 TUTOR-MARKED ASSIGNMENT

1. Explain with example four output and input device you know.
2. What is system bus?
3. Explain briefly the three types of system bus.
4. How does the microprocessor communicate with the memory and input/output devices?

7.0 REFERENCES/FURTHER READING

Nasrat, H. "Introduction to Computer & Microcomputers."

MODULE 3

UNIT 1 BASIC CONCEPT OF MICROCOMPUTER

CONTENTS

1.0 INTRODUCTION

A computer system can be divided into four components:

- the hardware (CPU, memory, input/output devices, and so on)
- the operating system
- the system programs (word processors, spread sheets, accounting software, compilers)
- the application programs.

This various components will be discussed in this unit.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain microcomputer
- discuss microcomputer revolution.

3.0 MAIN CONTENT

3.1 Description Microcomputers

A microcomputer (micro) is a digital computer whose processing unit consists of one or more microprocessors, and includes storage and input-output facilities. Today there are fabricated microcomputer assemblies including almost all of the basic microcomputer parts and they all are called the *single-chip microcomputers or microcontrollers.* Such a microcomputer reduces the computer to a small, inexpensive, and easily replaceable design component.

A computer system can be divided into four components:

- the hardware (CPU, memory, input/output devices, and so on.)
- the operating system
- the system programs (word processors, spread sheets, accounting software, compilers,…)
- the application programs

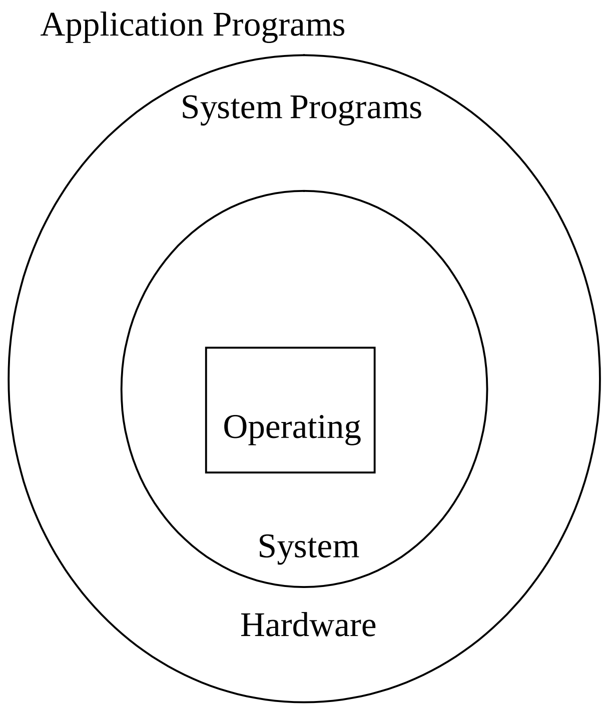


Figure 4: Hierarchy of computer components

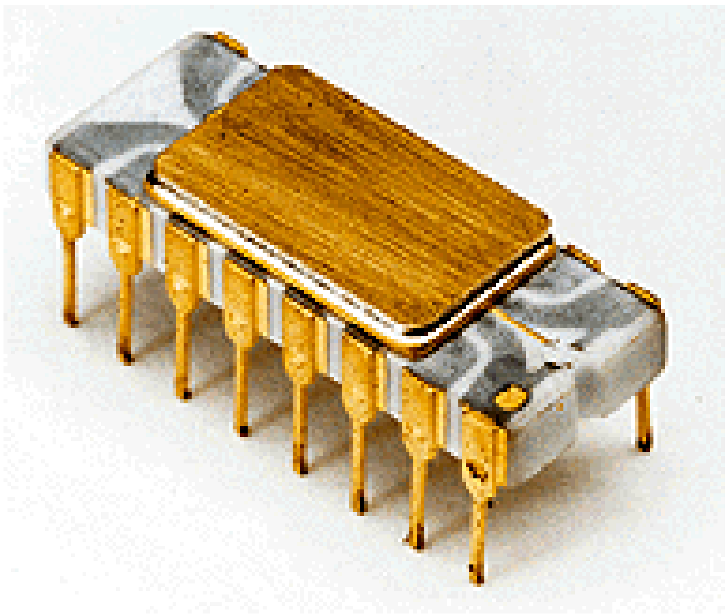
3.2 The Microcomputer Revolution

Integrated circuit:

