6.1)

a) IF P >= Q

cp P, Q

brsh ELSE

b) IF Q > P

cp P, Q

brlo ELSE

c) iF P = Q

cp P, Q

breq ELSE

6.2)

a) IF P >= Q

cp P, Q

brge ELSE

b) IF Q > P

cp P, Q

brlt ELSE

c) IF P = Q

cp P, Q

breq ELSE

6.11)

Using a space in memory is a waste since we know the constant before compile time.

6.12) FOR Loop

ldi r16, 0

LOOP:

cpi r16, 10

breq DONE

inc r16

rjmp LOOP

DONE:

rjmp DONE

6.15)

lds r16, K1

lds r17, K2

lds r18, K3

WHILE:

cp r16, r17

brlt DONE

cp r17, r18

brlt THEN

mov r17, r18

inc r16

rjmp WHILE

THEN:

mov r17, r16

rjmp WHILE

6.16)

iter K1 K2 K3

------------------

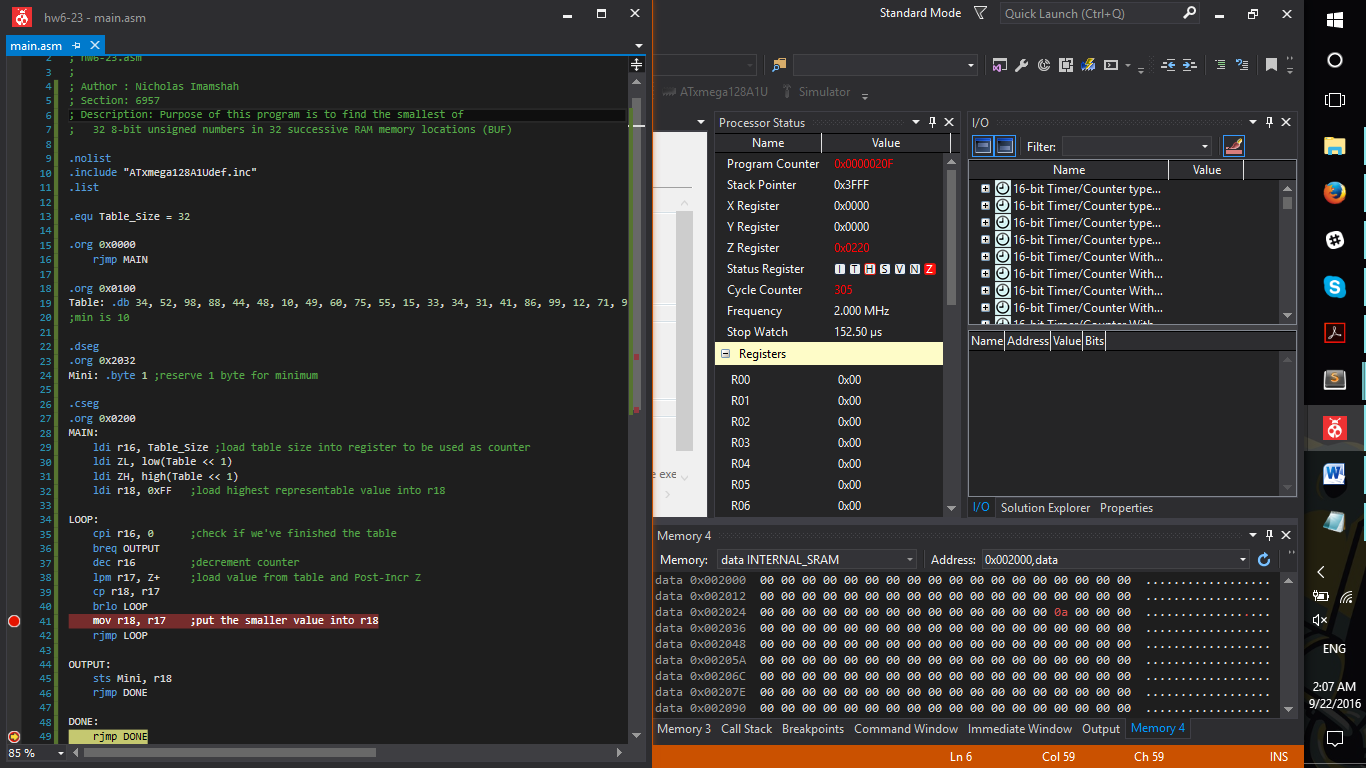
0 1 3 -2

1 2 -2 -2

------------------

1 pass 2 -2 -2

6.23)



;

; hw6-23.asm

;

; Author : Nicholas Imamshah

; Section: 6957

; Description: Purpose of this program is to find the smallest of

; 32 8-bit unsigned numbers in 32 successive RAM memory locations (BUF)

