МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА

УЧЕБНЫЕ ЗАДАНИЯ ПО ТЕМЕ «ИНТЕЛ 8085А»

САМАРА 2003

Рецензия

Учебное пособие предназначено для студентов VI факультета. II курса.

Цель пособия – обучение студентов различным видам чтения, а также совершенствование навыков говорения в области программирования.

Пособие состоит из 5 частей. Каждая часть сопровождается аннотацией по пройденной теме, вопросами и задачами, а также англо-английским словарем. Словарь даёт полное представление о технических терминах данного урока и включает в себя расшифровку аббревиатур и акронимов, широко используемых в технической литературе. Структура предложений и текстов пособия просты для усвоения содержания и задач, поставленных перед студентами.

Рецензент А.Г. Лещенко

Составители: А.В. Романов

ББК Ш

Учебные задания по теме «Интел 8085А». Учебные задания по англ. яз. /Самарский государственный аэрокосмический университет; Сост. А.В.Романов. Самара, 2003. 39с.

Тексты пособия дают возможность ознакомиться с работой терминологии. Задачи и вопросы направлены на совершенствование навыков чтения и умения передавать содержание.

Печатаются по решению редакционно-издательского совета Самарского государственного аэрокосмического университета имени академика С.П. Королева.

Рецензент: А.Г. Лещенко

UNIT I

INTRODUCTION TO MICROPROCESSORS.

HISTORY OF THE MICROPROCESSOR

The microprocessor was conceived by Intel Corporation in California in 1971 as a 4-bit microcontroller designed for use in an electronic calculator. It did not take long before other applications for this device were dreamed up by inventive minds at such companies as Bailey Corporation. One of the early applications of this device was a video arcade game.

THE INTEL 4004

This early computer on a chip was the Intel 4004 microprocessor, which could add two 4-bit binary numbers and perform many other operations. (A bit is binary digit capable of storing a 1 or a 0.) You might think that a 4-bit microprocessor has very limited application today, but believe it or not, it still has wide application. If you were to buy a microwave oven or similar device, you would probably find that it contained a 4-bit microprocessor. Two of the most common microprocessors used in applications such as microwave ovens are the TMS-IOOO, which is manufactured by Texas Instruments, and the Intel 4040, which is an updated version of the original 4004.

The 4004 was a fairly primitive device by today's standards, capable of addressing 4096 different memory locations. For many applications this is not enough memory. In addition to memory size, the word size of the 4004 proved to be too restrictive in many cases. A 4-bit number can store only 16 possible codes. For the many applications that handle alphabetic informations, 4 bits proves to be too limiting because there are 26 letters in the alphabet plus a variety of special characters and numbers. It normally takes at least a 6-bit and often a 7-bit binary number to encode all of these different characters. To solve this problem Intel introduced an 8-bit microprocessor, the 8008.

THE INTEL 8008

The Intel 8008 could handle 8-bit numbers(bytes), which was a great improvement over the 4004.(A byte is usually equal 8 bits.) In addition to the increase in word size,the memory size was increased from 4096 four-bit words in the 4004, to 16K eight-bit words in the 8008.(A computer K is equal to 1024.) The 4004, and the 8008 could both add numbers at the rate of 20,000 per second, which at the time was fast enough for many of the early applications. With engineers dreaming up new tasks for the microprocessor every day, this speed eventually began to limit the application of these early microprocessors. The main reason that these early devises were so slow that they were constructed from the then state-of-the-art PMOS logic circuitry. (PMOS in an acronym for P-channel metal-oxide semiconductor.) PMOS logic is inherently slow, and in addition to being slow, it is fairly difficult to interface to standard TTL logic because it uses a negative power supply.
Then, at about this same period of time, there was a breakthrough in the fabrication of NMOS logic circuitry. (NMOS is an acronym for N-channel metal-oxide semiconductor.) NMOS is much faster than PMOS logic and it also uses a positive power supply, which makes it more readily adaptable to be connected to TTL logic. This is important because many of the ancillary integrated circuits connected to a microprocessor are TTL circuits. NMOS allowed speeds to be increased by a factor of about 25 times, which is significant. This new technology was used in the construction of the now famous Intel 8008 microprocessor.

THE INTEL 8080

The Intel 8080 was introduced in 1973 and its introduction is responsible for catapulting the world into the age of microprocessors. The 8080 was a greatly enhanced 8080 that could perform 500,000 operation per second and address 64K bytes of memory space. This device was also responsible for ushering in the age of the home computer, which was first introduced by MITS in 1974. This first hobby or home computer was the Altair

8800, which generated keen interest in microprocessing.
Although the Altair 8800 computer is no longer manufactured, it did represent a change in the way people viewed the computer. A computer was no longer viewed as a mystical device suited only for large corporations or the military. It was a device that would, in a few short years, begin to populate American homes in the form of Apples, Ataris, Commodores, IBMs, and various other microproces-sor-based systems.

OTHER EARLY MICROPROCESSORS

Up until 1973, Intel was the major producer of microprocessors; then other manufacturers began to see that this new device had a future and started to manufacture their own modified versions of the Intel 8080 microprocessor. Many of the early microprocessors are no longer actively produced because of the popularity exhibited by the Intel 8080 microprocessor.

THE MICROPROCESSOR

What is a microprocessor? The microprocessor is a device that can be commanded to perform a variety of various functions-it is programmable controller. All microprocessors perform the same three basic functions is a system: data transfer, arithmetic and logic, decision making. These are the same three tasks that can be performed by any microprocessor, minicomputer, or mainframe computer system.

BUSES

The address bus connections are used to supply a memory address or an I/O address to the memory and I/O blocks. The address, which is a binary number, is used to point to a unique memory location or I/O device. Memory can be envisioned as a series of numbered (addressed) boxes that hold an 8 - bit binary number in an 8- bit microcomputer. The data bus connections, which would be 8 bits wide for an 8- bit microprocessor, are used to carry the information to or from the memory and I/O. Data bus lines are, in most cases, bidirectional lines capable of transmitting information in either direction. The control bus connections are used to control both the memory and I/O systems. The control bus consists of three general signals: RD (read), WR (write), and IO/M (input / output or memory). RD is used to control the reading of data from memory or I/O, WR is used to control the writing of data to memory or I/O, and IO/M is used to select I/O or memory.

DATA TRANSFER

The microprocessor spends a great deal of its time transferring data to and from the memory and I/O. About 50 percent of its time is spent fetching instructions from the memory and preparing to execute them. Computers today store the commands or instructions in their memory so that they can be executed at a very high rate of speed.

ARITHMETIC AND LOGIC

A small portion of the microprocessor's time is spent performing arithmetic or logic operations. These operation can be executed in 1 to 2 microseconds in most microprocessors. Arithmetic and logic operations found in many microprocessors.

Operations	Notes
Addition	
Subtraction	Two's-complement addition
AND	Logical multiplication
OR	Logical addition
XOR	Exclusive -OR
NOT	Inversion
Shift	Either arithmetic or logical
Rotate	

DECISIONS

The ability of the microprocessor to make decisions allows it to process information more efficiently. It also allows the programmer to develop software that can choose different paths through a program. This ability has made the computer system the powerful device that it is today. Microprocessors make decisions based on numerical facts,

Common testable conditions used by microprocessors to make decisions.

Numeric condition.	Testable conditions
Zero	Equal to zero
	Not equal to zero
Sign	Positive
	Negative
Parity	Odd parity
	Even parity
Carry	Carry equal to one
	Carry equal to zero

THE MEMORY SYSTEM

The memory system in a computer performs two very important tasks:(l) memory is used to store the instructions of a program, and (2) memory is used to store data for use by the program. In most systems the program is stored in a read-only memory (ROM) and the data are stored in a random access memory (RAM), the term used for a read and write memory.

THE PROGRAM

The program in a computer system is made up of various instructions that direct the operation of the microprocessor. Instructions are binary numbers that are interpreted by the microprocessor as various operations that are to be executed by the microprocessor. A grouping of these instructions is called a program. Programs vary in length from a few hundred instructions in simple systems to thousands of instructions in complex systems.

A simple example of a program is adding the numbers 3 and 2 together. To direct the computer to perform this addition, a series of instructions are stored in the memory as a program. The first instruction fetches the 3 from the memory. This is comparable to writing it down on a piece of paper. The second instruction gets the 2 out of the memory and adds it to the 3. This is comparable to writing the 2 beneath the 3 and adding them together in your mind. Finally, a third instruction is required to store, or save, the answer in the memory-the same thing that you do when you write the answer down on a piece of paper. Notice how the simple task of adding two numbers together seems more difficult than normal. This is because each step must be though of and written down as a sepa-rate instructions in a program.

TYPES OF MEMORY

As mentioned earlier, the ROM, or some form of ROM such as an EPROM(erasable programmable real-only memory), is used to store the program and the RAM is used to store the data. This is not always true, of course , but it is many cases. The ROM memory that is commonly found associated with the microprocessor is typically either a factory- or mask-programmed ROM or a field-program-mable EPROM The EPROM is programmable in the field by a device called an EPROM programmer. In addition to being programmable, the EPROM is constructed using a process that allows it to be erased if a mistake is detected.

The RAM memory is most often some form of NMOS or CMOS memory device. In small systems, memory is often constructed from static RAM (SRAM) and large memory systems are often constructed from dynamic RAM (DRAM).

INPUT/OUTPUT DEVICE

The input/output block in computer systems is the microprocessor's connection to the outside world. The microprocessor communicates to human beings and/ or machines through this block. Without this communications path, the computer would truly be a worthless device. I/O devices are defined as devices that can either accept an electrical signal for processing or generate an electrical signal to accomplish work. Just about any device in use by today's modern society could be, and often is, interfaced to microprocessor as an I/O device. This has largely been made possible since analog-to-digital (ADS) and digital-to-analog (DAS) converters are available.

Common input/output devices

Device	Type	Common usage
Switch	Input	Used to sense events such as key depressions on a keyboard, limits in certain mechanical systems, etc.
ADS	Input	Used to convert any analog voltage into a digital signal that can be sensed by the microprocessor. Indicator Output An LED or similar device that can indicate a single condition or, in some cases, display a numeric or alphabetic character.
Motor	Output	Used to position external mechanical devices; may be a stepper motor or an ac or dc motor.
Solenoid external	Output	Used to control electrical or contacts or physically move relay an device.
DAC	Output	Used to convert the digital output of the microprocessor into an

analog

voltage.

TRAINER FUNCTIONS

The trainer normally contains a hexadecimal keypad for data entry plus additional keys for control. The keys control:

1. The reading of data from the memory.
2. The writing of data to memory.
3. The displaying and modifying of the contents of the microprocessor's registers.
4. The execution of a program stored in the memory.
5. The debugging of a faulty program by the use of a single-instruction execution mode of operation.
6. The resetting of the microprocessor if the program becomes entangled in an infinite loop.

The trainer also contains a set of numeric displays that are capable of displaying hexadecimal data as a two-digit number and four-digit number. The displays normally indicate any one of the following:

1. A sign-on message such as 8085, which is displayed when the unit is first turned on or reset.
2. An address and data from any memory location.
3. The contents of an internal register and its letter designation.
4. Some indication that a user program is being executed, such as the letter E.
5. Breakpoint addresses during programming.

In addition to the keyboard and displays, the trainers also have an area that can be used to expand the circuitry. For example, suppose that you wished to add an analog-to-digital converter to the trainer far an experiment. This is accomplished by wiring up the circuit on the breadboard area of the SDK-85 or on an external breadboard for the

Micromodule-85.

SUMMARY.

1. Intel Corporation developed the first microprocessor, the 4004, a 4- bit machine that could directly address 4K nibbles of memory.
2. Four - bit microprocessors still find wide applications in such systems as microwave ovens, dish-washers, and even in automobiles.
3. Eight - bit microprocessors are much more useful in many applications because alphabetic data requires at least a 6 - bit binary code for representation and more often a 7 - bit code.
4. PMOS logic, which was used in early microprocessors, proved to be too slow. This led to the development of NMOS logic and the modern gender of the microprocessor.
5. As early as 1974, the home computer began to appear on the scene in the form of the Altair 8800 computer system.
6. Today microprocessors are available in 4-, 8-, 16-, and 32-bit versions.
7. The microprocessor is a device that performs arithmetic and logic, data transfer, and makes some rudimentary decisions based on numerical facts.
8. Buses are used to control the system connected to a microprocessor. All microprocessors contain an address bus, a data bus, and a control bus.
9. The memory in a microprocessor - based system is used to store the instructions of a program and the data used by a program.
10. A program is a collection of instructions that are used to perform a useful task in a computer.
11. RAM (random access memory) is used to store temporary data, and ROM (read-only memory) is used to store programs and more permanent data.
12. An input / output device is a device that will either accept an electrical signal or generate one.
13. A microprocessor trainer is a device that is used to learn how to program the microprocessor. Most trainers allow the user to enter data or commands, view the data

from the memory or internal registers, and execute or single-step through a program.

GLOSSARY.

Address bus - A set of connections that are used to provide the memory and 1/0 with an address.

Bit - One binary digit (0 or 1) or position in a binary number. Also a contraction for binary digit.

Buses - Common paths that are used to interconnect the many components in a system.