- First patent to J.S. Kilby in 1959. Introduced by Texas Instruments and Westinghouse in the early 1960s, came into real commercial use in 1964 with Fairchild Semiconductors 702 linear IC.
- The first desktop electronic calculators were introduced about 1963. In 1965, Texas Instruments began work on a four-function pocket calculator based on a single IC--they patented that design. They had trouble getting their invention into production--it came out in 1972 at $150 (the Japanese got copies on the market as early as 1970).

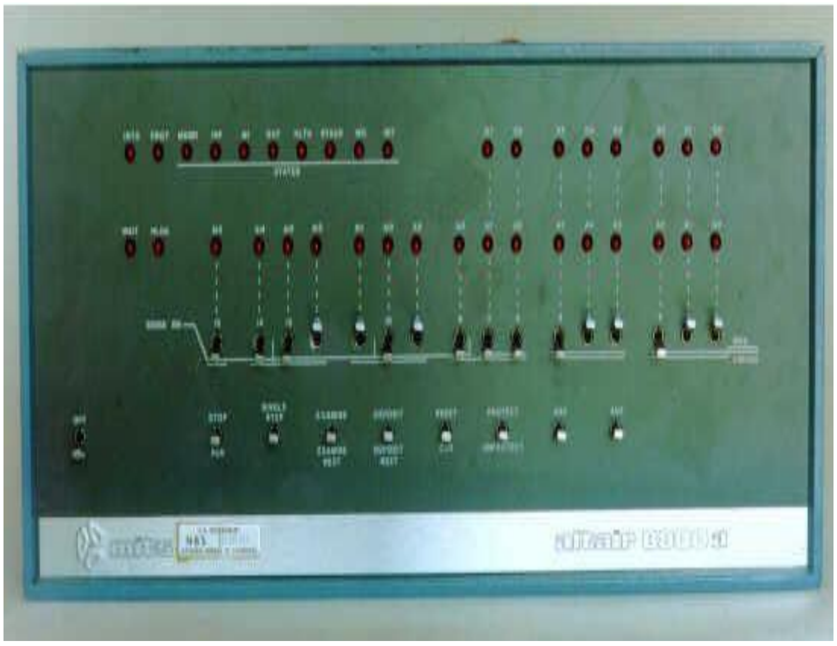
- These had a more predictable market and got more initial interest from big companies than the personal computer. They also had a tremendously steep price curve--by 1975, you could buy a 4-function calculator for under $20.
- Integrated circuits were also used for special-purpose word processors by Wang (1971) and IBM (the Display writer)

The key step for the computer was the introduction of the microprocessor:



Intel 4004

- The idea of creating an entire simple computer from integrated circuits began with amateurs.
- The first microprocessor was the Intel 4004 in Nov. 1971 and the 2 MHz 8080 in 1972. The 8080 was an 8-bit microprocessor that could access 64 k of memory.
- In April 1975 Altair starts selling a kit (called the Altair 8800) with a 8080 microprocessor and 1 kilobyte of memory for $375--users entered data in binary form with 16 toggle switches and read the results in binary form on an array of 16 pairs of panel lamps. Users had to develop their own applications using machine language



Altair 8800

- Gary Kildall was already at work writing the CP/M operating system
- In 1975, a BASIC interpreter was demonstrated on the Altair, and IBM introduces a 55 lb. luggable computer with BASIC, 16 KB of RAM, and tape storage for the price of $9000.
- In June 1976 Southwest Technical Products Company offered a machine with an editor and assembler, and shortly afterwards with a BASIC interpreter. This expanded the potential market, though it was still hobbyists, and all sorts of small companies lept into the business.

