1020 Test

1. Define microprocessor & draw the block diagram for components
2. Find the decimal value of the 8 bit binary number 10110011
3. Explain Digital to Analog conversion.
4. An 8 bit DAC has V ref of 10V. the binary input is 100011011. Find the analog output voltage.
5. Explain serial interface
6. Write the assembly language program for the following

   Immediate and direct addressing of register A at FF and EE

   Data in A is stored at address 20A2 and another consecutive memory location.

   Load register pair HL with these data

   Store the same data loaded at HL into a two consequent memory locations.

7. Construct the program to add together the ten numbers 1 to 10

   B -register contains the running total. C –register contains the number to be added to the running total. (Start from 1).

   Increment by 1 and add. If count is less than or equal to , it will be repeated, If the count is greater than 10, it will be ended.

8. Write the program.

   Load the desired delay time parameter into C- register. Delay parameter is 02.

   Clear A

   Compare the contents of C – register with zero if yes, end, otherwise proceeds.

   Jump if the time is not set to zero.

   Non operation stages--- 6 stages.

   Execute delay instructions. Decrement C-register

   Jump loop.

9. Explain the controller programming.

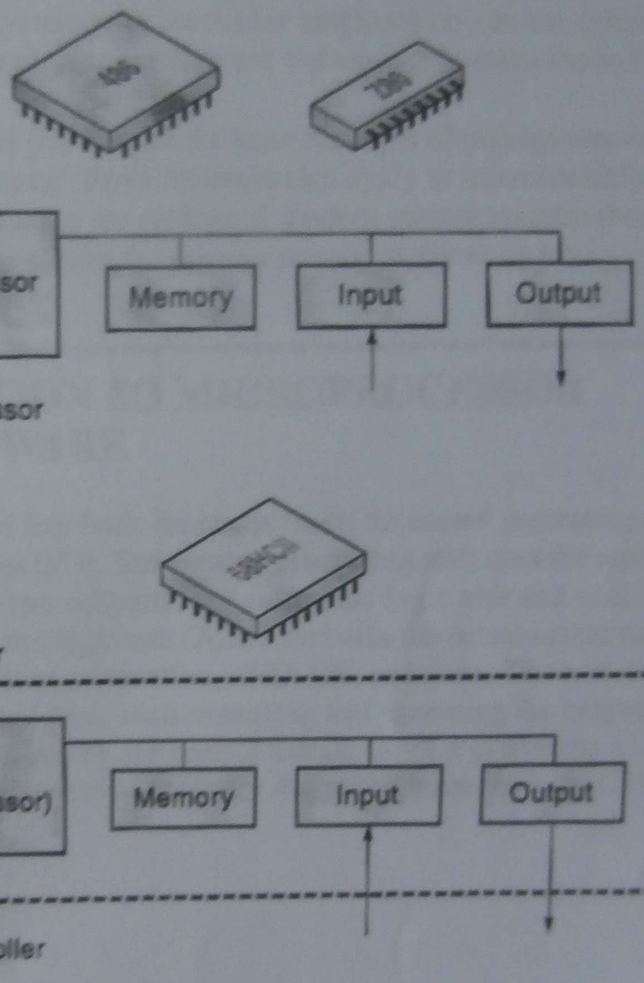


Figure 2.1
Microprocessor and microcontroller.

a close relative of the microprocessor, *does* contain all the computer functions on a single IC. Microcontrollers lack some of the power and speed of the newer microprocessors, but their compactness is ideal for many control applications; most so-called microprocessor-controlled devices, such as vending machines, are really using microcontrollers. Some specific reasons for using a digital, microprocessor design in control systems are the following:

- Low-level signals from sensors, once converted to digital, can be transmitted long distances virtually error-free.
- A microprocessor can easily handle complex calculations and control strategies.
- Long-term memory is available to keep track of parameters in slow-moving systems.
- Changing the control strategy is easy by loading in a new program; no hardware changes are required.
- Microprocessor-based controllers are more easily connected to the computer network within an organization. This allows designers to enter program changes and read current system status from their desk terminals.