Byte - Generally, an 8-bit binary number. A contraction formed from binary (B) and eight (YTE). ***CMOS*** - Complementary metal-oxide semiconductor technology-.

Control bus - A common set of connections that is used to control both the I/O and the memory.

Data bus - A common set of connections that is used to transfer all the information in a computer system.

DRAM - Dynamic random access memory.

EPROM - An erasable programmable read-only memory is a device that can be erased with an ultraviolet lamp and reprogrammed electrically.

Input device. A device, such as a keyboard, that sends data to the microprocessor.

I/O - The input/output equipment provides the microprocessor with its interface to the outside world. ***K*** - In computer terminology a K is generally equal to 1024 decimal.

Mainframe computer - A mainframe computer system is a large computer designed to handle extremely large jobs. The term mainframe comes from the fact that the machine is often mounted in one or more rack panels or frames.

Memory - A device that stores binary information for the microprocessor.

Microcomputer. A computer system constructed around a microprocessor.

Microprocessor. A device that can be programmed to control just about any situation. It is a program-mable controller.

Microprocessor trainer - A device used to learn the operation and programming of a particular microprocessor.

Minicomputer - A scaled-down version of a mainframe computer that is today losing definition because of the microprocessor-based microcomputer.

NMOS - N-channel metal-oxide semiconductor technology.

Output device - A device, such as a printer, that accepts data from the microprocessor.

PMOS - P-channel metal-oxide semiconductor technology.

Program - A collection of instructions that direct a computer to perform a task.

RAM - Random access memory is memory that can be written into as well as read out.

ROM - Read-only memory is memory that has been permanently programmed with information that can only be read from the memory.

SRAM - Static random access memory.

TTL - Transistor-transistor logic is a bipolar transistor technology.

Word size - Generally, the word size is determined by the width of the microprocessor's internal arithmetic circuitry.

QUESTIONS AND PROBLEMS.

1. What corporation did develop the first microprocessor?
2. What are some of the early applications of the microprocessor?
3. What did lead to the development of the 8-bit microprocessor?
4. How many bits are equal to one byte?
5. How-many decimal memory locations does a 4K memory device contain?
6. What breakthrough did usher in a faster and more versatile microprocessor?
7. List three different 8-bit microprocessors.
8. List three different 16-bit microprocessors.
9. Give a brief definition of the microprocessor.
10. What three main blocks do comprise the block diagram of a typical microprocessor-based computer system?
11. What are three main operations performed by the microprocessor?
12. What buses do interconnect the blocks in a microprocessor-based computer system?
13. List four common arithmetic and logic operations performed by the microprocessor.

14. List three common factors that the microprocessor uses to make a decision.
15. What type of memory is the program in a microprocessor-based computer system stored in?
16. What type of memory are the data of a transitory nature stored in ?
17. Define the term program.
18. What types of RAM read and write memory are found in various computer systems?
19. Define the term input/output.
20. List four commonly found I/O devices.
21. What hardware features are usually found in a microprocessor trainer?
22. What functions are mostly often found in a typical microprocessor trainer?

UNIT II

SOFTWARE DEVELOPMENT.

Before a detailed study of the programming of the 8085A microprocessor can be conducted, it is important to develop an understanding of the programming task. (Programs are often called software.) In this chapter we approach this by introducing the systems most often used to develop software, the development task, various programming approaches, and software design, and discuss assemblers, interpreters, and compilers. Once these areas are fully understood, it is much easier to learn how a particular microprocessor functions and finally, to write efficient, well-documented software.

SOFTWARE DEVELOPMENT SYSTEMS

What is a software development system? A software development system is a machine that has specifically been designed to make software development as efficient as possible. A software development system, should have the following essential component parts:

1. A video display terminal for data entry and display.
2. An editor for entering the source program into the system.
3. An assembler for developing bit manipulation and 1/0 software.
4. A high-level programming language such as PL/M or Pascal for developing the intricate portions of the software, which more often uses complicated arithmetic.
5. A linker, which is used to connect software modules together to form the software system.
6. A disk memory system for saving the software for latter use.
7. An 1/0 port that can connect a printer to the system for printing a hard copy of the

software.

8. An EPROM programmer for storing the completed software on an EPROM.

THE VIDEO DISPLAY TERMINAL

The video display terminal (VDT) portion of the software development system should have a standard typewriter keyboard, including additional cursor positioning keys. It is also desirable to have some programmable function keys or other mechanism for entering often used keying sequences. A numeric keypad can also be helpful at times, especially if the software development task requires the entry of long series of numeric data. For operator convenience the position and angle of the keyboard should be adjustable.

THE EDITOR

The software development system's editor is extremely important because it is used not only to enter the source program into the system, but also to modify it. (The source program is a program written in BASIC or assembly language.) The editor should be capable of deleting unwanted lines of the program, moving entire lines or blocks of lines from one point to another, replacing labels with other labels, finding any line or label, and repeating lines of software. An editor without these minimum features is very hard to use and makes the software development task more difficult. Most modern word processing software can function as an editor because most and usually all of the features are provided.

AN ASSEMBLER

The assembler is a program that converts the symbolic source program into the binary machine language object program for the desired microprocessor. It should at least be capable of indicating simple syntax errors as they appear in the source program. (A syntax error is an error in form: for example, a missing or misplaced comma). The assembler is used to develop the software required to control 1/0 devices because 1/0 control software usually requires a considerable amount of bit manipulation. Bit manipulation is difficult to accomplish efficiently with a high-level programming language. The assembler is also used where the time required to execute the software is critical. For most other software functions a high-level programming language is often more expedient at generating software.

PASCAL OR PL/M

Pascal and PL/M are two examples of high-level programming languages that are quite useful in developing software for a microprocessor-based system. They are commonly used to develop portions of the software that require extensive arithmetic manipulation of complicated data structures. A high-level languages advantage is not software efficiency, because the software it generates is not nearly as efficient as that developed by an experienced assembly language programmer. A high-level languages advantage lies in the fact that it can generate complicated code at a tremendous rate of speed compared to that needed by an experienced programmer using an assembler.

THE LINKER

The linker is used to build the final machine language system by linking together (connecting) all the software modules. Software modules may consist of assembly language programs, high-level-language programs, and subroutines from one or more system software libraries.

THE EPROM PROGRAMMER

The EPROM programmer is an important part of the system because it is the place the final system software is sent after the development task. The EPROM holds the program so that it can be plugged into the hardware system for testing. Whenever software is stored on an EPROM it is called firmware. The EPROM programmer should be able to program a wide variety of different EPROMs. Today, EPROMs vary in size from IK by 8 (2708) to 64K by 8 (27512).

THE DEVELOPMENT TASK

Where does one start when developing a software system? The first thing not to do is to start writing a program. Writing a program in a minimal amount of time, efficiently, requires a considerable amount of forethought. The thinking process is the first step in software development. This section details the proper steps required to develop software in the shortest possible time and at the lowest cost.

PROBLEM DEFINITION

1. What is the form of the input data? (ASCII string, ASCII character, binary byte, binary word, packed BCD, unpacked BCD, etc.)
2. What is the required form of the output data? (ASCII string, ASCII character, binary byte, binary word, packed BCD, unpacked BCD, etc.)
3. Do the input data require a buffer?
4. Do the output data require a buffer?
5. Is there any change required in the form of data between the input and the remainder of the program?
6. Do the output data need to be conditioned before they are sent out of the program?

Suppose that a system uses a simple decimal keyboard for input data and a six-digit display for output data. The data from the keyboard are in BCD code; the output data are also BCD. This means that the data must be in BCD form to enter and exit the software. An output buffer is probably required because there are six digits of output data. These questions are designed to make you think about the problem before the software is actually written.

Once the input and output specifications are made, the program itself can be defined. This section will vary from program to program, but a few steps can be followed for every program, which are listed below.

1. Is there any need to store temporary data? If so, how much space will be required?
2. Are any portions of the program repeated many times? If so, these portions should be developed and debugged and written as subroutines.
3. Can the program be broken into separate areas so that several programmers can develop the software simultaneously?
4. Can a portion of the software be developed with a high-level language such as Pascal or PL/M?
5. How much memory space is required for the entire program?

PROGRAM DEVELOPMENT

Once all of this information is gathered, the programmer or team of programmers can begin developing the software. If the program is rather large, a software manager may break the task into individual assignments for members of a programming team. Teamwork on a large program reduces the total development time considerably if the work is assigned to the most qualified individual programmers. Each section of the software is developed and fully tested. The testing of the software is usually accomplished by using a software driver. The software driver is a program that is written to test every possible input and output condition of the section of software under test. Suppose that you are assigned the task of developing a section of software that is to divide any number by a 12. A software driver to test your part of the software must provide all possible numbers so that your divide program can be fully tested. It must also be able to determine if the outputs obtained are correct.

Once all the software for a large program is fully tested in sections, it is linked together and tested as a unit. The final test usually points out that there has been some error in passing information from one section of the software to another. This would be corrected by respecifying a portion of the software.

THE FINAL TEST

The last step in developing system software is to burn an EPROM and test the software in the finished hardware system. If all goes well, the final system will work correctly. If not, either the hardware is at fault and it must be corrected or an unforeseen software error has occurred. This must be corrected by returning to the software development stage. Hardware testing is accomplished by another device, called an in-circuit emulator. The emulator is a device that replaces the microprocessor in the hardware system. Once the microprocessor is replaced, the emulator can completely control the hardware. Functions that emulators commonly perform are: single-step, register and memory displays, and traces. Each of these functions allows the test engineer to find bugs in software and hardware.

PROGRAMMING APPROACHES

Three basic programming techniques are discussed in this chapter: topdown, modular, and structured programming. Each technique has its merits, but today it seems that the most efficient technique is a combination of all three techniques. The idea of this combination will be introduced after each technique has been fully described.

TOP-DOWN PROGRAMMING

When developing software using the top-down approach, a shell of the entire system is created using dummy calls to lower-level software modules. Suppose that a program is required to read a character from the keyboard and display it on a VDT screen. This process is to continue until the carriage return (enter) key is pressed on the keyboard. Notice that there are three places where lower-level software is used by the program shell: read key, display character, and check for a carriage return. Once the outer shell is constructed, the lower-level software is designed for each of the three tasks. In many cases the lower- level software will form its own shell, which may require that even lower-level software is constructed. This technique is useful for the system design manager because it allows the entire system to be designed without undue detail to the low-level software required to implement the design. The software is designed in a top-down fashion: from the top (outer shell) to the bottom (lower-level software) or inner shell.

MODULAR PROGRAMMING

This approach is not unlike the top-down approach in that the program is broken down into parts. A module by definition is a program that implements just one function. This technique is very helpful to a software manager because the manager can assign portions of a system by choosing and assigning the modules to various programmers. The problems involved with this approach are that it is sometimes difficult to determine the length of a module before it is written and it is often difficult to interface modules.

STRUCTURED PROGRAMMING

Structured programming. It has recently been associated with a standard set of rules that apply to most programs in general. The rules set the structure of the program. The top-down and modular approaches both fit one of the rules: develop programs in a stepwise fashion, and push details lower and lower into simpler software modules.

Another rule, which is often implied with structured programming, is never use a GOTO (BASIC or Pascal language). Instead, use a well-structured sequence like the DO WHILE, REPEAT UNTIL, or IF-THEN-ELSE. GOTO statements can cause a lot of confusions at a later date when the software may require a modification.

This text will develop structures in assembly language that are useful in developing software. These structures, together with a programming methodology for their use, are presented in later chapters together with the software.

The PROGRAMMING APPROACH

As can be seen from the prior discussion of various programming approaches, there is no correct approach. The correct approach is actually a combination of top-down, modular, and structured programming. In fact, it is extremely difficult to separate one approach from another. We will see in later chapters that the structured programming technique aids in developing software because it teaches certain design algorithms (constructs) that can be applied to a wide variety of problems without much modification. These algorithms are the structure in structured programming.

STEPS USED TO DESIGN SOFTWARE

The following steps are most often used to develop a software system:

1. Define the system as clearly as possible.
2. Design the overall system using the top-down approach to develop the outer shell.
3. Determine which modules are to be broken out of the outer shell for assignment to individual programmers (the modular approach).
4. Individual modules are designed using structured programming techniques.
5. The modules are tested using software drivers or other appropriate techniques.
6. The modules are linked together and tested as a system.
7. Based on the outcome of step 6, the system is either redesigned or it is complete and it goes into production.

These steps provide a general method for obtaining the final software product. They are a suggested sequence of events that the author has found to be effective in managing software products over the years. These steps apply to microprocessor-based systems using assembler or high-level languages as well as to systems for much larger mainframe computer systems.

System design. In this step, the specifications are used to create the shell of the program.

Module specification. This step is used to locate modules in the shell of the program for assignment to individual programmers. Not only are the modules selected, but the skill level required to develop the software for each module is also decided. The most qualified programmers are then selected to develop or code the modules using structured programming techniques.

Module design. It is at this stage that the most cooperation is required from each team member. Each person developing each module must communicate with each other programmer so that the data will flow properly from module to module.

Module testing. As mentioned, modules are tested using special programs called software drivers. In addition to testing modules with a software driver, it is extremely important that the programmer go through each step of the software on paper. This can often save a great deal of time in debugging faulty modules later. It also saves time on the software development system, which can be utilized by other programmers.