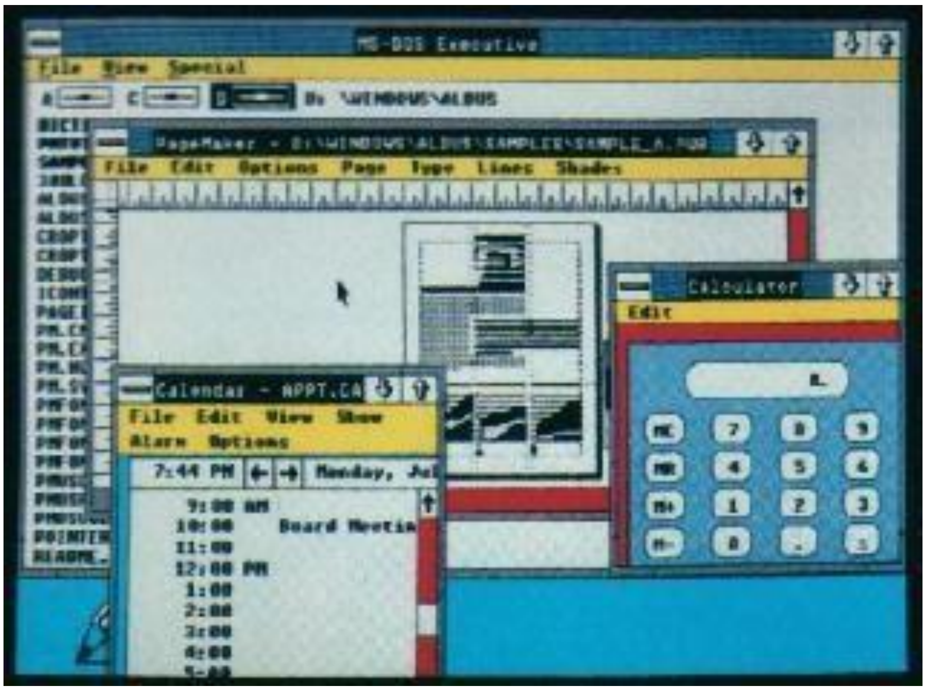
In 1977, the home computer began to move past the electronics hobbyist:

Apple II

- Apple II and Commodore PET came to the market in 1977--both included keyboard, game paddles, BASIC in ROM, and storage of data via audiocassette recorder. The Apple II cost $1300 and had 4 KB RAM--175 kits were sold in the first 10 months. The first model did not include a monitor--you hooked it to a TV set for display. Only in Jan 1977 did Apple Computer move from Steve Jobs's garage to an office.
- In Aug. Radio Shack announced the TRS-80 microcomputer with 4 KB RAM, keyboard, video display, and tape cassette for $600--10,000 are sold in the first month.
- the 8086 microprocessor in 1978 had a speed of 4.77 MHz, 16 bit processing, and could access 1 MB of memory
- The first application was VisiCalc in 1979--the first spreadsheet--creating a business market. Applewriter and WordStar were released in the same year--they had been preceded by Electric Pencil in 1977.
- In the 1979 fiscal year Apple sold 35,000 Apple II computers, 78,000 in 1980
- The most popular computer of the time was the Commodore 64, introduced in 1982, which sold an estimated 22 million units. It had colour graphics and 64 K of RAM; it cost only $400 (it

hooked to a TV set for display) and was used primarily for games.

- Lotus 1-2-3 introduced in 1982 was the killer business application--the microcomputer became more than a toy. dBase was the first data base program. WordPerfect was introduced in 1984 and came to dominate the wordprocessing market.
- Apple tried to compete with the IBM PC with the Macintosh, introduced in 1984 at a price of $2495. A revolutionary new way of using a computer, with a graphical user interface, a mouse, and What-You-See-Is-What-You-Get word processing.
- The IBM AT in 1985 went in a different direction, with a speed of 6 MHz, 512 KB of RAM, and a 20 MB hard drive for about $5000.
- Commodore released the Amiga in 1985 with much improved graphics and sound, but people did not see any use for it except for games.
- Windows was first released in 1985 but did not earn a following until Windows 3 in 1990. Apple sued Microsoft for copying the Macintosh user interface, but lost. Windows represented a major change in the way people used PCs.



