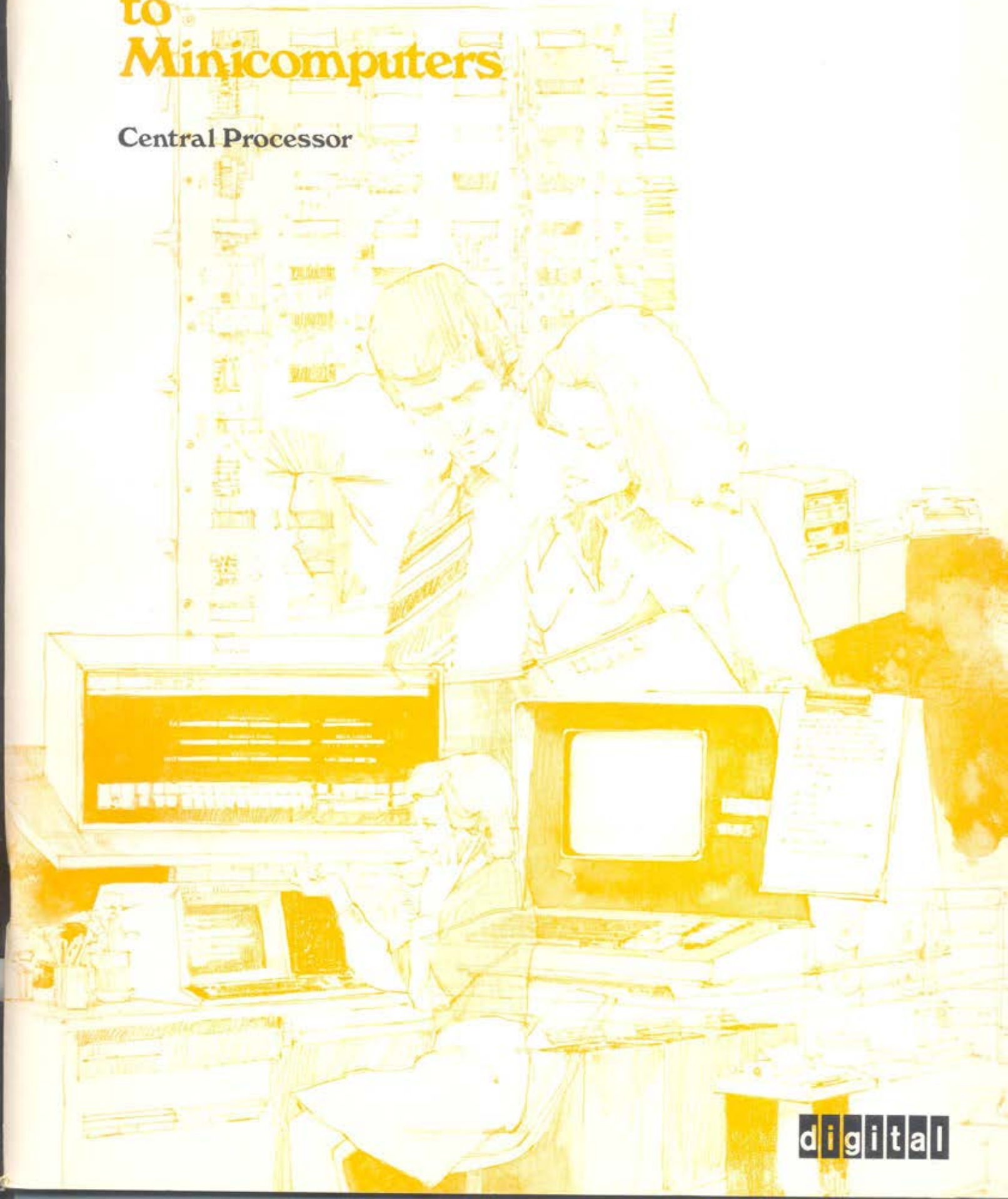
Introduction to Minicomputers

Central Processor



digital

1st Printing, June 1976
2nd Printing (Rev), October 1977
3rd Printing, August 1979

Printed in U.S.A.

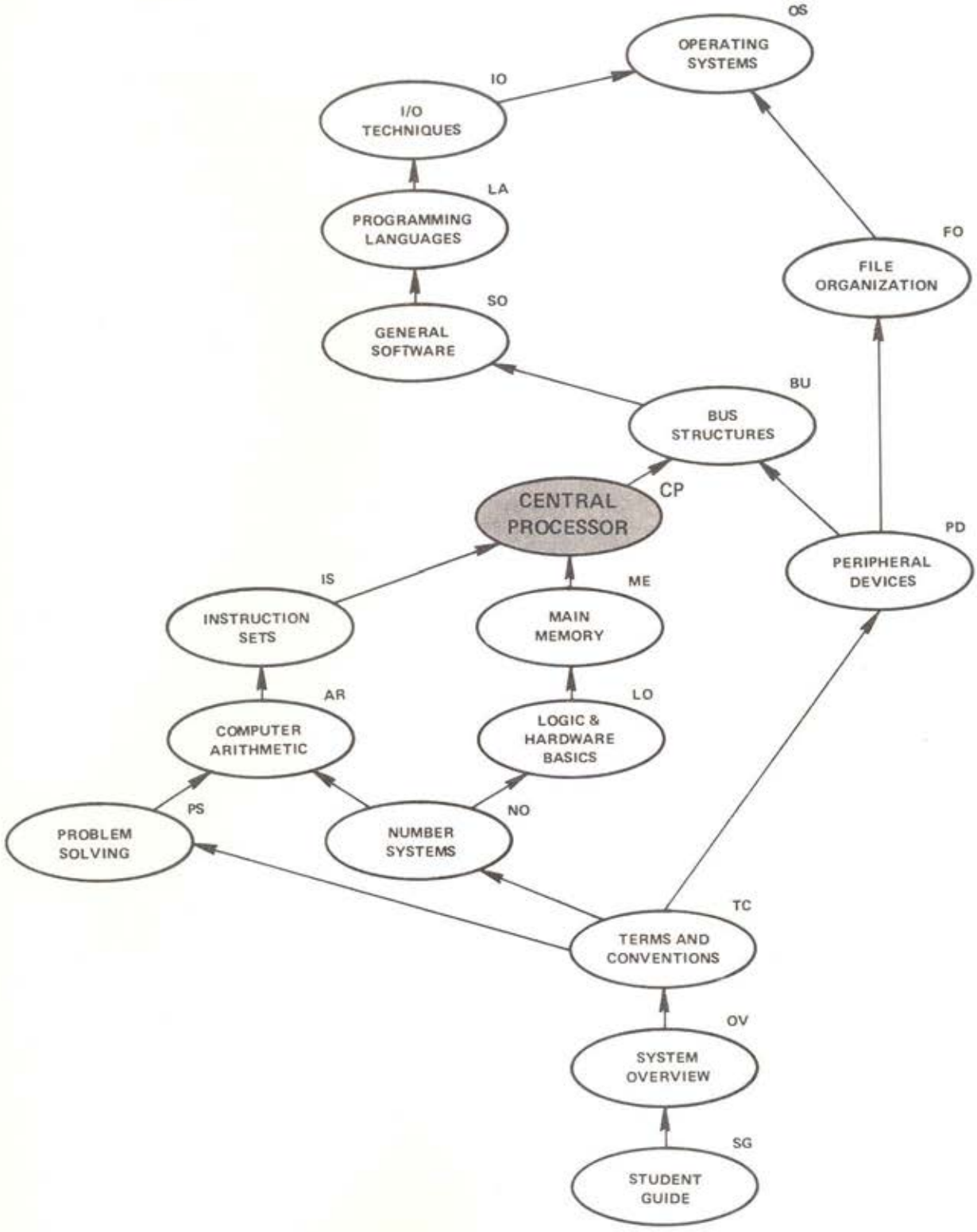
INTRODUCTION TO MINICOMPUTERS

Central Processor

Student Workbook

Audio-Visual Course by Digital Equipment Corporation

COURSE MAP


OS
OPERATING SYSTEMS
IO
I/O TECHNIQUES
LA
PROGRAMMING LANGUAGES
SO
GENERAL SOFTWARE
FO
FILE ORGANIZATION
BU
BUS STRUCTURES
CP
CENTRAL PROCESSOR
PD
PERIPHERAL DEVICES
IS
INSTRUCTION SETS
ME
MAIN MEMORY
AR
COMPUTER ARITHMETIC
LO
LOGIC & HARDWARE BASICS
PS
PROBLEM SOLVING
NO
NUMBER SYSTEMS
TC
TERMS AND CONVENTIONS
OV
SYSTEM OVERVIEW
SG
STUDENT GUIDE

CONTENTS

Central Processor

Introduction

Before a computer can solve a problem, it must be given a sequence of instructions (i.e., a program) that tells it precisely what operations to perform and where to find the data to be operated upon. Once the program is stored in main memory, the instructions can be retrieved and executed by the computer. A specific unit is responsible for retrieving individual instructions from memory and then executing them. This unit is called the *central processor unit,* or simply the *CPU.*

The central processor is vital to the operation of the entire computer system. It controls and supervises operations involving the other system units per the instructions retrieved from main memory. The CPU also executes the decision-making functions that are called for by the program.

In addition to its control and decision-making functions, the central processor performs all of the arithmetic computations. Basic operations, such as binary addition and complementing numbers, are handled by special circuits that are part of the CPU.

In this module we will explain how the central processor retrieves and executes instructions. The first lesson in this module identifies the major *functional components* of the CPU and explains the purpose of each component. The second lesson describes the individual operations that the central processor performs each time it retrieves and executes an instruction. This sequence of operations is called an *instruction cycle.*

The third lesson in this module discusses some of the uses of the computer *console.* It explains how computer operations can be initiated, controlled, and monitored using switches and lamps that are provided on the computer console.

Components of a CPU

OBJECTIVES

1. Given the two major parts of a central processing unit and five statements of CPU functions, be able to match each statement with the part it describes.

2. Given the eight components of a typical central processing unit and a list of eight functions, be able to match each component with its function.

SAMPLE TEST ITEMS

1. Indicate that each of the functions below refers to the control unit (C) or the arithmetic-logic unit (A) of the CPU by writing the correct letter in the space provided.

Function	Unit of CPU
Executes all computations.	_____
Decodes each instruction and generates the signals to start the specific action.	_____
.	.
.	.
.	.

SAMPLE TEST ITEMS

2. Match each of the following CPU components with its function.

Component	Function
Accumulator	_____
Address Register	_____
.	.
.	.
.	.

Functions

a. A register in the ALU that is used as a working area for computations.

b. A register in the CU that is used to hold the address of the memory location currently being referenced by the CPU.

.
.
.

Mark your place in this workbook and view Lesson 1 in the A/V program, "Central Processor."

The central processor is made up of two major parts – the *control unit* (CU) and the *arithmetic-logic unit* (ALU). The control unit is the "decision maker." It coordinates and directs the activities of the entire computer system; its specific functions include:

- the location and retrieval of instructions from memory, one at a time,
- the decoding of each instruction and the generation of control signals to start the specified action (for example, an input/output operation, a memory store or fetch operation, or an arithmetic operation), and
- the direction and control of data movements among the CPU, memory, and input/output devices.

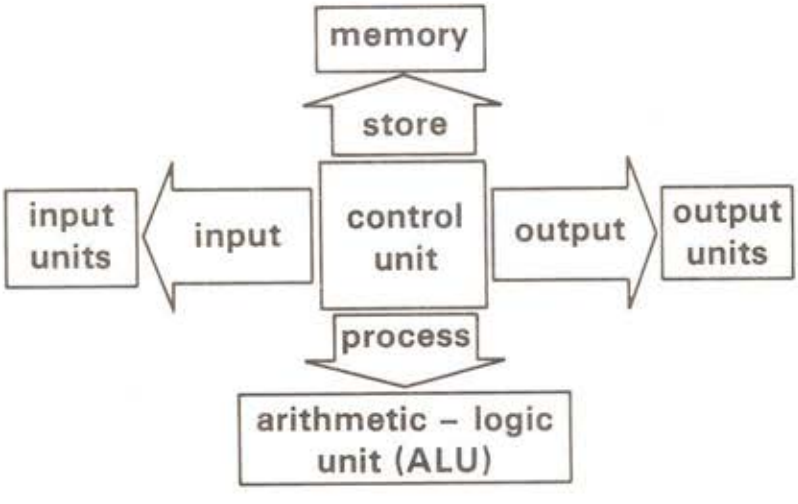


Figure 1 Control Unit Functions

The arithmetic-logic unit has two primary functions:

- the performance of all arithmetic computations, and
- the performance of logical tests, such as the comparison of two values or a test for a zero value.

An *arithmetic computation,* then, produces a *result,* while a *logical operation* is used to make a *decision.* Note that many ALUs reduce all arithmetic computations to a series of addition and complementary operations. It is also important to remember that all operations of the ALU are directed by signals from the control unit.

There are five major registers within the ALU and the control unit (see Figure 2):

Instruction Register (IR) Program Counter (PC) Address Register (AR)	Control Unit
Buffer Register (BR) Accumulator (AC)	ALU

Typically, the IR, PC, and AR are part of the control unit; the BR and AC are part of the ALU. These registers are used for storing small amounts of information – usually one computer word.

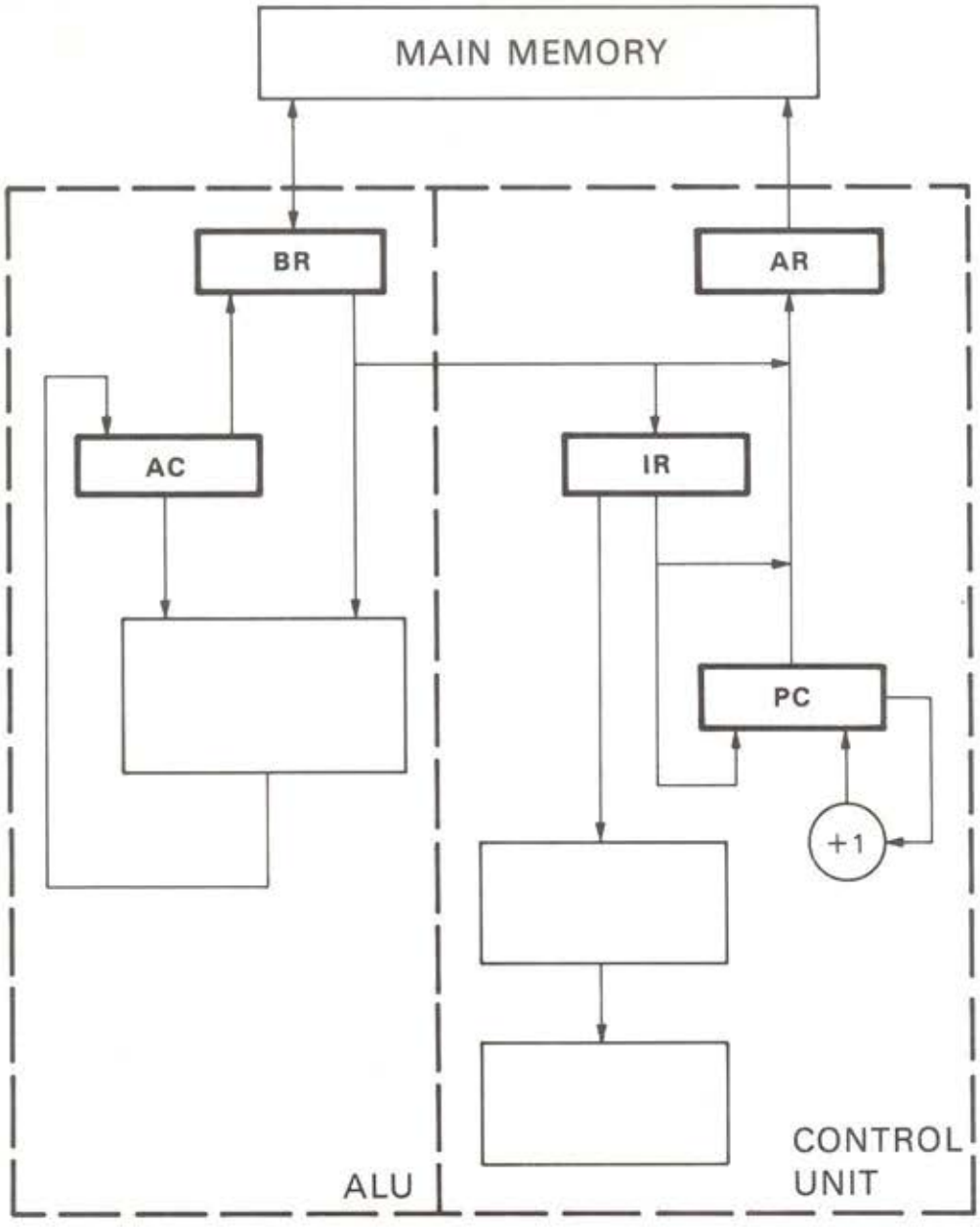


Figure 2 Major CPU Registers

Buffer Register (BR)

The buffer register (BR) is used to store information temporarily (an instruction or a word of data) that is being fetched from or stored into main memory. Thus, the BR is a stopover point between the memory and the other components of the CPU. It is necessary to use a buffer register to compensate for the differences in operating speeds between the CPU and external units, such as memory.

