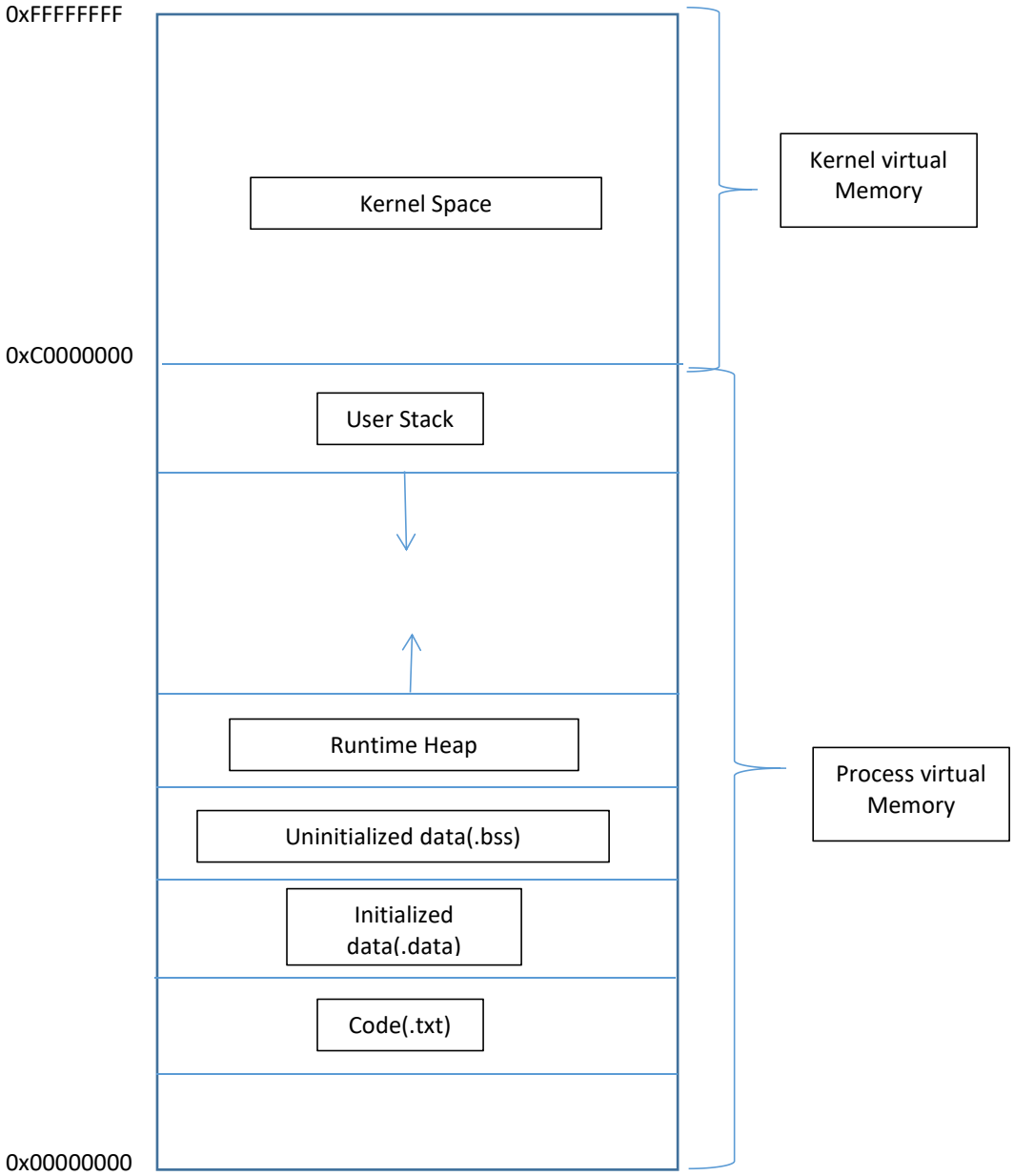
1. 1.Draw an image for the virtual address space of the Linux process in 4G memory

   Soln:



2. What is the difference between a, b and c? From the stack perspective?

Soln:

1. variable a (in main)
   a) a is a local variable in main function
   b) it is allocated on the stack frame when its declared
   c) it is deallocated when when main exit
2. variable b(argument of test_fucn)

a) b is a function parameter of test_func
b) when test_fun is called b is stored in the caller function(main) stack frame.
c) The way b is stored depends on the calling convention
    i. In most systems, function arguments are pushed onto the stack before calling the function
    ii. In some calling conventions, registers may be used instead of the stack

3. variable c(test_fun)
a) c is a local variable of test_fun
b) It is allocated inside the stack frame of test_fun
c) It remains valid while test_fun is executing and is deallocated when test_fun returns

3.The practice based on the program "Function call stack.c"
a.Write the corresponding C code next to the assembly statement

