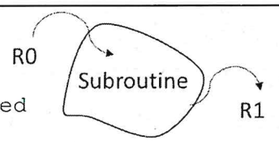
2. (30 pnt) The digit-sum of a 4-digit number is the sum of its individual digits. For example Digit-sum of 1857 is calculated as: 1 + 8 + 5 + 7 = 20.

Task: There are **five numbers (words)** stored in ASCII format in memory under the label **NUMS**. Calculate the digit-sum of each number and store the results in separate memory locations, starting at **SUMS**. Each result should be stored as a word.

Implement a subroutine named **DigitSum** to calculate the digit-sum of one number: Read four consecutive ASCII bytes from memory, starting at the address **pointed to by R0.** Calculate the digit-sum and store the result at the address **pointed to by R1**.
You can use the template below or write your own complete program. **Using a proper subroutine is a must.**



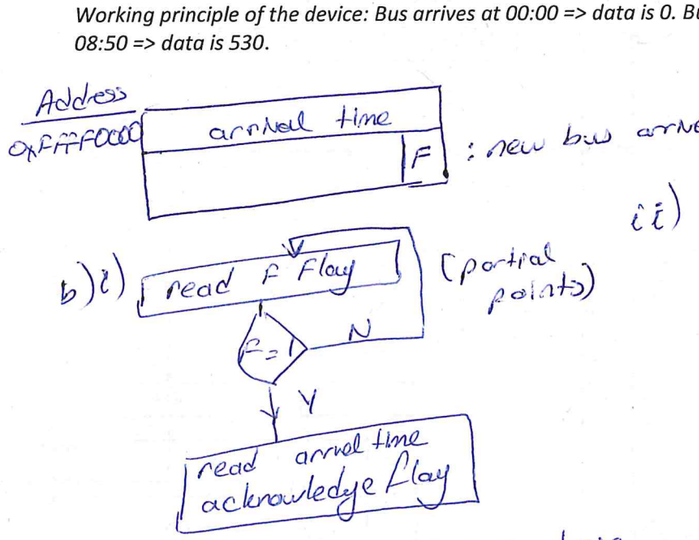

```
.global _start
_start:
MOV R5, #5      // number of numbers to be processed
LDR R1,=SUMS    // address of the digit-sum result.
LDR R0,=NUMS    // address of the first word

Loop: BL DigitSum          // branch to the subroutine.

   SUBS R5, R5, #1         // decrement the number of numbers to be
                           // processed.
   BEQ End                 // exit loop if 5 numbers are processed.

   ADD R0, R0, #4          // update the number pointer.

   ADD R1, R1, #4          // update the SUMS pointer.


   B Loop                  //unconditional branch to loop.
End: B End                 // stay here after all is completed.

DigitSum:      // copy certain registers into stack!
   PUSH {LR, R0, R1, R2, R3}
   LDRB R2, [R0], #1   // read byte 1
   AND R2, R2, #F      // mask
   LDRB R3, [R0], #1   // byte 2
   AND R3, R3, #F
   ADD R2, R2, R3      // ones + tens
   LDRB R3, [R0], #1   // b3
   AND R3, R3, #F
   ADD R2, R2, R3      // ones + tens + hunds.
   LDRB R3, [R0]       // b4
   AND R3, R3, #F
   ADD R2, R3, R2      // 0 + t + h + t
   STR R2, [R1]
   POP {PC, R0, R1, R2, R3}
            // return back from the subroutine properly!
NUMS: .word 0x31383537, 0x39373635, 0x39363335, 0x39393939, 0x30383038
SUMS: .word 0x0, 0, 0, 0, 0

.end
```

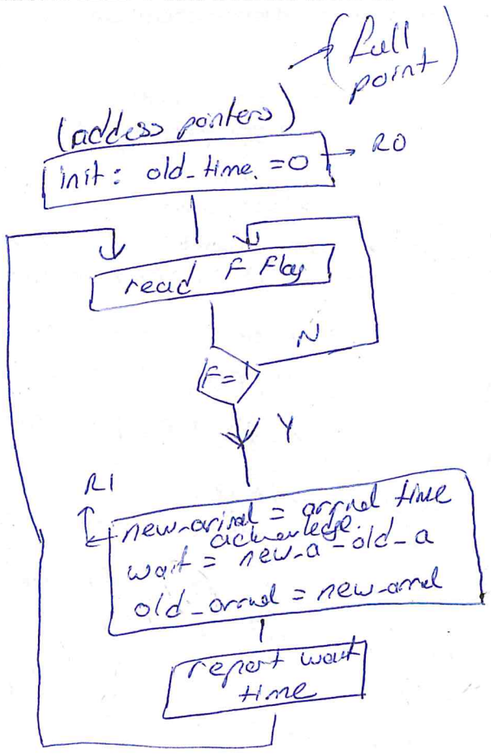
12 pnt

6 x 3
* PUSH - POP
* LR → PC (or BX LR)
* read mem.
* mask
* accumulate
* store sum.

3. (30 pnt) A university-metro station shuttle bus is ideally scheduled to arrive every 15 minutes. A device samples the arrival time of each bus in minutes and provides this data to a microprocessor.
a) Design the device interface and assign a base address to the device. Draw the device registers.
b) Design a microprocessor program to:
   i. Read Arrival Times: Continuously poll the device and read the arrival time each time a new bus arrives.
   ii. Report Wait Time: Calculate and record the wait time between two consecutive buses in the memory.

Write a flowchart for the program. Write an assembly program that performs the required operations. Assume the first bus arrives at 0.
Working principle of the device: Bus arrives at 00:00 => data is 0. Bus arrives at 01:15 => data is 75. Bus arrives at 08:50 => data is 530.






```
LDR R2, =0xFFF0000 // device base
LDR R3, =Memory    // for records.
MOV R0, #0         // init old_arrival

WaitBus: LDR R4, [R2, #4]
         ANDS R4, #1          // mask 1
         BEQ WaitBus

         STR R4, [R2, #4]//acknowledge
         LDR R1, [R2]         // new arrival time

         SUB R5, R1, R0       // wait time
         STR R5, [R3], #4     // record wait time
         MOV R0, R1           // new_arrival becomes old.

         B WaitBus.
```

grading: device - 8 (address - data - flag (status) - control is opt.) (12)
flowchart - (init - read F - ◇ - statement - loop) - 10
code - (init - waitloop - diff and record - old/new) 12 pnt.
3 3 3 3