Instruction Register (IR)

The second component of a CPU is the instruction register (IR). The instruction register is used to hold each instruction during the time it is being executed. As an instruction is to be executed, it is fetched from memory into the BR and then transferred to the IR. Remember that instructions are binary combinations subdivided into operation codes and operand fields. In order to execute any given instruction, the op code must be decoded so that the desired operation may be identified. A component, the *instruction decoder,* does exactly that task. (See Figure 3.) The contents of the IR are used as input for the instruction decoder. Once the op code has been decoded, a unique signal for the particular operation can be transmitted to the control logic. The identity of the operation determines whether memory references are necessary for the operation. Examination of the "I-field" determines whether direct or indirect addressing will be used.

Program Counter (PC)

Another of the five major registers is the program counter (PC). The CPU must have some means of determining which instruction is the next one to be executed. The program counter fulfills this role. Before a program can be executed, the starting address (the location of the first instruction) must be loaded into the PC. The contents of the PC are then used as an address to fetch the first instruction from memory. As an instruction is executed, the PC is automatically updated so that it always contains the address of the next instruction to be executed. Because most instructions are executed sequentially, this updating process is usually a simple incrementing of the PC by one (Figure 3).

However, when a jump (JMP) instruction is encountered, the *address* of the next instruction is taken from the *operand field* in the IR. Thus, the next instruction retrieved from memory will be the one specified by the JMP instruction instead of the instruction at the next sequential location in memory.

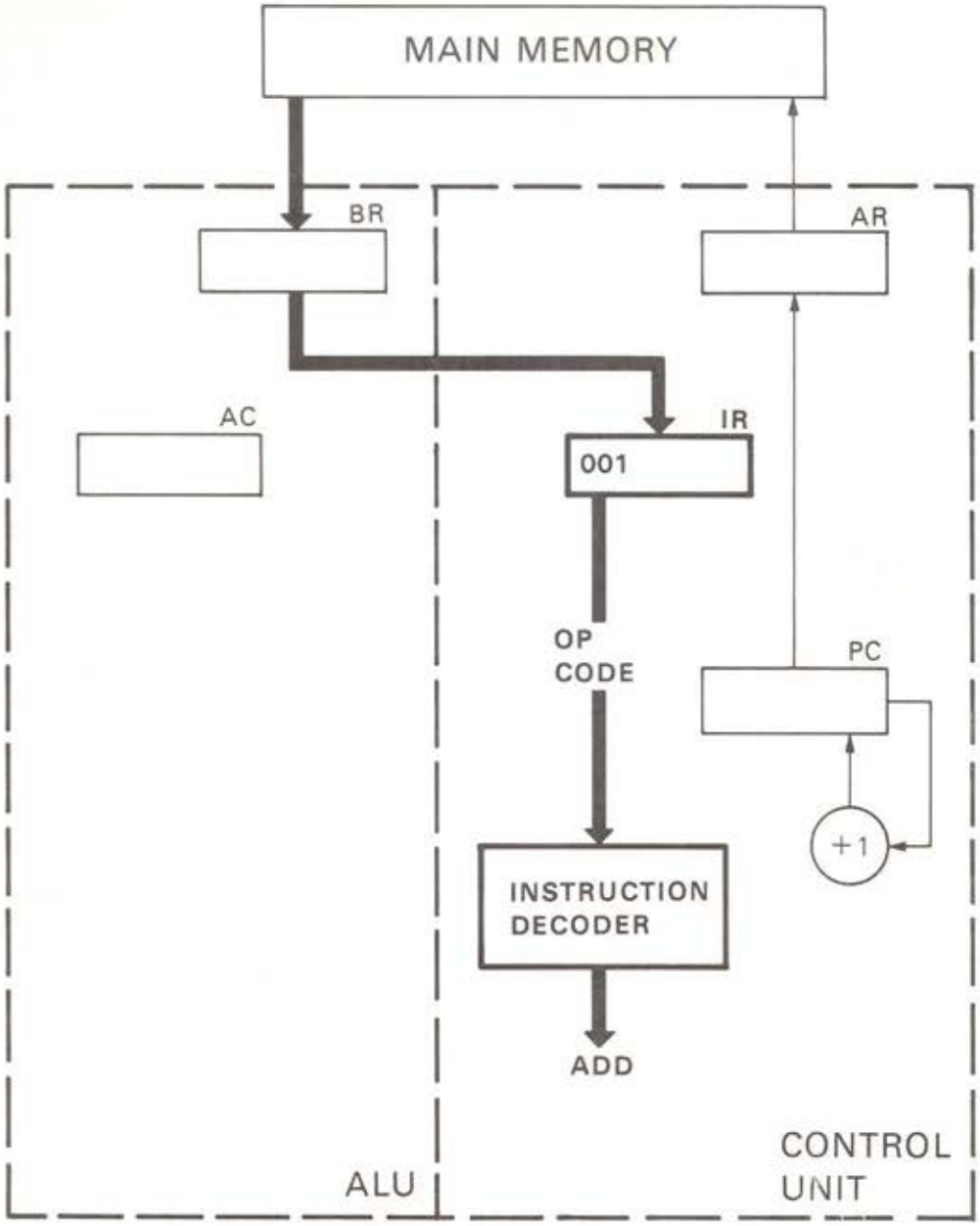


Figure 3 Instruction Register (IR) and Decoder

Address Register (AR)

The address register (AR) is used to hold the address of the memory location being referenced by the CPU. Because of the difference in operating speeds between memory and the CPU, the AR is used to hold the address information until it can be accepted by main memory. Thus, the AR and the BR are used when the CPU stores information in or retrieves information from memory.

Accumulator (AC)

The accumulator (AC) functions as a working area for all computations performed by the ALU (Figure 4). For example, before an addition is performed, one of the operands is temporarily placed in the AC. The second operand is then retrieved from memory and is added to the contents of the AC. The resulting sum appears in the AC, destroying the original contents of the AC. Some instructions, such as "Clear Accumulator" or "Complement Accumulator," operate directly on the contents of the AC.

Some central processors may have more than one accumulator. In other processors the functions normally provided by the AC are handled by general purpose registers (GPRs).

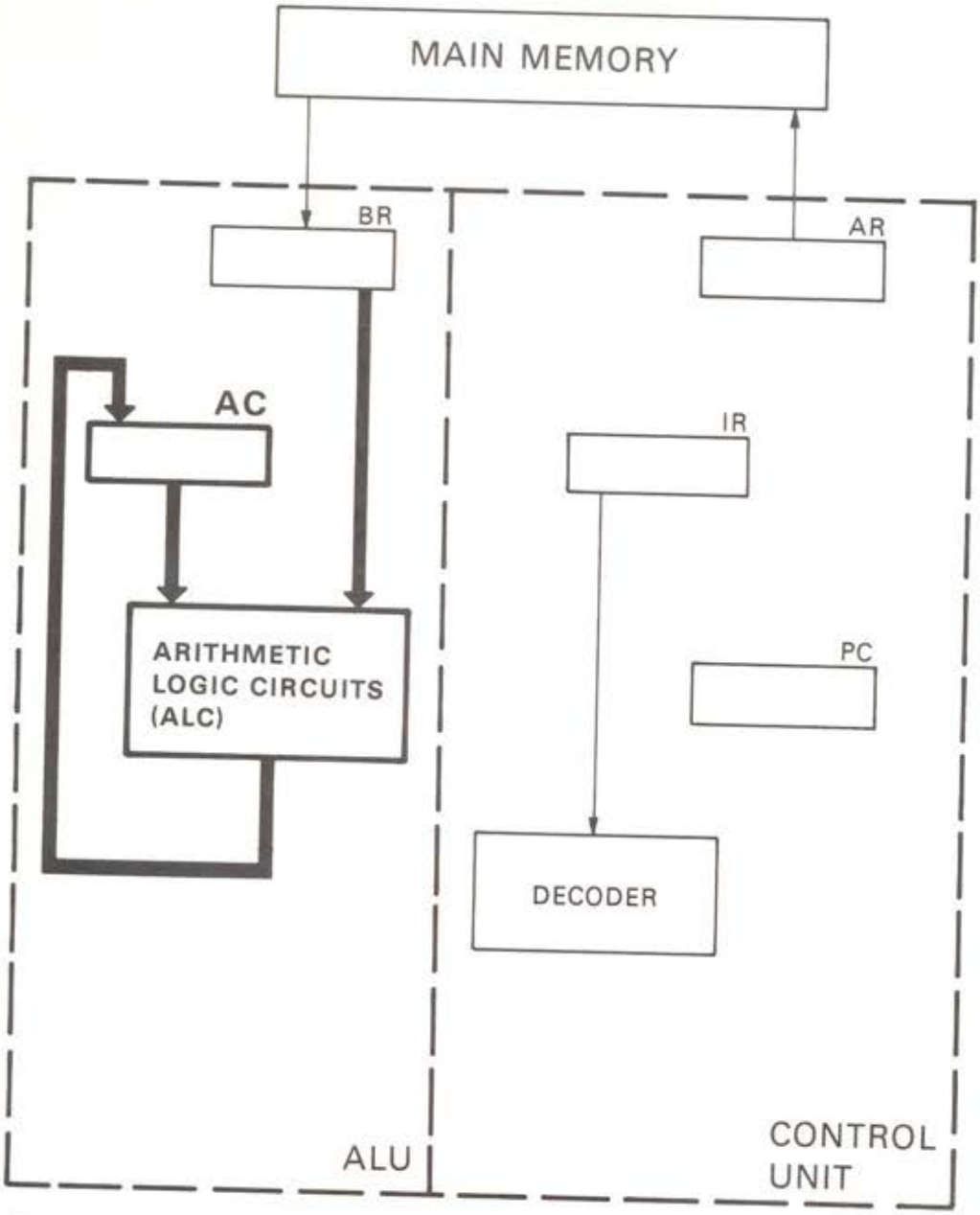


Figure 4 Accumulator (AC) and Arithmetic-Logic Circuits (ALCs)

Control Logic

The central processor consists of five major types of registers; each type of register performs a specific function. How is information routed from one register to another? This function is handled by a special group of logic circuits called the control logic.

The control logic establishes necessary electrical paths so that information can be transferred between registers. These electrical paths are often referred to as "data paths." Figure 5 illustrates typical data paths established by the control logic.

In conclusion, you have learned about the components that make up the control unit and arithmetic-logic unit of a typical central processor unit (CPU). Five major registers contain various items of information required by the CPU in the performance of its tasks. Two components, the arithmetic-logic circuits and the instruction decoder, operate upon information contained in the registers. Finally, the control logic establishes the data paths between the registers.

The next lesson discusses how the components are used during a typical instruction cycle and should reinforce your understanding of the material in this lesson.

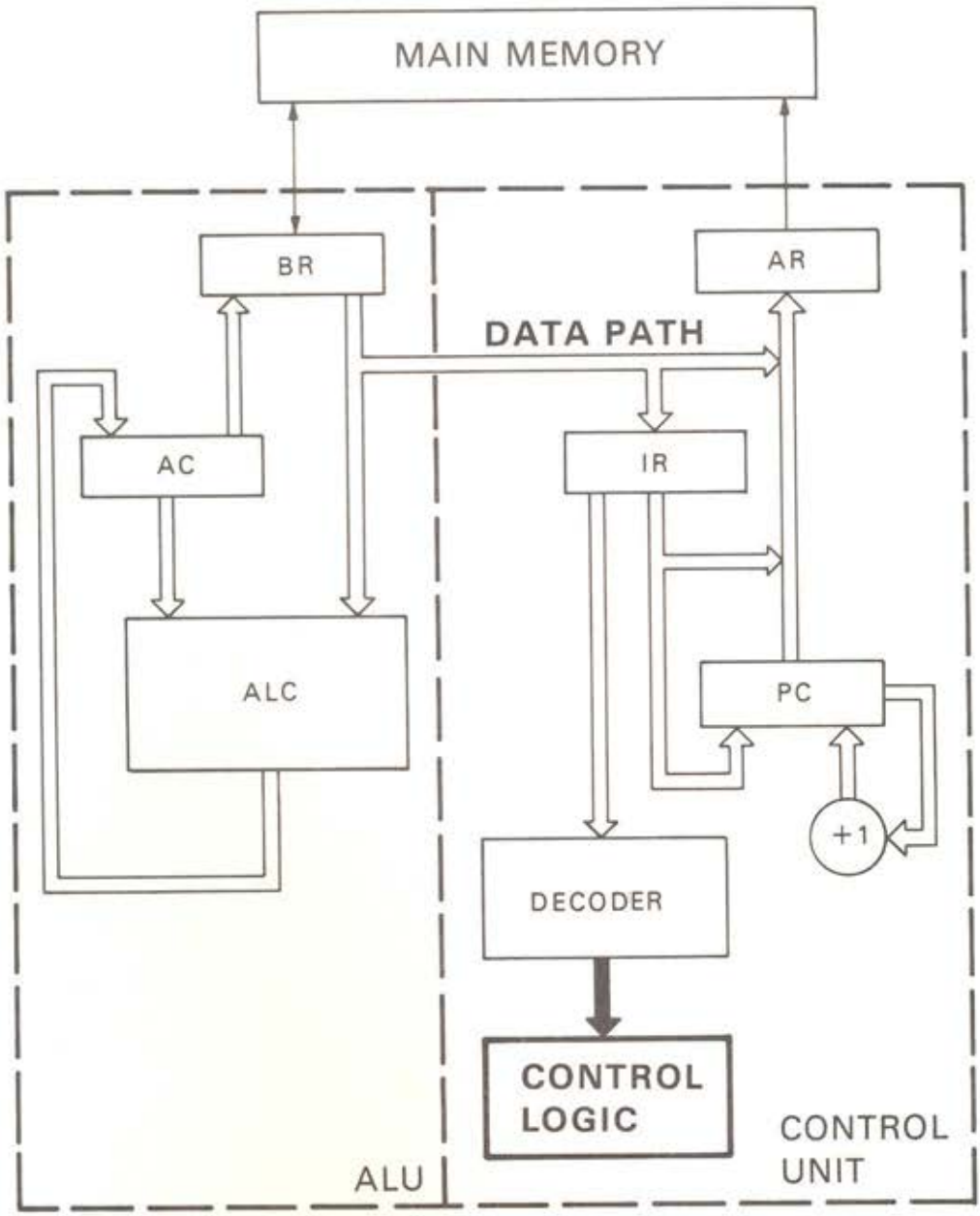


Figure 5 Control Logic and Data Paths

EXERCISES

1. List the two major parts of the CPU and explain their functions.

2. List at least six components of a typical CPU and give a function of each.

SOLUTIONS

1. List the two major parts of the CPU and explain their functions.

   Control Unit – The CU is responsible for fetching and decoding instructions and for controlling data movements throughout the computer system.