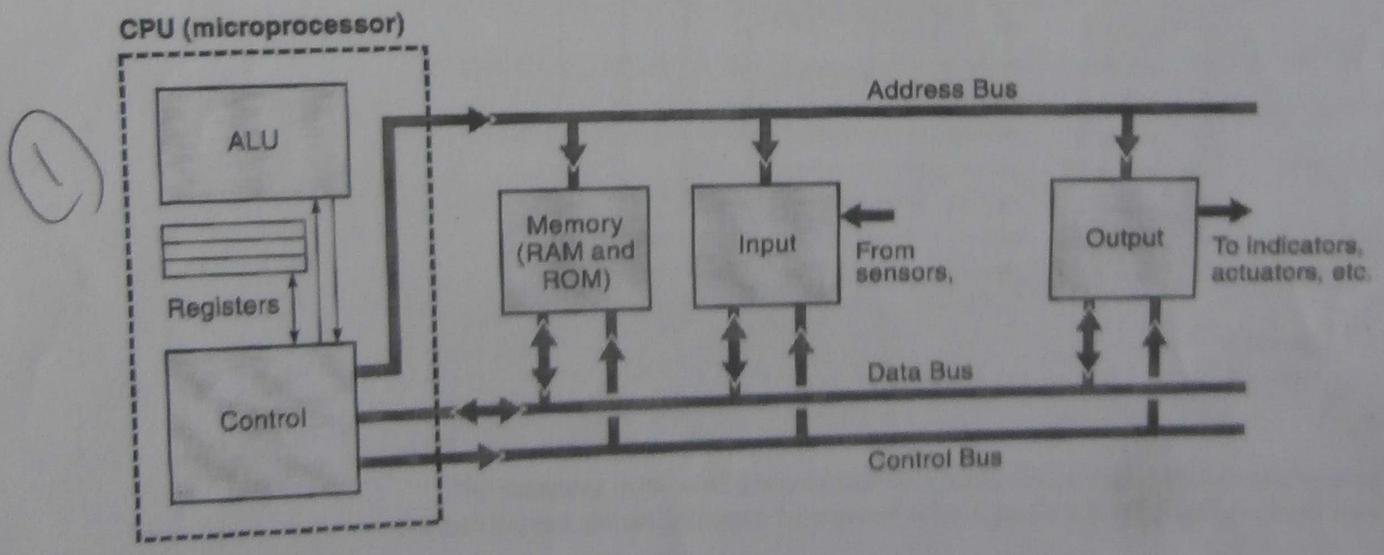
In this chapter, we will present the basic concepts of a microprocessor- and microcontroller-based system with particular emphasis on control system applications. It is by no means an in-depth treatment, but enough to make the rest of the text more meaningful.

In the first sections of this chapter the basic concepts of microprocessor hardware and operation are introduced (these concepts also apply to microcontrollers). I have included this material because the student of modern control systems should have at least a general knowledge of how the microprocessor performs its job.

2.1 INTRODUCTION TO MICROPROCESSOR SYSTEM HARDWARE

A computer is made up of four basic functional units: the central processing unit (CPU), memory, input, and output (I/O). The **central processing unit** does the actual computing and is composed of two subparts: the arithmetic logic unit and control sections (Figure 2.2). The arithmetic logic unit (ALU) performs the actual numerical and logic calculations such as addition, subtraction, AND, OR, and so on. The control section of the CPU manages the data flow, such as reading and executing the program instructions. If data require calculations, the control section hands it over to the ALU for processing. In a microprocessor-based computer, the microprocessor is the CPU.

Figure 2.2
A block diagram of a microprocessor-based computer.



