

IST-ASM Final Exam — Fall 2025

Name:

- First, write your name in the box above. Then, have a quick read through all 8 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 $0x9E + 0x52$ where both numbers are positive integers. In the boxes below, convert both numbers to binary, compute the sum in binary, then convert the final result back to hexadecimal. Show the details of your work.

0x9E as binary:

0x52 as binary:

addition:

```
1001 1110
+ 0101 0010
  11 11
-----
1111 0000

= 0xF0
```

Question 2 Convert binary values below from two's complement on eight bits to decimal notation. Show the details of your work.

11011110

01101101

```

11011110
~ 00100001
+1 00100010
    32 + 2
-->    -34

```

```

01101101
64 + 32 + 8 + 4 + 1
--> 109

```

Question 3 What is the ASCII encoding of 2025-10-23 (ten characters) ? Write each byte as a hexadecimal number (i.e. “42” will be read as 0x42, not “decimal 42”).

```

32 30 32 35 2D 31 30 2D 32 33

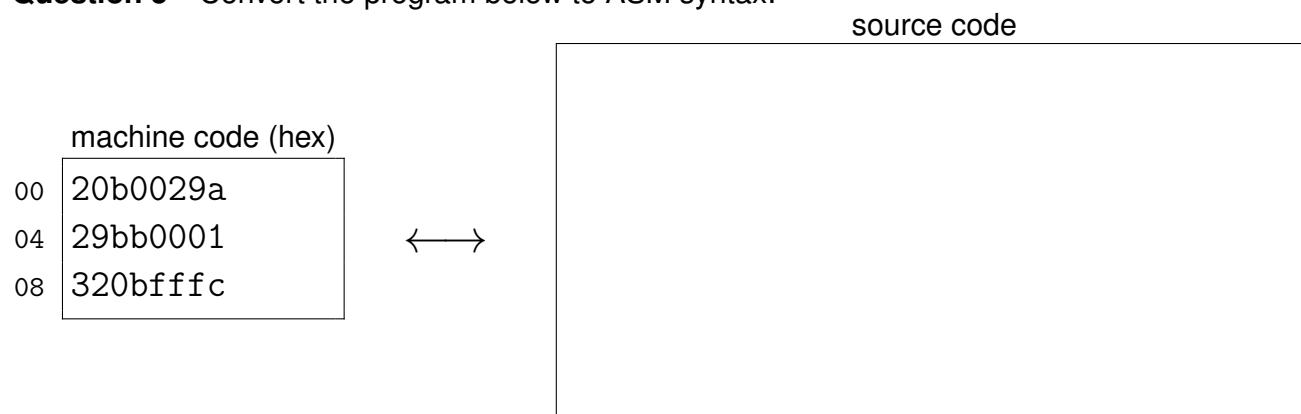
```

Question 4 For each color below, give the corresponding hex triplet.

yellow	
purple	
orange	
cyan	

yellow FEAR00
 purple FF00FF
 orange FF8000
 cyan 00FFFF

Question 5 Convert the program below to ASM syntax:



```

      leti R11, 666      or      addi  r11, zero, 666
loop: lsri R11, R11, 1
      bgtz R11, loop     or      blt   zero, r11, -4

```

Question 6 The program below implements a certain mathematical function f : given two integers A and B (both assumed to be strictly positive) it calculates $C = f(A, B)$. Give a simple expression for f .

$f(A, B) =$

A^B

```

A:      .word ...
B:      .word ...
C:      .word ...
main:   load R1, [A]
        load R2, [B]
        leti R3, 1
loop:   mul R3, R1, R3
        dec R2
        bgtz R2, loop
        store [C], R3
        halt

```

Question 7 Write a program which takes a positive number X in R1 as input and computes result Y in R2 such that the decimal notation of Y is the same as X but with all even digits (0, 2, 4, 6, 8) removed. If all the digits of X are even, then Y should be zero. The table below gives a few examples ; don't hesitate to ask for clarifications if needed.