System testing. The last phase, or at least it should be, is final system testing. This stage is where the actual modules are linked together and the entire system is tested as a unit. (This type of testing can take many months for a large and complicated system.) Based on the outcome of this test, either the faulty module or modules are redesigned or the system is placed into operation in the field. (It is rare that the system is 100 percent functional on the first pass of this design technique.)

Once a system has been placed into service in the field, it is often necessary to modify the system. This happens because if it is a large system, it cannot be fully tested in the laboratory. In fact, very large systems may never be completely debugged.

ASSEMBLERS, INTERPRETERS, AND COMPILERS

This section of the text introduces the assembler used to develop 1/0 and bit manipulation software, the interpreter used for testing high-level-language programs, and the compiler as a software development tool. These three software development tools are used extensively in the field.

THE ASSEMBLER

As mentioned earlier, an assembler is a program that converts symbolic machine code (source code), sometimes called assembly language, into binary machine code (object code). Symbolic machine code is a language that is much easier to understand than binary machine code. It is constructed using mnemonic codes that are easily recognizable by human beings. An example mnemonic code is ADD, which is easy to recognize as addition. If the binary machine code for an addition is written (0100 0111, for example), it is virtually impossible for a beginner or often an experienced programmer to decipher the function.

Binary machine code is the actual language of the microprocessor. The microprocessor does not understand mnemonic code; it only understands binary machine code. For human beings to be able to easily write a program for a microprocessor, a device called an assembler is used to convert symbolic machine language into binary machine language. In addition to converting symbolic machine code into binary machine code, many assemblers also allow the programmer to call up prewritten sequences of instructions called macros. A macro is a sequence of predefined steps that are invoked by using a special command that the programmer has written. An example is division. The 8085A cannot divide. To circumvent this problem, a macro can be written and labeled DIV. Every time a division is required in a program, the programmer can invoke the division macro by using DIV. For an assembler to be truly effective, one more addition is generally provided, the pseudo operation. Pseudo operations are special commands that may or may not generate machine code. One such example is the EQU pseudo operation. EQU is used to equate a label with a value or another label. It is much more readable if your program is written so that the number 7 is equated to the word SEVEN. This is accomplished by using the equate mnemonic.

INTERPRETERS

Most of us are familiar with the interpreter in the form of the BASIC language. In a machine that contains a BASIC interpreter, a program is entered and as it is executed, any error that occurs is indicated at the exact time of the error. This makes debugging a very simple task because in many cases you know exactly which line is in error. The interpreter is capable of doing this because it executes a line at a time. It does not convert the entire program into machine code, as the assembler, and execute it. If it did, it would be very difficult to locate the line or lines of code that are in error. Interpreters are not used directly in system software development for microprocessors. They can be used to recheck a program written in a high-level language before it is compiled (converted into binary machine language).

COMPILERS

If a high-level language is used in program development, it will probably be in the form of a compiler. A compiler is a device that converts a high-level-language program into binary machine code. This makes the program fairly efficient because it usually takes less room than if it were in an interpretive form and it also executes at a much higher rate of speed.

As mentioned before, high-level-language compilers are extremely useful when developing systems that require a lot of complicated arithmetic. They are not suitable for all program development because it is difficult to use them to control 1/0 devices and to manipulate binary data.

SUMMARY

1. The major components of a software development system are: a VDT, an editor, an assembler, a high-level programming language, a linker, a disk memory system, a printer, and an EPROM programmer.
2. Assemblers are used for developing bit manipulation and I/0 control software and

compilers are used for developing intricate arithmetic.

3. The linker connects or links software modules together to form the complete software system.
4. The EPROM programmer is used to store the final software system onto an EPROM.
5. The first thing not to do when developing a software system is to write the program,
6. To define the software problem completely, a lot of thought and information gathering must occur.
7. Three programming techniques are commonly used in software development: top-down, modular, and structured.
8. The top-down programming technique requires that a shell of the program be constructed without any detail to lower-level software.
9. Modular programming is a technique that separates the software into modules which are then assigned to individual programmers.
10. Structured programming requires the use of a strict set of predefined structures, such as DO WHILE and REPEAT UNTIL, when developing software.
11. A good software design methodology requires that the system is defined, top- down techniques are used to design the program shell, modules are broken out of the shell, programmers use structured programming to develop the modules, and extensive testing of all the software, both the modules and the entire system.
12. The assembler is a program that converts mnemonic code, in the source program, into the binary machine code of the object program.
13. Interpreters are high-level-language systems that are excellent for error checking because they execute the software a line at a time.
14. The compiler is a program that converts a high-level language into binary machine language.

GLOSSARY

Assembler - a program that concerts symbolic source code into binary object code.

Assembly language - the symbolic source program is often referred to as an assembly language program.

Binary machine code - the actual binary commands that the microprocessor understands as instructions.

Compiler - a program that converts a high-level language into binary machine language.

EPROM programmer - a device that is used to program EPROMs.

Hacker - an inexperienced programmer who writes software in a random pattern.

Interpreter - a program that accepts high-level-language commands and executes them one at a time.

Linker - a program that connects or links together software modules.

Macro - a sequence of events in symbolic machine code that can be invoked by using a single user-created mnemonic code.

Mnemonic code - an abbreviated command for a computer system, such as ADD for addition and SUB for subtraction.

Module - a section of software that performs one task.

Pseudo operation - a special assembly language command that is used to control the assembler. Pseudo operations may or may not generate machine code.

Shell - the outer portion of a program developed using top-down programming techniques.

Software development system - a system that is used to develop programs that includes a VDT,
assembler, high-level language, linker, printer, disk memory, and EPROM programmer.

Software driver - a program that completely tests software.

Symbolic machine code - a program that uses mnemonic codes instead of binary machine codes.

Syntax error - a mistake in the normal form of a statement in an assembler, interpreter, or compiler.

VDT - the video display terminal contains a video screen and a keyboard.

QUESTIONS AND PROBLEMS

1. What is the purpose of a software development system?
2.List four of the essential components of a software development system.
3.Why is it desirable to have a numeric keypad on a VDT?
4.List three of the operations that an editor can perform.
5.The source program is converted to what type of program by an assembler?
6. What is a syntax error?
7.Where would a high-level language such as Pascal or PL/M be useful when developing software?
8. List two types of software modules linked together by a linker.
9.What is the first thing not to do when writing software?
10. Why is teamwork important when developing software for a large system?
11.What is a software driver, and how is it used in software development?
12.Name the three programming approaches most often used for software development.
13.Which programming approach does use a shell?
14.Which programming approach a software module is used in ?
15.Which programming approach is associated with a standard set of rules?
16.List the seven steps used for proper software design.
17.Should experts always be called in when developing a software system? Explain your answer.
18.Are communications an important part of software development? Explain your answer.
19.Is it normal for a software system to function perfectly at the end of the development cycle? 20.Define the term mnemonic code.
21.Briefly explain the structure of symbolic machine code.
22.What is the difference between symbolic machine code and binary machine code?
23. How does the pseudo operation EQU function?
24.What feature does the interpreter make an excellent high-level-language debugging

tool?

25.Describe the difference between a high-level-language interpreter and a compiler.

UNIT III

8085A ARCHITECTURE.

The key to understanding the microprocessor is its architecture. In this section of the text 8085 A's internal operation, programming model, memory and 1/0 structure, data and command words are explained and an overview of the instruction set is present.

THE 8085A ARCHITECTURE.

The 8085A is an 8-bit general-purpose microprocessor that is ideally suited to many applications. In this section we introduce the internal architecture of the 8085A microprocessor.

THE 8085A BLOCK DIAGRAM.

Although a detailed understanding of the internal operation of the 8085A is not required for programming, it does help to explain why some of the instructions operate in a particular fashion.

The 8085A contains a register array, a timing and control section, an arithmetic and logic unit (ALU), an instruction register and decoder, and bus connections to the outside world. Notice that there are 16 address bus connections, A8-15. and a shared (multiplexed) address/data bus containing the least significant 8 bits of the memory address, AO-7 and the data bus DO-7. A 16-bit memory address allows the 8085A to directly address 64K different memory locations. The address/data bus also allows the 8085A to access 8 bits of data at a time, which means that each memory location or I/O device attached to the address/data bus contains 8 bits of data.

The registers. The 8085A contains a variety of internal registers that are used to hold temporary data, memory addresses, instructions, and information about the status of the 8085A.

Instruction register. This register is used to hold the instruction that the 8085A is currently executing. Its outputs are connected to the instruction decoder, which decodes the instruction and controls the rest of the processor, memory, and I/O through the timing and control block and external pins to the outside world.

Temporary register. The temporary register is used to hold information from the memory or the register array for the ALU. The other input to the ALU comes from the accumulator. The result, available at the output of the ALU, is fed to the internal 8-bit data bus for distribution to the accumulator, register array, or memory.

Incrementer / decrementer address latch. This block is used to hold the address of data to be accessed in the memory or 1/0. It is also used to add one or subtract one from any of the other registers in the register array.

Miscellaneous sections. The interrupt control section is used to determine the priority of the interrupt control inputs and also to supply, in some cases, an interrupt instruction to the instruction register. An interrupt is an operation where the external circuitry interrupts a program. When the program is interrupted, the microprocessor executes another program that responds to the external interrupt.

The serial I/O control section is used to control the two serial I/O data pins: SID (serial input data) for reading a bit of external data and SOD (serial output data) for writing a bit of external data.

The timing and control section provides the basic control bus signals RD, WR, and IO/M. In addition to these basic control signals, other signals are supplied to control the external hardware and also accepted to control some internal functions.

THE PROGRAMMING MODEL.

Before an instruction can be explained or a program written, the structure of the internal register set must be fully understood. This section details the programming model-the set of available registers that can be affected by a program.

GENERAL-PURPOSE REGISTERS.

The 8085A contains a set of six general-purpose registers: B, C, D, E, H, and L. These registers are called general purpose because they can be used in any manner by the person programming the microprocessor. The general-purpose registers can be used to hold numeric data, BCD data, ASCII data, and in fact, any type of information that may be required. They are flexible enough so that they can be used as six 8-bit registers or as three 16-bit register pairs.The valid register pairs are Be, DE, and HL. Registers pairs can be used to hold 16-bit numeric data or any other 16-bit ceded information. In addition to being able to hold 16 bits of data, register pairs can also be used to address data in memory. If a memory address is placed into a register pair, certain instructions allow the contents of the location addressed by the register pair to be manipulated.
The special-purpose registers-A, F, SP, and PC- are used for accumulating results from arithmetic and logic instructions and also for housekeeping. The term housekeeping is used to refer to tasks that are required but normally occur without the intervention of the programmer.

THE ACCUMULATOR REGISTER (A).

The accumulator is a very important register in the 8085A microprocessor because it is used to accumulate the answer after almost every arithmetic and all logic operations. You might call the A register the answer register because the answer is normally found here.

THE FLAG REGISTER (F).

The flag register contains 5 bits that are used as flags or indicators for the ALU. Any time the 8085A executes most arithmetic or logic instruction, the flags will change. The results reflected by the flag bits indicate the condition of the outcome of the answer from the ALU. The 5 flag bits include:

1. The sign flag bit (S) is used to indicate whether the result of an arithmetic or logic operation is positive or negative. A logic I in this bit indicates a negative outcome and a logic 0 a positive outcome.
2. The zero flag bit (Z) indicates whether the outcome of an ALU operation is zero or not zero. A logic I in this bit indicates a zero result and a logic 0 indicates a not-zero result.
3. The auxiliary carry flag bit (AC) holds the carry that occurs between the least significant and most significant halves of the result from the ALU. (This flag is normally used only by the DAA command.)
4. The parity flag bit (P) indicates the parity of the result from the ALU. A logic I in this bit indicates even parity and a logic 0 indicates odd parity. (Parity is a count of the number of 1's in a number expressed as even or odd.) 5. The carry flag bit (CY) holds the carry that occurs from the most significant bit of the accumulator after an addition, the borrow after a subtraction, or a logic zero after all logic operations.

THE PROGRAM COUNTER (PC).

The program counter does not count programs. The program counter is used by the 8085A to locate the next instruction to be executed. Why is it called a counter? It is called a counter because it is a counter. It counts up through the memory, allowing the microprocessor to sequentially execute the next instruction from the memory. The importance of this register and a greater discussion of its operation appears in Unit VI.

THE STACK POINTER (SP).

The stack pointer allows the 8085A to track its last- in, first-out (LIFO) stack. The stack in the 8085A processes data so that the first data into the stack are the last data out of the stack. For example, if a 2, a 3, and a 4 are placed on the stack, they come off the stack in reverse order, as a 4, a 3, and a 2. You might think that this is a strange way of storing information. It is, unless subroutine nesting is important.

3-4 MEMORY AND I / O.

As mentioned earlier, the memory is used to store programs and data, and the 1/0 system is used to allow human beings and machines to communicate with the microprocessor. Both areas are very important and it is critical that they are understood before a program is written.

