IST-ASM Final Exam — Fall 2024

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 addition 100 + 50 in binary notation: convert both numbers to binary, then compute the sum entirely in binary. Show the details of your work.

decimal 100 as binary:	decimal 50 as binary:	addition:

Question 2 Write number -37 in two's complement on eight bits. Show the details of your work.

Question 3 Disassemble word `3120fff8` into ASM syntax.

Question 4 Write a program which uses a loop to compute the sum of all positive integers up to *N*. For instance with *N* = 5 you should find 1 + 2 + 3 + 4 + 5 = 15. Initially *N* is stored in R1, and at the end the result should be in R2.

```
leti R1, 5
```

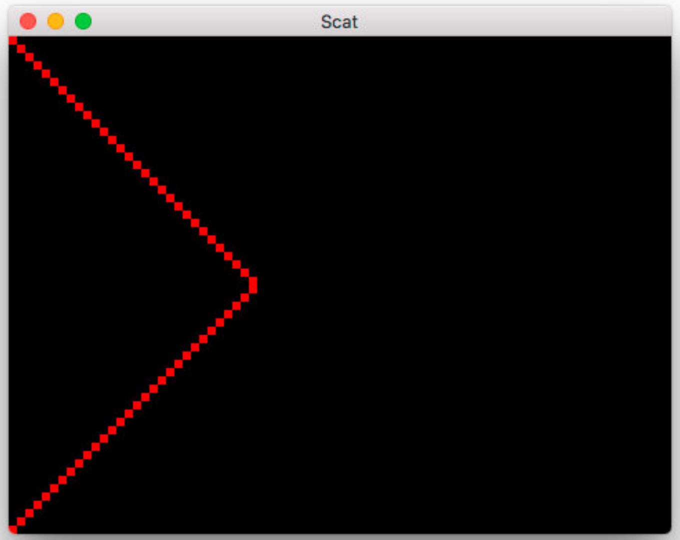
Question 5 Write a program which loops over an array of integers and computes their average value, rounded down to the nearest integer (e.g. the “average” of 9, 15, 10 and 17 is 12). The length of the array is also given in memory, as illustrated below. At the end the result should be in R1.

```
bra main

A:    .word 9, 15, 10, 17
len:  .word 4

main:
```

Question 6 In 30 lines or less, write a program that draws two diagonal red lines like illustrated on the right. The lines should start from the top left and bottom left corners of the screen, respectively, and they should end at mid-height i.e. where they cross each other.



Question 7 Exponentiation is defined as repeated multiplication. In other words, expression a^n denotes "*a* multiplied by itself *n* times" i.e. $a \times a \times ... \times a$. Obviously, we could compute all these multiplications iteratively. But to optimize execution time, we can leverage the fact $a^k \times a^k = a^{2k}$, which leads to this recursive formulation (with base case $a^0 = 1$):

$$a^n = \begin{cases} (a^2)^{n/2}, & \text{if } n \text{ is even} \\ a\,(a^2)^{(n-1)/2}, & \text{if } n \text{ is odd} \end{cases}$$

In other words, exponentiation can be implemented by the following algorithm:

```
integer fast_exp(a: non-negative integer,
                 n: non-negative integer)
{
    if ( n == 0 )    return 1
    if ( n is even ) return fast_exp(a*a, n/2)
    if ( n is odd )  return a * fast_exp(a*a, (n-1)/2)
}
```

Your task is to translate this pseudo-code to assembly language. Please add comments to help us understand your answer. You can test your implementation with obvious examples like $10^7 = 10000000$ or $2^{20} = 1048576$.

```
    leti SP, 0x1000000
    leti R1, 10 ; base A
    leti R2, 7  ; exponent N
    call fast_exp
    ; expected result: R1 == A**N
    halt

fast_exp:
```