CSEE 3827: Homework 4 Due: 4/11/11

- For this assignment you will write MIPS code, which you can test by simulating it with SPIM, a MIPS simulator. SPIM is a command line tool available on cunix (cunix.cc.columbia.edu).
- We have provided templates which you must use for your solutions to these problems. You can download the templates as follows:
  - Download the templates from http://www.cs.columbia.edu/ martha/courses/3827/sp11/homeworks/templates.tar
  - Unpack the templates: `% tar xvfz templates.tar.gz`
  - The preceding step will create a directory named `templates` which you should rename to your own UNI, e.g., `% mv templates mak2191`.
- To turn in this homework, you will recreate a tarball with your solutions. When we unpack this tarball we should find a directory named with your UNI containing two files: `min.s` and `histogram.s`.
  - Make sure the directory containing your solutions is named after your UNI.
  - Make sure your solutions use the templates and preserve the original filenames.
  - Go to the directory above the one containing your solutions and create a tarball, e.g., `% tar cvfz mak2191.tar.gz mak2191/`
- For help using spim on cunix, run 'spim' with no arguments, followed by the command 'help'

1. **min.s** The purpose of this problem is to have you write a simple program and get you accustomed to using spim. You should complete it before moving on to the other portions of the assignment.

```
####################################################################
##
##                PROBLEM 1
##
## Write a function that takes two integer arguments and
## and returns the minimum of the two.  If it were a C function
## the declaration would be:
##
##    int min(int a, int b)
##
## Your implementation must adhere to all MIPS calling conventions,
## and use exactly this template.  Modify the file only where indicated.
##
#####################################################################

              .data
newline:      .asciiz "\n"

              .text
              .globl main

main:         addi  $a0, $zero, 10
              addi  $a1, $zero, 20
              jal   min
              move  $a0, $v0
              jal   print           # should print 10

              addi  $a0, $zero, 10
              addi  $a1, $zero, -10
              jal   min
              move  $a0, $v0
              jal   print            # should print -10

              li    $v0, 10          # exit
              syscall
```

```
min:
##
## Your implementation of min() here.
##

print:          li    $v0, 1
                syscall
                li    $v0, 4
                la    $a0, newline
                syscall
                jr    $ra
```

2. **histogram.s** The last three problems culminate in the construction of a histogram printer. The file contains the full specifications for each component.

```
####################################################################
##
##                  Problems 2-4
##
## This series of problems culminates in the implementation of a
## histogram function which counts the number of occurrences of each
## character in a string, and prints the resulting histogram to the
## screen. For example, the histogram for the string "Hello, World!"
## is:
##
##     |
##   ! |
##   , |
##   H |
##   W |
##   d |
##   e |
##   l |||
##   o ||
##   r |
##
## It is HIGHLY RECOMMENDED that you implement these functions in
## problem order because the latter ones will depend on a working
## implementation of the earlier ones.
##
## NB: At the bottom of this file several helper functions
##     are provided for your use.
##
## NB: Each of your functions must adhere to MIPS calling conventions,
##     and take and return the arguments exactly as specified
##     (including their order).
##
## We have provided code in the body of main to test each function
## individually.  Simply uncomment the tests that you would like to
## run.
##
####################################################################

                .Data
min_char:       .asciiz " "
max_char:       .asciiz "~"
bar:            .asciiz "|"
separator:      .asciiz " "
newline:        .asciiz "\n"
hello:          .asciiz "Hello, World!"
gettysburg:     .asciiz "Four score and seven years ago our fathers brought forth on this continent, a

                .text
                .globl main
main:
###################################################################
##                   PROBLEM 2 TEST CODE (print_bucket)
```

```
####################################################################

##            li    $a0, 42       # $a0 = '*'
##            li    $a1, 10       # $a1 = 10
##            jal   print_bucket  # should print: "* ||||||||||" on its own line

##            li    $a0, 45       # $a0 = '-'
##            li    $a1, 12       # $a1 = 12
##            jal   print_bucket  # should print: "- ||||||||||||" on its own line

##            li    $a0, 80       # $a0 = 'P'
##            li    $a1, 8        # $a1 = 8
##            jal   print_bucket  # should print: "P ||||||||" on its own line

####################################################################
##                 PROBLEM 3 TEST CODE (count)
####################################################################

##            li    $a0, 108        # $a0 = 'l'
##            la    $a1, hello
##            jal   count
##            move $a0, $v0
##            jal   print_int       # should print "3"

##            li    $a0, 33         # $a0 = '!'
##            la    $a1, hello
##            jal   count
##            move $a0, $v0
##            jal   print_int       # should print "1"

##            li    $a0, 98         # $a0 = 'b'
##            la    $a1, gettysburg
##            jal   count
##            move $a0, $v0
##            jal   print_int       # should print "13"

####################################################################
##                 PROBLEM 2 TEST CODE (histogram)
####################################################################


##            la    $a0, hello
##            jal   histogram

##            la    $a0, gettysburg
##            jal   histogram

              li    $v0, 10          # exit
              syscall

#######################################################################
##                         PROBLEM 4
##
## Implement the histogram function:
##              void histogram(char* s)
##
## For each character c between *min_char and *max_char, histogram
## counts the number of occurrences of c in s, and prints out a bucket
## in the histogram for all non-zero counts.
##
#######################################################################

histogram:
##
##       Your code here
##

#######################################################################
```

```
##                         PROBLEM 3
##
## Implement the count function:
##            int count(char c, char* s)
##
## count counts the number of instances of a character (c) in a string
## (s).
##
#######################################################################

count:
##
##      Your code here
##

######################################################################
##                         PROBLEM 2
##
## Implement the print_bucket function:
##            void print_bucket(char c, int n)
##
## print_bucket takes two arguments, a character (c) and an
## integer (n) indicating how many instances of c were found in
## some string.  This function should print the corresponding bucket
## in the histogram.
##
## A bucket consists of a label, followed by n bars, followed by a
## newline.  For example, print_bucket('a', 2) should print "a ||\n".
##
#######################################################################

print_bucket:
##
##      Your code here
##

####################################################################
##                 HELPER FUNCTIONS                              ##
####################################################################

print_label:                          # void print_label(char c)
              addi  $sp, $sp, -4
              sw    $ra, 0($sp)
              jal   print_char
              jal   print_separator
              lw    $ra, 0($sp)
              addi  $sp, $sp, 4
              jr    $ra

print_bar:                            # void print_bar()
              la    $a0, bar
              li    $v0, 4
              syscall
              jr    $ra

print_separator:                      # void print_separator()
              la    $a0, separator
              li    $v0, 4
              syscall
              jr    $ra

print_newline:                        # void print_newline()
              la    $a0, newline
              li    $v0, 4
              syscall
              jr    $ra

print_char:   li    $v0, 11           # void print_char(char c)
```

```
              syscall
              jr    $ra


print_int:    li    $v0, 1             # void print_int(int i)
              syscall
              li    $v0, 4
              la    $a0, newline
              syscall
              jr    $ra
```