Windows 286 (Windows 2)

3.3 Basic Blocks of a Microcomputer

If we think of the computer as an information manipulation device, then the basic components of a microcomputer are:

Input Units -- ***"How to tell it what to do"***

Devices allow us to enter information into the computer. A keyboard and mouse are the standard way to interact with the computer. Other devices include mice, scanners, microphones, joysticks and game pads used primarily for games.

Output Units -- ***"How it shows you what it is doing"***

Devices are how the manipulated information is returned to us. They commonly include video monitors, printers, and speakers.

Memory -- "How the processor stores and uses immediate data"

3.4 Uses of a Microcomputer

"Microcomputer" is the term coined in the 1970s for a personal computer. Until that point, computers had been bulky room-sized electronics; even the smallest models were the size of large cars. The microcomputer has many uses, especially in the home, in business and in the medical field.

1. Home

Families use microcomputers for education; software can hold thousands of book volumes worth of information. In addition, the first portable video games were built for the microcomputers. The home microcomputers paved the way for the invention of laptops.

2. Business

Businesses took a huge leap forward in bookkeeping, inventory and communication when microcomputers were made readily available. An owner could have years of information at the tap of a button instead of going through multiple cabinets of documents and receipts. Information also can be saved to disks and shared among branches of a business, making the business able to trade inventory and update sales figures much faster than before.

3. Medical Uses

The first microcomputer (dubbed the "Sac State 8008") was built specifically for storing medical records. Before microcomputers were available, medical records were stored in paper form. Microcomputers make it possible to download patients' medical histories.

4.0 CONCLUSION

Microcomputer

5.0 SUMMARY

A microcomputer (micro) is a digital computer whose processing unit consists of one or more microprocessors, and includes storage and input-output facilities.

The microcomputer has many uses, especially in the home, in business and in the medical field.

The computer as an information manipulation device the basic components of a microcomputer are:

Input Units -- "How to tell it what to do"
Output Units -- "How it shows you what it is doing"
Memory -- "How the processor stores and uses immediate data"

6.0 TUTOR-MARKED ASSIGNMENT

1. What is microcomputer?
2. What are the uses of microcomputers?
3. Explain briefly the evolution of microcomputers.

7.0 REFERENCES/FURTHER READING

Kastner, D. (2003).“Embedded System”

Mack Pamela E. (2005). “The Microcomputer Revolution” http// www. Clemson.edu/caah/history/Facultypages/Pammack/lec122/micro. htm

UNIT 2 MICROCOMPUTERS ARCHITECTURE

CONTENTS

1.0 INTRODUCTION

Identify the separate components of your computer system if you have a desktop computer. Note, for instance, that while your monitor -- the piece of equipment that contains your screen -- is the primary part of the computer that you pay attention to, it does not do the computing. Locate the tower or CPU (central processing unit), which is the part of your system that is actually the "computer," by finding your on/off switch. Peripherals such as the "mouse" and "keyboard" help you use the computer, but they have only limited computer characteristics within them.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain microcomputer architecture
- identifies types of Microcomputers
- discuss the Von Neumann Architecture
- discuss the Harvard-Architecture

3.0 MAIN CONTENT

3.1 Microcomputer Architecture

In computer engineering, computer architecture is the conceptual design and fundamental operational structure of a computer system. It is a blueprint and functional description of requirements (especially speeds and interconnections) and design implementations for the various parts of a computer — focusing largely on the way by which the central processing unit (CPU) performs internally and accesses addresses in memory.

Computer architecture comprises at least three main subcategories

- ***Instruction set architecture*, or ISA**, is the abstract image of a computing system that is seen by a machine language (or assembly language) programmer, including the instruction set, memory address modes, processor registers, and address and data formats.
- ***Microarchitecture***, also known as *Computer organization* is a lower level, more concrete, description of the system that involves how the constituent parts of the system are interconnected and how they interoperate in order to implement the ISA. The size of a computer's cache for instance, is an organizational issue that generally has nothing to do with the ISA.
- ***System Design*** which includes all of the other hardware components within a computing system such as:

a. system interconnects such as computer buses and switches
b. memory controllers and hierarchies
c. CPU off-load mechanisms such as direct memory access issues like multi-processing.

