

CS/COE 447 Computer Organization

Fall 2009

Programming Project #1

Assigned: September 21. Due: October 5 by 11:59PM.

Project Description

Each year, on New Year's Eve, a big shiny ball slowly drops in Times Square to usher in the New Year. A countdown timer keeps track of how much time remains until the New Year arrives; of course, everybody counts down the last few seconds together in joyous celebration!

In this assignment, you will implement the countdown timer for the New Year's Eve ball. The timer is programmable — an user can enter an initial time remaining. The timer is represented by “M:SS”, where M is the minute and SS is the seconds remaining. For example, suppose an user enters “1:59”. This means there is 1 minute and 59 seconds until the New Year. The timer counts down one second at a time. The timer's current value is shown and updated on a display. When the timer expires (i.e., it reaches 0:00), the display flashes to indicate the New Year has arrived.

You will use a simulated display in Mars. Every second, your program will update the display with the timer's current value. It will show the minutes and seconds remaining. The display is a 8x128 grid of light emitting diodes (LEDs). It has 1,024 LEDs. To show the timer value, your program will turn on/off individual LEDs. The simulated display is “memory mapped”: the contents of certain memory addresses control the operation of the display. Each LED is associated with a single bit in memory. To turn on the LED, the bit is set to 1 and to turn off the LED, the bit is reset to 0. In this way, your program will store values to memory to display the timer.

Here is an example display with the timer value “1:23”:

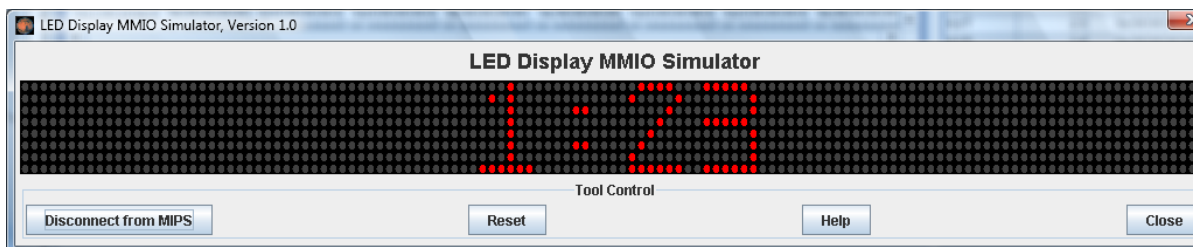


Table 1: Example LED Display

Once the timer expires and the display has been flashed, the user is prompted to again enter a new timer value. Your program should exit when the user enters “.” (a period).

Project Requirements

The project has several requirements:

- The user is prompted to enter an initial timer value as a string in the format “M:SS”, where M is the minute (0-9) and SS is the seconds (00-59). A leading zero is needed when the minute is 0, or the seconds are 0-9. The colon is entered. E.g., the user might enter “0:09”, “1:59”, etc.
- You can assume that the user input is correct. You don’t have to display prompts; Mars will pop up a dialog box to handle the string syscall.
- The timer’s value is shown on the display as M:SS. The current minute (M) is shown followed by a colon, which is followed by the two-digit second (SS). The display is centered. A leading 0 is shown when the second is in the range 0-9 or the minute is 0 (e.g., “0:09”). The display should be similar to Figure 1.
- The timer is updated every second; the LED display is also updated with the time remaining.
- When the timer reaches 0:00, the display is quickly flashed 16 times.
- After the display is flashed on timer expiration, the user is prompted to enter a new time.
- The program terminates when the user enters “.”.
- The timer should keep reasonably accurate time, but it doesn’t have to be exact.
- You must design a good font to display the characters ‘0’ to ‘9’ and ‘:’ on the LED display.

The program must run with the Mars simulator and be implemented as a MIPS assembly language program. The programming project is an individual effort. You may ask other students how they are approaching the problem, but you must not work together on the coding.

LED Display Simulator

The project requires a special version of Mars available from the CS/COE 447 web site. This version of Mars includes an LED display simulator developed by a graduate student in the CS Department. There are two demo programs that illustrate the LED display. For off-campus access to these files, you’ll need the user name and password announced in class.