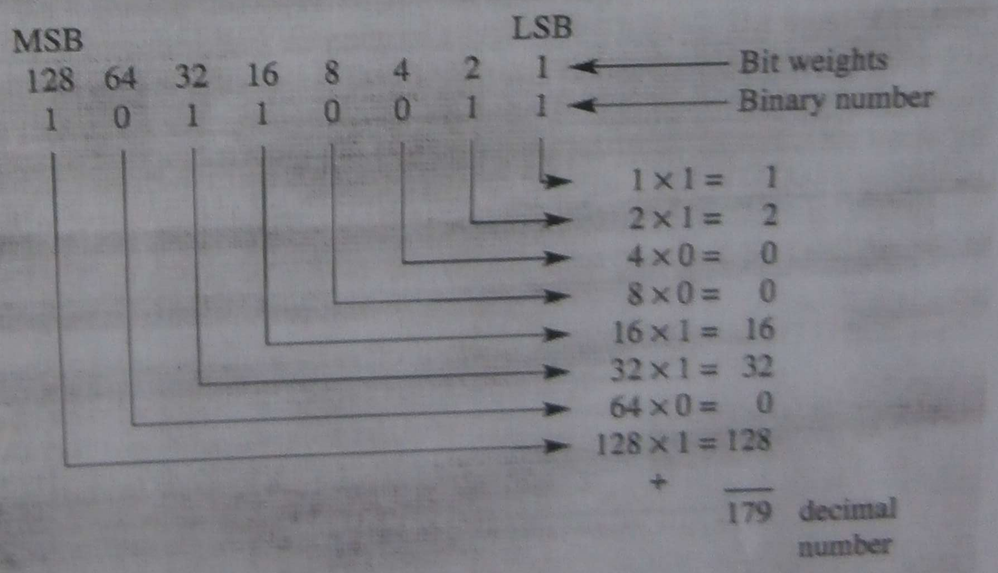
Digital data is in the form of **bits,** where each bit has a value of either 1 or 0. Digital circuits usually use 5 Vdc to represent logic 1 and 0 Vdc to represent logic 0. Eight bits together is called a **byte.** A microprocessor handles digital data in **words,** where a word may be 8, 16, or 32 bits wide. For example, an 8-bit microprocessor has a byte-sized word, with a maximum decimal value of 255. (Computers represent numbers in the *binary number system*; for example, 11111111 binary = 255 decimal.) The rightmost bit in a binary number has the least value (usually 1) and is called the **least significant bit** (LSB). The leftmost bit represents the highest value and is called the **most significant bit** (MSB). The conversion between binary and decimal can be performed directly with most scientific calculators or manually using the technique shown in Example 2.1. To express values larger than 255, two or more words are put together. In this text, we will assume 8-bit microprocessors are used unless otherwise stated.

EXAMPLE 2.1

Find the decimal value of the 8-bit binary number 10110011.

SOLUTION

Each bit in the binary number has a different value, or weight. The LSB has a weight of 1. The bit to the left of the LSB has a weight of 2, the third bit has a weight of 4, and so on, with the weight doubling for each bit up to 128 for the MSB. To find the value of an 8-bit number, you can set up a chart (shown below) and then sum the values that correspond to the 1s in the binary number.



The **memory** section of the computer is a place where digital data in binary form (1s and 0s) are stored. Memory consists of cells organized in 8-bit groups. Each byte