THE 8085A MEMORY MAP

The memory locations are numbered in hexadecimal from 0000 through location FFFF. This means that the capacity of the memory is 64K locations. Each memory location holds one byte of information, which can be an instruction or any form of data.
In addition to the numbered memory locations, there are also some RST locations listed at certain memory locations. RSI' is an acronym for restart. The restart is an instruction that is discussed in Unit VI. (Trap is a special type of restart.) The memory of the 8085A is most often broken into two areas. One area of memory is devoted to the system program and is most often populated with ROMs. The other area of memory is used to store data and also programs, in microprocessor-based training systems, and is usually populated with semiconductor RAMS.
The reset location is a very important memory location in the 8085A memory map. The reset location (0000) is where the 8085A will begin to execute a program after the reset button, or input, of the 8085A is activated. If you are developing the system software for an 8085A, you must start the program at location 0000. If you are using a microprocessor trainer, its system program begins at location 0000. This means that your program must begin in some location in the trainer's RAM. This varies from trainer to trainer, of course. The Intel SDK-85 has its RAM at location 2000 through 20FF and the DeVRY Micromodule-85 has its RAM at location 1400 through 17FF.

THE 8085A I/O MAP

The 8085A can directly address 256 different input and 256 different output devices.

External 1/0 devices are called 1/0 ports in an 8085A-based system. An input port is an external device that passes information to the microprocessor and an output port is an external device that accepts information from the microprocessor. The port number is like a memory address because it is used to address an external I/O device and it is often called the port address.

The I/O ports are numbered from 00 to FF and all ports are available to the user unless a microprocessor-based trainer is in use. A complete discussion of the two 1/0 commands, IN and OUT, is presented in Chapter 4.

3-5 DATA WORD FORMATS .

There are many different data word formats that must be understood before a program can be written. This section covers unsigned integers, signed integers, ASCII-ceded alphanumeric data, BCD-ceded data, binary fractions, and floatingpoint numbers.

THE UNSIGNED INTEGER.

Unsigned integers, in the 8085A microprocessor, are most often either 8 or 16 bits in width, but they could be any multiple of 8 bits in width. The 8-bit unsigned integers are found in the memory system and also in any single register. The 16-bit unsigned integers are found in two contiguous bytes of memory and in register pairs.

8-bit unsigned integer. All 8 bits are used to hold the value of a number with the binary weights. Some examples of 8-bit unsigned integers include 0000 0001_2 (01H) = 1, 1000 0000_2 (80H) - 128JO, 1100 0000_2(COH) - 192JO, and 1111 1111_2 (FFH) = 255JO. The allowable range of 8-bit unsigned integers is 0-25510. (Note: Hexadecimal quantities are always denoted with the letter H following the hexadecimal number. For example, 67H is equal to 67 hexadecimal.)

16-BIT UNSIGNED INTEGER.

When a 16-bit integer is stored in the memory it is always stored with the least

significant 8 bits in the lowest-numbered memory location and with the most significant 8 bits in the next contiguous memory location. The least significant portion is called the low-order part and the most significant portion is called the high-order part.
Some examples of 16-bit unsigned integers include 0000 0000 0010 0000_2 (0020H) = 32_10, 0010 0000 1000 1110_2 (208EH) = 8334JO, and 1111 1010 0000 1011_2 (FAOBH) -64,011_10. The allowable range of 16-bit unsigned integers is 0-65,535. Note: When this text uses binary numbers they are always presented in BCH (binary-ceded hexadecimal). A BCH number is a binary number that is grouped in 4-bit segments: for example, 0000 1100 2 = OCH.

SIGNED INTEGERS.

Single-byte signed integers are 7-bit numbers plus a sign bit. Positive numbers are' stored with a 0 in the sign bit followed by a 7-bit magnitude, and negative numbers are stored with a I in the sign bit followed by a 7-bit two's complement of the magnitude. Positive numbers range in value from 0 to 127 and negative numbers range in value - 1 to - 128. Some examples of positive signed 8-bit numbers are 0000 1000_2 = 8,0111 1111_2 - 127JO and 0101 0000_2 - 80_10. The weight (value) of the sign bit is - 128. If a I appears in this position, it is equal to - 128, and if a 0 appears, it is equal to 0. The remaining bit positions are numbered +64, +32, + 16, + 8, +4, +2, and + 1. If a number such as 1000 0011_2 is converted to decimal, a - 128, +2, and + 1 are added together to arrive at a value of- 125. If a number such as 0001 0101_2 is converted to decimal, a + 16, +4, and + 1 are added together to arrive at +21. Sixteen-bit signed numbers are treated in the same manner as 8-bit signed numbers. The extreme left-hand bit is the sign bit and the remaining bits contain the magnitude for a positive number and the two's complement of the magnitude for a negative number.

ASCII DATA FORMAT.

Refer to Appendix D for a complete listing of ASCII-ceded characters. ASCII is an acronym for American Standard Code for Information Interchange. This code is used by virtually all manufacturers of computer peripheral equipment, and it is therefore an excellent idea to become familiar with it. The first 32 codes are used for control codes and are numbered from OOH to IFH. To obtain any of these codes, on most computer keyboards hold the control key down and type a letter A for ОШ, a letter B for 02H, and so on. The OOH code is most often obtained by holding the control and shift keys down while typing the letter P.

It is important to notice that ASCII code is a 7-bit code. The eighth bit is used to hold the parity bit in a data communications system. In computer systems this bit is often a logic 0. In some printers a 0 in the eighth bit causes the printer to print ASCII characters and an I causes it to print graphics characters.

BCD DATA FORMAT.

In the many systems that do not contain an ASCII keyboard, data are entered on a numeric keyboard and encoded by the keyboard circuitry as a binary-ceded-decimal (BCD) number. This number is usually processed as an unpacked BCD number. An unpacked BCD number is a number that is stored one digit per byte. In other words, to store a 76 in unpacked BCD code would take two bytes of memory. The first byte contains a 0000 0111(7) and the second byte a 0000 0110 (6). In certain cases it is desirable to conserve memory space. When this is the case, BCD numbers are usually packed two BCD codes per byte. A packed BCD number is stored as a two-digit BCD number per byte. To store a 76 in packed BCD code would take one byte of memory (O П1 0110).

BCD CODES FOR THE NUMBERS 0 THROUGH 9.

Signed BCD numbers are not nearly as common as unsigned BCD numbers. If they are used, the negative numbers are stored in ten's-complement form. For example, a - 14 is stored as a 1000 0110 (86), which is the ten's complement of a 14. To form a ten's-complement number, first subtract the number to be converted from a 99 and then add a 1 to the result.

BINARY FRACTIONS.

Binary fractions can also be stored in either byte or two-byte form for use by the 8085A, although they are not very commonly found. Usually, they are unsigned numbers that use the leftmost bit position, which has a weight of 2-1. For example, if the binary number 1100 0000 is found in a memory location and it is known that it is an unsigned fraction, its value is 0.75_10.

FLOATING POINT DATA.

How are numbers that are not integers or fractions (mixed numbers) stored in a computer system's memory? The floating-point format is used to store noninteger as well as integer data in many compu-ter systems. Floating-point numbers are similar to scientific notation in base 10. They have a mantissa and an exponent. The mantissa is a normalized number between I and less than 2. The expo-nent is a power of 2 that rep-resents the position of the binary point in the original number. Floating-point num-bers are often stored in four bytes of memory. The left-hand bit position is used to indicate the sign of the mantissa; the next eight bit positions are used for the exponent, which is stored in excess 127 nota-ion; and the last 23 bits are used to store the magnitude of the mantissa.

The excess 127 notation exponent is an unsigned integer that is equal to the exponent plus 127. For example, if the value of the exponent is determined to be a 6, the number ceded for the excess 127 notation exponent is 133. If the exponent is determined to be a

-2, the excess 127 notation exponent is 125. The mantissa is a 23-bit number with a hidden or implied twenty-fourth bit position. Notice that when a number is normalized, the left-hand bit is always a logic 1. Because this I is always present, we do not need to store it in the memory. (A zero is the only case where an implied I is missing. In this case all four bytes of the number are zero, to indicate a zero number.)

3-6 COMMAND WORD FORMATS .

There are basically three different command word formats used by the 8085A: one-byte-, two-byte-, and three-byte-long commands. The one-byte commands represent 204 instructions; the two-byte commands represent 18 instructions; and the three- byte commands represent 24 instructions. The 8085A has 246 of its 256 possible commands implemented. The remaining instructions are not implemented in all versions of the 8085A. Again, be aware that not all versions of the 8085A will respond to these instructions.

ONE-BYTE COMMANDS.

The one-byte commands, which are the most numerous, are used for most of the commands in a program. They are used for moving numbers from a register to a register, from a register to memory, from memory to a register, and also to accomplish most of the arithmetic and logic operations. The first byte of any command is always the op-code. The op-code tells the 8085A microprocessor which operation to execute.

TWO-BYTE COMMANDS.

The two-byte commands, all 18 of them, are used for immediate data instructions or for the input/output commands, IN and OUT.

THREE-BYTE COMMANDS.

The 18 three-byte commands are used either to specify an opcode and two bytes of immediate data or an op-code and a 16-bit binary address. (The operand is the data used by an opcode). Notice that the data are encoded so that the least significant 8 bits (low-order data) follows the opcode and the most significant 8-bits (high-order data) follow the low order byte. It may take some time to get used to this, because this is different from the way that a number is normally written.

THE INSTRUCTION SET.

The instruction set of the 8085A microprocessor can be broken into three main categories: data transfer instructions, arithmetic and logic instructions, and program control instructions. This section introduces these three forms of instructions.

DATA TRANSFER INSTRUCTIONS.

Data transfer instructions represent a lot of processing time in most software systems. For this reason it is important that these instructions are understood. There are actually five types of data transfer instructions available for use in the 8085A instruction set. These types are: register-to-register moves, move immediates, indexed moves, direct moves, and input/output. (None of these instructions affect the flag register).

ARITHMETIC AND LOGIC INSTRUCTIONS.

Another common operation in a microprocessor-based system is some simple arithmetic or logic operation. The 8085A is capable of a wide assortment of arithmetic and logic operations. Instructions that the 8085A is capable of executing are: addition, subtraction, AND, OR, NOT, exclusive-OR, and various forms of rotation. These operations can be used to address any internal register or the contents of any memory location. They can even be performed using immediate data. (Most of these instructions affect the flag register.)

PROGRAM CONTROL INSTRUCTIONS.

The 8085A microprocessor can execute four different types of program control instructions. These four types are: unconditional and conditional jumps, indirect jumps, unconditional and conditional subroutine calls, and unconditional and conditional subroutine returns. The conditional instructions are used to test the flag bits. If the condition under test is true, a jump occurs to some other instruction in the program, and if the condition is false, the program continues with the next sequential step.

SUMMARY.

1. The 8085A microprocessor is a general-purpose 8-bit microprocessor that is ideally suited to many control applications.
2. The 8085A microprocessor contains a register array, finning and control section, an arithmetic and logic unit (ALU), and bus connections to the outside world.The 8085A microprocessor is capable of addressing 64K bytes of memory through an 8-bit data bus via a 16-bit address bus.
3. The timing and control section generates the major system control signals: RD, WR, and IO/M.
4. The programming model of the 8085A contains two sections: a general-purpose section and a special -purpose section.
The general-purpose section consists of six registers (B, C, D, E, H, and L), which are used as 8-bit registers or in pairs as 16-bit register pairs (Be, DE, and HL). The special-purpose registers include the accumulator (A), the stack pointer (SP), the flag register (F), and the program counter (PC).
5. The accumulator is used to hold an operand before and the result after an arithmetic and logic operation.
6. The stack pointer is used by the 8085A to address the LIFO stack.

7. The flag register is used to hold information about the result of an arithmetic and logic operation. The flag bits are: zero (Z), sign (S), parity (P), carry (CY) and auxiliary carry (AC).

9. The program counter (PC) addresses the next instruction to be executed by the 8085A. It always points to the next instruction in the program.

10. The 8085A is capable of directly addressing 256 different input and 256 different output devices. External I/O devices are often called I/O ports in an 8085A-based system.

11. Data to be used by the 8085A can take many forms: unsigned and signed binary integers, binary fractions, packed and unpacked BCD, ASCII, and floating point.

12. Instructions for the 8085A are one, two, or three bytes in length.

13. Instructions types are: data transfer, arithmetic, and logic or program control.

GLOSSARY.

Accumulator register. The accumulator register is used to hold an operand before an arithmetic or logic operation and the result after the operation.

Address/data bus. A common set of connections that are used to convey half of the address
(A0_7) to the memory of I/O and to transfer the data (D0_7).

ASCII code. ASCII data consist of an alphanumeric code used in many computer systems.

BCD. Binary-decimal code are data stored in groups of 4 bits that represent the decimal numbers 0 through 9.

Exponent. The binary power of 2 in a floating-point number that is used to reference the position of the binary point in the mantissa.

Flag register. A register that holds the condition of the ALU after an arithmetic or logic operation.

Floating-point number. A number that is used to store very large or very small integers, fractions, or mixed numbers.

General-purpose registers. A **set** of registers that can be used in any way that the programmer sees fit.

Housekeeping. Tasks that are normally processed by the microprocessor without intervention from the programmer.

Instruction register. A register that is used to hold the instruction so that the microprocessor can decode it.