To run the display simulator, click on the Tools menu tab in Mars. Select the menu option LED Display Simulator. When the display opens, select Connect to MIPS. This action establishes communication between the MIPS simulator and the LED display simulator. Load your program and run it (try the `leddemo.asm` program). The display should change.

To end the LED Display Simulator, Disconnect and Close it. You can clear the LED display with the Reset button. You may need to Reset and Connect whenever you load or assemble a program.

If you have problems (bugs!) with the LED Display Simulator, let Dr. Childers and/or Santiago know. This simulator was written for this project. You might encounter a bug, but it is well tested.

Technical Specification of the LED Display Simulator

The LED Display Simulator models a display with 8 rows and 128 columns of LEDs. Each LED is associated with a bit in main memory. To turn on an LED, its bit is set to 1 and to turn off the

LED, its bit is set to 0. There are 1,024 LEDs, so 1,024 bits (128 bytes) in memory are needed. The values stored in the address range [0xFFFF0000, 0xFFFF007F] are used for the display.

The bytes in the display's address range are mapped to specific LEDs. The display rows are numbered 0 to 7 (top to bottom) and the columns are 0 to 127 (left to right). The upper left corner is coordinate (row 0, column 0) and the lower right corner is coordinate (7, 127).

The upper left corner of the display is mapped to the lowest address (0xFFFF0000) and the lower right corner is mapped to the highest address (0xFFFF007F). Bit 7 in the byte at 0xFFFF0000 controls the LED at (0,0). Bit 6 in the byte at 0xFFFF0000 controls the LED at (0,1). Similarly, bits 5 to 0 in the byte at 0xFFFF0000 control the LEDs at (0,2), (0,3), (0,4), (0,5), (0,6) and (0,7). The byte at 0xFFFF0001 control the next eight LEDs in the row, and so forth. The table below gives the mapping between LED coordinates and addresses. The table shows word addresses.

| | Columns 0-31 | Columns 32-63 | Columns 64-95 | Columns 96-127 |
|-------|--------------|---------------|---------------|----------------|
| Row 0 | 0xFFFF0000 | 0xFFFF0004 | 0xFFFF0008 | 0xFFFF000C |
| Row 1 | 0xFFFF0010 | 0xFFFF0014 | 0xFFFF0018 | 0xFFFF001C |
| Row 2 | 0xFFFF0020 | 0xFFFF0024 | 0xFFFF0028 | 0xFFFF002C |
| Row 3 | 0xFFFF0030 | 0xFFFF0034 | 0xFFFF0038 | 0xFFFF003C |
| Row 4 | 0xFFFF0040 | 0xFFFF0044 | 0xFFFF0048 | 0xFFFF004C |
| Row 5 | 0xFFFF0050 | 0xFFFF0054 | 0xFFFF0058 | 0xFFFF005C |
| Row 6 | 0xFFFF0060 | 0xFFFF0064 | 0xFFFF0068 | 0xFFFF006C |
| Row 7 | 0xFFFF0070 | 0xFFFF0074 | 0xFFFF0078 | 0xFFFF007C |

The *least significant byte* of a word controls the LEDs at the low coordinates for the word. For example, the *least significant byte* of the word at 0xFFFF0024 controls row 2, columns 39 to 32. Within a byte, the *most significant bit* of the byte (bit 7) controls the low coordinate for the byte. For example, bit 7 in the byte at address 0xFFFF0024 is LED position (2, 32) and bit 0 is position (2, 39). Below is an example: the word values at the specified addresses draw a right arrow.

| <u>Address</u> | <u>Binary value</u> |
|----------------|---------------------|
| 0xFFFF0004 | 0x08000000 |
| 0xFFFF0014 | 0x04000000 |
| 0xFFFF0024 | 0x02000000 |
| 0xFFFF0034 | 0xFFFFFFFF |
| 0xFFFF0044 | 0xFFFFFFFF |
| 0xFFFF0054 | 0x02000000 |
| 0xFFFF0064 | 0x04000000 |
| 0xFFFF0074 | 0x08000000 |

You may want to write a simple program that stores these values to the given addresses.