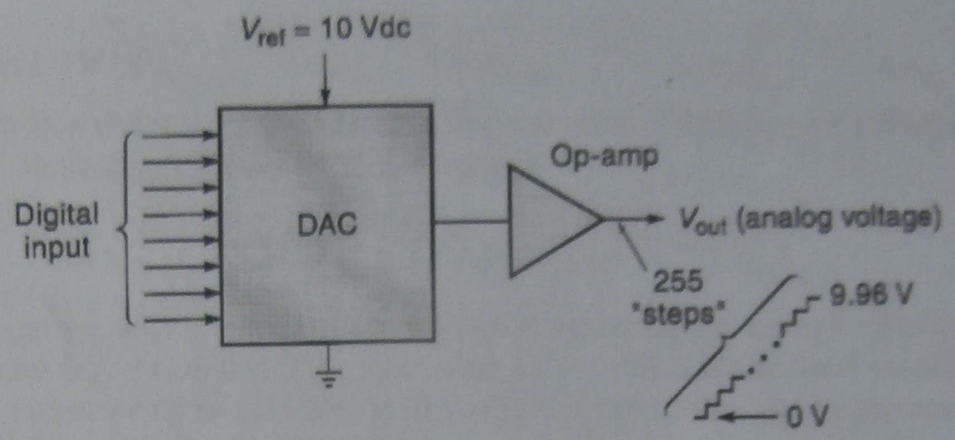


Figure 2.5
A digital-to-analog converter (DAC) block diagram.

In other applications, the controller may use a parallel interface to connect to an analog device—for example, driving a variable-speed DC motor. In such a case, the binary output of the controller must first be converted into an analog voltage before it can drive the motor. This operation is performed by a special circuit called a digital-to-analog converter.

Digital-to-Analog Conversion

The **digital-to-analog converter** (DAC) is a circuit that converts a digital word into an analog voltage. It is not within the scope of this text to describe the internal workings of the DAC, but a general understanding of the operating parameters is appropriate.

Figure 2.5 shows the block diagram of a typical 8-bit DAC. The input is an 8-bit digital word. The output is a current that is proportional to the binary input value and must be converted to a voltage with an op-amp. A stable reference voltage (V_{ref}) must be supplied to the DAC. This voltage defines the maximum analog voltage—that is, for a digital input of 11111111, V_{out} is essentially V_{ref}. If the input is 00000000, the V_{out} will be 0 Vdc. For all values in between, the output voltage is a linear percentage of V_{ref}. Specifically, the output voltage for any digital input (for the 8-bit DAC) is

$$V_{out} = \frac{\text{input} \times V_{ref}}{256} \tag{2.1}$$

where
V_{out} = DAC output analog voltage
input = decimal value of the binary input
V_{ref} = reference voltage to the DAC

EXAMPLE 2.2

An 8-bit DAC has a V_{ref} of 10 V. The binary input is 10011011. Find the analog output voltage.

SOLUTION

The binary input of 10011011 has a decimal value of 155. Applying Equation 2.1, we can calculate the analog output voltage:

$$V_{out} = \frac{\text{input} \times V_{ref}}{256} = \frac{155 \times 10\ \text{V}}{256} = 6.05\ \text{V}$$

Therefore, 6.05 V is the voltage we would expect on the analog output pin. [It is interesting to note that if the input were all 1s (which is a decimal value of 255), the output would be (255/256) × 10 V = 9.96 V, not 10 V as you might expect. This is a characteristic of the DAC.]

An important consideration of digital-to-analog conversion is **resolution.** The resolution of a DAC is the worst case error that is introduced when converting between digital and analog. This error occurs because digital words can only represent discrete values, as indicated by the stair-step diagram in Figure 2.5. For example, the maximum value of an 8-bit number is 255 decimal, which means there are 255 possible "steps" of the output voltage. The difference between steps is the value of the least significant bit (LSB). Because the smallest increment is one step, the resolution (for 8-bit data) is 1 part in 255, or 0.39%. This resolution is adequate for many applications, but if more is needed, two (or more) 8-bit ports can be used together. Two ports provide 16 bits of data. The maximum decimal value of 16 bits is 65,535. Being able to divide an analog number into 65,535 parts means that each part will be much smaller, so we can more precisely represent that number.

E 24/4/12

EXAMPLE 2.3

A computer uses a DAC to create a voltage that represents the position of an antenna. The antenna can rotate 180° and must be positioned to within 1°. Can an 8-bit port be used?