IO/M. The IO/M (input/output or memory) signal indicates whether a transfer is via memory or 1/0.

LIFO. A LIFO (last-in, first-out) stack memory is used to store data and return addresses in the memory.

Mantissa. The part of a floating-point number that is equal to I through less than 2.

Op-code. The op-code is the part of an instruction that specifies the operation for the microproces-sor.

Operand. The data used by the op-code.

Parity. A count of the number of 1's in a number expressed as even or odd.

Port, *1/0.* An external I/O device.

Program counter. A register that holds the address of the next instruction to be executed.

Programming model. A model of the internal user-affectable registers in a microprocessor.

RD. The RD (read) signal is used to cause the memory or I/O to read or send data to the microprocessor.

SID. The serial input data pin is used to allow the microprocessor to accept serial data.

Signed integer. A whole number that can be either positive or negative.

SOD. The serial output data pin is used to allow the microprocessor to send serial data.

Special-purpose registers. A group of registers that are used for processor housekeeping.

Stack pointer. A register that is used to address the LIFO stack.

Unsigned integer. A whole number that has no arithmetic sign.

WR. The WR (write) signal is used to enable the memory or I/O to accept data from the microprocessor.

QUESTIONS AND PROBLEMS.

1. The 8085A address bus contains how many bits?
2. The 808 5A can directly address how many different memory locations?
3. The SID and SOD pins on the 8085A microprocessor are used for what purpose?
4. List the three main 8085A control signals.
5. Draw the programming model of the 8085A microprocessor.
6. List at least three types of data normally held in the general-purpose registers.
7. When two general-purpose registers are connected together to form a 16-bit register, what is this new register called?
8. What is the answer register?
9. List each of the flag bits and briefly describe their function.
10. Why is the program counter called a counter?
11. The stack pointer (SP) is used to reference what type of stack memory?
12. Describe the memory of the 8085 A microprocessor.
13. Where would the ROM in an 8085A-based system probably be found?
14. How many different external input devices can be accessed by the 8085 A microprocessor?
15. What is an I/O port?
16.Convert the following decimal numbers to 8-bit unsigned integers: 12, 33, 55,100,155,196, 212.
17. Convert the following decimal numbers to 16-bit unsigned integers: 156, 522,1000, 2009, 10,000.
18. Convert the following 8-bit signed integers to decimal numbers: 1111 1111, 1000 0111,
0110 1000, and 0111 0000.
19. Convert the following decimal numbers to 8-bit signed integers: 12, - 12, 32, -63,

and - 100.

20. Using the ASCII ceding chart in Appendix D, convert your name to ASCII code.

21. How is an ASCII BS (backspace) obtained on many keyboards? (Not by using the backspace button!)

22. Write the following decimal numbers as both packed and unpacked BCD numbers: 12, 3, 10, 99, 13, and 712.

23.Convert the following decimal numbers to four-byte binary floating-point form: 12, -22, 10.5, 0.002, and -4.25.

24. Convert the following three four-byte binary floating-point numbers into decimal numbers:

25. How many different commands are implemented in the 8085A microprocessor's instruction set?

26. What is an op-code?

27. What is an operand?

28. Explain the order of the data in a three-byte command.

29. Convert the following hexadecimal numbers to the correct form for data storage in the second and third bytes of a three-byte-long command: 1234H, ACDCH, 87FFH, 3443H, and 9080H.

30. What three main categories of instructions comprise the 8085A instruction set?

31. Which types of instructions do not affect the flags?

32. What type of instruction will affect the flags?

UNIT IV.

DATA TRANSFER INSTRUCTIONS.

One of the most common types of instructions in the 8085A microprocessor's instruction set is the data transfer instruction. Data transfer instructions come in many forms: register to register, register to memory, memory to register, and stack operations.
In this section we present the various addressing modes available for all instructions and also all the data transfer instructions, including the machine language and assembly language versions.

ADDRESSING MODES.

The 8085A microprocessor uses four different addressing modes for most instructions: direct, register, register indirect, and immediate. Before a particular instruction is examined, it is essential that each of these addressing modes is completely understood. In this section we detail the addressing modes so that the subsequently explained instructions can be completely-understood.

DIRECT ADDRESSING.

Instructions that directly address the memory always include the memory address of the data. This address is stored following the op-code in the program.
Notice that this type of addressing requires that the instruction contains an opcode followed by a 16-bit memory address stored in two additional bytes of memory following the opcode. (All instructions that use direct addressing are three bytes in length.) It is also important to note that the address of the data is stored so that the least

significant byte follows the op-code and the most significant byte follows the least significant byte. This was discussed in Units III for storing 16-bit data in the memory.

REGISTER ADDRESSING.

Register addressing is one of the more common forms of addressing used by the 8085A micropro-cessor. The instruction specifies the register (B, C, D, E, H, L, or A) or the register pair (Be, DE. HL, or SP) used with the instruction. All register addressed instructions are one byte in length. This mode of addressing will be heavily illustrated when the MOV data transfer instructions are discussed later in this chapter.

REGISTER INDIRECT ADDRESSING.

With register indirect addressing, a register pair holds the address of the memory location accessed by the instruction-the memory location is indirectly addressed by the register pair. If the HL register pair is used to indirectly address memory, the letter M is used in place of a register. For example, suppose that the HL pair contains a 1000H and the letter M is used as a register. An instruction using the M will access memory lo-cation 1000H because the HL register pair points to that memory location.
There are also a few instructions that allow memory to be addressed indirectly through the DE and Be register pairs. These are also discussed later in this chapter. The SP (stack pointer) is also used to indirectly address the stack memory through a few commands-the stack operations-which are also discussed later in this chapter.

IMMEDIATE ADDRESSING.

The immediate addressing mode is used when constant data are used in a program. The data are encoded immediately following the op-code in the program. The 8085A microprocessor has two forms of immediate addressing: 8- and 16-bit immediate addressing. The 8-bit form uses the notation d8 in Appendix B and the instruction lists in this and the next two chapters and the 16-bit form uses d!6. The 8-bit immediate instructions are always two bytes in length and the 16-bit immediate data instructions are always three bytes in length.

IMMEDIATE DATA TRANSFER INSTRUCTIONS.

The 8085A microprocessor has two basic forms of the immediate data transfer instruction. One form, MVI (move immediate), is used to transfer an 8-bit number into a register or memory location indirect-ly addressed by the ILL register pair, and the other form, LXI (load immediate), is used to load a pair of registers with a 16-bit number. The term immediate is used to indicate that the data immediately fol-low the op-code in the program.

THE MOVE IMMEDIATE INSTRUCTION (MVI).

This instruction is used to place an 8-bit number into any register or memory location indirectly addressed by the HL register pair. Each MVI instruction is two bytes in length: the first byte contains the op-code and the second byte contains the data-an 8-bit number. A 12H is placed into the B regis-ter in both hexadecimal machine language (06-12) and symbolic assembly language (MVI B, 12H).
Just as a number can be moved into the B register, a number can be moved into an internal 8085A register. The MVI M, d8 instruction is used to store a byte of data in the memory. This instruction uses the HL register pair to refer to (point to) a location in the memory. M always refers to the memo-ry location indirectly addressed by the HL register pair. Suppose that the HL register pair contains a 1000H and the MVI M, 1Ш instruction is executed. This instruction moves the immediate byte of data (11H) into the memory location indirectly addressed by the HL register pair (1000H). In other words, an 11H is stored in memory location 1000H. This instruction is stored in the memory together with the instructions required to place a 1000H in the HL register pair.

THE LOAD IMMEDIATE INSTRUCTION (LXI).

Load immediate is used to load a 16-bit number into any register pair (Be, DE, and HL) or the stack pointer register (SP). Each LXI instruction is three bytes in length: the first byte contains the op-code and the second and third bytes contain the 16 bits of immediate data. The data are always stored with the low-order portion (11) following the op-code and the high-order portion (hh) following the low-order portion. A 10CDH is placed in the HL register pair with a LX1 instruction. Notice how the least significant byte (CD) immediately follows the op-code (21) and the most significant byte (10) follows the least significant byte. Of course, in symbolic assembly language, the number appears as 1OCDH.

DIRECT DATA TRANSFER INSTRUCTIONS .

Direct data transfer instructions are useful if only one byte or word of data is to be transferred to or from the memory. If more than one byte or word is transferred, it is more efficient if the indirectly addressed instructions are chosen for the transfer. The 8085 A has two forms of the direct addressed instruction: the load and store accumulator and the load and store HL register pair instructions. All directly addressed instructions are three bytes in length: the first byte is the op-code and the second and third bytes contain the memory address of the operand.

THE LOAD/STORE ACCUMULATOR INSTRUCTIONS (LDA/STA).

Two instructions that directly address memory are available to load and store the contents of the accumulator. In both cases the op-code is followed by a 16-bit memory address. In the LDA instruc-tion, the address (a!6) is used by the microprocessor to locate the location of the data to be moved into the accumulator from the memory.
The STA instruction uses the address to point to the memory location where a copy of the accumu-lator is to be stored. A number stored in memory location 1OOOH is copied into location 1200H by using a LDA and a STA instruction.

THE LOAD/STORE HL REGISTER PAIR INSTRUCTIONS (LHLD/SHL).

The LHLD and SHLD instructions are similar to the LDA and STA instructions except that instead of transferring the contents of the accumulator to and from the memory, they transfer the contents of the HL register pair. These two commands used to transfer the contents of memory locations 1OOOH and loom into locations 1200H and 1201H. In this example LHLD 1000H copies the contents of loca-tion 1000H into the L register and the contents of location 1001H into the H register. The SHLD instruction then stores the L register at memory location 1200H and the H register at location 1201H. Notice that the data in HL are stored in the standard Intel format: low-order byte at the lowest-number-ed memory location and the high-order byte at the highest-numbered memory location. This form of storage is true anytime that 16-bit data are stored in the memory.

INDIRECT DATA TRANSFER INSTRUCTIONS.

Although the letter M is used to indirectly address memory, there are other ways to indirectly address memory. In this section we will not cover the M operand, which is presented in Section 4-3 and later sections; it will cover the LDAX and STAX instructions, which also indirectly address memory. The LDAX and STAX instructions have different forms. Notice that each instruction is only one byte in length because the address of the data is stored in a register pair rather than with the op-code, as in the direct addressing instructions. To illustrate the operation of the LDAX and STAX instructions. Example will be repeated using the BC and DE register pairs to indirectly address memory location 1000H and 1200H. The register pairs are loaded with the memory addresses and then uses the LDAX B instruction to load the accumulator from memory location 1OOOH and the STAX D instruction to store the accumulator at memory location 1200H.

REGISTER DATA TRANSFER INSTRUCTIONS .

The largest group of data transfer instructions is the register data transfer group. This group contains 63 different instructions called moves (MOV). The rightmost register is called the source register and the leftmost register is called the destination register. The MOV instruction transfers a copy of the data in the source register to the destination register. Note that the destination register changes and the source register does not change.

Some of these instructions use indirect addressing as well as register addressing. Also, not all of the instructions have a useful function-MOV B,B, for example, will actually copy the contents of the B register into the B register, but this does not serve any useful function.

A number in the accumulator is moved into both the D and the E registers. The first instruction copies the contents of the accumulator into the D register and the second copies the accumulator into the E register.

Another example using MOV instructions is to clear the contents of all the internal registers. Here the accumulator is first cleared to zero with a MVI instruction and then moves are used to clear the remaining internal registers.

STACK DATA TRANSFER INSTRUCTIONS .

The Intel 8085A microprocessor has a LIFO (last-in, first- out) stack memory that is used to store both return addresses from subroutines and data temporarily. This section of the text deals with the latter purpose, lists the stack data transfer functions available in the 8085A microprocessor's instruction set. This set of instructions consists of PUSHes, POPS, and an XTHL instruction.

STACK MEMORY OPERATION.

Before the stack data transfer instructions can be covered, it is important that the operation of the LIFO stack be understood. The microprocessor does not know where the stack memory is located when power is first applied to the system; it must be told. This is normally accomplished by first loading the SP (stack pointer) register with an address in the memory. (Loading the stack pointer at the beginning of a program is essential if any of the stack operations are used!) The programmer decides what portion of the read/write memory is to function as a stack and then loads the SP with the top location plus one. The stack pointer always points to the current exit point.

OPERATION OF PUSH AND POP.

The stack in the 8085A memory is a LIFO stack or more descriptively a push down- pop up stack. The name push down-pop up describes how the stack functions. If data are pushed (placed) on the stack, they move into the memory locations SP-1 and SP-2. (Note that data are stored in pairs.) The high-order register is stored first (SP-1), followed by the low-order register (SP-2). The SP is then decremented by two so that the next push occurs below the first.
Suppose that the SP is loaded with 1000H and BC contains a 1234H. A PUSH B instruction places the 12H from B in memory location OFFFH (SP-1) and the 34H from C in memory location OFFEH (SP-2). The SP is then decremented by 2 to OFFEH. If this is followed by a POP PSW (POP A and flags), data from location (SP) OFFEH are moved into the flags and data from (SP + 1) OFFFH are moved into the accumulator. The SP is then incremented by 2 to 1000H.The stack can be used to ex-change the contents of the DE and Be register pairs. First Be is pushed on the stack followed by DE. Because DE was the last information placed on the stack, it is the first to come off the stack. A POP B will remove previous contents of DE from the stack and place it into Be. If this is followed by a POP D, the prior contents of Be are removed from the stack and placed into DE. The contents of the two registers have been swapped. Notice that the SP would be back at its original value, which means that the same area of memory can be reused by the next PUSH-POP sequence. It is also important to note that PUSHes and POPS must occur in pairs: one PUSH, one POP, two PUSHes, two POPS, and so on. If not, the stack pointer will eventually fill or read every memory location in the computer system.

