AN ABSTRACT OF THE THESIS OF

Seung Min Lee for the degree of Master of Science in Electrical and Computer Engineering presented on May 20, 1988.

Title: Design and Simulation of a Simple Digital Computer Central Processing Unit Using Computer-Aided Design Software on a Personal Computer

Redacted for Privacy

Abstract approved: _____ J. H. Herzog

Computers have made a great contribution to the design process in various applications, such as design analysis and simulation. A new technology called Computer-aided design (CAD) has helped the designer to design and build very intricate systems.

While most of CAD software is run on workstations or microcomputers with one or more display devices of high resolution for graphics, a considerable amount of CAD software has recently been developed and introduced for personal computers. This software can be run on personal

computers with minimum configuration, making them suitable for use in educational environments such as Computer Engineering design courses.

Schematic Logic Analyzer and Verifier (SLAV) is one such software package for personal computers and has been granted to the Electrical and Computer Engineering Department of Oregon State University by Automated Logic Design Company (ALDEC) for educational purposes.

SLAV is being evaluated for use in a digital computer design course at the Department. A major drawback is its lack of a step-by-step instruction manual for the users, who are not familiar with the software.

This thesis provides a solution to this problem by showing how to use the software, eventually providing a full instruction manual for SLAV. To demonstrate its potential, the Central Processing Unit (CPU) of a simple digital computer has been designed by using SLAV. This computer is simple with a minimum hardware configuration. It is an 8-bit machine with a memory unit, an arithmetic logic unit, a control unit and six registers. Three representative instructions are implemented to allow simulation of the computer to show how it works.

DESIGN AND SIMULATION OF A SIMPLE DIGITAL COMPUTER CENTRAL PROCESSING UNIT USING COMPUTER-AIDED DESIGN SOFTWARE ON A PERSONAL COMPUTER

by

Seung Min Lee

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed May 20, 1988

Commencement June 1989

APPROVED:

Redacted for Privacy

Associate Professor of Electrical and Computer Engineering in charge of major

Redacted for Privacy

Head of department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented May 20, 1988

Typed by Seung Min Lee for Seung Min Lee

ACKNOWLEDGEMENTS

Many thanks go to Professor J.H. Herzog, my major professor, for his considerate guidance and hint for writing up this thesis. His support with information on the SLAV software is greatly appreciated.

Many thanks are also due to my fellow students who have given helpful suggestions and advice for designing the computer system.

I would like to thank my mother, brother and sister for their spiritual encouragement and support even though they are away from me.

I also would like to thank my little boy, Inyoung, for being very good and patient with his dad, who could not spend more time with him.

Last, but not least, I thank my beloved better half, Nam Hee, for her devoted support and backup while I was working on this thesis. Although she has had her hands full with taking care of a new-born baby, she never pulled a long face with my work. I do appreciate her having been patient of all manner of spiritual and physical tribulation.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

DESIGN AND SIMULATION OF A SIMPLE DIGITAL COMPUTER CENTRAL PROCESSING UNIT USING COMPUTER-AIDED DESIGN SOFTWARE ON A PERSONAL COMPUTER

CHAPTER 1

INTRODUCTION

1.1 Background

A major activity of computer engineers is to design and develop marketable products. This requires making sketches, drawing flowcharts and block diagrams, building test models, and doing simulation. To meet the design specifications and requirements, the designer should go through trial processes and make adjustments, to improve and finally complete the design [PAO].

Computers have been used to facilitate the design process in a variety of ways including analysis of designs and simulation. They are well suited for performing repetitive, time-consuming and extremely detailed work at very high speed [BARR].

In past years, before a convenient and efficient new technology called computer-aided design (CAD) was introduced, it was a time-consuming and complicated job for a design engineer to design and draft systems. It could take so long to generate a design for a complex system or an integrated chip that it might become obsolete before it could be manufactured.

The advent of CAD technology provided a convenient and efficient way of coping with the problems associated with the design and construction of very complex systems, robots, and integrated circuits [BEGG]. It also increased the productivity of the designer by reducing the design time. The use of CAD also reduces design errors, leading to better designs.

1.2 Computer-Aided Design (CAD)

A versatile and powerful technique known as computer graphics has facilitated development of CAD. Computer graphics is based on the creation and manipulation of images on a display device with the aid of a computer [BESA]. The more sophisticated computer graphics has become, the more rapidly CAD technology has been developed.

CAD can be defined as the use of computer systems to help in the creation, modification, analysis, or optimization of a design. The computer systems are made up of hardware and software to meet the specific design specifications and requirements required by the particular user. The computer, one or more graphics display devices, keyboards, and other peripheral equipment are included in the CAD hardware. CAD software is made up of the computer programs to implement computer

graphics on the system plus application programs to facilitate the design functions of the user [GROO]. Simulation and test functions are included in application programs.

For visual display of a designed part, a television-like cathode ray tube (CRT) hooked to a computer is required. The designer enters his design into the computer by an input device, and commands the computer to alter his design in some way, such as adding or deleting connections. Simulation of the real-world operation of the part being created by the designer can be done at any time. Simulated testing and evaluation of the components being designed are also allowed with the test results displayed on the screen [LOGS].

Most CAD software is difficult and complex to use and requires a workstation, microcomputer or even a larger computer with a costly display terminal of very high resolution. Other computer facilities such as plotters and secondary storage are also required. For these reasons they are seldom used in computer engineering courses for educational purposes.

A great variety of CAD software techniques and packages have been developed and introduced as more sophisticated personal computers were developed, and as processing speed and graphics quality increased dramatically. Personal computers now have high enough

performance that they can handle CAD software. Very high-quality monitors have enough resolution for the graphic display of the CAD task.

1.3 Schematic Logic Analyzer and Verifier (SLAV)

Schematic Logic Analyzer and Verifier (SLAV) by Automated Logic Design Company (ALDEC) is an example of CAD software for personal computers.

SLAV enables the designer to trouble-shoot logic design at the schematic level. Using SLAV, the user can make schematic entry and perform logic simulation concurrently, and develop logic designs step by step. The user can concentrate on design analysis and verification rather than on schematic generation.

When it comes to the drafting capability of SLAV, drawing sizes of up to 430"*500" are allowed. Up to 16,380 nodes and 600 IC chips per sheet are allowed. Any connection line or IC chip can be deleted or added at any time. The user can jump into any area of the schematic screen by positioning the cursor within the "scanner".

Any signal line or IC can be compiled and simulated at any time by SLAV simulator. Up to 28 input and output signals can be monitored concurrently during simulation. The user can easily change his design at any time because he can simulate his design schematic and see the results

at any time. The simulation screen can be scrolled at three different speeds. Simulation timing diagram can be printed in either vertical or horizontal direction. This eliminates a need for pasting of multiple sheets of timing data.

Another important feature of SLAV is ALDEC Logic Compiler (ALC). ALC is SLAV's model compiler, which compiles a user model source file called a netlist file. This compilation produces a new executable file with ".ald" extension. Any schematic developed under SLAV can be converted into a chip by ALC as a combinational logic circuit can be implemented in a Programmable Logic Array (PLA). It helps a lot to minimize logic complexity.

SLAV comes with numerous logic libraries including TTL, memories and gate array cells to accommodate a large variety of logic designs.

1.4 Scope of the Study

SLAV has been granted to the Electrical and Computer Engineering Department of Oregon State University by ALDEC for educational purposes. It is hoped that it will prove suitable for student design projects in the Department. Unfortunately, a well written instruction manual for SLAV is not provided for potential users.

The purpose of this study is to provide a solution

to this problem by first writing an instruction manual for system designers, who are unfamiliar with SLAV. Then an example involving the design and simulation of a small digital computer will be attempted.

A step-by-step instruction manual for SLAV has been written and may be found in Appendix A. The instruction manual covers hardware requirements for running SLAV and software installation depending on the user's computer system configuration. It also covers schematic entry on a drafting screen and simulation on a simulation screen with a full description of drafting and simulation screen menus. In Chapter 2, a simple digital computer has been designed and simulated as an example to illustrate how to use SLAV efficiently.

CHAPTER 2

DESIGN AND SIMULATION OF A SIMPLE COMPUTER

2.1 Overview

A simple computer has been designed and simulated using SLAV in this study as an example to show how to use SLAV on a personal computer. The computer consists of a memory unit and six registers. Each of the registers is 8 bits long with the exception of Instruction Register and Timing Counter of 4 bits.

The control unit is designed as a combinational circuit block according to the control functions obtained from the register-transfer information [MANO].

Step-by-step instructions and a full description of SLAV are given in Appendix A. This covers hardware requirements for running SLAV, software installation depending on the user's personal computer system configuration, schematic entry on the drafting screen, simulation on the simulation screen with a full description of drafting and simulation screen menus. Design details such as simulation results are found in Appendix B.

All the files related with this study are provided in a separate diskette along with a SLAV system diskette.

2.2 Design

Design of a simple computer is considered as an example to see how SLAV works and how it can be used in actual design work. The block diagram of the computer is shown in Figure 2. It would be helpful to discuss a little about Central Processing Unit (CPU) before we plunge into design of a computer.

