6.23:

Code:

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

.equ tabelLen = 32

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x400

Table: .db 9,1,7,2,4,3,8,6,5,-1,21,53,45,-12,2,32,42,53,51,56,74,23,1,63,56,456,34,89,60,467,40,12 ; table for use in testing BSORT subroutine

.dseg

.org 0x2000

output: .byte 1

.cseg

.org 0x200

init:

ldi ZL, low(Table << 1)

ldi ZH, high(Table << 1)

ldi YL, low(output)

ldi YH, high(output)

ldi r18, 0x00

loopinit:

cpi r18, tabelLen

breq main

lpm r16, Z+

st Y+, r16

inc r18

jmp loopinit

main:

rcall smallest

jmp done

done:

jmp done

.org 0x300

smallest:

push r16

push r17

push r18

ldi YL, low(output)

ldi YH, high(output)

ldi r18, 0x00

ld r16, Y+

loop:

cpi r18, tabelLen

breq end

ld r17, Y+

cp r17, r16

brge endloop

mov r16, r17

endloop:

inc r18

jmp loop

end:

st -Y, r16

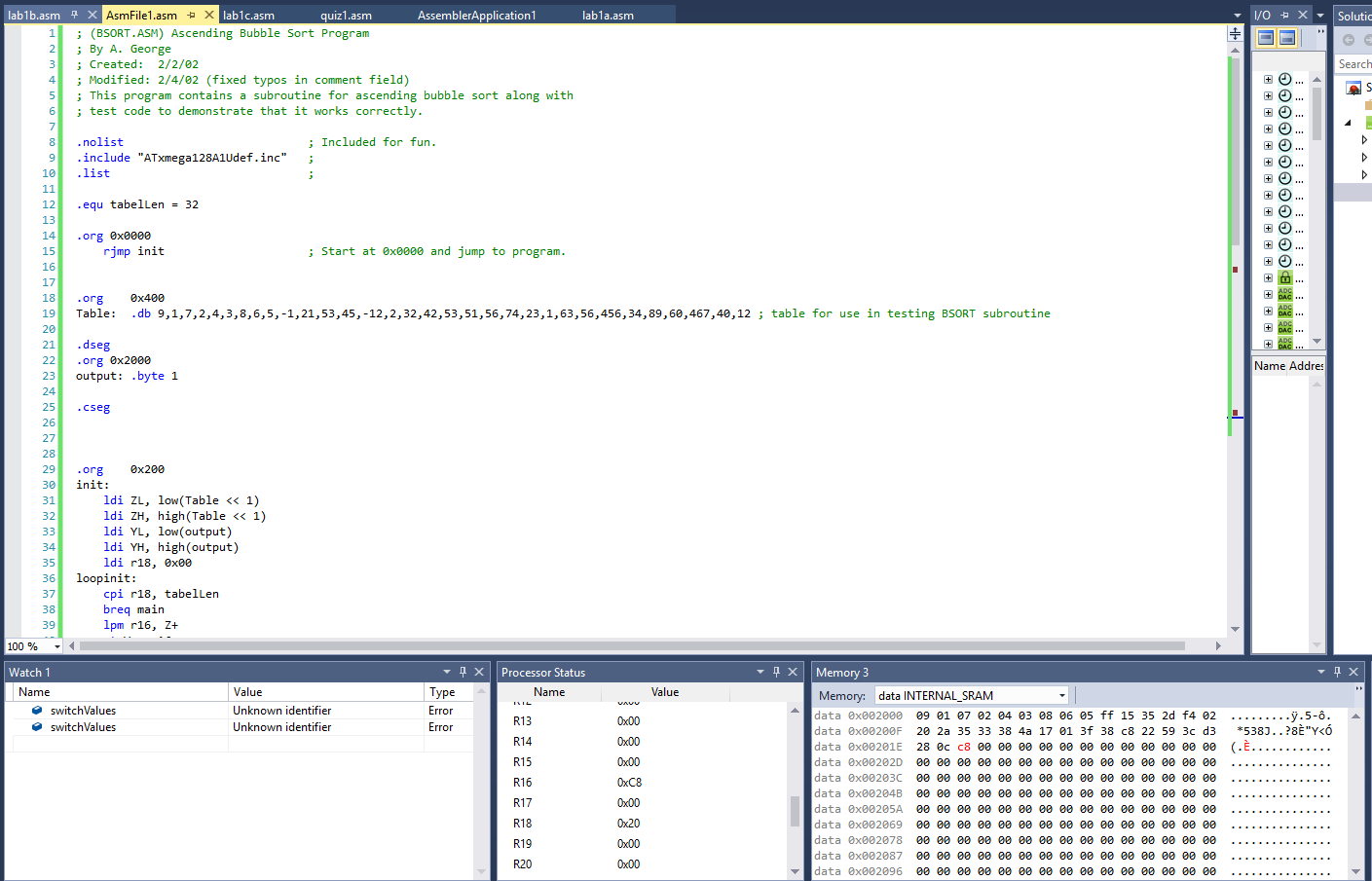
pop r18

pop r17

pop r16

ret

Screenshot:



2:

The stack initially grows by 4 bytes putting at address 0x3FFD. Then the rcall pushes the return address onto the stack, increasing its size by another 3 bytes putting it ad 0x3FF8. Then 0x1c is pushed, moving the stack to 0x3FF7. Finally the program returns from the subroutine placing the stack pointer to 0x3FFA.

Code:

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

ldi r16, 0x37

push r16

ldi r16, 0xAB

push r16

ldi r16, 0xEF

push r16

ldi r16, 0x12

push r16

rcall subr

jmp done

done:

jmp done

subr:

ldi r16, 0x1c

push r16

ret

Screenshot: