ECED 3204 Microprocessor

Midterm Reference Solution

Date: October 26 2016 **Room: Sexton Gym**
3:30PM-5:30PM

Student name	
ID	

1) Write an instruction sequence to add 35 to the two-byte value stored at data memory locations 0x2000~0x2001.

[10points]

Solution: The following instruction sequence will add 35 to the data memory locations 0x2000~0x2001:

```
ldi     YL,0x00         ; use Y to point to data memory location 0x2000
ldi     YH,0x20         ;    "
ld      r24,Y           ; fetch data memory location 0x2000
ldd     r25,Y+1         ;    "
adiw    r24,35          ; add 35 to r25:r24
st      Y,r24           ; store the sum back to the original data memory
std     Y+1,r25         ; locations
```

2) Read following program and determine the value of r17, r18, r19 and r20, and the content at address buf， buf+1， buf+2, and buf+3 when the program is done.

[15points]

	r17	r18	r19	r20
binary value	00000111	10011000	01100101	00110100
	buf+3	buf+2	buf+1	buf
binary value	01111001	10000110	01010011	01000010

```
            .include    <m2560def.inc>
            .equ    bt3     = 0x79 ; msb of test data
            .equ    bt2     = 0x86 ;        "
            .equ    bt1     = 0x53 ;        "
            .equ    bt0     = 0x42 ; lsb of test data
            .def    lpcnt   = r16
            .dseg
            .org    0x200  ; set to 0x2000 for Xmega devices
buf:        .byte   4      ; buffer to hold right-shift result
            .cseg
            .org    0x00
            rjmp    start
            .org    0xF6
start:      ldi     r17,bt3 ; msb
            ldi     r18,bt2 ; second to msb
            ldi     r19,bt1 ; second to lsb
            ldi     r20,bt0 ; lsb
            sts     buf+3,r17       ; initialize msb in data memory
            sts     buf+2,r18       ; initialize second to msb in data memory
            sts     buf+1,r19       ; initialize second to lsb in data memory
            sts     buf,r20 ; initialize lsb in data memory
            ldi     lpcnt,4 ; set up loop count
srloop:     lsr     r17     ; shift the msb to the right one place
```

```
    ror     r18     ; rotate the second to msb to the right one place
    ror     r19     ; rotate the second to lsb to the right one place
    ror     r20     ; rotate the lsb to the right one place
    dec     lpcnt   ; decrement loop count
    brne    srloop

    nop
// end of program
```

3) Write a loop to compute the sum of the square of the first one hundred odd numbers. [10 points]

Solution:

```
long  int  oddSum;
int    ix;
…
oddSum = 0;
for (ix = 1; ix < 201; ix++2){
      oddSum = (unsigned long)ix * (unsigned long)ix;
}
```

4) Load your last two digit banner number in hexadecimal format to register Rd then execute each following instructions and write out the new Rd value. Assume K=0x53.

[15points]

Your Rd in Hexdecimal	
	Results
and Rd, Rd	
andi Rd, K	
ori Rd, K	
sbr Rd, K	
cbr Rd, K	
clr Rd	
ser Rd	
com Rd	
neg Rd	
eor Rd, Rd	

5) What are the contents of PORTA after the following AVR program is assembled:
[5points]

```
out     PORTA,0x53
sbi     PORTA,2
cbi     PORTA,1
```

solution： 01010101

6) Write a subroutine that can divide a 24-bit unsigned integer into another 24-bit unsigned integer. Please also write a test program for the subroutine.
[15points]

```
.include <atmega644def.inc>
.macro   dealloc_stk
in       r28,SPL
in       r29,SPH
adiw     r28,@0          ; allocate k bytes in the stack
out      SPL,r28
out      SPH,r29
.endmacro
.equ     dd2 = 0x58      ; test data for dividend
.equ     dd1 = 0x41      ;    "
.equ     dd0 = 0x70      ;    "
.equ     dr2 = 0x00      ; test data for divisor   "
.equ     dr1 = 0x21      ;    "
.equ     dr0 = 0x98      ;    "

.cseg
.org     0x00
rjmp     start
.org     0xF6
start:   ldi      YL,low(RAMEND) ; initialize stack pointer SP
         ldi      YH,high(RAMEND)    ;         "
         out      SPL,YL         ;    "
         out      SPH,YH         ;    "
         ldi      r28,dr2        ; pass divisor in the stack
         push     r28            ;    "
         ldi      r28,dr1        ;    "
         push     r28            ;    "
         ldi      r28,dr0        ;    "
         push     r28            ;    "
         ldi      r16,dd0        ; pass dividend in r16~r18
         ldi      r17,dd1        ;    "
         ldi      r18,dd2        ;    "
         call     div24U         ;
         dealloc_stk 3
forever: rjmp     forever
; ---------------------------------------------------------------------------------------------------------------
```

```
; The subroutine divides a 24-bit unsigned integer into another 24-bit unsigned
; integer. The 24-bit divisor is passed in the stack whereas the 24-bit
; unsigned dividend is passed in registers r16~r18. The quotient is returned in
; registers r22~r24 and the remainder is returned in registers r16~r18.
; ---------------------------------------------------------------------------------------------------------------
            .def    lpcnt   = r21     ; loop count for repeated shifted-subtraction
            .def    tmp2    = r10     ; used to hold difference temporarily
            .def    tmp1    = r9      ;    "
            .def    tmp0    = r8      ;    "
            .equ    dvr2    = 7       ; offset of msb of divisor from stack pointer
            .equ    dvr1    = 6       ;    "
            .equ    dvr0    = 5       ;
Div24U:     push    YH                ;
            push    YL                ;    "
            in      YL,SPL            ; use Y register as pointer to the stack top
            in      YH,SPH            ;    "
            ldi     lpcnt,24          ; there should be 24 iterations
            movw    r22,r16           ; transfer dividend to Q register
            mov     r24,r18           ;    “
            ldd     r0,Y+dvr0         ; place divisor in registers r0~r2
            ldd     r1,Y+dvr1         ;    "
            ldd     r2,Y+dvr2         ;    "
            clr     r16               ; use registers r16~r18 as R register and
            clr     r17               ; initialize them to 0
            clr     r18               ;    "
dvloop24:   lsl     r22               ; shift R:Q to the left one plce
            rol     r23               ;    "
            rol     r24               ;    "
            rol     r16               ;    "
            rol     r17               ;    "
            rol     r18               ;    "
            movw    tmp0,r16          ; transfer R register to tmp0~tmp3
            mov     tmp2,r18          ;    "
            sub     tmp0,r0
            sbc     tmp1,r1
            sbc     tmp2,r2
            brmi    less24
            movw    r16,tmp0          ; save the difference in R register
            mov     r18,tmp2          ;    "
            ori     r22,0x01          ; set the lsb of Q register to 1
            rjmp    nextb
less24:     andi    r22,0xFE          ; clear the lsb of Q register to 0
nextb:      dec     lpcnt
            brne    dvloop24
            pop     YL                ; restore Y from stack
            pop     YH                ;    "
            ret
// End of program
```

