Creating Graphics using Microprocessor Programming

Ryan Tyson, Kevin Stanton and Nripendra Sarker

Department of Engineering Technology
Prairie View A&M University
Prairie View, TX 77446

Abstract

Microprocessor programming is tedious which depends on the architecture of registers of a particular CPU and the associated memory system. However, interrupt routines available in the memory offer many free ingredients to create amazing outputs. With IA-32 architecture, the Interrupt 10h allows setting screen resolutions of up to 800x600 pixels with 8-bit colors including routines to create graphics in 16-bit programming. The basic method of graphics programming is to set the video output to graphics mode and then calling appropriate routines of INT 10h to draw one pixel at a time at a desired location. This paper describes the method of drawing some algebraic expressions. The method is next translated into assembly language program to draw straight line, circle, parabola or ellipse as per runtime option.

Introduction

Microsoft Assembly Language compiler together with Interrupt routines available in the IBM PC memory offer strong programming capabilities at machine level using processor registers. This body of capabilities offers an opportunity to develop applications to run at a higher speed. Displaying the graphics of linear and quadratic equations are good examples of the power of programming in assembly language. Creating a graphic object requires coloring pixels one at a time. All pixels together in a sequence give the shape of the desired object.

The objective of this work was to design and write an assembly language program to draw graphic objects in the computer monitor. The selected objects are i) a straight line, ii) a parabola, and iii) a circle. The objects are decided during runtime including the required arguments.

Methods

The framework for the program was to develop graphic elements in several different procedures and macros. The Functions 00h and 0Ch of Interrupt 10h set the video mode, and colors a pixel, respectively. Coloring pixels in the Cartesian coordinate system for straight line and quadratic equations involved the major programming effort.

The program draws the origin of the x-y coordinates at pixel locations 400 and 400 pixels in x and y directions respectively with the top-left corner as the base pixel of the screen with 800x600 resolution, 4-bit color environment.

The arithmetic needed to produce the coordinates to plot points on the screen requires multiplication and addition. In a linear equation, $y = mx + b$, we find the y coordinate by solving for every values within the limit of the screen resolution with slope, m and y-intercept, b. Since, our program computes values in the negative domain of the Cartesian coordinates system; we used IMUL instruction in performing signed multiplication. We used similar instructions to perform the quadratic calculations for equations, $f(x) = ax^2 + bx + c$.

Drawing a circle requires trigonometric calculations that are processor intensive. We used the Bresenham's algorithm [1]. This algorithm is called the incremental approach, because the position of the next pixel is calculated on the basis of the last plotted one, instead of just calculating the pixels from a global formula. This logic allows plotting circles without trigonometry, for example.

The flowchart for developing the program is given in Figure 1.

