```
/* Homework 2 Problem 1
 *
 * Copies switch pattern to LEDs, but
 * uses "or" so the LED never goes off
 * once it is set.
 */

.include "ubc-de1media-macros.s"

.global _start

.text
_start: movia   r23, IOBASE
                movi    r3, 0

loop:           ldwio   r2, SWITCH(r23)
                or      r3, r3, r2
                stwio   r3, LEDR(r23)

                br      loop
.end
```

```
/* Homework 2 Problem 2
 *
 * Computes A*B + C/D, where
 * A, B, C and D are defined below.
 *
 * Note:
 * B and D are constants, so they can be used
 * with "immediate" instructions like "muli"
 * and "movi". (Note: movi is used with D
 * because there is no divi instruction.)
 *
 * A and C are labels that point to variables stored
 * in memory, so you must use "ldw" to get their value.
 */

.equ    B, 5
.equ    D, 6

.global _start

.text
_start:         movia   r10, A
                ldw     r2, 0(r10)              /* read A */
                muli    r11, r2, B              /* compute A*B */

                movia   r10, C
                ldw     r2, 0(r10)              /* read C */
                movi    r3, D
                div     r12, r2, r3             /* compute C/D */

                add     r11, r11, r12   /* add A*B and C/D */

STOP:           br      STOP

.data
A:
.word 7
C:
.word 24

.end
```

```
/* Homework 2 Problem 3
 *
 * Copies switch pattern to LEDs only
 * when KEY3 changes from 0 to 1.
 *
 * Question: when KEY3 is not being pressed,
 * is it a 0 or a 1? How can you find out?
 *
 */

.include "ubc-de1media-macros.s"

.equ            KEY3mask,       0x08

.global _start

.text
_start:         movia   r23, IOBASE

loop:

/* wait for KEY3 to go from 1 back to 0 */
/* without this wait, it behaves like a flow-through latch */
while1:         ldwio   r3, KEY(r23)
                andi    r3, r3, KEY3mask
                bne     r3, r0, while1  /* wait while KEY3=1 */

/* wait for KEY3 to go from 0 to 1 */
while0:         ldwio   r3, KEY(r23)
                andi    r3, r3, KEY3mask
                beq     r3, r0, while0  /* wait while KEY3=0 */

/* since KEY3 went from 0 to 1, copy switches to LEDs */
copy:           ldwio   r2, SWITCH(r23)
                stwio   r2, LEDR(r23)

                br      loop
.end
```

```
/* Homework 2 Problem 4
 *
 * Count the number of times SW0 is moved to the "1" position,
 * and display this on the red LEDs.
 *
 * Sometimes the count goes up by more than 1 when
 * you move the switch.  Why do you think this happens?
 */

.include "ubc-de1media-macros.s"

.equ            SW0mask,        0x01

.global _start

.text
_start:         movia   r23, IOBASE
                movi    r4, 0

loop:

/* wait for SW1 to go from 1 to 0 */
while1:         ldwio   r3, SWITCH(r23)
                andi    r3, r3, SW0mask
                bne     r3, r0, while1  /* wait while SW0=1 */

/* wait for SW0 to go from 0 to 1 */
while0:         ldwio   r3, SWITCH(r23)
                andi    r3, r3, SW0mask
                beq     r3, r0, while0  /* wait while SW0=0 */

increment:      addi    r4, r4, 1
                stwio   r4, LEDR(r23)

                br      loop

.end
```

```
/* Homework 2 Problem 5
 *
 * Count the number of times SW0 is moved to the "1" position,
 * and display this on the 7-segment display.
 *
 * After a count of 9, it wraps around to 0.
 *
 */

.include "ubc-de1media-macros.s"

.equ    SW0mask,        0x01

.global _start

.text
_start:         movia   r23, IOBASE
                movi    r4, 0

loop:

/* display count on LEDs and 7SEG */
display:        call    count2ten               /* keeps count between 0
    and 9 */
                stwio   r2, LEDR(r23)
                mov     r4, r2

                call    ten2hex7seg     /* converts decimal to 7seg value
                    */
                stwio   r2, HEX7SEG(r23)

/* wait for SW1 to go from 1 to 0 */
while1:         ldwio   r3, SWITCH(r23)
                andi    r3, r3, SW0mask
                bne     r3, r0, while1  /* wait while SW0=1 */

/* wait for SW0 to go from 0 to 1 */
while0:         ldwio   r3, SWITCH(r23)
                andi    r3, r3, SW0mask
                beq     r3, r0, while0  /* wait while SW0=0 */

increment:      addi    r4, r4, 1

                br      loop

/* function: count2ten
 * operation: reduces the count to a value between 0 and 9
 *            by subtracting all multiples of 10.
 * incoming parameter: r4 is a count
 * return value: r2 is between 0 and 9 inclusive
 */

count2ten:
```

```
                movi    r2, 10
                div     r2, r4, r2
                muli    r2, r2, 10
                sub     r2, r4, r2
                ret

/* function: ten2hex7seg
 * operation: converts a value between 0 and 9 into the 32-bit
 *            value needed for the 7-segment display.
 * incoming parameter: r4 is a value between 0 and 9 inclusive
 * return value: r2 is a 32-bit value for the 7-segment display
 */
ten2hex7seg:
                movia   r2, TABLE               /* use a lookup table */
                muli    r3, r4, 4
                add     r2, r2, r3              /* 4*r4 is # of bytes into
                    table for 7segment value */
                ldw     r2, 0(r2)               /* lookup value at address
                    TABLE + 4*r4 */
                orhi    r2, r2, 0xffff  /* turn off LEDs in HEX3 and HEX2
                    */
                ori     r2, r2, 0xff00  /* turn off LEDs in HEX1 */
                ret

.data
TABLE:
.word DIGIT0, DIGIT1, DIGIT2, DIGIT3, DIGIT4
.word DIGIT5, DIGIT6, DIGIT7, DIGIT8, DIGIT9

.end
```

```
/* Homework 2 Problem 6
 *
 * Decompress the values in the data section.
 * The COMPRESSED section contains pairs of words, COUNT and VALUE.
 * The DECOMPRESSED section should have COUNT copies of VALUE.
 * This repeats with each pair until a COUNT of 0 is discovered.
 */

.global _start

.text
_start: movia   r2, COMPRESSED
                movia r3, DECOMPRESSED

newpair:        ldw     r10, 0(r2)      /* COUNT */
                beq     r10, r0, STOP

/* write the word VALUE to DECOMPRESSED exactly COUNT times */
                ldw     r11, 4(r2)      /* VALUE */
nextword:       stw     r11, 0(r3)
                addi    r3, r3, 4
                subi    r10, r10, 1
                bne     r10, r0, nextword

/* advance to the next COUNT,VALUE pair in COMPRESSED stream */
                addi    r2, r2, 8
                br      newpair

STOP:           br      STOP


.data

COMPRESSED:
.word 3, 0xEECE, 2, 0x0259, 4, 0xF00D, 5, 0xCAFE, 0

DECOMPRESSED:
.skip 4*(3+2+4+5)

.end
```