Computer Science Courses
ANU College of Engineering & Computer Science

Assignment 2 Specification - rPeANUt - Drawing Program

The aim of this assignment is to give you a challenging assembly program to write. Unlike the first assignment, this assignment will be completely marked by computer. Thus the focus is on correctness and performance. Coding style and coding formatting will not be marked (not to say these are not important as a clear consisted approach in your formatting and coding style will help with the correctness of your submission).

The basic part of the assignment aims to be simple enough for most students to be able to complete in less then 5 hours. The assignment also aims to provided a more challenging programming task for the student who wish to gain extra experience (and a higher mark).

The assignment will be marked out of 100. And is worth 10% of the final mark. A late penalty cap of 10% per day applies (these are working days). Also note this penalty cap is limited so it does not take your mark below 50%. Suppose the assignment was due on a Monday if you submit it any time before midnight on the following Tuesday then your mark for this assessment will be capped at 90/100. If it is handed in on the Wednesday then the maximum mark you may gain is 80/100, by Monday of the following week (and any time after) the maximum would be 50/100. In terms of a formula let 'r' denote your raw mark for the assignment (out of 100) and 'd' is the number of working days late then your mark for the assignment will be: min(max(50,100-10*d),r) . This assignment will not be accepted after the last day of semester. Extensions are possible in documented exceptional cases (e.g. medical certificate).

Drawing Program

The drawing program takes input commands from the terminal and uses these to draw into the frame buffer. The program halts when the 'h' command is issued. The program must not do any output to the terminal.

Commands are given by a single character followed by optional parameters. You may assume the input provided to the program has no formatting problems. The parameters of the commands are given as 2 digit hex numbers (using lower case). You may assume the frame buffer starts off empty (a fresh reset and load has taken place when your program starts). When the program finishes a dump of the frame buffer is obtained and compared for correctness.

The commands are as follows:

- h - this halts the machine.
- f - this fills the entire screen with white.
- c - this clears the entire screen (fills with black).
- p<X><Y> - this draws a single white pixel at coordinate (<X>,<Y>). You may assume that the coordinate is alway within the screen dimensions. Also the top left pixel has coordinate (0,0). (e.g command "p010b" will draw a single white pixel at coordinate (1,11))
- r<X><Y><W><H> - this draws and fills a white rectangle which has its top left hand pixel at coordinate (<X>,<Y>) and is <W> pixels wide and <H> pixels high. You may assume that the parameters are such that the rectangle drawn is within the bounds of the screen. (e.g. The command "r00000202" would draw a square that has a side length of two pixels and is in the top left position on the screen.)
- l<X1><Y1><X2><Y2> - this draws a white line from the pixel at coordinate (<X1>,<Y1>) to the

pixel at coordinate (<X2>,<Y2>). You may assume that both coordinates are within the screen. For this line drawing you must use the follow algorithm (from Wiki Bresenhams line algorithm page): "

```
function line(x0, y0, x1, y1)
   dx := abs(x1-x0)
   dy := abs(y1-y0)
   if x0 < x1 then sx := 1 else sx := -1
   if y0 < y1 then sy := 1 else sy := -1
   err := dx-dy

   loop
     setPixel(x0,y0)
     if x0 = x1 and y0 = y1 exit loop
     e2 := 2*err
     if e2 > -dy then
       err := err - dy
       x0 := x0 + sx
     end if
     if e2 <  dx then
       err := err + dx
       y0 := y0 + sy
     end if
   end loop
```

" Note, you may apply other optimizations, however, any changes must be pixel identical to the above algorithm.

Your assignment must be implemented within a SINGLE assembler "mysolution.s" file. So do not use the #include statement.

Marking

The marking of this assignment will be automatic. So it is important that your solution compiles and works. Also, get the assignment working in stages. Different tests will be performed such that if you only complete some of the commands you will still gain marks for the command you were able to get going. If you can't get the harder parts of the assignment working just leave them out of your submission and get the marks for the earlier parts. The marking stages of the assignment are as follows:

1. Your program compiles. [10 marks] (Note - submit an empty file and you will get 10 marks for the assignment!)
2. Halt works. [10 marks]
3. Fill Screen works. [20 marks]
4. Clear and fill screen works. [10 marks]
5. Drawing points work. [10 marks]
6. Drawing rectangles work. [10 marks]
7. Drawing lines work. [10 marks]
8. Performance. [20 marks] This requires all of the other stages working correctly. Basically your submission will be benchmarked (in terms of the number of steps it takes to run) against my solution for a particular input. The final test in the tests below give you an idea of the types and amounts of commands issued for this performance test. However, the actual performance test will not be identical.

Stages 3+ also require that multiple commands and halt work.

Tests

The tests will be available from: http://cs.anu.edu.au/courses/COMP2300/ass2tests. These give you an idea of the types of tests performed. Your program will be tested using the latest version of rPeANUt.

To run a test you need to check that when your program is given a set of commands it produces identical output. This can be done using the following terminal command:

```
java -jar rPeANUtxxx.jar -dump mysolution.s < testX.commands | diff - testX.dump
```

There should be no difference between your output and what is expected.

Individual Assignment

This is an individual assignment. You may help each other, however, the final submission must be original and your own work. Take care not to give parts of your solution to other students.

Assignment Submission

The assignment must be submitted via a computer in the CSIT labs (or remotely via ssh to partch or one of our servers). Submit the assignment using the following command (remember you should only submit one file):

```
submit comp2300 ass2 mysolution.s
```

A few hits for submission:

- Don't have spaces in your file names (the submit program does not like them!).
- Copy the files to your home directory, rather, than having them mounted on a USB drive.
- You may submit as many times as you like, I will mark the most recent one that is not late.
- If you are in comp6300 then use comp6300 rather than comp2300 in the above command.

Hints

- Get the assignment working in stages (backup each stage as you go).
- Create some scripts that automates the testing of your solution.
- Don't be concerned about performance until you have all the stages working and you are certain that your solution is correct/robust.