Mark Schuster | EEL 3744 C HW3 | 09/20/2017

Schuster, Mark
3744
Hw3

6.1 a. CP QP
BRLO

b. CP PQ
BRLO

c. CP PQ
BREQ

6.2 a. CP PQ
BRGE

b. CP QP
BRLT

c. CP PQ
BREQ

6.11 The first as it is handled in compilation and doesn't take up ROM space while the device is running

---

6.12
```
 ldi r16, 0x00
LOOP:
  CPi r16, 10
  BREQ END
  ---
  inc r16
  jmp LOOP
END
```

6.15

```
      K1: db ...
      K2: db ...
      K3: db ...
       :
       :
      ldi ZL, low(K1)
      ldi ZH, high(K1)
      lpm r16, Z+
      lpm r17, Z+
      lpm rK, Z
loop:
      cp r18, r17
      BRLT END
      cp r17, r18
      BRLT ELSE
      mov r17, r18
      jmp ENDER
ELSE:
      mov r17, r18
ENDIF:
    inc R16
    jmp loop
END:
```

6.16 1, 3, -2

$\downarrow$

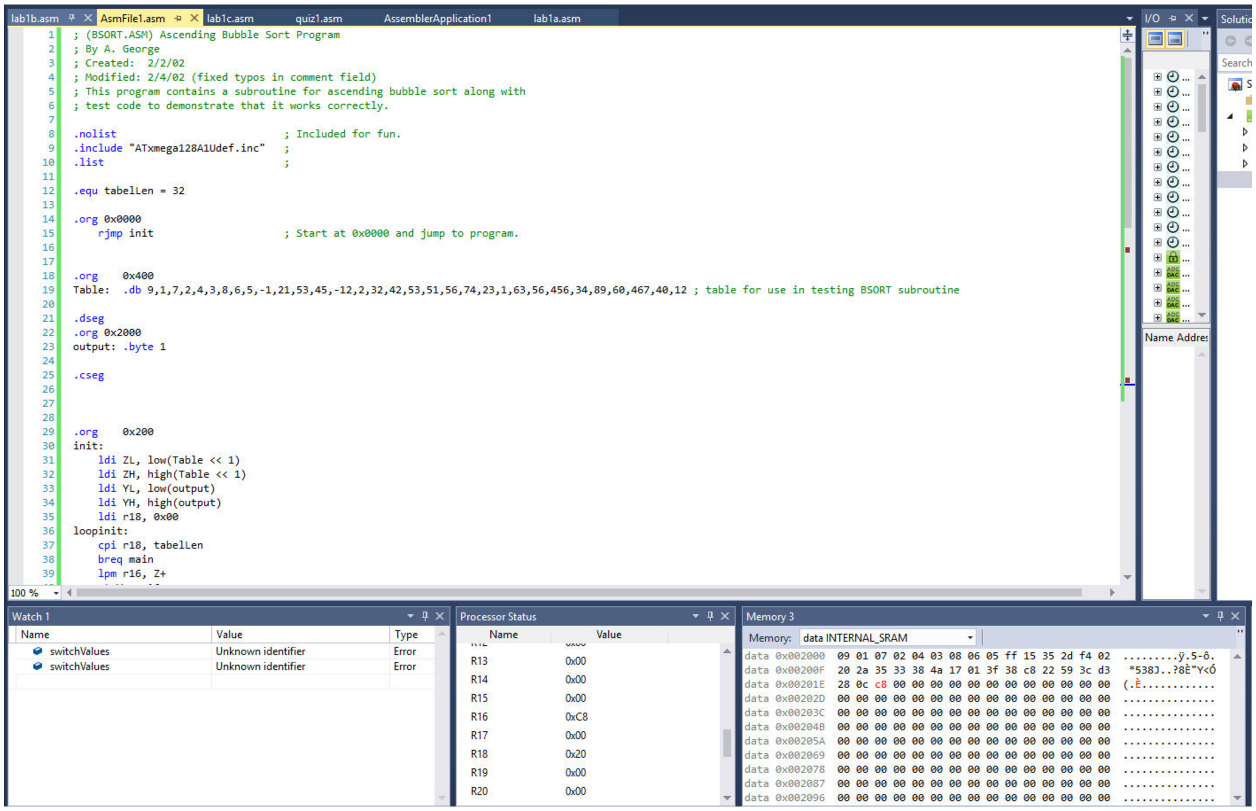
k1 = 2, k2 = -2, k3 = -2

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:



```
; (BSORT.ASM) Ascending Bubble Sort Program
; By A. George
; Created:  2/2/02
; Modified: 2/4/02 (fixed typos in comment field)
; This program contains a subroutine for ascending bubble sort along with
; test code to demonstrate that it works correctly.

.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+
```

Watch 1

Name	Value	Type
switchValues	Unknown identifier	Error
switchValues	Unknown identifier	Error

Processor Status

Name	Value
R13	0x00
R14	0x00
R15	0x00
R16	0xC8
R17	0x00
R18	0x20
R19	0x00
R20	0x00

