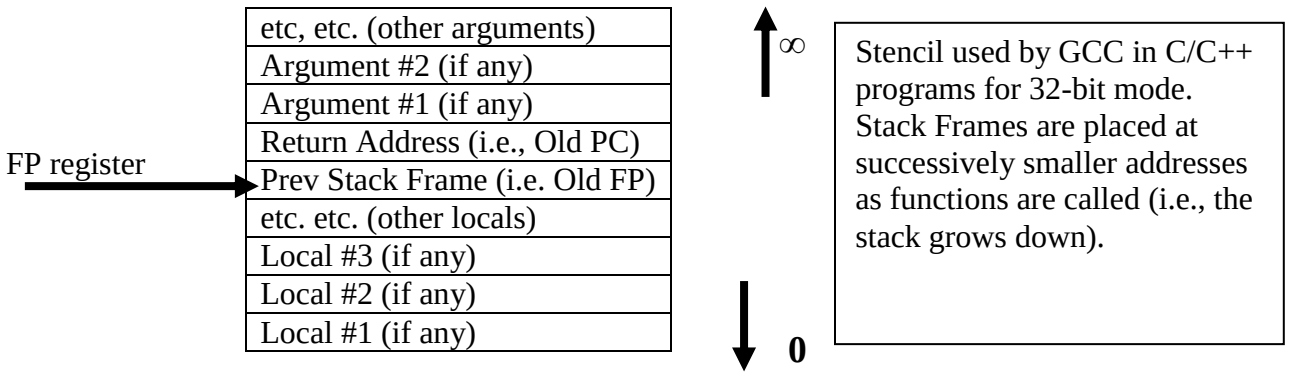
EE312 Fall 2014

Exercise 1R, Sept 11 and 12, 2014

There are two parts to this exercise. There is no prelab. The first part of the exercise is to draw a diagram representing the initial state of all seven parameters for the multiplyMatrixChain function in Project2. Please keep in mind that the actual arrays will be local variables from the calling function. The parameters will be (of course) pointers to those arrays. Array parameters are always pointers. This is particularly significant with the parameter named *out*. The second part of the exercise is a page from Midterm #1 in Spring 2013. Please work through these questions to confirm your understanding of basic pointer mechanics.

1. Draw a diagram showing a possible initial state of all the parameters for the multiplyMatrixChain function from Project2. The state you choose should have at least three input matrices to be multiplied, and the matrices should be different sizes. Note that even though we are using matrices in this project the arrays are all one-dimensional arrays. Review your diagram with the TA to ensure that the diagram is reasonably close to correct (you'll need this knowledge before you can successfully complete Project2).

Shown below is an approximation of the layout stencil that GCC uses when building a stack frame. You should assume that a 32-bit, little-endian computer is used, and that the following stencil is used to organize the information contained in a stack frame.



2. Please assume the following variables are all local variables for some function "doit". Refer to the stencil above and note that the compiler assigns memory locations to variables based on the order variables are declared – the first variable declared has the smallest address and each subsequent variable occupies the next larger address (subject to the size of each variable). **Also please assume that the address assigned to the variable x is 1000**. Answer each of the questions below.

```
int x = -1; //  recall these are local variables in a function, and &x == 1000
int* p = &x;
char* q = (char*) p;
char** r = &q;
```

   a. What is the value of: *p?

   b. What is the value of: &p?

   c. What is the value of: *(p + 1)?

   d. What happens to the variable x when I do the assignment **r = 0;
      i. x becomes zero
      ii. x becomes some value other than zero
      iii. x doesn't change

   e. Note that I have two variables, p and q where p == q, but p is of type int* and q is of type char*. I want to be certain that *p != *q. What value can I assign to *p that will guarantee *p != *q. I'm looking for a number here, like "0" or "1". (there's more than one right answer and 0 and 1 are not the right answers).