IST-ASM Final Exam — Fall 2023

Name:

- First, write your name in the box above. Then, have a quick read through all 7 questions.
- In the end, you will write up your answers on this paper.
  - But please make a draft elsewhere first. Only hand in something readable. Really.
- This is an open-book open-laptop exam: you may work on scrap paper and/or on your screen.
- Each question is independent from others, except stated otherwise.

Question 1 Perform the binary addition 53+82: convert both numbers to binary, then compute the sum entirely in binary. Show the details of your work.

53 in binary

82 in binary

addition

```
    110101
+  1010010
----------
  10000111      = 128 + 7 = 135
```

Question 2 Convert the program below to ASM syntax

machine code (hex)

```
00  2030003a
04  204001f4
08  3243000c
0C  10504000
10  20ff0000
14  10503000
```

⟷

source program (asm)

```
leti R3, 58
leti R4, 500
blt R4, R3, +12
add R5, R0, R4
halt
add R5, R0, R3
halt
```

Explain the purpose of this code using a simple sentence:

R5 := MAX(R3, R4)

Question 3 In the table below, encode your last name in ASCII (if some letters are missing, use the closest equivalent e.g. É↦E). Write each byte as a hexadecimal number (i.e. “42” will be read as 0x42, not “decimal 42”).

Letter												
ASCII (hex)												

SALAGNAC = 53 41 4C 41 47 4E 41 43
MOREL = 4D 4F 52 45 4C

Question 4 Write a program which computes the average of four integers initially stored in R1 to R4, and stores the result in R5. For instance, if R1=50, R2=10, R3=70, and R4=30, then the program should calculate R5=40. We are not interested in fractional digits: the average of 50, 11, 71 and 31 is also 40. However the average of 51, 11, 71, and 31 is 41.

```
leti R1, 50
leti R2, 11
leti R3, 71
leti R4, 31

add R1, R1, R2
add R1, R1, R3
add R1, R1, R4
divi R5, R1, 4

halt
```

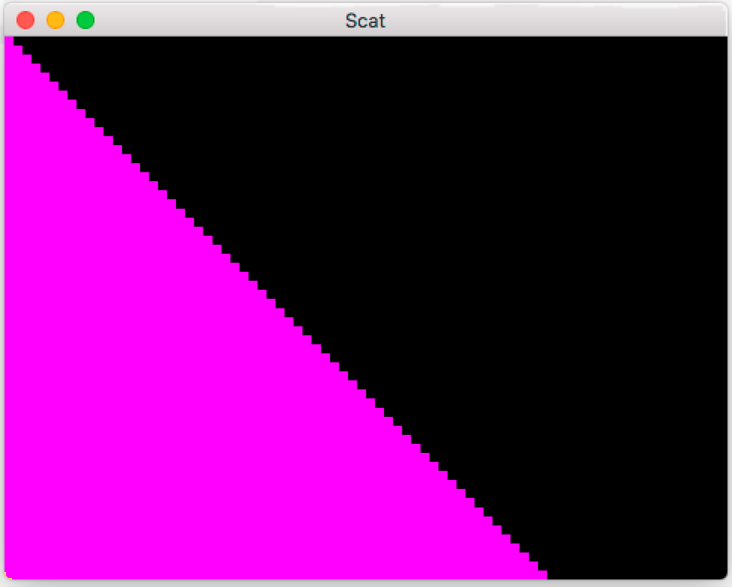
Question 5 Given two arrays A and B of the same (known) length, we define their *element-wise distance* as the array C such that for all n, C[n]=$\left|\text{A}[n] - \text{B}[n]\right|$. In other words, each element of C is defined as the absolute value of the difference between corresponding elements of A and B.

