More I/O

① **masking** is the process of isolating a single bit (or set of bits) by forcing all others to a known value (0 or 1)

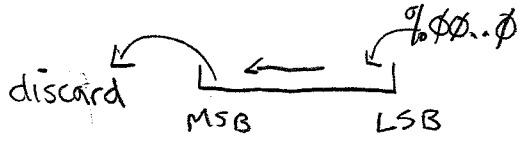
and – force others to 0, eg [andi r2, r2, 0x10

or – force others to 1, eg [ori r2, r2, 0xFFEF
orhi r2, r2, 0xFFFF

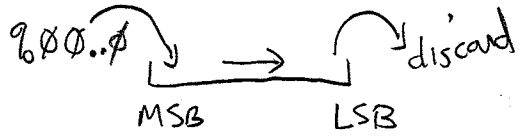
In both examples, bit #4 is left intact

② **shifting** is the process of moving bits left or right

shift left logical



shift right logical



sll r3, r4, r6
srl r3, r4, r6

value to shift

amount of shift

also: slli r3, r4, 16
srli r3, r4, 3

small 5-bit constant (unsigned)

example scrolling LEDs

```
_start:  movia  r23, 10BASE
restart: movi   r2, 0x03
loop:    stwio  r2, LEDR(r23)
         call   delay-1s
         slli   r2, r2, 1
         bne    r2, r0, loop
         br     restart
```

③ precise timing — COUNTER is a 32 bit hardware timer that counts up by 1 every 50MHz clock cycle

1 cycle = 20 ns

(2^{32} cycles = 4.294967296 x 10^9 x 20 x 10^-9)

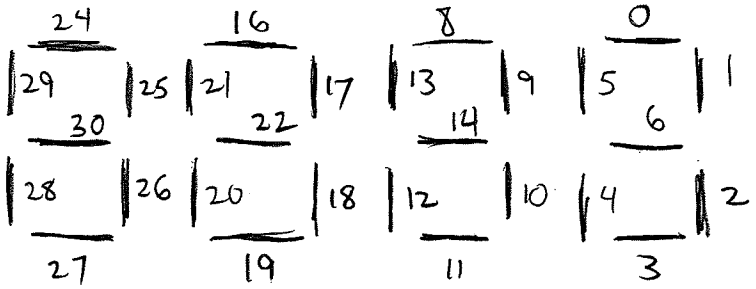
≈ 86 seconds

```
ldwio r2, COUNTER(r23)
{ do something
ldwio r3, COUNTER(r23)
sub   r3, r3, r2      /* elapsed time
                         in clock cycles */
```

④ external inputs — GPIO0 and GPIO1 are connectors on your DE1 board, 32 bits each

```
ldwio r2, GPIO1(r23)
```