The main function of the CPU is to regulate the operation of all system components and perform the arithmetic and logical operations on the data. The CPU is made up of the following operating units to accomplish the functions [GROO]:

1. Control Unit
2. Arithmetic and Logic Unit (ALU)

The control unit generates control signals to control the various register transfer operations required to execute program instructions. The control unit is designed with logic circuits to interpret instructions and generate control signals. This is known as a hardwired control unit. However, if the number of instructions and control lines is in the hundreds, the control unit could be extremely complicated, costly and difficult to design. In such a case, it might be

designed by microprogramming [HAYE]. The control unit directs the operation of the ALU and the CPU registers.

Various arithmetic manipulations and logic functions are performed in the ALU in binary form. These operations are selected by the function-select lines of the ALU, and include addition and data transfers.

Both the control unit and the ALU need some registers called CPU registers to perform their functions. These registers are small memory devices that can receive, hold, and transfer data. The number of bits in the register determines the word length the computer can handle. The CPU registers and their functions are as follows [BESA]:

Program Counter: The program counter holds the address of the next instruction to be performed by the CPU. The program counter is incremented by one to indicate the next instruction word after each instruction word is taken.

Memory Address Register: The memory address register indicates the location of data stored in memory.

Instruction Register: The instruction register holds the current instruction code for decoding.

Accumulator: An accumulator is a register used to store temporarily the immediate results from an

arithmetic or logic operation.

The computer is considered as an aspect of register-level logic design. The physical distance between the components is not very long, probably less than a meter. The computer does not have very many registers. Hence, the components are interconnected point to point, not by buses.

A preprogrammed ROM or EPROM would be ideal as the memory unit where all instructions and data are stored, but an appropriate ROM and associated information are not available in SLAV. SLAV does not provide a way to modify the contents of EPROM. Hence, a 8192*8 RAM is chosen from the SLAV memory libraries. The size of it is more than enough for a simple 8-bit computer, but would be good for future expansion. The RAM would be more convenient when the computer is simulated with sundry different instructions because it is easy to change the contents of memory. All instructions and data should be set in the RAM before simulation of the computer. The sequence of operations to store instructions and data would be as follows:

1. Transfer the address bits of the desired contents into MA.
2. Transfer the data bits of the desired contents

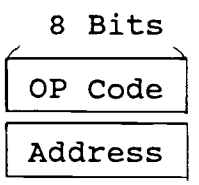
into MB.

3. Activate the WRITE control signal of memory.

SLAV does not provide a way to modify the contents of RAM directly. To modify its contents, it is necessary to assign the desired address bits directly to the address lines and the desired contents to the input lines.

Now consider designing a simple 8-bit computer central processing unit to see how SLAV can be run on a personal computer and how it works. The computer consists of a memory unit, an arithmetic logic unit, a control unit and six registers: Accumulator (AC), Memory Address Register (MA), Memory Buffer Register (MB), Program Counter (PC), Instruction Register (IR), and Timing Counter (TC). All registers are 8 bits long with the exception of IR (4 bits) and TC (4 bits).

The computer adopts the following instruction format with operand specification of direct addressing:



An operation code (OP Code) is 8 bits wide, and so is the address of the operand. With this instruction format, the computer has to go through some extra microoperations

in the instruction execution cycle to read the operand address.

As a simple example, consider the three instructions, which the computer can execute. While most computers are able to execute a variety of instructions, only the following three instructions have been implemented in this design.

ADD Add	Add operand specified by Address to AC
JMP Add	Jump to Address
LDA Add	Load operand specified by Address into AC

Each instruction requires a fetch cycle for decoding its operation code. Before the computer starts executing an instruction, it should have all information about the instruction. The following sequence is called the fetch cycle because the computer fetches the operation code of the instruction from memory [MANO]:

1. The address of the operation code is read from the Program Counter (PC) to the Memory Address Register (MA).
2. The operation code is read from memory to the Memory Buffer Register (MB) according to the

address of the operation code in the MA. The PC is incremented by 1 to indicate the next address.

3. The operation code is transferred from the MB to the Instruction Register (IR) where it is decoded by the control unit.

After the operation code is decoded in the fetch cycle, the execution of the instruction proceeds in the execution cycle. When the computer is done with the execution of the instruction in the execution cycle, it returns to the fetch cycle to fetch the first byte of next instruction from memory. All instructions go through the fetch cycle.

The use of a task diagram provides a convenient way to specify the action of the computer [HERZ]. The task diagram in Figure 1 shows the flow of the tasks the computer performs to execute the three instructions. The task diagram makes it easy and clear to observe the transfer of the register contents, and shows the step-by-step procedure of the tasks the computer performs.

For better understanding of the action of the computer, microoperations can be defined as the operations performed on the data stored in registers.

First start with the control unit in Figure 2. The IR receives the operation code of instructions from the

