Real Time Embedded Systems
Worksheet 2, I/O Programming

Wordlength

Questions in the previou[illegible]ed that you were working with 16-bit values, which are the default for this proce[illegible]erations, however, it is usually necessary to transfer 8 bit (1-byte) values. Instr[illegible]e to act on 8 bits, instead of 16, by suffixing the instruction mnemonic w[illegible]

For example, if the LED[illegible]o address 2000H, then the following sequence will set it to logic-1.

```
        move.b      #$01,d0     ;moves 8 bits with the value 01H to the RH 8 bits in D0
        move.b      d0,$2000    ;moves RH 8 bits of D0 to location 2000H
```

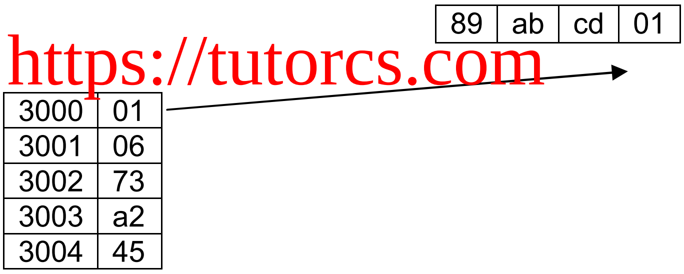
Note that a '.B' instruction will act only on the RH byte of the register. For example, if D0 contains the following value

89	ab	cd	ef

then the instruction

```
        move.b      $3000,d0
```

will move the byte from memory location 3000H to the RH byte of the register, leaving the other three bytes unchanged.



89	ab	cd	01

3000	01
3001	06
3002	73
3003	a2
3004	45

There will be other occasions when you need to work with values that are too large to be represented by 16 bits. The registers are, in fact, 32 bits (4 bytes) long. 32-bit operations may be specified by suffixing the instruction with '.L' (longword).

```
        move.l      $2000,d0    ;move 32 bits from locations 2000H .. 2003H to D0
        move.l      $2004,d1    ;move 32 bits from locations 2004H .. 2007H to D1
        add.l       d0,d1       ;adds all 32 bits in D0 to D1
```

You need to exercise extreme care when, as is often necessary, you are using byte and longword operations within the same code sequence. Think carefully about what will happen when the unchanged upper 3 bytes following a .B instruction are subsequently used as an input to a .L instruction. Try some examples on the simulator, and single-step through the programme if in any case you are unsure.

Practical Work

1.

The simulator has 8 pus[illegible] napped to address E00014H and wired so that each switch returns a logic-0 [illegible] ogic-1 when released. It also has 8 LEDs, mapped to address E00010H. Prog[illegible] so that the RH LED changes state each time the RH switch is pressed, and c[illegible] simulator.

2.

The simulator also cont[illegible]splays. From left to right, the digits are mapped to addresses E00000, E00002, E00004 .. E0000E. Programme the simulator so that the right-hand digit displays zero, and increments in hexadecimal up to F each time the RH push-button is pressed. After the display reaches F, it resets to zero.

Each segment is set by one of the bits in the output byte. The following patterns correspond to each displayed value.

```
kseg                                    ;7-seg display patterns
          dc.b     $3f          ;0
          dc.b     $06          ;1
          dc.b     $5b          ;2
          dc.b     $4f          ;3
          dc.b     $66          ;4
          dc.b     $6d          ;5
          dc.b     $7d          ;6
          dc.b     $07          ;7
          dc.b     $7f          ;8
          dc.b     $67          ;9
          dc.b     $77          ;A
          dc.b     $7c          ;b
          dc.b     $39          ;C
          dc.b     $5e          ;d
          dc.b     $79          ;E
          dc.b     $71          ;F
          dc.b     $80          ;.
```

Nowadays, memory is so large and cheap that there is little to be gained from using 16-bit values, and recent versions of this processor therefore perform all operations except moves at 32 bits. Therefore, all instructions except move should be suffixed '.L', and will relate to 4-byte values. Move operations may optionally still work at 8 bits, in order to deal with character data or to allow data transfers to 8-bit peripheral devices, so a move instruction may be suffixed either '.L' or '.B'. Although the simulator is designed to accommodate older devices that did use 16-bit values (no suffix), it would be a good idea from now on to work with 32-bit values consistently, except in some move instructions for which it is essential to use 8-bits. Answers to this and all subsequent questions have been programmed accordingly.

