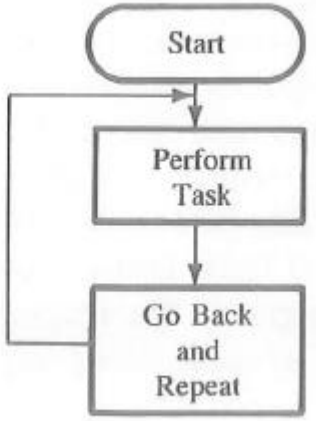
Loop

- The programming technique used to instruct the microprocessor to repat task is called looping
- A loop is set up by instructing the microprocessor to change the sequence of execution and perform the task again
- This process is accomplished by using Jump instruction
- Loops can be classified into two groups
  - Continuous loop – repeat a task continuously
  - Conditional loop – repeat a task until certain data conditions are met

Continuous loop

- A continuous loop is set up by using the unconditional Jump instruction
- A program with the continuous loop does not stop repeating the tasks until the system is reset.

Flowchart of a Continuous Loop

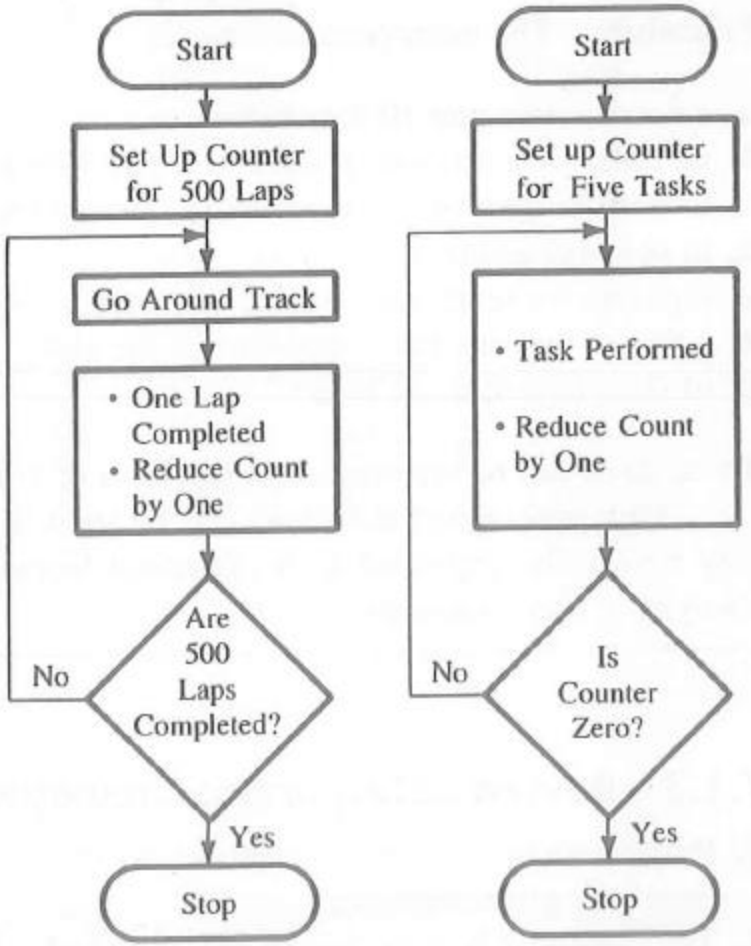


Conditional loop

- A conditional loop is set up by the conditional Jump instruction. These instructions check flags (Zero, Carry, etc.) and repeat the specified tasks if the conditions are satisfied.
- These loops usually include counting and indexing

Conditional loop and counter

- A counter is a typical application of the conditional loop.
- Needed for the microprocessor to repeat the task five time.
- The microprocessor needs a counter, and when the counting is completed, it needs a flag.



(a) Laps in a Car Race (b) Task Repetitions

The computer flow-chart of figure (b) is translated into a program as following

- Counter is set up by loading an appropriate count in a register
- Counting is performed by either incrementing or decrementing the counter
- Loop is set up by a conditional Jump instruction
- End of counting is indicating by a flag

It is easier to count down to zero than to count up because the Zero flag is set when the register becomes zero.