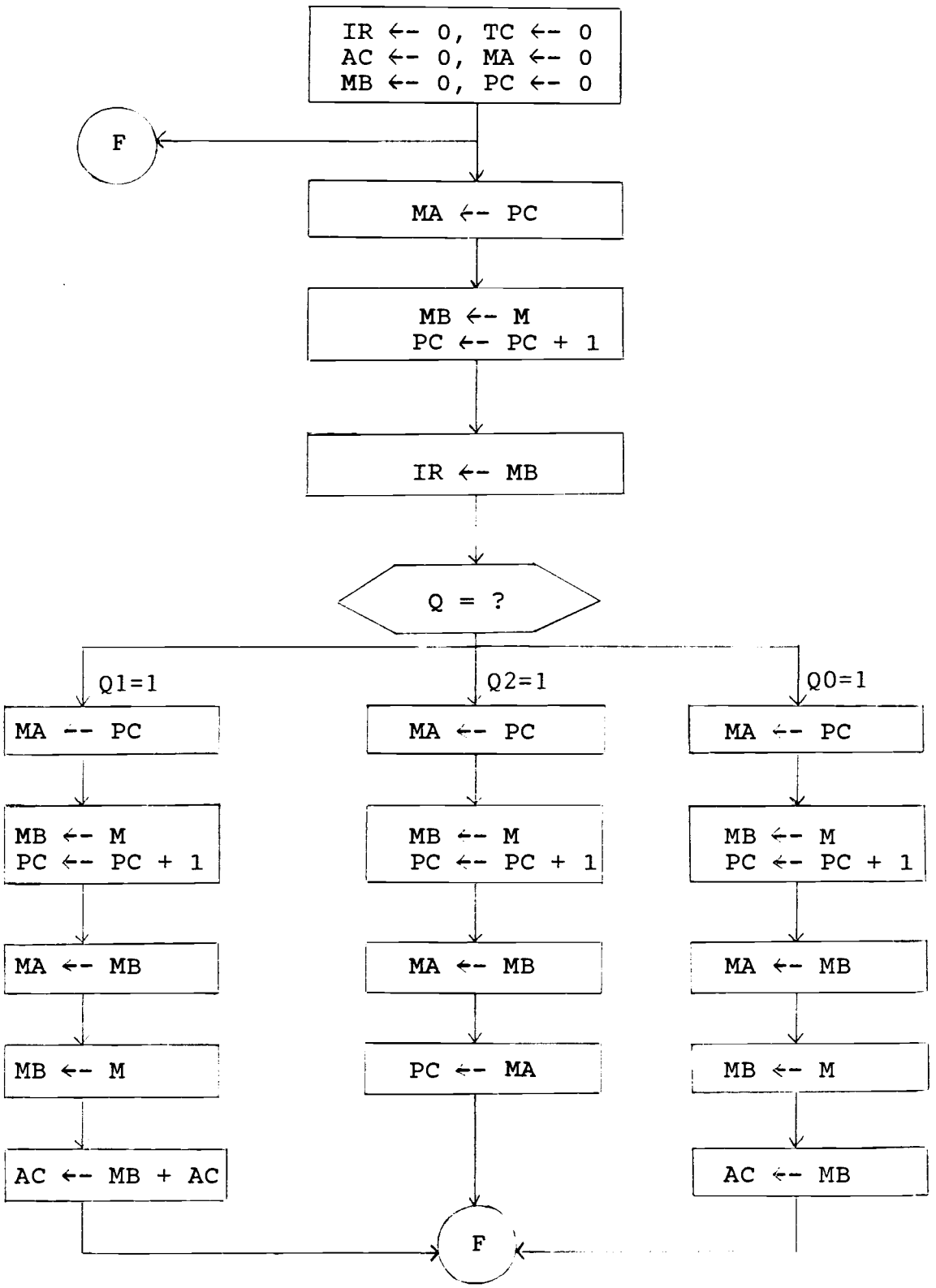


Figure 1. Task Diagram of a Simple Computer

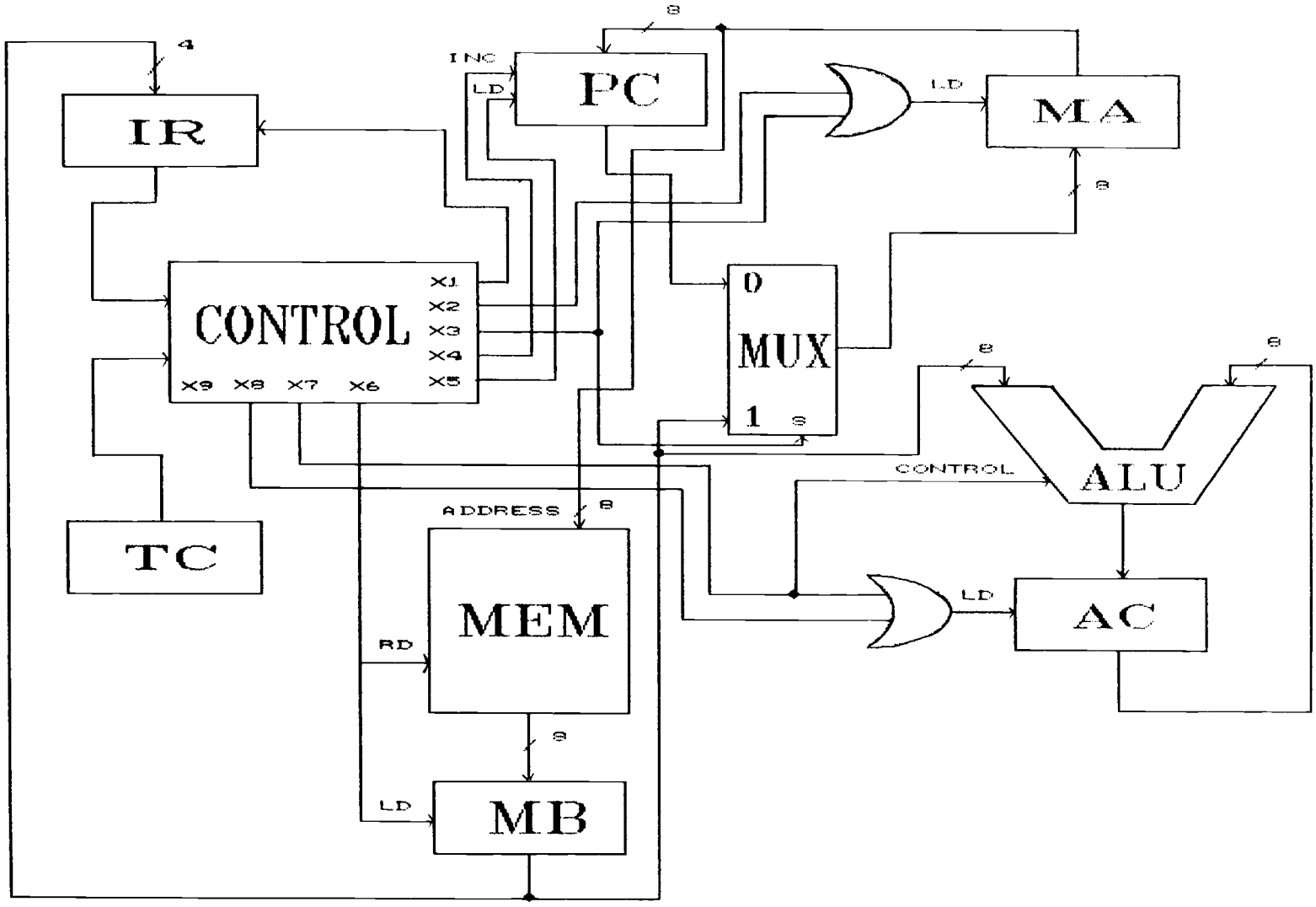


Figure 2. Block Diagram of a Simple Computer

MB. The operation code is decoded by the decoder to supply one output for each operation code. The output is called operation variable Qi, i=0,1,2,...: Q0=1 if the op code is binary 0, Q1=1 if the code is binary 1 and, Q2=1 if the code is binary 2. The TC is to supply eight timing variables, T0 to T7. The timing variables are used to sequence the microoperations. Only three bits of the TC are sufficient to generate the eight sequential timing pulses. Then a control signal to reset the TC at T0 is not necessary, making the control unit a little simpler. However a control signal to reset the TC at T0 is provided in the control unit as X9 for the future use.

Each microoperation is specified by one timing variable and one operation variable. The following sequence of microoperations execute the three instructions:

Fetch cycle

T0: MA ← PC

T1: MB ← M, PC ← PC + 1

T2: IR ← MB

ADD Add

Q1T3: MA ← PC

Q1T4: MB ← M, PC ← PC + 1

Q1T5: MA ← MB

Q1T6: MB ← M

```
        Q1T7:     AC ← MB + AC
JMP Add
        Q2T3:     MA ← PC
        Q2T4:     MB ← M
        Q2T5:     MA ← MB
        Q2T6:     PC ← MA
LDA ADD
        Q0T3:     MA ← PC
        Q0T4:     MB ← M, PC ← PC + 1
        Q0T5:     MA ← MB
        Q0T6:     MB ← M
        Q0T7:     AC ← MB
```

The above register-transfer logic is to specify the sequence of control functions in a digital computer. The list of microoperations and control functions for a digital computer specifies the types of registers and associated digital functions that must be incorporated in the computer [MANO]. So it is very important to obtain them.

The first step in obtaining control functions is to pick up the same microoperation on the same register from the above list. For example, the microoperation MA ← MB is performed several times in the sixth line with control function Q1T5, in the eleventh with control function Q2T5, and in the fifteenth with control function Q0T5.

The three control functions can be combined into one control function:

$$X3 = Q0T5 + Q1T5 + Q2T5 \qquad : MA \leftarrow MB$$

In like manner, just group the same microoperations and combine them to generate other control functions as follows:

$$X1 = T2 \qquad : IR \leftarrow MB$$

$$X2 = T0 + Q0T3 + Q1T3 + Q2T3 \qquad : MA \leftarrow PC$$

$$X3 = Q0T5 + Q1T5 + Q2T5 \qquad : MA \leftarrow MB$$

$$X4 = T1 + Q0T4 + Q1T4 + Q2T4 \qquad : PC \leftarrow PC + 1$$

$$X5 = Q2T6 \qquad : PC \leftarrow MA$$

$$X6 = T1 + Q1T4 + Q1T6 + Q2T4 + Q0T4 + Q0T6$$

$$= X4 + Q1T6 + Q0T6 \qquad : MB \leftarrow M$$

$$X7 = Q0T7 \qquad : AC \leftarrow MB$$

$$X8 = Q1T7 \qquad : AC \leftarrow AC + MB$$

The above control signals are applied to the associated control inputs of registers, multiplexer and ALU.

Each register has control inputs of IN, LOAD, CLEAR and CLOCK. IN is count-enable input and must be in high state to count. Setting up a low level at the LOAD input loads the preset data to the register. A low level at

the CLEAR input clears the register regardless of the levels of CLOCK, LOAD and IN. The CLOCK input triggers the change on the rising (positive-going) edge.

The multiplexer has two control points, SEL and E. SEL is select input, and E is enable input. E must be at low level to select one of the two input data.

ALU has four function-select lines (S0, S1, S2, S3) by which arithmetic and logic operations are selected.

The control unit can be designed with AND and OR gates along with two decoders whose inputs come from IR and TC. The circuit diagram of the control unit is shown in Figure 3. In Figure 3, there is no need to write in memory. So the READ input of memory and LOAD input of MB can be always set in low state, leaving the control signal X6 idle. When it is necessary to modify the contents of memory, the WRITE input of memory should be set low with READ input in high state. Then X6 can be used to control READ input.

All the components used are in the SLAV TTL and Memory libraries. They are as follows:

Decoders:	7442- BCD-to-Decimal Decoder (1-10)
Registers:	74161- 4-Bit Binary Counter
	74163- 4-Bit Binary Counter
Multiplexer:	74157- Quad 2-Input Data SEL/MUX
ALU	74181- 4-Bit Arithmetic Logic Unit

