CPE 315
Spring 2019
Seng
Lab #5
Complete by 11:59pm Friday night (5/31/19)

Description:

For this lab, you will implement a correlating branch predictor in your simulator. You only need to obtain the branch prediction accuracy. When implementing the branch predictor, remember to use the GHR (Global History Register - the shift register) as an index into the array of 2-bit counters. Initialize all 2-bit counters and the GHR to 0. Provide accuracy results for when the GHR is 2, 4, and 8 bits.

In addition, for this assignment you will implement (in MIPS assembly) the Bresenham line and circle drawing algorithms. These algorithms were invented by Bresenham to draw lines and circles using just integer computation. When you code the algorithms, implement them as functions so that you can call the code with the appropriate coordinate parameters.

In your main program, execute the following commands to draw a stick figure:

```
Circle(30,100,20) #head
Line(30,80,30,30) #body
Line(20,1,30,30) #left leg
Line(40,1,30,30) #right leg
Line(15,60,30,50) #left arm
Line(30,50,45,60) #right arm
Circle(24,105,3) #left eye
Circle(36,105,3) #right eye
Line(25,90,35,90) #mouth center
Line(25,90,20,95) #mouth left
Line(35,90,40,95) #mouth right
```

To plot a pixel, the x and y coordinates for each pixel should be stored into the data memory. For example, when your program is finished running, your data memory should look like this:

data memory location 0 = x coordinate of pixel 1
data memory location 1 = y coordinate of pixel 1
data memory location 2 = x coordinate of pixel 2
data memory location 3 = y coordinate of pixel 2
...and so on...

To view your picture, load the coordinates.csv file in Excel and plot the data as an x-y scatterplot.

Implement two additional commands in your simulator (should work in interactive and script):

- o = output a comma separated listing of the x,y coordinates to a file called coordinates.csv

- b = output the branch predictor accuracy.

With this additional branch predictor support, your simulator should accept a third optional argument which is the size of the GHR. If no GHR size argument is supplied, default to a size of 2. Here is an example command to run your simulator for this lab with a GHR size of 8:

```
java lab5 lab4_fib20.asm lab5.script 8
```

The pseudocode for the line algorithm is:

```
def Line (x0, y0, x1, y1) {
  if abs(y1 - y0) > abs(x1 - x0) {
    st = 1
  } else {
    st = 0
  }

  if st == 1 {
    swap(x0, y0)
    swap(x1, y1)
  }

  if x0 > x1 {
    swap(x0, x1)
    swap(y0, y1)
  }

  deltax = x1 - x0
  deltay = abs(y1 - y0)
  error = 0
  y = y0

  if y0 < y1 {
    ystep = 1
  } else {
    ystep = -1
  }

  for x from x0 to x1 { //include x0 and x1
    if st == 1 {
      plot(y,x)
    } else {
      plot(x,y)
    }

    error = error + deltay

    if 2*error >= deltax {
      y = y + ystep
      error = error - deltax
    } //end if
  }  //end for loop
} //end function
```

The pseudocode for the circle algorithm is:

```
def Circle(xc, yc, r) {  //xc = center x coordinate, yc = center y coordinate, r = radius
  x = 0
  y = r
  g = 3 - 2*r
  diagonalInc = 10 - 4*r
  rightInc = 6

  while x <= y {
    plot (xc+x, yc+y)  //plot the 8 different points to construct the circle
    plot (xc+x, yc-y)
    plot (xc-x, yc+y)
    plot (xc-x, yc-y)
    plot (xc+y, yc+x)
    plot (xc+y, yc-x)
    plot (xc-y, yc+x)
    plot (xc-y, yc-x)

    if g >=  0 {
      g += diagonalInc
      diagonalInc += 8
      y -= 1
    } else {
      g += rightInc
      diagonalInc += 4
    }

    rightInc += 4
    x += 1

  }  //end while loop
}  //end BresenhamCircle
```

Results

figure.script coordinates.csv

lab5.script

lab5_ghr2.output

lab5_ghr4.output

lab5_ghr8.output

2-bit GHR lab4_fib20.asm: 61.79% (8360 correct predictions, 13529 predictions)
4-bit GHR lab4_fib20.asm: 80.83% (10935 correct predictions, 13529 predictions)
8-bit GHR lab4_fib20.asm: 91.34% (12357 correct predictions, 13529 predictions)

Hand in

- Hand in all source files, Makefile and 1 assembly file (named figure.asm).