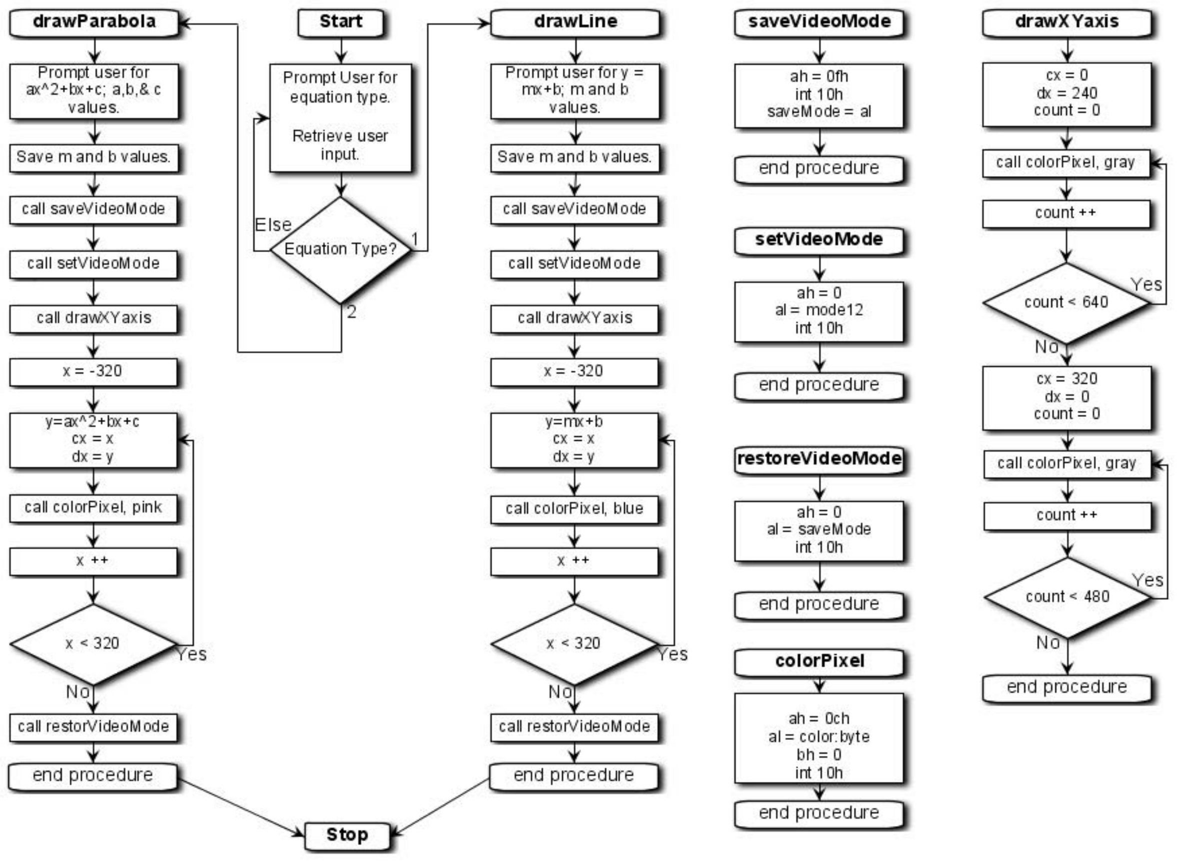


Figure 1. The program design in flowchart

Step 1

We wrote the following macros and procedures to complete the program

▪ setCursor,	accepts row and column values
▪ displayText,	display a test message
▪ saveVideoMode,	saves the current video mode
▪ setVideoMode,	sets the video to pixel programming mode
▪ colorPixel,	draws a single pixel at a set location
▪ drawXYaxis,	draws x-y coordinate axes
▪ drawLine,	draws a line based on the line parameters
▪ drawParabola,	draws a parabola based on its parameters

Step 2

Convert Bresenham’s algorithm into assembly language program

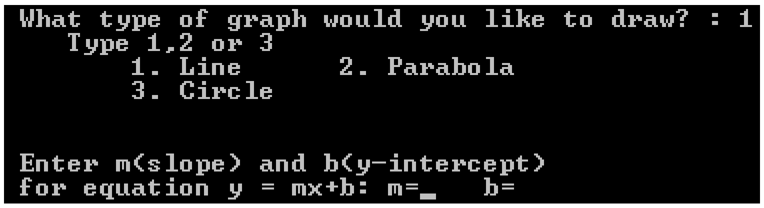
- Build drawCircle procedure

Step 3

- Compose and build the final program

Results

The program offers options for one of three objects to draw at a time. Once one object is selected, it asks for relevant parameters. Then it draws the object in the x-y coordinates. As shown in Figure 2, the user response is a ‘1’ to the question, “What type of graph would you like to draw?” The response ‘1’ is to draw a straight line. Next the program asks for slope, m and y-intercept, b.



```
What type of graph would you like to draw? : 1
   Type 1,2 or 3
       1. Line      2. Parabola
       3. Circle


Enter m(slope) and b(y-intercept)
for equation y = mx+b: m=_    b=
```

Figure 2. Option 1 is selected. The program asks for the parameters (slope and intercept) to draw a straight line.

Figure 3 shows two straight lines for two combinations of slope and y-intercepts. The slopes and intercepts were selected at random. Both the x- and y-scales were selected moderately large to accommodate a wide range of intercepts.

