CS1022 Tutorial #4
2-D Arrays

1 Matrix Multiplication

(a) Write an ARM Assembly Language **subroutine** to retrieve the value of an element from a two-dimensional array of word-size values stored in row-major order. Your subroutine must accept as parameters the start address of the array in memory, the row and column indices of the element to retrieve and the number of elements in each row. Begin by designing an appropriate interface for the subroutine before implementing the subroutine itself.

(b) Write a second ARM Assembly Language **subroutine** to set the value of an element in a two-dimensional array of word-size values stored in row-major order. Your subroutine must accept as parameters the start address of the array in memory, the row and column indices of the element to retrieve, the number of elements in each row and the new value to be stored in the array. Begin by designing an appropriate interface for the subroutine before implementing the subroutine itself. (Note: There are five parameters to be passed so you will need to use the system stack to pass one of them.)

(c) Re-implement the matrix multiplication program from Lab #1 as a subroutine, making use of your get2D and set2D subroutines from Parts (a) and (b) above. Modified pseudo-code is provided below. (The parameters A, B, R and N are the starting addresses of arrays A, B and R and the dimension of of the three arrays respectively.) Begin by designing an appropriate interface for your matmul subroutine.

```
matmul(A, B, R, N) {
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      r = 0;
      for (k = 0; k < N; k++) {
        r = r + ( get2D(A, i, k, N) * get2D(B, k, j, N) );
      }
      set2D(R, i, j, N, r);
    }
  }
}
```

2 Image Manipulation

(a) Write ARM Assembly Language subroutines to retrieve the Red, Green and Blue components of a specified pixel in an image. Assume the image is set out in memory as described in the handout for Assignment #1. Suggested pseudo-code interfaces for the subroutines are provided below but you may wish to design your own interface.

```
int getPixelR(address, i, j, width)
int getPixelG(address, i, j, width)
int getPixelB(address, i, j, width)
```

(b) Write corresponding subroutines to set the Red, Geen and Blue components of a soecified pixel.

(c) Use your subroutines to implement a subroutine that converts an image to greyscale. You may use the subroutines described in the Assignment #1 handout (getPicAddr, getPicWidth, getPicHeight, putPic) in your solution.