⑤ 7 segment display



write 32b value to HEX7SEG: 0 = on, 1 = off

```
movia  r2, 0x8686C686
stwio  r2, HEX7SEG(r23)
```

```
/************* ubc-de259-macros.s ***************/

/*
 * Version 3.00 February 7, 2009
 *
 */

/* I/O addresses, eg:
 *    movia r23, ADDR_LEDR
 *    movi  r2, 0x08
 *    stwio r2, 0(r23)
 */
.equ   ADDR_LEDR  , 0x00004800    /* output only, RED (DE1 10b, DE2 18b) */
.equ   ADDR_LEDG  , 0x00004810    /* output only, GREEN (DE1 8b, DE2 9b) */
.equ   ADDR_SWITCH, 0x00004820    /* input only, (DE1 10b, DE2 18b) */
.equ   ADDR_KEY,           0x00004830    /* input only, KEY3 to KEY0  (4b) */
.equ   ADDR_HEX7SEG,0x00004840    /* output only, HEX3 to HEX0 (32b) */
.equ   ADDR_HEX7SEGA,      0x00004840    /* output only, HEX3 to HEX0 (32b) */
.equ   ADDR_HEX7SEGB,      0x00004850    /* output only, HEX7 to HEX4 (32b) only on DE2 */
.equ   ADDR_COUNTER,     0x00004860      /* input only, 50 MHz counter (32b) */
.equ   ADDR_PS2,           0x00004870    /* in/out, PS2 connector for kb and mouse (8b data)*/
.equ   ADDR_RS232,  0x00004878    /* send/recv chars to RS232 terminal */
.equ   ADDR_AUDIO,  0x00004880    /* audio device */

.equ   ADDR_JTAG,          0x00004890    /* send/recv chars to Terminal window in Debug Client */
.equ   ADDR_SYSID,         0x00004898    /* unique systemID */

.equ   ADDR_GPIO0,  0x000048A0    /* I/O connections to connector GPIO0 */
/* Note: GPIO1 is not available. Instead, the GPIO1 connector is
 * dedicated for the use of the graphics LCD panel
 *     .equ   ADDR_GPIO1,  0x000048B0    / * I/O connections to connector GPIO1 * /
 */

.equ   ADDR_IRDA,          0x000048C0    /* DE2 IrDA controller */
.equ   ADDR_LCD,           0x000048D0    /* DE2 16x2 character LCD */
.equ   ADDR_TIMER,         0x000048E0    /* timer controller */

.equ   ADDR_PLLSTATUS,     0x00004900    /* PLL lock status input: b0=vga, b1=sdram, b2=audio */
.equ   ADDR_LCDCONFIG,     0x00004910    /* configure pixel LCD */
.equ   ADDR_ACONFIG,0x00004920    /* configure audio */
.equ   ADDR_VCONFIG,0x00004930    /* configure vga */
.equ   ADDR_DCCONFIG,      0x00004950    /* configure digicam */

.equ   ADDR_SDCARD, 0x00004C00    /* SD card slot controller */

.equ   ADDR_CHARS,  0x00010000    /* vga character framebuffer */
.equ   ADDR_VGA,           0x00080000    /* vga pixel framebuffer */


.equ   ADDR_SRAM,          0x00000000    /* onchip SRAM */
.equ   ADDR_FLASH,         0x00400000    /* onboard Flash ROM */
.equ   ADDR_SDRAM,         0x00800000    /* onboard SDRAM */

.equ   SRAM_START,         0x00000000    /* onchip SRAM 8kB */
.equ   SRAM_END,           0x00002000    /* end of onchip SRAM (this should default to DE1) */
.equ   SRAM_END_DE1,0x00002000    /* end of onchip SRAM for DE1 */
.equ   SRAM_END_DE2,0x00002000    /* end of onchip SRAM for DE2 */

.equ   DRAM_START,         0x00800000    /* onboard SDRAM 8MB */
.equ   DRAM_END,           0x01000000    /* end of onboard SDRAM 8MB */


/* Address offsets for preferred form of I/O, eg:
 *    movia r23, IOBASE
 *    movi  r2, 0x08
 *    stwio r2, LEDR(r23)
 */
.equ   IOBASE,             ADDR_LEDR
.equ   LEDR   ,            (ADDR_LEDR          - IOBASE)
.equ   LEDG   ,            (ADDR_LEDG          - IOBASE)
.equ   SWITCH,             (ADDR_SWITCH  - IOBASE)
.equ   KEY    ,            (ADDR_KEY           - IOBASE)
.equ   HEX7SEG,            (ADDR_HEX7SEG - IOBASE)
.equ   HEX7SEGA,           (ADDR_HEX7SEGA      - IOBASE)
.equ   HEX7SEGB,           (ADDR_HEX7SEGB      - IOBASE)
.equ  COUNTER,          (ADDR_COUNTER   - IOBASE)
.equ   PS2,                (ADDR_PS2           - IOBASE)

.equ   SNDCTL,             (ADDR_AUDIO - IOBASE + 0)
```

```
.equ   SNDRDY,              (ADDR_AUDIO - IOBASE + 4)
.equ   SNDL,                (ADDR_AUDIO - IOBASE + 8)
.equ   SNDR,                (ADDR_AUDIO - IOBASE + 12)

.equ   TERMINAL,            (ADDR_JTAG  - IOBASE)
.equ   TERMINALCTL,  (ADDR_JTAG  - IOBASE + 4)
.equ   RS232,        (ADDR_RS232 - IOBASE)
