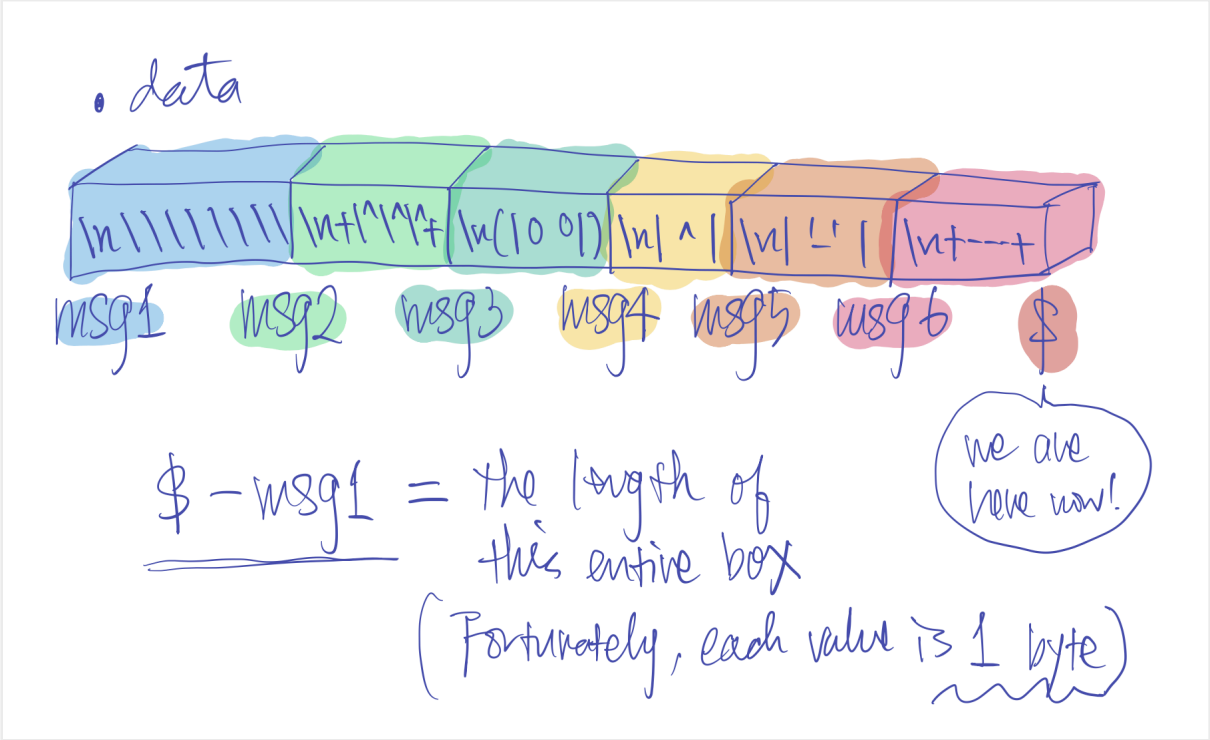
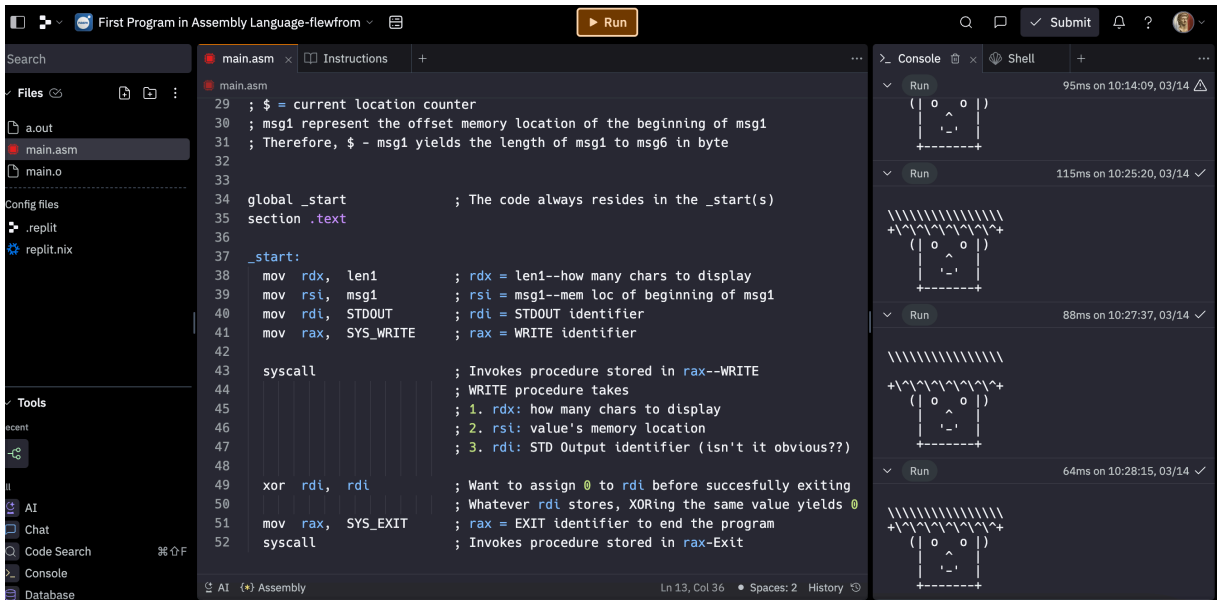
Extra questions:
A: 0xa represents a decimal value of 10 in hex. In ASCII value, 10 means a newline. Therefore, if you add 0xa at the end of msg6, there will be another new line (empty line) displayed.

