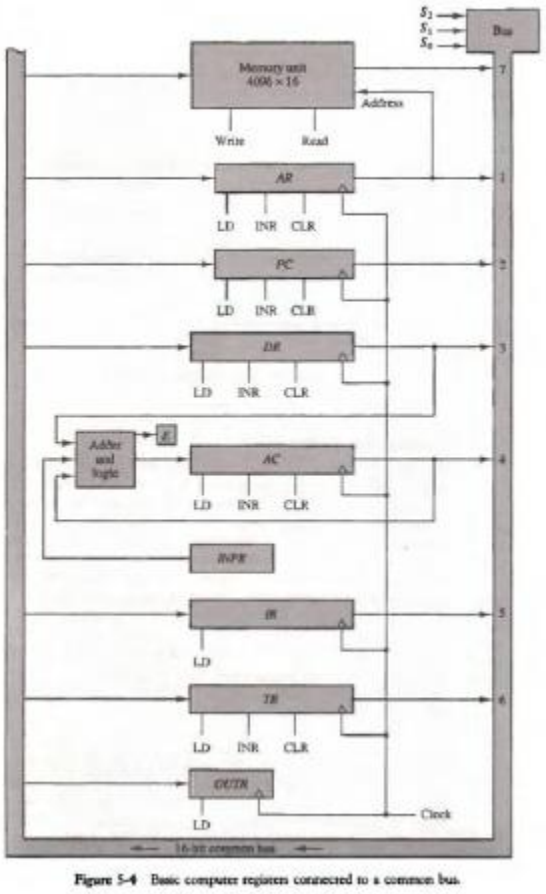


Figure 5-4 Basic computer registers connected to a common bus.

Bus design continued…

Four registers, DR, AC, IR, and TR, have 16 bits each.

Two registers, AR and PC, have 12 bits each since they hold a memory address.

When the contents of AR or PC are applied to the 16-bit common bus, the four most significant bits are set to 0's. When AR or PC receive information from the bus, only the 12 least significant bits are transferred into the register.

The input register INPR and the output register OUTR have 8 bits each and communicate with the eight least significant bits in the bus.

Five registers have three control inputs: LD (load), INR (increment), and CLR (clear).

Two registers have only a LD input.

The input data and output data of the memory are connected to the common bus, but the memory address is connected to AR. Therefore, AR must always be used to specify a memory address.

By using a single register for the address, we eliminate the need for an address bus that would have been needed otherwise.

The 16-bit inputs of AC come from an adder and logic circuit. This circuit has three sets of inputs. One set of 16-bit inputs come from the outputs of AC. They are used to implement register microoperations such as complement AC and shift AC. Another set of 16-bit inputs come from the data register DR. The inputs from DR and AC are used for arithmetic and logic microoperations, such as add DR to AC or AND DR to AC. The result of an addition is transferred to AC and the end carry-out of the addition is transferred to flip-flop E (ex-tended AC bit).

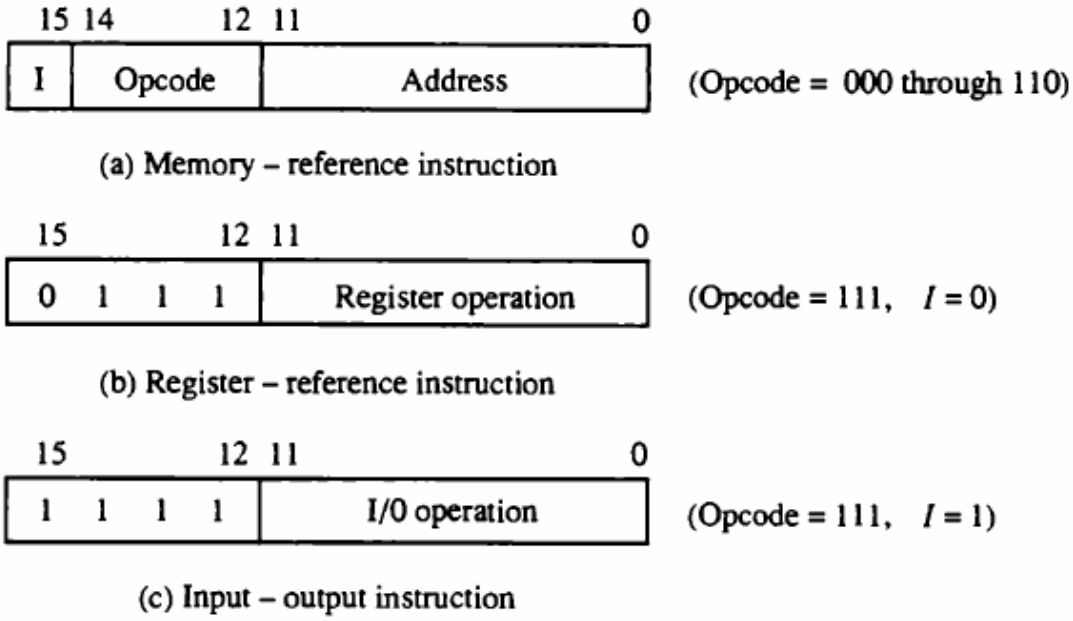
Note that the content of any register can be applied onto the bus and an operation can be performed in the adder and logic circuit during the same clock cycle. The clock transition at the end of the cycle transfers the content of the bus into the designated destination register and the output of the adder and logic circuit into AC. For example, the two microoperations

