7. ??? ????

Literally me

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"

.global main
main:
  RCALL SER_init

  LDI   ZH, hi8(opening_msg)
  LDI   ZL, lo8(opening_msg)
  RCALL SER_print

  ; 16 Bit Addition
  LDI   ZH, hi8(addition_msg)
  LDI   ZL, lo8(addition_msg)
  RCALL SER_print

  ; DEC 26983 = HEX 0x6967
  LDI R16, 0x69   ; upper byte
  LDI R17, 0x67   ; lower byte

  ; DEC 4882 = HEX 0x1312
  LDI R18, 0x13   ; upper byte
  LDI R19, 0x12   ; lower byte

  ADD R17, R19    ; add lower byte
  ADC R16, R18    ; add upper byte + carry from lower

  ; R16:R17 = 31865
  RCALL SER_hex
  MOV R16, R17  ; print the lower byte this time
  RCALL SER_hex

  ; 16 Bit Substraction
  LDI   ZH, hi8(substraction_msg)
```

```
LDI   ZL, lo8(substraction_msg)
RCALL SER_print

; DEC 26983 = HEX 0x6967
LDI R16, 0x69   ; upper byte
LDI R17, 0x67   ; lower byte

; DEC 4882 = HEX 0x1312
LDI R18, 0x13   ; upper byte
LDI R19, 0x12   ; lower byte

SUB R17, R19    ; substract lower byte
SBC R16, R18    ; substract upper byte - borrow (C) from lower

; R16:R17 = 22101
RCALL SER_hex
MOV R16, R17  ; print the lower byte this time
RCALL SER_hex

; 16 Bit Immediate Addition
LDI   ZH, hi8(iaddition_msg)
LDI   ZL, lo8(iaddition_msg)
RCALL SER_print

LDI R16, 0x13       ; upper byte
LDI R17, 0x12       ; lower byte

.EQU num, 0x5457    ; immediate variable directive
SUBI R17, lo8(-num) ; 0x12 - -0x57 = 0x12 + 0x57
SBCI R16, hi8(-num) ; 0x13 - -0x54 - C = 0x13 + 0x54 + C

RCALL SER_hex
MOV R16, R17  ; print the lower byte this time
RCALL SER_hex

; 16 Bit Immediate Substraction
LDI   ZH, hi8(isubstraction_msg)
LDI   ZL, lo8(isubstraction_msg)
RCALL SER_print
```

```
  LDI R16, 0x67       ; yeah you get the idea atp
  LDI R17, 0x69

  .EQU num, 0x1312
  SUBI R17, lo8(num)  ; this time we simply just substract it
  SBCI R16, hi8(num)  ; no need to be negative :)

  RCALL SER_hex
  MOV R16, R17  ; print the lower byte this time
  RCALL SER_hex

  ; Multiplication
  LDI   ZH, hi8(multiplication_msg)
  LDI   ZL, lo8(multiplication_msg)
  RCALL SER_print

  LDI R16, 23
  LDI R17, 9
  MUL R16, R17

  MOV R16, R1
  RCALL SER_hex
  MOV R16, R0
  RCALL SER_hex

loop:
  RCALL loop

; Initializes USART
; Registers Affected: R24
SER_init:
  CLR   R24
  STS   UCSR0A, R24                    ; clear UCSR0A register
  STS   UBRR0H, R24                    ; clear UBRR0H register
  LDI   R24, 103                       ; store in UBRR0L 103 value
  STS   UBRR0L, R24                    ; to set baud rate 9600
  LDI   R24, 1 << RXEN0 | 1 << TXEN0    ; enable RXB & TXB
  STS   UCSR0B, R24
  LDI   R24, 1 << UCSZ00 | 1 << UCSZ01  ; asynch, no parity, 1 stop, 8 bits
  STS   UCSR0C, R24
```

```
  RET

; Prints character in R16 to USART
; Blocks while UDRE0 is not ready.
; Registers Affected: R24
SER_send_byte:
  LDS   R24, UCSR0A
  SBRS  R24, UDRE0                    ; test data buffer if data can be sent
  RJMP  SER_send_byte                 ; loop back if not ready
  STS   UDR0, R16                     ; sends R16 to USART
  RET

; Prints entire message of data pointed in Z until string end (0)
; To fill Z with string message:
;   LDI   ZH, hi8(message)
;   LDI   ZL, lo8(message)
; Registers Affected: R16, R24 (SER_send_byte)
SER_print:
  LPM   R16, Z+                       ; load char of string onto R18
  CPI   R16, 0                        ; check if R16 = 0 (end of string)
  BREQ  exit_SER_print                ; if yes, exit
  RCALL SER_send_byte                 ; send the character byte
  RJMP  SER_print                     ; loop back & get next character
exit_SER_print:
  RET

; Prints the lower nibble of R16
; Registers Affected: R24 (SER_send_byte)
SER_nibble:
  PUSH  R16           ; preserves R16
  ANDI  R16, 0x0F     ; mask that removes the higher nibble
  SUBI  R16, -'0'     ; add '0' to R16 to represent ASCII '0' - '9'.
  CPI   R16, '9' + 1  ; compare with '9' + 1 = ':'
  BRLT  print_nibble  ; if no problem just simply print the character
  SUBI  R16, -7       ; otherwise add with 7 first to adjust for 'A'
print_nibble:
  RCALL SER_send_byte
  POP   R16           ; retrieves R16
  RET
```

```
; Prints R16 as HEX to USART
; R16 is preserved.
SER_hex:
  SWAP R16            ; swap to get upper nibble first
  RCALL SER_nibble
  SWAP R16            ; revert the nibbles back for the lower one
  RCALL SER_nibble
  RET

opening_msg:
    .byte 10,13 ; new line, carriage return
    .ascii "== Literally Einstein and Tesla =="
    .byte 0

addition_msg:
    .byte 10,13 ; new line, carriage return
    .ascii "16 Bit Addition: "
    .byte 0

substraction_msg:
    .byte 10,13 ; new line, carriage return
    .ascii "16 Bit Addition: "
    .byte 0

iaddition_msg:
    .byte 10,13 ; new line, carriage return
    .ascii "16 Bit Immediate Addition: "
    .byte 0

isubstraction_msg:
    .byte 10,13 ; new line, carriage return
    .ascii "16 Bit Immediate Substraction: "
    .byte 0

multiplication_msg:
    .byte 10,13 ; new line, carriage return
    .ascii "Multiplication: "
    .byte 0
```

Output

Revision #1
Created 2026-02-25 09:17:17 UTC by MF
Updated 2026-02-25 09:20:25 UTC by MF