   Arithmetic-Logic Unit – The ALU is responsible for the performance of all arithmetic calculations and logical tests (such as comparing values or testing a location for zero value).

2. List at least six components of a typical CPU and give a function of each.

   Components of a typical CPU include:

   Accumulator – A register in the ALU that is used as a working area for computations.

   Address Register – A register in the CU that is used to hold the address of the memory location currently being referenced by the CPU.

   Arithmetic-Logic Circuits – The component in the ALU that performs the actual calculations and tests through the use of adders, shift registers, and other circuits.

   Buffer Register – A register in the CPU that is used to hold one instruction or one word of data during a store or fetch operation involving memory.

   Control Logic – The component in the CU that is responsible for switching the data paths between CPU registers and for triggering components (such as the arithmetic-logic circuits) when information is ready for them.

   Instruction Decoder – The component of the CU responsible for decoding the op code of an instruction into a unique signal to the control logic.

   Program Counter – A register in the CU that indicates which instruction is the next to be executed.

   Instruction Register – A register in the CU that holds an instruction while it is being decoded and executed.

Typical Instruction Cycles

OBJECTIVES

1. Given a complete flowchart and a specific instruction, be able to label those flowchart steps that are included in the "instruction cycle" of the specified instruction.

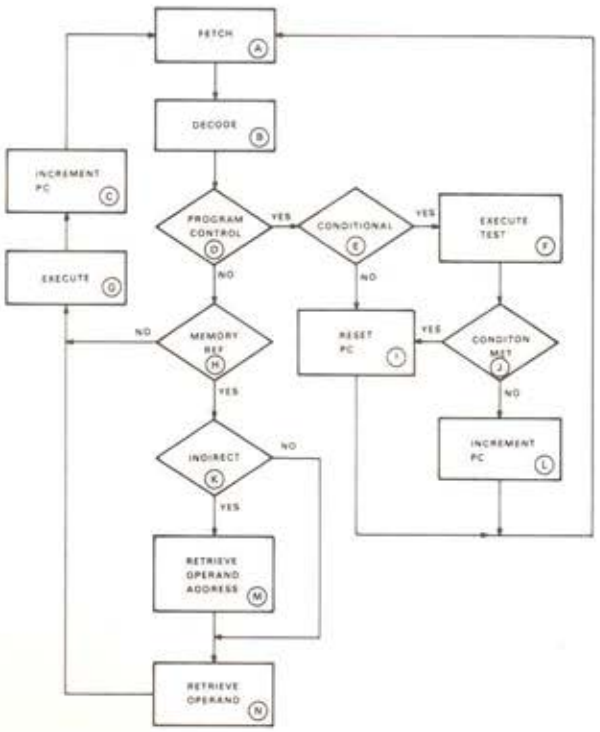
2. Given a simple program of seven instruction cycles and a table listing the five major central processing unit registers, be able to write the contents of each of the five registers for each of the seven instruction cycles.

3. Given three descriptive statements, be able to label those statements that refer to synchronous computers and those that refer to asynchronous computers.

SAMPLE TEST ITEMS

1. Referring to the flowchart below, write the *letters* of *all* the flowchart steps that are included in the instruction cycle of the instruction: ADD I 300. *Place your answers in alphabetical order*!

ADD I 300



Answers: __ __ __ __ __ __ __ __ __ __

SAMPLE TEST ITEMS

2. The simple program at the right has been stored in the computer's memory.

   Step through this program one instruction at a time by indicating what information is contained in each of the five major CPU registers at the end of each intruction cycle. Use the table below which shows the first instruction as an example.

Program

Address	Contents	
273	CLA	
274	ADD	300
275	ADD	302
276	ADD	303
277	JMP	304
300	15	
301	222	
302	1000	
303	1	
304	STR	301
305	HLT	

REGISTERS

Program Counter		Buffer Register	Address Register	Instruction Register	Accumulator
Before Execution	After Execution				
273	274	"CLA"	273	"CLA"	0

SAMPLE TEST ITEMS

3. Indicate which of the following statements refer to synchronous (S) and those that refer to asynchronous (A) computers by writing the correct letter in the space provided.

Statement	Type of Computer
Faster than the other type because there is no waiting time between operations.	_____
When an operation is completed, it transmits a signal to immediately initiate the next operation.	_____
Each operation allotted a fixed time; next operation begun when time interval is exhausted.	_____

Now view Lesson 2 of the A/V program, "Central Processor."

Each time the central processor retrieves and executes an instruction, it performs a sequence of operations called an *instruction cycle*. The operations that make up an instruction cycle depend on the type of instruction that is being executed. For example, the instruction cycle for a memory reference instruction (ADD 210) is *not* the same as the instruction cycle for a non-memory reference instruction (CLA).

In this lesson we will describe typical instruction cycles for three types of instructions:

- Non-memory reference instructions (CLA)
- Memory reference instructions (ADD)
- Program control instructions (JMP)

Non-Memory Reference Instructions

An instruction cycle for a non-memory reference instruction is shown in Figure 6. First, the CPU *fetches* the instruction from main memory. Next, the CPU *decodes* the instruction to determine what operations are to take place. The CPU can then *execute* the instruction. In the case of a non-memory reference instruction, the execute phase usually involves the contents of the *accumulator;* that is, during the execute phase, the contents of the AC may be cleared to all zeros or converted to the *one's complement.* While the instruction is being executed, the CPU increments the address contained in the PC, so it points to the next instruction to be fetched.

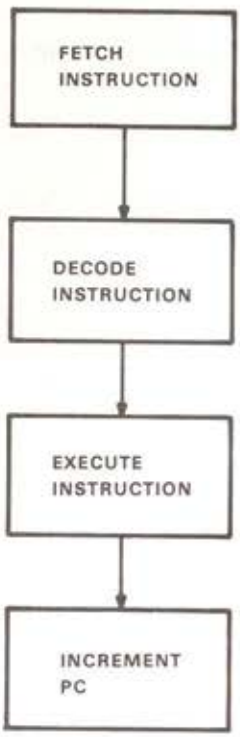


Figure 6 Instruction Cycle: Non-Memory Reference Instruction

In the previous lesson, we developed a block diagram of the CPU (Figure 7). We will now use this diagram to illustrate the actions that take place in the CPU during the four phases of the instruction cycle – fetch, decode, execute, and increment PC.

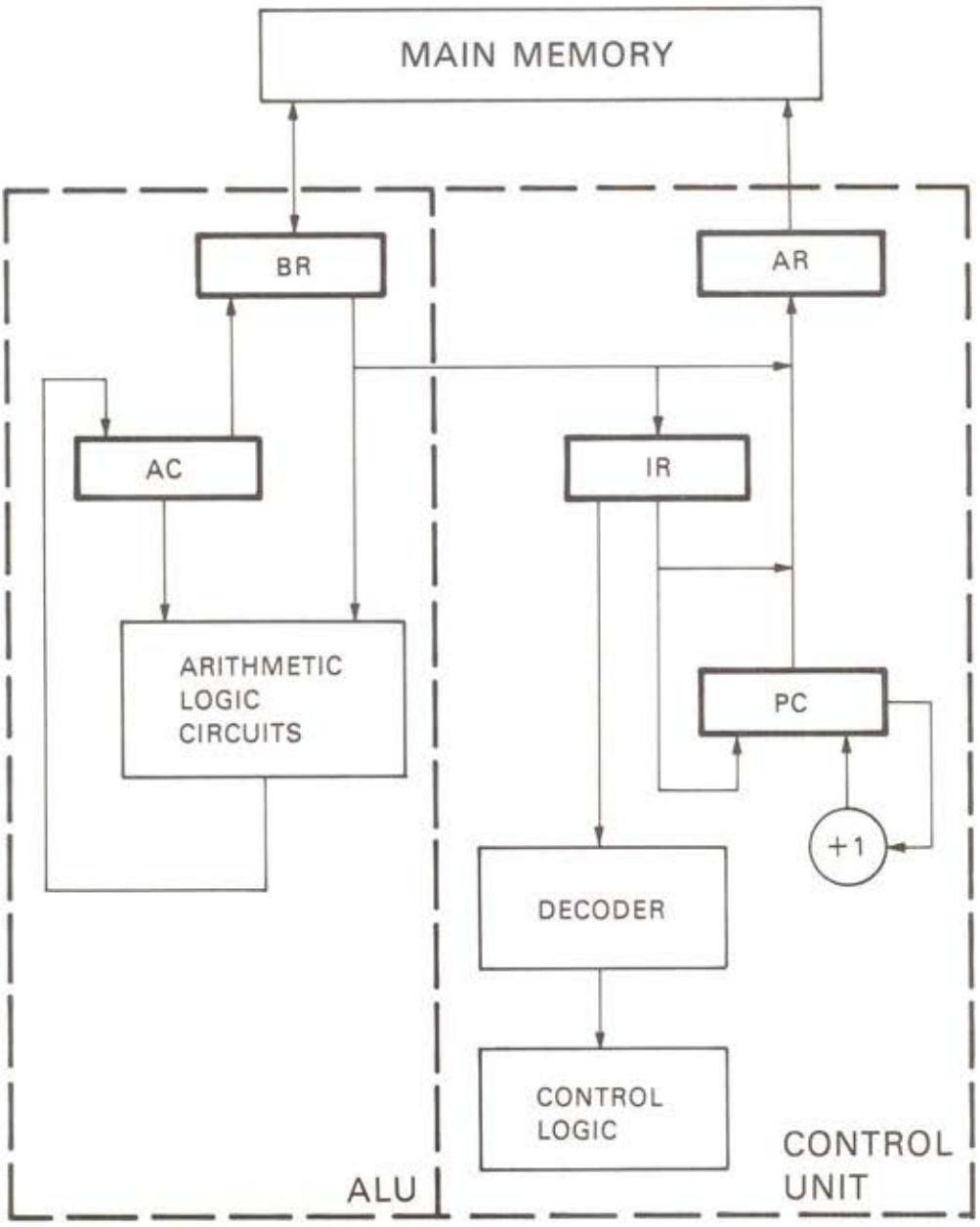


Figure 7 Block Diagram of CPU and Main Memory

Fetch Phase

Figure 8 illustrates the first phase (fetch) in the instruction cycle for a non-memory reference instruction (CLA). In this example, the CLA instruction is stored in memory location 200, and the program counter (PC) has been previously set to address 200.

The following events take place in the central processor during the fetch phase:

(a.) The control logic establishes a data path between the PC and the AR.

(b.) The address of the CLA instruction (200) is transferred to main memory by way of the AR.

(c.) The CLA instruction is retrieved from memory location 200 and is temporarily placed in the BR.

(d.) The control logic establishes another data path between the BR and IR.

(e.) The CLA instruction is transferred to the IR where it can be decoded.

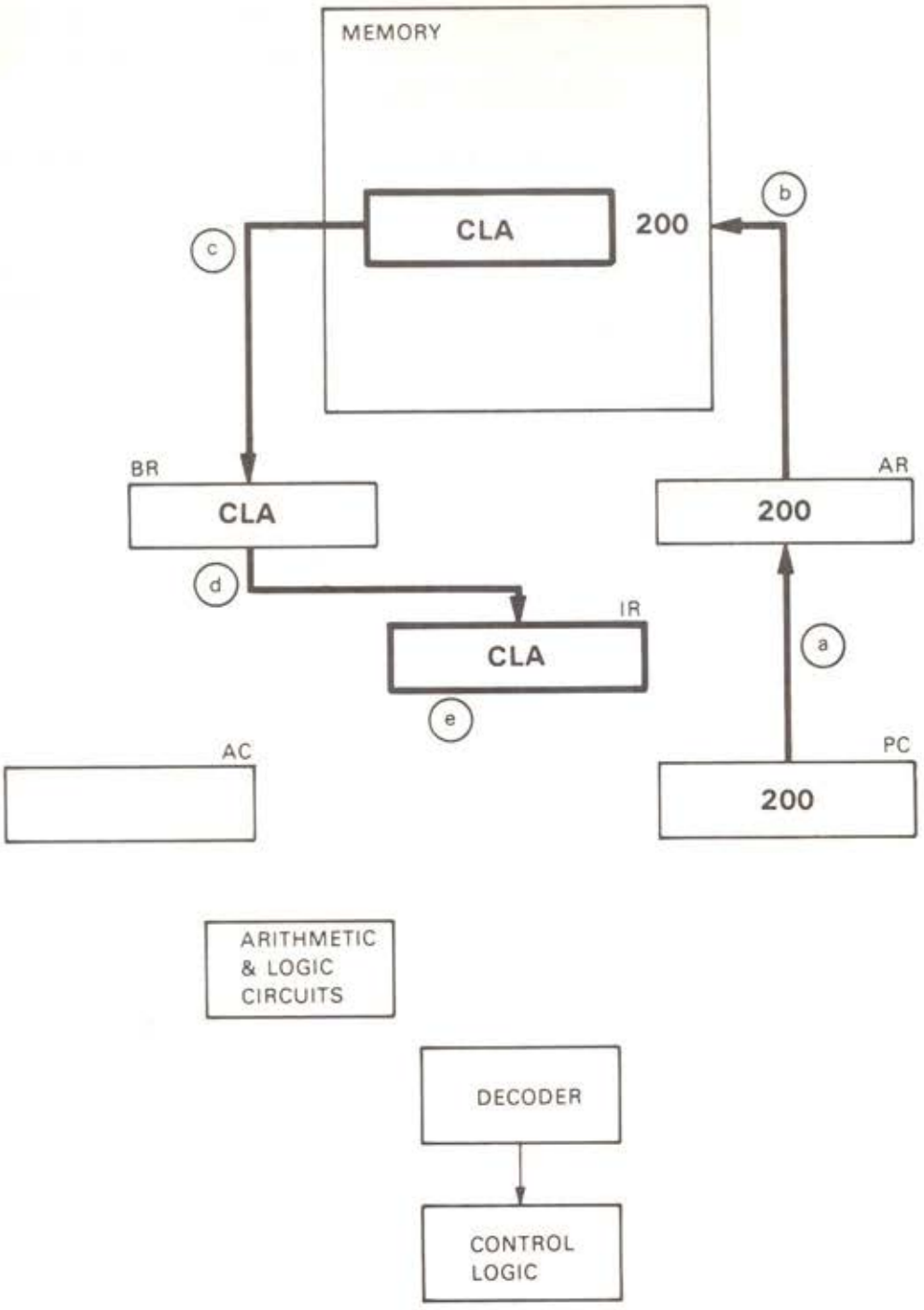


Figure 8 Fetch Phase: CLA Instruction

Decode Phase

During the decode phase, the CPU examines the contents of its IR to determine what type of instruction is to be executed. Figure 9 illustrates the events that take place during this decode phase:

(a.) Op code 111 010 000 000 (the op code for the CLA instruction) is routed to the instruction decoder.