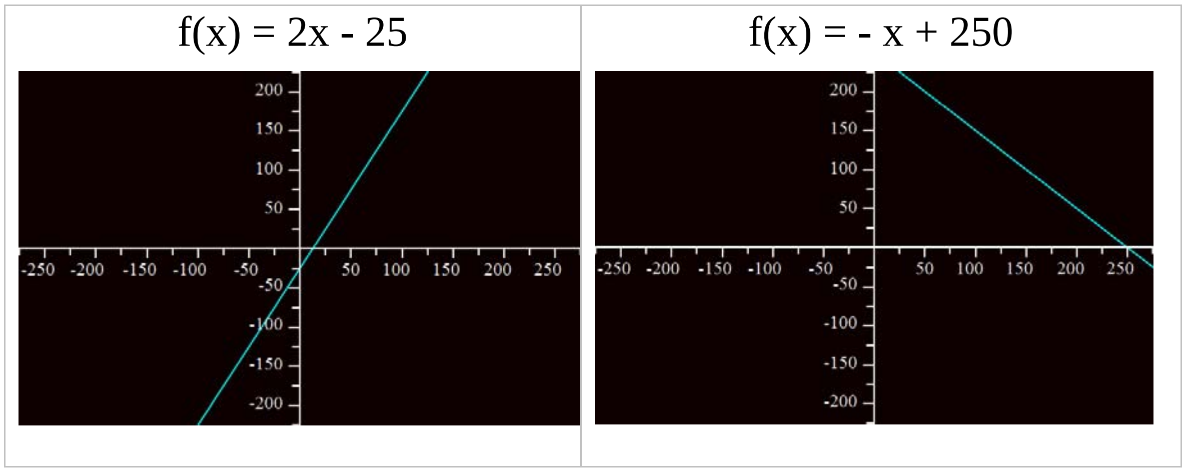
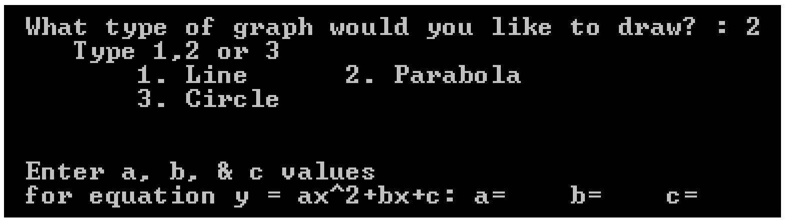


Figure 3. Two straight lines are drawn for two sets of inputs from the user.

To draw a parabola, option 2 is selected and the program asks for parameters, a, b, and c as shown in Figure 4. Figure 5 shows two parabolas for two sets of parameters.



```
What type of graph would you like to draw? : 2
   Type 1,2 or 3
       1. Line        2. Parabola
       3. Circle


Enter a, b, & c values
for equation y = ax^2+bx+c: a=     b=     c=
```

Figure 4. Option 2 is selected. The program is asking for its parameters to draw a parabola.

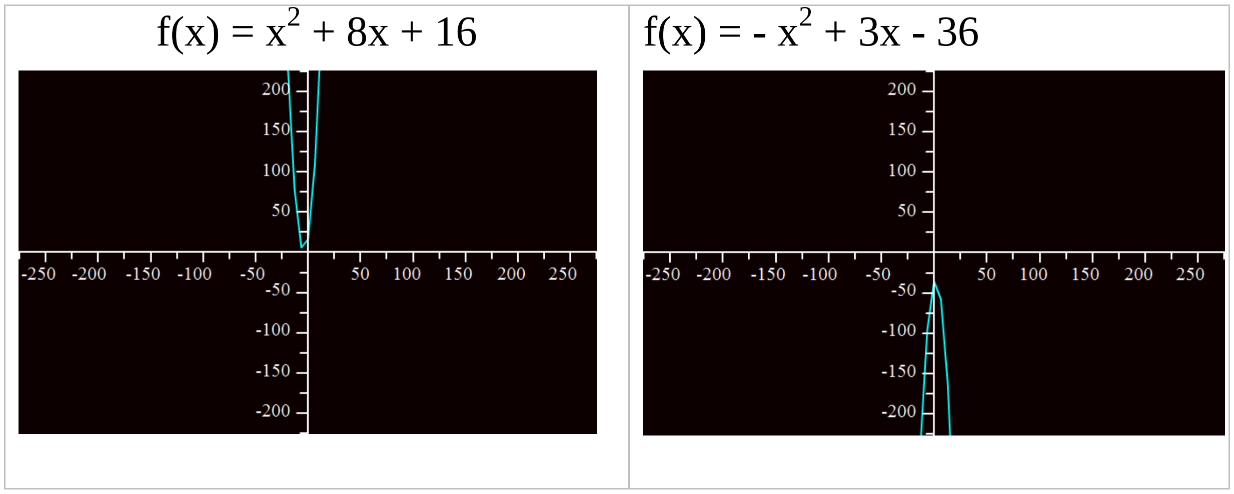
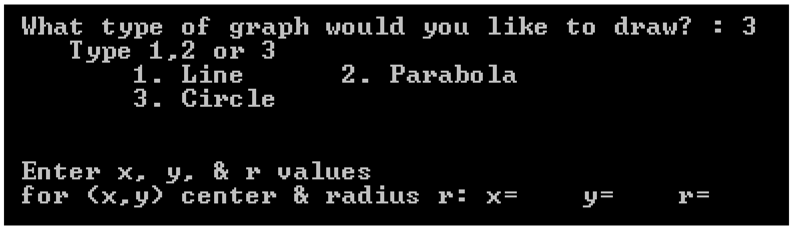