soln:

```
fun2:                          // function name
        pushl   %ebp           //Save caller's base pointer
        movl    %esp, %ebp     //Set new base pointer
        subl    $16, %esp      //Allocate 16 bytes for local variables
        movl    $111, -4(%ebp)  //int aaa = 111;
        movl    $222, -8(%ebp)  //int bbb = 222;
        movl    8(%ebp), %edx   // int m =(first value passed from caller)
        movl    -4(%ebp), %eax  // int x = aaa;
        addl    %eax, %edx      // m = m + x;
        movl    12(%ebp), %eax // int n =(second value passed from
caller) ;
        addl    %eax, %edx      // m = m+ n;
        movl    -8(%ebp), %eax  // x = bbb;
        addl    %edx, %eax      // x  = x + m;
        leave                   // restore stack and base pointer
        ret                     // return x // return caller
```

```
fun1:                          // function name
        pushl   %ebp           // Save caller's base pointer
        movl    %esp, %ebp     //Set new base pointer
        subl    $16, %esp      //Allocate 16 bytes for local variables
        movl    $11, -4(%ebp)   //int aa = 11;
        movl    $22, -8(%ebp)   //int bb = 22;
        movl    -8(%ebp), %edx   // retrive local variable bb;
        movl    12(%ebp), %eax  // int n =(second value passed from
caller) ;
        addl    %eax, %edx      // bb = bb + n;
        movl    -4(%ebp), %ecx  // retrieve local variable aa;
        movl    8(%ebp), %eax   // int m =(first value passed from caller) ;
        addl    %ecx, %eax      // aa = aa + m;
```

```
                pushl   %edx            // passing second parameter for function
call
                pushl   %eax            // passing first parameter for function call
                                        // fun2( a, b )
                call    fun2             // calling func2 that means storing
returning address
                addl    $8, %esp        // release stack memory after func2
complete its task
                leave                    // restore stack and base pointer
                ret                      // return value save saved in eax
//return caller



        main:                           // function name
                pushl   %ebp             // Save caller's base pointer
                movl    %esp, %ebp      //Set new base pointer
                subl    $16, %esp       //Allocate 16 bytes for local variables
                movl    $1, -4(%ebp)     //int a = 1;
                movl    $2, -8(%ebp)     //int b = 2;
                pushl   -8(%ebp)        // passing second parameter for function
call
                pushl   -4(%ebp)        // passing first parameter for function call
                                        //fun1( a, b )
                call    fun1             // calling func1 that means storing
returning address
                addl    $8, %esp         // Clean up the stack ( remove functions
arguments)
                movl    $0, %eax         // return 0;
                leave                   // restore stack and base pointer
                ret                     // return to the caller
```

b.Draw some pictures or tables to illustrate the memory layout and content of the register after each assembly instruction.

Label	Instruction	%ebp	%esp	%eax	%ecx	%edx	stack change

M1	pushl %ebp	0x1100	0x0FFC				[0x0FFC] = 0x1100
M2	movl %esp, %ebp	0x0FFC	0x0FFC				
M3	subl $16, %esp	0x0FFC	0x0FEC				Reserved 16 bytes
M4	movl $1, -4(%ebp)	0x0FFC	0x0FEC				[0x0FF8] = 1
M5	movl $2, -8(%ebp)	0x0FFC	0x0FEC				[0x0FF4] = 2
M6	pushl -8(%ebp)	0x0FFC	0x0FE8				[0x0FE8] = 2
M7	pushl -4(%ebp)	0x0FFC	0x0FE4				[0x0FE4] = 1
M8	call fun1	0x0FFC	0x0FE0				[0x0FE0] = RET_ADDR_MAIN
F11	pushl %ebp	0x0FFC	0x0FDC				[0x0FDC] = 0x0FFC
F12	movl %esp, %ebp	0x0FDC	0x0FDC				
F13	subl $16, %esp	0x0FDC	0x0FCC				Reserved 16 bytes
F14	movl $11, -4(%ebp)	0x0FDC	0x0FCC				[0x0FD8] = 11
F15	movl $22, -8(%ebp)	0x0FDC	0x0FCC				[0x0FD4] = 22
F16	movl -8(%ebp), %edx	0x0FDC	0x0FCC			22	
F17	movl 12(%ebp), %eax	0x0FDC	0x0FCC	2		22	
F18	addl %eax, %edx	0x0FDC	0x0FCC	2		24	
F19	movl -4(%ebp), %ecx	0x0FDC	0x0FCC	2	11	24	
F110	movl 8(%ebp), %eax	0x0FDC	0x0FCC	1	11	24	

F111	addl %ecx, %eax	0x0FDC	0x0FCC	12	11	24	
F112	pushl %edx	0x0FDC	0x0FC8	12	11	24	[0x0FC8] = 24
F113	pushl %eax	0x0FDC	0x0FC4	12	11	24	[0x0FC4] = 12
F114	call fun2	0x0FDC	0x0FC0				[0x0FC0] = RET_ADDR_FUN1
F21	pushl %ebp	0x0FDC	0x0FBC				[0x0FBC] = 0x0FDC
F22	movl %esp, %ebp	0x0FBC	0x0FBC				
F23	subl $16, %esp	0x0FBC	0x0FAC				Reserved 16 bytes
F24	movl $111, -4(%ebp)	0x0FBC	0x0FAC				[0x0FB8] = 111
F25	movl $222, -8(%ebp)	0x0FBC	0x0FAC				[0x0FB4] = 222
F26	movl 8(%ebp), %edx	0x0FBC	0x0FAC			12	
F27	movl -4(%ebp), %eax	0x0FBC	0x0FAC	111			
F28	addl %eax, %edx	0x0FBC	0x0FAC	111		123	
F29	movl 12(%ebp), %eax	0x0FBC	0x0FAC	24		123	
F210	addl %eax, %edx	0x0FBC	0x0FAC	24		147	
F211	movl -8(%ebp), %eax	0x0FBC	0x0FAC	222		147	
F212	addl %edx, %eax	0x0FBC	0x0FAC	369		147	
F213	leave	0x0FBC	0x0FBC	369		147	
F214	ret	0x0FDC	0x0FC4	369		147	Jump to fn1

F115	addl $8, %esp	0x0FDC	0x0FCC	369		147	
F116	leave	0x0FDC	0x0FDC	369		147	
F117	ret	0x0FFC	0x0FE4	369		147	Jump to main
M9	addl $8, %esp	0x0FFC	0x0FEC	369		147	
M10	movl $0, %eax	0x0FFC	0x0FEC	0		147	
M11	leave	0x0FFC	0x0FFC	0		147	
M12	ret	0x1100	0x1000	0		147	Jump to os