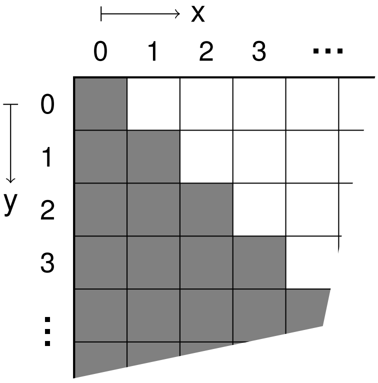
The program below allocates two arrays A and B of length 10. Complete the code so that it computes their element-wise distance in array C.

```
start:
    jmp main

A:      .word 13, 50,  2, 42, 27, 12,  1, 8, 37, 19
B:      .word  1,  5, 24, 42, 51, 21, 36, 2, 71,  7
C:      .word  0,  0,  0,  0,  0,  0,  0, 0,  0,  0

main:
```

```
start:
    jmp main

T1:     .word 13, 50,  2, 43, 27, 12,  1, 8, 37, 19
T2:     .word 1,   5, 24,  4, 72, 21, 36, 2, 71,  7
T3:     .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    ;; should be
    ;; 12, 45, 22, 39, 45, 9, 35, 6, 34, 12
    ;;  c, 2d, 16, 27, 2d, 9, 23, 6, 22, c
main:
    leti r1, T1                         ; pointer to T1
    leti r2, T2                         ; pointer to T2
    leti r3, T3                         ; pointer to T3

    leti r9, 0                          ; i
    leti r10, 10

loop:
    load r4, [r1]                       ; r4 contains T1[i]
    load r5, [r2]                       ; r5 contains T2[i]
    ble r4, r5, first                   ; if T1[i] <= T2[i] ... goto first ...
    sub r6, r4, r5                      ; otherwise compute T1[i]-T2[i]
    jmp store_result
first:
    sub r6, r5, r4                      ; ... and compute T2[i]-T1[i]

store_result:
    store [r3], r6

advance:
    addi r1, r1, 4
    addi r2, r2, 4
    addi r3, r3, 4
    addi r9, r9, 1
    blt r9, r10, loop

    halt
```

Question 6 Write a program that draws a pink triangle like illustrated in the pictures below. Your triangle should occupy all the screen's lines. Your entire program must not exceed 30 lines.





```
leti R11, 0xB0000000            ; base address of BRAM buffset
        leti R12, 0xFF00FF00    ; RGB for pink

        leti R6, 1              ; number of pixel per line
        leti R1, 0              ; Y offset

y_loop:
        leti R2, 0              ; X offset
        add R10, R11, R1        ; base address of our line of pixels
x_loop:
        add R9, R10, R2         ; address of pixel
        store [R9], R12

        addi R2, R2, 4          ; move on to the pixel on the right
        ble R2, R6, x_loop      ; if we’re not over drawing the current line’s pixels, c

        addi R1, R1, 320        ; move on to the next line ...
        leti R3, 19200
        addi R6, R6, 4          ; ... that will be one pixel more in width
        blt R1, R3, y_loop      ; stop whenever we’ve reach the window’s
                                ;  lower border

done:
        bra +0
```

Question 7 Translate the pseudo-code below to assembly language. Add comments in the code to explain how you implement variables A and B.

```
integer fibo(N: non-negative integer)
{
    if(N == 0) return 0;
    if(N == 1) return 1;

    A = fibo(N-1);
    B = fibo(N-2);

    return A+B;
}
```

```
    leti SP, 0x1000000
    leti R1, 7
    call fibo
    halt

fibo:
```

```
leti SP, 0x10000000

main:
        leti R1, 10

        call fibo
        bra +0

fibo:
        push LR
        push R2
        push R3

        ; when N <= 1 we have F(N)=N
        leti R2, 1
        ble  R1, R2, fibo_epilogue

        mov R2, r1 ; save N

        dec r1  ; N-1
        call fibo
        mov R3, r1 ; A = F(N-1)

        subi r1, r2, 2 ; N-2
        call fibo ; returns with B in R1

        add R1, R1, R3

fibo_epilogue:
        pop R3
        pop R2
        pop LR
        ret
```