Figure 5. Two parabolas are drawn for two sets of parameters

To draw a circle, option 3 is selected and the program asks for its parameters as shown in Figure 6. Figure 7 shows two circles for two sets of parameters.



```
What type of graph would you like to draw? : 3
   Type 1,2 or 3
       1. Line         2. Parabola
       3. Circle


Enter x, y, & r values
for (x,y) center & radius r: x=     y=     r=
```

Figure 6. Option 3 is selected. The program asks for parameters for drawing the circle.

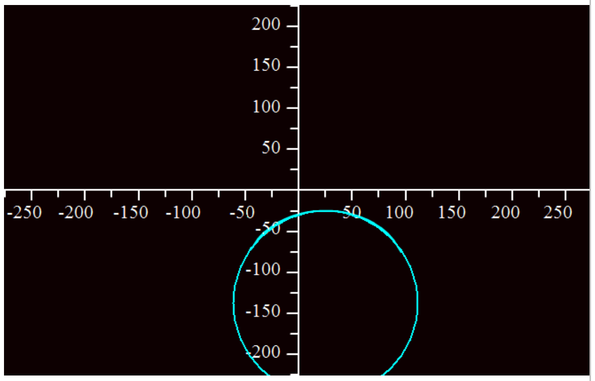
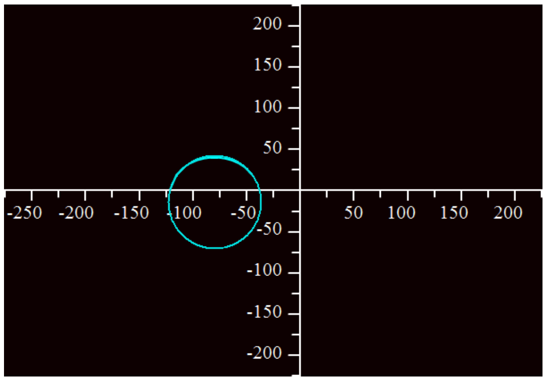
r = 75 with center at (25, -125)	r = 40 with center at (-80, -25)
	

Figure 7. Two circles are drawn with radii 75 and 40 at (25, -125) and (-80, -25) respectively.

Conclusion

Although the trials were successful, the program has natural limitations on what can be calculated and drawn on the screen. Since 16-bit registers were used, the maximum and minimum values depended on the size limitations inherent with the register sizes. Video resolution (800x600) restricts the coordinate space to x-coordinate as -400 to 399 and y-coordinate as -300 to 299. A zoom out and zoom in feature can be added by scaling in and out the parameter values in each case (line, parabola and circle)

In this paper, we confirmed the ability to process algebraic functions very quickly and efficiently. We also discovered the core assembly language instructions to manage trigonometry and draw smooth lines for an equation using Bresenham's [1] algorithm. An excellent extension to this project would to write modular programs to include various functions to mimic functionality of a graphics calculator.

References

1. Dazibao, Mandelbrot. 2007. “The Bresenham’s algorithm – Multiplication and Division.” Fifth Edition, pp. 204-206.

TYSON RYAN

Mr. Tyson is a graduating senior at Prairie View A&M University. He will receive his Bachelor of Science degree in Computer Engineering Technology on December 13, 2008. Mr. Tyson is currently a Network Engineer for Verizon Telecommunications, where he specializes in network operation, management, and analysis. Over the past 10 years he has gained industrial experience in server technology, web applications development, database programming, and various methods of compiled coding and scripting.

KEVIN STANTON

Mr. Stanton is a graduating senior at Prairie View A&M University. He will receive his Bachelor of Science degree in Computer Engineering Technology on December 13, 2008. Mr. Stanton has specialized research experience in networking technologies with emphasis on IP networks based in ATM, SONET and Wireless.

NRIPENDRA N. SARKER
Dr. Sarker is currently Lecturer in the Department of Engineering Technology of the Prairie View A&M University, TX. He also worked at universities in Bangladesh, Japan and UT - San Antonio. He received his Master’s and PhD degrees from the Texas A&M University at College Station. His research interests include simulation, algorithm development, and computer networking. He is the Chair of departmental ABET/SACS committee and a member of the College Committee for ABET at PVAMU.