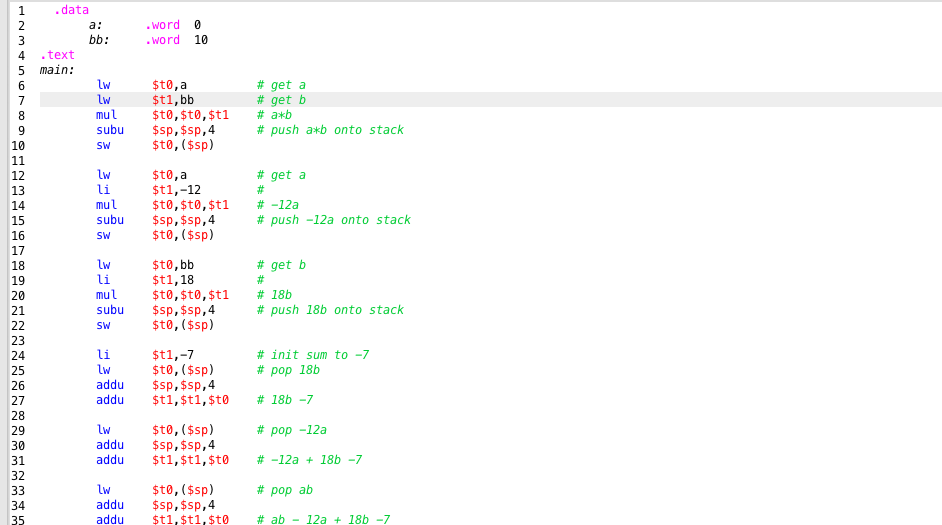
Asif Saifullah

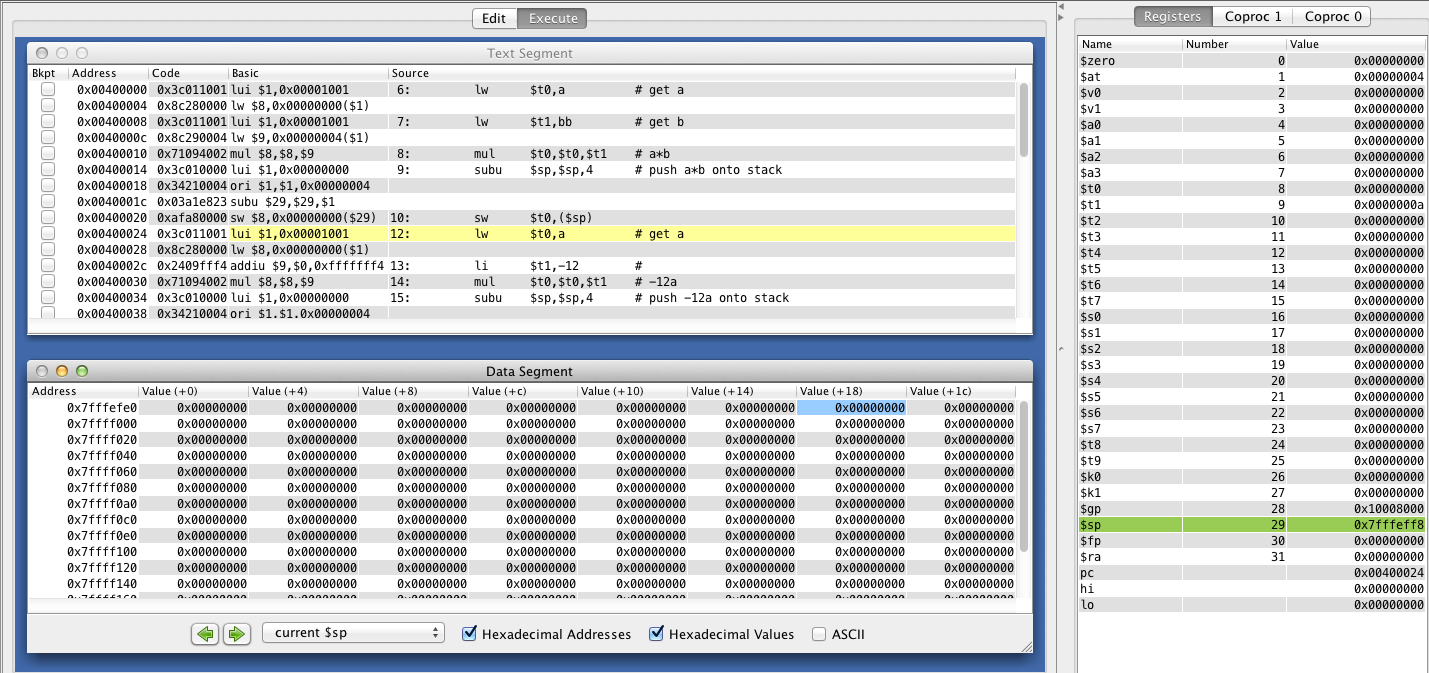
CSC 343 Spring 2016

Homework: Stack.