Once both ISA and micro architecture has been specified, the actual device needs to be designed into hardware. This design process is often called *implementation*. Implementation is usually not considered architectural definition, but rather hardware design engineering.

Computer Organization deals with the advances in computer architecture right from the Von Neumann machines to the current day super scalar architectures.

3.2 Types of Microcomputer Architecture

This section will discuss two types of microcomputers, which are:

- **Von Neumann Architecture**
- **Hardvard Architecture**

1. Von Neumann Architecture

The von Neumann architecture is a computer design model that uses a processing unit and a single separate storage structure to hold both instructions and data as shown in Figure 5. It is named after mathematician and early computer scientist John von Neumann. Such a computer implements a universal Turing machine, and the common "referential model" of specifying sequential architectures, in contrast with parallel architectures. The term "stored-program computer" is generally used to mean a computer of this design, although as modern computers are usually of this type, the term has fallen into disuse. All general-purpose computers are now based on the key concepts of the von Neumann architecture.

Though the von Neumann model is universal in general-purpose computing, it suffers from one obvious problem. All information (instructions and data) must flow back and forth between the processor and memory through a single channel, and this channel will have finite bandwidth. When this bandwidth is fully used the processor can go no faster. This performance limiting factor is called the *von Neumann bottleneck.*

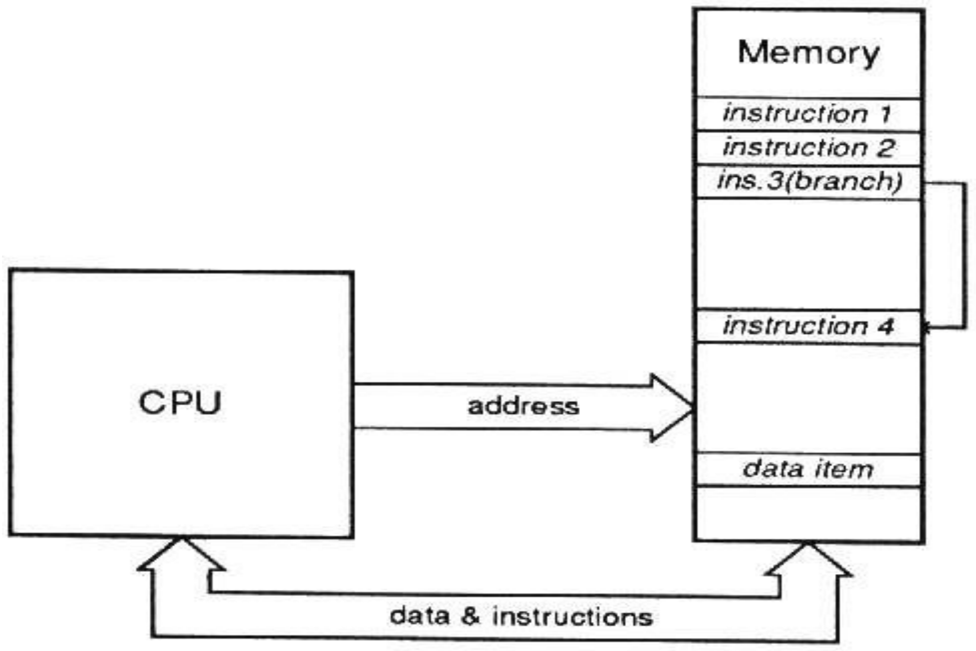


Figure 5: von Neumann architecture

2. Hardvard Architecture

A *Harvard Architecture* as shown in Figure 6 has one memory for instructions and a second for data. The name comes from the Harvard Mark 1, an electromechanical computer which pre-dates the stored-program concept of von Neumann, as does the architecture in this form.

