CS3340 - Assignment 4: MergeSort and Assembly Decoding (6%+4%Bonus)

Due 11:59pm, Sunday, 29 March 2020

- **Mergesort (6%):**

Convert "merge" in Assignment 3 into a subroutine. Write a "main" program to perform merge-sorting of a list of integers by calling "merge" repeatedly. For example, if the sorting program takes (6,5,9,1,7,0,-3,2) as input, it will produce a sorted list (-3,0,1,2,4,6,7,9).

The original unsorted list of integers should be received from the keyboard input. Your program should first ask for the user to input the number of integers in the original list, and then ask for inputting all the integers. The total number of integers to be sorted by this program should be a power of 2. This means, the program should work with a list of 2, 4, 8, 16, or 32 (...) integers (but your program needs only to handle up to 32 integers).

The final sorted list (in **increasing** order) should be stored in the data area, that starts with the label "list:". The sorted list should **also** be displayed on the screen (console).

You should **not** do this part by implementing a different sorting algorithm (e.g. quick sort). You will receive 0% if you do not make use of "merge", or if your sorted list is in decreasing order.

[HINTS]: The bottom-up approach of implementation without using recursion is to initially sort the smallest possible sub-lists, and then merge every two neighboring sub-lists, until the whole list is merged and sorted. This non-recursive approach is more efficient in general, and is thus required for this assignment. [An alternative is to sort the list by dividing it into two sub-lists, recursively sorting the sub-lists, and then joining the sub-lists together to give the sorted list. But this recursive approach is not required in this assignment.]

- **Speed competition (Bonus 2%)**

The top FOUR fastest submissions will be awarded with 0.5% - 2% bonus. The fastest gets 2%, the 2nd fastest gets 1.5%, 3rd gets 1%, and 4th gets 0.5%. You should use the new QtSPIM version with clock cycle count. The only measure for the speed is the cycle count.

- **Machine code -> MIPS (Bonus 2%)**

During 2019 Dallas Hackathon, your company's server computer was crashed by a hacker (fabricated story☺). The hacker has left behind a paper with the following piece of information:

```
            .data
            .globl  count
count:      .word   8
            .globl  list
list:       .word   -2,3,12,0,10,-3,9,5
            .globl  str
str:        .asciiz "The result is:"
            .globl  space
space:      .asciiz " "
            .globl  newln
newln:      .asciiz "\n"
            .align  2

            .text
            .globl main
main:
[0x00400020]        0x3c011001
[0x00400024]        0x342b0004
[0x00400028]        0x3c011001
[0x0040002c]        0x8c2e0000
[0x00400030]        0x000e6080
[0x00400034]        0x000ca021
[0x00400038]        0x000e6843
[0x0040003c]        0x018b6020
[0x00400040]        0x000d6880
[0x00400044]        0x01ab6820
[0x00400048]        0x218cfffc
[0x0040004c]        0x8d6f0000
[0x00400050]        0x8d900000
[0x00400054]        0xad700000
[0x00400058]        0xad8f0000
[0x0040005c]        0x216b0004
[0x00400060]        0x016d8822
[0x00400064]        0x1620fff9
[0x00400068]        0x34020004
[0x0040006c]        0x3c011001
[0x00400070]        0x34240024
            syscall
            li      $15, 0
print:      li      $v0, 4
            la      $a0, space
            syscall
            lw      $a0, list($15)
            li      $v0, 1
            syscall
            addi    $15, $15, 4
            slt     $8, $15, $20
            bnez    $8, print
            li      $v0, 4
            la      $a0, newln
            syscall
            li      $v0, 10
            syscall
```

As an acknowledged expert of MIPS assembly, you realize that this is the critical part of the server software with addresses (labels) on the left and machine code on the right. You now try to

save the server by translating the machine code to make a complete and sensible MIPS program. (Your tasks: write down the sequence of equivalent MIPS assembly instructions, and state what will be printed on the console window if the complete program is executed).

- **SUBMISSION:**

You should store your mergesort program in one SINGLE file. At the top of the program, use comments (starting with "#") to write your name and also comment how your programs are implemented. Your submitted program must be in plain text format and must be runnable on QtSPIM. Put your answers to the "**Machine code -> MIPS**" part AND YOUR NAME in a WORD or plain text document (call it "Bonus.doc", or "Bonus.txt"), and compress it with your mergesort program before submitting to eLearning.

In general, you need to document clearly which instruction does what in terms of the problem solving (instead of saying "#add 4 to $s1 and send back" for "addi $s1, $s1, 4"), and which register stores what data (e.g. list index, counter, etc). Write your general comments on top of your program and specific comments after individual instructions.

There will be no extension of the deadline and late submissions will be penalized (1% deducted for each day delay and no submissions accepted after 3-day delay, this % is the SAME as % in 6%, e.g. after one-day delay, the max you could get is 5%).

Kang Zhang