SOLUTION

The resolution required is 1 part in 180. Because 8 bits provide a resolution of 1 part in 255, an 8-bit port is certainly adequate. In fact, we have a choice: We could have the LSB = 1°, in which case the input values would range from 0 to 180, or we could equate 180° with 255, which makes the LSB = 0.706°. The latter makes maximum use of the 8 bits to give a better resolution, but if the system really doesn't need it, the clear, simple relationship of LSB = 1° is desirable.

switches. The limit switches are used as a "back up" to detect it if the load has gone out of its designated range.

Operation of the system proceeds as follows: The controller inputs the data from port 03 to determine if the start (or stop) button has been pressed. If the start button has been pressed, then the set point is read in from port 01 and the digitized sensor data is read in from port 02. Based on its control strategy, the controller outputs to port 00 a binary word representing the motor-control voltage. This digital data is converted to an analog voltage with the DAC. This entire sequence is repeated over and over until the stop button is pushed.

The Serial Interface

In a **serial interface,** the data are sent 1 bit after the other on a single wire. There are a number of good reasons for doing this. First, the cabling is simpler because only two wires are needed (at a minimum), those being "data" and "return." Second, shielding a small group of wires, which is often necessary in an electrically noisy industrial environment, is easier. Third, serial data can make use of existing single-channel data lines such as the telephone system (which may require using a modem). For these reasons, serial data transfer is usually recommended for distances greater than 10-30 ft.

Because data always exist in a parallel form inside the computer, it must be converted to serial data before coming out the serial port. This is accomplished with a special parallel-to-serial converter IC called a **universal asynchronous receiver transmitter** (UART). On the other end of the line, a receiver must convert the serial data back into parallel data, which is done with another UART. Figure 2.10 shows the basic serial data circuit.

Serial data are classified as being either synchronous or asynchronous. *Synchronous data* require that the data bytes be sent as a group in a "package." It is used in sophisticated communication systems that move a lot of data and will not be further discussed here. *Asynchronous data* transfer is the more common (but slower) type of serial transfer and allows for individual bytes to be sent when needed.

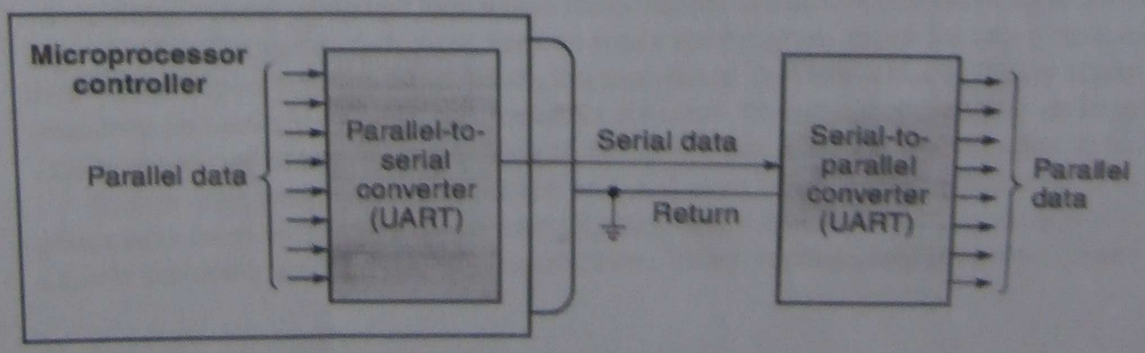


Figure 2.10
Components in a serial interface circuit.

