Лабораторна робота №3

ФІ-91 Житкевич Іван, Рубан Денис

Завдання

Дано натуральні числа $n, a_1, a_2, \dots, a_n$. Отримати середнэ арифметичне тих членів послідовності $a_1, \dots, a_n$, які при діленні на 5 дають залишок 1 чи 2.

В секції .data було створено константний масив as з чисел та n - кількість елементів в цьому масиві

```
.data

n:  .word 5

as: .word 13, 42, 73, 7, 5
```

На початку програми записали в регістр r4 адрес константи n, а потім в r5 значення кількості елементів. Потім в r4 адресу на перший елемнт масиво. Також ініцілізували r7, r8 із значенням 0 для зберігання суми відповідних чисел і кількості сумандів.

```
cycleinit:
  ldr r4, =n
  ldr r5, [r4]      @ load number of iterations
  ldr r4, =as       @ load pointer to first element
  mov r7, #0        @ store the sum here
  mov r8, #0        @ store number of summands
```

В циклі на кожній ітерації ми віднімаємо 1 від кількості елемнтів і проходимся по елементам масиву, робляи зсув на 4 байти. Далі викликається процедура divie, яка ділить кожний елемнт на 5 і повертає ціле число та остачу від ділення, нам потрібна остача, яку ми перевіряємо чи дорівнює вона 1 або 2. Перевірка відбуваєтсья таким чином: від остачі віднімається 1, і далі перевіряється, чи отримане число менше ніж 1. Якщо 2, то 2-1=1, якщо 1, то 1-1=0. (Якщо 0, то 0 - 1 > 1, тому що ми вважаємо число беззнаковим).

```
cycle:
  cmp r5, #0
  sub r5, r5, #1
  beq endcycle      @ break if all elements are checked
  ldr r6, [r4], #4  @ check current element and r4 is updated automatically
  mov r0, r6
  mov r1, #5
```

```
  bl divide         @ the remainder is stored in r0, divisor stored in r1
  sub r0, r0, #1
  cmp r0, #1
  addls r7, r7, r6
  addls r8, r8, #1
  b cycle
```

В процедурі `divide` ми отримуємо 2 аргументи в регістрах r0, r1, r0 - число, яке ділиться, r1 - дільник. В стек пушимо адресу входу у процедуру. Далі від r0 віднімаємо дільник, допоки він не стане меньше дільника та додаємо 1 в регістр r2.

```
# take the number in r0
divide:
          push {lr}
          mov r2, #0 @r0 for remainder, r1 for divisor and r2 for quotient
dividec:  cmp r0, r1
          poplt {pc}
          sub r0, r0, r1
          add r2, #1
          b dividec
```

На кінець циклу обраховуємо ціле та остачу від ділення суми елементів процедурою divide

```
endcycle:
  mov r0, r7
  mov r1, r8
  bl divide
```

Та виходимо з програми викликаючи переривання 0

```
quit:
  mov r7, #0x1
  mov r0, #0x0
  swi 0
```