EXCHANGE ML WITH STACK DATA (XTHL).

This instruction exchanges the contents of the HL pair with the most recent data on the stack. For example, if a 1000H is pushed on the stack and HL contains a 2000H, an XTHL command will exchange these two values so that HL equals 1000H and the stack data equal 2000H.

4-8 MISCELLANEOUS DATA TRANSFER INSTRUCTIONS.

In previous sections in this chapter we have explained the operation of most of the data transfer instructions except for the I/O instructions and some of the special instructions.

INPUT/OUTPUT DATA TRANSFER INSTRUCTIONS.

As mentioned in Unit III, a computer system contains memory and I/O in addition to the CPU or MPU. To allow data transfers between the microprocessor and I/O. the 8085A instruction set contains two I/O instructions: IN and OUT. The IN instruction inputs data from an I/O device into the accumu-lator and the OUT instruction sends accumulator data out to an I/O device.
The I/O system for the 8085A microprocessor consists of 256 unique device addresses or port num-bers. A port number is an address for an I/O device, just as a memory address is an address for the memory. The port number is used to select a particular I/O device. Both the IN and the OUT instruc-tions are two bytes in length: the first byte is the op-code and the second is the I/O port number.

LOAD SP FROM HL (SPHL).

The SPHL instruction is a one-byte instruction that copies the contents of the HL register pair into the stack pointer (SP). This instruction is used in some systems to initialize the stack pointer, and in other systems the LXI SP, d!6 instruction is used to load the stack pointer. In any case it is probably one of the least used 8085A instructions.

EXCHANGE DE WITH ML (XCHG).

The XCHG instruction exchanges the contents of the HL register pair with the contents of the DE register pair. Suppose that this instruction is used in a program to move a number from memory location 1000H into 1200H. This task is accomplished by pointing to location 1200H with DE and location 1000H with HL. The MOV commands and an XCHG can then be used to transfer the data.

SUMMARY.

1. Data transfer instructions are used to transfer information from register to register, from register to memory, from memory to register, to and from the stack, and to and from the ,I/O devices in a sys-tem.
2. Four different addressing modes are used in the Intel 8085Л microprocessor: direct, register, register indirect, and immediate.
3. Direct addressing is used whenever a memory location is accessed by storing the address of the memory location with the instruction.
4. Register addressing is used to address either a single 8-bit register (B, C, D, E, H, L, or A) or a 16-bit register pair (Be,DE, HL, and SP).
5. Register indirect addressing allows the instruction to address memory through the address held in a register pair.
6. Immediate addressing is used whenever the data (8 or 16 bits) are a constant. Immediate data immediately follow the op-code in the program.
7. MVI and LXI are the two immediate data transfer instructions in the 8085A instruction set. MVI is an 8-bit immediate instruction and LXI is a 16-bit.
8. The M register or operand is used to indirectly address memory through the HL register pair.

9. Only the accumulator and the HL register pair can be directly stored in the memory. LDA and STA are used for accumulator direct storage, and LHLD and SHLD are used for the HL register pair.

10. In addition to M for indirectly addressing the memory, the DE and Be register pairs are also avail-able. LDAX and STAX allow the accumulator to be indirectly stored or loaded from the memory using the Be or DE register pairs.

11. Register data transfer instructions are the most numerous form of data transfer-63 instructions.

12. The stack memory in the 8085A microprocessor is a LIFO (last-in, first-out) memory used to store data and return addresses from subroutines.

13. The stack pointer (SP) register is used to indirectly address the stack for the stack data transfer instructions PUSH, POP, and XTHL.

14. The PSW is the processor status word which contains both the accumulator, as the high-order register, and the flag byte, as the low-order register.

15. IN and OUT are used to effect data transfer to and from the external I/O devices, often called I/O ports.

GLOSSARY.

Destination register. The register that receives a copy of the data in an instruction.

Direct addressing. If memory data are directly addressed, the memory address is stored with the instruction in a program. The address is stored so that the second and third bytes of the instruction contain the memory location of the operand.

Immediate addressing. An immediate instruction contains the data used with the instruction in the form of the byte (8-bit data) or bytes (16-bit data) of data immediately following the op-code in the program.

Immediate data. 8- or 16-bit data that immediately follow the op-code.

M register. The M register is the memory location indirectly addressed by the HL register pair. Port An I/O device in the 8085A system is called an I/O port.

Port address. An 8-bit number used to address a unique external I/O device.

Register addressing. A register-addressed instruction specifies the register or register pair where the data are located.

Register indirect addressing. Register indirect addressing is used to address memory through a register pair. The register pair holds the address of the memory data.

Source register. The register that supplies the data in an instruction. This register is never changed by an instruction.

QUESTIONS AND PROBLEMS.

1. What four addressing modes are available for use in the 8085A microprocessor instruction set?
2. Direct addressed instructions are______________bytes in length.
3. Where is the memory address of a direct-addressed instruction located?
4. Convert the following 16-bit memory addresses into the form required when stored with a direct addressed instruction: moon, 234AH, ABCDH, 5000H, and 456FH.
5. What 8-bit registers are available for use with a register addressed instruction? What register pairs are available?
6. When using register indirect addressing, to what does theletter M refer?
7. Which immediate instruction is two bytes in length, and why?
8. Convert the following symbolic assembly language instructions into hexadecimal machine language instructions: LXI D, 1200H, MVI C, 90H, LXI SP, 1234H, MVI M, 10, and MVIM, 10H.
9. Write a sequence of immediate instructions that will place a 0000 into BC and a 12H into the accumulator.
10. Write a sequence of immediate instructions that will store a 16H in memory location 1200H and a 17H in memory location 1202H.
11. Explain how the LDA 1000H instruction functions.
12. Explain what answer is found in memory location 1200H and 1201H in the following sequence of instructions. 26-22 MVI H, 22H 2E-44 MVI L, 44H 22-00-12 SHLD 1200H

13. Which register indirect instruction is used to store the contents of the accumulator into the memory location indirectly addressed by the BC register pair?

14. Explain what answer is found in memory location 1200H in the following sequence of instructions.

```
06-12                  MVIB,    12H
OE-00              MVI    C,    OOH
3E-77              MVI    A,    77H
12       STAXB
```

15. Write a sequence of instructions that will use register indirect addressing to transfer the number stored in memory location 1300H into memory location 1301H.

16. Why is it rare to find the MOV B, B instruction in use in a program?

17. Explain what the MOV M, C instruction does if HL =1233H and C = 34H.

18. Write a sequence of instruction that use MOV instructions to swap the contents of the BC register pair with the DE register pair.

19. Write a sequence of instructions that will store a zero in memory location 1000H through 1003H.

20. If a 1000H is pushed on the stack followed by a 2000H, which number is the first to come off the stack?

21. The push instruction is used to place the contents of any on the stack.

22. What is the PSW?

23. What number appears in the BC register pair after the following sequence of instructions?

```
21-00-30                  LXIH.3000H
11-00-20                  LXID.2500H
E5        PUSH H
D5        PUSH D
El        POP H
Cl        POP B
```

24. If a PUSH PSW is immediately followed by a POP B, in which register do the flag

data appear?

25. Explain what the OUT 12H instruction accomplishes.
26. Which two instructions from this chapter can be used to place a number into the SP?

UNIT V

ARITHMETIC AND LOGIC INSTRUCTIONS .

The 8085A arithmetic and logic instructions described in this chapter include the following operations: addition, addition with carry, subtraction, subtraction with borrow, inversion, AND, OR, exclusive-OR, and rotation. In addition to the arithmetic and logic operations, in this chapter we present a detail-ed view of the operation of the flag bits with each instruction. None of the instructions described in Unit V affected the flags, whereas all the instructions in this chapter affect the flags.

ADDITION.

Addition takes several forms in the 8085A microprocessor: 8-bit binary. 16-bit binary, and two-digit binary-ceded-decimal (BCD) addition. In binary addition either signed or unsigned numbers are added, and in BCD addition only unsigned numbers are added. The instruction set supports additions using register addressing, register indirect addressing, and immediate addressing, but not direct addressing.

8-BIT BINARY ADDITION.

Because the 8085 A is an 8-bit microprocessor, most of the addition instructions are 8-bit additions. All the addressing modes are represented except for direct addressing. It is also critically important to recognize that the flag bits will always be affected by any of these instructions. Any addition instruc-
tion will always add the operand to the accumulator. After the 8085A adds the 12H and

the 55H toge-ther, the result (67H) is placed into the accumulator. The flags change as follows:

Z = 0	The result is not zero.
P = 0	The parity of the result is odd.
CY = 0	No carry occurred for this addition.
AC = 0 •	No half-carry occurred.
$ = Q	The result is positive.

Suppose that it is desirable to add the number in the B register to the number in the A register. This is accomplished by using the ADD B instruction, which adds the contents of the B register to the con-tents of the A register and places the sum in the A register.

In the example a 40H and an EEH are added together, resulting in a sum of 2EH. The carry out of the accumulator is held in the carry flag. The flags are changed to the following conditions:

Z = 0	The result is not zero.
P = 1	The parity of the result is even.
CY = 1	A carry occurred for this addition.
AC = 0	No half-carry occurred.
S - 0	The answer is positive.

Suppose that it is desirable to add the number in the B register to the number in the C register and place the sum in the D register. This addition is accomplished by using the ADD B instruction, which adds the contents of the B register to the fcontents of the A register. Before the addition takes place, the number in C must be moved to A," and after the addition,' the answer, in A, must be moved into D. A 2FH is added to an 8 to generate a result of 37H. Notice the extra instructions required to posi-tion C before the addition and also position the result D after the addition. The flags change as indica-ted:

Z - 0 The result is not zero. P = 0 The parity of the result is odd.

CY = 0 No carry occurred for this addition. AC - 1 A half-carry occurred. S - 0 The answer is positive.

ADDITION WITH CARRY.

Whenever large numbers, numbers with more than 8 bits, or multiple-byte numbers are added together, the carry must be propagated from one 8-bit segment to the next. To accomplish a carry in multiple-byte addition, the add with carry instruction is used to propagate the carry from one byte to the next. Suppose that the DE register pair contains a 16-bit number that is to be added to the number in the Be pair. To accomplish this multiple-byte addition, it is necessary to add E and C together and then add D and B together with the carry from the addition of E and C. Notice that the ADD E instruction generates a carry which is added into the most significant byte of the answer with the ADC D instruction. The answer, which is found in the Be register pair, is 30FH.

BCD ADDITION.

BCD addition is like binary addition except that the numbers that are added together can only range in value from 0 through 9. In the 8085A a special instruction is provided that allows BCD addition to be accomplished by using the binary addition instructions. The DAA instruction is used after a BCD addition (with a binary add instruction) in order to correct the BCD result. After the addition, the accumulator contains a 2AH, which is not a valid BCD number-the answer should be a 30bcd. DAA corrects the answer and provides a 3OH after the instruction is executed. (Note that 3OH and 30bcd are exactly the same when ceded in the memory.) The DAA instruction changes the result through the two tests listed by adding a OOH, 06H, 60H, or 66H to the accumulator.

1. If the least significant half-byte is greater than 9 or if the AC flag bit is set, a 06H is added to the accumulator.
2. If the most significant half-byte is greater than 9 or if the CY flag bit is set, a 60H is added to the accumulator.

INCREMENT.

The last form of addition available is to increment or add 1. The increment command is either an 8-bit (INR) increment or a 16-bit (INX) increment instruction. The INR instructions affect all the flags except CY, and the INX instructions affect no flags. Although there is no example of these instruc-tions at this point in the text, the increment instruction is extremely useful, as we shall discover in later chapters on programming.

SUBTRACTION.

The 8085A supports 8-bit binary subtraction and decrement. It also supports a subtraction instruction which allows a borrow, if it occurs, to be propagated through additional bytes of a number. This instruction-subtract with borrow-is often used for multiple-byte subtraction. If 16-bit or BCD subtrac-tion is required, a program must be written to accomplish it.

8-BIT SUBTRACTION.

All the various subtraction instructions, includes register, register indirect, and immediate addressing, but not direct addressing. Each of these instructions affects the flag bits so that they reflect various conditions about the difference after a subtraction. A simple sequence of instructions that find the difference between 2EH and 3FH. The accumulator equals an EFH after the subtraction and the flags change as indicated:
Z = 0 The result is not zero.
P = 1 The parity of the result is even. CY =1 A borrow occurred for
this subtraction. AC = 1 A half-borrow occurred. S = 1 The answer is
negative.
Notice that the borrow is held in the carry flag and the half-borrow is held in the AC flag after a subtraction. Borrows that occur for 8-bit subtraction are most often ignored, but borrows that occur for multiple-byte subtractions are cascaded through the more significant bytes of the difference.

