


Programming Lab 7G

 Click to download Lab7G-Main.c

ASCII Art & Font Tables

Topics: Bit manipulation, shift and bitwise instructions, nested loops.

Prerequisite Reading: Chapters 1-7

Revised: June 22, 2021

Background¹: ASCII art is a graphic design technique that uses computers for presentation and consists of pictures pieced together from the printable characters defined by the ASCII Standard. ASCII art can be created with any text editor, and is often used with free-form languages. Most examples of ASCII art require a fixed-width font such as Courier for presentation. ASCII art was invented, in large part, because early printers often lacked graphics ability and thus characters were used in place of graphic marks. Also, to mark divisions between different print jobs from different users, bulk printers often used ASCII art to print large banners, making the division easier to spot so that the results could be more easily separated by a computer operator or clerk. ASCII art was also used in early e-mail when images could not be embedded.

Font Tables: Each symbol is written to the screen as a two-dimensional array of pixels. Fonts come in various sizes, specified by their *Height x Width* in pixels. Our run-time library provides five fonts of 94 characters; each font table is a sequence of symbols, starting with the ASCII space. Each symbol is a two-dimensional bitmap with one bit per pixel. This lab uses the 8x5 font in which each symbol is represented by eight bytes – one for each row of the symbol’s bitmap. The left-most 5 bits of each byte represent the 5 pixels in that row of the character.

```
0xF8, // #####
0x48, // # #
0x60, // ##
0x40, // #
0x48, // # #
0xF8, // #####
0x00, //
0x00, //
```



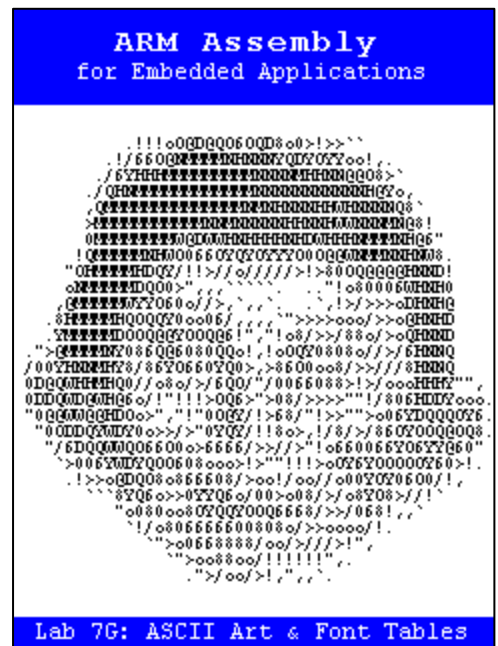
Algorithms: The 240x240 gray scale image (shown at left) is stored as data inside the main program, which converts it from BMP24 format into ASCII art for display. To do that, it analyzes the 8x5 font table to select the best symbols to use to represent a range of approximate gray scale values. When displaying black text on a white background, symbols with very few 1’s in their bitmap appear whiter than those with lots of 1’s in their bitmap. For example, the space (blank) is used for very white areas since there are no 1’s in its bitmap, while an uppercase ‘M’ is used for very dark areas. When the bitmaps of more than one symbol contain the same number of 1’s, the program uses the symbol whose 1’s are most evenly distributed over the rows and columns of its bitmap. The symbol with the most evenly distributed 1’s is determined by computing the product of the maximum number of 1’s per row times the maximum number of 1’s per column; the smaller that product, the better the distribution.

Assignment: The main program will compile and run without writing any assembly. However, your task is to create equivalent replacements in assembly language for the following three functions found in the C main program. The original C versions have been defined as “weak” so that the linker will automatically replace them in the executable image by those you create in assembly; you do not need to remove the C versions. This allows you to create and test your assembly language functions one at a time. Each function receives a single parameter that is the 8x5 bitmap of a specific symbol. These functions are called by a main program that converts a grayscale image stored in the program into an array of ASCII characters. Error messages will be display in **white text on a red background** and cause the program to halt.

```
unsigned TotalBits(uint8_t bitmap[]);
Returns the total number of black pixels (1’s) in the symbol’s bitmap.

unsigned MaxPerRow(uint8_t bitmap[]);
Returns the maximum number of black pixels per row in the bitmap.

unsigned MaxPerCol(uint8_t bitmap[]);
Returns the maximum number of black pixels per column in the bitmap.
```



¹ https://en.wikipedia.org/wiki/ASCII_art