(b.) The decoder converts the binary op code into a unique signal that is applied to the control logic. This signal tells the control logic to initiate a "Clear AC" operation during the execute phase.

Thus, the instruction decoder and the control logic allow the CPU to identify the instruction stored in the IR and prepare to execute the instruction.

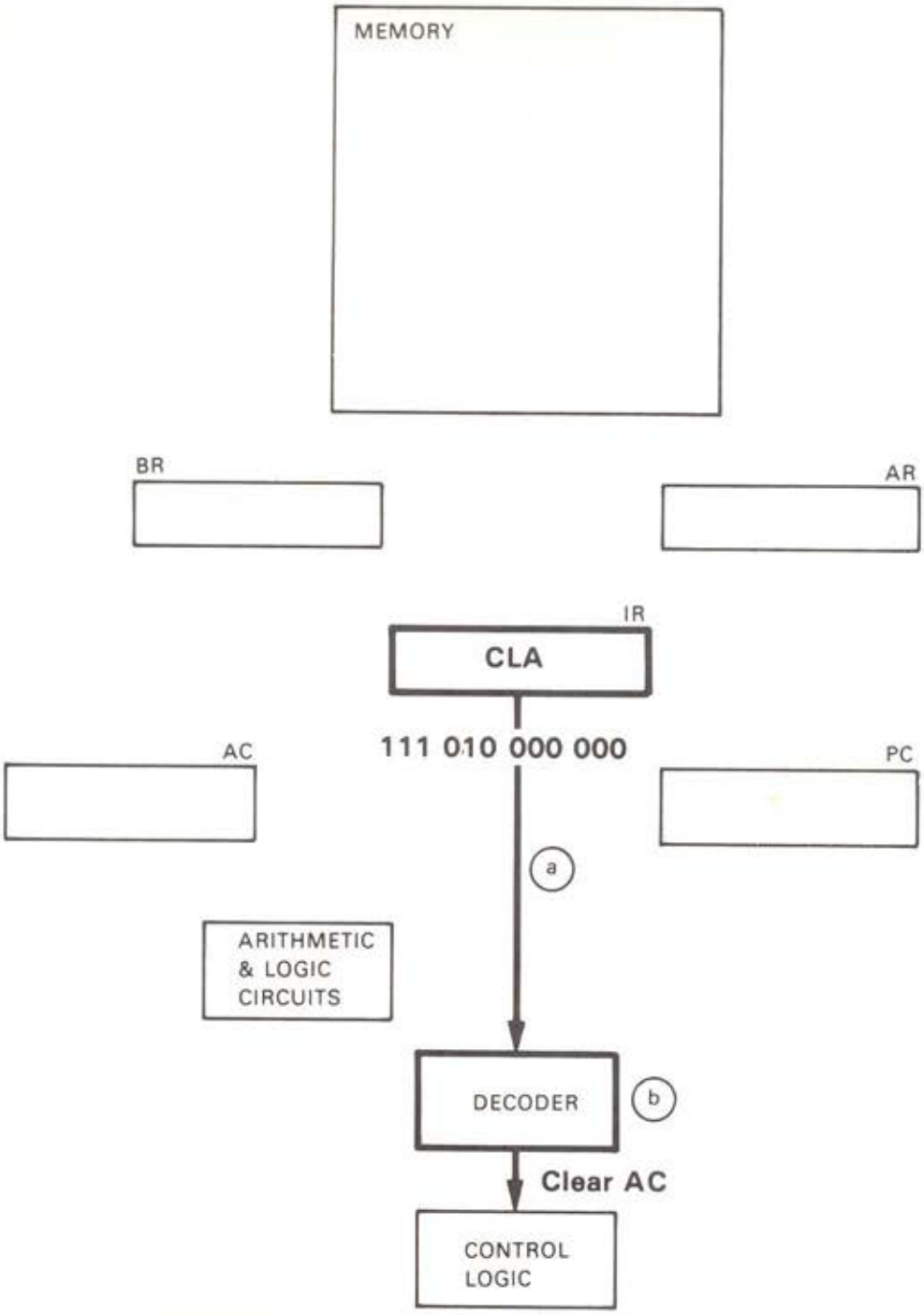


Figure 9 Decode Phase: CLA Instruction

Execute Phase

Figure 10 illustrates the major events that take place in the central processor during the execute phase:

(a.) The control logic transmits a signal to the arithmetic-logic circuits (ALC).

(b.) The ALC responds to this control signal by clearing the contents of the AC to all zeros, thereby executing the CLA instruction.

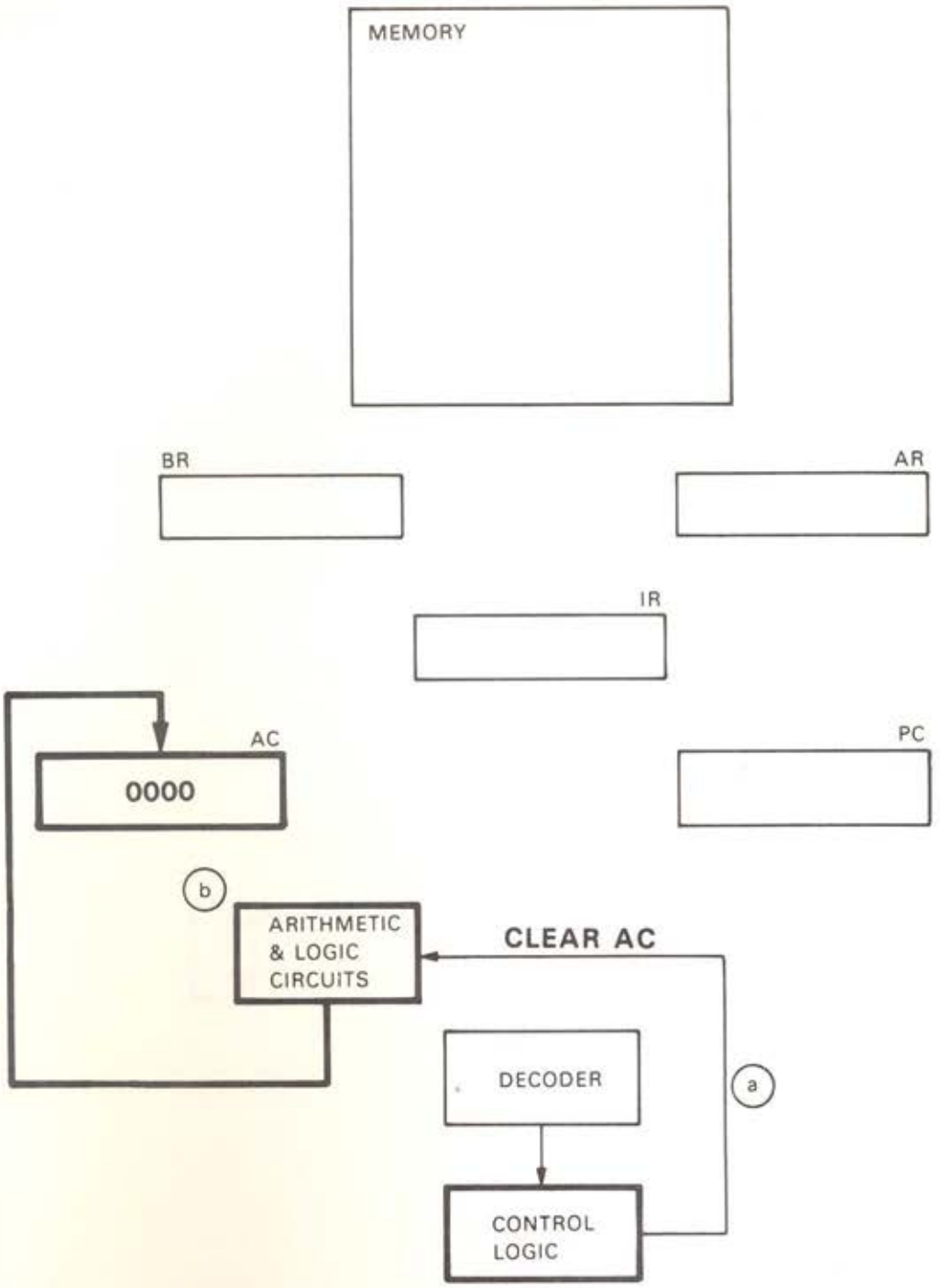


Figure 10 Execute Phase: CLA Instruction

Increment PC Phase

The final phase in this instruction cycle is to update the contents of the program counter (see Figure 11).

a. The address stored in the PC is incremented by one.

b. The new address (201) is then used to retrieve the next instruction from main memory, thus initiating another instruction cycle.

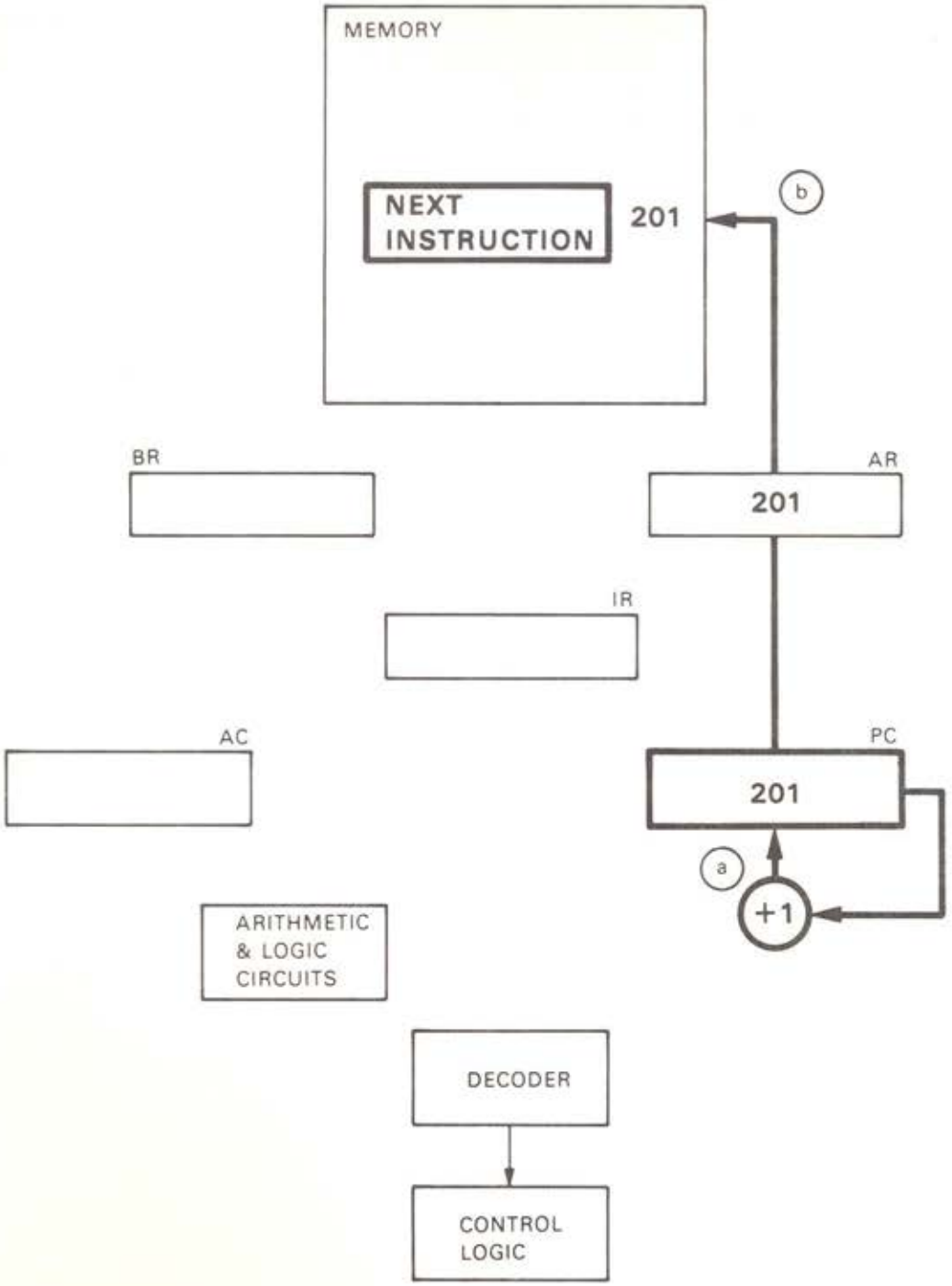


Figure 11 Increment PC Phase: CLA Instruction

Memory Reference Instructions

Figure 12 illustrates the instruction cycle for a memory reference instruction such as ADD. Note that the first two phases in the cycle (fetch and decode) are the same as for the CLA instruction that we discussed earlier. However, after the CPU decodes the instruction, it must reference memory to obtain the operand.

If the instruction calls for *direct* addressing, memory is referenced just once to retrieve the operand. On the other hand, if *indirect* addressing is used, memory must be referenced at least twice (once to retrieve the address of the operand and again to retrieve the operand).

After the CPU obtains the operand from memory, it can *execute* the instruction. While the instruction is being executed, the central processor increments the PC so that it points to the next instruction in the sequence.

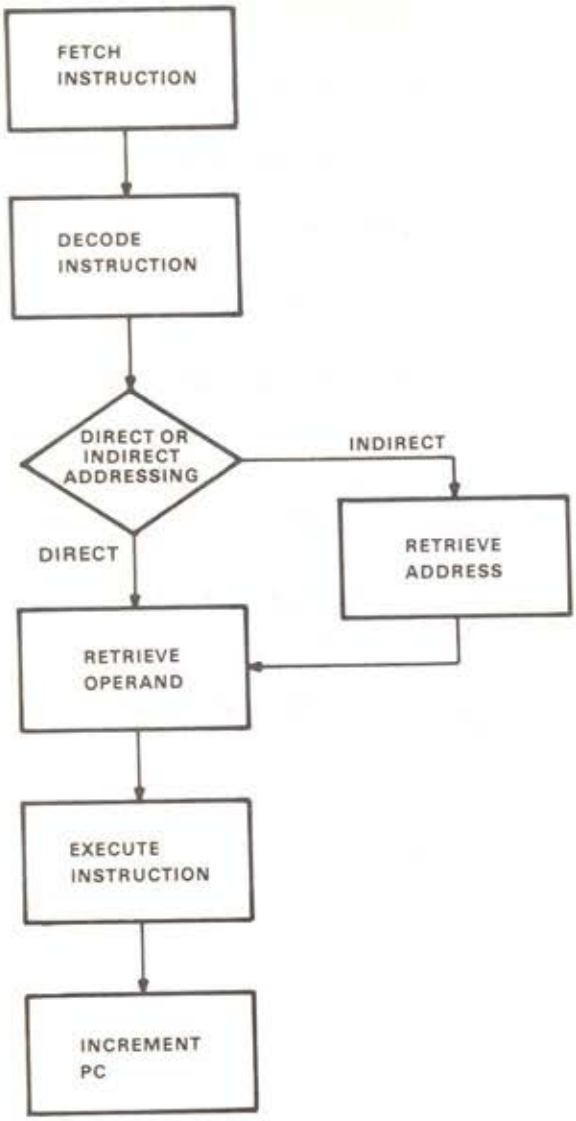


Figure 12 Instruction Cycle: Memory Reference Instructions

Fetch Phase

Figure 13 illustrates the first phase in the instruction cycle for the memory reference instruction **ADD I 300.** In this example the following information is already stored in main memory:

- Location 201 contains the ADD instruction.
- Location 300 contains the address 1500.
- Location 1500 contains the operand 3.