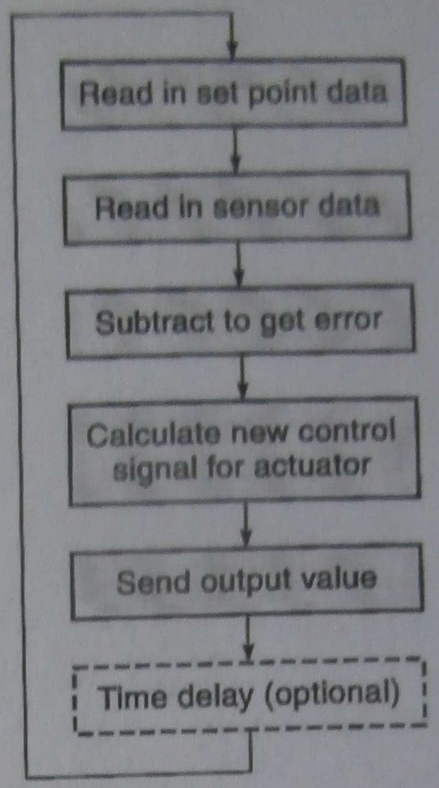


Figure 2.14
A generalized controller program.

2. The program directs the computer to read (from a sensor) the actual value of the controlled variable.
3. The actual data are subtracted from the set point to get the error.
4. Based on the error data, the computer calculates a new actuator control signal.
5. The new output is sent to the actuator.
6. The programs loops back to step 1 and starts over again.

The time it takes for the computer to execute one pass through the loop determines the time interval between input readings (known as the **sampling rate**). If this interval is too long, the computer may not get an accurate picture of what the controlled variable is really doing (see Chapter 11 for a discussion of aliasing). Execution of the loop can be accelerated by specifying a faster computer or streamlining the program. In other situations, the computer must pause and wait. For example, a pause might be inserted to give an operator time to make some adjustment or to allow time for a motor to "spin down." This is done by inserting time-delay loops in the program. A **time-delay loop** is simply a do-nothing, "wheel-spinning" loop where the computer is instructed to count up to some large number. Using this technique, we can make the program pause for any length of time—from a few microseconds to hours. If a time-delay loop is inserted in the main program loop (as shown in Figure 2.14), the effect is to slow the cycle time for the main loop. This is sometimes done to force matching of the sample rate to some predetermined value.

At one time, people thought that the best and most efficient microprocessor programs were those written directly in assembly language—that is, the programmer would directly select the machine language instructions. Today, sophisticated programs (called

DIRECT ADDRESSING (7.3)

PR(7) – Immediate and direct addressing of register A at FF and EE

~~data 20A2 and another consecutive memory location~~

data in A is stored at address 20A2 and another consecutive memory location

Load register pair HL with these data

Store the same data loaded at HL into a two consequent memory locations.

	Assembly Instructions	Actions
1(a)	MVI A, FF	(A) ← FF(hex)
2(a)	STA 20A2	(20A2) ← (A)
1(b)	MVI A, EE	(A) ← EE(hex)
2(b)	STA 20A3	(20A3) ← (A)
3	LHLD 20A2 (load)	(L) ← (20A2) FF (hex) (H) ← (20A3) EE (hex)
4	SHLD 20A4 (store)	(20A4) ← (L) (20A5) ← (H)