DR <= AC and AC <= DR

can be executed at the same time. This can be done by placing the content of AC on the bus (with $S_2S_1S_0$= 100), enabling the LD (load) input of DR, transferring the content of DR through the adder and logic circuit into AC,

and enabling the LD (load) input of AC, all during the same clock cycle.

Figure 5-5 Basic computer instruction formats.



Computer instruction format

The basic computer has three instruction code formats, as shown in Fig. 5-5.

A memory-reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode I.

I is equal to 0 for direct address and to 1 for indirect address (see Fig. 5-2).

The register-reference instructions are recognized by the operation code 111 with a 0 in the leftmost bit (bit 15) of the instruction. A register-reference instruction specifies an operation on or a test of the AC register. An operand from memory is not needed; therefore, the other 12 bits are used to specify the operation or test to be executed.

An input-output instruction does not need a reference to memory and is recognized by the operation code 111 with a 1 in the leftmost bit of the instruction. The remaining 12 bits are used to specify the type of input-output operation or test performed.

Note that the bit in position 15 of the instruction code is designated by the symbol I but is not used as a mode bit when the operation code is equal to 111.

Only three bits of the instruction are used for the operation code. It may seem that the computer is restricted to a maximum of eight distinct operations.

However, since register-reference and input-output instructions use the remaining 12 bits as part of the operation code, the total number of instructions can exceed eight. In fact, the total number of instructions chosen for the basic computer 1s equal to 25.

TABLE 5-2 Basic Computer Instructions

Symbol	Hexadecimal code $I = 0$	Hexadecimal code $I = 1$	Description
AND	0xxx	8xxx	AND memory word to *AC*
ADD	1xxx	9xxx	Add memory word to *AC*
LDA	2xxx	Axxx	Load memory word to *AC*
STA	3xxx	Bxxx	Store content of *AC* in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear *AC*
CLE	7400		Clear *E*
CMA	7200		Complement *AC*
CME	7100		Complement *E*
CIR	7080		Circulate right *AC* and *E*
CIL	7040		Circulate left *AC* and *E*
INC	7020		Increment *AC*
SPA	7010		Skip next instruction if *AC* positive
SNA	7008		Skip next instruction if *AC* negative
SZA	7004		Skip next instruction if *AC* zero
SZE	7002		Skip next instruction if *E* is 0
HLT	7001		Halt computer
INP	F800		Input character to *AC*
OUT	F400		Output character from *AC*
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

A memory-reference instruction has an address part of 12 bits. The address part is denoted by three x's and stand for the three hexadecimal digits corresponding to the 12-bit address.

Instruction Set Completeness

Before investigating the operations performed by the instructions, let us discuss the type of instructions that must be included in a computer. A computers should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable. The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories:

1. Arithmetic, logical, and shift instructions

2. Instructions for moving information to and from memory and processor registers.

3. Program control instructions together with instructions that check status conditions

4. Input and output instructions

Arithmetic, logical, and shift instructions provide computational capabilities for processing the type of data that the user may wish to employ.

The bulk of the binary information in a digital computer is stored in memory, but all computations are done in processor registers. Therefore, the user must have the capability of moving information between these two units.

Decision-making capabilities are an important aspect of digital computers. For example, two numbers can be compared, and if the first is greater than the second, it may be necessary to proceed differently than if the second is greater than the first. Program control instructions such as branch instructions are used to change the sequence in which the program is executed. Input and output instructions are needed for communication between the computer and the user. Programs and data must be transferred into memory and results of computations must be transferred back to the user.

Although the set of instructions for the basic computer is complete, it is not efficient because frequently used operations are not performed rapidly. An efficient set of instructions will include such instructions as subtract, multiply, OR, and exclusive-OR. These operations must be programmed in the basic computer. The programs are presented in Chap. 6 together with other programming examples for the basic computer. By using a limited number of instructions it is possible to show the detailed logic design of the computer. A more complete set of instructions would have made the design too complex.