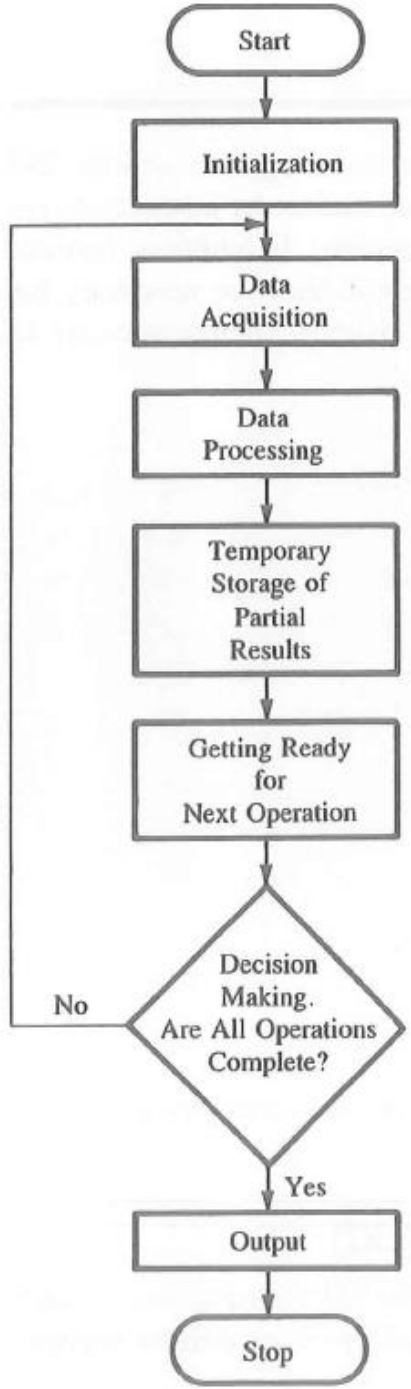
Conditional loop, counter and indexing

- Indexing means pointing or referencing objects with sequential numbers. In a library, books are arranged according to number and they are referred to or sorted by number i.e. called indexing.
- Similarly, data bytes are stored in memory locations, and those data bytes are referred to by their memory locations

Example: illustrate the steps necessary to add ten bytes of data stored in memory locations staring at a given location, and display the sum. Draw a flowchart.

Procedure: the microprocessor needs

- A counter to count 10 data bytes
- An index or a memory pointer to locate where data bytes are stored
- To transfer data from a memory location to the microprocessor (ALU)
- To perform addition
- Registers for temporary storage of partial answers
- A flag to indicate the completion of the task
- To store or output the result



Block 1 **1.** This is a planning stage where the initial setting up is done. This is the most difficult step because the initial setup depends on the requirements of the following steps (steps a and b in Example 7.1: setting up counter and indexing).

Block 2 **2.** Data are generally stored in memory or read from an input port. This step is concerned with bringing data into the microprocessor (step c: transfer of data from a memory location to the microprocessor).

Block 3 **3.** This step involves all data manipulations (step d: addition).

Block 4 **4.** This step involves storing data of partial results (step e).

Block 5 **5.** At this point, the microprocessor does not know whether it has completed all the operations. Before it can repeat the task, it needs to get ready for the next operation; for example, point to next memory location or update count.

Block 6 **6.** This is a decision-making step; it decides whether to change the sequence of execution and repeat or to go to the next instruction (step f: a flag check).

Block 7 **7.** This step involves either sending the result to an output port or storing in memory (step g).

A program has always the following general structure:

Structure of an Assembly Language Program

```
.model small ; Select a memory model.
.stack stack_size ; Define the stack size
.data ; Variable and array declarations; ; Declare variables at this level
.code
main proc ; Write the program main code at this level
main endp ; Other Procedures
    ; Always organize your program
    ; into procedures
end main ;To mark the end of the source file
```

```
include 'emu8086.inc'
.stack 100h
.model small
.data
.code
main proc
   mov cx,10  ; condition
   start:
            print 'BscCSIT'
            mov dl,10       ; new line
            mov ah,02       ; print
            int 21h

            mov dl,13
            mov ah,02
            int 21h

   loop start
   main endp
end main
```