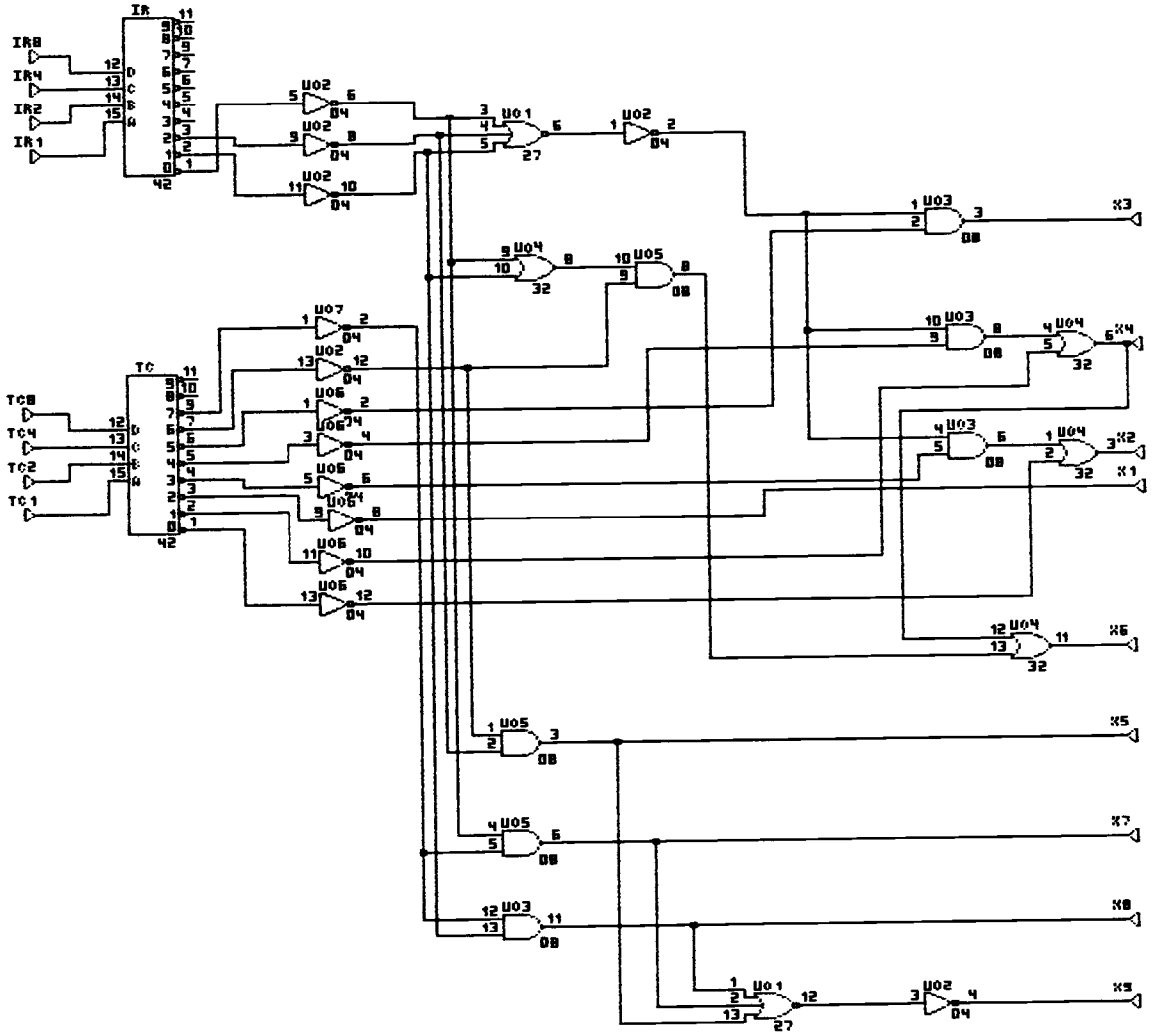


Figure 3. The Control Unit

All the registers require a load or an increment control input. While 74161 has asynchronous CLEAR input, 74163 has synchronous CLEAR input for TC. You must refer to the data book to get detailed information on the chips. The whole and complete diagram of the simple computer implemented with the appropriate parts is shown in Figure 4.

As you can notice in Figure 4, the control unit is implemented in a chip instead of many AND and OR gates. Control functions obtained from microoperations are realized in the sum-of-products form as a set of combinational functions. These can then be realized and minimized in a programmable logic array (PLA). Function minimization techniques can be used to minimize chip counts and logic complexity. The size of the PLA is determined by the number of inputs, the number of product terms, and the number of outputs. So these numbers might be limitations to implementing the PLA. The same technique was applied to the registers except IR and TC, Multiplexer and ALU. You can convert any circuit design into a chip by generating a netlist file and compiling it under ALDEC Logic Compiler (ALC). Detailed information on how to convert your own design into a chip by using a netlist file and ALC is found in Appendix A.

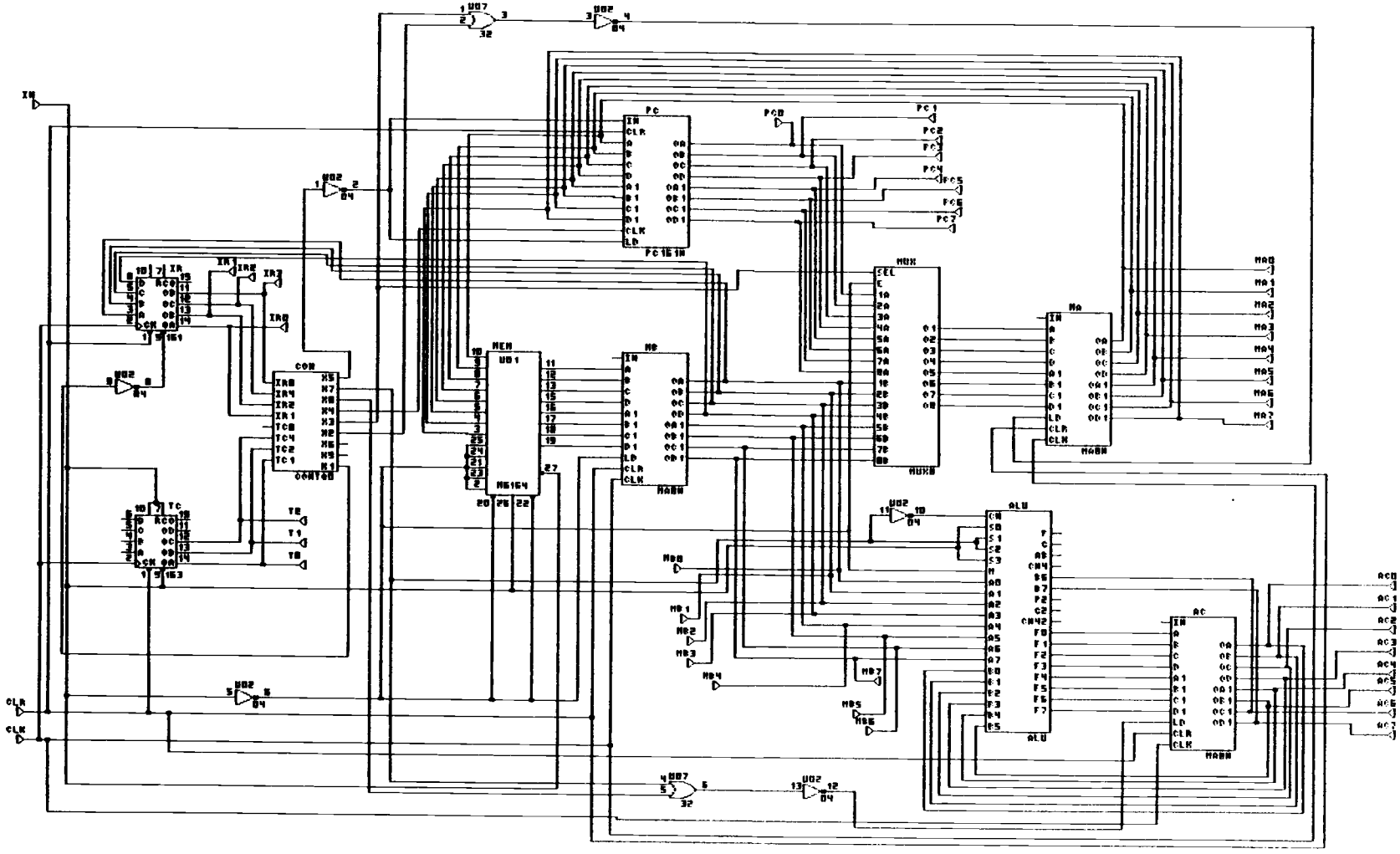


Figure 4. Circuit Diagram of the Simple Computer

2.3 Results

The computer designed in Section 2 as in Figure 4 might be considered very simple and primitive with minimum components, but good enough for an example to show how to use CAD software like SLAV on a personal computer in a computer design course at the Electrical and Computer Engineering Department.

The computer has been simulated by SLAV on a personal computer to see if it works properly. Simulation of the computer can be done with ease because SLAV provides for concurrent schematic drafting and logic simulation. It was simulated with the following three operations and worked as expected:

```
START:    LDA  1011
          ADD  1100
          JMP  START
1011:     0100          :Data1=4
1100:     0101          :Data2=5
END
```

The results of simulation by registers from the simulation screen of SLAV are found in Appendix B.

CHAPTER 3

SUMMARY AND CONCLUSION

SLAV is a representative example of CAD software running on personal computers. It has been granted to the Electrical and Computer Department for educational purposes. However, a detailed instruction manual is not provided for the designers, who are new to SLAV. Hence an instruction manual is written and found in Appendix A.

As an example to show how to use SLAV and how it is running on a personal computer, a simple digital computer has been designed with a minimum hardware configuration. The computer has been simulated by SLAV to see if it works as expected. The computer worked fine as expected. The results of simulation are found in Appendix B.

Some drawbacks in SLAV were found during the design and simulation. SLAV does not provide a convenient way to modify the contents of memory. So it might be a little cumbersome to do it during simulation. When implementing ALU, the signal designation for addition was not corresponding to the data book. The use of an inverter solved the problem.

The study thus far covers the computer design from the system designer's point of view. It includes schematic entry and logic simulation processes and provides step-by-step instructions to use SLAV. The

computer designed in this study is basic and simple, but a good exemplary model to show how to design a simple computer system and how to use SLAV.