During the fetch phase, the following events occur:

(a.) The control logic establishes a data path between the PC and the AR.

(b.) The address of the ADD instruction (201) is sent to main memory by way of the AR.

(c.) The **ADD I 300** instruction is retrieved from memory and is temporarily placed in the BR.

(d.) The control logic establishes another data path between the BR and the IR.

(e.) The **ADD I 300** instruction is transferred to the IR so it can be decoded by the decoder.

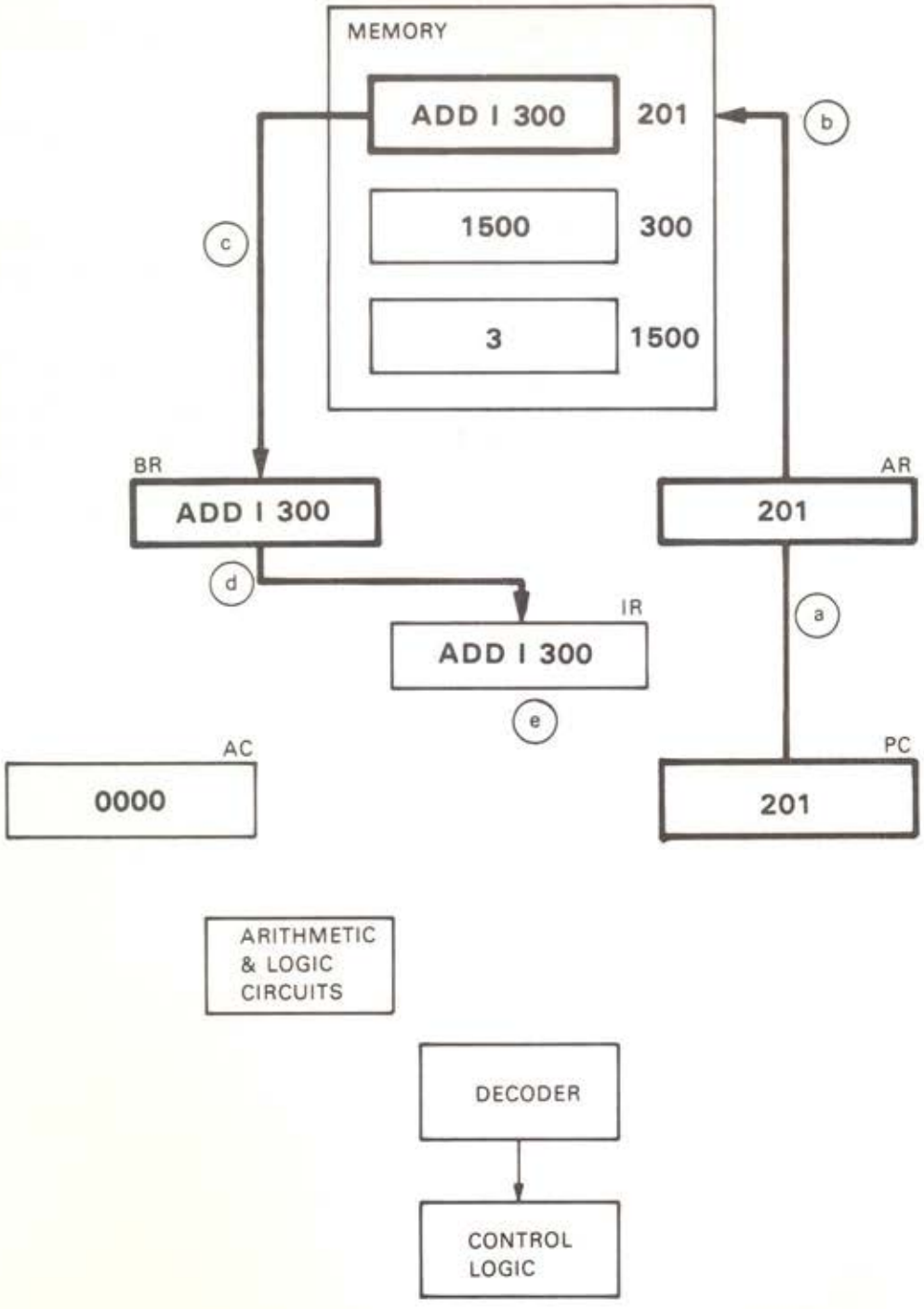


Figure 13 Fetch Phase: ADD Instruction

Decode Phase

Figure 14 illustrates the events that take place in the central processor during the decode phase.

a. Op code 001 (the op code for the ADD instruction) is routed to the instruction decoder.

b. The decoder converts op code 001 into a unique signal that is applied to the control logic. This signal tells the control logic to initiate an "Add" operation during the execute phase.

c. The decoder also converts the binary one in the I-field of the ADD instruction to a signal that designates that *indirect* addressing is to be used. This tells the CPU that memory must be referenced twice – first, the CPU must retrieve the *address* of the operand from memory; then it can use the address to retrieve the *operand* (refer to Figure 12).

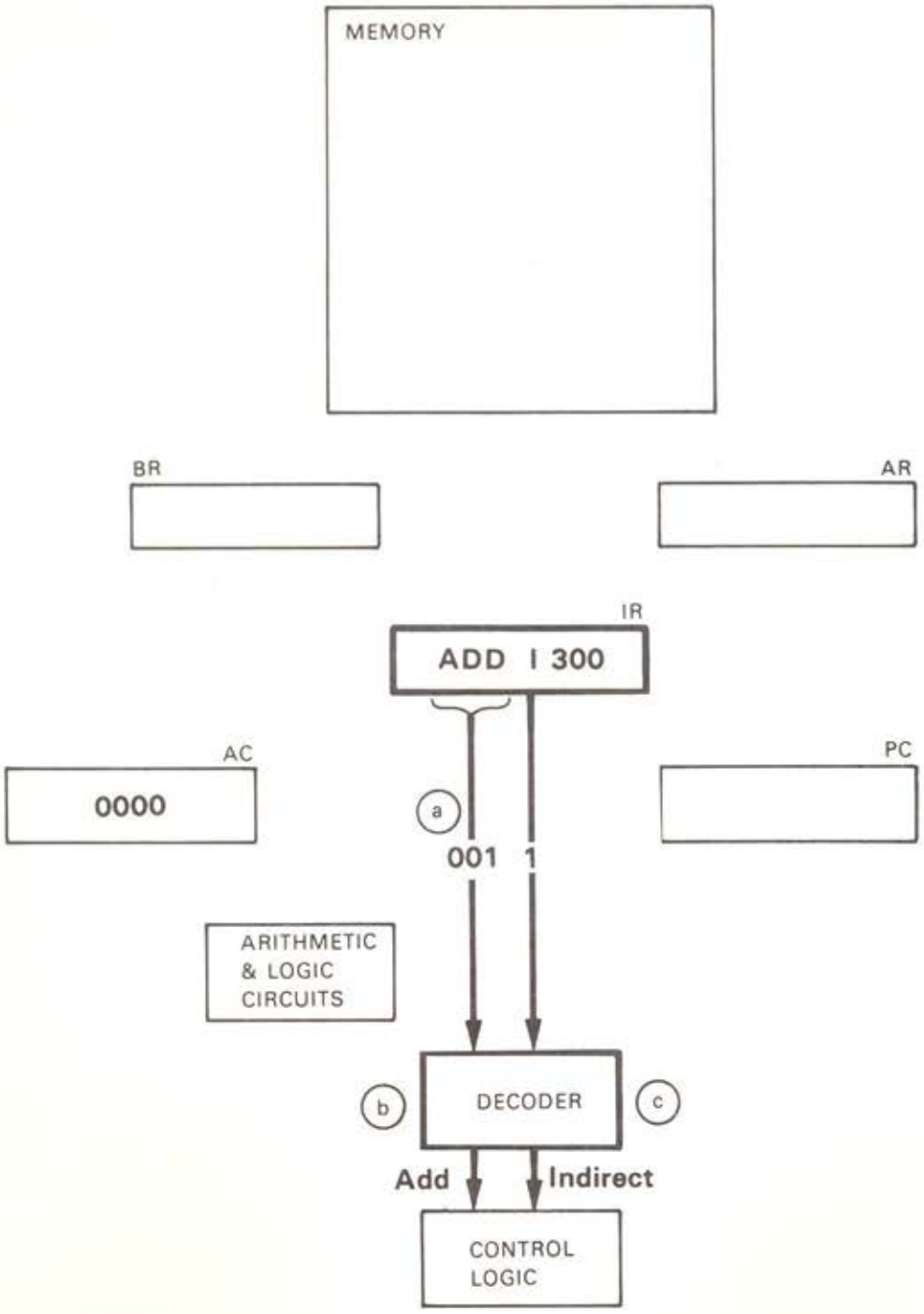


Figure 14 Decode Phase: ADD Instruction

Retrieve Address (Indirect Addressing Only)

The following events take place in the CPU in order to retrieve the operand address from main memory (see Figure 15):

(a.) The control logic establishes a data path between the IR and AR.

(b.) Memory location 300 is addressed using the contents of the IR (operand field only).

(c.) Address 1500 is retrieved from memory location 300 and is temporarily placed in the BR.

At this point in the instruction cycle, the central processor has obtained the operand address. The next step is to retrieve the operand from main memory by referencing memory a second time.

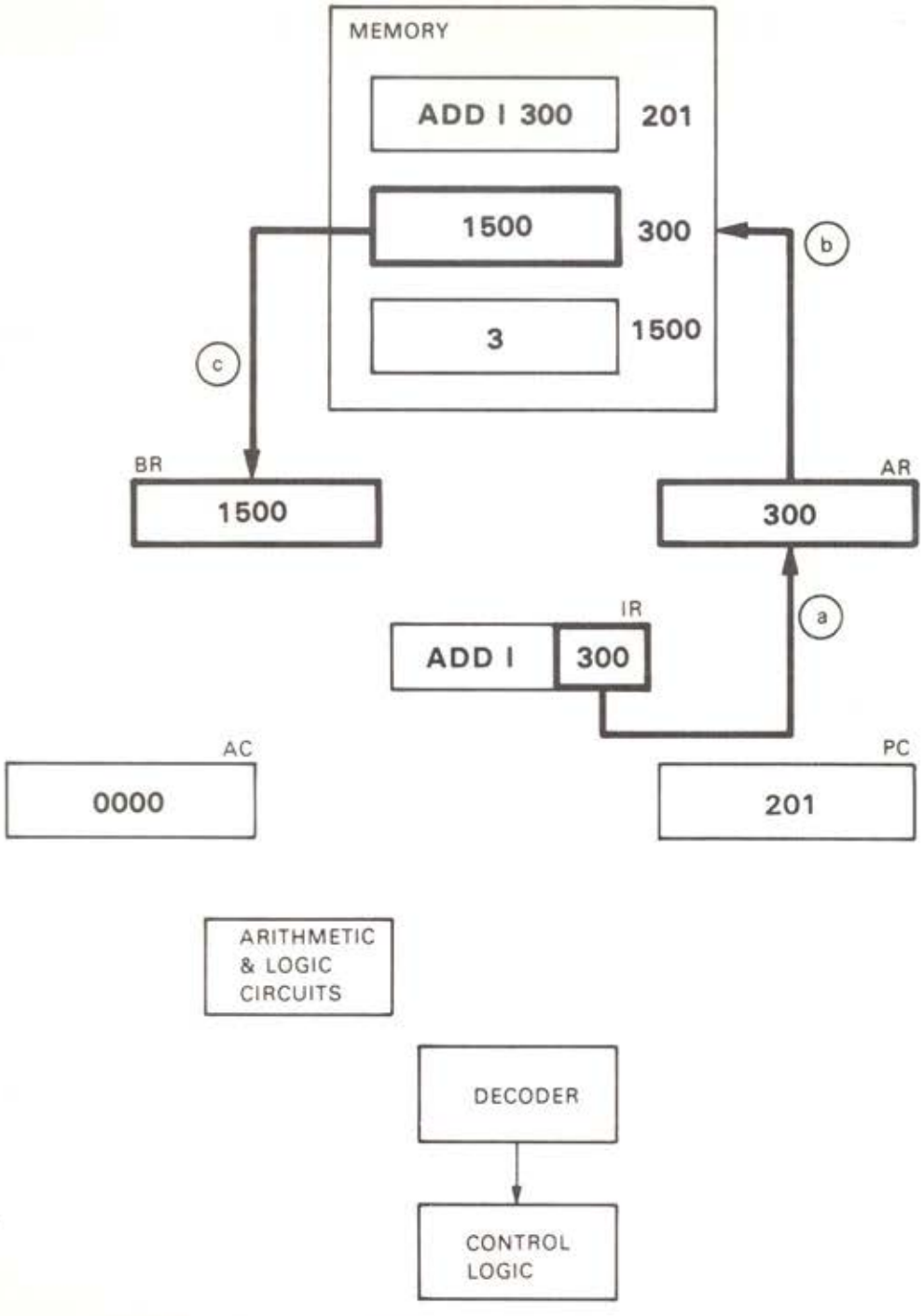


Figure 15 Retrieve Address of Operand

Retrieve Operand

The following events take place in order to retrieve the operand from memory (see Figure 16):

a. The control logic sets up a data path between the BR and the AR.

b. Memory location 1500 is addressed using the contents of the BR.

c. Operand 3 is retrieved from memory location 1500 and is temporarily placed in the BR, thereby replacing 1500 which is no longer needed.

Now that the CPU has retrieved the operand, it can proceed to execute the ADD instruction.

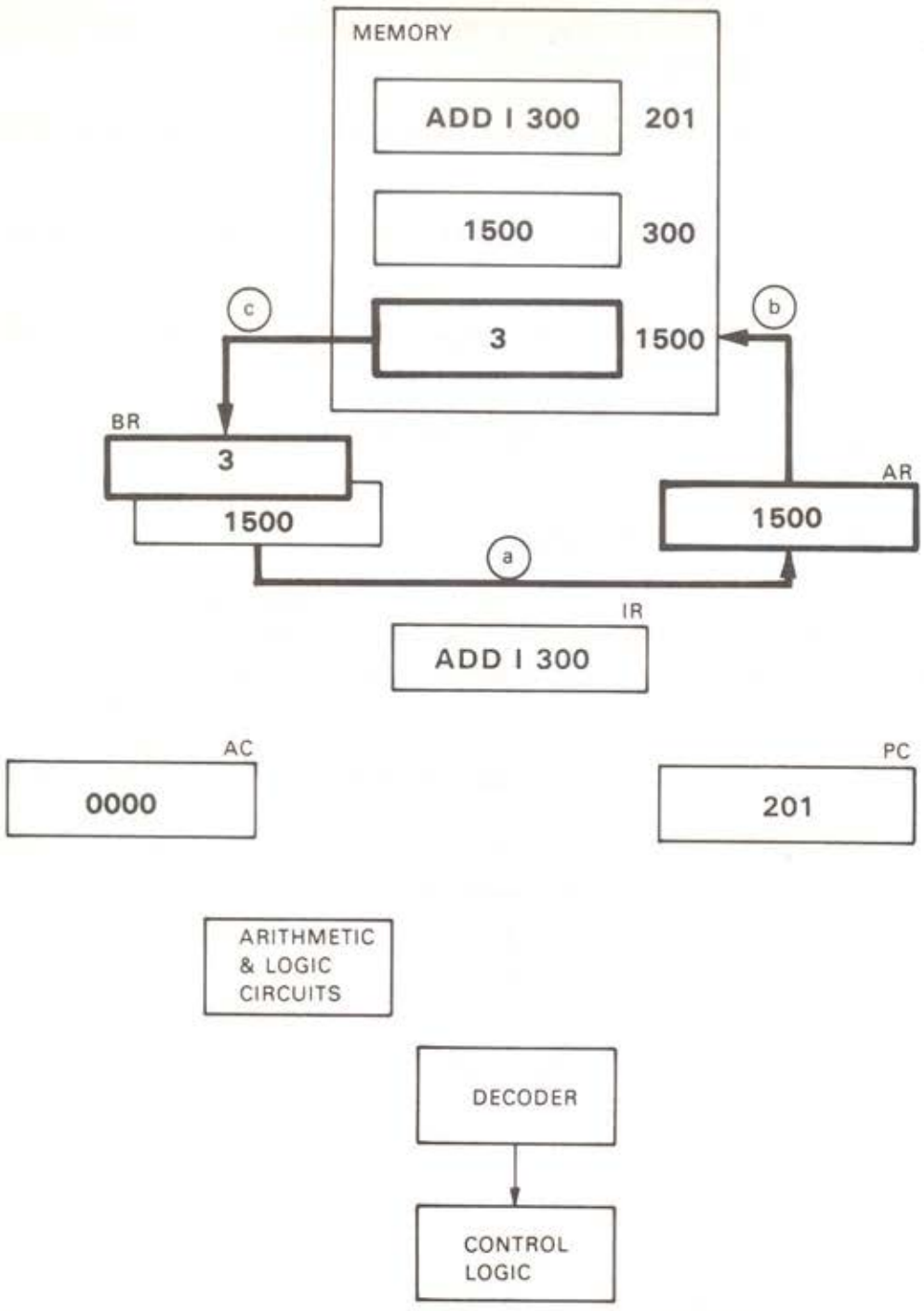


Figure 16 Retrieve Operand

Execute Phase

Figure 17 shows the events that take place in the central processor during the execute phase.