B: rdi will contain 0 no matter what value rdi holds. XORing the same value guarantees to yield 0 because each bit is the same. Only when each bit's value is different (i.e. 0 ^ 1 = 1), it returns 1.

C: $ represents the memory location where we are currently at. In Assembly language, variable's name represents the start of the memory location where the variable is stored. From msg1 to msg6, each char value is stored consecutively. Therefore, $ - msg1 = the length of the whole strings from msg1 to msg6.
SYS_WRITE takes an argument stored in rdx, which indicates how many characters to display. So, we want to store $ - msg1 in rdx.



Sample run:


First Program in Assembly Language-flewfrom
Run
Submit
Search
Files
a.out
main.asm
main.o
Config files
.replit
replit.nix
Tools
AI
Chat
Code Search
Console
Database
main.asm
Instructions
main.asm
; $ = current location counter
; msg1 represent the offset memory location of the beginning of msg1
; Therefore, $ - msg1 yields the length of msg1 to msg6 in byte
global _start ; The code always resides in the _start(s)
section .text
_start:
mov rdx, len1 ; rdx = len1--how many chars to display
mov rsi, msg1 ; rsi = msg1--mem loc of beginning of msg1
mov rdi, STDOUT ; rdi = STDOUT identifier
mov rax, SYS_WRITE ; rax = WRITE identifier
syscall ; Invokes procedure stored in rax--WRITE
; WRITE procedure takes
; 1. rdx: how many chars to display
; 2. rsi: value's memory location
; 3. rdi: STD Output identifier (isn't it obvious??)
xor rdi, rdi ; Want to assign 0 to rdi before succesfully exiting
; Whatever rdi stores, XORing the same value yields 0
mov rax, SYS_EXIT ; rax = EXIT identifier to end the program
syscall ; Invokes procedure stored in rax-Exit
AI
Assembly
Ln 13, Col 36
Spaces: 2
History
Console
Shell
Run
95ms on 10:14:09, 03/14
Run
115ms on 10:25:20, 03/14
Run
88ms on 10:27:37, 03/14
Run
64ms on 10:28:15, 03/14