It is assumed that the computer can execute many more instructions even though only three of them are implemented here to show if it works and how it works. Some other instructions can similarly be implemented if the contents of memory are modified. It would be worth while to design a larger computer using it as a processor with buses. Alternatively, the control unit can be designed with microprogramming using a ROM controller if more instructions are to be implemented.

The instruction manual for SLAV found in Appendix A can be used as a reference in the computer design course. SLAV is easy to learn if the user follows the instructions step by step.

BIBLIOGRAPHY

[BARR] P.C. Barr and R.L. Krimper, et al, CAD: PRINCIPLES AND APPLICATIONS, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985, pp. 1-9.

[BEGG] V. Begg, Developing Expert CAD Systems, UNIPUB, New York, New York, 1984, pp. 7-18.

[BESA] C.B. Besant and C.W.K. Lui, COMPUTER-AIDED DESIGN AND MANUFACTURE, Ellis Horwood Ltd., Chichester, England, 1986, pp. 13-24, 25-28.

[GROO] M.P. Groover and E.W. Zimmers, Jr., CAD/CAM: COMPUTER-AIDED DESIGN AND MANUFACTURING, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984, pp. 1-3, 13-20.

[HAYE] J.P. Hayes, COMPUTER ARCHITECTURE AND ORGANIZATION, McGraw-Hill Book Company, New York, New York, 1978, pp. 160-167, 244-260.

[HERZ] J.H. Herzog, PROGRAMMABLE SYSTEMS, STRUCTURED DESIGN, AND ROBOTICS, 1984, pp. 3.17-3.21.

[LOGS] T. Logsdon, COMPUTERS TODAY AND TOMORROW, Computer Science Press, Rockville, Maryland, 1985, pp. 217-225.

[MANO] M.M. Mano, Digital Logic and Computer Design, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979, pp. 289-297, 339-352, 464-484.

[PAO] Y.C. Pao, Elements of Computer-Aided Design and Manufacturing, John Wiley & Sons Inc., New York, New York, 1984, pp. 4-12, 24-29.

E.E. Murphy and K.I. Werner, Design Automation, IEEE Spectrum, January 1988, pp. 35-37.

The Bipolar Digital Integrated Circuits Data Book for Design Engineers, Part A TTL/Interface Circuits, Texas Instruments, Dallas, Texas, 1985.

CMOS Microprocessors, Memories and Peripherals Data Book, RCA Corporation, Somerville, New Jersey, 1984.

Memory Components Handbook, Intel Corporation, Santa

Clara, California, 1983.

Schematic Logic Analyzer (SLAV) User's Manual, Rev. 3.0, Automated Logic Design Company (ALDEC), Newbury Park, California, 1987.

TTL Applications and Practice, Korea Advanced Institute of Science and Technology, Seoul, Korea, 1980.

APPENDICES

APPENDIX A

SCHEMATIC LOGIC ANALYZER AND VERIFIER (SLAV)

A.1 SYSTEM INSTALLATION

A.1.1 Required Hardware

The minimum hardware configuration for this program is:

-IBM PC/XT/AT or compatible computer.

-256K bytes of RAM memory.

-Single disk drive.

-Color/Graphics Adapter or EGA.

-Color or monochrome display.

-One serial port for mouse device.

You may install a second monitor, which provides a display of simulation concurrently with schematic drafting.

It is recommended that a three button mouse be used because the entire SLAV software has been written around it. Mouse operation with SLAV is discussed later in this chapter.

A.1.2 Software Installation

Single Floppy Disk Operation

If you have a single disk drive, you must use the supplied diskettes or their copies to run SLAV software:

1. Boot your DOS.
2. Load your mouse driver.
3. Place the SLAV/SYS diskette in the disk drive and enter:

```
sla
```

The SLAV software displays the drafting and simulation screens alternately, and finally drafting screen appears ready for any operation

Dual Floppy Disk Operation

You can make a selfbooting SLAV disk with a write-protected DOS diskette and any blank diskette before you first run SLAV if your system has dual disk drive configuration:

1. Boot your DOS.
2. Place the SLAV/SYS diskette in drive A and

enter:

bootable

3. Follow the screen instructions.

4. Your bootable SLAV diskette is ready to use.

The bootable SLAV disk enables you to start SLAV without DOS. Just place the bootable SLAV disk in drive A and turn the power on. You can start SLAV from a DOS prompt by entering:

slav

Note: Enter sla instead of slav when using the SLAV/SYS diskette which is not bootable.

Hard Disk Installation

It would be much more convenient to run SLAV if you fortunately have hard disk in your computer. The first thing you have to do before running SLAV is to install the software on your hard disk.

To install SLAV on hard disk, you must know what port your mouse is installed on and whether the current drive is A or B. Simply follow the following steps:

1. Place the SLAV/SYS diskette in the current drive and enter:

```
install 1 a     ;     if mouse is on com1 and
                      if current drive is A
install 1 b     ;     if mouse is on com1 and
                      if current drive is B
install 2 a     ;     if mouse is on com2 and
                      if current drive is A
install 2 b     ;     if mouse is on com2 and
                      if current drive is B
```

2. "harddisk installator" comes on the screen with instructions, and follow them step by step.

The "installator" makes a directory named Aldec and asks you if you have various chip library diskettes to copy all the necessary files, then just follow its prompts.

If you happen to have dual disk drive configuration, you have only to know what port your mouse is installed on:

1. Place the SLAV/SYS diskette in drive A and enter:

```
harddisk 1      ; if mouse is on com1
harddisk 2      ; if mouse is on com2
```

2. "harddisk installator" comes on the screen with instructions, and follow them.

To start SLAV from the hard disk, make sure that you are in the root or Aldec directory and enter:

```
slav
```

A.2 DRAFTING SCREEN

A.2.1 Mouse Operation

It is recommended that a three button mouse should be used. The following button designations will be used throughout this manual:

(L) ; left button
(M) ; middle button
(R) ; right button

The mouse buttons can be used individually or in tandem. Operation of the buttons for SLAV is explained when it is needed.

A.2.2 Screen Menu

The drafting screen menu is located on the right side of the screen in the form of windows with appropriate labeling. A description of each window and its function is given below with an example.

Simulate--This menu allows SLAV to switch to the simulation screen from drafting screen.

Ex: 1. Place cursor in the "Simulate" window and press

any button.

2. Now you are on the simulation screen with a rectangular cursor. Move cursor a little bit to see how it works.
3. To get back to the drafting screen, move your cursor over the "Draw" window on the bottom right side of the screen and press any button.
4. You are back in the "Simulate" window of the drafting screen.

Note: When selecting a menu or option of the menu, any button can be used.

Options--This menu allows you to select one of the 6 drawing sizes. The default drawing size is C (17"x22"). Once you enter the "Drawing size" option, you cannot get out of it without selecting a drawing size even by pressing both (L) and (R).

Ex:
1. Place cursor in the "Options" window and press any button.
2. Select "Drawing Size". You get "Change size from C to:" with the 6 drawing size options.
3. Select the size A.
4. Do not expect any change on the screen. No change is made on the screen.

Chip--With this menu you can select chips from the various chip libraries available to you. The menu displays only 16 chips with "any other" option, which allows you to select any other chip from the optional chip libraries. When "any other" option is used, you must enter the chip type according to the following:

TTL: enter only the last two meaningful digits of the chip, e.g., "00" for the 7400.

Memory: enter the complete part number of the chip with the prefix M.

You cannot get out of this menu without selecting a chip even by pressing both (L) and (R). Select any chip and press both (L) and (R) if you want to get out of this menu without selecting a chip. Selecting operation can be canceled by pressing both (L) and (R) right after you select a chip.

Ex: 1. Place cursor in the "Chip" window and press any button. You get 16 chip choices with "any other" option.

2. Select "00". You are asked to give a name to the chip. You can select the SLAV suggested name or any other of your choice.

3. Select "any other", and you are asked for a name.
4. Enter "A01". The chip named "A01" appears on the screen, moving together with cursor.
5. Find a location for the chip and press any button, freeing cursor for another job.
6. In like manner, select another same chip "A01" and place it about 2" to the right of the first one.

Erase--This menu allows you to erase anything from the drafting screen. The option "Connection" erases a connection line while "Tree" erases entire node. "Chip" erases chips only, and "I/O" erases the triangular I/O symbols and their names. "All" erases everything on the screen.

Ex: 1. Place cursor in the "Erase" window and press any button, getting 6 options including "Abort".
2. Select the "Chip" option.
3. Move cursor onto the second "A01" and press (L) or (R). Cursor freezes on the chip with a beep.
4. Press (L) or (R) again, and see "A01" disappear.

5. Press (L) and (R) at the same time to get out of this menu.

Note: If you want to use another "Erase" option, you must get out of the menu first and enter again.

Disk--With this menu you can store and retrieve data files, such as sheet of drawings, timing diagrams of simulation, and netlists. Unless otherwise stated, all disk options are to the current directory, that is SLAV/SYS diskette in drive A or the Aldec directory on the hard disk. With the "Directory" option you can not only scan the current directory but also save and retrieve data files. The "Change dir" option enables you to change the current directory to any other. With "Netlist" you can create netlist files from sheet files. Netlist files are DOS files with .src extension, including listing of all chips and their I/O connections.

