

Lab 10 bonus

- ; Greg Violan, 011706641
- ; Andy Sien 012064856
- ; Lab 10 bonus

```
;variable and constant definitions
keyBytesRAMAddress EQU 0x40 ; symbolic constant for base address of
; encryption key in RAM
keyLength EQU 0x30 ; variable to track length of key
keyvalIndex EQU 0xe0 ; variable to index the keyval constant array
; keyvalIndex is also an alias for accumulator
```

chIndex EQU 0xe0 ;alias variable for Accumulator

```
;begin section from lab 9  
;vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv  
jmp main ;jump past interrupt vector table  
org 0x0030 ;put main program at rom location 0x0030
```

```
main:
    mov tmod, #0x21      ; configure timer 0, mode 1
    mov scon, #0x50      ; config serial 8-data, 1 start, 1 stop, no parity
    mov th1, #0xFD       ; 9600 baud
    setb tr1             ; start timer 1 to enable serial communication
    mov dptr, #Prompt     ; initialize ROM pointer
    Call displayChar      ; call displayChar
    mov r0, keyBytesRAMaddress ; initialize RAM pointer
```

```
main_loop2:
    jnb ri, $                                ; wait to receive a char
    call getchar                            ; char received, get it!
    clr ri                                  ; acknowledge char receive
    mov @r0, a                              ; move key into RAM
    inc r0                                  ; increment RAM pointer
    cjne a, #0x00, main_loop2 ; check for null character in string
    mov dptr, #keyFile                    ; initialize ROM pointer
    Call displayChar                      ; call displayChar
```

```
;----- Initialization/configuration -----;
;keyval variable no longer used
; mov keyval, #0x23 ;load the keyval variable with encryption key
mov tmod, #0x20 ;config timer 1 mode 2
mov scon, #0x50 ;config serial 8-data, 1 start, 1 stop, no parity
```

```

mov th1, #0xFD ;9600 baud
setb tr1 ;start timer 1 to enable serial communication
;end section from lab 9
;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
;In the following section load the key bytes from ROM into RAM
mov r0, #keyBytesRAMAddress ;initialize RAM pointer
mov dptr, #keyvals2 ;initialize ROM pointer
mov keyvalIndex, #0x00 ;initialize keyvalIndex
LoadKey:
push keyvalIndex ;preserve keyvalIndex variable
movc a,@a+dptr;load byte of key from ROM
cjne a, #0x00, notNull ;check for null terminating character
jmp LoadDone ;if null is found, enter main_loop
notNull:
mov @r0, a ;put byte of key into ram
pop keyvalIndex;restore keyvalIndex variable
inc keyvalIndex;increment keyvalIndex
inc r0;increment RAM pointer
jmp LoadKey;continue the loop
LoadDone:
mov @r0, #0x00 ;append null char to string
mov r0, #keyBytesRAMAddress;re-initialize RAM pointer
;----- END of Initialization/configuration -----;
;begin section from lab 9
;vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
mainloop:
jnb ri, $ ;wait to receive a char
call getchar ;char received, get it!
; cjne a, #0x00, encrypt ;check for null character in string
cjne a, #0x00, checkKeyVal ;check for null character in string
jmp terminate ;terminate program if null character is recieved
;end section from lab 9
;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
checkKeyVal:
cjne @r0, #0x00, Encrypt;go to Encrypt if keyVal is not null
mov r0, #keyBytesRAMAddress ;re-initialize RAM pointer
;begin section from lab 9
;vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Encrypt:
xrl a, @r0 ;encrypt the character contained in the accumulator
call writechar ;write the encrypted character
inc R0
jmp mainloop

```

```

terminate:
mov a, #0x00 ;load null character into accumulator
call writechar ;append the null character to text output
sjmp $ ;halt

```

```

;----- getchar -----;
;subroutine receives nothing before it is called
;writes the character to the serial console
;returns a byte in the accumulator
getchar:
mov a, sbuf ;get serial data (char)
clr ri ;acknowledge data received
ret ;return from subroutine call
;----- writechar -----;
;receives byte or character
;reads a character that has been received serially
;returns the c

```

```

writechar:
mov sbuf, a ;send data (char) serially
jnb ti, $ ;wait until data is sent
clr ti ;acknowledge data has been sent
ret ;return from subroutine call
;end section from lab 9
;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```

; BONUS

```

displayChar:
    mov chIndex, #0x00          ; put char in serial buffer
loopBack:
    push chIndex                ; preserve accumulator
    movc a, @a + DPTR           ; load byte of key from ROM
    jnz not0                    ; jump to not0 if accumulator != 0
    pop chIndex                 ; restore accumulator
    ret

```

```

writeCharBonus:
    mov sbuf, a                  ; send data (char) serially
    jnb ti, $                    ; wait until data is sent
    clr ti                       ; acknowledge data sent
    ret

```

```

not0:
    Call writeCharBonus          ; if accumulator is not nul call writeCharBonus
    pop chIndex                  ; restore accumulator

```

```
inc chIndex          ; increment accumulator
jmp loopBack         ; jump back to loopBack
```

;multibyte keys are defined below, only one will be used at a time

```
org 0x200
```

Prompt: db "Welcome to Andy and Greg's Lab 10 Bonus Encrytor. Please send a key value file:
",0

```
keyFile: db "Key value file has been received. Now please send plain.txt to be encrypted",0
```

```
keyvals: db '12345678',0
```

```
keyvals2: db 0x23, 0x34, 0x45, 0x56, 0x67, 0x78,0x89, 0x90, 0xCD, 0xAB, 0x00
```

DutyValues: DB 0x60, 0x70, 0x80, 0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0, 0xF0

End

[illegible]

C:\Users\Greg\Desktop\plain.txt - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins

change.log x plain.txt x New Text Document.txt x key.txt x CAPTURE.TXT

```
1 This is our plain text exampleNUL
```

C:\Users\Greg\Desktop\CAPTURE.TXT - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window

change.log x plain.txt x New Text Document.txt x key.txt x CAPTURE.TXT x

```
1 w\, $GDC1ú° cBQDC45 : ACKDC1ç°: î[ @e3USEMää; îNUL
```

Lab 10 Bonus Manual Verification

aintxt) This is our plain text example

Ascii	Hex	key value	XOR
T	→ 0x54	0x23	$\begin{array}{r} \text{XOR} \quad 0101 \ 0100 \\ \quad \quad 0010 \ 0011 \\ \hline 0111 \ 0111 \\ \text{OR } 77 \end{array}$

h	→ 0x68	0x34	$\begin{array}{r} \text{XOR} \quad 0110 \ 1000 \\ \quad \quad 0011 \ 0100 \\ \hline 0101 \ 1100 \\ \text{OR } 5C \end{array}$
---	--------	------	---

i	→ 0x69	0x45	$\begin{array}{r} \text{XOR} \quad 0110 \ 1001 \\ \quad \quad 0100 \ 0101 \\ \hline 0010 \ 1100 \\ \text{OR } 2C \end{array}$
---	--------	------	---

s	→ 0x73	0x56	$\begin{array}{r} 0111 \ 0011 \\ 0101 \ 0110 \\ \hline 0010 \ 0101 \\ \text{OR } 25 \end{array}$
---	--------	------	--

Cipher	Hex	→	Ascii
	0x77		w
	0x5C		\
	0x2C		,
	0x25		%

This matches our cipher!

