Zhu, Pengzhao
EEL 3744
HW 2

$\boxed{6.1}$ Unsigned number for 6.1

part a) ; If P >= Q

```
ldi  YL, low (P)
ldi  YH, high (P)
ldi  XL, low (Q)
ldi  XH, high (Q)
ld   r16, Y
ld   r17, X
cp   r16, r17
brsh ELSE

ELSE:
        ; do something
```

part b) ; if Q > P

```
ldi  YL, low(P)
ldi  YH, high(P)
ldi  XL, low(Q)
ldi  XH, high(Q)
ld   r16, Y
ld   r17, X
cp   r17, r16
brsh SECOND
rjump HI ; do something

SECOND: cp r17, r16
        brne ELSE

ELSE: do something
```

part c)

```
ldi  YL, low (P)
ldi  YH, high (P)
ldi  XL, low (Q)
ldi  XH, high (Q)
ld   r16, Y
ld   r17, X
cp   r17, r16
BREQ Else

ELSE:
        do something
```

Signed number for 6.2

$\boxed{6.2}$

part a) ; if p >= Q

```
ldi  YL, low(p)
ldi  YH, high(p)
ldi  XL, low (Q)
ldi  XH, high (Q)
ld  r16 , y
ld  r17, x
 cp  r16, r17
brge    ELSE

ELSE:
        ; do something
```

part b ; if Q>P

```
ldi  YL, low (p)
ldi  YH, high (p)
ldi  XL , low (Q)
ldi  XH,  high( Q)
ld  r16 , Y
ld  r17, X
cp  r17, r17
brge   SECOND

SECOND:   cp r17, r16
brne  ELSE

ELSE ;    do something
```

part ( )

```
ldi  YL, low (p)
ldi  YH, high (p)
ldi  XL , low (Q)
ldi  XH, high (Q)
ld r16,y
ld r17, x
cp  r17, r16
BREQ   ELSE

ELSE:
    ; do something
```

$\boxed{6.11}$

The use of a constant defined by an equate is the better programming technique when compared to using a constant stored in ROM to initialize register. When you use a equate to define a constant, It is easier to get access to it and easier to remember. Also, Dr. Schwartz told us to use the equ. directive in class. He said it is a good habit.

$\boxed{6.12}$

This assembly code will run in a similar way as a for loop in C. verified with Atmel Studio

```
.org 0x00
jmp MAIN

.org 0x200

MAIN:
ldi r16,5          ; our loop counter

Loop:
 ldi r17,4
 cpi r16,0
 breq DONE         ; stop when r16 =0
 dec r16
 rjmp Loop

DONE:
 rjmp DONE
```

Scanned by CamScanner

6.15

```
MAIN:          ; let us treat k1, k2, and k3 as registers

    cp  k1 , k2
    brlt  DONE          ; stop loop when k1 < k2
    cp  k3,  k2         ; check if greater or equal
    brge  NEXT
    mov  k2 , k3        ; k2 = k3
    inc  k1
     rjump  MAIN;
 NEXT :
      cp k3, k2         ; check if equal, branch if not
      brne  NEXT2          equal
     mov k2, k3         ; k2 = k3
      inc k1
       rjump MAIN;


  NEXT2;
        mov k2, k1      ; k2 = k1
        rjump MAIN;

  DONE;                  ; infinite loop
          rjmp DONE
```

Scanned by CamScanner

$\boxed{6.16}$

k1 = 1, k2 = 3, k3 = -2 initially

Double check ← 1st loop

1st loop	
k2 = -2	k2 = -2
k3 = -2	k3 = -2
k1 = 2	k1 = 2

The loop only loop 1 time. The final. The final values are k1 = 2, k2 = -2, and k3 = -2