Ex: 1. Place cursor in the "Disk" window and press any button. You get 11 options including "Abort" and "Exit to DOS".

2. Select "Save sheet". You are asked to give a file name.

3. Enter "Sam" for the file name. A sheet file SAM.SHT is created in the current directory.

Print--This menu allows you to have both the schematic drawing and timing diagrams printed. Without an IBM/Epson printer or compatible printer, the system will hang up on you. This menu has two options: "Drawing" and "Timing". The "Drawing" option asks you to select the printer type; either "Epson 80" or "Epson 100". The "Timing" option asks you to select the printing type; "Horizontal" or "Vertical".

Move--With this menu you can move chips around the drafting screen, but cannot move I/O symbols. This menu can be canceled by pressing both (L) and (R).

Ex:
1. Place cursor in the "Move" window and press any button.
2. Move cursor to the chip "A01" and press (L) or (R). Cursor sticks to "A01" with a beep. Move it around and see the chip follow cursor.
3. Find a new location.
4. Press (L) or (R) again, making your cursor free.
5. Press both (L) and (R) to get out of this menu.

Text--This menu allows you to write text anywhere on the screen. The text is part of the drawing file.

Pressing both (L) and (R) gets you out of this menu.

Ex: 1. Place cursor in the "Text" window and press any button.

2. Move your cursor to any location where you want to place a comment.

3. Press (R). A bar-shaped text cursor replaces your normal crosspoint cursor.

4. Enter your message from the keyboard.

5. Move your cursor to the next location if you need to make some more message.

6. Press both (L) and (R) to get out of this menu.

If you want to delete some text, enter the "Text" menu and:

1. Place your cursor on the text you want to delete.

2. Press (L). Then the text cursor pops up at the tail end of the text with a beep.

3. Press "Del" key from the keyboard. The text is gone now.

I/O symbols--These symbols are used for input/output circuit connections. The triangular symbols are used for off-the-sheet connections, while the "o" symbol is used

for on-the-sheet connections. The "E" symbol is used for a temporary line termination. It must be replaced with a permanent I/O assignment, or connected to another line before simulation.

Ex: 1. Place cursor at the triangular I/O symbol on the left and press any button. The menu requests a name for the symbol.

2. Enter "IN1". The I/O symbol named "IN1" appears on the screen, sticking to cursor.

3. Move "IN1" to the left of the chip "A01" and press any button. "IN1" is located at the selected location.

RAM--This window shows what percentage of the RAM memory has been used by the chips entered.on the drafting screen. It is noticed that the first few chips take quite large RAM space while the following chips take considerably less. It is because the next chips of the same type requires very little memory.

Scanner display--This square window shows the actual current location of the cursor and the chips in relation to the overall schematic layout. The window represents the whole schematic screen. A little dot and a small square in the "Scanner" window represents the chip

("A01") and cursor respectively. You can change your cursor location by entering the "Scanner" window and placing your cursor where you want.

Ex: 1. Move your cursor and enter the "Scanner" window. Your crosspoint cursor changes to a dot.

2. Move your dot cursor around in the "Scanner".

3. Place it to the bottom right corner of the "Scanner" and press any button.

4. Move cursor out of the "Scanner" and see where you are. You are on the bottom right corner of the screen.

A.2.3 Netlist

Any schematic developed under SLAV can be automatically converted into a single chip by making a netlist file and compiling it under ALDEC Logic Compiler (ALC).

The netlist file is automatically generated by selecting "Netlist" option from the "Disk" menu. The netlist file is automatically given an extension of ".src" and is stored in the current directory or disk specified by "Change dir" option. The netlist file includes listing of all chips and their input/output

connections.

The netlist file is a DOS file, and can be edited under any DOS compatible editor. It can be printed on the screen or out to a printer using the associated DOS commands.

The following steps enable you to convert any circuit design into a single chip by netlist file:

1. With your design on the drafting screen, select the "Disk" menu. "Netlist" option is one of the "Disk" options.
2. Select the "Netlist" option, then you are asked to give a file name for the netlist. Give a name for the netlist file.
3. Get out to DOS and see you have a file with .src extension, i.e. CONTROL.SRC.
4. Now have the ALC disk in the current drive and type:

alc (file name)

If there is no error, a file CONTROL.ALD is created. Notice that extension is .ald. The TTL chip library file also has the same extension. You can now use the chip CONTROL by selecting "any other" option in the "Chip" menu.

A.2.4 Schematic Entry

Thus far you have learned how to use the drafting screen menu to enter schematic design. The schematic entry process is described in detail by an example of designing a single chip of 8-bit register, MA8N, using two 4-bit binary counter.

The registers require a load and increment control inputs. The counter chip 74161 satisfies the requirements. You need two 74161 chips because the chip is a 4-bit counter.

The final diagram is shown in Figure 5, A. In Figure 5, B, the completed single chip of 8-bit register is shown. To enable successive cascaded stages of the two 74161 chips, the ripple carry output (pin 15) of U04 should be fed forward to the count-enable input (pin 10) of U05. The input "IN" is kept in "1" (high) state for normal count operation of the chip. You can clear the counter by applying "0" (low state) to the input "CLR". You can preset it by applying "0" (low state) to the input "LD". You must know the pin assignments of the chip.

Now get into SLAV and do:

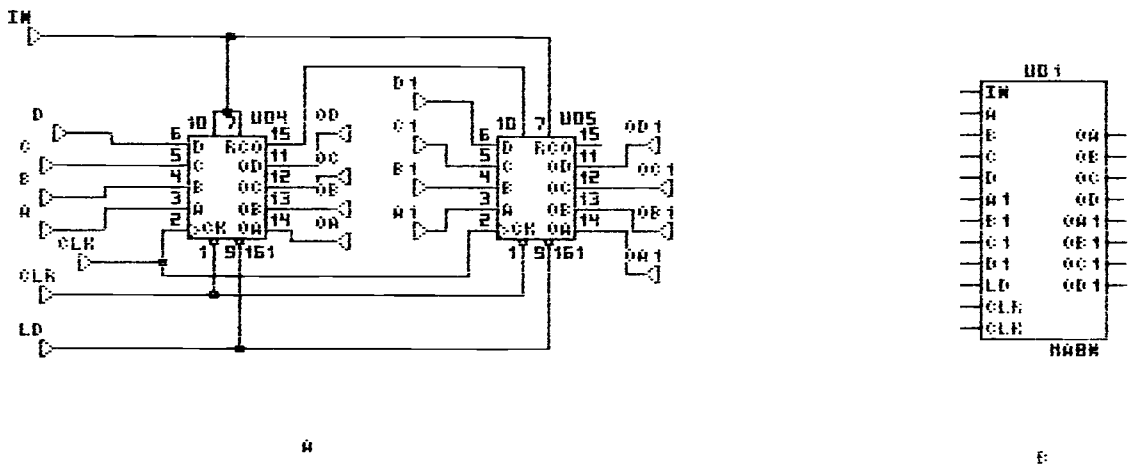


Figure 5. Design of an 8-bit Register, MA8N

1. Select the "Options" menu. You get three options.
2. Select the "Drawing size" option. You get six drawing size options.
3. Select the size A.
4. Select the "Chip" menu. You get 16 chip options with "any other" option.
5. Select "any other" option. You are asked for the chip type.
6. Enter "161" for the chip type. You are then asked for the chip name; suggested name or "any other" option.
7. Select "U04". The chip named "U04" appears on the screen.
8. Place the chip as in Figure 5, A.

Note: You want to take into consideration how to connect the chips when locating them.

9. Select one more "161" with a name of "U05".
10. Place the chip beside "U04". Now you have picked up all the chips needed.
11. Select twelve input symbols and name them "A", "B", "C", "D", "A1", "B1", "C1", "D1", "CLR", "LD", "CLK", and "IN".
12. Locate them as in Figure 5, A.
13. Place cursor near to the input "IN" and press (L),

then cursor snaps on "IN" with a beep.

14. Move cursor away from "IN" and see a yellow line coming off and following cursor.
15. Press (M) and (R) several times to see how the yellow line change its shape. You can cancel the yellow line by pressing both (L) and (R).
16. Place cursor in the proximity of "U05"/7 and press (M) to select the best fitting connection line.

Note: "U05"/7 mean the pin #7 of "U05".

17. Press (L) for the best fitting connection line, then cursor sticks with a beep.
18. Press (L) again to make a permanent connection between "IN" and "U05"/7. The yellow line changes to a green one.

Note: The yellow lines can be erased by pressing both (L) and (R) while the green lines can only be erased through the "Erase" menu.

19. In like manner, connect the eight inputs, " A" to "D1", to the inputs of the "U04" and "U05"; "LD" to LOAD inputs (pin #9) of "U04" and "U05"; "CLR" to CLEAR inputs (pin #1) of "U04" and "U05"; "CLK" to CLOCK inputs (pin #2) of "U04" and "U05".