SUBTRACT WITH BORROW.

Whenever multiple-byte numbers are subtracted, the borrow must be propagated from one 8-bit seg-ment into another. To develop a program that propagates the borrow, a new instruction is required.
Suppose that the number in the DE pair is subtracted from the Be pair. Just as with addition, the least significant byte is operated on first. Once the difference of C and E is determined, the D register is subtracted from the B register with a borrow. This effectively propagates the borrow through the most significant byte of the result.
In this example the 01H in E is subtracted from the OOH in C. This generates a result in C of FFH and also a borrow, which is held in the CY flag. When the OOH in D is subtracted from the 01H in B with borrow, the result is OOH. The borrow is subtracted from B as well as the OOH in the D register. The difference after this sequence is a OOFFH, and it is found in the BC register pair.

DECREMENT.

The final form of subtraction available in the 8085A is decrement or subtract I. The decrement com-mand is either an 8-bit (OCR) decrement or a 16-bit (DCX) decrement instruction. The OCR instruc-tions affect all the flags except CY and the DCX instructions affect no flags. Although there is no example of these instructions at this point in the text, the decrement instruction, as well as the incre-ment instruction, are extremely useful, as we shall discover in later chapters on programming.

COMPARE.

The compare instruction is a modified subtraction instruction. It performs 8-bit binary subtraction with one unique modifica-tion-me difference is not routed into the accumulator. This instruction changes only the flag bits so that they reflect the dif-ference. You might wonder why. Suppose that you were required to determine if the number in the accumulator is a 12H. You could subtract 12H from the accumulator and look at the zero flag bit. The problem with this is that the original number in the accumulator is destroyed (lost). The compare allows this comparison without the loss of the number in the accumulator. The compare instructions are available with the same addressing modes as the subtract instructions, is learned in later chapters, this instruction will become indispensable.

LOGIC INSTRUCTIONS.

The 8085A microprocessor is capable of executing four basic logic functions: invert, AND, OR, and exclusive-OR. Why does a microprocessor instruction set contain logic instructions? One reason is that logic instructions are sometimes used to replace discrete logic gates. Today, program storage costs about '/2 of a cent per byte. If an instruction can be used to replace an external logic circuit, imagine the amount of money saved by programmed logic! Another reason is that system control software usually requires bit manipulation-a logic operation.

INVERSION.

The CMA instruction, 2FH in machine language, is used to one's complement or invert the contents of the accumulator. This operation, which affects no flag bits, causes each bit of the accumulator to be inverted (changed from I to 0 or 0 to 1). CMA causes the accumulator to appear as eight inverters. This means that this one- byte instruction can be used to replace eight discrete inverters provided that the speed is not too great. The amount of circuitry replaced by the CMA instruction is 1 1/3 of 7404 TTL hex inverter. Cost advantage: software = $0.0005 versus hardware = $0.40, a saving of 80,000 percent. In addition to inverting the accumulator (often called NOT), this instruction together with an INR A is used to form the two's complement of the accumulator. Whenever a number is two's complemented, its arithmetic sign is changed.

THE AND OPERATION.

The AND instruction, whose operation is represented as a^A, actually has two separate functions in a microprocessor-based system: selectively clearing bits of the accumulator and replacing discrete AND gates.

The binary multiplication, or AND instruction, functions as eight independent two-input AND gates, as illustrated in Fig. 5-3. This instruction is used to replace two 7408 quad two-input AND gates. The cost advantage: software = $0.0005versus hardware = $0.80, a saving of 160,000 percent. The effect on the flag bits is as follows: Z, S, and P are set or reset depending on the outcome of the AND operation, C Y is always cleared, and AC is always set.

In addition to replacing external logic circuitry, the AND function can also be used to selectively clear (mask) any number of bit positions in the accumulator. This command will turn off individual bits in the accumulator. The 8 bits of the accumulator are available to use as eight switches for controlling external hardware.

This effect by using x 's (don't cares) to represent each bit of the accumulator and 00001111 as a test bit pattern ANDed with the accumulator. The outcome clearly indicates that the O's in the test bit pat-tern force the corresponding bit positions to 0 and the 1's in the test pattern allow the correspond-ing bit positions to pass through to the result unchanged. The ANA A instruction has a special func-tion. With ANA A, the accumulator is ANDed with the accumulator and as a result the value of the accumu-lator does not change. This instruction does change the flags, so that a number in the accumulator can be tested for a zero-not zero, positive-negative, or an even-odd parity condition. For this reason the ANA A instruction should be thought of as a TEST A instruction.

THE OR OPERATION.

The inclusive-OR instruction, whose operation is represented as a V, actually has two separate func-tions in a microprocessor-based system: selectively setting bits of the accumulator and replacing dis-crete OR gates.

The binary addition or inclusive-OR instruction functions as eight independent two input OR gates.

This instruction is used to replace two 7432 quad two-input OR gates. The cost advantage: software = \$0.0005 versus hardware = 0.80, a saving of 160,000 percent. The effect on the flag bits is as fol-lows: Z. S, and P are set or reset depending on the outcome of the OR operation; CY and AC are always cleared.

In addition to replacing external logic circuitry, the OR function can also be used to selectively set any number of bit positions in the accumulator. This command will turn on individual bits in the accumu-lator. The 8 bits of the accumulator are available to use as eight switches for controlling external hard-ware. The AND operation is used to turn bits off, and the OR operation is used to turn bits on. The effect by using X 's (don't cares) to represent each bit of the accumulator and 00001111 as a test bit pattern ORed with the accumulator. The outcome clearly indicates that the 1's in the test bit pattern force the corresponding bit positions to 1. and the O's in the test pattern allow the corresponding bit positions to pass through to the result unchanged. The ORA A instruction, as the ANA A instruction, has a special function. With the ORA A instruction, the accumulator is ORed with the accumulator and the value of the result does not change. This instruction does change the flags so that a number in the accumulator can be tested for a zero-not zero, positive-negative or even-odd parity condition. For this reason, the ORA A and the ANA A instructions should both be thought of as TEST A instructions.

THE EXCLUSIVE-OR OPERATION.

The exclusive-OR instruction, whose operation is represented as a V, actually has two separate func-tions in a microprocessor-based system: selectively inverting bits of the accumulator and replacing discrete exclusive-OR gates. The exclusive-OR instruction functions as eight independent two-input exclusive- OR gates. This instruction is used to replace two 7486 quad two-input exclusive-OR gates. The cost advantage: software =

$0.0005 versus hardware = $0.80, a saving of 160,000 percent. The effect on the flag bits is as follows: Z, S, and P are set or reset depending on the outcome of the exclusive-OR operation; CY and AC are always cleared.
In addition to replacing external logic circuitry, the exclusive-OR function can also be used to selectively invert any number of the bit positions in the accumulator. This command will, in other words, complement individual bits in the accumulator. The 8 bits of the accumulator are available to use as eight switches for controlling external hardware. The AND operation is used to turn bits off, the OR operation is used to turn bits on, and the exclusive-OR operational inverts bits. The effect by using x's (don't cares) represents each bit of the accumulator and 00001111 as a test bit pattern exclusive-ORed with the accumulator. The outcome clearly indicates that the 1's in the test bit pattern force the corresponding bit positions to invert, and the O's in the test pattern allow the corresponding bit positions to pass through to the result unchanged. An added bonus is the XRA A instruction, which always clears the accumulator to zero. This command should be thought of as a CLEAR A instruction.

SHIFT AND ROTATE INSTRUCTIONS .

In certain applications it is desirable that information be shifted or rotated. The 8085A microprocessor is capable of all types of logical rotation and certain forms of shifting.

THE ROTATE INSTRUCTIONS.

There are four instructions that allow the contents of the accumulator to be rotated left or right. Each rotate instruction affects the contents of the accumulator and the CY flag bit. The operation of the four rotate instructions are: RRC and RLC, 8-bit rotates, and RAR and RAL, 9-bit rotates. Suppose that the accumulator contains two 4-bit BCD digits and software is required to reposition them so that the most significant digit and test significant digit exchange places. The RLC instruction is also able to accomplish the same result by rotating the number left instead of right.

THE SHIFT INSTRUCTIONS.

Although there are no shift instructions in the instruction set, some of the instructions described earlier in this chapter are used for shifting. The ADD A instruction will shift the accumulator left one bit posi-tion, and the DAD H instruction will shift the HL register pair left one bit position. The shift-right ope-ration must be synthesized from the rotate instructions. Two right shifts are normally used in practice: logic and arithmetic. The logical shift right requires that a zero be placed in the leftmost bit position, and the arithmetic shift right requires that the sign bit be copied through the number.

SUMMARY.

1. The arithmetic and logic instructions affect the flag bits, whereas the data transfer instructions of Unit IV did not affect the flags.
2. Most of the arithmetic and logic instructions gate the result into the accumulator.
3. Most arithmetic and logic instructions use register, immediate, and register indirect addressing.
4. Addition is available as add I to any register or register pair, 8- and 16-bit binary, 8-bit binary with carry, and binary-ceded decimal (BCD).
5. Subtraction is available as subtract I from any register or register pair, 8-bit binary, 8-bit binary with borrow, and as a compare, which is a form of subtraction.
6. The logic operations are AND, OR, exclusive-OR, and invert.
7. The logic instructions are ideal for control because AND is used to clear bits, OR is used to set bits, and exclusive-OR is used to complement bits. This gives the programmer complete control over each bit of a number.
8. Programmed logic is used to replace discrete logic circuits at a tremendous cost advantage.
9. Some of the 8085A instructions have hidden functions: ADD A is used as a shift the accumulator left, DAD H is used as a shift HL left, ANA A and ORA A are used to test

the ac cumulator, and SUB A and XRA A are used to clear the accumulator.

10.The rotate commands are used to create the shift-right functions: logical and arithmetic shift right.

GLOSSARY.

Accumulator adjust. The act of correcting the result of a BCD addition. The result of each BCD digit is corrected by adding a 6 if a carry occurred or the result exceeded 9.

AND. The logical multiplication operation (AND) is: 0 * 0 =0, 0* 1 = 0, I* 0 = 0, and 1* 1 = 1.

Arithmetic right shift. Whenever a number is shifted to the right, the sign bit is copied through the number for an arithmetic right shift.

Borrow. A borrow out of the most significant bit of the result after a subtraction is held in the carry flag. CY = I for a borrow and CY = 0 for no borrow.

Compare. A special form of subtraction where the value of the difference is lost but the flag bits reflect the difference.

Decrement. To decrement is to subtract I from a register, register pair, or memory location.

Don't cares. A don't care (x) is a bit that contains either a 1 or aO.

Exclusive-OR. The exclusive-OR operation is: 0 $ 0 = O, O Φ 1 = 1, 100 = 1, andiei =0.

Increment. To increment is to add I to a register, register pair, or memory location.

Inversion. Whenever a bit is inverted, it is changed from a I to a 0 or from a 0 to a 1.

Logical shift. When a number is shifted, a zero is moved into the left- or rightmost bit for a logical shift.

Mask. The act of removing part of a number, usually through the AND instruction. The portion that is removed becomes 0.

Multiple-byte number. A number that is more than one byte in width. OR The logical addition operation (OR) is: 0 + 0 - O, O + 1 = 1, 1+0=1, and 1 + 1-1.

Programmed logic. Software that is used to replace the function of hardware circuitry.

Rotate. Whenever a number is shifted to the right or left, a bit drops off the end of a register. If this bit is recirculated through the register, the number is said to be rotated.

QUESTIONS AND PROBLEMS.

1. What forms of addressing are used with most of the arithmetic and logic instructions?
2. List all the flag bits and indicate their contents after the following additions: 12H + 33H, FOH + 33H, OFH + 40H, and 3FH+ ABH.
3. Develop a sequence of instructions in both machine and assembly language that will add a 55H to the number in the B register.
4. Develop a sequence of instructions in both machine and assembly language that will add the number in the H register to the number in the L register.
5. Where would the add-with-carry instruction find most of its application?
6. Develop the sequence of instructions, in both machine and assembly language, that will add the number in the DE register pair to the number in the HL register pair. (You may not use DADD.)
7. Explain what the DAD D instruction accomplishes.
8. The DAD instructions affect which flag bits?
9. The 8085A has a special command for BCD addition. Does this command precede or follow the BCD addition?
10. Develop a program in both machine and assembly language that will add the BCD number in the B register to the BCD number in the L register.
11. What instruction is used to add a I to the HL register pair? Which flag bits are affected by this instruction?
12. List all the flag bits and indicate their contents after the following subtractions: 12H - 33H, FOH - 33H, OFH - 40H, and 3FH - ABH.
13. Where is the borrow found after a subtraction, and what does it indicate?
14. What instruction is used to subtract the contents of the memory location pointed to by the HL pair from the accumulator?

