# Assembly programming
# Proposed exercises

**Exercise 1**. Given the following expression of a high-level language:

```
int a = 6;
int b = 7;
int c = 3;
int d;
d = (a+b) * (a+b);
```

Write a fragment of RISC-$V_{32}$ assembly code that allows to evaluate the previous expression. The result is stored in the t5 registry.

**Exercise 2.** Write an RISC-$V_{32}$ assembly program to calculate the sum of the first 100 natural numbers. The program should leave the result in the a0 register.

**Exercise 3**. Modify the above program to print the result on the screen.

**Exercise 4.** Write a program that reads two integers A and B and indicate if one of them is a multiple of the other.

**Exercise 5**. Write an RISC-$V_{32}$ assembly program that reads an N number and displays the following:

1
1 2
1 2 3
1 2 3 4
…..
1 2 3 4 5 …. N

**Exercise 6**. Write the sequence of RISC-$V_{32}$ instructions needed to execute the following sentence in C language (assuming that a and b are int variables):

a = b + c + 100;

**Exercise 7**. Write an assembly program that reads two numbers. The program should print the larger of the two.

**Exercise 8.** Write an RISC-$V_{32}$ assembly program that reads a number and indicates whether the number is odd or even.

**Exercise 9.** Write a RISC-$V_{32}$ function that receives in the register a0 the starting address of a string, in the register a1 the ASCII code of a character and in the register a2 the ASCII code of another one. The function must replace in the string all occurrences of the character stored in a1 by the character stored in a2.

**Exercise 10**. Consider a function called *func* that receives three integer parameters and returns an integer type result, and consider the next fragment of the data segment:

```
.data
      a: .word 5
      b: .word 7
      c: .word 9
.text
```

Write the necessary code to call the previous function by passing as parameters the values of the memory locations a, b and c. Once the function is called, the value returned by the function must be printed.

**Exercise 11**. Given the following program fragment:

```
.data
      a: .word 10
      b: .word 5
.align 2
      v: .space 800
.text
. . .
```

If v represents an array of integers. Indicate:

a) The number of elements in the vector.
b) Indicate the assembly instructions needed to execute the following high-level sentence v[20] = v[30];
c) Write the instructions in assembly necessary to execute: v[10] = b;

**Exercise 12**. Given the following code fragment written in C, write, using the RISC-$V_{32}$ the code of the equivalent function.

```
int max(intA, int B)
{
      if (A > B)
            return A;
      else
            return B;
}
```

Using the above assembly function, implement the code for the following function using the RISC-$V_{32}$ assembly language.

```
void maxV (int v1, int v2, int v3, int N)
{
      int i;
      for (i = 0; i < N; i++)
            v3[i] = max(v1[i], v2[i]);
      return;
}
```

For the development of this exercise, the convention of passing parameters seen in the subject's course must be followed.

**Exercise 13.** Given the following program fragment, write an equivalent program using the RISC-V32 assembly.

```
int vector[1024];  // global variable

void funcion1 ( void )
{
   int z;
   int i;

   for (i = 0; i < 1024; i++) {
        vector[i] = i;
   }

   z = add(1024);
   print_int(z);
}

int add ( int n )
{
     int s = 0;
     int i;

     for (i = 0; i < n; i ++) {
          s = s + vector[i];
     }

     return s;
}
```

**Exercise 14.** Given the following program fragment, write an equivalent program using the RISC-V32 assembly.

```
void funcion1 ( void )
{
  int z;
  int vector[1024];     // local variable

  for (i = 0; i < 1024; i++) {
     vector[i] = i;
  }

  z = add(vector, 1024);
  print_int(z);
}

int add ( int vector[],  int n )
{

     int s = 0;
     int i;

     for (i = 0; i < n; i ++) {
          s = s + vector[i];
     }

     return s;
}
```

**Exercise 15.** Consider a 32-bit computer with a set of 140 machine instructions and a register file with 64 registers. Consider the following machine instruction: add R1, R2, address. The instruction adds the contents of register R1, with the contents of register R2 and stores the result in the memory position given by address. It is requested:

a) Indicate in a reasoned way a possible format for the previous instruction.
b) Using the RISC-$V_{32}$ assembly language, indicate a code fragment equivalent to the previous instruction.

**Exercise16.** Given a 16-bit computer, which addresses the memory by bytes and includes a repertoire of 60 machine instructions. The register file includes 8 registers. Indicate the format of the ADDV instruction R1, R2, M, where R1 and R2 are registers and M is a memory address.

**Exercise 17.** Given a 32-bit computer, which addresses the memory by bytes. The computer includes 64 machine instructions and 128 registers. Consider the SWAPM instruction addr1, addr2, that exchanges the content of the memory positions addr1 and addr2. It is requested:

a) Indicate the addressable memory space on this computer.
b) Indicate the format of the previous instruction.
c) Specify an assembly program fragment of the RISC-$V_{32}$ equivalent to the previous machine instruction.
d) If the instruction is forced to fit into one word, what range of directions could be contemplated considering that the directions are represented in binary.

