```
/****************************************************************************/
/*                                                                          */
/*                                 DISPLAY.S                                */
/*                        Display Interface Functions                       */
/*                       Digital Oscilloscope Project                       */
/*                                 EE/CS 52                                 */
/*                               Santiago Navonne                           */
/*                                                                          */
/****************************************************************************/

/*
   Display interface and control routines for the EE/CS 52 Digital Oscilloscope
   project. Function definitions are included in this file, and are laid out
   as follows:
    - clear_display: Completely clears the display;
    - clear_trace: Clears the pixels on the display that are the color of the
                   trace;
    - plot_pixel: Changes the color of the pixel at a given location;
    - pixel_color: Accesses the color of the pixel currently being displayed at
                   a given location.


   Revision History:
      6/3/14  Santiago Navonne  Initial revision.
*/

#include "general.h"
#include "system.h"
#include "interfac.h"
#include "display.h"


.section .text  /* Code starts here */


/*
 *  clear_display
 *
 *  Description:       This procedure clears the display, setting the color of every
 *                     pixel to black immediately.
 *
 *  Operation:         The procedure loops through every pixel in the display-mapped
 *                     region of the VRAM, storing 0 (black; clear pixel) into every
 *                     location.
 *
 *  Arguments:         None.
 *
 *  Return Value:      None.
 *
 *  Local Variables:   None.
 *
 *  Shared Variables:  None.
 *
 *  Global Variables:  None.
 *
 *  Input:             None.
 *
 *  Output:            Clears every pixel on the display (changes color to black).
 *
 *  Error Handling:    None.
 *
 *  Limitations:       None.
 *
 *  Algorithms:        None.
 *  Data Structures:   None.
 *
 *  Registers Changed: r8, r9, r10, r11, r12.
 *
 *  Revision History:
 *      6/03/14   Santiago Navonne     Initial revision.
 *
 */
    .global clear_display
clear_display:                /* clear the whole display */
    MOVHI   r8, %hi(VRAM_BASE) /* start at base of VRAM */
    ORI     r8, r8, %lo(VRAM_BASE)
    MOVI    r9, SIZE_X          /* and will loop through all columns */
    MOVI    r10, SIZE_Y         /*  and rows */
    MOV     r11, r0             /* starting at coordinates (0, 0) */
    MOV     r12, r0             /* (top left corner) */

row_loop:                       /* go through an entire row */
    STWIO   r0, (r8)            /* first clear the current pixel */
    ADDI    r8, r8, WORD_SIZE   /* then go to next column */
    ADDI    r11, r11, 1         /*  also incrementing the index */
    BLT     r11, r9, row_loop   /* and if we're still within display, repeat */

next_row:                       /* move to next row */
    ADDI    r8, r8, REMAINDER   /* add the remainder to finish up a VRAM row */
    MOV     r11, r0             /* reset the column index */
    ADDI    r12, r12, 1         /* and increment the row index */
    BLT     r12, r10, row_loop  /* if we're still within display, repeat */

    RET                         /* all done, so return */

/*
 *  clear_trace
 *
 *  Description:       This procedure clears the trace from the display, changing the
 *                     color of every pixel that is currently the trace or cursor color
 *                     to black.
 *
 *  Operation:         The procedure loops through every pixel in the display-mapped
 *                     region of the VRAM. For every location, if the current value
 *                     matches either trace or cursor colors (both part of the trace)
 *                     the pixel is cleared by storing 0 into that memory location.
 *
 *  Arguments:         None.
 *
 *  Return Value:      None.
 *
 *  Local Variables:   None.
 *
 *  Shared Variables:  None.
 *
 *  Global Variables:  None.
 *
 *  Input:             None.
 *
 *  Output:            Clears every trace pixel on the display (sets color to black).
 *
 *  Error Handling:    None.
 *
 *  Limitations:       None.
 *
 *  Algorithms:        None.
 *  Data Structures:   None.
 *
 *  Registers Changed: r8, r9, r10, r11, r12, r14, r15.
 *
 *  Revision History:
 *      6/03/14   Santiago Navonne     Initial revision.