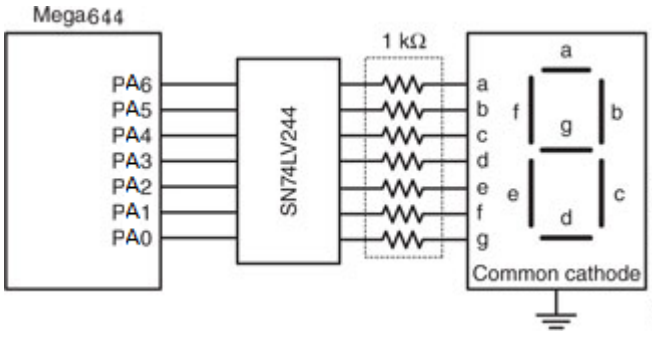
7) Write a function to find numbers of odd integers from an array and store the results at 0x1100.

[10points]

```
        .include   <m644Adef.inc>
           .equ    NN      = 20
           .def    j       = r16   ;
           .def    Cnt     = r17
           .def    tmp     = r18   ;
           .cseg
           .org    0x00
           rjmp    start
           .org    0xF6
start:     ldi     ZL,low(array<<1)     ;
           ldi     ZH,high(array<<1)    ;        "
           ldi     j,0
           ldi     Cnt,0
loop:      lpm     tmp,z+ ;
           sbrs    tmp,1  ;
           inc     Cnt
 next:     inc     j
           cpi     j,NN-1 ;
           brne    loop
           ldi     Xl,0x00
           ldi     XH,0x11
           mov     X,Cnt
here:      rjmp    here    ;
array:     .db     11,12,13,14,15,16,17,18,19,20
           .db     21,22,23,24,25,26,27,28,29,30
        // end of program
```

8) Use the AVR Mega PORTA to drive a single seven-segment display. Please write assembling code to display decimal digits 2,0,1,6 with 100 ms delay between digits. Assume the subroutine for delay is available **delayby100ms.**

[10points]



```
            .cseg
            .org    0x00
            jmp     start
            .org    0xF6
Start：     ldi     tmp, low(RAMEND)
            out     SPL,tmp
            ldi     tmp, high(RAMEND)
            out     SPH, tmp
            ldi     r16, 0xFF           ;
            out     DDRA, r16

Display:    ldi     r16, 0x6D
            sts     PORTA, r16
            call    delayby100ms
            ldi     r16, 0x7E
            sts     PORTA, r16
            call    delayby100ms
            ldi     r16, 0x30
            sts     PORTA, r16
            call    delayby100ms
            ldi     r16, 0x5F
            sts     PORTA, r16
            call    delayby100ms
            jmp     display
```

9) Configure odd interrupt pins (INTn) to interrupt on the rising edge and the even interrupt pins to interrupt on the falling edge.

[10points]

Solution: Each interrupt pin (eight in total) uses two bits to configure the interrupt edge. The following instruction sequence will configure the odd interrupt pins to interrupt on the rising edge and the even interrupt pins to interrupt on the falling edge:

```
ldi  r16,0xEE     ; configure odd interrupt pins to interrupt on the rising edge
sts  EICRA,r16  ; and even interrupt to interrupt on the falling edge
sts  EICRB,r16  ;    "
ldi  r16,0xFF     ; enable all INTn pin to interrupt locally
sts  EIMSK,r16  ;    "
sei  ; enable global interrupt
```