CS 271 Computer Architecture and Assembly Language

Self-Check for Lecture#14

Solutions are posted

Here is a partial data segment:

```
MAX = 50
.data
...
list      DWORD        MAX DUP(0)
a         DWORD        25
b         DWORD        15
 ...
```

1. Given: the address of `list` is 0x0300.
   a. What is the (hexadecimal) address of a? ____________

   b. What is the (hexadecimal) address of the 33rd element of list? ____________
   (Hint: in C or Java, the 33rd element is list[32])

Here is a partial "listing file" that uses the data segment above:

```
00000000        main         PROC
00000000                     push  a
00000005                     push  b
0000000A                     push  OFFSET list
0000000F                     call  someProc
00000014        next         ...

                             exit  ;exit to operating system
0000006C        main         ENDP

0000006C        someProc     PROC
0000006C                     push  ebp
0000006F                     mov   ebp, esp
00000072        etc          ...

0000008B  C3                 ret   ;return to calling procedure
0000008C        someProc     ENDP
```

2. Initially, `esp` contains 0A04, and `ebp` contains 0BB9. `main` has called `someProc`, and the first two statements of `someProc` have been executed.

   a. ebp contains ___________
   b. Show the contents of the system stack →
   c. Write a statement to move the value of actual parameter a into the eax register. (Global name a is not permitted.)

      ____

   d. Write the statements to move the value of the bth element of list into the ebx register. (Consider b=0 to be the 1st element of list) (Global names b and list are not permitted.)

Address	Contents	Meaning
09E4		
09E8		
09EC		
09F0		
09F4		
09F8		
09FC		
0A00		
0A04	?	unknown

3. Given the following partial data segment:

```
.data
loVal          DWORD      ?
hiVal          DWORD      ?
randVal        DWORD      ?


.code
main     PROC
         call  Randomize         ; from the Irvine library

;  Code to get loVal and hiVal from the user goes here.

         push  loVal
         push  hiVal
         push  OFFSET randVal
         call  nextRand

;  More main procedure code

         exit
main     ENDP
```

Write the `nextRand` procedure so that it satisfies the following header documentation. You may use appropriate Irvine library procedures. Note that used registers must be saved and restored.

```
; Procedure nextRand
; Procedure to get the nest random number in the range specified by the user.
; Receives parameters on the system stack (in the order pushed):
;        Lowest acceptable value (loVal)
;        Highest acceptable value (hiVal)
;        Address of return value
; Preconditions:  loVal < hiVal
; Registers used:  none
```