20. Select eight output symbols with the names of "OA", "OB", "OC", "OD", "OA1", "OB1", "OC1" and "OD1".
21. Locate the first four of them in the vicinity of the outputs of "U04", and the rest in the vicinity of the outputs of "U05" as in Figure 5, A.
22. Connect the eight outputs, "OA" to "OD1" to the outputs of "U04" and "U05" as in Figure 5, A.
23. Select the "Disk" menu. You get 11 options including "Exit to DOS" and "Abort".
24. Select "Save sheet". You are asked for a file name.
25. Enter "REG" for the file name.
26. Select the "Disk" menu again.
27. Select the "Directory" option. You get four options.
28. Select "Sheet". The sheet files are displayed on the screen including "REG.SHT" you just created and saved.
29. Select the "Disk" menu to create a netlist file.
30. Select the "Netlist" option. You are asked for a file name.
31. Enter the same name "REG". Now "REG.SRC" for the netlist file is created.
32. Select the "Disk" menu and "Exit to DOS" option to compile the netlist file, "REG.SRC".
33. Enter the following command and wait a while until

compilation of the netlist file "REG.SRC" is completed by ALC:

```
alc       REG
```

34. Type the command "dir" to check if "REG.ALD" is created in the current directory. A file "REG.ALD" should be created when compiling is finished with no error. Now you have a single chip of 8-bit register.

A.3 SIMULATION SCREEN

A.3.1 Screen Description

The simulation screen has five basic fields: Simulation, Comment, Command Selection, Utility and Information.

Simulation--This field has five vertical fields across the screen.for different simulation functions:

1. Test vector assignment--This field is located on the right of the left margin of the screen. This field is used to assign a test vector for the signal on the right. You can assign any letter [a] through [z] to any signal line on the drafting screen. The associated signal line is toggled at any time during simulation when you press the assigned key. You can assign "B0" to "B9 to any signal driven by the binary counter on the bottom of the signal field.

2. Simulation signal--This field is on the right of the test vector assignment field with ten dots. You can load any signal line into this field.

3. Signal state--This field is on the right of the signal field and shows the current logic state of the signal line.

4. Breakpoint--This field is on the right of the

signal state field. Only 8 of the 16 breakpoints are shown on the screen. To access the other breakpoints, move the cursor in the arrows at the top of this field, and press (L). Press (L) to set "0" state and (R) to set "1" state. (L) and (R), or (M) is used to delete any state. Press both (L) and (M) to set a rising edge. Press both (R) and (M) to set a tailing edge.

5. Simulation data--This field is widest on the right of the breakpoint field. Simulation data is displayed in this field. You can write any text in this field.

Comment--This field is located at the bottom of the screen, right above the bottom margin. This field gives you information about the options available for each cursor location.

Command Selection--This field is located right above the comment field with 7 fields.

1. Simulate--Selection of this field activates the simulation process at speed determined by (L), (M), and (R).

2. Mode--This field allows you to select either Functional (F) or Spike (S) mode. The (F) mode displays functional (steady state) circuit behavior. A unit

propagation time delay is assumed in the (S) mode.

3. Simulated signal--This field determines the maximum number of the simulated signals. The "+" and "-" signs within this field are used to increase and decrease the number of the signals. Place the cursor on "+" and press (L), (M), or (R) to increase the number by units, tens, or hundreds. This has a range of 19 to 27, so (M) and (R) are not used.

4. Simulation cycles--This field operates in the same way as the simulated signal field but determines the maximum number of simulation cycles.

5. Delete--Placing cursor in this field and pressing all the buttons or [Shift Del] key deletes all the waveforms and text in the simulation data field.

6. Filter--Placing cursor in this field and pressing all the buttons or [Shift Del] key deletes all the waveforms whose signal names are not assigned.

7. Draw--This field switches operation to the drafting screen.

Utility--This field is located around the screen frame in a set of arrows. The two fields on the top and the bottom of the screen control vertical scrolling of the simulation data field, while the two on the left and the right of the screen control horizontal scrolling of the simulation data field. The remaining two fields

control scrolling of the simulation cursor. Scrolling speed is controlled by the buttons as in simulate field.

Information--This field has 5 fields.

1. Cycle--This field indicates the location of the display screen within the simulations memory (display's left margin).

2. Signal--This field shows the location of the top line of the display screen within the simulation memory.

3. Current cycle--This field indicates the current simulation cycle, that is, position of the vertical cursor.

4. Binary counter--This 10 stage binary counter is displayed above the "Simulate" window. For each simulation cycle the counter is incremented by 1. You can toggle each bit of this counter by placing cursor on it and pressing any button.

A.3.2 Simulation

Schematic design on the drafting screen can only be simulated, which means you must have your schematic design on the drafting screen at all times during simulation.

Load the schematic design shown in Figure 5 you saved as "REG.SHT", and prepare simulation screen for simulation of it. The simulation of the schematic design is shown in Figure 6.

1. Select the "Disk" menu.
2. Select the "Load sheet" option. You are asked for the file name.
3. Enter "REG". "REG.SHT" is loaded on the drafting screen.

Note: Another way to load "REG.SHT" is to select "Directory", then "Sheet" directory and desired file, and finally "Load sheet".

4. Select the "Simulate" menu. The simulation screen pops up.
5. Place cursor on the topmost dotted line of the simulation signal field and press any button. Cursor moves to the "Simulate" window on the drafting screen with a beep.
6. Place cursor in the vicinity of the input "IN" and press (L). Cursor sticks to "IN" with a beep.

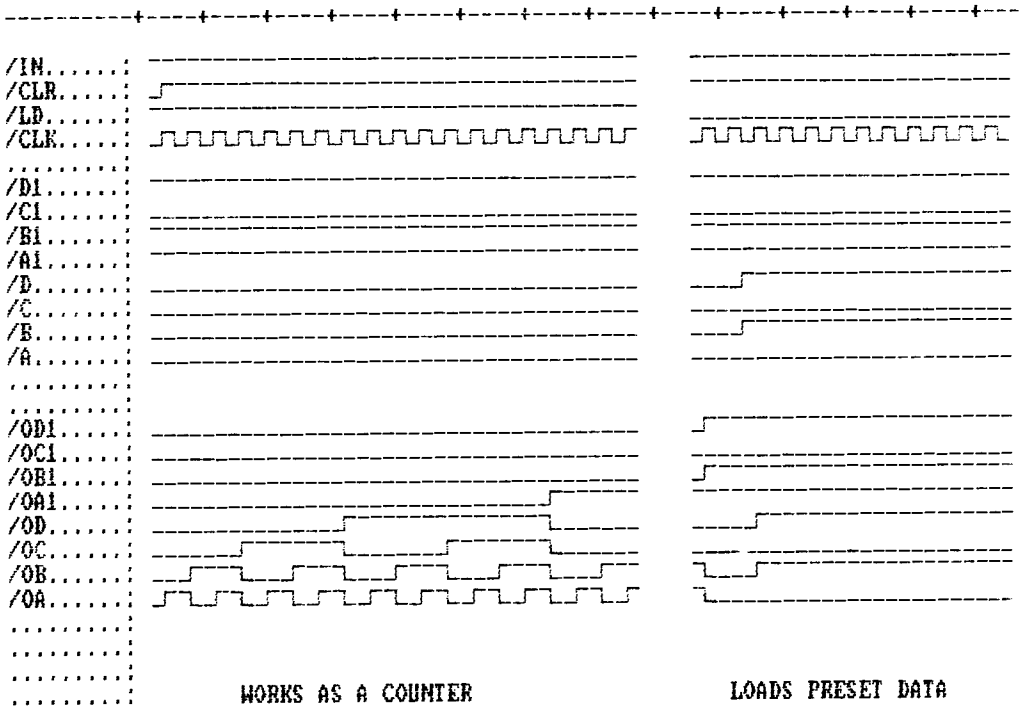


Figure 6. Simulation of MA8N

7. Press (L) again. "IN" is transferred to the simulation screen and "0" state is given to "IN".
8. In the same manner, transfer the other input signals and the eight output signals from the drafting screen as in Figure 6.
9. Locate cursor in the test vector assignment field, to the left of "IN" and press any button. Cursor snaps with a beep.
10. Enter "a" as a test vector for the "IN" signal line. With a beep "a" is entered.
11. In like manner, enter the other test vectors for the other input signals as in Figure 6.
12. Have "CLK" driven by the binary counter.
13. Press the assigned keys to toggle the associated signals to the logic "1" state. The signal "CLR" must be toggled to the logic "1" at the second cycle.
14. Place cursor in "Simulate" field and press (L) to start simulation. The long vertical simulation cursor in the simulation data field moves one step to the right.
15. Press the associated key to toggle the "CLR" signal to the logic "1" state.
16. Press (L), (M), or (R) for one step, slow speed, or high speed simulation, respectively.

17. Place cursor in the "Mode" field and press any button for simulation in the "S" mode.
18. To write text, place cursor where you want and press (L) or (R). A text cursor appears
19. Press RETURN key.
20. Enter the desired text.
21. Place cursor in the "Draw" field and press any button. You are back to the drafting screen.
22. Select the "Disk" menu.
23. Select the "Save timing" option. You are asked for the file name.
24. Enter "REG".
25. Select the "Disk" menu again to check if the file is saved as "REG.TIM".
26. Select the "Directory" option. You get four options
27. Select "Timing". The file "REG.TIM" you just saved is one of the timing files in the directory.