Memory 3

Memory: data INTERNAL_SRAM

```
data 0x002000  09 01 07 02 04 03 08 06 05 ff 15 35 2d f4 02  .........ÿ.5-ô.
data 0x00200F  20 2a 35 33 38 4a 17 01 3f 38 c8 22 59 3c d3   *538J..?8È"Y<Ó
data 0x00201E  28 0c c8 00 00 00 00 00 00 00 00 00 00 00 00  (.È............
data 0x00202D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x00203C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x00204B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x00205A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x002069  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x002078  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x002087  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x002096  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
```

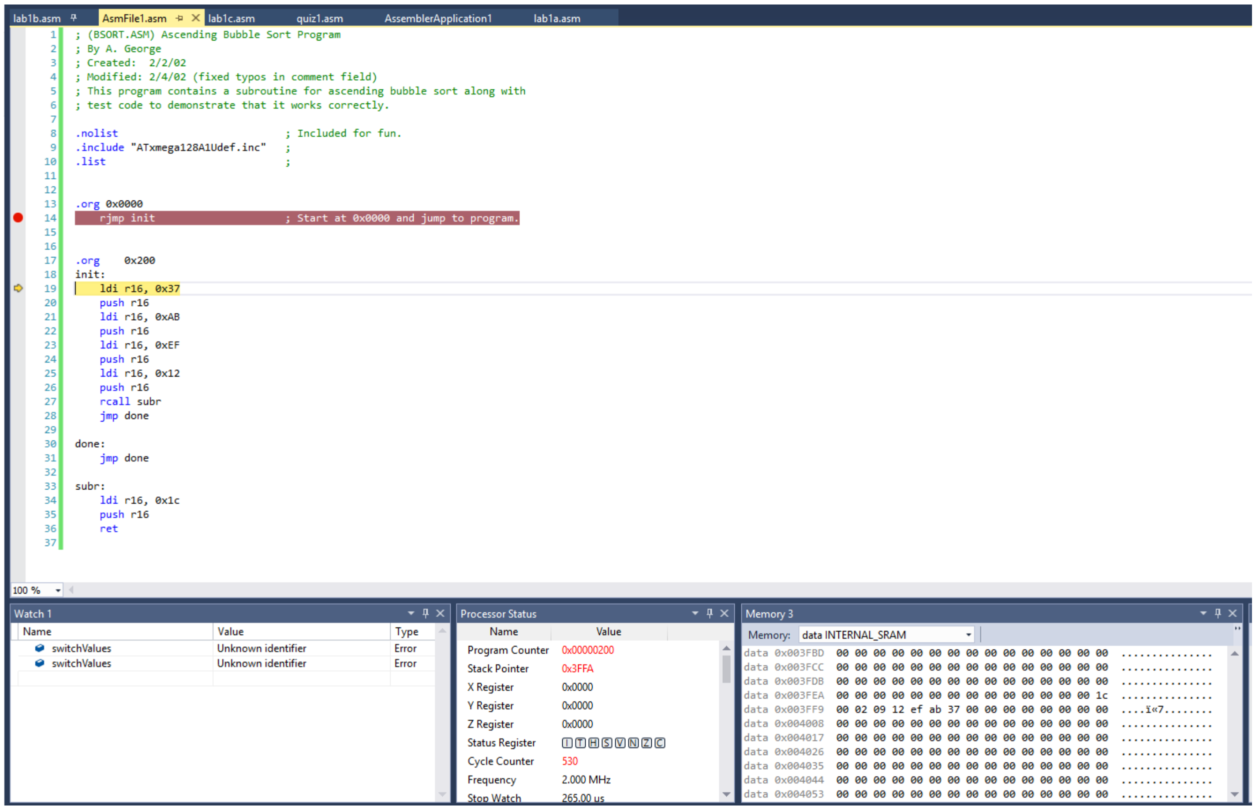
2:

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:



lab1b.asm | AsmFile1.asm | lab1c.asm | quiz1.asm | AssemblerApplication1 | lab1a.asm

```
; (BSORT.ASM) Ascending Bubble Sort Program
; By A. George
; Created:  2/2/02
; Modified: 2/4/02 (fixed typos in comment field)
; This program contains a subroutine for ascending bubble sort along with
; test code to demonstrate that it works correctly.

.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
```

100 %

Watch 1

Name	Value	Type
switchValues	Unknown identifier	Error
switchValues	Unknown identifier	Error

Processor Status

Name	Value
Program Counter	0x00000200
Stack Pointer	0x3FFA
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	530
Frequency	2.000 MHz
Stop Watch	265.00 us

Memory 3

Memory: data INTERNAL_SRAM

```
data 0x003FBD  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x003FCC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x003FDB  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x003FEA  00 00 00 00 00 00 00 00 00 00 00 00 00 00 1c  ...............
data 0x003FF9  00 02 09 12 ef ab 37 00 00 00 00 00 00 00 00  ....ï«7........
data 0x004008  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x004017  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x004026  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x004035  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x004044  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
data 0x004053  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
```