It is still used for applications, which run fixed programs, in areas such as digital signal processing, but not for general purpose computing. The advantage is the increased bandwidth available due to having separate communication channels for instructions and data; the disadvantage is that the storage is allocated to code and data in a fixed ratio.

In Harvard architecture, there is no need to make the two memories share characteristics. In particular, the word width, timing, implementation technology, and memory address structure can differ. Instruction memory is often wider than data memory. In some systems, instructions can be stored in read-only memory while data memory generally requires read-write memory. In some systems, there is much more instruction memory than data memory so instruction addresses are much wider than data addresses.

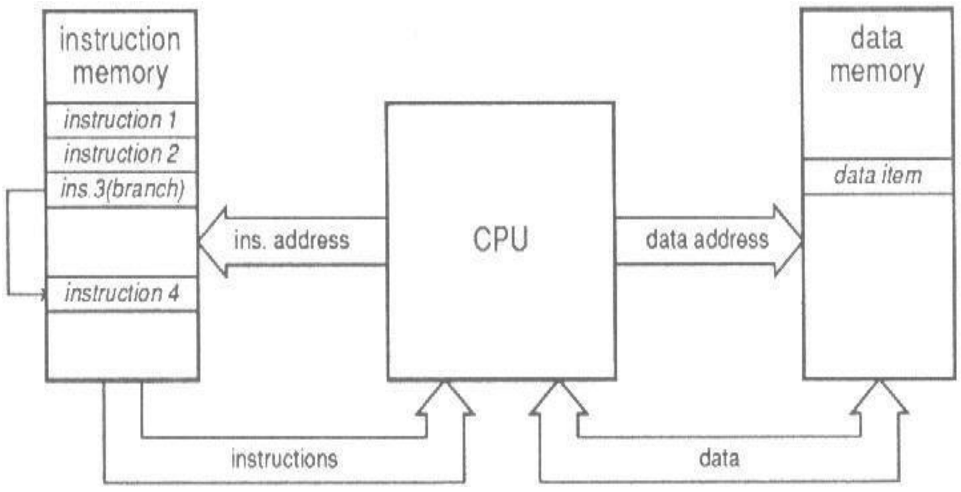


Figure 6: Hardvard Architecture

4.0 CONCLUSION

The two major microcomputers architectural design is discussed which the main foundation for architectural design is. The Von Neumann Architecture and Hardvard Architecture are well discussed.

5.0 SUMMARY

Computer architecture is the conceptual design and fundamental operational structure of a computer system. It is a blueprint and functional description of requirements (especially speeds and interconnections) and design implementations for the various parts of a

computer — focusing largely on the way by which the central processing unit (CPU) performs internally and accesses addresses in memory.
Von Neumann architecture is a computer design model that uses a processing unit and a single separate storage structure to hold both instructions and data.

A *Harvard Architecture* has one memory for instructions and a second for data. The name comes from the Harvard Mark 1, an electromechanical computer which pre-dates the stored-program concept of von Neumann.

6.0 TUTOR-MARKED ASSIGNMENT

1. Explain the von Neumann Architecture.
2. Differential between the Hardvard architecture and von Neumann architecture.

7.0 REFERENCES/FURTHER READING

Microprocessor Architecture
"http://www.ehow.com/way_5581467_microprocessor-architecture-
"tutorials.html

UNIT 3 MICROCOMPUTERS NETWORKING

CONTENTS

1.0 INTRODUCTION

A network is a group of computers and other devices, such as printers and modems, connected to each other. This enables the computers to effectively share data and resources. The concept of sharing resources over a network is called networking. The computers in a network can share data, messages, graphics, printers, fax machines, modems, and other hardware and software resources.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain microcomputer networking
- identify types of microcomputer networking.

3.0 MAIN CONTENT

3.1 What is Microcomputer Networking?

A computer network consists of several computers that are connected to one another using devices that allow them to communicate. Computer networks may be connected using network cables called Cat5 cables and network cards. Your network may also be wireless and connected using wireless routers. If your computer network is wireless, it would not require that they be connected using hardware. When the computers are connected via a computer network, they are able to transmit files from one computer to another. These computers are also able to connect to the internet using one connection.

