Name: Nick Petro **CoE 0147**

Lab 4, Spring 2015

Lab 4: LED Simulator

Important:

- While collaborations are allowed, names of all the collaborators must be listed in the assignment. Each student should submit individual assignments for grading, despite collaboration.
- Solution codes (***.asm files**) for this assignment should be submitted to Courseweb/Course Documents/Recitation/Lab4 before the next recitation; the subject line should carry "COE0147 Lab-4". Please zip the files together.
- Each program of the assignment must be in a separate file. You must write your name in the first line of every .asm file in the following format.
  #Name: First_name Last_name
  #rest of the file...

For this lab, we require a modified version of Mars, which is available at Courseweb:
Courseweb/Course Documents/Recitation/Lab4/Mars-4.1-With-Keypad-LED128x8.jar

A note about how the LED simulator works: You have to enable the LED simulator by selecting "Keypad and LED Simulator" from the **tools** menu. Once it has been enabled, click "connect to MIPS". You can draw to (read from) the LED display by writing to (reading from) its memory, which begins at address 0xFFFF0008.

The first byte (8 bits) of LED memory corresponds to the first 4 dots in the display (2 bits per dot). The 7th and 6th bits of the byte control the left most dot (of the first row), the 5th and 4th bits control the second dot from the left, and so on. Since there are two bits per dot, each dot can be set in only 4 colors (including the 'off' color).

1 drawPattern()

Write a function *drawPattern (int* address, int bitPattern)* that stores a word-sized *bitPattern* in the memory location pointed to by the *address*. You may assume that the *address* is word-aligned. In the function definition, *int* is the size of a word and *int** is a pointer to a word, i.e., address of a word. Use the following code to test your function:

```
.text
li $a0, 0xFFFF0008 #LED memory starts at this address
li $a1, 0x7EF965BD #LEDs to turn on
jal drawPattern #Jump and link to setLED
li $v0, 10 #Exit
syscall
```

If your *drawPattern()* function is correct, the above test code should light up the first 16 LEDs (starting from the top left) of the LED display. The output pattern for this part is shown in Fig. 1.
Note: Submission not required for this part.

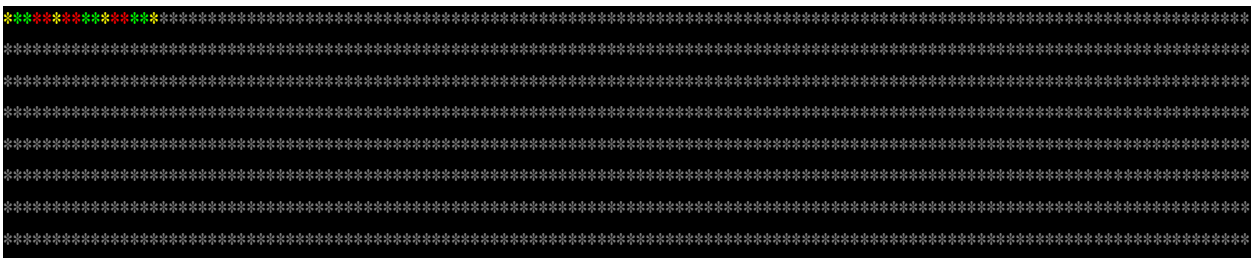

Figure 1: Output Pattern for section 1

2 getPattern()

Write a function *int getPattern (int* address)* that returns the word-sized bit pattern currently stored in the memory location pointed to by the *address*. You may consider using the code below to test your function:

```
.data
ok:  .asciiz "The LED pattern matches."
not_ok:  .asciiz "The LED pattern doesn't match!"
.text
li $a0, 0xFFFF0008 #LED memory starts at this address
li $a1, 0x7EF965BD #LEDs to turn on
jal drawPattern #Jump and link to drawPattern
jal getPattern #Jump and link to getPattern
bne $a1, $v0, else #Return values should be in $v0
la $a0, ok #Load ok string if equal
j end
else:  la $a0, not_ok #Load not_ok string if not equal
end:  li $v0, 4 #Print the string
syscall
li $v0, 10 #Exit
syscall
```

Note: Submission not required for this part.

3 disruptPattern()

Write a function *void disruptPattern (int* address)* that reads the word-sized bit pattern stored in the memory location pointed to by the *address*, `XORs` that word with the value `0xC31601C9`, and stores it back at the same location in the LED memory. You may assume that this address is word-aligned.