**Exercise 18**. Given the following section of data from an assembly program

```
.data:
      .align 2                  # aligns the following data to a limit of 2^2
      vector:   .space 1024     # space reserved for a vector of integers of 32 bits
```

Answer:

a) The number of elements in the vector**.**
b) Indicate the instructions in assembly needed to execute the following sentence of a high level language: `vector[8] = 30;`
c) Write a fragment of assembly code that reads an integer and assigns that integer to all components of the previous vector.

**Exercise 19**. Given the following assembly program fragment:

```
      .text
      main:       li   a0, 5
                  jal  ra, function
                  li   a7, 1
                  ecall

                  li   a7, 10
                  ecall

      function:   mv   t0, a0
                  li   t1, 0
         loop1:   beq  t0, x0, end1
                  add  t1, t1, t0
                  addi t0, t0, -1
                  j    loop1
          end1:   mv a0, t1
                  jr   ra
```

a) Indicate in a reasoned way the value that is printed on the screen (first call to the system of the previous code).
b) If in the register a0, which is used to pass parameters to the function, the value that is stored is represented in one's complement, what range of numbers could be passed to the function?

**Exercise 20**. Consider a function called Vowels. This function is passed as a parameter the start address of a string. The function calculates the number of times the character 'a' (in lower case) appears in the string. In case of passing the null string the function returns the value -1. If the string has no 'a', the function returns 0.

a) Program using the RISC-V32 assembly the function code.
b) Which register is used to pass the argument to the function and in which register the result will be stored?
c) Given the following fragment:

```
.data
        cadena:   .asciiz    "hello"
.text

 main:
```

include in the main above, the necessary sentences to invoke the Vowels function implemented in section a) and print on the screen the value returned by the function. The objective is to print the number of times the character 'a' appears in the string "Hello".

**Exercise 21**. Given the following fragment:

```
.text
 main:            li   a0, 5
                  jal  ra, function
                  li   a7, 1
                  ecall

                  li   a7, 10
                  ecall

function:         li  t0, 10
                  bgt a0, t0, et1
                  li  t0, 10
                  add t1, t0, a0
                  j   et2
      et1:        li  t0, 8
                  add t1, t0, a0
                   mv  a0, t1
      et2:        jr  ra
```

Indicate in a reasoned way the value that is printed on the screen (first call to the system of the previous code).

**Exercise 22.** Consider a function called AddValue. Three parameters are passed to this function:

- The first parameter is the starting address of a vector of integers.
- The second parameter is an integer value that indicates the number of components of the vector.
- The third parameter is an integer value.

The AddValue function modifies the vector, adding the last value as a third parameter to all the components of the vector. You are asked to:

a) Indicate in which registers each of the parameters should be passed to the function.
b) Write a program using the RISC-V32 assembly the code of the SumValue function.
c) Given the following fragment:

```
.data
        v:  .word    7, 8, 3, 4, 5, 6
.text

 main:
```

include in the main function, the necessary assembly sentences to be able to invoke the AddValue function implemented in section b) so that it adds to the components of vector v defined in the data section, the number 5. Implement after the call function, the assembly statements that allow to print all the components of the vector.

**Exercise 23**. Consider a 32-bit computer with 48 registers and 200 machine instructions. Indicate the format of the hypothetical instruction beq x0 t1, t2, address where t1 and t2 are registers and address represents a memory address

**Exercise 24** . Consider a 64-bit computer that addresses the memory by bytes, with 256 registers and an instruction repertoire composed of 190 instructions. Given the following three instructions

- `LOAD Reg, addr`, load in `Reg` the value stored in addr.
- `STORE Reg1, (Reg2)`, store in register `Reg1` the data store in position stored in `Reg2`.
- `BEQZ Reg1, addr`, conditional jump instruction. If the Reg1 content is zero, the next instruction to be executed will be the one stored in `addr`.

Answer:

a) Indicate the addressing mode or modes present in each of the above instruction.
b) Indicate a possible format for each of the above instructions, assuming that in this computer all the instructions occupy one word.
c) According to the format indicated in the previous section, what is the maximum value of the address that can be specified in the instruction LOAD and BEQZ?

**Exercise 25**. Consider the following function of a high-level language:

```
float func(float A[ ], int N, float x)
{
      int i;
      for (i = 0; i < N; i = i + 1)
            A[i] = x;
      return 0.0;
}
```

Where A represents an array of float type numbers and N the number of elements in the array. Write a function in assembly that implements the same functionality, using the correct convention of parameters passage seen in class.

**Exercise 26**. Write an assembly program that reads three numbers (A, B, and C) and codes the following program fragment:

```
if (A == 2 &&  B == 3 || C != 4)
      Print (1);
else
      Print (2);
```

**Exercise 27**. Translate to assembly the following functions:

```
inf f (int k)
{
      int v[100];
      int i = 0;
      int s = 0;

      for (i = 0; i < k; i = i +2)
            v[i] = g(k + 2);

      for (i = 0; i < k; i = i + 2)
            s = s + v[i];

      return (s);
}
```

```
int g(int k)
{
            if (k % 2 == 0)
                  return (k * k + 2);
            else
                  return k * (-1);
}
```