Turning in the Project

You must submit a compressed file (.zip or .tar.gz) containing:

- timer.asm (the timer program)
- README.txt (a help file - see next paragraph)

Put your name and e-mail address in both files at the top. Use the README.txt file to explain the algorithm you implemented for your programming assignment. If you have known issues (e.g., bugs, etc.) with your code, you should specify those clearly in this file.

The filename of your submission (the compressed file) should have the format:

```
<your username>-pa01.zip (or .tar.gz)
```

For example:

```
sabl04-pa01.zip
```

Files submitted after October 5 at 11:59 PM will not be graded. **It is strongly suggested that you submit your file well before the deadline.** That is, if you have a problem during submission, we can not guarantee to respond to the last minute problem before the deadline.

Where to Submit

To submit your compressed file, you have to use anonymous FTP to cs.pitt.edu and go to 'incoming/CS0447-bock/pa-01/' directory and put your file inside that directory. To use anonymous FTP, you can use any software that can do FTP (e.g., the PuTTY utilities) or use your favorite file explorer and type ftp://cs.pitt.edu in the address-bar. If it asks for username and password, just use anonymous as the username and leave the password field blank. Alternatively, you can type ftp://anonymous@cs.pitt.edu in the address bar.

NOTE: you will **not** be able make any kind of modification (rename, delete, copy and so on) to your file once it is submitted. If you want to make changes, you have to resubmit your file with a version number appended to the filename (but it is not recommended, please submit your final version). If you get problems in submitting your file there, let Santiago know by sending an email to sabl04@cs.pitt.edu.

NOTE: Your assembly language code must be properly documented and formatted. Use enough comments to explain your algorithm, implementation decisions and anything else necessary to make your code easily understandable.

Programming Hints

Write pseudocode for the program prior to thinking about the assembly language. *Plan your program carefully! Don't try to write any MIPS code until you are happy with your pseudocode!*

I recommend that you separate the program into four tasks: 1) get a timer value from the user; 2) show the timer value on the LED display; 3) decrement the timer value and check for timer expiration; and 4) flash the LED display. You can develop and test these tasks one at a time.

At the very least, I strongly recommend that you write the LED display code separately. Design your font and figure out how to update the display. Once you understand the display's operation and you have a good font, then work on the timer code.

You may want to use three values for the timer. The three values correspond to the minute, the "high second" and the "low second". You can decrement the individual values to update the timer. If the low second becomes negative, it is reset to 9 and the high second is decremented. When the high second becomes negative, it is reset to 5 and the minute is decremented.

You'll need to design a font for the characters '0' to '9' and ':'. The font is a set of bit patterns that make the LEDs turn on to look like a character. If you make all characters 8x8 (8 rows by 8 columns), it will greatly simplify your program and make it easy to center the display.

You need a way to represent and access the font in your program. You need to map a number in the range 0 to 9 to the characters '0' to '9' in the font. I suggest a "lookup table". The table is indexed by the number and a table entry holds the bit pattern for the character associated with the number. For an 8x8 font, each table entry can be 2 words. For a fast lookup, you can convert the number to a byte offset in the table. The conversion is done by scaling the number by the table entry size. For example, suppose you want the LED character (bit pattern) for the number 2. This character is located at byte offset $2 (\text{index}) * 8 (\text{table entry size}) = 16$. This offset is added to the table's base address in memory to read the bit pattern.

To flash the display, you can alternate between showing the complement and non-complement of the final timer value (this can be done very simply by reading/modifying the contents in the LED display memory!). You'll need to delay for a short time period between each flash to get the desired effect.

You can use the Mars sleep syscall (32) to make your program delay between each update of the timer. After you update the LED display, sleep for around 800-900msec. When the program resumes, you can decrement the counter and continue. Because your program takes some time and Mars also introduces extra time, you need to sleep for less than a second to keep reasonably accurate time. You may want to tune the sleep time with a stop watch. ;-)

Similarly, to flash the display, sleep for around 20-40msec between each flash.