The above steps will alter the LED lights at that address. Your function must use the functions defined in the previous two parts, i.e., your *disruptPattern()* function must make use of your *getPattern()* and *drawPattern()* functions. The output for this part is shown in Fig. 2. For this part of the lab, be sure to use the template available at: http://www.pitt.edu/~pmp30/CoE147/Lab/Lab4addl/lab4part3.asm
Submit *lab4part1.asm* for this section.

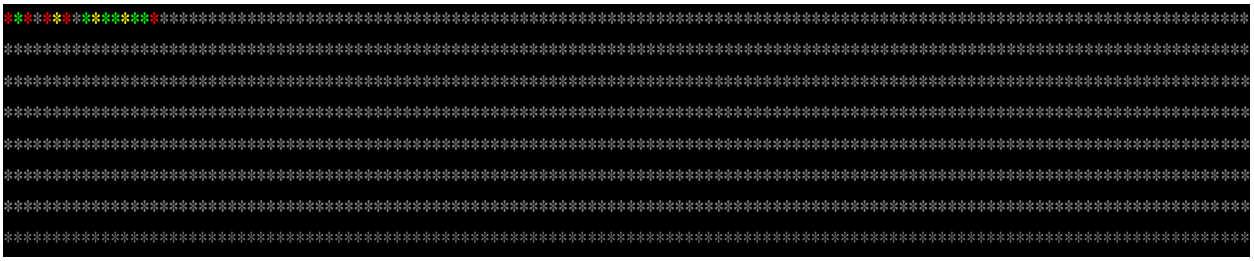

Figure 2: Output Pattern for section 3

4 drawShape()

Write a function *drawShape* to draw the pattern shown in Fig. 3 on the LED display. Your *drawShape()* function must use your *drawPattern()* function in a loop to generate the desired pattern. For each iteration of the loop, turn on (light up) a section of LED memory; the section that you turn on will be immediately below the previous section. Note that each row consists of 128 LEDs, and each LED requires 2 bits, therefore each row is 256 bits wide (8 words).
Submit *lab4part2.asm* for this section.

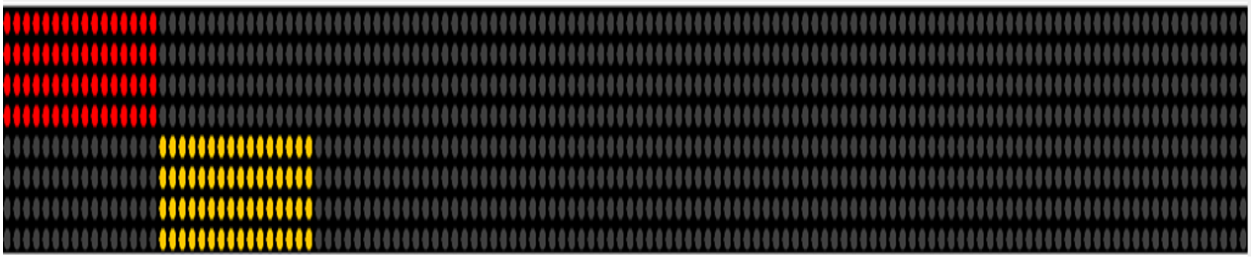

Figure 3: Output Pattern for section 4

5 Practice Problems

1. (Q2.31) Implement the following C code in MIPS assembly. What is the total number of MIPS instructions needed to execute the function?

```
int fib(int n) {
  if(n==0)
    returen 0;
  else if (n==1)
    return 1;
  else
    return fib(n-1) + fib(n-2);
}
```

2. (Q2.34) Translate function `f` into MIPS assembly language. If you need to use registers `$t0` through `$t7`, use the lower-numbered registers first. Assume the function declaration for `func` is "`int f(init a , init b);`". The code for function `f` is as follows:

```
int f(int a , init b , init c , init d) {
  return func(func(a , b) , c + d)
}
```

3. (Q2.38) Consider the following code:

```
lbu $t0, 0($t1)
sw  $t0, 0($t2)
```

Assume that the register `$t1` contains the address `0x1000 0000` and the register `$t2` contains the address `0x1000 0010`. Note the MIPS architecture utilizes big-endian addressing. Assume that the data (in hexadecimal) at address `0x1000 0000` is: `0x11223344`. What value is stored at the address pointed to by register `$t2`?

0x00000044 is stored in the address pointed to by $t2