X	37	69621	42	2025	123456789
Y	37	91	0	5	13579

```
;; leti R1, 1234567
;; leti R1, 123456789
leti R1, 2025
leti r2, 0

leti r8, 1 ; power we're looking at
```

```
main:
    modi R3, R1, 2
    bnez R3, odd

    bra bottom

odd:
    modi R7, R1, 10

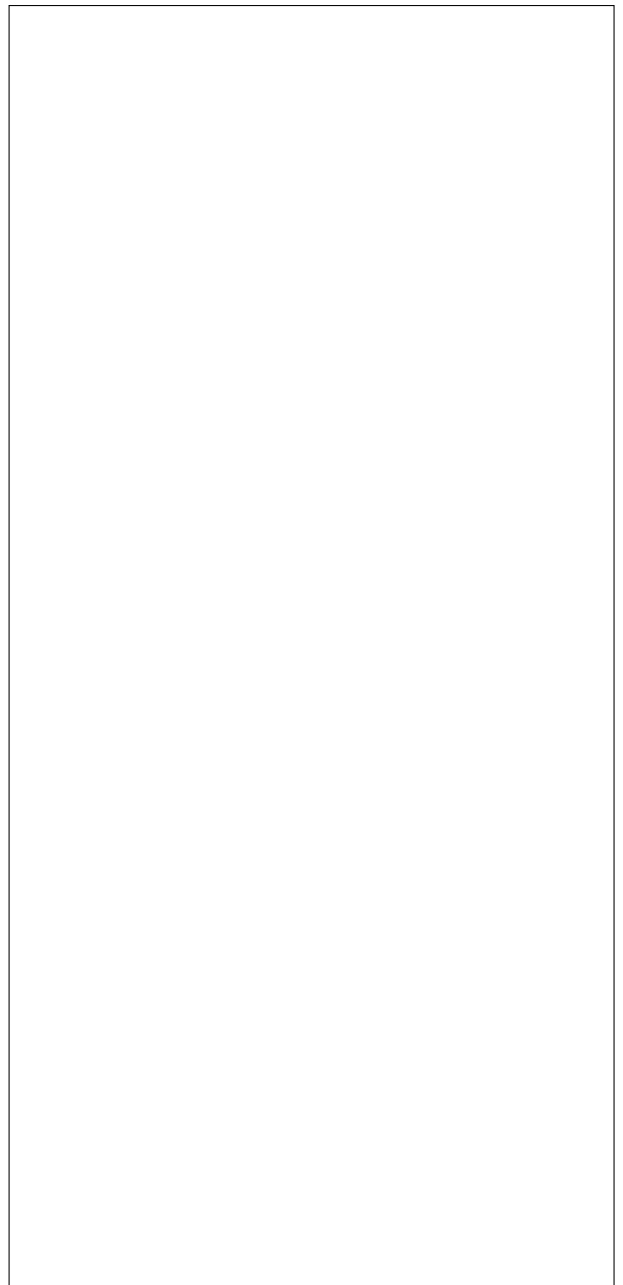
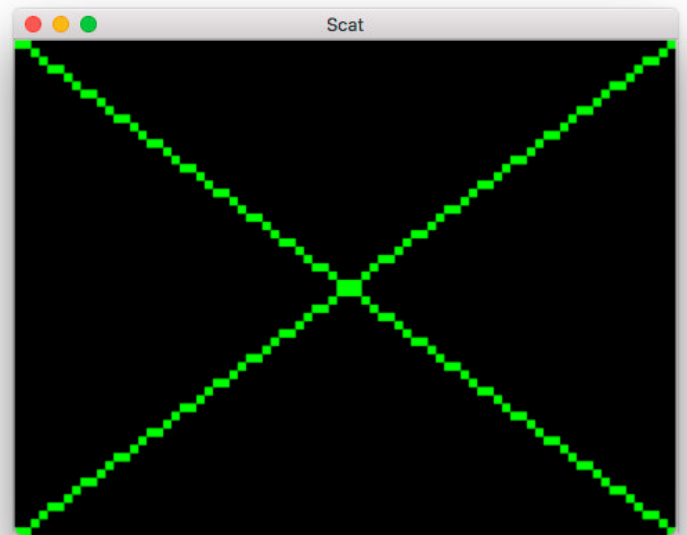
    mul R7, R7, r8
    add R2, R2, R7
    muli R8, R8, 10

bottom:
    divi R1, R1, 10

    bnez R1, main

    halt
```

Question 8 Write a program that draws two “diagonal” straight lines across the screen in a pure green color. Like illustrated on the right, your lines should connect pixels from opposite corners. However, as long as your lines are sufficiently “straight” (to within one pixel), you are not required to match the exact same pixels as in our picture.



This one is a bit hard because it combines several difficulties: MMIO, function calls, and working with non-integer coordinates (in our solution, coords are scaled 1000-fold)

```

    leti sp, 0x10000000
    leti r3, 0x00FF0000
    leti R5, 80000
    leti R6, 60000
loop:
    divi R1, R5, 1000
    divi R2, R6, 1000
    call drawpixel

    leti R8, 59
    sub R2, R8, R2
    call drawpixel

    subi R5, R5, 1000
    subi R6, R6, 750
    bgez R5, loop
    halt
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; input parameters
;; R1 == X
;; R2 == Y
;; R3 == RGB color
drawpixel:
    push R1
    push R2
    push R3
    push R4
    ;; push LR ; optional since we are a leaf

    bltz R1, drawpixel_epilogue
    bltz R2, drawpixel_epilogue
    leti R4, 80
    bge R1, R4, drawpixel_epilogue
    leti R4, 60
    bge R2, R4, drawpixel_epilogue

    muli R1, R1, 4
    muli R2, R2, 320
    add R1, R1, R2 ;; pixel offset in VRAM

    leti R2, 0xB0000000
    add R1, R1, R2 ;; pixel absolute addr

    store [R1], R3
drawpixel_epilogue:
    ;; pop LR

```

```
pop R4  
pop R3  
pop R2  
pop R1  
ret
```