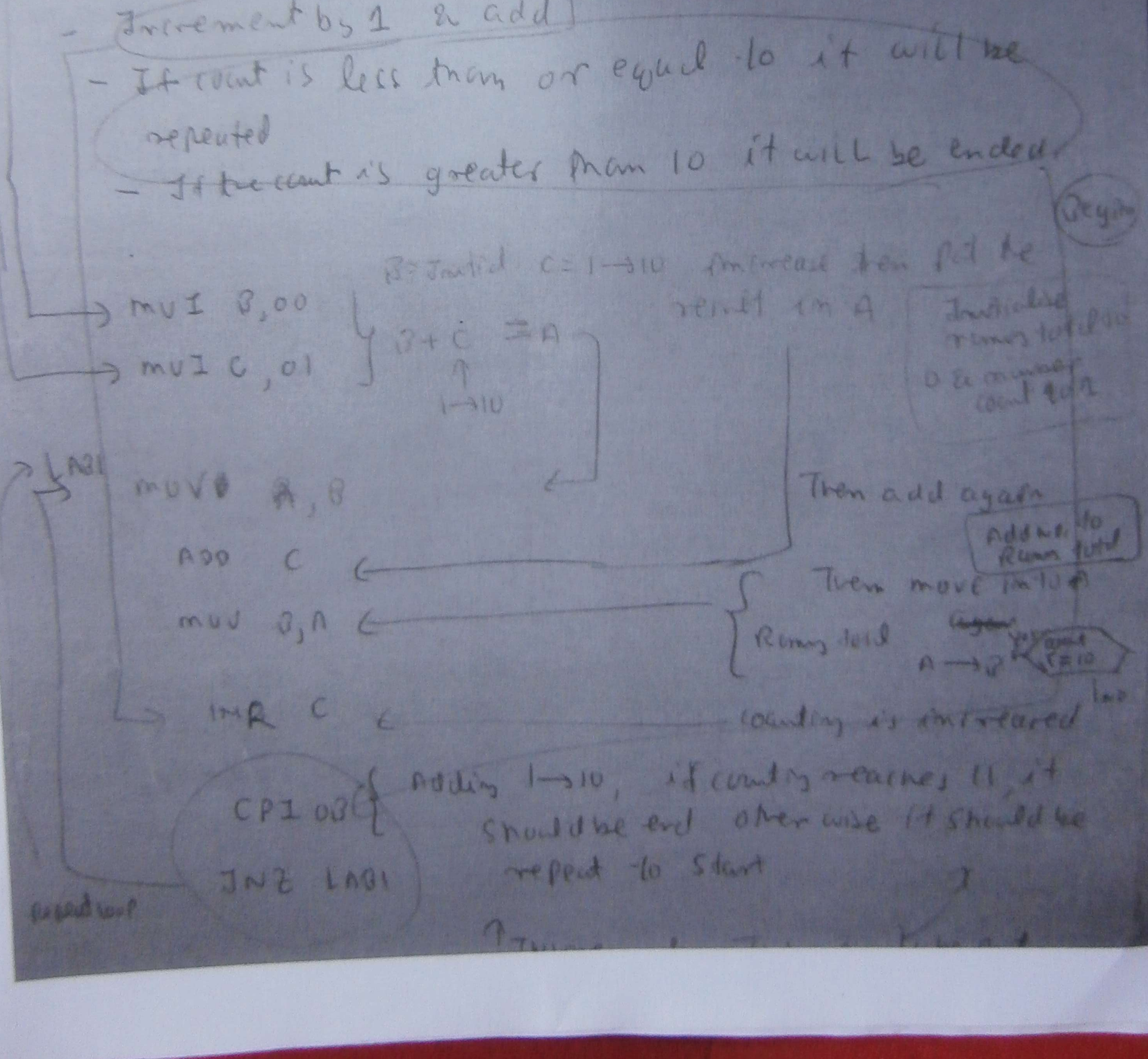
Flow Chart

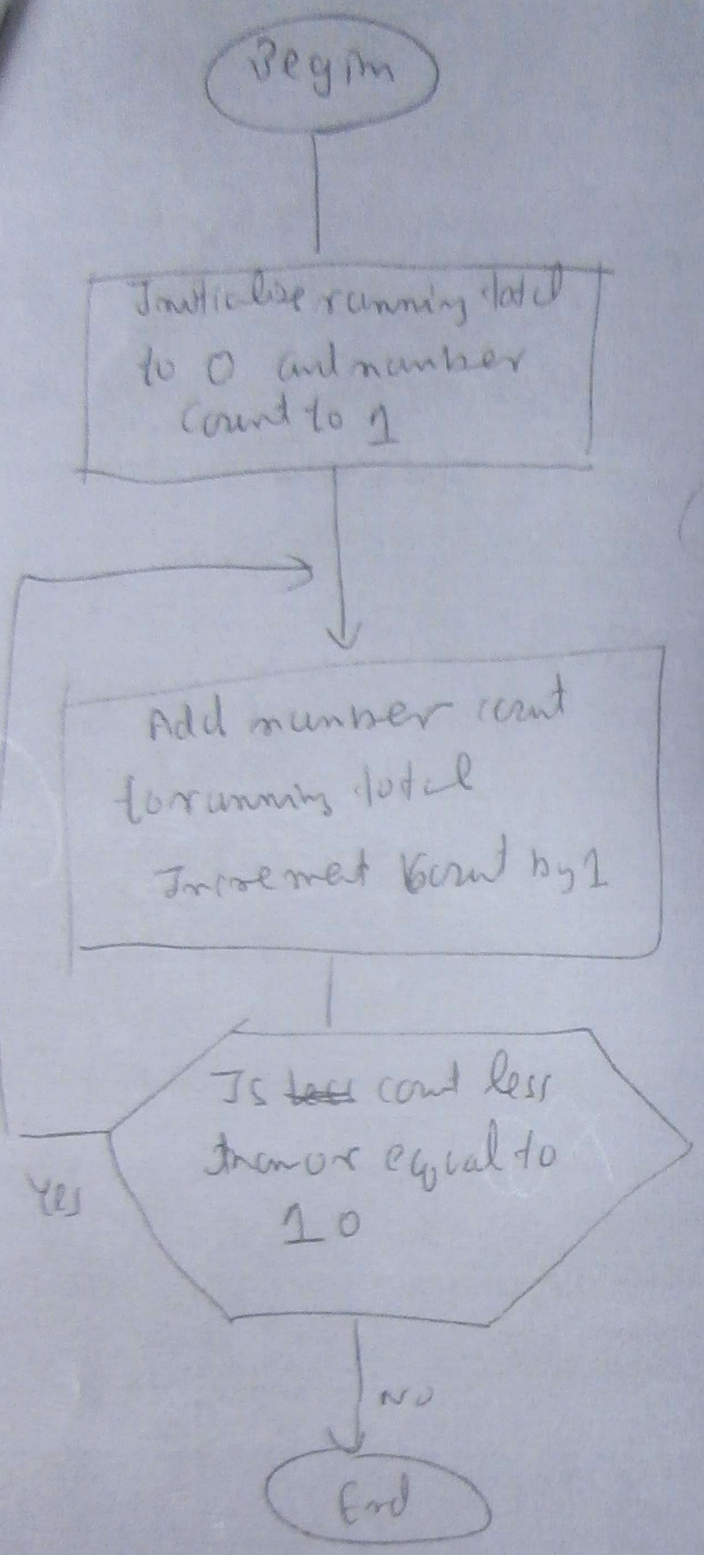
A diagram which indicates the sequence of, and actions required for, a Program and the points where branching is required.

Q7

PD1 Construct the Program to add together the ten numbers 1→10

- B register contains the running total
- C register contains the number to be added to the running total. (~~Initial~~ start from 1)
- Increment by 1 & add
- If count is less than or equal to it will be repeated
- If the count is greater than 10 it will be ended.





	Assembly Instruction	Comment
	MVI B, 00	Initialise running total
	MVI C, 01	Initialise number count
LAB1	MOV A, B	Bring running total to A
	ADD C	Add number count
	MOV B, A	Restore running total in B
	INR C	Increment count
	MOV A, C	Bring count to A
	CPI 0B	Has count reached 11 ?
	JNZ LAB1	No - Jump back to LAB1
		Yes - End total in B

Loop

Jump Instruction

Time delay in the Program

- A common requirement when a microcomputer is interfaced to other equipment is to compute a time delay in the program.
- A microcomputer based road traffic light controller, for example, would need to compute a time delay to implement the sequencing of the light changes
- Desired delay & loop count to be held in C-register.

Time delay NOP - NO-operation instruction

Q) write a program

Ans
1. Load desired delay time parameter into C-register. delay parameter is 02
2. Clear A
3. Compare the contents of C-register with zero. If yes end
4. Jump if not, otherwise proceed.
5. Num operation stages — 6 stages
6. Execute add delay instructions Decrement C register
7. Jump Loop