# Assembly: Chapters 3, 4, 7

1. Be able to translate C code into assembly that is callable from C.
   1. Be able to translate if, for, while, and function calls
   2. Be able to translate expressions involving multidimensional arrays declared either statically or dynamically
   3. Be able to use the advanced indexing mode to access arrays
      1. Know which bytes of memory are accessed by an instruction. For example if eax = 10 and ebx = 5 which bytes of memory are accessed by
         1. Movl (%eax), %ecx
         2. Movw (%eax), %cx
         3. Movb (%eax), %cl
         4. Movl (%eax,%ebx, 4), %ecx
         5. Movw (%eax, %ebx, 2), %cx
         6. Movb (%eax, %ebx, 1), %cl
   4. How does the typing of variables affect the assembly code you write? For example translate the following lines of C
      1. Int A = 7; //assume A is in EAX
      2. Short B = 7; //assume B is in EBX
   5. Be able to use the leal instruction.
   6. Be able to translate recursive functions
2. What are the gcc C calling conventions? What happens if you break them?
3. What is defined to be the stack?
4. What is defined to be the current stack frame?
5. What does it mean for the stack frames to be chained? How do they become chained?
6. What is the code for the prologue? What is its purpose?
7. What is the code for the epilogue? What is its purpose?
8. Know what push, pop, call, and ret do.
9. Where is space for local variables made?
   1. Global?
   2. Static?
      1. What prevents you from accessing a static variable outside of the function it is declared in?

# Debugging

Be able to do following

1. Print out a variable
2. Print out a register
3. Put out the value of an argument on the stack after the prologue has run
4. Print out the value of a local variable after the prologue has run
5. Print out the elements in an array if the pointer to an array is in the register
   1. If EAX contains the pointer to an array of integers named ar, print out the first 10 elements of ar.
6. Print out the elements in an array if the pointer to the array is on the stack and the prologue has been run
   1. Print the first 5 elements of ar in: foo(int a, int b, int\* ar)
   2. Print the first 5 elements of ar in: foo(int a, int b, char\* ar)
   3. Print the first len elements of ar in: foo(int a, int\* ar, int len)
7. Print out a row of elements of a 2D array
   1. int\*\* ar is stored in esi. Print out the first 4 elements of row at index 7
   2. foo(int\*\* ar, int num\_rows, int num\_cols). Print out the first 10 elements of ar at row index 2
   3. foo(int\*\* ar, int num\_rows, int num\_cols). Print out num\_cols elements of ar at row index 5
8. Set a break point on a line
9. Go to the next break point
10. Go to the next line of code skipping over function calls
11. Go to the next line of code going inside of function calls