3.2 Types of Microcomputer Networking

When one computer needs to talk to another, they do so through a vehicle known as a network. The network is responsible for taking data sent by one computer and passing it along a predefined route to another computer. Depending on where the computers are and what sort of reliability are required, different networks are used in various environments.

Personal Area Networks

The personal area network, commonly known as a PAN, refers to an individual computer's network of peripherals. Devices that are in immediate access to your computer, such as a printer or IP telephone, are commonly said to exist within a PAN.

Local Area Networks

The local area network (LAN) is designed for working in a small regional space. An example of a LAN is a home with two or more computers connected to a single router. These computers share network resources and can communicate with one another through the connecting device. Wireless access points are an example of a LAN connector, as are hubs and switches. LANs are ideal for fast transfer of data, as the short distance enables above-average data rates.

Campus Area Networks

The campus area network (CAN) is similar to a local area network; however, it contains a great deal more router complexity in order to link buildings and establish different network classifications. A university might deploy a CAN by having an administrative network with access to important records, such as grades and bills, linked across 20 buildings. It could then establish a second class of network over the same 20 buildings for students, requiring different log in policies while disabling access machines containing sensitive information.

Metropolitan Area Networks

A metropolitan area networks, also known as a MAN, links a much larger geographical region than a CAN, LAN or PAM. These networks typically span the entirety of a city and are commonly used by public utilities companies and state services to maintain a private network that covers all regional boundaries.

Wide Area Networks

The wide area network (WAN) is the type of network most users are intimately familiar with. These large networks require complex forwarding and addressing schemes because they must transmit data across wide geographic regions. The most famous WAN is the Internet, requiring vast numbers of routers and switches to forward data based upon Internet Protocol addresses. While these networks are of great benefit to wide-reaching communications, they are typically not as reliable or as fast as smaller networks.

3.3 Host-Based Networks vs. Client Server Networks

Host-based networks and client-server networks differ in the type of application architecture they utilise. As their names suggest, host-based networks have host-based application architecture, while client-server networks have client-server application architecture. Both types of networks are made up of servers and clients. Client-server networks include web browsers and email clients. Host-based networks are more likely to suffer from server bottlenecks than their client-server counterparts are.

Both host-based and client-server networks are comprised of servers and clients.

Host-based networks and client-server networks differ in the type of application architecture they utilise. As their names suggest, host-based networks have host-based application architecture, while client-server networks have client-server application architecture. Both types of networks are made up of servers and clients. Client-server networks include web browsers and email clients. Host-based networks are more likely to suffer from server bottlenecks than their client-server counterparts are.

Application Programs and Application Architectures

Application programs comprise four functions: data storage, data access, program logic and presentation logic. The application architecture lays out how these four functions are distributed among the servers and the clients. A server can be a mainframe, a minicomputer, a microcomputer or a server farm. A client can be a simple terminal, a microcomputer, a workstation, a network computer or a transactional terminal.

Host-based Network

In a host-based network, the server performs all four-application program functions. The client merely captures the user's keystrokes and sends them to the server. Because all processing is done by the server (or host), the server can become a bottleneck in such a network.

Client-Server Network

In a client-server network, the application program functions are divided between the server and the client. The server handles data storage, and data access. The client handles presentation logic. The program logic may be split between server and client, or assigned to one of the two.

4.0 CONCLUSION

A computer system is a network of electronic components connected by cables. Computer network make communication between computer and the user closer. That is, when one is connecting to the net or through cables; it is easy to access files and communicate. Computer networks are very important in many areas.

5.0 SUMMARY

A computer network consists of several computers that are connected to one another using devices that allow them to communicate. Computer networks may be connected using network cables called Cat5 cables and network cards.

A LAN is a network that is made up of computers that cover small areas, such as a home network or a small office. The network can be connected using a router. A CAN network is a computer network that is made up of several LANS. This type of network is used on college campuses. A MAN computer network is made up of several CAN computer networks. This type of network would use routers and hubs to connect it. A VPN network does not use wires or cable to connect it. This type of connection is also secure and is mainly used in larger environments.