Control Unit design

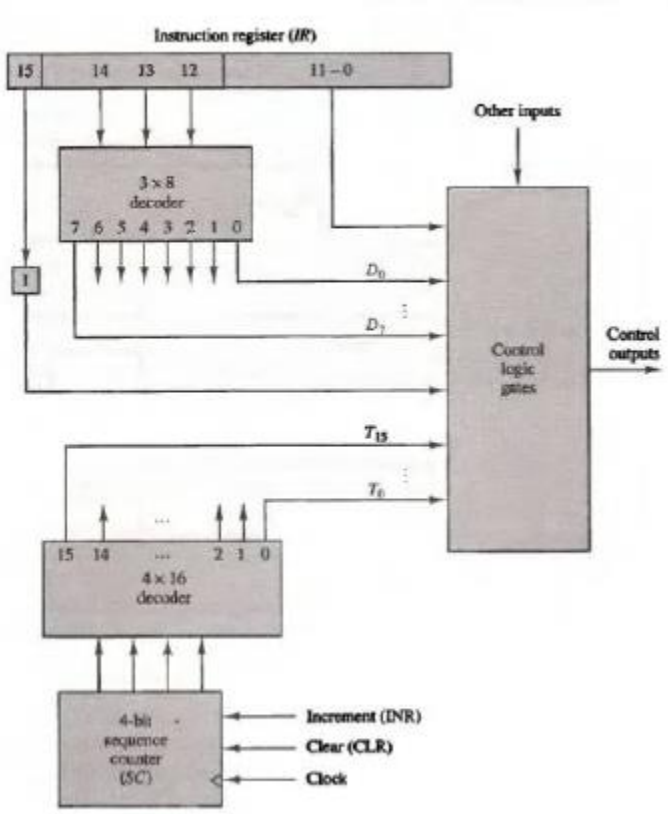


Figure 5-6 Control unit of basic computer.

The block diagram of the control unit is shown in Fig. 5-6. It consists of two decoders, a sequence counter, and a number of control logic gates. An instruction read from memory is placed in the instruction register (IR). The position of this register in the common bus system is indicated in Fig. 5-4. The instruction register is shown again in Fig. 5-6, where it is divided into three

parts: the I bit, the operation code, and bits 0 through 11. The operation code in bits 12 through 14 are decoded with a3 x 8 decoder. The eight outputs of the decoder are designated by the symbols D_0 through D_7 . The subscripted decimal number is equivalent to the binary value of the corresponding operation code. Bit 15 of the instruction is transferred to flip-flop designated by the symbol I. Bits 0 through 11 are applied to the control logic gates. The 4-bit sequence counter can count in binary from 0 through 15. The outputs of the counter are decoded into 16 timing signals T_0 through T_{15}. The internal logic of the control gates will be derived later when we consider the design of the computer in detail.

Instruction cycle

A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction.

Each instruction cycle in turn is subdivided into a sequence of subcycles or phases.

In the basic computer each instruction cycle consists of the following phases:

1. **Fetch an instruction from memory.**
2. **Decode the instruction.**
3. **Read the effective address from memory if the instruction has an indirect address.**
4. **Execute the instruction.**

Upon the completion of step 4, the control goes back to step 1 to fetch, decode, and execute the next instruction. This process continues indefinitely unless a HALT instruction is encountered.

Fetch and Decode

Initially, the program counter PC is loaded with the address of the first instruction in the program.

The sequence counter SC is cleared to 0, providing a decoded timing signal T_0.

After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0, T_1, T_2, and so on.

The microoperations for the fetch and decode phases can be specified by the following register transfer statements.

T_0: AR←PC

T_1: IR←M[AR], PC←PC +1

T_2: D_0,… D_7← Decode IR(12-14), AR←IR(0-11), I← IR(15)