**Stack :** When a program starts executing, a particular contiguous section of memory is set aside for the program called the stack. There largest valid address of a stack when stack initialized. That’s base of the stack or stack bottom. At the time of the initialization, we base pointer and stack pointer both points to the same address of the stack which base of the stack. Base pointer stays the same and stack pointer points to the top of the stack. There is also a smallest valid address for the stack, which is the limit for that particular stack. When stack pointer gets above the stack, we get the overflow of the stack.

Here in this homework, there is an example of how the stack works, and also how two important concepts of stack push and pop works. The example is given in MIPS assembly language. Here, the expression ab - 12a + 18b – 7 is evaluated with the concept of the stack.



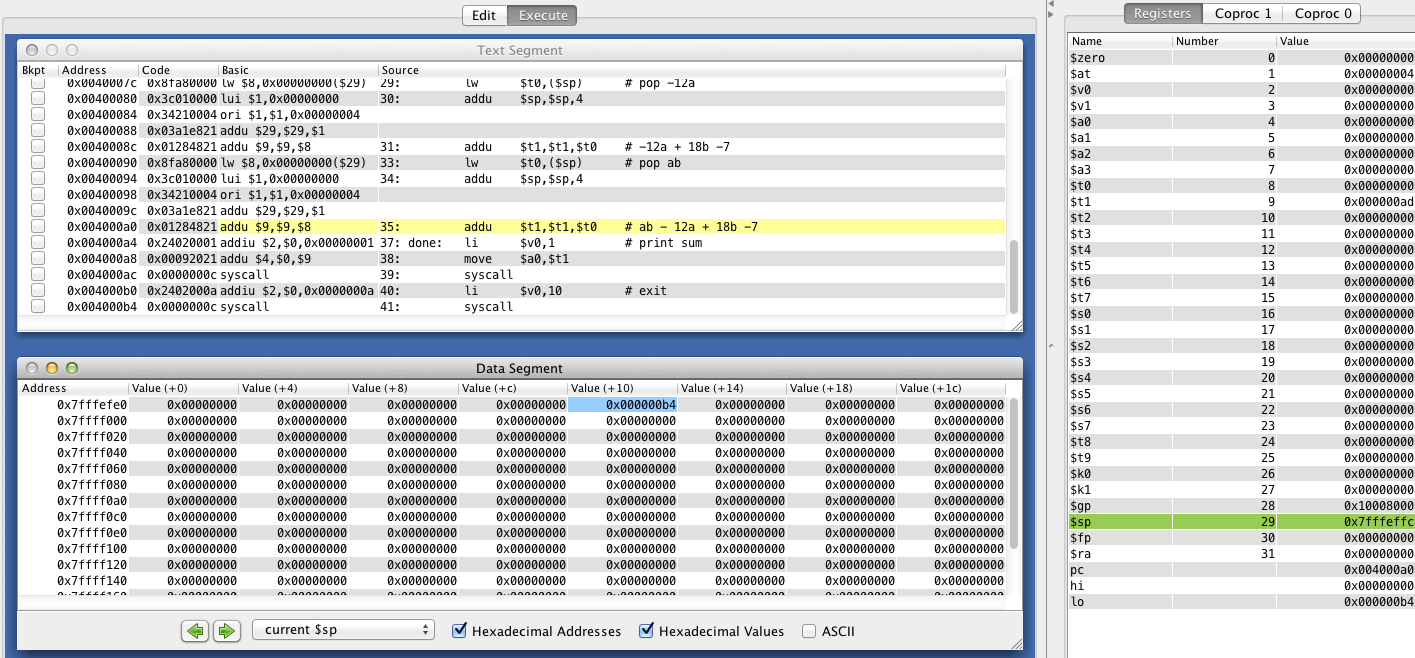


**Push:**

From the above screenshots, we can see that we took two data a and b and doing multiplication. After that, we pushed that value to the stack by decrementing the stack pointer ($sp) to it lower address and then storing the element on that address. From the register window, we could see that initially stack pointer points to the address of 0x7fffeffc. For this program, it could be our base pointer as well which points to the base of the stack. We decremented the stack value by 4 we could see that stack pointer points to the next smaller address, which is 0x7fffeff8. And the value (a\*b) stored in the memory of that address. Thus we push our operations in the stack by decrementing the stack pointer address and store the value -12 , 18b.

**Pop:**

In that when we reached -7 then we will add with the value of the top of the stack. As we know, in stack the last value of the stack which is in will be out first. To take the value out of the stack we usually use pop operation.



From the code, we could see that to pop up the top element from the stack we first load the value into a register then we move our stack pointer to the next larger address. For example, when we want to pop up a\*b which is in the top of the stack we can see from the register window that stack pointer ($sp) is pointing to the address of 0x 7fffeff8. Then after loading the value into a register when we increment the value of the stack by 4 and now our stack pointer is pointing to the next larger address 0x7fffeffc. We used the same method to do get value from the top of the stack in other cases as well.