(a.) The control logic transmits an "add" signal to the arithmetic-logic circuits (ALC).

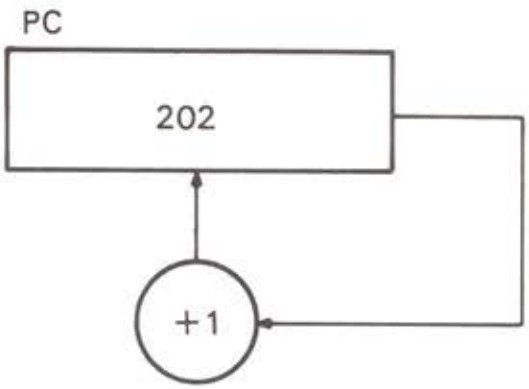
(b.) The control logic also establishes data paths between the BR, AC, and ALC.

(c.) The arithmetic-logic circuits add the contents of the BR and AC (3 + 0).

(d.) The resulting sum (3) is stored back into the AC.

Increment Program Counter

While the ADD instruction is being executed, the CPU increments the address stored in its PC. The new address contained in the PC (202) points to the next instruction in the sequence.



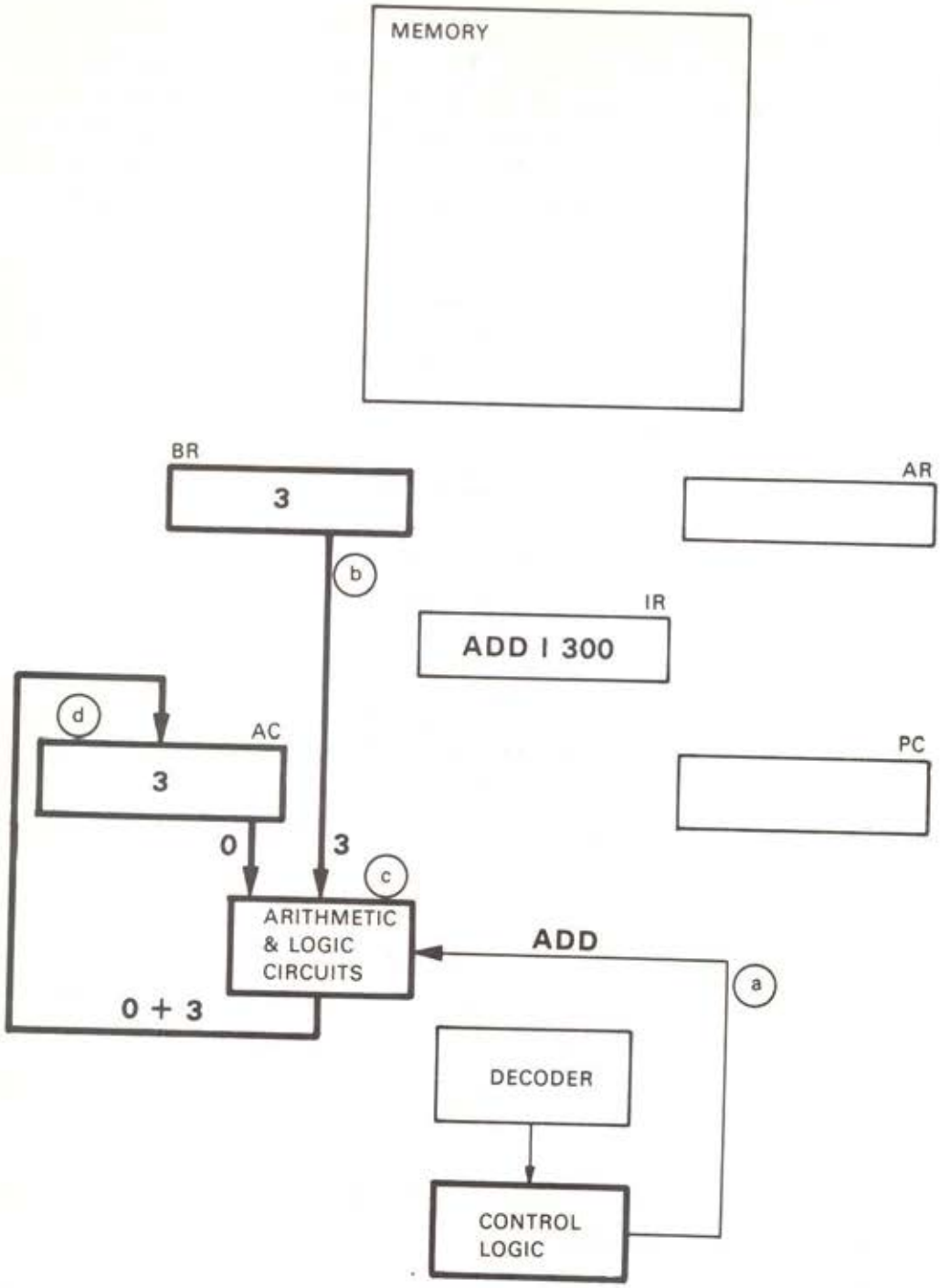


Figure 17 Execute Phase

Program Control Instructions

Program control instructions, such as JMP and ISZ, are handled somewhat differently by the CPU. The instruction cycle for the JMP instruction is shown in Figure 18. Note that there are only three phases. The major events that occur during the fetch and decode phases are basically the same as for the ADD and CLA instructions. The events that take place during the *execute* phase are illustrated in Figure 19.

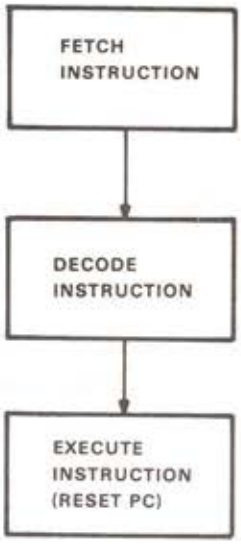


Figure 18 Instruction Cycle: Unconditional Jump Instruction

Execute Phase

The following events occur in the central processor when it executes the JMP instruction (see Figure 19):

a. The control logic establishes a data path between the IR, PC, and AR.

b. The PC and AR are reset to the address contained in the operand field of the JMP 210 instruction.

c. The central processor jumps to memory location 210 for its next instruction.

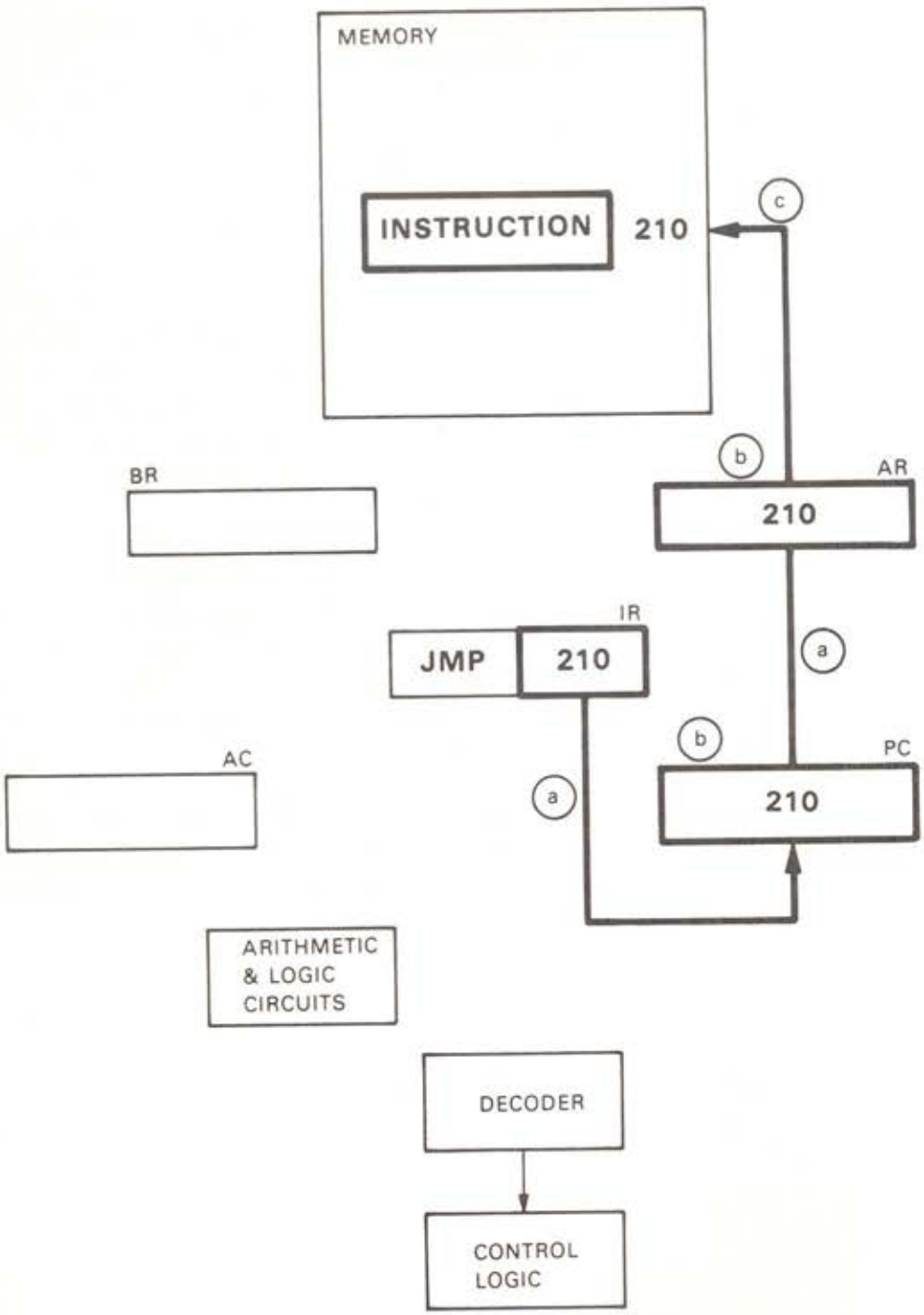


Figure 19 Execute Phase: JMP Instruction

Synchronous and Asynchronous Computers

Remember that the control logic establishes the data paths between registers and initiates actions by components such as the ALC and the instruction decoder. The control logic, however, must have some means to determine when one action has been completed and the next one can be initiated. There are two solutions to this problem.

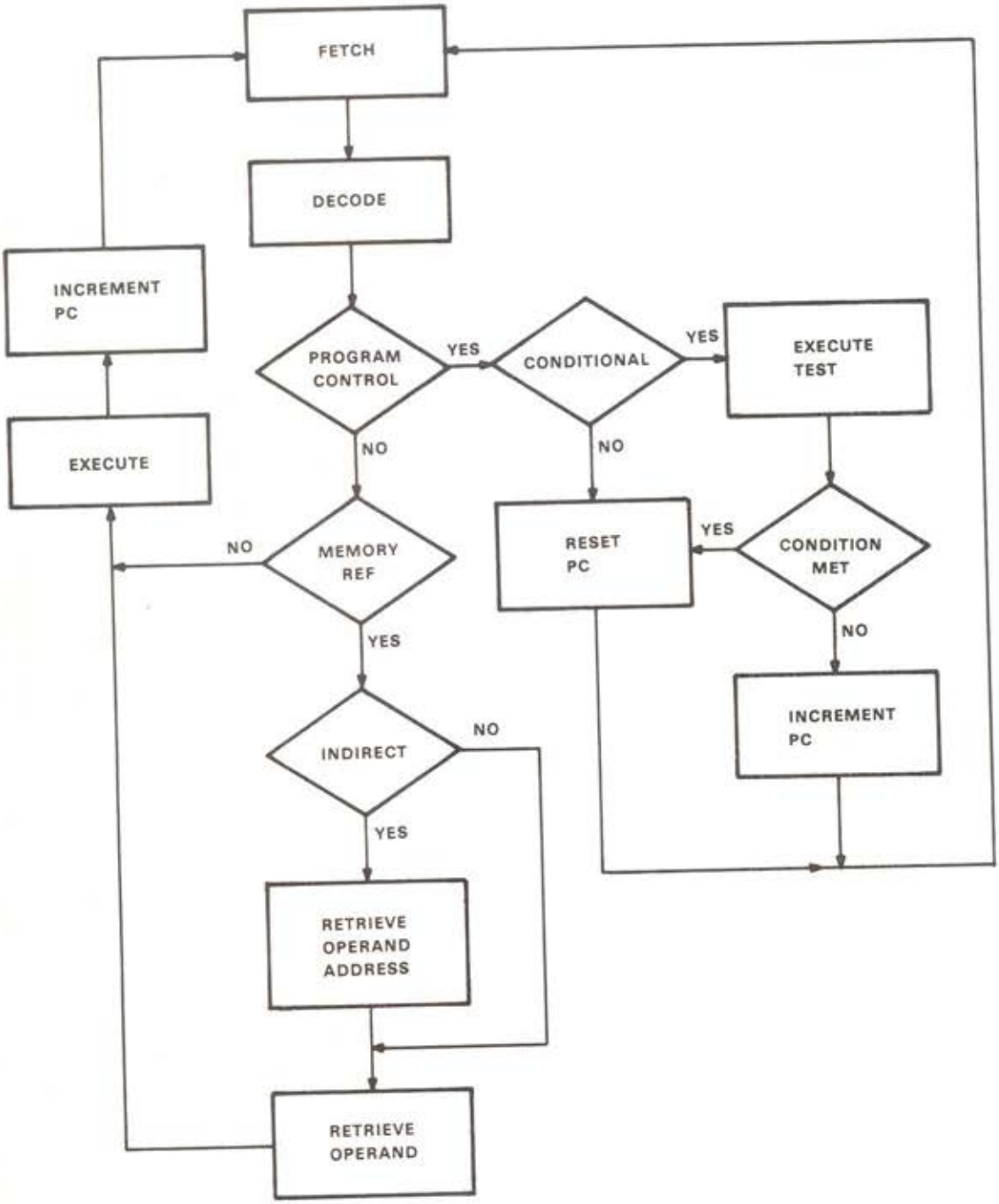
Many computers are *synchronous* – all their internal activities are synchronized with an internal clock. The clock sends several million pulses a second. Each activity of the CPU is assigned a maximum execution time in pulses (machine cycles). In executing an instruction, the clock pulses trigger the next operation after the previous time interval has elapsed. Small amounts of wasted CPU time may appear after each action if the actions take less time than their allocated time interval. Then, the CPU (specifically the control logic) simply waits for the next clock pulse.