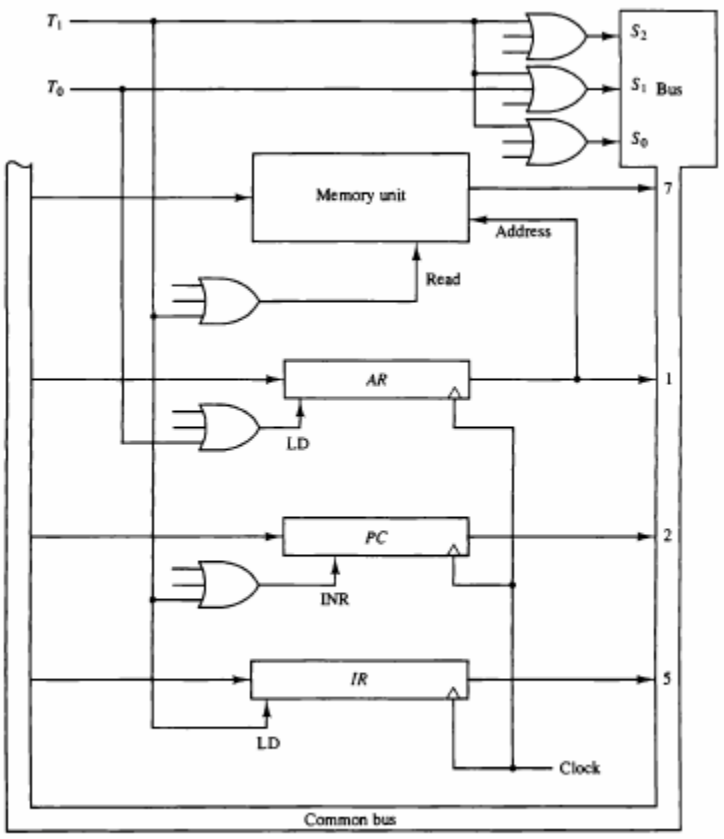


Figure 5-8 Register transfers for the fetch phase.

Since only AR is connected to the address inputs of memory it is necessary to transfer the address from PC to AR during the clock transition associated with timing signal T_0.

The instruction read from memory is then placed in the instruction register IR with the clock transition associated with timing signal T_1. At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.

At time T_2 the operation code in IR is decoded, the indirect bit is transferred to flip-flop I, and the address part of the

instruction is transferred to AR. Note that SC is incremented after each clock pulse to produce the sequence T_0, T_1, and T_2.

To implement operation at time T_0 i.e.

T_0: AR←PC

1. Place the content of PC onto the bus by making the bus selection inputs $S_2S_1S_0$ equal to 010.

2. Transfer the content of the bus to AR by enabling the LD input of AR.

To implement the operation:

T_1: IR ←M[AR], PC← PC + 1

it is necessary to use timing signal T_1 to provide the following connections in the bus system.

1. Enable the read input of memory.

2. Place the content of memory onto the bus by making $S_2S_1S_0$= 111.

3. Transfer the content of the bus to IR by enabling the LD input of IR.

4. Increment PC by enabling the INR input of PC.

Determine the type of instruction

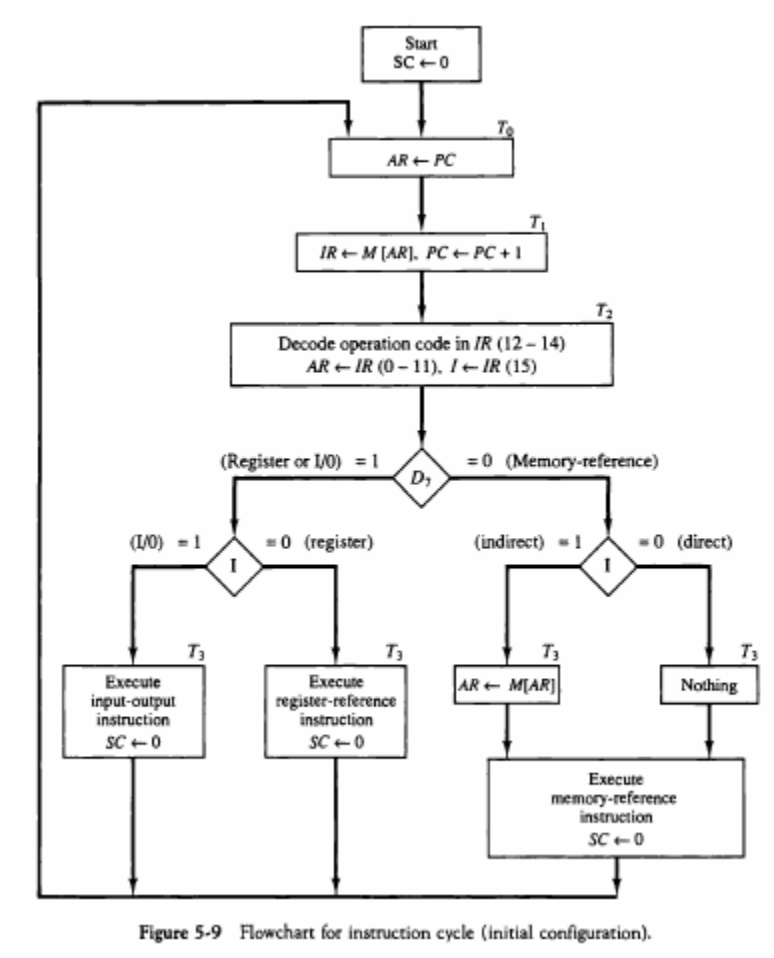


Figure 5-9 Flowchart for instruction cycle (initial configuration).