3. ***Assessment question***
Work in pairs on this question, and keep a copy of your answer, together with a note of the input you used to test it.

The 8 push button switches each correspond to one bit in the byte at E00014H. The left hand switch corresponds to bit 7, and the right hand one to bit 0. Programme the simulator so that the user may press any of these switches, one at a time, in a sequence of any length. The user then indicates that the sequence is complete by pressing the RH permanent switch which is similarly mapped to bit 0 at address E00012H. At this point, the system plays the sequence back by lighting the LEDs that correspond to each of the input values, in the order in which they were entered.

The sequence will probably play back very quickly. Therefore you should insert a short delay before changing from one LED to the next, so that the playback sequence is clearly visible. A delay can be programmed by setting [illegible] value, then executing a loop that repeatedly decrements it until it rea[illegible]e number of instructions executed as a result will hold the simulator up for a n[illegible]me, but since the simulator will run at different speeds on different computers, [illegible] some experimentation to find a value that works well on yours.

Answers

1.

```
*-----------------------------------------------------------
* Title       : Single-task loop
* Written by  : JNC
* Date        :
* Description: Toggles LED0 when pushbutton SW0 is pressed
*-----------------------------------------------------------

led     equ     $e00010     ;led
sw      equ     $e00014     ;switch

        org     $1000

start:  move    #0,d0       ;set led off
        move    d0,ledstat
        move.b  d0,led
                            ;repeat
l0:     move.b  sw,d0       ;  wait until switch pressed
        and     #1,d0
        bne     l0
                            ;*********

        move    ledstat,d0  ;  invert led
        eor     #$01,d0
        move    d0,ledstat
        move.b  d0,led

l1:     move.b  sw,d0       ;  wait until switch released
        and     #1,d0
        beq     l1

        bra     l0

ledstat ds      1           ;led state

        end    start
```

Try moving the three lines of code at l1 to the point marked with a row of asterisks. What effect does this have on the behaviour of the programme?

2.

```
*------------------------------------------------------------
* Title       : Single-task loop
* Written by  : JNC
* Date        :
* Description: Incr[illegible]n pushbutton SW0 is pressed
*------------------------------------------------------------

sevseg  equ     $e0[illegible]
sw      equ     $e0[illegible]

        org     $10[illegible]

; the abbreviation '^' means 'holds the address of' or 'points to'
start:  move.l  #0,d1       ;set count in d1 = 0
        move.l  #kseg,a0    ;a0 ^ segment pattern table
        move.b  (a0),d0     ;set display = count
        move.b  d0,sevseg

        add.l   #1,d1       ;increment count

                            ;repeat
l1:     move.b  sw,d0       ;  wait until switch pressed
        and.l   #$1,d0
        bne     l1

        move.l  #kseg,a0    ;  set display
        add.l   d1,a0
        move.b  (a0),d0
        move.b  d0,sevseg

l2:     move.b  sw,d0       ;  wait until switch released
        and.l   #1,d0
        beq     l2

        add.l   #1,d1       ;  increment count

        cmp.l   #$10,d1     ;  if count = 10
        bne     l9

        move.l  #0,d1       ;    count = 0

l9:     bra     l1

kseg                        ;7-seg display patterns
        dc.b    $3f         ;0
        dc.b    $06         ;1
        dc.b    $5b         ;2
        dc.b    $4f         ;3
        dc.b    $66         ;4
        dc.b    $6d         ;5
        dc.b    $7c         ;6
        dc.b    $07         ;7
        dc.b    $7f         ;8
        dc.b    $67         ;9
        dc.b    $77         ;A
        dc.b    $7c         ;b
        dc.b    $39         ;C
        dc.b    $5e         ;d
        dc.b    $79         ;E
        dc.b    $71         ;F
        dc.b    $80         ;.

        end    start
```