On the other hand, *asynchronous* computers do not waste CPU time. Rather than being "clock driven," asynchronous computers are "event driven." That is, when an operation is completed, it produces a signal that initiates the next operation. In asynchronous computers a master clock is not required, but more hardware is generally required to keep track of the status of CPU operations.

Asynchronous computers are generally faster than synchronous computers because operations do not have to "fill up" fixed-time intervals. Rather, the next operation begins as soon as the previous one is completed.

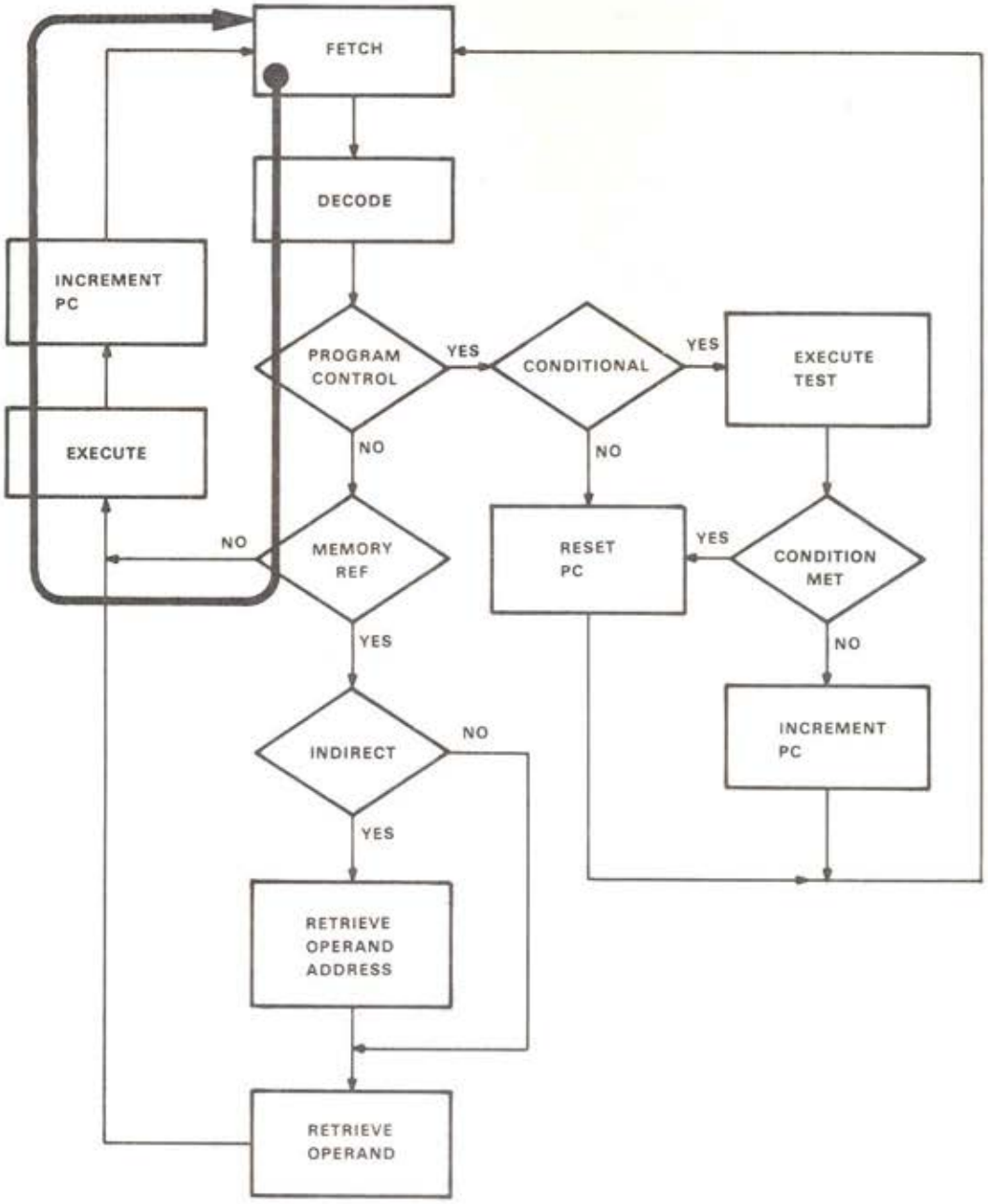
EXERCISES

1. Draw a heavy line on the flowchart below to indicate the steps in the instruction cycle for the instruction **CMA**.



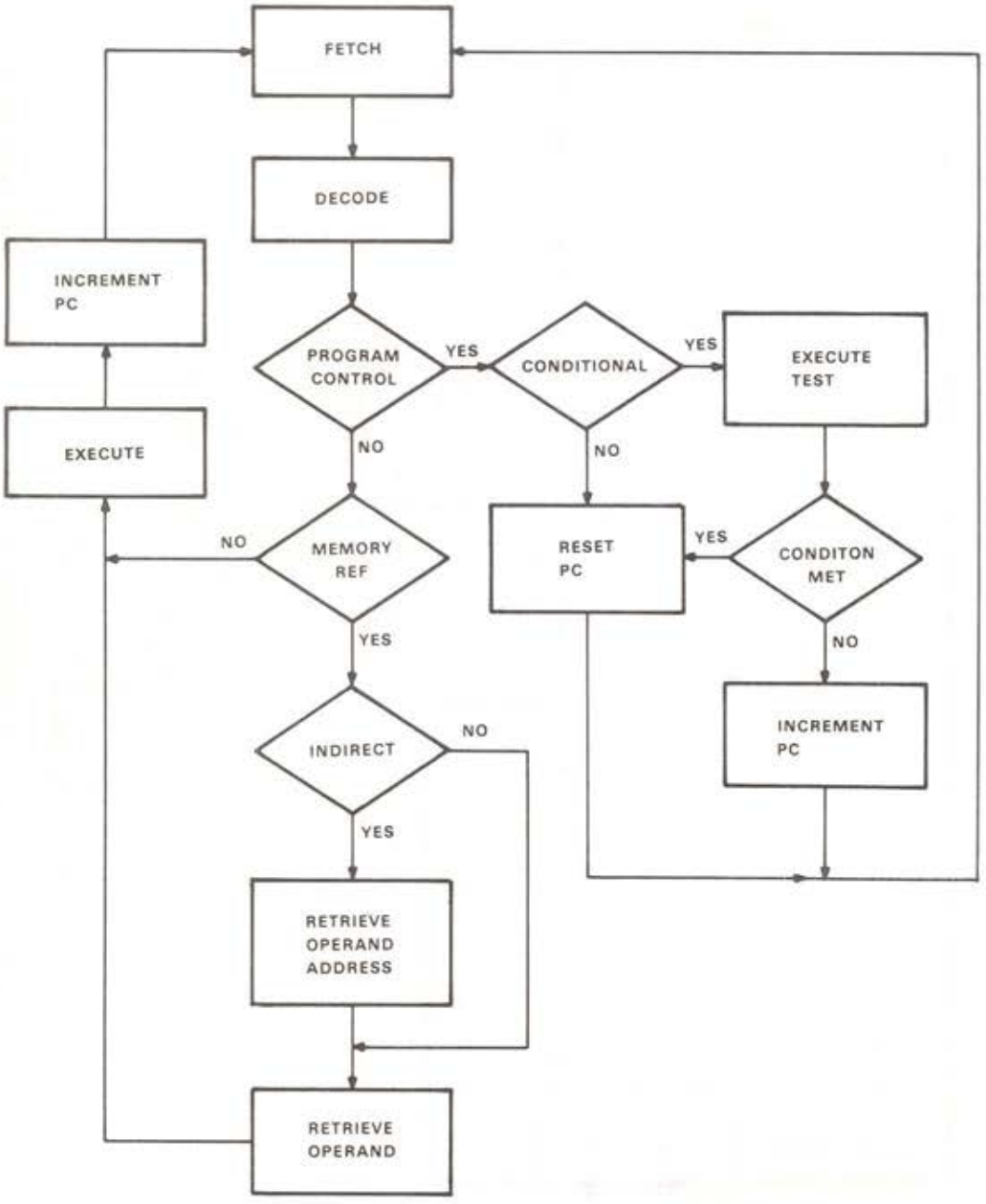
SOLUTIONS

1. Draw a heavy line on the flowchart below to indicate the steps in the instruction cycle for the instruction **CMA.**

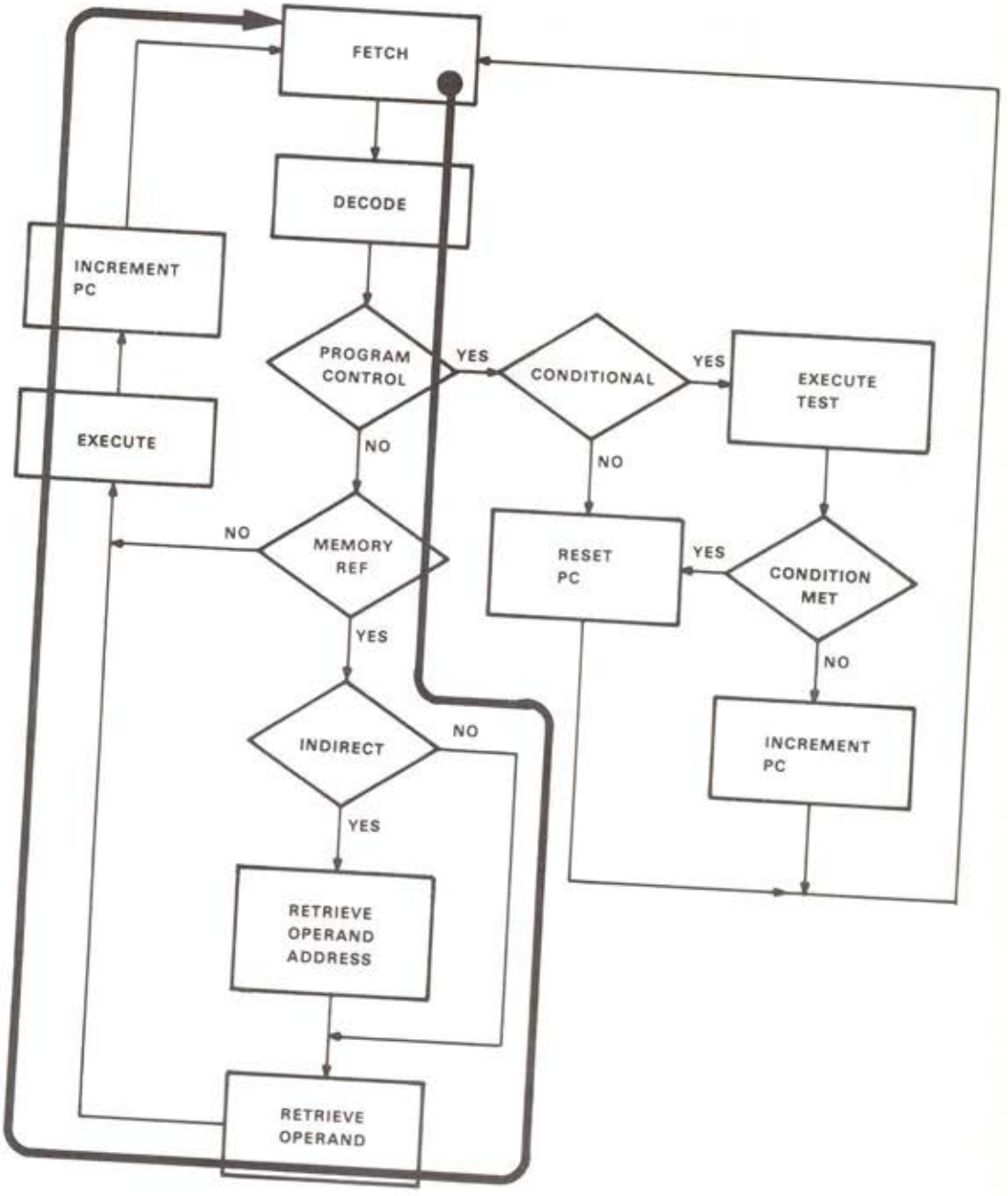


EXERCISES

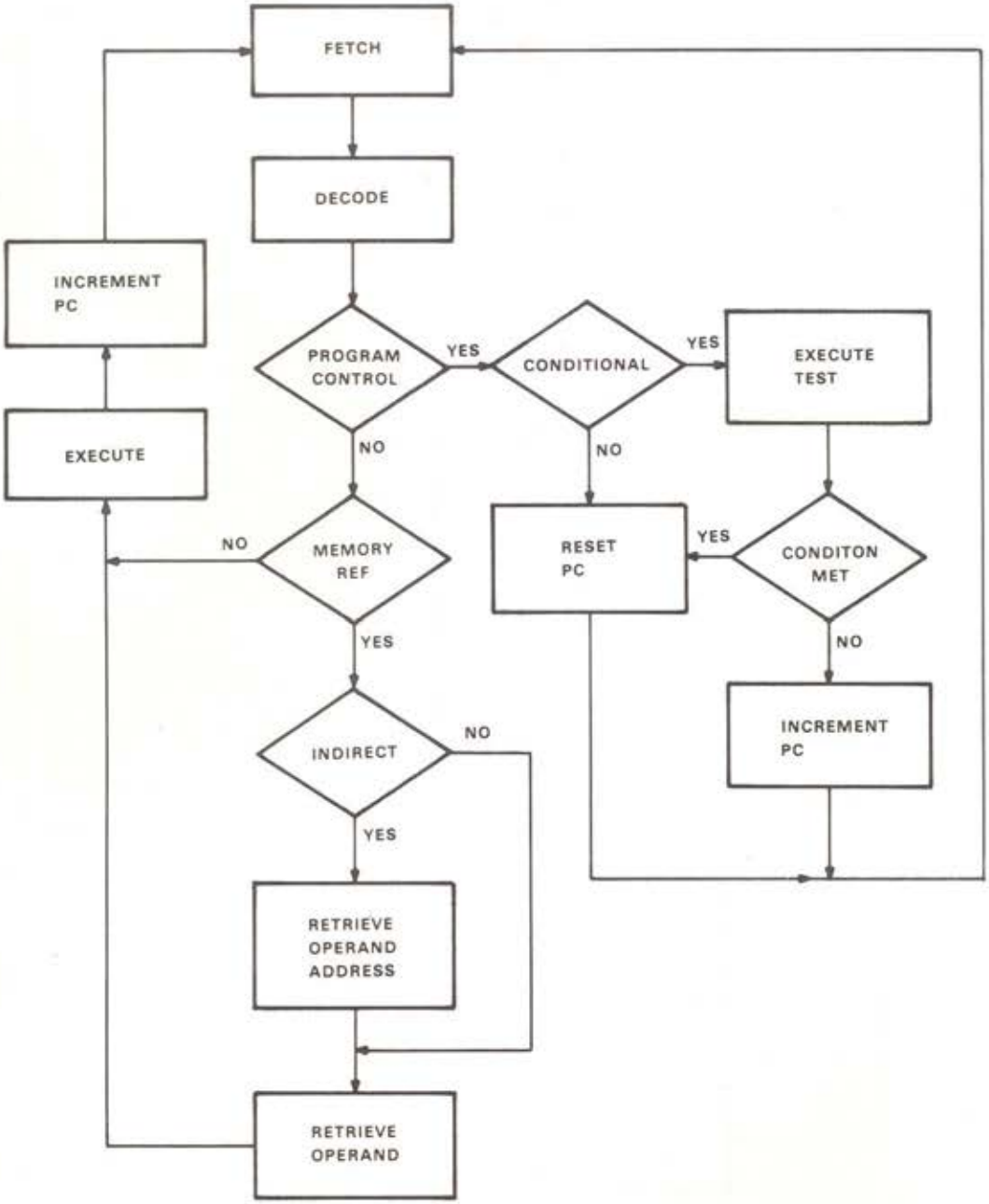
2. Draw a heavy line on the flowchart below to indicate the steps in the instruction cycle for the instruction **ADD 257.**