15. Develop a sequence of instructions, in both machine and assembly language, that will subtract the number in the D register from the number in the E register.
16. Develop a sequence of instructions, in both machine and assembly language, that will subtract the number in the DE register pair from the number in the HL register pair.
17. True or false: If the OCR B instruction is executed, all the flags will change.
18. What is the main difference between a compare and a subtract instruction?
19. Why is the compare instruction useful?
20. What 8085A instruction is used to invert the contents of the accumulator?
21. Write a sequence of instructions in both machine and assembly language that will one's-complement the contents of the DE register pair.
22. The AND operation is used to __________ bits in the accumulator.
23. The OR operation is used to ___________ bits in the accumulator.
24. The exclusive-OR operation is used to ________ bits in the accumulator.
25. If NAND is NOT AND, is it possible to replace a NAND gate with programmed logic?
26. What special function is provided by the ANA A instruction?
27. Explain the difference between an RRC instruction and an RAR instruction.
28. Why can the RLC instruction be used in place of the RRC instruction of Example 10?
29. What is the difference between an arithmetic and a logic shift right?
30. Develop a sequence of instructions in both machine and assembly language that will shift the number in the Be register pair to the left one bit position.

Культурная осведомлённость

Термин, который используется для обозначения чувствительности к влиянию культурного поведения на использование языка и коммуникацию. «Культурная осведомлённость» охватывает образ жизни в Англии или США, убеждения, обычаи, привычки, культурные ценности, а также ежедневное отношение и чувства, выраженные не только языком, но и паралингвистическими элементами, такими как одежда, жесты, выражение лица, движения. Термин «Культурная осведомлённость» охватывает, по мнению авторов, три основных качества, свойства которых направлены на развитие, а именно:

- осведомлённость собственного поведения, обусловленного культурой.
- осведомлённость, обусловленного культурой, поведения других людей.
- способность объяснить собственную культурную точку зрения.

Хотя межкультурный обмен - одна из самых быстрорастущих областей изучения языка, систематическое изучение межкультурного взаимодействия может быть новым для многих педагогов. Для этого необходимо использовать ряд подготовленных вопросов, которые чаще всего задают учителя. Почему важно изучать межкультурное взаимодействие?

За последние годы ряд факторов, как лингвистических, так и социально-экономических, подняли изучение межкультурного взаимодействия до высокого международного уровня.

1. **Увеличение экономической значимости стран Юго-Восточной Азии.**

Такие страны как Япония, Корея, Малайзия, Тайвань и Таиланд имеют очень разные традиции и культурное поведение, сильно отличающиеся от таковых в Северной Америке. С увеличением числа студентов, изучающих английский язык за рубежом, стало необходимо переоценить содержание обучения для того, чтобы принять во внимание необходимость использования и объяснения различий в культуре более подробно.

2. **Влияние возрастающей иммиграции на учебные планы ВУЗов.**

Учителя английского как второго или иностранного языка в англоязычных странах давно признали необходимость обучать иммигрантов образу жизни своей страны. Тем не менее, за последние годы, более открытое признание в необходимости понимать образ жизни общества иммигрантов привело к более критичной осведомлённости в области культуры страны, изучаемого языка.

3. Изучение практики.

Исследования лингвистики в области практики (способы, в которых социальный контекст влияет на использование языка) повысили степень осведомленности, на которую влияют факторы, связанные с культурой межнациональных коммуникаций. Такие факторы включают ожидания людей обращать пристальное внимание на должный уровень формальности и степени вежливости речи.

4. Изучение невербальных аспектов коммуникации.

Очень важной была работа по невербальным аспектам коммуникации, таким как жесты, позы, выражения лица. Исследования выявили, что эти невербальные элементы являются частью поведения наиболее влияющей на культуру.

Какой культуре мы обучаем

Изучение жизни и институтов в Великобритании, США и Канаде явилось традиционной частью школьного курса в Европе и Северной Америке. Иногда это принимало форму специальных курсов, таких как Civilization во Франции, Landeskunde в Германии и Civilita в Италии. Эти курсы делали акцент на элементах британских и американских Культур с большой буквы «К»- истории, географии, институтах, литературе, искусстве, музыке и образе жизни

Мы должны признать, что сам предмет расширялся в результате вышеописанных влияний.

«Большая Культура» (культура достижений) осталась той же, но вот культура с маленькой « к » (культура поведения) расширилась, включая убеждения и восприятия в зависимости от культуры, которые выражены как с помощью языка, но также через культурное поведение, которое влияет на восприятие в обществе страны изучаемого языка.

Гэйл Робинсон, американский исследователь в области межкультурного образования, в 1985 году сообщил, что когда учителей спрашивают: «Что значит для вас культура?», чаще всего они называли элементы трёх взаимосвязанных категорий: продукты, мысли, поведение. Расширение культуры с маленькой буквы « к » (культуры поведения) может быть выражено следующим образом:

Элементы культуры (артефакты)

	Продукты	
	литература	поведение
идеи	фольклор	привычки
убеждения	искусство	обычаи
ценности	музыка	одежда
		еда
		досуг

«Культура» с большой буквы извлекла пользу из точно отождествлённого курса обучения по предметам, которые должны быть освещены, а также из учебников, которые имеют отношение к ним.

К влияющим на культуру типам поведения, которые составляют «культуру» с маленькой буквы стали относиться как к чему-то малозначимому, второстепенному, в зависимости от интереса и осведомлённости преподавателей и студентов.

По мнению ряда американских исследователей в области лингвистики, изучение типов поведения влияющих на культуру должно являться результатом изучения языкового материала, но в то же время должно быть ясно определено и то, с чем необходимо систематически иметь дело как с постоянной неотъемлемой частью уроков иностранного языка.

Очень трудно подробно определить программу курса лекций по изучению поведения влияющего на культуру, хотя пересмотренные советом Европы Waystage 90 и Threshold 90. Спецификации для английского языка включают раздел по социально–экономическому аспекту. Нед Силай в своей книге преподавания культуры определил благоприятные условия развития навыков межкультурной коммерции.

Модификацией его «7 задач обучения культуре» являются следующие задачи:

1. Помочь студентам развить понимание того факта, что все люди проявляют тот или иной поведения, обусловленный культурой.
2. Помочь студентам в развитии понимания того, что социальные отличия, такие как возраст, пол, социальный класс и место жительства, влияют на то, как люди говорят или ведут себя.
3. Помочь студентам стать более осведомлёнными в поведении носителей языка в типичных ситуациях.
4. Помочь студентам расширить их осведомленность в культурной аннотации слов и фраз в иностранных языках.
5. Помочь студентам развить способность к оцениванию и усовершенствовать обобщение о культуре той страны. (в терминах сопутствующих данным.)

6. Помочь студентам развить необходимые навыки для того, чтобы определить и организовать информацию об иностранной культуре.
7. Стимулировать интеллектуальное любопытство студентов к культуре и воодушевить сопереживание по отношению к людям этой культуры и языка.

Выше указанные цели можно использовать при составлении рабочих программ и планов и объединить их в следующие практические принципы обучения:

1. Достижения культуры посредством обучения языку.
2. Сделать изучения типов поведения данной культуры неотъемлемой частью каждого урока.
3. Поставить перед студентами задачу достичь социально–экономической компетенции, которая, по их мнению, им необходима.
4. Поставить для всех уровней цель достичь межкультурного понимания – осведомлённости в своей собственной культуре, а так же в культуре изучаемого иностранного языка.
5. Признать, что не все обучение культуре предполагает изменение поведения, но только культурная осведомленность и терпимость влияет на собственное поведение и поведение других.

REPLICA – копия (реплика).

Какие материалы и какой подход?

Для изучения поведения, зависимого от культуры, подходят как обычные учебники, так и учебники специалистов из U.K. и U.S.A., а так же аудио– и видеозаписи, вырезки из газет и журналов и всевозможные, начиная от настоящих билетов на лондонскую подземку и заканчивая копиями Статуи Свободы или Биг Бена.

При составлении курса необходимо тщательно спланировать подход, направленный на изучение культуры. Подход характеризуется совмещёнными обучающими задачами, в которых студенты

- работают в парах или небольших группах, чтобы составить информацию по частям
- делятся и обсуждают то, что они открыли, для того, чтобы сформировать более полную картину
- интерпретировать информацию в контексте иностранной культуры и в сравнении со своей собственной культурой.

Ряд педагогов выяснили, что когда студенты понимают определённые фразы иностранного языка, используемые в данной ситуации, затем продолжают достигать понимания культурных факторов, при работе это для них является самой захватывающей и интересной частью любого урока иностранного языка. Изучение культуры с целенаправленным подходом обучения расширяет область понимания, как для студентов, так и для преподавателей.

1. РАСПОЗНАВАНИЕ СИМВОЛОВ И ИЗОБРАЖЕНИЙ В КУЛЬТУРЕ.

Когда мы живем в определённой стране, то мы автоматически становимся

к ряду изображений и символов, содержащихся в песнях и картинах, достопримечательностях и обычаях. Эти изображения и символы включают в себя знаменитых в искусстве людей, а в архитектуре или природе, такие как Белый Дом в Вашингтоне и белый утес Довера.

Знакомство с этими образами помогает студентам почувствовать себя более свободно и стать увереннее. Целью занятий в данном случае является ознакомление студентов с популярными образами и символами в английской культуре. Второй по важности задачей является способность студентов находить общее и сравнивать образы и символы в Британской и Американской культурах, а затем, по контрасту сравнивать их с образами и символами в собственном языке. Слова и фразы также помогают найти сходства в культуре. Занятия направлены на исследование происхождения слов в языке и как они использовались в древней культуре. Область культуры, которая часто игнорируется, – это звуки. Культурное
может быть определено звуками людей, машин, и даже сельской местности.

Другие виды курса имеют дело со способами подхода к культуре. Они побуждают студентов разрабатывать концепцию самой культуры, или помогают преподавателю понять, что знали студенты о предмете до занятий в классе.

2. РАБОТА С ПРОДУКТАМИ КУЛЬТУРЫ.

Каждый преподаватель знаком с концепцией РЕАЛИИ – физические предметы, такие как открытки, фотографии, картинки и символы, ассоциируемые с культурой иностранного языка. Изображения и символы можно обнаружить в словах песен, в идиомах, определённых словах и выражениях. Эти предметы не так полезны как обучающий языку материал. Знакомство с ними предлагает студентам культурный поток, который помогает им почувствовать себя увереннее и говорить более свободно на иностранном языке. На уроках могут быть использованы такие реалии как: сувениры, мультфильмы, монеты, фотографии, рассказы о путешествиях, газеты, новости радио и телевидения, марки и значки. Эти предметы относительно легко приобрести иностранным преподавателям, и они обладают многоцелевыми функциями перенесения мира иностранной культуры в класс. Более того, чрезвычайно важно, для коммуникации студентов в классе, иметь в руках действительно существующие вещи и работать с ними на уроке.

Цель работы с продуктами культуры – помочь студентам достичь свободы в языке ,используя аутентичный материал .И позволить им ,рассматривая и описывая реалии иностранного языка сравнить их с такими же реалиями в своей собственной культуре .

Важным аспектом таких видов работы является фокусирование на языковых навыках:устное сочинение «Сочинение о культуре» , письменные навыки в «ролевых играх «радионовости »и навыки чтения в описании «обложки журнала или газеты и Заголовки».

Ключевым элементом таких занятий является расширение границ для личного участия и возможности выразить личный ответ .

Для проведения данных занятий преподавателем не требуется уезжать заграницу. Достаточно найти какую-нибудь футболку с надписью, постер из журнала , буклет турфирмы и любой материал имеющийся в наличии у англичан \ американцев , работающих в России.

3.Изучение примеров из ежедневной жизни .

Каждая культура обладает отличными от других опциями и предполагает особенные примеры , связанные с различными областями ежедневной жизни такие как устройство на работу , жилье , покупки , с увеличением числап студентов ,имеющих возможность путешествовать , равботать и учиться в англоговорящих странах , им необходимо бать более осведомленными в области жизнедеятельности людей в этих странах :

занятия в этом разделе направлены не только на то , чтобы раскрыть образ жизни в англоязычных культурах, но также на то , чтобы воодушевить к сравнению и дискуссии на тему :как могут быть похожи и отличаться опции и примеры в культуре студентов . Таким образом студенты приходят к более глубокому пониманию как англоязычных культур ,так и собственной , они лучше подготовлены к коммуникации с носителями языка и владению обычной ситуацией , с которой они вероятно столкнутся в англоязычных странах .

Работа в этом разделе требует , чтобы студенты использовали аутентичные источники такие как рекламные обьявления в газетах, видеоклипы для того чтобы собрать информацию и выявить факты о ежедневной жизни в англоязычных странах .

В другом виде работ действительная информация об обычаях и привычках в англоязычных странах представлены ввиде задания , которое студенты используют как основу для межкультурного сравнения и дискуссии . В третьем виде работ студентам предоставляется возможность оценить их собственное восприятие ежедневных примеров культуры англоязычных стран и видоризменить любое направленное представление , которое может иметь место . Занятия в этом разделе могут бытиь эффективно использованы как в монолингвистических так и в мультилингвистических классах.В классах, где стуленты выходят из однлой и той же культуры эти виды работ будут также полезны как средства возрастающей осведомленности в разнообразии существующих в культуре самих студентов и индивидуальных концепций студентов по таким культуроосвещенным темам как «семья» с «дом» .

4.Изучение культурного поведения .

Целью данных занятий является увеличение осведомленнности и чувствительности к различным типам поведения иноязычных культур.