.nolist

.include "ATxmega128A1Udef.inc"

.list

.equ Table\_Size = 32

.org 0x0000

rjmp MAIN

.org 0x0100

Table: .db 34, 52, 98, 88, 44, 48, 10, 49, 60, 75, 55, 15, 33, 34, 31, 41, 86, 99, 12, 71, 96, 95, 25, 32, 37, 21, 70, 99, 85, 54, 19, 23

;min is 10

.dseg

.org 0x2032

Mini: .byte 1 ;reserve 1 byte for minimum

.cseg

.org 0x0200

MAIN:

ldi r16, Table\_Size ;load table size into register to be used as counter

ldi ZL, low(Table << 1)

ldi ZH, high(Table << 1)

ldi r18, 0xFF ;load highest representable value into r18

LOOP:

cpi r16, 0 ;check if we've finished the table

breq OUTPUT

dec r16 ;decrement counter

lpm r17, Z+ ;load value from table and Post-Incr Z

cp r18, r17

brlo LOOP

mov r18, r17 ;put the smaller value into r18

rjmp LOOP

OUTPUT:

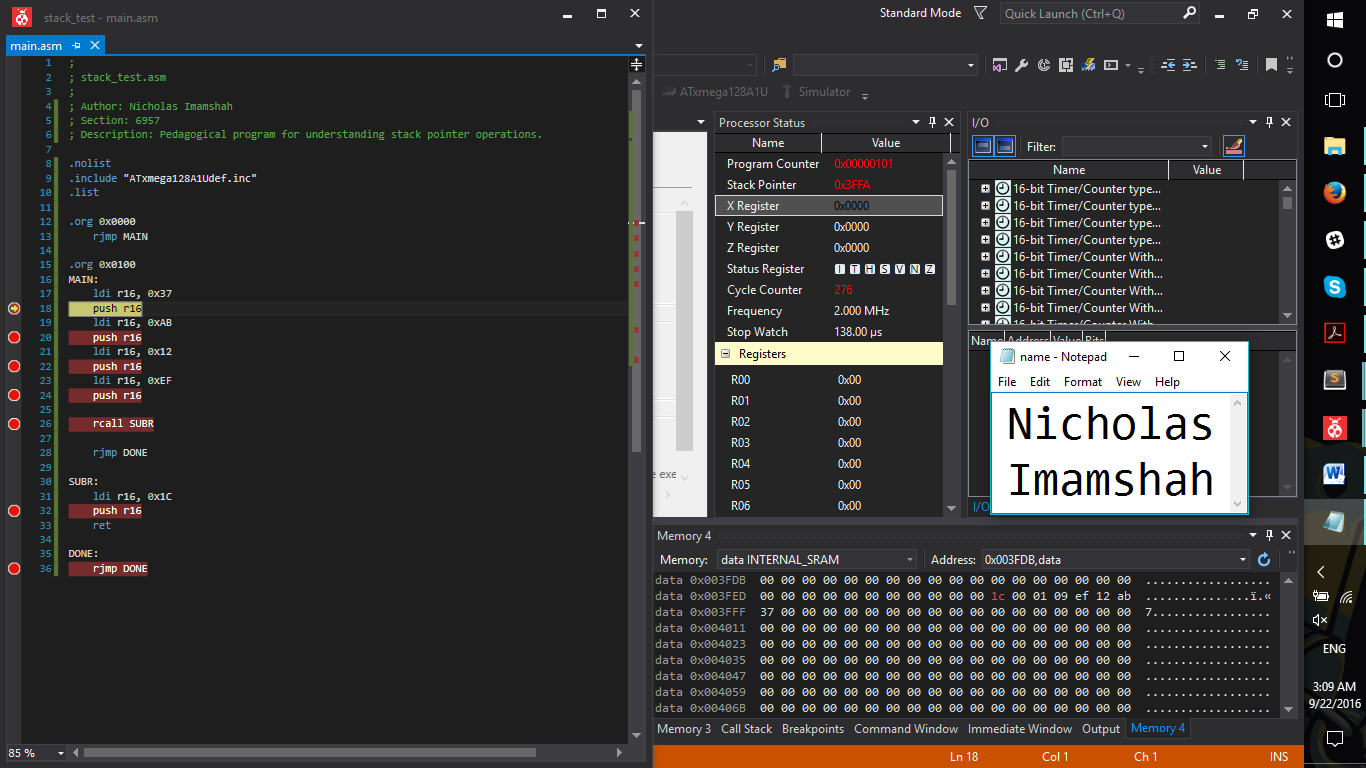
sts Mini, r18

rjmp DONE

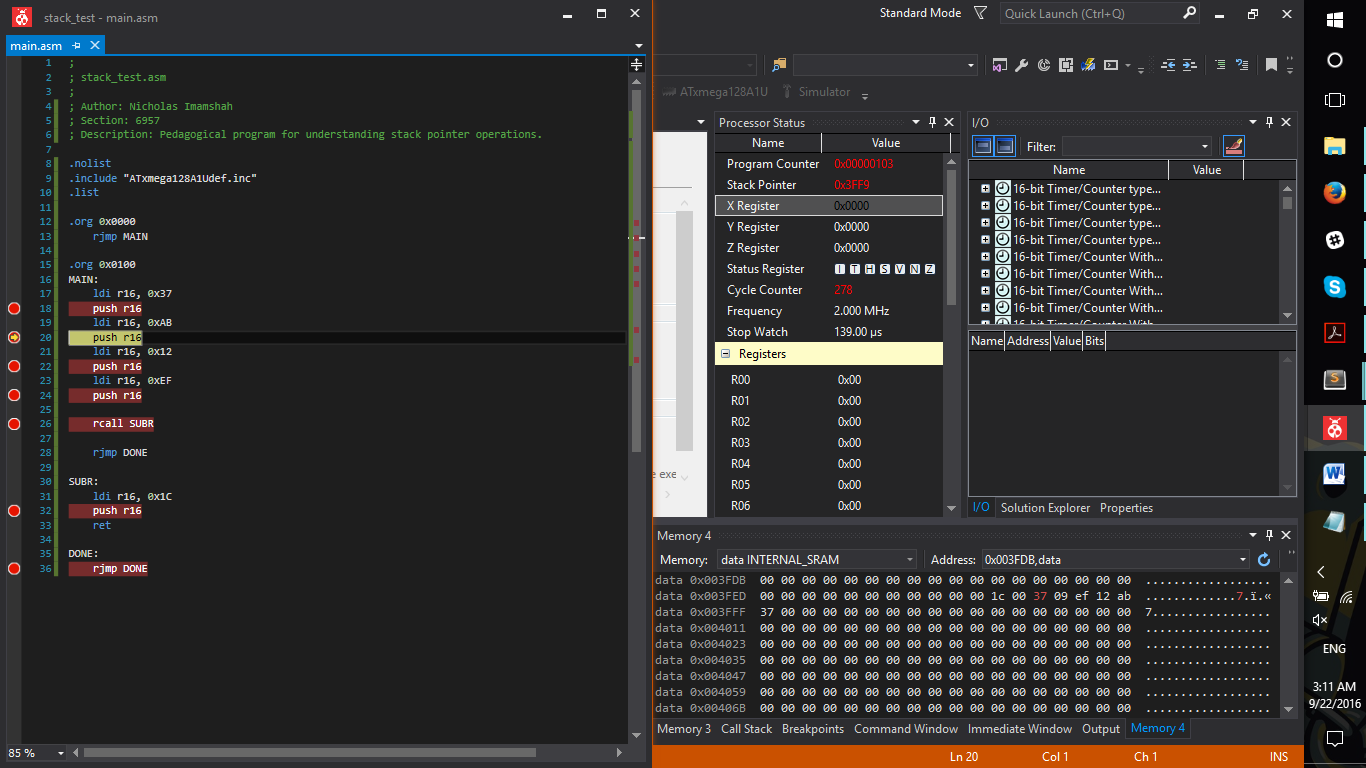
DONE:

rjmp DONE

Stack Question:



Initial return from subroutine



First push to stack after return

;

; stack\_test.asm

;

; Author: Nicholas Imamshah

; Section: 6957

; Description: Pedagogical program for understanding stack pointer operations.

.nolist

.include "ATxmega128A1Udef.inc"

.list

.org 0x0000

rjmp MAIN

.org 0x0100

MAIN:

ldi r16, 0x37

push r16

ldi r16, 0xAB

push r16

ldi r16, 0x12

push r16

ldi r16, 0xEF

push r16

rcall SUBR

rjmp DONE

SUBR:

ldi r16, 0x1C

push r16

ret

DONE:

rjmp DONE

|  |  |
| --- | --- |
| **Address** | **Data** |
| 0x3FF8 | 0x1C |
| 0x3FF9 | 0x00 |
| 0x3FFA | 0x01 |
| 0x3FFB | 0x09 |
| 0x3FFC | 0xEF |
| 0x3FFD | 0x12 |
| 0x3FFE | 0xAB |
| 0x3FFF | 0x37 |

Before return

Instead of using 0x0109 as return address, it uses 0x0001, but the code segment is .org’d to 0x0100; so it instead returns to 0x0101.