// **1. Program to find out the sum of an array of integers.**

**Address Instructions Comment**

```
0000   RD      R5    Inpt       // Read the no. of integers to be added from the input buffer
0004   MOVI    R6    0          // Set a counter to reg-6 and initialize to 0
0008   MOVI    R1    0          // Set the Zero register to its value
000C   MOVI    R0    0          // Clear Accumulator
0010   LDI     R10   Inpt       //  Load address of input buffer into reg 10
0014   LDI     R13   Temp       //  Load address of temp buffer into reg 13
0018   LOOP1 :ADDI   R10   4 // Point to the next address of input buffer by adding 4
001C   RD      R11   (R10)      // Load  the content(data) of address in reg-10 in reg-11
0020   ST      (R13) R11        // Store the data in the address pointed to by reg-13
0024   ADDI    R13   4          //  Point to the next address of temp buffer
0028   ADD I   R6    1          // Increment the counter
002C   SLT     R8    R6  R5     // Set reg-8 to 1 if  reg-6 < reg-5, and 0 otherwise
0030   BNE     R8    R1  LOOP1  // Branch  if  content of Reg- 8 and Reg-1 is not equal
0034   MOVI    R6    0          // Reset the counter to Zero

0038   LDI     R9    Temp       // Loading the address  temp into reg 9
003C   LOOP2: LW   R7  0(R9)  // Loads the content of the address in reg-9 in reg-7 , reg-9 is
                                //  B-reg .  0 is the offset
0040   ADD     R0    R0   R7  //  Add  the content of  accumulator  with reg-7 and stored in acc.
0044   ADDI    R6    1          //  Incrementing the counter by 1
0048   ADDI    R9    4          //  Incrementing  the B-register  by 4 bytes
004C   SLT     R8  R6  R5       // Reg-8  is set to 1  ,if Reg6 < Reg5, and 0 otherwise
0050   BNE     R8   R1   LOOP2  // Branch  if  content of Reg- 8 and Reg-1 is not equal
0054   WR      R0    Oupt       //  Write the content of the aacumulator into output buffer
0058   HLT                      // Logical end of program
//Data segment starts
005C   (beginning address of Inpt 'Data')
```

---

// **2. Program to find out the largest of the integers in an array**

```
0000   RD      R5    Inpt       // Read the no. of integers to be added from the input buffer
0004   MOVI    R6    0          // Set a counter to reg-6 and initialize to 0
0008   MOVI    R1    0          // Set the Zero register to its value
000C   MOVI    R0    0          // Clear Accumulator
0010   LDI     R10   Inpt       //  Load address of input buffer into reg 10
0014   LDI     R13   Temp       //  Load address of temp buffer into reg 13
0018   LOOP1:ADDI R10  4     // Point to the next address of input buffer by adding 4
001C   RD      R11   (R10)      // Load  the content(data) of address in reg-10 in reg-11
0020   ST      (R13) R11        // Store the data in the address pointed to by reg-13
0024   ADDI    R13   4          //  Point to the next address of temp buffer
0028   ADD I   R6    1          // Increment the counter
```

```
002C   SLT      R8    R6  R5        // Set reg-8 to 1 if  reg-6 < reg-5, and 0 otherwise
0030   BNE      R8    R1   LOOP1  // Branch  if  content of Reg- 8 and Reg-1 is not equal
0034   MOVI   R6   0                 // Reset the counter to Zero
0038   LDI      R9      Temp        // Loading the address  temp into reg 9
003C   LW       R0     0(R9)         // Loads the content of the address in reg-9 in acc, reg-9 is
                                     //  B-reg .  0 is the offset
0040   ADDI    R6    1               //  Incrementing the counter by 1
0044   ADDI    R9    4               //  Incrementing  the B-register  by 4 bytes to point to next
element
0048 LOOP:    LW       R2  0(R9) // Loads the content of the address in reg-9 in reg-2 , reg-9 is
                                     //  B-reg .  0 is the offset
004C   ADDI   R6    1               //  Incrementing the counter by 1
0050   ADDI    R9    4               //  Incrementing  the B-register  by 4 bytes
0054   SLT      R8  R0  R2          // Reg-8  is set to 1  ,if Reg0 < Reg2, and 0 otherwise
0058   BEQ     R8   R1   LOOP2  // Branch  if  content of Reg- 8 and Reg-1 is  equal
005C   MOV     R0   R2              // Moves the larger no. in reg-2 to accumulator
0060   LOOP2: SLT      R8  R6  R5          // Reg-8  is set to 1  ,if Reg6 < Reg5
0064   BNE      R8   R1   LOOP    // Branch  if  content of Reg- 8 and Reg-1 is  equal
0068   WR       R0    Oupt          //   Write the content of the aacumulator into output buffer
006C   HLT                           // Logical end of program

// Data segment starts
0070  (beginning address of Inpt 'Data')
```