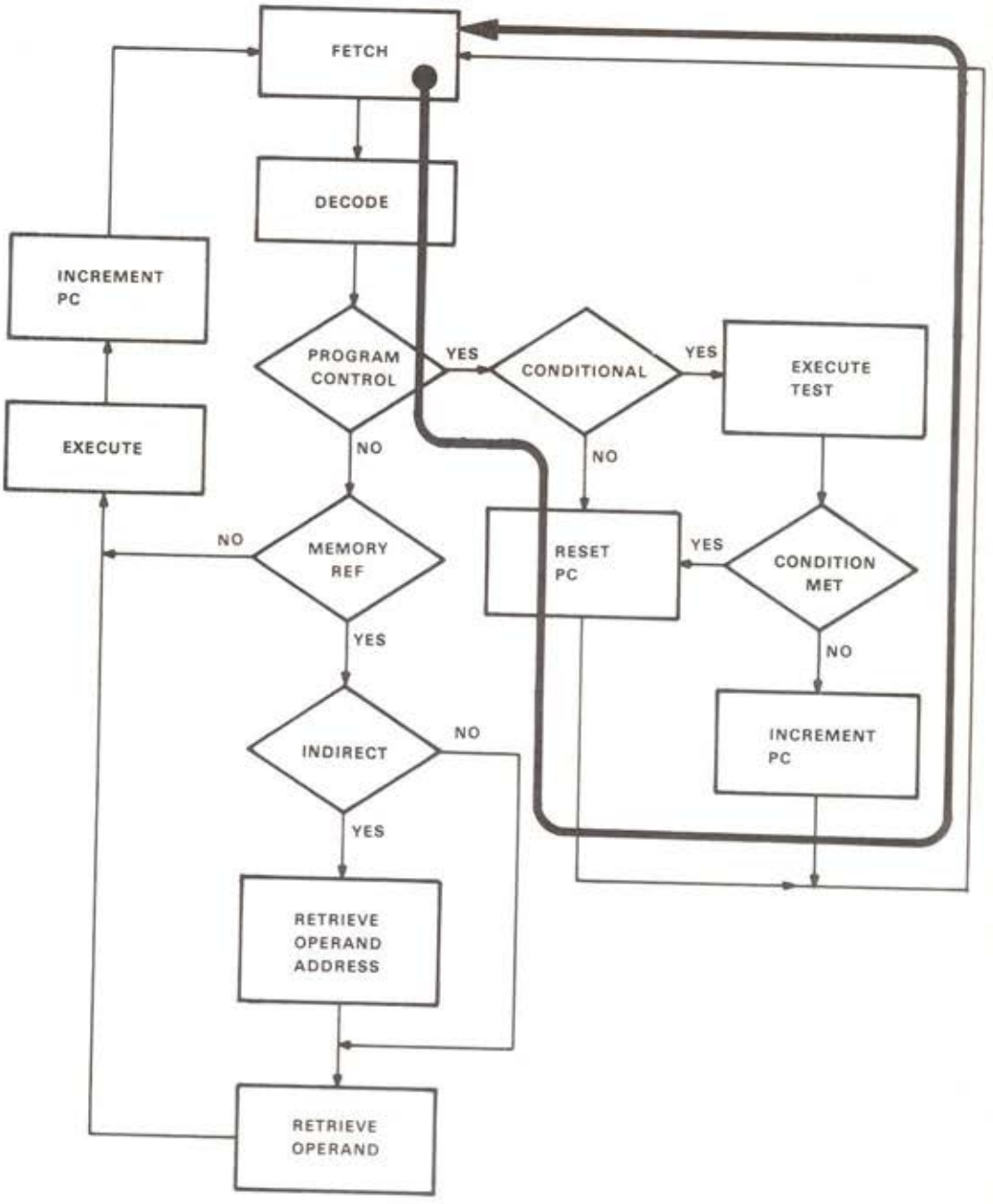
2. Draw a heavy line on the flowchart below to indicate the steps in the instruction cycle for the instruction **ADD 257.**



3. Draw a heavy line on the flowchart below to indicate the steps in the instruction cycle for the instruction **JMP 220.**



3. Draw a heavy line on the flowchart below to indicate the steps in the instruction cycle for the instruction **JMP 220.**



4. The simple program below has been stored in the computer's memory.

PROGRAM

Address	Contents
502	CLA
503	ADD 510
504	ADD 511
505	ADD 507
506	JMP 512
507	3
510	2
511	1
512	STR 514
513	HLT
514	0

Step through this program, one instruction at a time, by indicating what information is contained in each of the five major CPU registers at the *end* of each instruction cycle. Use the table below which shows the first instruction as an example.

REGISTERS

Program Counter		Buffer Register	Address Register	Instruction Register	Accumulator
Before Execution	After Execution				
502	503	"CLA"	502	"CLA"	0

4. The simple program below has been stored in the computer's memory.

PROGRAM

Address	Contents
502	CLA
503	ADD 510
504	ADD 511
505	ADD 507
506	JMP 512
507	3
510	2
511	1
512	STR 514
513	HLT
514	0

Step through this program, one instruction at a time, by indicating what information is contained in each of the five major CPU registers at the *end* of each instruction cycle. Use the table below which shows the first instruction as an example.

REGISTERS

Program Counter					
Before Execution	After Execution	Buffer Register	Address Register	Instruction Register	Accumulator
502	503	"CLA"	502	"CLA"	0
503	504	2	510	"ADD 510"	2
504	505	1	511	"ADD 511"	3
505	506	3	507	"ADD 507"	6
506	512	"JMP 512"	512	"JMP 512"	6
512	513	6	514	"STR 514"	0
513	514	"HLT"	513	"HLT"	0

EXERCISES

5. Differentiate between synchronous and asynchronous computers. Explain why an asynchronous computer is usually faster than a synchronous computer.

5. Differentiate between synchronous and asynchronous computers. Explain why an asynchronous computer is usually faster than a synchronous computer.

Synchronous computers use an internal clock to control a fixed time interval for each operation. When the interval has elapsed, the next operation is initiated.

Asynchronous computers do not use an internal clock. When an operation is completed, it triggers the next operation. There is no fixed-time interval for any operation.

The speed advantage of asynchronous computers is a result of asynchronous computers not having to "use up" fixed-time intervals. The next operation begins as soon as the previous one is completed – without waiting.

Typical Functions of the Console

OBJECTIVE

Given seven computer functions, be able to select those four functions that refer to the computer console.

SAMPLE TEST ITEM

Circle the *letters* of the functions that refer to the computer console.

a. Initiate execution of a program.

b. Load small- to medium-size programs into memory.

c. Check status of a program.

d. Examine or alter the contents of a memory location.

e. Selct status of a program.

f. Load small programs into memory.

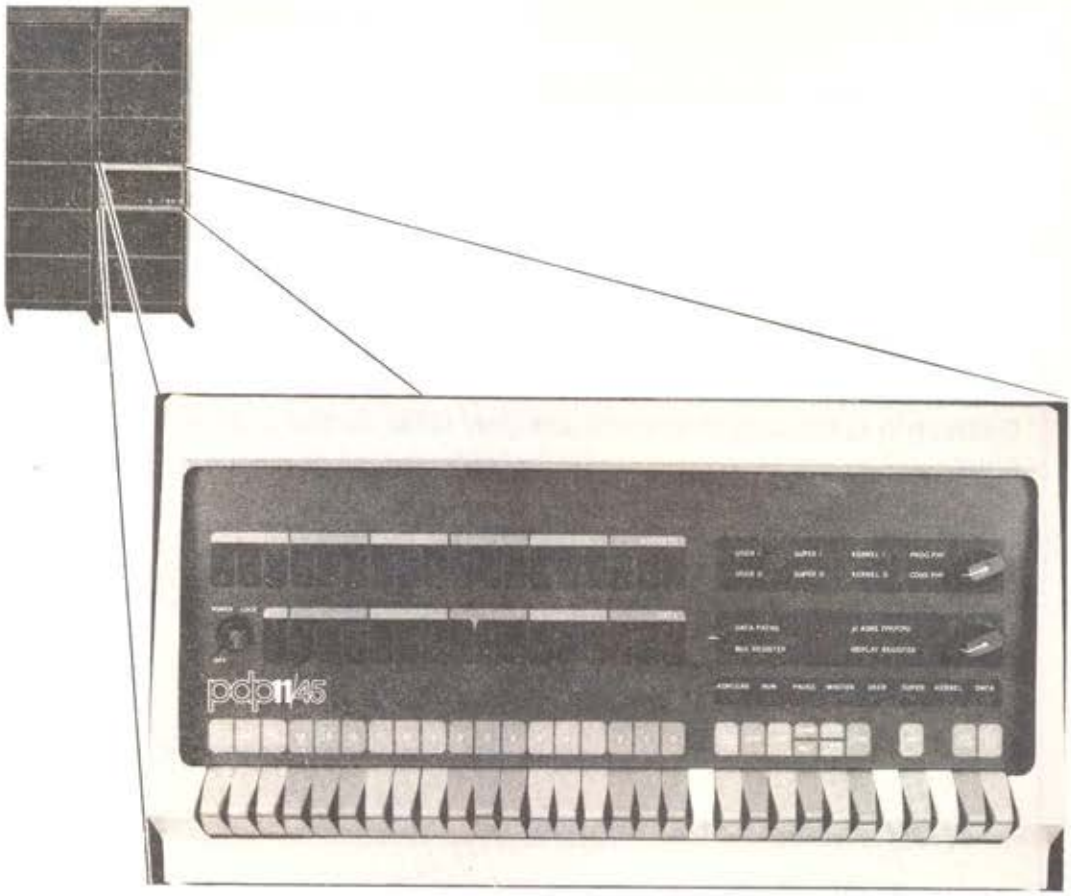


Figure 20 Computer Console

Mark your place in this workbook and view Lesson 3 in the A/V program, "Central Processor."

The computer console (Figure 20) is a set of switches and lamps that allow the computer operator to monitor and control certain aspects of computer operation. Often, the console is the only part of the computer mainframe that is readily accessible to the user or operator. In many cases, the console is also the only visible indication of the computer's activities. In fact, the visibility and accessibility of the console cause many people to confuse the console with the central processor as a whole.

Normally, computer operations are controlled automatically by programs stored in main memory. Some situations, however, require manual intervention or assistance by the computer operator. The console supplies this ability. The switches and indicators on the console differ greatly from one computer model to another. This lesson, therefore, discusses only general features found on most consoles.

The four basic functions of a typical computer console are:

- to examine or alter information stored in main memory,
- to load small programs into main memory,
- to start execution of a program, and
- to check the status of a program.

Let's look at each of these functions.

Examining or Altering Data in Memory

The computer operator may examine or alter a specific location by performing the following steps. First, the desired address is loaded into the machine by setting a row of switches to the binary address. Setting a switch in the "up" position is equivalent to a "1" in the corresponding address bit. Leaving a switch down is interpreted as a "0" in the address. To enter this address from the switches (also called the *switch register)* into the PC and the AR, the operator presses a LOAD ADDRESS switch. To examine the contents of this location, the operator merely presses an EXAMINE switch. The contents of this location (now in the BR) are displayed by indicator lamps over the switch register. A lamp "on" represents a "1" bit – an "off" lamp indicates a "0" bit. If the operator wants to modify the contents of this location, a new binary value is entered with the switch register, and then the operator presses the DEPOSIT switch. The value in the switch register is then stored in main memory at the selected address.

Loading Small Programs into Memory

This function is simply an extension of the ability to alter single locations in memory. To enter a small program into memory, the following steps are performed:

1. the starting address of the program is set in the switch register;

2. a LOAD ADDRESS switch is pressed (the address is now in the PC and AR);

3. the binary machine code form of the first instruction is now entered in the switch register;

4. a DEPOSIT switch is pressed to insert the instruction into memory (the PC is incremented for the next instruction);

5. steps 3 and 4 are then repeated for each instruction in the program.

Because this is a tedious and error-prone process, it is *not* practical for entering programs of more than twenty instructions. However, there is a definite need to be able to manually load small programs.

For example, when a computer is first delivered, its memory is completely blank. In order to perform normal operations, the computer must first be given a program to load other programs. Because a complete loader program is significantly larger than twenty instructions, a simple program, called the *bootstrap loader,* is first entered using the switches on the console. The bootstrap loader is then used to read more instructions from an input device (such as a paper tape reader) and load them into memory. These new instructions are the general purpose loader. Thus, the computer "pulls itself up by its own bootstrap."

Initiating Execution of a Program

A program loaded in memory may be executed by performing the following steps:

1. the starting address of the program is entered through the switch register,

2. the LOAD ADDRESS switch is pressed to set the PC and AR to the starting address,

3. the RUN (or START) switch is pressed, causing the CPU to start executing the program.

The program may be stopped by pressing a HALT switch.

Checking the Status of a Program

Various other indicator lamps may be present on the console. These lamps may be used to display the contents of registers other than the AC and the PC, to indicate error conditions (such as overflow or addressing errors), or to indicate whether the CPU is running (executing a program) or waiting. These lamps normally change too rapidly for the human eye to read them, but when the computer is halted, they are easily viewed. These indicators may provide clues concerning possible sources of trouble in the program by allowing the programmer to compare the *expected* behavior of the program with its actual executed behavior.

EXERCISES

1. Explain briefly three possible uses of the computer console.

2. Number the following operations to show the sequence required to alter the contents of a specific memory location.

   ( ) Enter the new contents of the location with the switch register.

   ( ) Enter the address of the desired location with the switch register.

   ( ) Press the DEPOSIT switch.

   ( ) Press the LOAD ADDRESS switch.

SOLUTIONS

1. Explain briefly three possible uses of the computer console.

   There are four basic functions of the computer console:

   a. Examining or altering specific memory locations through the switch register allows detailed corrections (of a program or data) to be made.

   b. Small programs, such as the bootstrap loader, may be deposited through the switch register to get a new computer "started" so that it can load more complex programs.

   c. Program execution can be initiated by specifying the starting address through the switch register and pressing the RUN or START switch.

   d. The status of a program may be checked by using various indicator lamps displaying the contents of registers, indicating error conditions, or indicating the RUN or WAIT states of the CPU.

2. Number the following operations to show the sequence required to alter the contents of a specific memory location.

   (3) Enter the new contents of the location with the switch register.

   (1) Enter the address of the desired location with the switch register.

   (4) Press the DEPOSIT switch.

   (2) Press the LOAD ADDRESS switch.

Take the test for this module before beginning another module.