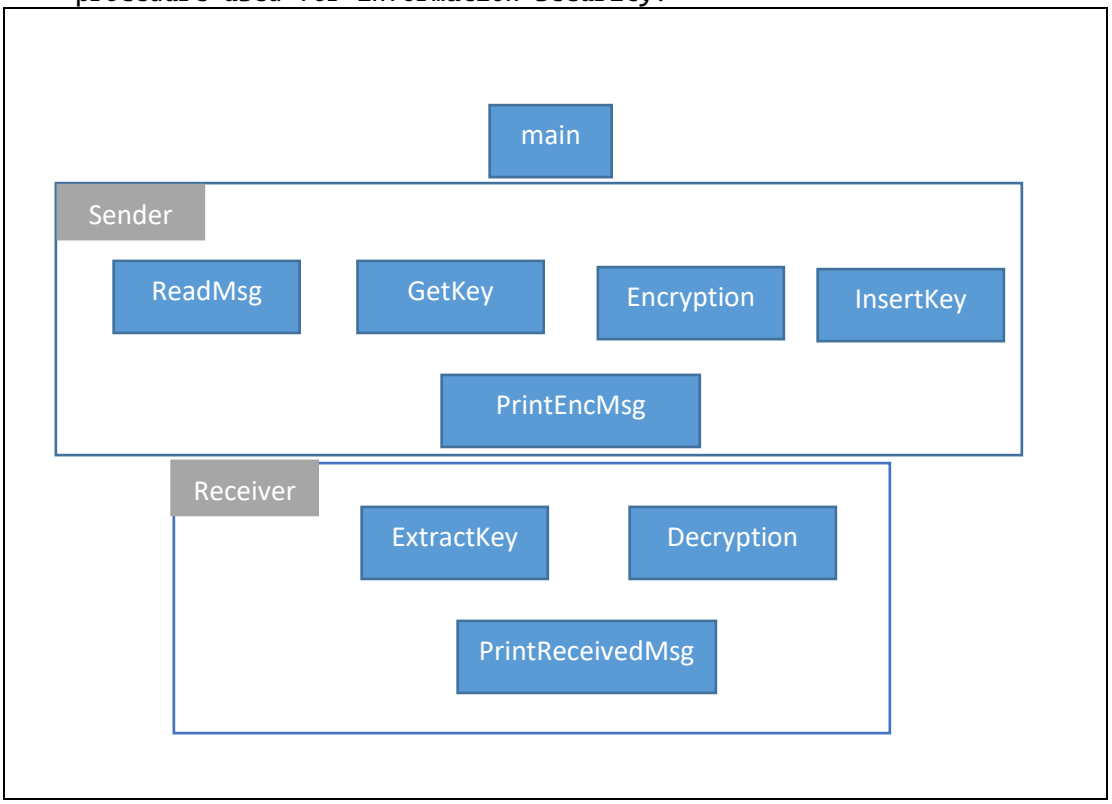
CS221 Assembly Language

Lab8

Prof. Mohamed Ben Othman

1- **Exercise:**

You will write a program that has three procedures besides the main procedure used for information security:



1- Sender: it calls:
   a. readMsg: to read a message from the user and put it is a variable called *Msg*.
   b. getKey: to read the key which is a number between 100 and 200 and put it in a variable (byte) called Key.
   c. Encryption: using base addressing mode (ebx for source and esi for destination) and loop to convert the message *Msg* using xor and put it in *EncMsg* variable.
   d. insertKey: to insert the value of the key at the end of the *EncMsg*.
   e. printEncMsg: that prints the encrypted message where the key is inserted.
2- Receiver: as if we send the encrypted message to a received, this procedure will call:

a. extractKey: that extracts the key from the end of the message
b. decryption: using the extracted key decrypts the message and put it in DecMsg variable using the index addressing mode (esi for source and edi for destination).
c. printRecivedMsg: prints the decrypted message.

Note: 1) in all the procedure where ecx that contains the size of the entered string should be preserved.
2) you can add two encryption levels as you wish: example using another xor with a second key or any function that you can reverse in the receiver.

2- **Quiz:**

You may have Quiz this week.

Program Skeleton

```
TITLE  "Lab 8: Information Security"
.686
.MODEL FLAT, STDCALL
.STACK
INCLUDE Irvine32.inc
.data
msgMaxSize EQU 100
msg byte msgMaxSize dup(0)
encMsg byte msgMaxSize dup(0)
key byte ?
decMsg byte msgMaxSize dup(0)

.code
main proc
   ;; Here you call your sender and receiver procedures


      exit
main endp

;; sender
sender proc
      ; add here the sender code


      ret
sender endp
;; sender functions
readMsg proc

      ret
readMsg endp

getKey proc

      ret
getKey endp
```

```
encryption proc uses ecx

	ret
encryption endp

insertKey proc


	ret
insertKey endp

PrintEncMsg proc


	ret
PrintEncMsg endp

;; receiver
receiver proc
	; add here the receiver code


	ret
receiver endp

extractKey proc uses ecx


	ret
extractKey endp

Decryption proc uses ecx


	ret
Decryption endp

PrintReceivedMsg proc


	ret
PrintReceivedMsg endp

END main
```