---

**// 3. Program to find the average of an array of numbers.**

```
0000   RD       R5     Inpt          // Read the no. of integers to be added from the input buffer
0004   MOVI   R6     0             // Set a counter to reg-6 and initialize to 0
0008   MOVI   R1     0             // Set the Zero register to its value
000C   MOVI   R0     0             // Clear Accumulator
0010   LDI      R10   Inpt          //  Load address of input buffer into reg 10
0014   LDI      R13   Temp        //  Load address of temp buffer into reg 13
0018    LOOP1 :ADDI    R10   4 // Point to the next address of input buffer by adding 4
001C   RD       R11    (R10)       // Load  the content(data) of address in reg-10 in reg-11
0020   ST       (R13)  R11        // Store the data in the address pointed to by reg-13
0024   ADDI    R13  4            //  Point to the next address of temp buffer
0028   ADD I   R6   1             // Increment the counter
002C   SLT      R8   R6  R5      // Set reg-8 to 1 if  reg-6 < reg-5, and 0 otherwise
0030   BNE      R8   R1  LOOP1  // Branch  if  content of Reg- 8 and Reg-1 is not equal
0034   MOVI   R6   0                 // Reset the counter to Zero
0038   LDI      R9      Temp        // Loading the address  temp into reg 9
003C LOOP:   LW       R7    0(R9) // Loads the content of the address in reg-9 in reg-7 , reg-9
is
                                     //  B-reg .  0 is the offset
0040  ADD    R0    R0   R7    //  Add  the content of  accumulator  with reg-7 and stored in acc.
```

0044 ADDI R6 1 // Incrementing the counter by 1
0048 ADDI R9 4 // Incrementing the B-register by 4 bytes
004C SLT R8 R6 R5 // Reg-8 is set to 1 ,if Reg6 < Reg5, and 0 otherwise
0050 BNE R8 R1 **LOOP** // Branch if content of Reg- 8 and Reg-1 is not equal
0054 DIV R0 R0 R5 // Finding the average ,keeps in R0
0058 WR R0 ***Outpt*** // Write the content of the aacumulator into output buffer
005C HLT // Logical end of program

0060 – 00AC ->Input buffer (beginning address of ***Inpt*** 'Data')
00B0 – 00DC -> Output buffer (beginning address of ***Outpt*** 'Data')
00E0- 010C -> Temp buffer (beginning address of ***Temp*** 'Data')

---

**// 4. Program to find out the sequence of Fibonacci number**

0000 RD R5 ***Inpt*** // Read the number of Fibonacci number to find out
0004 MOVI R6 0 // Initializing reg-6
0008 MOVI R0 0 // Clear the accumulator
000C MOVI R1 0 // Set the Zero register to its value
0010 MOVI R2 0 // Putting the first Fibonacci number in reg-2
0014 MOVI R3 1 // Putting the second Fibonacci number in reg-2
0018 LDI R7 ***Outpt*** // Loading the output buffer address into reg-7
001C WR R2 (R7) // Output the first Fibonacci number
0020 ADDI R7 4 // Pointing to the next location of output buffer
0024 ADDI R6 1 // Increment the counter
0028 **LOOP** : ADD R0 R3 R2 // Find the next Fibonacci number
002C WR R0 (R7) // Print the number
0030 ADDI R7 4 // Pointing to the next location of output buffer
0034 ADDI R6 1 // Increment the counter 1
0038 MOV R2 R3 // Moving the last but one number into reg-2
003C MOV R2 R0 // Moving the last number into reg-3
0040 SLT R8 R6 R5 // Reg-8 is set to 1 ,if Reg6 < Reg5, and 0 otherwise
0044 BNE R8 R1 **LOOP** // Branch if content of Reg- 8 and Reg-1 is not equal
0048 HLT // Logical end of program

004C – 0098 -> input buffer (beginning address of ***Inpt*** 'Data')
009C – 00B8 -> output buffer (beginning address of ***Outpt*** 'Data')
00BC – 00E8 -> temp buffer (beginning address of ***Temp*** 'Data') - not relevant in Fibo code

---