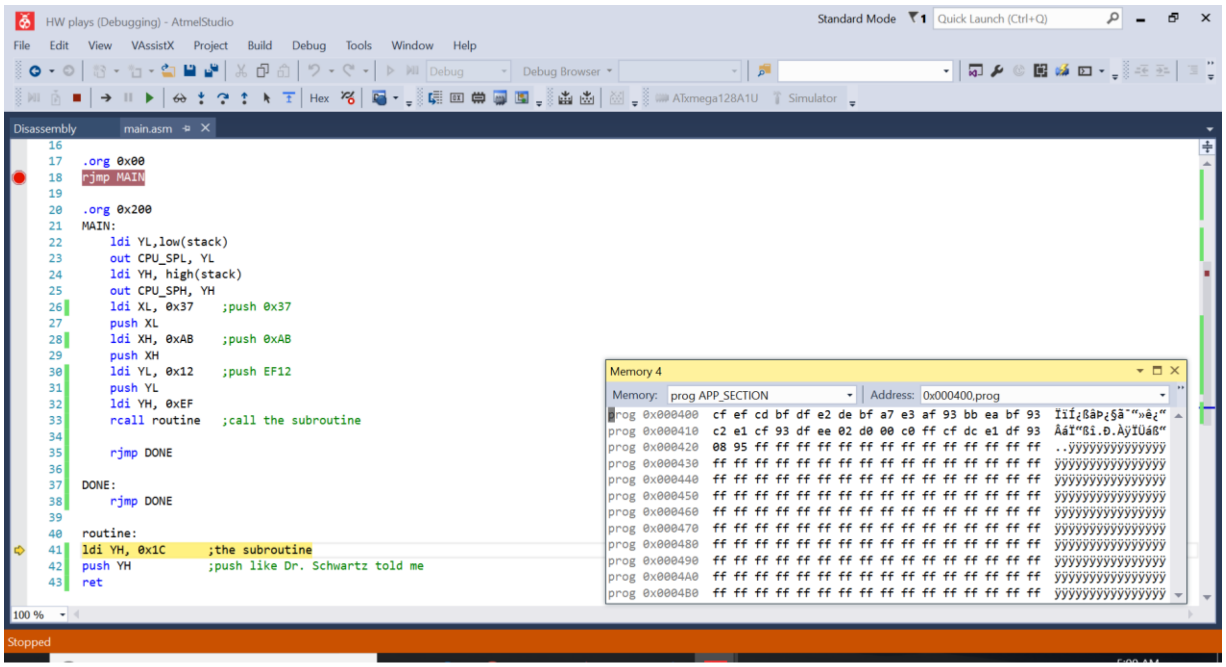
$\boxed{6.23}$

The code is on the pdf submission

```
/* HW2- Question 6.23
Name: Pengzhao Zhu
Section#: 112D
TA Name: Chris Crary
Description: This program find the largest of thirty-two 8-bit unsigned numbers in 32
successive RAM memory locations.
             It then places the answer in the 33rd location (Result).

I didn't program any data onto the SRAM (can't find a short way to do it and I didn't
have time as this is almost 8am), so I can't show any results. but this code works. this
code points to where 'filtable' starts (lets assume that is where the 32 data points are
at) and run the loop 32 times.After 32 times. Y pointer increments and program the
result there. I tried my best, please take it easy.
                      */

.include "ATxmega128A1Udef.inc"
.list

.equ   stack = 0x2FFF

.org 0x0000   ;start at address 0x000  rjmp MAIN ;jump to main code

.dseg
 .org 0x3000
store: .byte 1

.cseg
.org 0x200

MAIN:
ldi YL, low(stack)
       out CPU_SPL, YL

ldi YL, high(stack)
       out CPU_SPH, YL                   ;initialize high byte of stack pointer

rcall routine


DONE:
       rjmp DONE

routine:
push YL
push YH
push XL
push XH
push r16
push r17
push r20

ldi YL, low(filtable)  ;Y pointer point to where the table will start
ldi YH, high(filtable) ;low and high bytes
ldi XL, low(store)
ldi XH, high(store)

ldi r17, 32     ;counter
```

```
Third:
ld r16, Y+
mov r20, r16
ld r16, X
cp r20, r16
brlo store16
st X, r20
rjmp compare


store16:
st X, r16

compare:
cpi r17, 0
breq startpop
dec r17
rjmp Third

startpop:
ld r16, X
st Y, r16
pop r20
pop r17
pop r16
pop XH
pop XL
pop YH
pop YL
ret
```

```
/* HW2- Stack Pointer
Name: Pengzhao Zhu
Section#: 112D
TA Name: Chris Crary
Description: Stack Pointer Example. Trying to get myself familiar how to use stack
pointers.

*/
.include "ATxmega128A1Udef.inc"
.list

.equ stack=0x2FFF
.EQU length= 10                     ;test vector length

.ORG 0x100
VECTOR:       .DB 1,2,3,4, 5, 6, 7, 8, 9       ;trying to mess around the table with
pointers. but couldn't do it

.org 0x00
rjmp MAIN

.org 0x200
MAIN:
       ldi YL,low(stack)
       out CPU_SPL, YL
       ldi YH, high(stack)
       out CPU_SPH, YH
       ldi XL, 0x37    ;push 0x37
       push XL
       ldi XH, 0xAB    ;push 0xAB
       push XH
       ldi YL, 0x12    ;push EF12
       push YL
       ldi YH, 0xEF
       rcall routine   ;call the subroutine

       rjmp DONE

DONE:
       rjmp DONE

routine:
ldi YH, 0x1C      ;the subroutine
push YH           ;push like Dr. Schwartz told me
ret
```


HW plays (Debugging) - AtmelStudio

```
.org 0x00
rjmp MAIN

.org 0x200
MAIN:
    ldi YL,low(stack)
    out CPU_SPL, YL
    ldi YH, high(stack)
    out CPU_SPH, YH
    ldi XL, 0x37    ;push 0x37
    push XL
    ldi XH, 0xAB    ;push 0xAB
    push XH
    ldi YL, 0x12    ;push EF12
    push YL
    ldi YH, 0xEF
    rcall routine   ;call the subroutine

    rjmp DONE

DONE:
    rjmp DONE

routine:
ldi YH, 0x1C        ;the subroutine
push YH             ;push like Dr. Schwartz told me
ret
```

Memory 4

Memory: prog APP_SECTION   Address: 0x000400,prog

```
prog 0x000400  cf ef cd bf df e2 de bf a7 e3 af 93 bb ea bf 93  ÏïÍ¿ßâÞ¿§ã¯“»ê¿“
prog 0x000410  c2 e1 cf 93 df ee 02 d0 00 c0 ff cf dc e1 df 93  ÂáÏ“ßî.Ð.ÀÿÏÜáß“
prog 0x000420  08 95 ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ..ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000430  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000440  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000450  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000460  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000470  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000480  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000490  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0004A0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0004B0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Stopped

Yeah, it didn't work. I tried my best.