6.0 TUTOR-MARKED ASSIGNMENT

1. Explain briefly what is computer networking?
2. Explain four different types of computer network.

7.0 REFERENCES/FURTHER READING

Nasrat, H. (2001). “Microprocessors Architecture.” Lecture note.
http://www.ehow.com/list_6117483_types-computer-networking.html

UNIT 4 DIRECT MEMORY ACCESS (DMA)

CONTENTS

1.0 INTRODUCTION

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- explain how microcomputers memories are structured
- explain Direct Memory access
- identify benefits of DMA
- explain why we need DMA.

3.0 MAIN CONTENT

3.1 Memory Organisation and Segmentation

The 80386 system is used to explain how memory is organised and segmented. The physical memory of an 80386 system is organised as a sequence of 8-bit bytes. Each byte is assigned a unique address that ranges from zero to a maximum of 232-1 (4 gigabytes). 80386 programs, however, they are independent of the physical address space. This means that programs can be written without knowledge of how much physical memory is available and without knowledge of exactly where in physical memory the instructions and data are located; the model of memory organisation seen by applications programmers is determined by systems-software designers.

The architecture of the 80386 gives designers the freedom to choose a model for each task. The model of memory organisation can range between the following extremes.

- A "flat" address space consisting of a single array of up to 4 gigabytes.
- A segmented address space consisting of a collection of up to 16,383 linear address spaces of up to 4 gigabytes each.

3.2 What is Direct Memory Access?

A computer memory board is an essential hardware component of a personal computer.

Direct Memory Access, or DMA, is a method of accessing and transferring data stored in a computer's random access memory, or RAM, without having to involve the computer's central processing unit. This reduces demands on the CPU and allows for greater efficiency in running processes.

3.3 Why was Direct Memory Access Needed?

The CPU in early computers was involved in almost all activities that occurred in all processes. This placed huge demands on the CPU's attention and slowed it down with requests from peripheral devices. This drain on efficiency was addressed by developing methods that allowed peripherals to access memory directly and transfer data from memory without interrupting the CPU, thereby increasing efficiency.

3.4 How Does Direct Memory Access Work?

Direct memory access is accomplished by the use of special communication paths called DMA channels that are assigned to a specific peripheral. If a peripheral device has the required processing capability, it can use one of the DMA channels to transfer data from RAM own its own without consulting the CPU. It is like a toddler having to get mummy's (the CPU) help to get a drink while the teenaged sibling (with greater processing ability) can get it himself without interrupting mummy.

3.5 Benefits of Direct Memory Access

Modern computers are able to simultaneously run many processes at once. Without the benefit of direct memory access, the CPU could be tied up a great percentage of time responding to slower peripherals needing access to information stored in memory. A device using one of the several DMA channels that modern computers come with can bypass the CPU. This allows all processes to run faster.

3.6 What Devices Use DMA?

It is standard for most computers to have a total of eight DMA channels. They are numbered 0-7 with DMA channel 0 always reserved for system use. The remaining seven channels can be used by various devices, such as floppy drives, sound cards, hard drives, CD-ROM drives and scanners.

4.0 CONCLUSION

The memory of a microcomputer is an essential part of the system. The ability to access data is important to know how the memory are arranged and structured.

5.0 SUMMARY

The model of memory organisation can range between the following extremes.

- A "flat" address space consisting of a single array of up to 4 gigabytes.
- A segmented address space consisting of a collection of up to 16,383 linear address spaces of up to 4 gigabytes each.

Direct Memory Access, or DMA, is a method of accessing and transferring data stored in a computer's random access memory, or RAM, without having to involve the computer's central processing unit.

6.0 TUTOR-MARKED ASSIGNMENT

1. What is Direct Memory Access (DMA)?
2. Explain the reason why we need DMA.
3. What are the benefits of DMA?

7.0 REFERENCES/FURTHER READING

Nasrat, H. (2001). “Microprocessors Architecture” Lecture note.