 *
 */
    .global clear_trace_old
clear_trace_old:                    /* clear all trace pixels on display */
    MOVHI   r8, %hi(VRAM_BASE) /* start at base of VRAM */
    ORI     r8, r8, %lo(VRAM_BASE)
    MOVHI   r13, %hi(PIXEL_TRACE) /* load colors that will be cleared */
    ORI     r13, r13, %lo(PIXEL_TRACE)
    MOVHI   r14, %hi(PIXEL_CURSOR)/* which are trace and cursor */
    ORI     r14, r14, %lo(PIXEL_CURSOR)
    MOVI    r9, SIZE_X          /* will loop through all columns */
    MOVI    r10, SIZE_Y         /*  and all rows */
    MOV     r11, r0             /* starting at (0, 0) */
    MOV     r12, r0             /* (top left corner) */

trace_check:                    /* check if current pixel is part of trace */
    LDWIO   r15, (r8)           /* read value from VRAM */
    BEQ     r13, r15, trace_clear /* definitely clear if color is trace color */
```

```

cursor_check:                   /* check if current pixel is part of cursor */
    BNE     r14, r15, trace_row_loop /* also clear if part of cursor */

trace_clear:                    /* pixel is part of trace or cursor */
    STWIO   r0, (r8)            /*  so clear it */

trace_row_loop:                 /* done with current pixel */
    ADDI    r8, r8, WORD_SIZE  /*  so go to next */
    ADDI    r11, r11, 1         /*  and also increment column index */
    BLT     r11, r9, trace_check /* if still within display, repeat */

trace_next_row:                 /* done with current row */
    ADDI    r8, r8, REMAINDER  /* add remainder to finish up VRAM row */
    MOV     r11, r0             /* reset column index */
    ADDI    r12, r12, 1         /*  and increment row index */
    BLT     r12, r10, trace_check /* if still within display, repeat */

    RET                         /* all done, so return */


/*
 *  plot_pixel
 *
 *  Description:       This procedure changes the color to the pixel at the passed x, y
 *                     coordinates, where the top left corner is (0, 0), to the passed
 *                     color. Colors are specified with a 24-bit value, where the bottom
 *                     8 bits represent the amount of blue, the following 8 the amount
 *                     of green, and the next 8 the amount of red.
 *
 *  Operation:         The function simply translates the x and y coordinates into a VRAM
 *                     address by setting the top bits to the offset of the VRAM, and ORing
 *                     in the shifted row and column indeces. Then, it stores the passwed
 *                     color value at that address.
 *
 *  Arguments:         x - x coordinate of the pixel, where leftmost column is 0 (r4).
 *                     y - y coordinate of the pixel, where top row is 0 (r5).
 *                     color - 24-bit value with RGB color the pixel should change to (r6).
 *
 *  Return Value:      None.
 *
 *  Local Variables:   None.
 *
 *  Shared Variables:  None.
 *
 *  Global Variables:  None.
 *
 *  Input:             None.
 *
 *  Output:            Changes the color of one pixel on the display.
 *
 *  Error Handling:    None.
 *
 *  Limitations:       None.
 *
 *  Algorithms:        None.
 *  Data Structures:   None.
 *
 *  Registers Changed: r8, r9, r10.
 *
 *  Revision History:
 *      6/03/14   Santiago Navonne     Initial revision.
 *
 */
    .global plot_pixel
plot_pixel:                     /* draw a pixel of the specified color */
    MOVHI   r8, %hi(VRAM_BASE) /* find pixel location by first going to VRAM base */
    ORI     r8, r8, %lo(VRAM_BASE)
    MOVI    r9, ROW_ADDR_SHIFT /* shift the row to the row part of the address */
    SLL     r9, r5, r9
    MOVI    r10, COL_ADDR_SHIFT/* and the column to the column part */
    SLL     r10, r4, r10
    OR      r8, r8, r9          /* OR row, column, and VRAM base together */
    OR      r8, r8, r10         /*  to create final pixel address */
    STWIO   r6, (r8)            /* and finally save passed color value to that address */

    RET                        /* all done, so return */

/*
 *  pixel_color
 *
 *  Description:       This procedure returns the color of the pixel at the passed x, y
 *                     coordinates, where the top left corner is (0, 0). Colors are
 *                     specified with a 24-bit RGB value, where the bottom 8 bits
 *                     represent the amount of blue, the following 8 the amount of green,
 *                     and the next 8 the amount of red.
 *
 *  Operation:         The function simply translates the x and y coordinates into a VRAM
 *                     address by setting the top bits to the offset of the VRAM, and ORing
 *                     in the shifted row and column indeces. Then, it loads the color word
 *                     from VRAM and returns it in r2.
 *
 *  Arguments:         x - x coordinate of the pixel, where leftmost column is 0 (r4).
 *                     y - y coordinate of the pixel, where top row is 0 (r5).
 *
 *  Return Value:      color - 24-bit value with RGB color of requested pixel, or NO_TRACE
 *                             if no trace was found at the requested coordinate(r2).
 *
 *  Local Variables:   None.
 *
 *  Shared Variables:  None.
 *
 *  Global Variables:  None.
 *
 *  Input:             None.
 *
 *  Output:            None.
 *
 *  Error Handling:    None.
 *
 *  Limitations:       None.
 *
 *  Algorithms:        None.
 *  Data Structures:   None.
 *
 *  Registers Changed: r8, r9, r10, r2.
 *
 *  Revision History:
 *      6/03/14   Santiago Navonne     Initial revision.
 *
 */
    .global pixel_color
pixel_color:                   /* read a pixel from display */
    MOVHI   r8, %hi(VRAM_BASE) /* find pixel location by first going to VRAM base */
    ORI     r8, r8, %lo(VRAM_BASE)
    MOVI    r9, ROW_ADDR_SHIFT /* shift the row to the row part of the address */
    SLL     r9, r5, r9
    MOVI    r10, COL_ADDR_SHIFT/* and the column to the column part */
    SLL     r10, r4, r10
    OR      r8, r8, r9         /* OR row, column, and VRAM base together */
    OR      r8, r8, r10        /*  to create final pixel address */
    LDWIO   r2, (r8)           /* and finally read color value from that address */

    RET                        /* storing it in return register */

```