.equ   RS232CTL,            (ADDR_RS232 - IOBASE + 4)

.equ   ACONFIG,             (ADDR_ACONFIG - IOBASE)
.equ   VCONFIG,             (ADDR_VCONFIG - IOBASE)
.equ   SNDCONFIG,           (ADDR_ACONFIG - IOBASE + 8)

.equ   GPIO0,        (ADDR_GPIO0 - IOBASE)
.equ   GPIO0DIR,            (ADDR_GPIO0 - IOBASE + 4)
/*     .equ   GPIO1,        (ADDR_GPIO1 - IOBASE)
 *     .equ   GPIO1DIR,            (ADDR_GPIO1 - IOBASE + 4)  */

.equ   LCDCTL,              (ADDR_LCD - IOBASE)
.equ   LCDDATA,             (ADDR_LCD - IOBASE + 4)

.equ   TIMER_STATUS,(ADDR_TIMER - IOBASE + 0)
.equ   TIMER_CONTROL,       (ADDR_TIMER - IOBASE + 4)
.equ   TIMER_START_LOW,     (ADDR_TIMER - IOBASE + 8)
.equ   TIMER_START_HIGH,    (ADDR_TIMER - IOBASE + 12)
.equ   TIMER_VALUE_LOW,     (ADDR_TIMER - IOBASE + 16)
.equ   TIMER_VALUE_HIGH,    (ADDR_TIMER - IOBASE + 20)

/*
 * Each bit position at address ADDR_HEX7SEG works as follows:
 *
 *    HEX3         HEX2         HEX1        HEX0
 *     24           16           8           0
 *    ----         ----         ----        ----
 * 29|    |25  21|    |17  13|    |9   5|    |1
 *   | 30 |      | 22 |      | 14 |     |  6 |
 *    ----         ----         ----        ----
 *   |    |       |    |       |    |     |    |
 * 28|    |26  20|    |18  12|    |10  4|    |2
 *    ----         ----         ----        ----
 *     27           19           11          3
 * Note that bits 31, 23, 15, and 7 are not used.
 *
 * Write "1" to a bit to turn "OFF" a 7-segment display LED.
 *
 */
.equ   DIGIT0,      0xC0
.equ   DIGIT1,      0xF9
.equ   DIGIT2,      0xA4
.equ   DIGIT3,      0xB0
.equ   DIGIT4,      0x99
.equ   DIGIT5,      0x92
.equ   DIGIT6,      0x82
.equ   DIGIT7,      0xF8
.equ   DIGIT8,      0x80
.equ   DIGIT9,      0x98
.equ   DIGITA,      0x88
.equ   DIGITB,      0x83
.equ   DIGITC,      0xC6
.equ   DIGITD,      0xA1
.equ   DIGITE,      0x86
.equ   DIGITF,      0x8E
```

```
/**** Simple NIOS I/O Examples ****/

/* Example 2. Write a message on the HEX display. */

.include "ubc-de259-macros.s"

.global _start

.text
_start:     movia r23, IOBASE       /* use r23 as base register for I/O operations */

            movi  r3, DIGIT0

            slli  r3, r3, 8
            ori   r3, r3, DIGIT2

            slli  r3, r3, 8
            ori   r3, r3, DIGIT5

            slli  r3, r3, 8
            ori   r3, r3, DIGIT9

            stwio r3, HEX7SEG(r23)  /* put on hex display */

STOP:       br    STOP

.end
```

```
/**** Simple NIOS I/O Examples ****/

/* Scroll a message across the HEX display. */

.include "ubc-de259-macros.s"

.equ DIGITBLANK,    0xff
.equ ALLBLANK,      0xffffffff

.global _start

.text
_start:      movia  r23, IOBASE         /* use r23 as base register for I/O operations */

             movia  r8, ALLBLANK        /* initialize hex display, all "OFF" */
             stwio  r8, HEX7SEG(r23)

             movia  r10, TABLE_END
reset: movia  r9, TABLE_START

loop:  ldwio  r11, 0(r9)                /* read next digit from table */

             slli   r8, r8, 8           /* shift all digits one position left */
             or     r8, r8, r11         /* OR in value for next digit (next 8 bits) */

             stwio  r8, HEX7SEG(r23)
             call   delay

             /* advance to next digit in table */
             addi   r9, r9, 4
             bne    r9, r10, loop       /* not at table end, shift in next table entry */

             br     reset               /* at table end, start over with first table entry */


             /* delay subroutine */
delay: movia  r2, 5000000                /* count from 5,000,000 for ~0.2s delay */
delay_loop:  subi   r2, r2, 1
             bne    r2, r0, delay_loop  /* delay by counting down to 0 */
             ret

.data

TABLE_START:
.word DIGITE, DIGITE, DIGITC, DIGITE
.word DIGIT0, DIGIT2, DIGIT5, DIGIT9
.word DIGITBLANK
.word DIGITC, DIGITA, DIGITF, DIGITE
.word DIGITB, DIGITA, DIGITB, DIGITE
.word DIGITBLANK
TABLE_END:

.end
```