APPENDIX B

DETAILS OF A SIMPLE COMPUTER DESIGN AND SIMULATION

B.1 Computer Components

The following devices are chosen from the SLAV TTL and Memory libraries in the design of the simple computer:

CONTQ0:	Control Unit
PC161N:	Program Counter
MA8N:	Memory Address Register
	Memory Buffer Register
	Accumulator
MUX8:	Multiplexer
ALU:	Arithmetic and Logic Unit
M6164:	Memory Unit
74161:	Instruction Register
74163:	Timing Counter

The CONTQ0 used as the control unit is made up of many ANDs, ORs and inverters. PC161N is made out of two 74161 4-bit binary counters to work as an 8-bit counter, while MA8N is made out of the same two 74161 to work as an 8-bit register. MUX8 consists of two 74157 Qaud 2-Input Data Selector to operate in 8 bits. Two 74181 4-

bit arithmetic and logic unit are combined into one 8-bit unit.

Any schematic designed under SLAV can be converted into a netlist file using the "Netlist" option in the "Disk" menu. By compiling the netlist file under ALC, you can have a single chip. It should be noted that a netlist file has a ".src" extension. It also should be noted that the file contains the custom-made chip with a ".ald" extension after it is compiled by ALC. You can find the procedure of creating your own chip by generating a netlist file and compiling it under ALC in Section 2.3 of Appendix A.

For references, the sheet files of the schematic drawings of the aforementioned chips, CONTQ0, MA8N, PC161N, MUX8, and ALU are provided in a separate diskette along with the associated netlist files and timing files. The sheet files and the timing files have an extensions of ".sht" and ".tim" respectively.

B.2 Simulation Results

Simulation has been done by the registers to see if the computer works properly as expected. The following three operations are used in simulation:

```
START:    LDA  1011
```

```
            ADD  1100
            JMP  START
1011:       0100          : Data1=4
1100:       0101          : Data2=5
END
```

It starts from memory address 00000000, where the instruction code, 00000000, of LDA (Q0=1) is stored. Data 4 and 5 are stored in memory addresses 00001011 and 00001100 respectively. The contents of memory to execute the above three operations are as follows:

Memory Address	Memory Contents	
00000000	00000000	:LDA Code, Q0
00000001	00001011	:Address of 4
00000010	00000001	:ADD Code, Q1
00000011	00001100	:Address of 5
00000100	00000010	:JMP Code, Q2
00000101	00000000	:Address, JMP
.	.	.
.	.	.
00001011	00000100	:Data 4
00001100	00000101	:Data 5

The contents of each register at each time slot are listed in Figure 7. The results of simulation from the

	PC	MA	MB	IR	AC	TC
T0	00000000	00000000	00000000	0000	00000000	000
T1	00000001	00000000	00000000	0000	00000000	001
T2	00000001	00000000	00000000	0000	00000000	010
Q0T3	00000001	00000000	00000000	0000	00000000	011
Q0T4	00000010	00000001	00000000	0000	00000000	100
Q0T5	00000010	00000001	00001011	0000	00000000	101
Q0T6	00000010	00001011	00001011	0000	00000000	110
Q0T7	00000010	00001011	00000100	0000	00000000	111
T0	00000010	00001011	00000100	0000	00000100	000
T1	00000011	00000010	00000100	0000	00000100	001
T2	00000011	00000010	00000001	0000	00000100	010
Q1T3	00000011	00000010	00000001	0001	00000100	011
Q1T4	00000100	00000011	00000001	0001	00000100	100
Q1T5	00000100	00000011	00001100	0001	00000100	101
Q1T6	00000100	00001100	00001100	0001	00000100	110
Q1T7	00000100	00001100	00000101	0001	00000100	111
T0	00000100	00001100	00000101	0001	00001001	000
T1	00000101	00000100	00000101	0001	00001001	001
T2	00000101	00000100	00000010	0001	00001001	010
Q2T3	00000101	00000100	00000010	0010	00001001	011
Q2T4	00000110	00000101	00000010	0010	00001001	100
Q2T5	00000110	00000101	00000000	0010	00001001	101
Q2T6	00000000	00000000	00000000	0010	00001001	110

Figure 7. Contents of Registers

SLAV simulation screen are shown in Figure 8. Only 28 input and output signals can be monitored on the simulation screen at a time. The four most significant bits of MA, MB and AC remain low. So only the four least significant bits are monitored. With PC, IR and TC, only three least significant bits change and are monitored. The horizontal blank in Figure 8 separates the fetch cycle from the execution cycle.

The separate diskette contains the following files:

ALU.SHT:	ALU single chip design schematic (two 74181 ALU chips)
ALU.TIM:	Simulation timing diagram of ALU
ALU.SRC:	Netlist file for ALU
ALU.ALD:	Executable file of ALU.SRC
CONTQ0.SHT:	Control unit design schematic
CONTQ0.SRC:	Netlist file for control unit
CONTQ0.ALD:	Executable file of CONTQ0.SRC
MA8N.SHT:	8-bit register design schematic (two 74161 4-bit binary counter)
MA8N.TIM:	Simulation timing diagram of MA8N
MA8N.SRC:	Netlist file for MA8N
MA8N.ALD:	Executable file of MA8N.SRC
MUX8.SHT:	Multiplexer single chip design schematic (two 74157 quadruple 2-to-1 multiplexer)

MUX8.TIM:	Simulation timing diagram of MUX8
MUX8.SRC:	Netlist file for MUX8
MUX8.ALD:	Executable file of MUX8.SRC
PC161N.SHT:	8-bit counter design schematic (two 74161 4-bit binary counter)
PC161N.TIM:	Simulation timing diagram of PC161N
PC161N.SRC:	Netlist file for PC161N
PC161N.ALD:	Executable file of PC161N
REG.SHT:	Same as MA8N.SHT
REG.TIM:	Same as MA8N.TIM
REG.SRC:	Same as MA8N.SRC
REG.ALD:	Same as MA8N.ALD
SMLEES.SHT:	Whole computer design schematic
SMLEES.TIM:	Simulation timing diagram of the computer

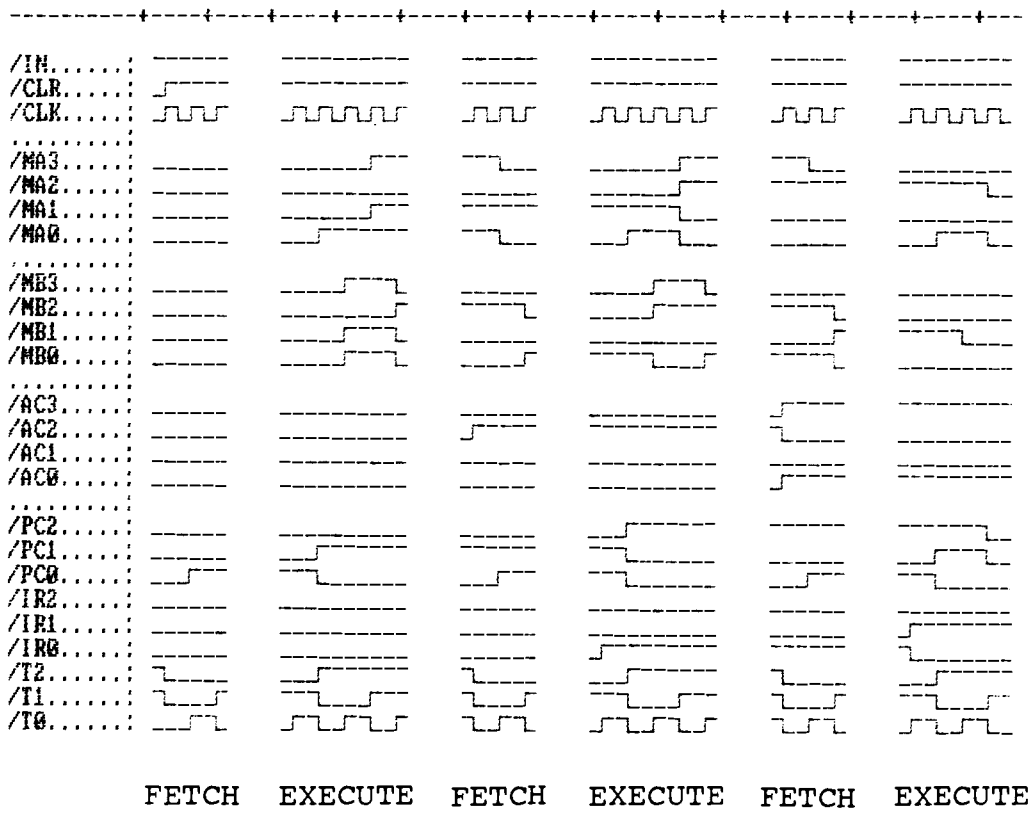


Figure 8. Results of Simple Computer Simulation